10

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DIVISION DE INGENIERIA ELECTRICA

SISTEMA DE DESPLIEGUE GRAFICO PARA UN ECOSONDA EMPLEANDO UNA COMPUTADORA PERSONAL

TESIS PROFESIONAL

ELABORADA PARA OBTENER EL TITULO DE

INGENIERO ELECTRICO ELECTRONICO

POR

SERGIO OCTAVIO ORTIZ MACEDO

MEXICO, D.F. SEPTIEMBRE 2001

Dedico este trabajo
a mis padres,
a mi hermana
y al Sr. Noguéz

Agradezco el apoyo
de Yukihiro,
de Victor López,
del doctor Luis Soto
y de la tripulación del JustoSierra

Sistema de despliegue gráfico para un ecosonda empleando una computadora personal

ÍNDICE

Sistema de despliegue gráfico para el ecosonda modelo 202C empleando una computadora personal

RESUMEN

El buque oceanográfico Justo Sierra es una embarcación que dentro de su equipo de medición cuenta con un aparato fundamental para la investigación submarina. Dicho aparato es el ecosonda, el cual, acompañado de una grabadora[1], revelan al investigador el fondo submarino.

La grabadora es un dispositivo que realiza fundamentalmente dos funciones; la primera es la generación de pulsos periódicos que permiten que el ecosonda envíe pulsos ultrasónicos al fondo oceánico; la segunda función es la de tomar los datos que recibe del ecosonda y desplegarlos en un rollo de papel termosensible, a través de una punta emisora de descargas voltaicas.

El despliegue de los datos en dicho papel termosensible presenta diversas problemáticas, como falta de nitidez, manchas no deseables y un elevado costo del papel termosensible. Los controles de la grabadora también presentan fallas, debido a la falta de mantenimiento del aparato.

Con base en lo anterior, se estableció como objetivo de este trabajo sustituir el despliegue gráfico que realiza actualmente la grabadora, por medio de una interfaz conectada a una computadora personal y un programa el cual realice las mismas funciones fundamentales y proporcione ventajas sobre el aparato actual.

Para ello, se desarrollaron una interfaz para la grabadora y la computadora personal y un programa de control en Java. La interfaz consta de una tarjeta de expansión de puerto paralelo que recibe el pulso que produce la grabadora, así como los datos que entrega el ecosonda, una vez que fueron digitalizados por la misma grabadora. El programa, por un lado, se encarga de controlar dicha interfaz, mientras que por otro, se encarga del despliegue gráfico en el monitor de la computadora. Este programa proporciona toda la funcionalidad necesaria de la grabadora original, además de proporcionar ciertas ventajas como almacenar en un medio magnético, fácil de transportar, las gráficas recabadas durante la campaña; si se desean tener en papel dichas gráficas, se utiliza una impresora convencional; también permite tener un control de resolución que amplía el área de interés.

Después de realizar las pruebas en el buque, se identificaron varias necesidades a desarrollar en el sistema actual, las cuales fueron satisfechas hasta obtener la última versión que se presenta en este trabajo.

Finalmente, se llegó a la conclusión de que es factible la sustitución del sistema de despliegue actual. Sin embargo, el sistema desarrollado aún tiene algunas carencias, que es deseable satisfacer con trabajos posteriores.

[1] Se adoptó el término grabadora, de la traducción directa de '*Recorder*' para el aparato encargado del despliegue gráfico en papel.

CAPÍTULO 1

INTRODUCCIÓN

Introducción al sistema de despliegue gráfico para el ecosonda

El ecosonda y la grabadora constituyen un par de instrumentos que, en conjunto, permiten al investigador analizar lo que se encuentra debajo del buque oceanográfico, esto es el fondo submarino y cualquier objeto en la masa de agua entre el fondo y el buque.

En esencia, el ecosonda permite identificar cualquier objeto dentro de la masa de agua, que sea capaz de reflejar parcialmente el pulso ultrasónico que emite. Para conseguir este fin, el ecosonda cuenta con un transductor ultrasónico que emite pulsos con determinada intensidad y frecuencia, e inmediatamente después de emitir el pulso, este mismo transductor comienza a captar el pulso parcialmente reflejado.

Una función de la grabadora es generar un pulso, con nivel de voltaje TTL. Este pulso es periódico; el periodo esta directamente relacionado con la profundidad que se desea explorar. Es decir, cuando el buque se encuentra a bajas profundidades el periodo es corto, mientras que cuando el buque se encuentra en aguas profundas el periodo es largo. Este pulso es el disparador del pulso ultrasónico del ecosonda. La grabadora cuenta con los controles necesarios para que el investigador modifique el periodo del pulso; el control más importante es el llamado "base de tiempo[2]", que es una perilla que puede ser colocada en los distintos valores de periodo posibles.

[2] Este control es llamado base de tiempo en la grabadora, por lo que se adopta de igual forma este término.

Por otro lado, la otra función de la grabadora es desplegar, en un papel termosensible, la señal que recibe el ecosonda del fondo marino. Para efectuar esta función la grabadora realiza lo siguiente: recibe la señal analógica recuperada por el ecosonda, y la convierte de analógica a digital en formato de cuatro bits, por lo que obtiene un rango de 16 valores, que en decimal van del cero al quince. Cada uno de estos valores es interpretado como un voltaje distinto que es amplificado y aplicado a la aguja. Esto permite imprimir una tonalidad distinta de gris en el papel al momento de ser quemado; dicha tonalidad va desde el blanco hasta el negro intenso.

Problemas con la grabadora actual

Como ya se mencionó, el sistema actual utiliza un rollo de papel térmico. Cuando comienza la campaña, es necesario embarcar un cierto número de estos rollos para utilizar durante los distintos transectos[3] que se realizarán. Esta cantidad de rollos es muy difícil de determinar, ya que los transectos se realizan, la mayoría de las veces, con el fin de detectar un área plana y libre de rocas del fondo submarino, para poder lanzar la red de arrastre y garantizar que ésta no se vaya a romper durante el arrastre; por lo tanto, durante la exploración de un transecto, se puede tener mala fortuna y encontrar una zona escarpada, por lo que se tendrá que buscar otra ubicación y esto puede ocurrir varias veces hasta encontrar una zona adecuada.

Otro problema muy importante, también relacionado con el papel, es evidentemente el gasto que implica la adquisición de estos rollos; el aparato es un modelo viejo, por consecuencia estos rollos no son utilizados en los aparatos nuevos y cada vez es más difícil conseguir dichos rollos.

Por otro lado, la grabadora actual debido a su constante uso, ya presenta ciertas imperfecciones tales como fallas en el control de contraste, lo que ocasiona que el papel se comience a quemar en exceso.

El sistema de sondeo actual tiene un problema, el cual consiste en una mala ubicación del transductor emisor del pulso ultrasónico. Éste se encuentra dentro del casco del barco, ubicación que no es del todo favorable. Al ser emitido el pulso ultrasónico, éste tiene impacto con la superficie metálica y es parcialmente reflejado. El transductor comienza a recibir dichos rebotes, entonces, lo primero que despliega la grabadora es una intensa franja negra en la primera parte del papel (normalmente una quinta parte de su ancho total). Estos datos obviamente no son útiles, y además, ocasionan que el papel se queme innecesariamente, provocando además un olor desagradable en el laboratorio de medición. Una solución del ingeniero de a bordo fue colocar una cinta de papel del largo de esta franja, de tal forma que la aguja no pueda quemar dicha superficie. Aún así, debido al resto del papel quemado, es constante el desagradable olor que ocasiona este aparato. Con la interfaz gráfica, es deseable que este problema también quede resuelto.

[3] Un transecto, es un recorrido a baja velocidad, a partir de un cierto punto, en una dirección preestablecida durante determinado tiempo o distancia, con el fin de estudiar la zona.

Expectativas de los usuarios del sistema

Las expectativas de los usuarios son las siguientes: se requiere que el sistema sea competitivo con el actual, es decir, deberá cubrir todas las bases de tiempo que maneja la grabadora y deberá presentar una gráfica del fondo submarino similar a la desplegada por el equipo actual. Por otro lado, es deseable que este diseño resuelva los problemas que muestra el aparato actual, como son las fallas en el contraste, el exceso de papel quemado en las primeras dos franjas, el mal olor debido al papel quemado, y correr el riesgo de quedarse sin papel térmico, entre otros.

Las ventajas que obtendrían los usuarios del sistema serían: eliminar el mantenimiento de las partes mecánicas, tener una interfaz con el mínimo de controles, y tener la facilidad de llevar las gráficas almacenadas en un medio magnético. Cada investigador podría tener una copia del programa en su lugar de trabajo para que, una vez desembarcados, se puedan reproducir las gráficas e imprimirlas fácilmente; se podría diseñar un procesamiento más complejo de las señales para obtener más información (como identificación de materiales en el fondo submarino).

Objetivos

De acuerdo a las necesidades y expectativas, se definieron los siguientes objetivos:

- Desplegar en el monitor de la computadora personal la señal proveniente del ecosonda, a una determinada base de tiempo, de una forma similar al sistema actual.
- Permitir el almacenamiento de las gráficas, tanto en papel impreso como en medio magnético (disco duro o flexible).
- Eliminar controles que no son utilizados por el investigador, con el fin de simplificar el uso.
- Agregar mejoras al sistema, de acuerdo con las experiencias obtenidas durante la campaña oceanográfica.
- Obtener un sistema competitivo con el actual.

Los equipos hidroacústicos

Para poder realizar este sistema, se hizo primeramente un estudio sobre los dispositivos utilizados. Estos son el ecosonda y la grabadora. En este trabajo se incluye un capítulo con las generalidades de estos equipos.

En dicho capítulo se incluye un estudio de los transductores utilizados por los ecosondas. Se mencionan sus partes fundamentales, su funcionamiento y los distintos tipos de transductores. También se incluyen explicaciones del perfil del haz sonoro y del patrón

de directividad. Posteriormente, se encuentra un subtema referente al criterio de selección de un transductor. Finalmente, se muestran las características del transductor actual utilizado en el ecosonda.

Por otro lado se describen las funciones que desempeña la grabadora; su modo de operación; los controles con los que cuenta; así como la utilidad y forma en que es utilizada en el buque. También se hace un análisis más detallado de las señales que intervienen en el funcionamiento de la grabadora y el tipo de datos que se manejan.

Análisis de requerimientos de circuitos electrónicos y programas de cómputo

Una vez que se ha efectuado la introducción al sistema actual y a las necesidades que deben ser cubiertas, es necesario llevar a cabo un estudio de los elementos necesarios para la solución de este problema. Este es el motivo de la sección, que se aborda en el tercer capítulo de este trabajo.

En esta parte se hace un estudio del funcionamiento de los circuitos que se utilizaron durante la búsqueda del sistema más adecuado. Se hace una descripción del funcionamiento del convertidor analógico digital ADC, utilizado en la primera aproximación de este sistema, así como un estudio del puerto paralelo convencional de la computadora, dado que éste fue utilizado durante la etapa de pruebas en el buque. Finalmente, se encuentra un estudio del puerto paralelo de expansión que resulta ser la mejor opción para la interfaz entre la computadora personal y la grabadora.
Por otro lado, se hace un estudio para determinar las condiciones del programa de control. Primeramente, se estudian los requerimientos de la computadora a utilizar, ya que como se trata de un sistema en tiempo real, se deben cubrir ciertas exigencias. Posteriormente, se estudian las características que debe tener el programa, para poder satisfacer las necesidades. Finalmente, se extiende este análisis para determinar cuál es el lenguaje de programación que resuelve los requerimientos de dicho programa.

La interfaz

El análisis de los circuitos electrónicos, mencionados en el párrafo anterior, es con el fin de tener los elementos necesarios para la creación de la interfaz. Esta interfaz constituye el medio de comunicación entre la grabadora y la computadora personal.

La señal importante de la grabadora es el pulso periódico que ésta genera. Otra señal de importancia es un pulso que le transmite un aparato denominado "corredera[4]". Finalmente, los datos a desplegar provienen del ecosonda. Esta última señal es entregada por la grabadora, en dos modalidades: una es tal como la recibe del ecosonda, mientras que la otra es en forma digital.

[4] La corredera es un aparato que emite un pulso una vez que el barco ha recorrido una cierta distancia. Por esta razón el intervalo de esta señal depende de la velocidad a la que se desplace el barco. Por lo tanto, en el despliegue gráfico estos pulsos aparecen como líneas con una separación proporcional a la velocidad del barco.

Por parte de la computadora hacia la grabadora no se tiene ninguna señal. Por lo tanto, la interfaz necesaria para la comunicación entre estos dos dispositivos debe ser capaz de recibir las señales provenientes de la grabadora y enviarlas a la computadora para su despliegue.

En primera instancia se optó por tomar la señal proveniente del ecosonda en su forma analógica, para lo que se diseñó una tarjeta de expansión de puerto paralelo con un convertidor analógico digital ADC0809, con los circuitos necesarios para el funcionamiento de este último, así como los necesarios para la comunicación con la computadora.

Debido a la naturaleza de la señal analógica, se tuvo que descartar la opción de la tarjeta de expansión con el convertidor analógico digital. Esto fue debido a que durante las pruebas, se observó que esta señal cambiaba su amplitud al cambiar de escala. Por lo tanto, con fines de realizar pruebas a escalas distintas, se construyó una interfaz utilizando el puerto paralelo LPT1 de la computadora, el cual únicamente contenía los circuitos necesarios para una comunicación adecuada con la misma, esta tarjeta tomaba la señal digitalizada de la grabadora. No obstante, debido a la necesidad de imprimir las gráficas en una impresora convencional, esta interfaz tuvo que ser cambiada, una vez terminadas las pruebas en el buque.

La interfaz final nuevamente utiliza la tarjeta de expansión de puerto paralelo, sólo que en esta ocasión ya no se toma la señal analógica del ecosonda, sino la señal digital que es entregada por la grabadora, la cual consta de cuatro bits. Por lo tanto, esta tarjeta tiene los circuitos necesarios para recibir las seis señales (los cuatro bits del dato, el pulso de la grabadora y el pulso de la corredera), con niveles de voltaje TTL, y los circuitos necesarios para la comunicación con la computadora.

El programa de control

Este programa requiere realizar dos funciones fundamentales: una interfaz gráfica para el despliegue de los datos y una sección encargada del control de la interfaz de la PC y la grabadora para la obtención de los datos. Para efectuar estas funciones se requiere lo siguiente:

- el programa será capaz de mantener un poleo[5] constante para la marca externa proveniente de la corredera, así como para el pulso periódico que emite la grabadora;
- se requiere una rutina de obtención de datos;
- es necesaria una rutina de despliegue de puntos;
- se necesita una interfaz gráfica con controles de ganancia, umbral, velocidad y cambio en la base de tiempo;
- finalmente, una parte encargada del almacenamiento en memoria de los datos y recuperación de gráficas anteriores.

[5] Se toma la palabra poleo, a partir del término en inglés *polling*, que consiste en tener un ciclo donde se consultan al inicio los estados de algunas variables con el fin de tomar decisiones y/o mantener sincronía.

La primera versión de este programa fue uno basado en lenguaje C, el cual desde sus inicios comenzó a mostrar mucha complejidad y a requerir de programación avanzada, por lo que se decidió abandonar esta idea.

Posteriormente se optó por realizar este programa en Java, ya que este lenguaje resolvía de una manera sencilla la mayor parte de las necesidades del sistema.

La opción más eficiente encontrada hasta el momento consiste en tres hilos de ejecución. El primer hilo, o hilo principal, se encarga de crear los otros dos hilos, el de conversión[6] y el de despliegue de datos; también, atiende los eventos generados por el usuario, de esta forma, cuando el usuario desencadena una acción, el hilo principal puede suspender los otros dos hilos, atender al usuario y reanudarlos. Si los dos hilos secundarios cuentan con la sincronía adecuada, una suspensión de ambos no tiene ningún problema.

Principalmente se utilizó Java, ya que este lenguaje es orientado a objetos, con un gran número de ventajas sobre el lenguaje C. Entre estas ventajas se pueden mencionar las siguientes:

- manejo de hilos de ejecución;
- manejo de eventos mediante el modelo de "Listeners" (escuchas);
- creación sencilla de interfaces gráficas;
- manejo sencillo de flujos de datos hacia archivos;
- compresión de estos últimos.

No obstante, la comunicación a bajo nivel con la tarjeta de expansión del puerto paralelo, se realizó con lenguaje C, debido a que en Java no fue posible la configuración de la dirección de puerto, y tampoco se encontraron clases que pudieran soportar el envío y la recepción de señales vía puerto paralelo.

[6] El nombre de conversión se tomó debido a la primer aproximación que requería llevar a cabo la conversión analógico-digital, aunque en las aproximaciones siguientes no se realiza ninguna conversión, el nombre de la clase fue respetado.

CAPÍTULO 2

GENERALIDADES DE LOS EQUIPOS HIDROACÚSTICOS

Introducción a los transductores hidroacústicos

Para el estudio del fondo submarino, el ser humano requiere de gran asistencia para la percepción y localización de sonidos bajo el agua. Actualmente la forma más común para detectar y generar sonido bajo el agua es a través de los dispositivos conocidos como transductores. Los equipos de generación y detección de ondas acústicas incluyen un número de dispositivos que varían en grado de complejidad, para hacer evidente a un observador la existencia de una onda sonora bajo el agua. El elemento que permite detectar una onda sonora convirtiendo ésta en una señal eléctrica se llama **hidrófono**, y al elemento que permite generar una onda sonora a partir de una señal eléctrica se le conoce como **proyector**. La habilidad de convertir estas dos formas de energía recae en propiedades muy específicas de ciertos materiales, llamadas **piezoelasticidad** (electroestricción) y **piezomagnetismo** (magnetoestricción).

Los transductores utilizados en la generación y recepción de ondas acústicas en líquidos y sólidos se pueden clasificar de la siguiente forma.

a) Transductores ultrasónicos utilizados en trabajo experimental de investigación, en la propagación de ondas acústicas de alta frecuencia en líquidos y en la medición de las propiedades elásticas de los sólidos.

b) Transductores ultrasónicos usados en la industria, como potentes microagitadores de líquidos para desengrasar, emulsificar y coagular.

c) Transductores ultrasónicos utilizados para localizar fallas en metales forjados y plásticos laminados.

d) Transductores para aplicaciones marinas como la detección de objetos sumergidos, detección del fondo, cardúmenes y aplicaciones militares.

Debido a la alta impedancia acústica específica de los líquidos, comparada con la del aire, los elementos vibratorios de los transductores diseñados para utilizarse en líquido, deben ser acoplados acústicamente para producir grandes fuerzas a través de pequeños desplazamientos, lo cual es posible si se colocan capas de materiales con impedancias acústicas intermedias entre el elemento vibratorio y el líquido.

Los vibradores piezoeléctricos más comúnmente utilizados, incluyen en su construcción materiales naturales tales como: cristales de cuarzo y turmalina, y materiales artificiales como la sal de Rochelle, el amonio dihidrógeno fosfato y el sulfato de litio.

Otros elementos vibratorios incluyen materiales cerámicos como el titanato de bario, y dieléctricos con una fuerte propiedad ferroeléctrica, o con efectos fuertes de propiedades magnetoestrictivas como son las aleaciones de níquel.

La existencia de la propiedad piezoeléctrica en un material depende de la disposición interna de sus átomos en el cristal, los cuales al ser sometidos a una tensión experimentan distorsiones en las que las cargas positivas y negativas tratan de alinearse así mismas a causa de la asimetría producida en el cristal, de manera que aparece un momento dipolar en éste.

Dos materiales piezoeléctricos que se emplean normalmente son el titanato de bario y el niobato de plomo, los cuales son ferroeléctricos además de piezoeléctricos; esto significa que están polarizados espontáneamente, y una deformación mecánica en éllos provoca un cambio en esta polarización. Ni el titanato de bario ni el niobato de plomo pueden obtenerse como cristales puros de gran tamaño, por lo que el material transductor se prepara como una cerámica a partir de la mezcla de óxidos y la posterior sintetización.

Los transductores piezoeléctricos se utilizan generalmente para altas frecuencias, arriba de los 50 kHz y tienen una alta eficiencia, hasta de un 75% o más.

La magnetoestricción tiene lugar en los materiales ferromagnéticos, los cuales dada su construcción puramente metálica se contraen y dilatan según la fuerza y dirección del campo magnéticos resultante del impulso eléctrico aplicado. Los transductores de este tipo son fuertes y sencillos, con una eficiencia del 20 al 30%, siempre y cuando se trabaje el material en el punto adecuado de su curva de magnetización[7].

Los transductores magnetoestrictivos se utilizan en aplicaciones de baja frecuencia, hasta los 50 kHz.

[7] Ver el apéndice de Curva de magnetización

Perfil del haz sonoro de los transductores

Los transductores pueden ser diseñados y fabricados para producir haces sonoros de diferentes perfiles, concentrando generalmente la energía a lo largo del eje ortogonal a la superficie de radiación. Un factor determinante en esto es la dimensión de la superficie de radiación, medida en longitudes de onda a la frecuencia de resonancia; así, una superficie angosta produce un haz angosto y una superficie amplia un haz amplio.

La geometría de la superficie vibratoria también influye en la forma y perfil del haz sonoro, teniéndose que un transductor circular produce un haz cónico y un transductor rectangular un haz igualmente rectangular.

La concentración de la energía sonora a lo largo del eje ortogonal a la superficie de radicación permite reconocer otro parámetro importante conocido como **ancho del haz**, el cual indica el grado de directividad que tiene el transductor, que es la habilidad de éste para concentrar la energía o potencia acústica.

El ancho del haz se define como el ángulo entre los puntos en los cuales, la intensidad de la energía acústica decae a la mitad a lo largo del eje central del haz, que expresado en decibeles equivale a los puntos a –3 dB.

En el caso de un transductor circular, el ancho del haz (β), en radianes, está dado por la expresión:

$$\beta=65\lambda/d$$

donde λ es la longitud de onda de la frecuencia de transmisión o recepción, y *d* el diámetro de la superficie de radiación (ambos en las mismas unidades).

Patrón de directividad

El patrón de directividad es una gráfica que representa la distribución de la sensibilidad del transductor a lo largo de su eje normal; estos patrones son obtenidos en laboratorio y se proporcionan al usuario al momento de su compra, ver figura 1.

La sensibilidad decrece rápidamente fuera del eje normal para transductores de haz angosto y más gradualmente para transductores de haz amplio.

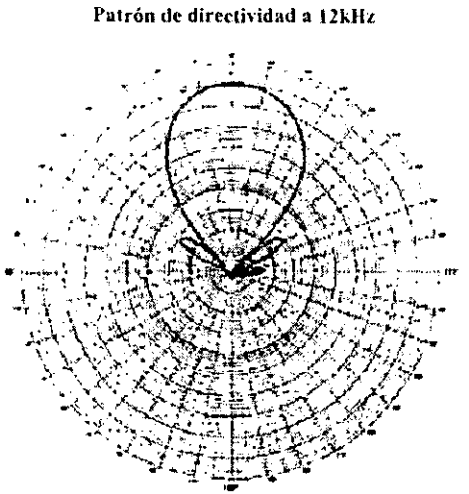

Fig. 1[8]. Muestra de un Patrón de directividad de un transductor

Debido a ciertas dificultades inherentes al diseño y construcción de los transductores, estas gráficas presentan anormalidades que se reflejan como incrementos de sensibilidad, formando los llamados lóbulos laterales. Estas anormalidades dificultan la interpretación de la señal acústica al incrementar la señal de ruido.

Criterio de selección de transductores

En el campo de la sónica, definiendo a ésta como la tecnología del sonido aplicada a problemas de medición, control y procesamiento, se enmarcan una gran variedad de aplicaciones que involucran niveles de potencia de algunas decenas de watts, como es el caso de la terapia médica con ultrasonido, a niveles de potencia de cientos o miles de watts, para el caso de proceso de líquidos.

En todos los casos, el criterio principal de selección se basa en la frecuencia a la cual el proceso a realizarse es óptimo; una vez fijada la frecuencia o rango de frecuencias, el siguiente paso es la selección del tipo de transductor a utilizar, con base en la eficiencia y costo del equipo, para finalmente afinar el criterio de selección con consideraciones de tipo mecánico de ingeniería.

Las frecuencias óptimas en la mayoría de las aplicaciones caen en el rango de 1 kHz a 20 kHz, para gases y entre los 10 kHz a 400 kHz para líquidos. En sólidos se utilizan herramientas vibratorias de acción local, y no se cuenta con una regla general para la determinación de la frecuencia óptima. En herramientas de corte y perforado se utilizan frecuencias de 20 Hz a 50 kHz, las frecuencias más altas, del orden de los Megahertz, son útiles en las aplicaciones donde se requiere alta direccionalidad, para afectar pequeñas áreas

[8] Ver la referencia número 10.

en líquidos o materiales viscosos, o donde la generación de calor debido a las altas frecuencias juega un papel importante, como es el caso de la terapia ultrasónica.

En procesos en fase gaseosa los transductores de fluido dinámico son los más recomendables, ya que son capaces de producir, a bajas frecuencias, fuertes desplazamientos (en amplitud) requeridos para proporcionar alta potencia a cargas de baja impedancia. Otra ventaja es el hecho de no requerir equipo electrónico, el cual en algunos casos es costoso y necesita de un buen mantenimiento.

En procesos de fase líquida, los transductores electromecánicos compiten con los de fluído dinámico ya que ambos son capaces de producir **cavitación**, que es un fenómeno importante en aplicaciones tales como: limpieza, mezclado, agitación, etc.

Para el caso específico de las aplicaciones marinas en la detección de objetos, fondo, cardúmenes y defensa, los transductores del tipo piezoeléctrico o magnetoestrictivo son los más utilizados, ya que se pueden obtener vibradores dentro de un rango amplio de frecuencias y con capacidad de producir fuertes presiones con desplazamientos relativamente pequeños.

Existen en el mercado diversos fabricantes de este tipo de transductores, quienes establecen como regla básica para especificar un transductor los siguientes puntos.

- **Rango de frecuencia**. Este parámetro influye mucho en las dimensiones del transductor ya que a mayor frecuencia, las dimensiones del transductor se reducen y aumentan cuando la frecuencia es más baja.
- **Nivel de presión acústica**. Este parámetro relaciona la potencia acústica, la directividad y la eficiencia del transductor.
- **Patrón de radiación**. Permite reconocer la directividad y sensibilidad del transductor para obtener mejores resultados de acuerdo a su aplicación.
- **Profundidad de operación**. Bajo este parámetro se especifican las condiciones de presión a las que será sometido el transductor, para evitar alterar su desempeño y/o provocarle daño físico.
- **Restricciones dimensionales**. Especifica el peso, tamaño, tipo de montaje y hasta costo del transductor.

El transductor EDO modelo 202C

Este transductor es un modelo de gran energía y haz estrecho, especialmente diseñado para operaciones oceanográficas de alta resolución. La porción activa de este transductor está constituida por 376 elementos piezoeléctricos preestresados, que constituyen un transductor de gran ancho de banda. Puede ser acoplado a cualquier impedancia requerida utilizando el elemento de acoplamiento.

Características:

Material Activo	Circonato titanato de plomo
Ancho de banda	12 kHz a 35 kHz
Patrón del haz	Cónico
Máxima potencia de entrada	10 kW
Máxima profundidad	500 ft

Introducción a la grabadora

La grabadora fundamentalmente se encarga de realizar dos acciones muy importantes con el fin de rastrear el fondo submarino.

La primer función fundamental de la grabadora es la generación de un pulso periódico, el cual es enviado al ecosonda, donde dicho pulso es el disparador del pulso ultrasónico, que es enviado por el transductor del ecosonda hacia el fondo marino.

La grabadora también se encarga de desplegar en un rollo de papel térmico, a través de voltajes que queman dicho papel, los valores capturados por el transductor del ecosonda, que a su vez son enviados a la grabadora, una vez que el pulso ultrasónico regresa después de ser parcialmente reflejado por el fondo submarino.

La actual grabadora recorre verticalmente 19.2 pulgadas por cada barrido, luego regresa a su posición cero (esquina superior izquierda), avanza el papel una pequeña distancia, y comienza nuevamente su barrido, ver figura 2.

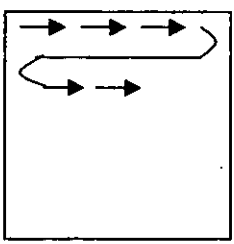

Fig. 2. Simulación de la dirección de barrido de l agrabadora

En un barrido se toman 4096 datos del ecosonda, es decir, cada 0.0047 pulgadas se imprime un dato, que según la intensidad (valor desde 0000_2 hasta 1111_2) coloca una de 16 tonalidades de gris, desde blanco (0000_2) hasta negro (1111_2). Coloca además 10 líneas de enrejado[9] (horizontal) igualmente espaciadas a lo largo del papel, para referencia de las distintas profundidades, ver figura 3.

[9] Este término de enrejado se toma a partir de una traducción de la palabra '*grid*' manejada en el manual de la grabadora.

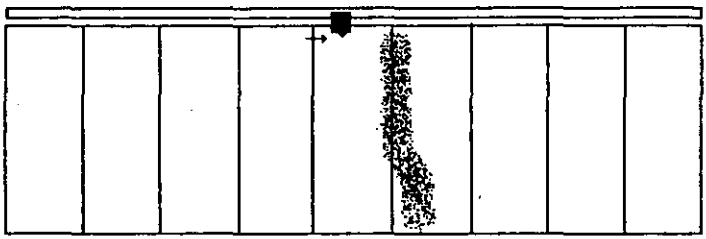

Fig. 3. Simulación de un fragmento de la grabadora actual

Controles de la grabadora

Controles de tiempo

La grabadora proporciona una señal periódica que es utilizada tanto por el ecosonda, como por sí misma, para iniciar un nuevo barrido. No obstante, esta señal puede ser proporcionada por otro medio, por lo que se encuentra en la grabadora una entrada que permite tomar esta señal. Cuando el pulso periódico es emitido por la grabadora, entonces se tiene una perilla que permite seleccionar una de las diferentes velocidades de barrido, que son: 8 s, 4 s, 2 s, 1 s, ½ s, ¼ s, 1/8 s, 1/16 s, 1/32 s por barrido.

Controles de la señal de entrada

La señal de entrada puede ser modificada por medio de varias perillas con el fin de obtener una mejor gráfica. Por otro lado, la grabadora cuenta con la opción de tomar una o dos señales distintas.

Controles de la carta[10]

Entre los controles de la carta se encuentran:

- el de velocidad de avance, el cual consta de 5 velocidades distintas;
- también se puede controlar la dirección del barrido (la forma en que se despliegan las dos señales, en caso de estarse utilizando);
- se cuenta con un interruptor de avance rápido de la carta, para dejar un espacio en blanco;

[10] Se denomina carta al papel o rollo que utiliza la grabadora.

- se cuenta con un control para colocar una marca vertical, en relación al fondo marino, para indicar algún evento, ver figura 4.

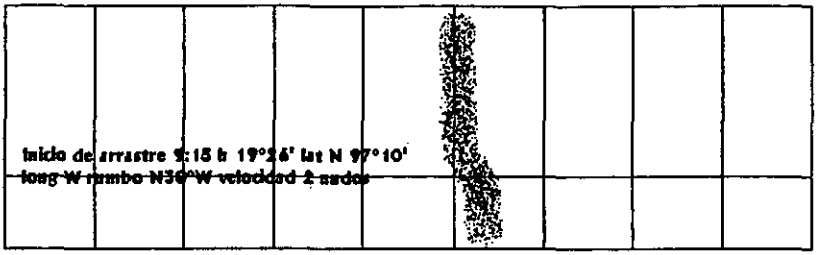

Fig. 4.Simulación de la marca de evento

Introducción al uso del sistema actual

Este sistema es utilizado principalmente cuando se desarrollan los transectos, es decir, cuando el barco toma una posición, en latitud y longitud deseadas, y comienza su recorrido en cierta dirección, hasta llegar a una nueva posición. Durante el recorrido, el investigador hace uso del ecosonda y la grabadora para observar la profundidad del fondo marino así como la cantidad y magnitud de discontinuidades y protuberancias que se encuentran en esa zona, esto con el fin de valorar si se puede o no sumergir la red de arrastre y poder capturar especies submarinas para su estudio. Cuando comienza el transecto, se registran datos como la ubicación exacta, fecha y hora, así mismo cuando se finaliza el transecto. Durante el recorrido se imprimen marcas periódicas sobre el papel de la grabadora, o bien, para marcar alguna zona de importancia. Normalmente una vez que termina el recorrido, el ecosonda se detiene, así mismo la grabadora. Es en este momento que los investigadores comienzan a tomar diversas muestras y mediciones. Durante este tiempo se pueden realizar maniobras con la grabadora, de tal forma que en la interfaz gráfica se podrían llevar a cabo respaldos e impresiones. Finalmente, los pliegos de papel se marcan y almacenan como bitácora.

Análisis de las señales de entrada

La señal proveniente del ecosonda puede ser tomada de distintas formas: la primera es directamente de dicho instrumento, y la segunda es tomar esta señal analógica desde el conector correspondiente de la grabadora, y finalmente, se puede tomar esta señal de una forma digital (en formato de 4 bits) desde el conector DIN de la grabadora, la cual se encarga de dicha conversión. Además de esta señal proveniente del ecosonda, la grabadora genera la base de tiempo, fundamental para el funcionamiento del ecosonda como también para la sincronización de ambos aparatos. Otra señal que es necesario obtener, es la de la corredera. Por lo tanto, se requiere tener una interfaz capaz de recibir las señales

provenientes de la grabadora y enviar esta información al programa encargado de manipularla y desplegarla en pantalla.

La primera forma de tomar la señal analógica es poco práctica, ya que la conexión entre la grabadora y el ecosonda es por detrás del panel de control, y a fin de cuentas dicha señal se tiene que traer al frente del panel. Esta opción tendría que ser realizada cuando se desee sustituir completamente la grabadora por otro sistema.

Para la segunda forma, tomar la señal analógica de la grabadora, es necesario convertirla con un ADC; además se necesitaría otro conector para la base de tiempo que proviene de la grabadora; esta señal se toma de los conectores GATE OUT; la duración del pulso es de 1 ms y su amplitud es de 5 V.

Finalmente, la tercer opción, que es la que se adoptó, es tomar la señal digitalizada de la grabadora, la cual es entregada por el conector DIN que presenta en su parte frontal; esta señal se encuentra en un formato de 4 bits. Se pueden tomar también de dicho conector DIN, la base de tiempo, la señal de la corredera y la referencia (tierra) de la grabadora.

Análisis de los datos.

En el sistema actual, como se mencionó, se despliegan puntos de diferente intensidad de gris en el papel de la grabadora, con base en una punta de alto voltaje que se encarga de quemar dicho papel. El número total de puntos es de 4096 por barrido, que se despliegan en 19.2 pulgadas de papel. Tal resolución es prácticamente imposible de obtener con un monitor convencional. Otra de las problemáticas encontradas en relación con la gran cantidad de puntos, es la del guardado de los datos, ya que si normalmente un dato de tipo byte, short o int ocupa uno o más bytes, esta cantidad al ser multiplicada por el número de datos por renglón y posteriormente por el número de renglones en la gráfica, da como resultado un archivo sumamente grande; este tipo de archivos se deben generar periódicamente, según la base de tiempo a la que esté trabajando el ecosonda, lo que puede originar una saturación del disco duro.

Dadas estas limitaciones, se tuvo que hacer un estudio de la factibilidad de reducir seriamente el número de puntos desplegados. De acuerdo con la experiencia de personas que habían visto en operación la grabadora, cosa que fue comprobada durante la etapa de pruebas en el barco, se supo que en realidad de estos 4096 puntos muchos son redundantes, es decir, algunos puntos están demasiado cerca, y lo único que permiten es comprobar o hacer más evidente el fondo submarino, por lo que se determinó que se podrían tomar por ejemplo uno de cada diez puntos para hacer un total de aproximadamente 400 puntos por barrido, o incluso una cantidad mucho menor (aproximadamente 150 puntos por barrido, durante algunas pruebas en el barco) y seguir teniendo una idea clara del fondo submarino. Además de esta reducción de puntos sustancial, si consideramos el problema antes mencionado de las dos franjas no utilizables, se pueden dejar de obtener puntos durante este intervalo, lo que permite aún poder tomar un menor número de puntos del aparato.

Por otro lado, en cuanto al avance horizontal en la interfaz gráfica, si se avanza un pixel cuando la velocidad esté en 1, aún con esto, el despliegue en la interfaz se nota más extendido que el despliegue en la grabadora. Por esta razón también es factible dejar de desplegar durante un barrido, o incluso más, de tal forma que se vea mucho más compacta la gráfica, incluso más compacta que en la grabadora original, sin perder detalle del fondo

submarino, ya que éste normalmente presenta cambios que son perceptibles hasta un número considerable de barridos. Esta consideración también presenta una mejora en el rendimiento del sistema, ya que al requerir menos datos, el procesador se encuentra menos saturado durante más tiempo. Esta opción no se emplea en el sistema actual, sin embargo se sugiere ampliamente para futuras versiones.

También se pensó en la posibilidad de utilizar un método de ahorro. Si en la pantalla, por ejemplo, se pueden poner 100 datos, dicho datos también estarán acompañados de su posición en la que serán colocados. Únicamente se almacenan los datos cuyo valor sea mayor que cero. Entonces cuando se recupera el dato se obtiene primeramente su valor y posteriormente su ubicación. Este método es eficiente cuando la cantidad de datos en cero sea suficiente para compensar el hecho de agregar un par de valores, que son los de las coordenadas, es decir, este método es útil cuando a lo sumo se despliega la tercera parte de la pantalla con datos y el resto son valores en cero. De acuerdo a las experiencias con las gráficas en papel, la cantidad de valores en la gráfica puede ser muy variable y esto fundamentalmente debido a la profundidad, ya que para profundidades grandes la cantidad de valores distintos de cero es muy pequeña, dado que el fondo marino es únicamente una pequeña franja; sin embargo, para profundidades relativamente pequeñas, casi todo el papel es ocupado con detalles del fondo y de capas de distintos sustratos debajo del mismo, por lo que aquí dicho método sería más costoso que benéfico.

Otra posibilidad estudiada, y que de hecho fue usada, fue la de la compresión de archivos, cosa que parecía muy difícil hacer desde el código en C, pero con el código en Java se pudo aplicar de una manera muy sencilla, lo que permitió reducir en gran medida el problema del espacio en disco.

Finalmente, otra posibilidad de reducir el espacio en memoria de dichas gráficas, es mandarlas a imprimir desde la computadora, de tal forma que los archivos puedan ser borrados con la garantía de tener un documento impreso.

CAPÍTULO 3

ANÁLISIS DE REQUERIMIENTOS DE CIRCUITOS ELECTRÓNICOS Y PROGRAMAS DE CÓMPUTO

Descripción general del ADC0809

El ADC0809 es un componente de adquisición de datos, de encapsulado CMOS monolítico, con un convertidor analógico-digital (A/D) de 8 bits, un multiplexor de ocho canales y lógica de control compatible con microprocesador. El convertidor A/D de 8 bits utiliza aproximaciones sucesivas como técnica de conversión[11]. El convertidor utiliza un comparador de alta impedancia estabilizado con un chopper[12], un divisor de voltaje 256R[13] con un árbol de interruptores analógicos, y un registro de aproximaciones sucesivas. El multiplexor de 8 canales permite acceder a cualquiera de las ocho señales analógicas en sus entradas, ver figura 5.

Este dispositivo elimina la necesidad de ajuste externo a cero o ajuste externo de sus valores mínimo y máximo. Se tiene una interfaz sencilla con el microprocesador debido a

[11] Para mayor información de esta técnica ver el apéndice sobre técnicas de conversión.
[12] Un circuito *chopper* es aquél que alterna el suministro de voltaje de +V a 0 V, de tal forma que la corriente promedio es menor a la corriente suministrada con un voltaje constante.
[13] Este arreglo consiste en 256 resistencias de un mismo valor colocadas en serie: la mitad de este valor al inicio, después 255 resistencias y finalmente la mitad del valor de la resistencia.

sus entradas para la dirección, constituidas por compuertas tipo latch[14] y un decodificador multiplexado y sus salidas con latch de triple estado.

El diseño del ADC0809 ha sido optimizado incorporando las ventajas de diversas técnicas de conversión A/D. Ofrece alta velocidad, alta precisión, mínima dependencia de la temperatura, precisión a largo plazo y repetitividad, y consume una potencia mínima.

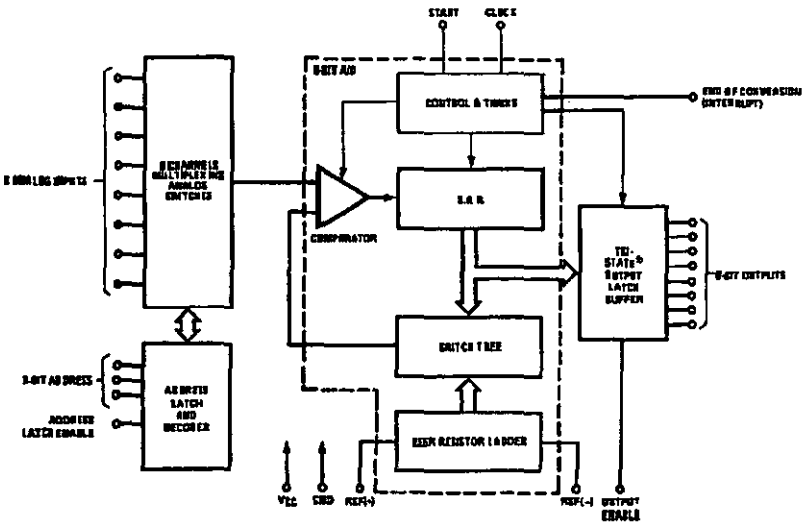

Fig. 5[15]. Diagrama de bloques del convertidor ADC0809

Características

- Interfaz sencilla con microprocesadores
- No se requieren ajustes a cero y ajuste de sus valores mínimo y máximo.
- Multiplexor de 8 canales con dirección lógica
- Rango de entrada de 0V a 5V con suministro único de 5V
- Salidas con niveles de voltaje TTL

[14] Una compuerta *latch* permite reflejar en su salida, el nivel de la entrada cuando su entrada de reloj se encuentra en estado alto, pero cuando éste cae, el último estado en la entrada de la compuerta es retenido y manifestado a la salida de la compuerta.

[15] Ver la referencia número 26.

Especificaciones básicas

Resolución	8 Bits
Error total	±½ LSB[16] y ±1 LSB
Suministro único	5 V_{DC}
Potencia mínima	15 mW
Tiempo de conversión	100 µs

Funcionamiento del puerto paralelo

Puerto original (SPP, puerto paralelo estándar)

El puerto paralelo original de las PC, tenía ocho salidas, cinco entradas, y cuatro líneas bidireccionales, como se muestra en la figura 6. En muchas PC actuales, las ocho salidas también pueden servir como entradas, para comunicaciones más rápidas.

- 8 salidas a través del puerto de datos (D0 a D7)
- 5 entradas (una invertida) a través del puerto de estado (S3 a s7)
- 4 salidas (tres invertidas) a través del puerto de control (C0 a C3)
- las 8 líneas restantes están aterrizadas

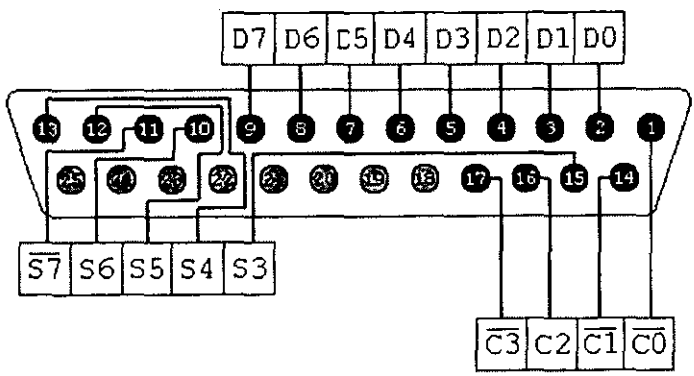

Fig. 6[17]. Conector hembra de 25 vías, DB25

Existen varias versiones mejoradas de la especificación original que han sido introducidas a lo largo de los años:

- "Bi-direccional" (PS/2)

[16] LSB es el acrónimo en inglés de *Less Significant Bit*, por lo que si el bit menos significativo representa el valor de 1, entonces un error de ½ LSB implica un error de 0.5

- "Enhanced Parallel Port", puerto paralelo mejorado (EPP)
- "Extended Capability Port", puerto de capacidad extendida (ECP)

Direcciones

Cada puerto paralelo utiliza un rango de direcciones. Muchos puertos tienen asignado un nivel de solicitud de interrupción, o IRQ, por sus siglas en inglés, y en los puertos ECP puede tener asignado un canal de DMA, o acceso directo a memoria. Estos recursos no deberán entrar en conflicto con otros componentes del sistema, incluyendo otros puertos paralelos. El puerto paralelo estándar usa tres direcciones contiguas, usualmente en alguno de los siguientes rangos:

3BCh	3BDh	3BEh
378h	379h	37Ah

La primera dirección es la dirección base del puerto, y permite tener acceso al Registro de Datos. La segunda dirección al Registro de Estado y la tercera al Registro de Control.

La mayoría de sistemas DOS y Windows se refieren al primer puerto en orden numérico como LPT1 (Line Printer Terminal 1, o puerto para impresora de línea), el segundo LPT2, y el tercero LPT3.

Interrupciones

La mayoría de puertos paralelos son capaces de detectar señales de interrupción provenientes de un periférico. El periférico puede usar una interrupción para anunciar que está listo para recibir un byte, o bien, que tiene un byte listo para enviar. Para usar interrupciones, el puerto paralelo debe tener asignado un nivel de solicitud de interrupción (IRQ).

Convencionalmente, el LPT1 utiliza IRQ7 y el LPT2 utiliza IRQ5. Pero IRQ5 es usada a menudo por muchas tarjetas de sonido, debido a que los niveles IRQ son escasos en el sistema, inclusive IRQ7 podría estar reservada por otro dispositivo. Algunos puertos permiten escoger otros niveles IRQ en lugar de los anteriores.

Canales DMA

Los puertos ECP pueden usar acceso directo a memoria (DMA) para transferencias de datos en el puerto paralelo. Durante las transferencias DMA, el CPU está libre de hacer otras cosas, por lo tanto las transferencias DMA pueden influir en el rendimiento total. Para utilizar DMA, el puerto debe tener asignado un canal DMA, el canal 1 o el 3.

[17] Ver la referencia número 22

Encontrando puertos existentes

Los sistemas DOS y Windows incluyen utilidades para encontrar los puertos existentes y examinar otros recursos del sistema. En Windows 95, siguiendo la secuencia siguiente, Panel de Control, Sistema, Dispositivos, Puertos, se puede elegir un puerto para ver su dirección asignada y, opcionalmente, su valor de IRQ y canal de DMA.

Configuración

Al puerto paralelo incluido en una PC se le pueden modificar sus valores predeterminados, con el fin de acoplarlo a alguna aplicación. Si se agrega un nuevo puerto a la PC se necesita configurarlo, asegurándose que no tenga conflicto con los puertos existentes o con otros recursos del sistema.

Hardware del puerto

El hardware incluido en un puerto paralelo, es un conector en el panel trasero de la PC, y los circuitos y conexiones entre dicho conector y los *buses*[18] de expansión del sistema. El microprocesador de la PC usa los buses de datos, de direcciones y de control para transmitir información entre el puerto paralelo y el CPU, memoria u otros componentes del sistema.

Conectores

El panel trasero de la PC tiene un conector para conectar un cable a la impresora u otro dispositivo con una interfaz para puerto paralelo. La mayoría de los puertos paralelos usan un conector 25 contactos D sub, conocido como conector subminiatura D, DB25, D-shell o sólo conector D. El estándar IEEE 1284 nombra a este conector como IEEE 1284-A. Algunos puertos paralelos nuevos pueden usar el nuevo IEEE 1284-C de 36 contactos.
El conector en la computadora es hembra, donde las terminales individuales son *sockets* o receptáculos.

Los circuitos contenidos

Los circuitos del puerto se conectan a las líneas de control, datos y dirección de la ranura de expansión, y a través de ellas con el microprocesador y otros componentes del sistema. El conector del puerto paralelo está especialmente diseñado para conectar impresoras con una interfaz de puerto paralelo, pero también puede ser usado como un puerto de entrada/salida genérico para cualquier dispositivo que tenga las capacidades de entrada/salida antes mencionadas. Tiene un buffer TTL de 12 bits de salida, los cuales son retenidos (*latched*) y

[18] Un bus es un conjunto de conductores que conecta a todos los circuitos de la computadora.

pueden ser escritos y leídos bajo el control de un programa usando las instrucciones 'In' o 'Out' propias del procesador. El conector también tiene cinco bits de entrada que pueden ser leídos usando la instrucción 'In' del procesador.

Uso múltiple de un puerto

Si se tiene más de un periférico para conectarse a un puerto paralelo, la solución más simple es agregar un puerto por cada uno. Pero hay ocasiones en que esto no es una opción. En este caso, las alternativas son alternar los cables cuando sea necesario o utilizar una caja de conmutación.
Si se utiliza sólo un dispositivo a la vez y se alternan ocasionalmente, la solución es tan sencilla como mover el cable cuando se desee cambiar el dispositivo. Para un cambio constante, una solución más conveniente es la caja de conmutación. Una caja típica tiene tres conectores hembra tipo DB25 y un interruptor que permite alternar entre un conector u otro.

Tarjetas personalizadas de entrada/salida

Muchos otros tipos de circuitos de entrada y salida están disponibles en tarjetas específicas de expansión. Una ventaja es que no existe limitación de emplear una interfaz existente. La tarjeta puede contener cualquier combinación de entradas y/o salidas digitales y analógicas. Además, esta tarjeta puede contener circuitos temporizadores, generadores de funciones, manejadores, filtros o cualquier otro tipo de componente. Con un puerto paralelo estándar, se pueden conectar estos componentes externamente, pero con una tarjeta personalizada, estos se pueden colocar dentro de la computadora.

Para usar una tarjeta de expansión se necesita, desde luego, una ranura de expansión libre, y este hardware personalizado requerirá igualmente de software personalizado.

Bus ISA

El bus ISA, *Industry Standard Architecture,* tiene su origen a principios de 1980 en un laboratorio de IBM en Boca Ratón, Florida. La computadora personal original de IBM introducida en 1981 incluía un subconjunto de 8 bits del bus ISA. En 1984, IBM introdujo la PC-AT, para la cual se realizó el primer desarrollo de una arquitectura de bus ISA de 16 bits.

Descripción de las señales ISA

- **SA19** a **SA0** *(System Adress)*. Son los bits de direcciones del sistema, y son utilizados para acceder a las direcciones de memoria y de dispositivos de entrada y salida, I/O, del sistema. Estas señales pueden ser utilizadas junto con el rango de señales LA23 a LA17 para asignar direcciones a más de 16 megabytes de memoria. Sólo los primeros 16 bits

son usados durante operaciones I/O para asignar más de 64 kilobytes de localidades. SA19 es el bit más significativo; SA0 es el bit menos significativo. Estas señales son activadas en el bus del sistema cuando la señal BALE está alta y son retenidas al momento de que ésta cae; permanecen a lo largo de algún comando de lectura o escritura. Estas señales son normalmente manejadas por el microprocesador o por el controlador de DMA, *Direct Memory Access*, pero también pueden ser manejadas por un bus maestro en una tarjeta ISA que tome el control del bus.

- **LA23** a **LA17** (*Unlatched Address)*. Estos bits son usados para acceder a direcciones de memoria dentro del sistema. Son utilizados junto con el rango de señales SA19 a SA0 para acceder a hasta 16 megabytes de memoria. Estas señales son válidas cuando BALE esta en alto; no son retenidas y tampoco permanecen estables o válidas durante el ciclo completo del bus. La decodificación de estas señales deberá ser hecha cuando la señal BALE tiene su transición de estado alto a bajo.

- **AEN** (*Address Enable)*. Es utilizada para desconectar el microprocesador y otros dispositivos durante las transferencias de DMA. Cuando esta señal es activa el controlador de DMA tiene control de las direcciones, los datos y de las señales de lectura y escritura. Esta señal debe ser incluida en la decodificación de las tarjetas ISA para prevenir que se seleccionen tarjetas incorrectas durante los ciclos de DMA.

- **BALE** (*Buffered Address Latch Enable)*. Es utilizada para retener las señales LA23 a LA17 o para decodificarlas. Las direcciones son retenidas en la transición de estado alto a bajo de la señal BALE. Ésta es forzada durante los ciclos de DMA. Cuando se usa en conjunción con AEN, indica una dirección válida del microprocesador o de DMA.

- **CLK** (*Clock)*. Es un reloj corriendo libremente, por lo regular en el rango de los 8 MHz a 10 MHz, aunque su frecuencia exacta no está garantizada. Es usado en algunas aplicaciones con tarjetas ISA para permitir la sincronización con el microprocesador del sistema.

- **SD15** a **SD0** (*System Data)*. Funcionan como los bits del bus de datos ISA. SD15 es el bit más significativo; SD0 es el bit menos significativo. De SD7 a SD0 son utilizados para la transferencia de datos con dispositivos de 8 bits. De SD15 a SD0 son usados para la transferencia de datos con dispositivos de 16 bits.

- **-DACK0** a **–DACK3** y **–DACK5** a **–DACK7** (*DMA Acknowledge) 0 a 3 y 5 a 7*. Son utilizados para reconocer las solicitudes de DMA de las señales DRQ0 a DRQ3 y DRQ5 a DRQ7 que son explicadas a continuación.

- **DRQ0** a **DRQ3** y **DRQ5** a **DRQ7** (*DMA Request)*. Son utilizados por las tarjetas ISA para solicitar el servicio de un controlador de DMA o para solicitar el control del bus a su dispositivo controlador de éste. Estas señales pueden ser manejadas asíncronamente. El dispositivo que solicita, debe sostener la señal de solicitud activa hasta que la tarjeta del sistema ponga en estado alto la señal DACK correspondiente.

- **-I/O CH CK** *(I/O Channel Check)*. Estas señales pueden ser activadas por las tajetas ISA para solicitar que se genere una interrupción no enmascarada (NMI) en el procesador. Es activada para indicar la detección de un error incorregible.

- **I/O CH RDY** *(I/O Channel Ready)*. Permite a las tarjetas ISA más lentas la prolongación de los ciclos de I/O o de memoria, insertando estados de espera. El estado normal de esta señal es el de activo alto (listo); las tarjetas ISA llevan esta señal a su estado bajo (no listo) para insertar ciclos de espera.

- **-IOR** *(I/O Read)*. Esta señal es recibida por quien controle el bus, e indica que el dispositivo I/O seleccionado debe colocar datos en el bus de datos.

- **-IOW** *(I/O Write)*. Esta señal es recibida por quien controle el bus, e indica que el dispositivo I/O seleccionado debe capturar los datos en el bus correspondiente.

- **IRQ3** a **IRQ7** y **IRQ9** a **IRQ12** y **IRQ14** a **IRQ15** *(Interrupt Requests)*. Son utilizadas para indicar al procesador del sistema que una tarjeta ISA requiere de su atención. Es generada una solicitud de interrupción cuando alguna línea IRQ es colocada en estado alto. Dicha línea debe ser sostenida hasta que el microprocesador reconozca la solicitud a través de su rutina de servicio de interrupción. Estas señales tienen prioridades, siendo de IRQ9 a IRQ12 y de IRQ14 a IRQ15 las de mayor prioridad (IRQ9 la de mayor prioridad) y de IRQ3 a IRQ7 las de menor prioridad (IRQ7 la de menor).

- **-SMEMR**, *(System Memory Read)*. Le indica al dispositivo de memoria seleccionado que debe llevar datos al bus de datos. Esta señal se encuentra activa sólo cuando la dirección de memoria decodificada está por debajo del megabyte.

- **-SMEMW** *(System Memory Write)*. Le indica al dispositivo de memoria seleccionado que debe almacenar los datos del bus. Esta señal se encuentra activa sólo cuando la dirección de memoria decodificada está por debajo del megabyte.

- **-MEMR** *(Memory Read)*. Le indica al dispositivo de memoria seleccionado que debe llevar datos al bus de datos. Se encuentra activa en todos los ciclos de lectura de memoria.

- **-MEMW** *(Memory Write)*. Le indica al dispositivo de memoria seleccionado que debe llevar datos al bus correspondiente. Se encuentra activa en todos los ciclos de lectura de memoria.

- **-REFRESH** *(Memory Refresh)*. Es llevado a estado bajo para indicar que una operación de refresco de memoria se está llevando a cabo.

- **OSC** *(Oscillator)*. Es un reloj con un periodo de 70 ns (14.318 MHz). Esta señal es síncrona con el reloj del sistema (CLK).

- **RESET DRV** (*Reset Drive).* Se lleva a estado alto para borrar o reiniciar la lógica del sistema.

- **TC** (*Terminal Count).* Proporciona un pulso para indicar que una cuenta terminal se ha alcanzado en una operación de DMA.

- **-MASTER** (*Master).* Es usado por las tarjetas ISA junto con la línea DRQ para ganar el control del bus ISA. Después de recibir la señal –DACK un dispositivo puede poner en estado bajo la línea –MASTER para obtener el control sobre las líneas de dirección, datos y control; Después de que –MASTER está baja, el dispositivo debe esperar un periodo CLK para manejar dichas líneas, y dos periodos antes de realizar alguna acción de lectura o escritura.

- **-MEM CS16** (*Memory Chip Select 16).* Es llevada a estado bajo por un dispositivo de memoria de tipo esclavo para indicar que es capaz de realizar la transferencia de datos de 16 bits. Esta señal es manejada con la decodificación de las líneas de dirección LA23 y LA17.

- **-I/O CS16** (*I/O Chip Select 16).* Es llevada a estado bajo por un dispositivo de I/O de tipo esclavo para indicar que es capaz de realizar la transferencia de datos de 16 bits. Esta señal es manejada con la decodificación de las líneas de dirección SA23 y SA17.

- **-0WS** (*Zero Wait State).* Es llevada a estado bajo por un dispositivo de bus de tipo esclavo para indicar que es capaz de realizar un ciclo de bus sin insertar más espacios de espera. Para realizar un ciclo de memoria de 16 bits sin estados de espera, -0WS es obtenido mediante la decodificación de una dirección.

- **-SBHE** (*System Byte High Enable).* Es llevada a estado bajo para indicar una transferencia de datos de la mitad alta del bus de datos (D15 a D8).

Diagramas de tiempo del Bus ISA

A continuación se muestran, en las figuras 7 y 8, las secuencias correctas para el funcionamiento del bus ISA, la primer secuencia es para efectuar un ciclo lectura o escritura en un dispositivo de entrada/salida (I/O) y la segunda es la necesaria para efectuar un ciclo de lectura o escritura en memoria.

Ciclo de I/O de 8 bits

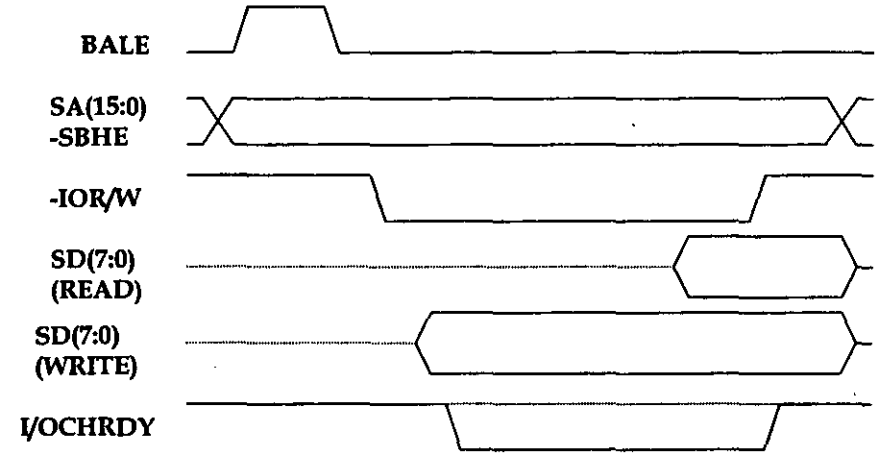

Fig 7[19]. Diagrama de tiempos de un ciclo de I/O de 8 bits.

Ciclo de memoria de 8 bits

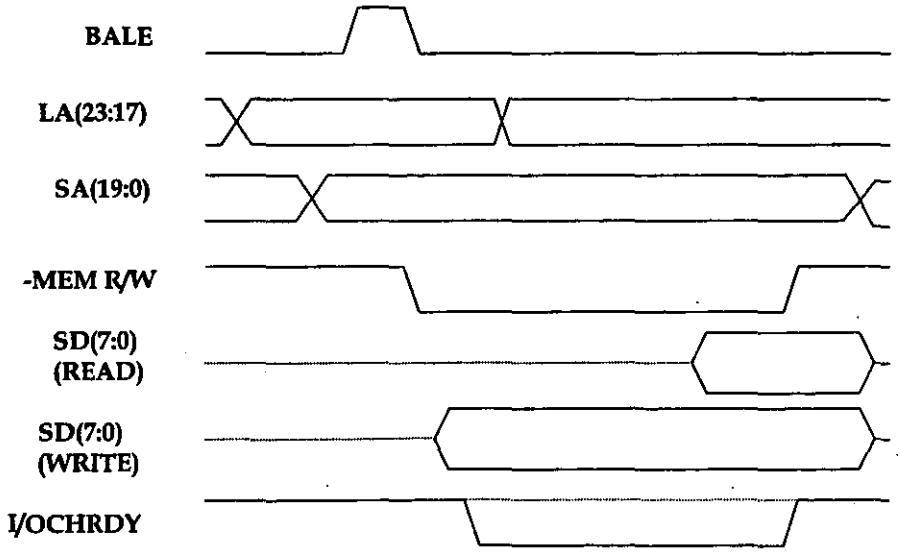

Fig 8[20]. Diagrama de tiempos de un ciclo de memoria de 8 bits.

[19] Ver la referencia número 12.
[20] Ver la referencia número 12.

Bus EISA

EISA es el acrónimo en inglés de *Extended Industry Standard Arquitecture*; es una extensión de la arquitectura ISA. EISA es compatible con el estándar ISA, lo que significa que las tarjetas originalmente diseñadas para el bus de 8 bits de IBM (a menudo conocido como bus XT) y que las tarjetas diseñadas para el bus de 16 bits (conocido como bus AT o ISA bus), funcionarán en una ranura tipo EISA. Las tarjetas EISA no funcionan en una ranura AT o XT.

El conector EISA utiliza múltiples renglones de conectores. El renglón superior es el mismo que una ranura ISA, y el renglón inferior contiene la extensión EISA. La ranura EISA está diseñada para que las tarjetas ISA no tengan contacto con las señales propias de la extensión.

Entre las señales extra que se manejan se encuentran las siguientes:

- **BE(x)** (*Byte Enable)*. Indica al dispositivo esclavo cuales bytes del bus de datos contienen datos válidos. Una transferencia de 16 bits afirmará BE0 y BE1, por ejemplo, pero no así BE2 y BE3.

- **EX16** (*EISA Slave Size 16)*. Es utilizada por un dispositivo de memoria de tipo esclavo para indicar al controlador del bus que está listo para transferencias de 16 bits.

- **EX32** (*EISA Slave Size 32)*. Es utilizada por un dispositivo de memoria de tipo esclavo para indicar al controlador del bus que está listo para transferencias de 32 bits.

- **IO16** (*I/O size 16)*. Es generada por una tarjeta de 16 bits cuando es decodificada por el controlador del bus.

- **M16** (*Memory size 16)*. Acceso a memoria de 16 bits.

- **SHBE** (*System Bus High Enable)*. Indica una transferencia de datos de 16 bits.

- **CMD** (*Command Phase)*. Esta señal indica que el ciclo actual de reloj se encuentra en la fase de comando. Después de la fase de inicio (START), el dato es transferido durante esta fase. CMD permanecerá alta desde que START decae, hasta el final del ciclo del bus.

- **START** (*Start Phase)*. Esta señal se encuentra baja cuando el ciclo de bus actual está en la etapa de inicio. La dirección y las señales M16/IO16 son decodificadas durante esta etapa. Los datos son transferidos durante la etapa de comando (CMD).

Análisis de los recursos físicos (hardware)

Requerimientos de la computadora personal

Como ya se mencionó, el sistema tiene bases de tiempo que son muy rápidas, difíciles de soportar con un procesador actual, es decir, las escalas inferiores al medio segundo exigen un programa eficiente, así como un procesador lo suficientemente rápido que pueda trabajar con estas bases de tiempo. Se sugiere utilizar un procesador Pentium a 600 MHz como mínimo. Dichas especificaciones requieren también que se evite utilizar alguna otra aplicación simultáneamente. En cuanto a memoria RAM, se sugiere un mínimo de 64 Mbytes. En disco duro, la capacidad depende del máximo de días que puede durar un crucero y el número de horas que se utilice la grabadora; dada la capacidad actual de los discos duros, un disco de 18 Gbytes sería adecuado.

Por otro lado, dada la resolución tan impresionante que permite la grabadora actual, es necesario contar con un monitor sofisticado de suficiente tamaño para aproximar la presentación de los resultados a los desplegados en la grabadora. El sistema actual funciona muy bien para un monitor de 1024 x 768 pixeles[21], aunque evidentemente, con una mayor resolución y algunas modificaciones al programa se podría tener un mejor despliegue gráfico, que por el momento no se considera necesario.

Tarjeta de expansión de puerto paralelo

Analizando los circuitos necesarios para la obtención y manipulación de la señal, se pueden concebir varias opciones:

La primera opción pensada fue la utilización de un convertidor analógico digital. Por costo y disponibilidad fue pensado un ADC0809, el cual tendría que ser incorporado a una tarjeta de expansión, con el fin de tomar la señal analógica de la grabadora. Esta alternativa presenta una seria desventaja: el convertidor elegido, aunque en sus especificaciones dice lo contrario, experimentalmente se comprobó que era demasiado lento, e incluso únicamente soportaba bases de tiempo superiores al segundo. La inclusión de un ADC más rápido, y por consecuencia más costoso, no tiene ninguna ventaja, ya que se cuenta con el de la grabadora, éste es lo suficientemente veloz para soportar todas las bases de tiempo. Debido a que la grabadora es la proveedora de la base de tiempo, no se puede pensar en sustituirla y por ende el convertidor integrado a la misma puede ser utilizado sin ningún problema. La única forma en que se tendría que pensar en utilizar un convertidor eficiente, es tomando la decisión de sustituir completamente la grabadora, lo que implicaría desarrollar un sistema capaz de enviar los pulsos de la base de tiempo al ecosonda; dicho sistema va más allá del objetivo de esta tesis.

La segunda opción, es tomar los datos vía puerto de la impresora, dado que la grabadora proporciona la señal analógica proveniente del ecosonda en forma digital; además, el conector DIN de la grabadora proporciona los pulsos de la base de tiempo, así como la señal de la corredera. Esta opción fue probada experimentalmente, durante el crucero una vez que se descartó la utilización del convertidor, obteniendo resultados

[21] Un pixel es la unidad mínima desplegable por un monitor de una computadora.

favorables. Se tuvo que descartar cuando se consideró la necesidad de imprimir las gráficas durante el recorrido, y aunque se podrían intercambiar la impresora convencional y la grabadora, esto es poco práctico, y no cumpliría con el objetivo de entregar al investigador un sistema fácil de usar.

Consecuencia de las carencias de la opción anterior, es la tercer opción, en la que se utiliza una tarjeta de expansión para la interfaz con la grabadora y el puerto de la impresora para la impresión de las gráficas desplegadas.

Otra opción pudo haber sido el uso del puerto serial de la computadora, para así permitir el manejo de las impresiones por el puerto paralelo. Sin embargo, esto implica tener circuitos de apoyo para efectuar la conversión de los datos a forma serial, para lo que se tendrían que elevar los valores de voltaje de los circuitos TTL de la señal digital, a los correspondientes de la transmisión en serie. Por otro lado, una de las ventajas de la comunicación serial, que es la capacidad de recorrer mayores distancias que la comunicación en paralelo, no se estaría utilizando, dado que la grabadora se encuentra muy cerca de las mesas de experimentación.

Análisis del código (software)

Se identificaron los siguientes requerimientos en cuanto a software:

- una interfaz gráfica para el despliegue;
- un programa que sea capaz de aceptar interrupciones o mantener un poleo[22] constante para la marca externa y el pulso del ecosonda;
- una rutina de conversión;
- una rutina de despliegue de puntos;
- una interfaz con controles de ganancia, umbral, velocidad y cambio en la base de tiempo;
- un módulo de almacenamiento de datos en memoria y recuperación de gráficas anteriores.

Dadas las consideraciones anteriores y el análisis de los datos se encontraron algunos puntos críticos, a considerar en las soluciones:

- primeramente, el sistema debe tener opciones en la base de tiempo que son difíciles de obtener con un procesador actual, es decir, las escalas inferiores al medio segundo, exigen un programa eficiente que trate de utilizar las capacidades de procesamiento de la máquina al máximo;

[22] Se toma la palabra poleo, a partir del término en inglés *polling*, que consiste en tener un ciclo donde se consultan al inicio los estados de algunas variables con el fin de tomar decisiones y/o mantener sincronía.

- los datos digitalizados tendrán que ser seleccionados en intervalos, y deberán poder ser modificados según los controles de umbral, ganancia y velocidad.

- se requerirá diseñar el módulo de almacenamiento de archivos, para cubrir el hecho de que el papel se puede almacenar y volver a interpretar en cualquier momento; esta misma problemática queda cubierta completamente con un módulo de impresión en una impresora convencional.

La solución más sencilla, pero a la vez menos eficiente consiste en un programa que se encargue de hacer todo en un mismo hilo de ejecución. Primeramente que se encargue de la creación de la interfaz gráfica, posteriormente que comience a obtener un cierto número de datos, finalmente que los despliegue en pantalla. Luego deberá repetir las operaciones de obtención y despliegue de datos hasta que se llene la pantalla para proceder a restaurarla. Si el usuario desea hacer uso de alguno de los botones o funciones del sistema, el flujo actual del programa deberá interrumpirse para atender dichas acciones, para posteriormente regresar a la obtención y despliegue de datos.

Una segunda opción es poder manejar dos hilos de ejecución, uno para el despliegue gráfico y otro para la conversión de datos; así, mientras se despliega un renglón de datos, se pueden obtener otros para hacer más rápido el sistema. Para este caso, es necesario contar con la suficiente sincronización para que los datos sean entregados y solicitados en el momento adecuado. El mismo hilo para el despliegue gráfico puede atender los eventos originados por el usuario, y de ser necesario, suspender el hilo de obtención, para ganar velocidad en la atención del evento.

La opción más eficiente hasta el momento encontrada, consiste en tres hilos de ejecución: el primero que se encarga de crear los otros dos hilos, el de conversión y el de despliegue de datos, así como de atender los eventos generados por el usuario. Cuando el usuario desencadena una acción, el hilo principal puede incluso suspender ambos hilos, atender al usuario, y posteriormente reanudarlos. Si cuentan con la sincronía adecuada, una suspensión de ambos hilos no causa ningún problema.

Algunas consideraciones que se requieren establecer al momento de programar los hilos de ejecución, son las siguientes:

- Es necesario un completo control en la sincronización, para que en ningún momento pueda llegar a suspenderse completamente el programa.

- Hay que considerar que aún cuando no sea necesario suspender alguno de los hilos en la atención de un evento, el procesador puede tener demasiados procesos que atender y resultar en una acción más lenta que haber suspendido ambos hilos.

- Dado que el hilo de conversión está sincronizado por el pulso proveniente de la grabadora, cada que este hilo sea suspendido, deberá sincronizarse nuevamente con el sistema externo, es decir, si el hilo de conversión esta acumulando datos antes de enviarlos al despliegue, y en ese momento es suspendido, cuando sea reanudado, deberá eliminar esos datos y esperar a que venga un nuevo pulso externo para comenzar a tomar un nuevo grupo de datos.

Análisis del lenguaje de programación.

En un principio se pensó y comenzó a hacer la aplicación en un lenguaje estructurado como C, ya que es un lenguaje muy difundido y de uso común. No obstante, al momento de la realización de la interfaz, dado que en esta aplicación el manejo gráfico es fundamental, se encontraron muchas dificultades para un desplegado interactivo, fácil de usar y a la vez con toda la funcionalidad deseada. Por esto se comenzó a buscar algún lenguaje orientado al manejo de gráficos.

Se decidió Java, por que es un lenguaje orientado a objetos, con un gran número de ventajas sobre el lenguaje C. Entre dichas ventajas, se pueden mencionar las siguientes:

- Manejo de hilos de ejecución.
- Manejo de eventos mediante el modelo de "Listeners" (escuchas).
- Creación sencilla de interfaces gráficas.
- Manejo sencillo de flujos de datos hacia archivos.
- Compresión de estos últimos.
- Facilidad de conexión hacia bases de datos (para futuro desarrollo).

No obstante las ventajas mencionadas anteriormente, también se encontraron algunas adversidades en Java, las cuales C pudo resolver. Estas fueron:

- Dificultad de interacción con los puertos de la computadora. Directamente la versión estándar de Java no contiene ninguna clase que permita acceder a bajo nivel a los puertos de la computadora. Existe una extensión de Java la cual está pensada para el manejo de los puertos serial y paralelo; sin embargo, después de un análisis y pruebas de esta API (*Aplication Programming Interface*) se concluyó que no podría cubrir las necesidades del sistema, tales como lectura y escritura vía puerto paralelo y asignación de una dirección de puerto distinta a la utilizada por la impresora convencional.

- Menor velocidad de ejecución del código. Dado que Java es un lenguaje tanto interpretado como compilado requiere mayor tiempo de ejecución; esta cuestión es muy delicada para sistemas que requieren recibir en tiempo real los datos del ecosonda. Esta problemática es resuelta mediante el manejo de dicha interfaz con C, ya que es un código compilado de buena velocidad durante su ejecución, capaz de manipular los circuitos rápidamente.

Introducción a la programación en Java

Programación orientada a objetos en Java

Java es un lenguaje orientado a objetos. Algunos conceptos de programación orientada a objetos resultan similares a los de la programación en C++; no obstante las analogías entre Java y C son más claras que las de Java con el primero.

Introducción a las clases y los objetos

Una clase es una colección de datos y métodos que operan sobre esos datos[23]. Los datos y los métodos, en conjunto, generalmente sirven para definir los contenidos y las capacidades de algún tipo de objeto.

Los objetos son instancias de una clase. Una vez que se define una clase, se necesita una instancia en particular para trabajar, es decir un objeto de esa clase.

Al definir una clase, se crea un nuevo tipo de dato. Por lo tanto, se pueden declarar variables de este tipo. Una variable es simplemente un nombre que hace referencia a un objeto del tipo de la clase creada y no es un objeto en sí. En Java todos los objetos se deben crear dinámicamente. Esto, en la mayoría de los casos, se hace con la palabra **new**. Por ejemplo:

```
Circulo c;
c = new Circulo();
```

Ahora se ha creado una instancia de la clase Circulo (un objeto de tipo círculo) y se ha asignado a la variable **c**, que es del tipo Circulo[24].

Ahora que se ha creado un objeto, se pueden usar sus campos de datos; de igual forma para tener acceso a los métodos de un objeto, se usa la misma sintaxis empleada para el acceso a los campos. Siguiendo con el ejemplo anterior:

```
Circulo c = new Circulo();
c.radio = 3;
double a;
a = c.area();
```

Creación de un objeto

Cuando se crea un objeto usando la palabra **new**, a continuación viene el nombre de la clase seguido de un par de paréntesis. Toda clase en Java tiene por lo menos un método constructor, con el mismo nombre de la clase, cuyo propósito es efectuar toda la inicialización[25] necesaria para el nuevo objeto. Cuando no se define explícitamente este

[23] Un método es el término orientado a objetos para un procedimiento o función.
[24] Las letras reconocidas por Java no incluyen acentos, por lo tanto en la clase Círculo se omite el acento.
[25] Se empleará este término de *inicializar* para indicar que los atributos de un objeto toman ciertos valores al momento de la creación de éste.

constructor, Java crea un constructor por omisión que no toma argumentos ni efectúa inicialización alguna.

Esta es la forma en que funciona: la palabra clave **new** crea una nueva instancia dinámica de la clase; esto es, asigna el objeto nuevo. Posteriormente, se llama al método constructor, se pasa implícitamente el objeto nuevo (una referencia **this**[26]) y se pasan explícitamente los argumentos especificados entre paréntesis.

Existen dos puntos importantes respecto a nombrar y declarar constructores:

- El nombre del constructor es siempre el mismo que el de la clase.
- El tipo de retorno es implícitamente una instancia de la clase. En las declaraciones de contructor, no se especifica un tipo de retorno ni se emplea la palabra clave **void**[27]. El objeto **this** se regresa implícitamente; un constructor no debe usar una declaración **return**[28] para regresar un valor.

En ocasiones es deseable inicializar un objeto de distintas maneras, dependiendo de lo que resulte más conveniente en una circunstancia particular.

Cuando se crean varios constructores para una misma clase, cada uno con distintos parámetros o tipos de parámetros, estos constructores tienen el mismo nombre. En Java un método se distingue mediante su nombre, el número, el tipo y la posición de sus argumentos. Esto no sólo se aplica a los constructores, ya que dos métodos no son los mismos, a menos que tengan el mismo nombre y el mismo número de argumentos del mismo tipo, que se pasen en la misma posición en la lista de argumentos. Cuando se llama un método y hay más de uno con igual nombre (métodos sobrecargados), el compilador automáticamente selecciona uno que concuerde con los tipos de datos de los argumentos que se pasen. Los métodos sobrecargados pueden tener distintos tipos de retorno.

Variables de clase

Java no permite variables globales. Toda variable en Java se debe declarar dentro de una clase, así que Java usa la palabra clave **static** para indicar que una variable particular es una variable de clase, en vez de una variable de instancia. Esto es, sólo hay una copia de la variable, asociada con la clase, en vez de muchas copias de la variable asociadas con cada instancia de la clase. La copia única de la variable existe sin importar el número de instancias de la clase que se creen; existe y puede usarse, aunque nunca se cree un objeto de la clase.

[26] La palabra clave *this* implica la referencia de un objeto a sí mismo.
[27] La palabra clave *void* indica, en algún método, que no hay valor de retorno.
[28] La palabra clave *return* indica que un método al alcanzar su fin regresa algún valor, el tipo de valor que regresa debe ser indicado después de esta palabra clave.

Métodos de clase

Math es el nombre de una clase y **sqrt()** es el nombre de un método de clase (o método estático) definido en **Math**, el cual difiere de los métodos de instancia. Los métodos de clase se parecen a las variables de clase en varias cosas:

- Los métodos de clase se declaran con la palabra clave **static**.
- Los métodos de clase a menudo se denominan "métodos estáticos".
- Los métodos de clase se invocan mediante la clase y no con una instancia (aunque dentro de la clase se pueden invocar mediante el nombre de método nada más).
- En Java, los métodos de clase son lo que más se parece a los métodos "globales".

Destrucción de objeto

Java se encarga de la destrucción de objetos, para que el programador se concentre en cuestiones de funcionalidad.

La técnica que Java usa para deshacerse de objetos cuando éstos ya no son necesarios, se denomina recolección de basura. Debido a que el intérprete de Java sabe qué objetos ha asignado, qué variables se refieren a qué objetos y cuáles objetos hacen referencia a otros objetos, puede saber también cuándo un objeto asignado ya no tiene referencia en otro objeto o variable. Una vez que encuentra dicho objeto, lo destruye sin peligro. El recolector de basura también puede detectar y destruir "ciclos" de objetos que tengan referencia entre sí, pero que no tengan referencia con ningún otro objeto.

Finalización de objeto

Justo como un método constructor inicia valores y estado de un objeto, un método finalizador de Java efectúa la finalización de un objeto.

La recolección de basura libera automáticamente los recursos de memoria empleados por los objetos. Sin embargo, los objetos pueden almacenar otros tipos de recursos, como descriptores de archivo o *sockets*, que el recolector de basura no puede liberar; por ello, se deberá escribir un método finalizador que cierre archivos abiertos, termine conexiones de red, etc.

Subclases y herencia

Este mecanismo es útil cuando se cuenta con una clase y se desea crear una especialización de la misma. Es decir, esta subclase tiene toda la funcionalidad de la clase original, más otra funcionalidad. Java permite definir esta clase como una extensión o subclase de la original. La palabra clave **extends** dice a Java que una clase determinada es una subclase de

otra especificada inmediatamente después de esta palabra clave, y hereda los campos y métodos de dicha clase:

```
public class Lienzo extends java.awt.Canvas{ ...
```

Si la clase padre, o superclase, contiene una variable de clase, ésta puede ser utilizada directamente por la subclase; de igual forma un método de la clase padre puede ser utilizado sin ser definido en la case hija o subclase.

Jerarquía de clases

Toda clase que se defina tiene una superclase. Si no se especifica la superclase con una cláusula **extends**, la superclase será la clase **Object**. **Object** es una clase especial por dos razones:

- Es la única clase sin una superclase.
- Los métodos definidos por **Object** pueden ser llamados por cualquier objeto de Java.

Como cada clase tiene una superclase, las clases en Java forman una jerarquía de clases, la cual se puede representar como un árbol, con **Object** como su raíz.

Sobreescritura o invalidación de métodos

Cuando una clase define un método que usa el mismo nombre, el mismo tipo de retorno y los argumentos como un método en su superclase, el método en la clase invalida al método en la superclase. Cuando se invoca el método para un objeto de la clase, se llama a la nueva definición del método y no a la definición antigua de la superclase.

La sobrecarga de método se refiere a la práctica de definir múltiples métodos (en la misma clase) con el mismo nombre pero con distintas listas de argumentos; esto es distinto a la sobreescritura, que involucra el concepto de herencia, mientras que la primera no lo hace.

Interfaces

Una restricción o cualidad, en Java, es que una clase únicamente puede tener una superclase. No obstante, es deseable en ocasiones que una clase tenga la funcionalidad tanto de una clase como de otra. La solución para este problema se denomina interfaz. Una interfaz se parece mucho a una clase abstracta, excepto en que usa la palabra clave **interface** en vez de las palabras **abstract** y **class**[29]. Una restricción es que toda variable declarada en una interfaz debe ser **static** y **final**[30]; esto es, deben ser constantes. Para instrumentar una interfaz, una clase primero debe declarar la interfaz en una cláusula

[29] Una clase de tipo *abstract* es similar a una interfaz, sin embargo, dado que Java sólo permite herencia sencilla, es preferible utilizar una interfaz y aprovechar la herencia para heredar métodos con cuerpo.
[30] La palabra clave *final* sirve para indicar, en caso de variables, que el valor asignado a una variable ya no podrá ser cambiado, es decir, se crea una constante.

implements[31] y después proporcionar una instrumentación (esto es, un cuerpo) para todos los métodos **abstract** de la interfaz. Es posible instrumentar más de una interfaz, simplemente se debe proporcionar una instrumentación para todos los métodos abstractos, en todas las interfaces.

La plataforma de Java

Las clases de la plataforma de Java están contenidas en grupos relacionados, conocidos como paquetes. Esta plataforma es una basta colección de clases predefinidas, disponibles para cada programa en Java, sin importar la plataforma en la que éste se ejecute.

Objetos String y el tipo de dato primitivo char

Las cadenas de texto son fundamentales, además de ser un tipo de dato sumamente utilizado. En Java, estas cadenas no son un tipo de dato primitivo, como **char**, **int** y **float**. Son representadas por la clase **java.lang.String**, que define muchos métodos útiles para la manipulación de las cadenas. Los objetos String son inmutables; una vez que se crea un objeto de este tipo, no hay manera de modificar el texto que éste representa. Por lo tanto, cada método que opera sobre una cadena, normalmente regresa un nuevo objeto de tipo String.

Dado que los objetos de tipo String son inmutables, no se pueden manipular los caracteres de un objeto de este tipo sin originar una gran generación de objetos en memoria. Por lo tanto, si se requiere una manipulación considerable de los caracteres que conforman una cadena, es preferible utilizar un objeto de la clase **java.lang.StringBuffer**.

Además de estas clases, existen otras también orientadas al manejo de cadenas. Otra clase de uso común es **java.util.StringTokenizer**, la cual se utiliza para separar cadenas en sus distintos componentes, por ejemplo por palabras.

Los caracteres individuales son representados en Java por el tipo de dato primitivo **char**. La plataforma de Java también define la clase **Character**, la cual, a su vez, define métodos útiles para identificar el tipo de carácter o cambiarlo de minúsculas a mayúsculas, por ejemplo.

Manejo de números

Java proporciona los tipos de dato **byte**, **short**, **int**, **float** y **double** para la representación de números. El paquete **java.lang** incluye sus clases correspondientes **Byte**, **Short**, **Int**, **Float** y **Double**, cada una de las cuales es una subclase de **Number**. Estas clases pueden ser útiles ya que proporcionan métodos de uso frecuente, al igual que algunas constantes de uso común.

[31] Dado que para el mecanismo de herencia se usan tanto **extends**, para superclases, como **implements**, para interfaces, en lo siguiente se utilizarán palabras como *extender* e *implementar* para evitar ambigüedades en el tipo de herencia que se está utilizando.

Fechas

Java utiliza diversas clases para el manejo de fechas y tiempos. La clase **java.util.Date** representa un instante de tiempo (con precisión en milisegundos). Esta clase no es más que una representación del valor contenido en un tipo **long**, que contiene el número de milisegundos desde la media noche GMT[32] del 1 de Enero de 1970.

Clases de tipo Collection

Estas son un conjunto de clases auxiliares en el paquete **java.util** para trabajar con colecciones de objetos. Se definen dos tipos fundamentales de colecciones. Una **Collection** es un grupo de objetos, mientras que un **Map** es un conjunto de relaciones o asociaciones entre objetos. Un **Set** es un tipo de **Collection** en la cual no hay duplicados y una **List** es una **Collection** cuyos elementos están ordenados. **Collection**, **Set**, **List** y **Map** son todas interfaces, pero el paquete **java.util** también define implementaciones concretas. Otras interfaces importantes son **Iterator** y **ListIterator**, las cuales permiten recorrer los objetos de una colección. El manejo de colecciones es propio de Java 1.2; previo a esta versión se utilizan las clases **Vector** y **Hashtable**, las cuales son similares a **ArrayList** y **HashMap**, respectivamente. Una clase **Vector**, como su nombre lo indica, permite almacenar objetos de cualquier tipo, su almacenamiento es indizado, es decir el primer objeto se ingresa en el índice 0 del vector, el siguiente en el índice 1 y así sucesivamente; un **Vector** puede crecer dinámicamente al serle ingresados más elementos. Una clase **Hashtable**, también permite almacenar pares de objetos, siendo el primer elemento de este par la llave o referencia para acceder al segundo objeto.

Hilos de ejecución

En Java es relativamente sencillo trabajar con múltiples hilos de ejecución en un programa. La clase fundamental para la creación de un hilo de ejecución es **java.lang.Thread**. Existen dos maneras de definir un hilo de ejecución. Una es extender de la clase **Thread**, sobreescribir el método **run()**, y luego crear una instancia de dicha subclase. La otra manera es definir una clase que implemente (*implements*) el método de la interfaz **Runnable** y luego pasar una instancia de este objeto **Runnable** al constructor de la clase **Thread**. En cualquier caso, el resultado es un objeto de la clase **Thread**, donde el método **run()** es el cuerpo del hilo de ejecución. Cuando se llama al método **start()** de la instancia de **Thread**, el intérprete crea un nuevo hilo de ejecución que ejecuta el método **run()**; en este momento, por lo tanto, se tienen dos hilos de ejecución, el principal, que es donde se crea el objeto de tipo **Thread**, y el hilo secundario que se inicia cuando la instancia de **Thread** ejecuta su método **start()**. Este nuevo hilo continúa ejecutándose hasta que el método **run()** termine. Mientras tanto, el hilo original continúa corriendo por su cuenta, continuando con la sentencia siguiente a la llamada al método **start()**.

[32] GMT es el acrónimo en inglés de *Greenwich Meridian Time*

Interfaces gráficas de usuario

Java proporciona un paquete llamado Herramientas de Ventanas Abstractas o "*Abstract Window Toolkit*" (AWT) que contiene elementos gráficos comunes. Estos elementos se pueden agregar a una interfaz y ser colocados en cierta posición utilizando un control de interfaz.

Todas las clases para manejo gráfico extienden de una clase común, **Component**. Para crear una interfaz gráfica de usuario (GUI), se agregan componentes a un objeto de tipo **Container**. Debido a que un **Container** es también un **Component**, estos pueden ser anidados arbitrariamente. A menudo, se usa un **Panel** o un **Frame** para crear interfaces con elementos anidados.

Cada componente AWT usa código nativo para desplegarse en la pantalla. Cuando se ejecuta una aplicación en Java con Windows, los botones son en realidad botones de Windows. Cuando se ejecuta en una máquina UNIX que usa Motif, los botones son realmente botones Motif.

Aplicaciones y Applets

Un Applet es un programa en Java que corre en una página web, mientras que una aplicación es aquélla que se inicia desde la línea de comandos. Un **Applet** es un **Panel** que es automáticamente insertado en una página web. El navegador (como *netscape* o *explorer*) que despliega la página, crea una instancia del **Applet** y lo agrega en la parte adecuada de la página. El navegador indica al **Applet** cuándo crear su interfaz gráfica (llamando su método **init()**) y cuándo iniciar, **start()** y detener, **stop()**, cualquier proceso en especial.

Las aplicaciones se inician desde la línea de comandos. Cuando se ejecuta una aplicación, el intérprete inicia llamando el método **main(String args[])** de la misma.

La lógica básica de una interfaz gráfica es la siguiente:

1. Crea una interfaz agregando componentes a los objetos **Container**.
2. Crea los manejadores de evento que responderán a las interacciones del usuario con la interfaz.
3. Despliega la interfaz.

Cuando se despliega una interfaz AWT, el intérprete comienza un nuevo hilo que vigila la interacción del usuario con la interfaz. Este nuevo hilo se mantiene latente hasta que el usuario presiona una tecla, mueve o presiona el mouse, o ejecuta cualquier otro evento a nivel de sistema que afecta la interfaz. Cuando este hilo recibe algún evento, llama a algún manejador de eventos, de los que se crean en la interfaz.

A continuación se muestran algunos de los componentes que tiene AWT. La mayoría, pero no todos, extienden directamente de la clase **Component**.

- **Button**. Botón con una etiqueta, que puede ser presionado con el mouse.

- **Canvas**. Es una región donde se puede dibujar libremente. Para extender de esta clase, se sobreescribe el método **paint()** para pintar los elementos deseados.
- **Checkbox**. Es una etiqueta con un pequeño cuadro seleccionable.
- **Choice**. Es una lista deplegable. La etiqueta visible en este objeto es el elemento actualmente seleccionado.
- **Label**. Es una etiqueta en la interfaz.
- **TextField**. Es una caja para insertar texto en línea.

Manejo de eventos

Los objetos se registran como susceptibles de ser escuchados. Si no existe escucha alguno cuando ocurre un evento, nada sucede. Si hay, por ejemplo, veinte escuchas registrados, cada uno tendrá la oportunidad de procesar el evento, en un orden no especificado.

Cuando un botón es presionado, un cierto método de la clase escucha es llamado. La clase escucha, en este caso debe implementar la interfaz **ActionListener**, debido a que los eventos del botón lo requieren. Cuando el botón es presionado, éste llama al método **actionPerformed** de la clase escucha; es en este método donde se realizan las acciones correspondientes a dicho evento.

De igual forma, la interfaz **ActionListener** es capaz de escuchar eventos de otro tipo de componentes, como **TextField** y **Choice**. También existen otras interfaces para cada uno de los distintos eventos que se pueden generar: **WindowListener**, **FocusListener**, **KeyListener**, **MouseListener**, **MouseMotionListener**, **ContainerListener** y **ComponentListener**

A continuación se muestra una tabla con los componentes más comunes y las interfaces capaces de escuchar sus eventos.

Componente gráfico	Interfaz capaz de escuchar los eventos
Button List MenuItem TextField	ActionListener
Choice Checkbox CheckboxMenuItem List	ItemListener
Scrollbar	AdjustmentListener
TextArea TextField	TextListener

Tabla 1[33]. Componentes gráficos más comunes y la interfaz correspondiente capaz de escuchar sus eventos

Cada uno de los escuchas de la tabla contiene un método. No obstante, los que no se encuentran en esta tabla contienen más de uno, ya que, por ejemplo, el resultado de mover el mouse, no necesariamente es el mismo de presionar su botón derecho, o presionar y arrastrar al mismo tiempo.

Análisis de eficiencia de Java

Para disminuir las carencias de Java en cuanto a velocidad se hizo un estudio de este lenguaje para aumentar al máximo su eficiencia.

Comparar la eficiencia de Java con lenguajes como C++ es poco justo. Cuando un lenguaje como C++ es compilado, el compilador toma el código fuente y produce código de máquina que puede ser entendido directamente por el procesador (código nativo). Por otro lado, Java es compilado produciendo algo conocido como "Java byte code" que es, por así decirlo, código de máquina para un procesador abstracto. Éste es independiente de la plataforma, por lo que cada sistema proporciona una Máquina Virtual de Java (JVM) que interpreta dicho "byte code". Claramente, interpretar este código siempre será más lento que ejecutar el código de máquina directamente, pero es el precio de tener independencia de plataforma.

La independencia de plataforma que proporciona Java, no es una cualidad que pueda ser aprovechada en este sistema, dado que se tiene una interfaz a bajo nivel con la máquina y dicha interfaz, por lo tanto, es dependiente de la plataforma, de tal forma que si se deseara ejecutar el sistema en otra plataforma, en cuanto al código de Java este cambio

[33] Ver la referencia número 5.

sería transparente, pero se tendría que desarrollar la interfaz correspondiente a dicha plataforma.

C++ es también un lenguaje ligado estáticamente: cuando el compilador produce un programa ejecutable, liga todas las referencias de las clases al código. Las clases de Java son dinámicamente ligadas. Cuando la JVM encuentra una referencia a una clase, busca en la ruta de clases al archivo correspondiente y lo carga en memoria. La ruta de clases está definida en la variable de ambiente CLASSPATH o bien por la bandera –classpath cuando la JVM es iniciada. La JVM busca las clases en cada directorio en el orden en que fueron especificadas. Simplemente poniendo las bibliotecas más frecuentemente usadas al principio de la ruta de clases, se puede mejorar la eficiencia de la JVM.

Finalmente, Java utiliza un recolector de objetos automático (Garbage Colector o GC) para el manejo de memoria. El GC libera memoria cuando ya no hay más referencias hacia un objeto. Esto tiene implicaciones en cuanto a velocidad y saturación de memoria. A pesar de que la mayoría de los diseños del GC son bastante eficientes, aún así es más lento que confiar en el programador para la liberación de recursos. Además, el GC puede accionarse en momentos inadecuados, lo cual puede afectar seriamente la velocidad y/o sincronización de la aplicación. Mientras la memoria se vuelve más escasa, el GC debe accionarse más a menudo, nuevamente haciendo lenta la aplicación. No obstante, el programador no tiene control sobre la liberación de memoria.

Aparentemente, existen más desventajas que ventajas para utilizar Java en un sistema en tiempo real, pero a continuación se describen técnicas que establecen la diferencia en cuanto a la eficiencia de este lenguaje.

La máquina virtual estándar funciona con estrictas restricciones de memoria. Cuando arranca, toma 1 Mbyte de memoria del sistema operativo. Durante la vida del programa, puede solicitar hasta un máximo de 16 Mbytes. Esta memoria mínima y máxima, puede ser configurada utilizando las banderas –ms y -mx (este rango de memoria es el bloque de memoria que el código en Java puede utilizar para instancias de nuevos objetos). Por ejemplo, el siguiente comando inicia la máquina virtual con 4 Mbytes y llega a un máximo de 24 Mbytes:

```
java -ms4m -mx24m Sonda
```

Esto evita que en la aplicación puedan ocurrir errores de desbordamiento de memoria, o que se vuelva más lenta cuando existen grandes cantidades de datos en ella (provocando que el GC sea invocado más a menudo). Hay que notar que cuando la máquina virtual ha solicitado memoria al sistema operativo, ya no la regresará (aún cuando ésta ya no sea utilizada en el programa).

La máquina virtual estándar de Java arranca el recolector de basura de manera asíncrona. Esto implica que el GC será llamado cuando la aplicación lo solicite explícitamente, cuando la memoria es escasa, o en ocasiones impredecibles en forma de un hilo de baja prioridad. Se proveen una serie de banderas que ayudan a controlar el recolector de basura.

La bandera –verbosegc indica a la JVM que escriba un mensaje en la consola cada vez que el recolector de basura se activa. Esto es útil para averiguar en qué momento se activa el recolector de basura; esto permitirá al programador detectar donde hay objetos

que, una vez usados por primera vez, se vuelven inútiles, y tratar de darles uso durante toda la aplicación[34].

La bandera –noasyncgc únicamente permite que se active el recolector de basura cuando se agota la memoria o cuando es llamado explícitamente.

De la misma forma que se almacenan instancias en memoria, Java debe almacenar información acerca de cada una de las clases utilizadas. Normalmente, cuando una clase no se utiliza, se remueve de la memoria; si se le necesita en una etapa posterior, debe ser almacenada nuevamente. La bandera –noclassgc evita que las clases sean recolectadas. Esto puede mejorar la velocidad del código, pero el precio puede ser una sobrecarga de memoria.

No cabe duda que la manera más eficiente de evitar que el recolector de basura se active es utilizando una programación en donde se utilicen al máximo los objetos creados al inicio de la ejecución y se evite la creación de objetos posteriormente. Es decir, si los objetos que son creados al inicio de la ejecución del programa se utilizan durante todo el código, el recolector de basura nunca se activará para eliminarlos, dado que se mantiene una referencia a ellos durante toda la ejecución. Si al contrario, en los ciclos o métodos que son llamados constantemente se crean objetos, estos pierden su referencia al momento de sustituirlos por nuevos objetos, y ocasiona que el recolector de basura se active constantemente.

Compiladores Just-In-Time

Los compiladores Just-In-Time o JIT optimizan enormemente la velocidad del código Java. Un JIT forma parte de la máquina virtual y es distinto al compilador de códigos fuente (javac).

Cuando un método es llamado por primera vez, el byte code de Java es compilado (en ese momento para la primera vez), y ejecutado como código nativo. El código compilado es almacenado para las llamadas subsecuentes al método. La fase de compilación es muy rápida y no introduce retardos perceptibles. La velocidad de ejecución se ve aumentada considerablemente, lo que puede ser la diferencia en una aplicación como la de este trabajo.

Si la intención de una aplicación es ser desarrollada para una plataforma en específico, probablemente es más conveniente escoger su compilación en código nativo. Entonces, la velocidad de la aplicación depende principalmente de la velocidad del compilador y las bibliotecas utilizadas en tiempo de ejecución. En resumidas cuentas, la diferencia entre utilizar un compilador JIT y la compilación a código nativo, resultan ser similares en cuanto a la eficiencia entregada.

Para este sistema, se utiliza la Máquina Virtual JDK1.2.2, la cual incluye un compilador JIT que presenta ciertas mejoras en relación con la versión 1.1 de esta máquina virtual. Entre las mejoras que favorecen directamente a este sistema, es que se agrega una memoria denominada "cache" para hilos de ejecución, que permite que los métodos sincronizados se ejecuten a velocidades cercanas a los métodos que no son sincronizados.

[34] Ver referencia número 2 capítulo 18 sección 4: *Java for real time systems.*

La máquina virtual HotSpot

Existe una versión más reciente de la máquina virtual de Java, llamada HotSpot, la cual contiene un recolector de basura más rápido y métodos sincronizados más rápidos. Sin embargo, lo más importante es que cuenta con un compilador Just-In-Time de optimización dinámica. Mientras el programa corre, se recolectan estadísticas de lo que sucede, ayudando a identificar los "puntos calientes" (hot-spots), o áreas críticas de código. Usando esta información, se pueden realizar optimizaciones dinámicas, lo que incluye optimización *in-lining* de métodos.

In-lining es una técnica utilizada por los compiladores para optimización en cuanto a velocidad, donde una llamada a un método es reemplazada por el código del método. Un compilador de código fuente puede hacer muy poco a este respecto, debido a que un programa en Java es de naturaleza polimórfica, ya que no puede determinar cuál método será ejecutado, dado que no puede determinar de qué clase es una instancia de un objeto. Sin embargo, debido a que HotSpot está optimizando dinámicamente mientras el código se ejecuta, puede determinar la clase de las instancias y hacer un *in-lining* adecuado.

Está técnica bien podría sobrepasar las velocidades de C/C++, dado que sólo realizan optimización usando información estática. Además, existen compañías, incluyendo Sun, que están desarrollando procesadores cuyo código nativo sea Java byte code.

Una desventaja de la máquina virtual Hot-Spot, es su técnica de *in-lining*, ya que ésta no es adecuada al manejarse hilos de ejecución, que implican ciclos infinitos. Debido a que la información relativa al funcionamiento interno de esta máquina virtual es restringida, no se tiene información de cómo se puede, si es que es posible, desactivar el *in-lining* en esta máquina virtual.

Por lo tanto, como se menciona previamente, se utiliza el compilador JIT proporcionado por la versión estándar del JDK 1.2.

Compilador de Java

La siguiente herramienta, que tiene un gran efecto en la eficiencia del código, es el compilador. El compilador estándar que contiene el JDK de Sun es javac. Este compilador cuenta con la bandera –O para optimización, esta bandera elimina mucha información que se utiliza durante la depuración, lo cual origina que las clases sean más pequeñas, remueve algo (no todo) de código redundante, e intenta hacer *in-lining* de algunos métodos.

Compiladores Java a código nativo

Este tipo de compiladores fueron creados para hacer los programas en Java tan competitivos como los programas en C. Es decir, eliminan la ventaja multiplataforma que tiene Java realizando una compilación a un código exclusivo de la plataforma.

No obstante, la mayoría de los compiladores de este tipo están orientados a sistema operativo linux. Los que están orientados al sistema operativo Windows 9x no son gratuitos, además de que muchos de éstos sólo soportan clases propias de la versión JDK1.1, e inclusive muchos excluyen las clases de los componentes gráficos, y finalmente otros no incluyen las clases para el manejo de hilos de ejecución.

Probablemente en un futuro exista alguna versión libre de cargo que soporte las clases que manejan gráficos, los hilos de ejecución y otras clases correspondientes a la versión 1.2 del JDK. Entonces, dadas estas condiciones, sería favorable para la eficiencia del programa un compilador de este tipo.

Perfiles

La máquina virtual de Java puede ayudar a identificar áreas críticas del código para la velocidad de la aplicación. Si la aplicación se corre con la bandera –prof:Sonda.prof, se escribirá información del perfil en el archivo Sonda.prof. El formato de este enorme archivo carece de documentación. Sin embargo, la información más útil se encuentra en el inicio, donde se listan todos los métodos que son llamados durante la vida de la aplicación, el método llamado, el número de veces que fue llamado y el tiempo utilizado por este método. Esta lista es ordenada por el número de veces que un método es llamado.

CAPÍTULO 4

DISEÑO DE LA INTERFAZ

Diagrama de conexión del ADC0809 en el puerto de expansión

Como ya se mencionó anteriormente, este convertidor fue colocado en una tarjeta de expansión de puerto paralelo, y controlado por medio de un programa en C desde la computadora, ver figura 9.

Se pueden identificar como salidas de la computadora hacia el convertidor la terminal *Address Latch Enable,* ALE, y la de inicio de conversión *Start Of Conversion,* SOC; la ultima señal que envía la computadora es la de *Output Enable,* OE para permitir la lectura del valor. Por el otro lado, como entradas a la computadora se encuentran la terminal *End Of Conversion,* EOC, así como las 8 terminales de salida, que dadas las características que se requieren en este proyecto, únicamente se utilizan 4 de estas.

A continuación se muestra la secuencia creada en el programa en C, para el funcionamiento adecuado del convertidor:

```
outport(puert,ale);        /*Habilita dirección*/
espera(z);
outport(puert,soc+ale);    /*Inicia conversión*/
espera(z);
outport(puert,0);          /*Esperando respuesta*/
```

```
espera(z);
x=inport(puert);

while((x&1)!=1||aux!=1){   /*Esperando fin de conversión*/
      if((x&1)!=0)
            aux=1;
      x=inport(puert);
}
outport(puert,oe);         /*Habilitando la lectura*/
espera(z);
x=inport(puert);           /*Nuevo dato*/
```

Previamente la constante *puert* tiene asignado el valor 31Fh, que es la dirección de la tarjeta; la dirección del multiplexor del convertidor se encuentra alambrada en la tarjeta, el programa da por hecho que existe una dirección fija; primeramente, se envían al puerto las señales de retención de la dirección del multiplexor e inicio de conversión, las constantes *ale* y *soc* tienen previamente asignados un par de bits del registro de datos, conectados a las terminales ALE y SOC del convertidor, respectivamente; tras un tiempo de espera, que permite hacer estable y lo suficientemente duradera esta salida, se llevan todas las salidas a estado bajo; entonces, empieza un ciclo de lectura de la tarjeta, hasta que la señal EOC del convertidor pasa de estado alto a bajo y regresa a su estado alto; una vez detectada esta señal, el ciclo de lectura termina; se envía a la tarjeta la señal OE, la constante *oe* tiene previamente asignado un bit del registro de datos, para que el convertidor entregue el valor de su conversión; y finalmente, se lee el valor de la conversión.

Diagrama de tiempos

Siguiendo el diagrama de tiempos proporcionado en la figura 10, se puede corroborar este algoritmo. Primeramente se necesita enviar al convertidor la dirección de la entrada analógica que va a tomar, la cual está físicamente alambrada. Mientras tanto, la señal analógica por capturar debe estar presente en la entrada deseada y debe ser estable. Posteriormente, se envía un pulso a la terminal ALE para fijar la entrada deseada. Inmediatamente después, se envía un pulso a la terminal SOC para que se inicie la conversión. Casi simultáneamente con este paso, el convertidor baja la terminal de salida EOC, indicando que la conversión se está llevando a cabo, y una vez que esta concluye, dicha terminal regresa a su estado alto, misma señal que indica al procesador que puede enviar la señal OE, para que las salidas cambien su estado de alta impedancia al valor correspondiente a la señal convertida y éste pueda leerlas.

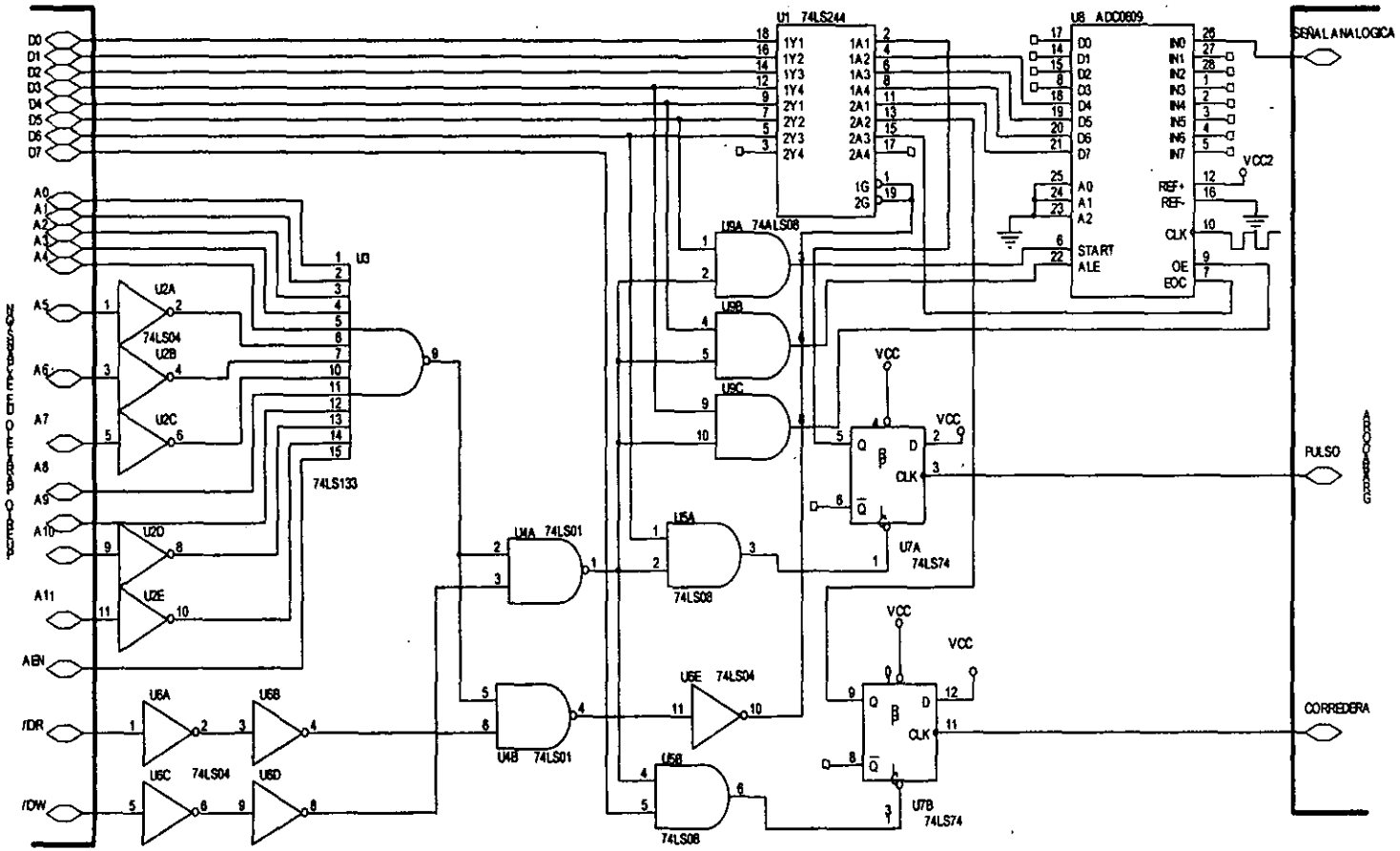

Fig. 9. Diagrama de bloques de la primer aproximación usando el ADC0809 con el puerto de expansión

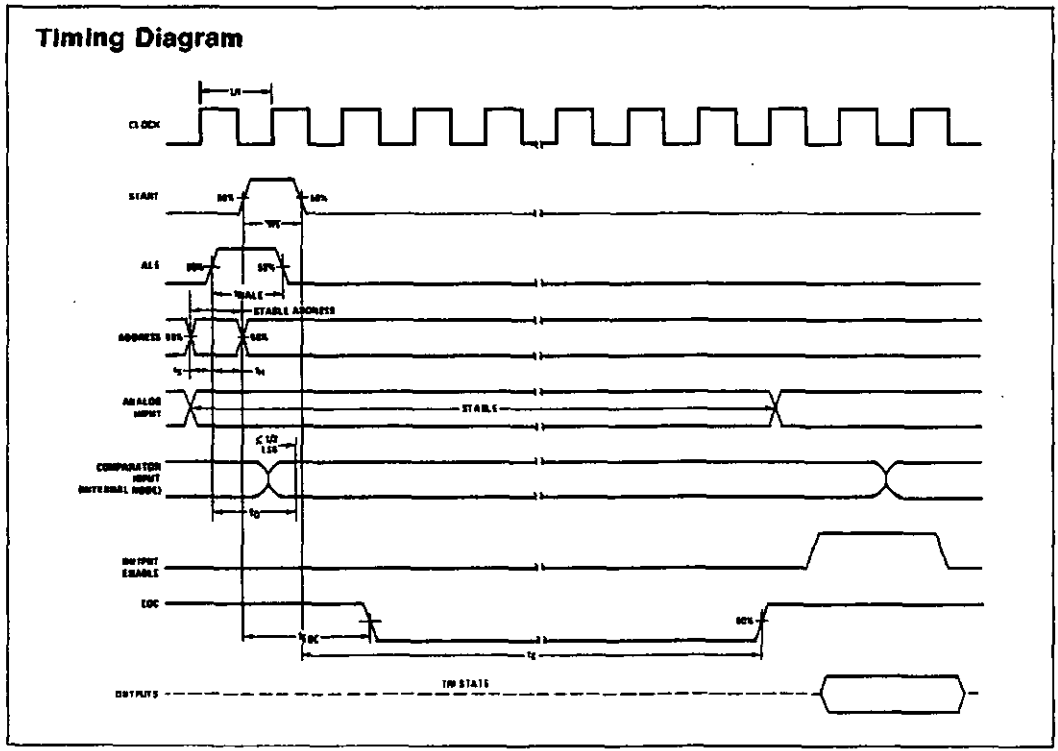

Fig. 10[35]. Diagrama de tiempos de las señales en el ADC0809

[35] Ver la referencia número 24.

Diagrama de la interfaz de la grabadora con el puerto paralelo

Como ya se mencionó, esta alternativa únicamente fue concebida como una prueba, dada la necesidad de probar el mayor número de bases de tiempo entregadas por la grabadora. El diagrama de la figura 11 tiene el par de señales de entrada, que son el pulso periódico y la señal de la corredera y las cuatro líneas digitales tomadas de la grabadora (D0 a D3). Primeramente se verifica la existencia del pulso o de la corredera, al existir el primero, éste queda retenido, después se captura cierta cantidad de valores digitales del puerto de datos, y posteriormente se borra el pulso retenido. En caso de la corredera, el programa registra la existencia de dicho pulso para reflejarla en la interfaz gráfica, y posteriormente borra el pulso retenido.

Se lee el pulso por el bit del registro de estado S3 del puerto paralelo de la impresora; este queda retenido, por lo que el programa envía el pulso activo bajo de borrado, por la terminal D0. Este pulso (S3) indica que se debe leer un dato; los 4 bits, que componen el dato, se reciben por el registro de estado S4, S5, S6 y S7, siendo S4 el bit menos significativo y S7 el bit más significativo, el cual es necesario invertir dado que dicha terminal se encuentra invertida.

```
while(pulso == 0){
  pulso=inport(puertstat);
  pulso=pulso&8;                /* s3 */
}
/* Pulso para borrar el flip-flop */
outport(puertdat,clear);        /*Flanco de bajada*/
espera(z);
outport(puertdat,D0);           /*Flanco de subida*/
espera(z);
while(i<=170000 && interno==0 ){
      x=inport(puertstat);
      /* s4:LSB s7:MSB */
      x=x^128;                  /* !s7 */
      x= x>>4 ;                 /* x tiene valor de 0 a 15*/
      ...
```

El segmento de algoritmo anterior, muestra la forma en que se consulta la existencia del pulso de la grabadora; una vez registrado éste, se comienza la obtención de los datos, la cual termina una vez que se registra la entrada de un nuevo pulso, o bien, cuando se alcanza el máximo número de puntos desplegables en la interfaz gráfica.

El diagrama de bloques muestra la conexión de los circuitos necesarios para la interfaz de la computadora con la grabadora mediante el puerto paralelo LPT1. Se utiliza el circuito buffer de triple estado, U1, para recibir las cuatro líneas de los datos digitales, así como el par de pulsos de la grabadora y la corredera. Se utilizan un par de bits salida, para borrar los flip-flops, U2A y U2B, una vez que se registra algún pulso

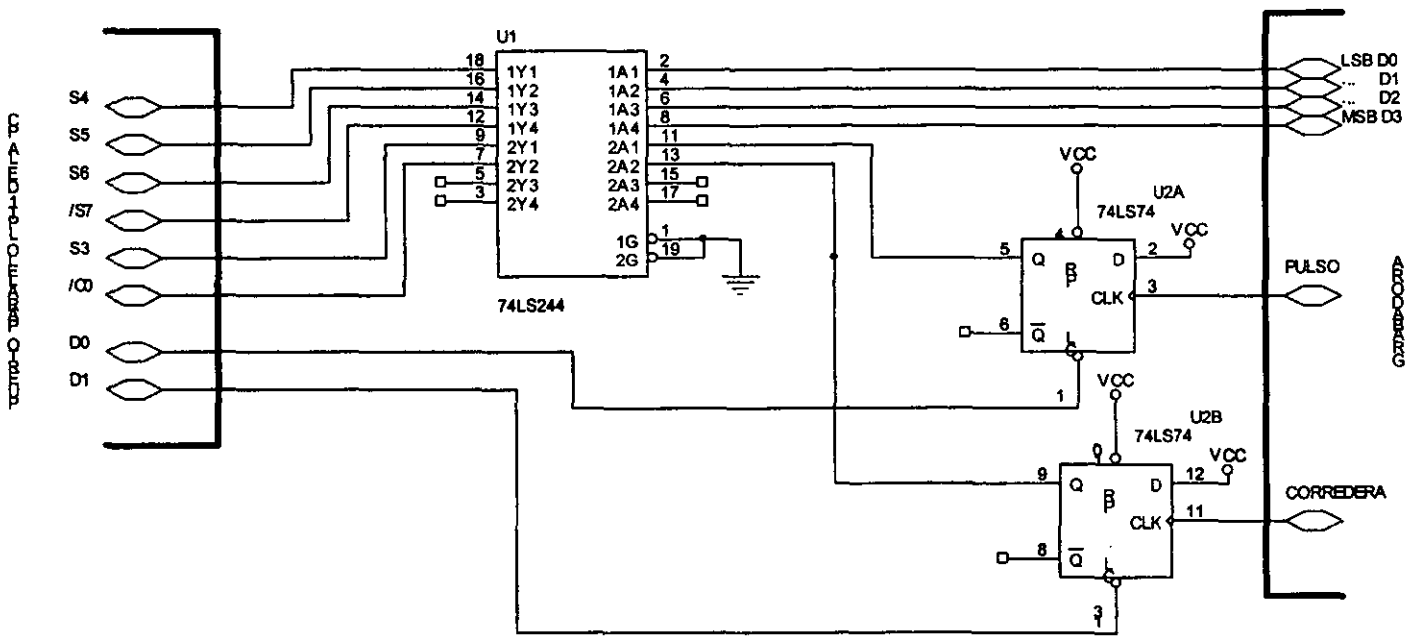

Fig. 11. Diagrama de bloques de la interfaz de la grabadora con el puerto paralelo

Diagrama de la interfaz de la grabadora con el puerto de expansión

Como se comentó anteriormente, esta es la propuesta final para la interfaz. La tarjeta de expansión se conecta a la computadora y recibe 6 señales del exterior: el par de pulsos constituidos por la corredera y la base de tiempo, y las cuatro líneas digitales (D1 a D4). La tarjeta no tiene necesidad de enviar al exterior señal alguna, por lo que sólo estas líneas conectan la PC con la grabadora.

Como se puede ver en el diagrama de la figura 12, los dos pulsos son retenidos en la tarjeta y una vez que el programa de control advierte la presencia de alguno, lo registra y borra el estado del circuito que lo retuvo (circuitos U7A y U7B).

Del lado izquierdo del diagrama se encuentran las entradas y salidas hacia la computadora, mientras que del lado derecho están las señales que se reciben de la grabadora.

Del lado de la computadora, se encuentran básicamente las líneas de dirección A0-A11, que como ya se mencionaba, la tarjeta responde a la dirección hexadecimal 31Fh. El bus de datos recibe los cuatro bits que componen cada uno de los valores tomados de la grabadora (D1-D4), además del pulso periódico emitido por esta misma (PULSO) y por último, el pulso emitido por la corredera (CORREDERA). Mientras que, a través de este mismo bus, la computadora envía un par de señales que sirven para borrar, en su momento, los flip-flops que reciben los pulsos (U7A y U7B), con el fin de poder recibir nuevamente otro pulso.

Del lado del conector DIN de la grabadora, como se mencionaba se reciben las señales de ésta; cuatro correspondientes los datos digitalizados, una del pulso periódico y la sexta señal correspondiente a la corredera.

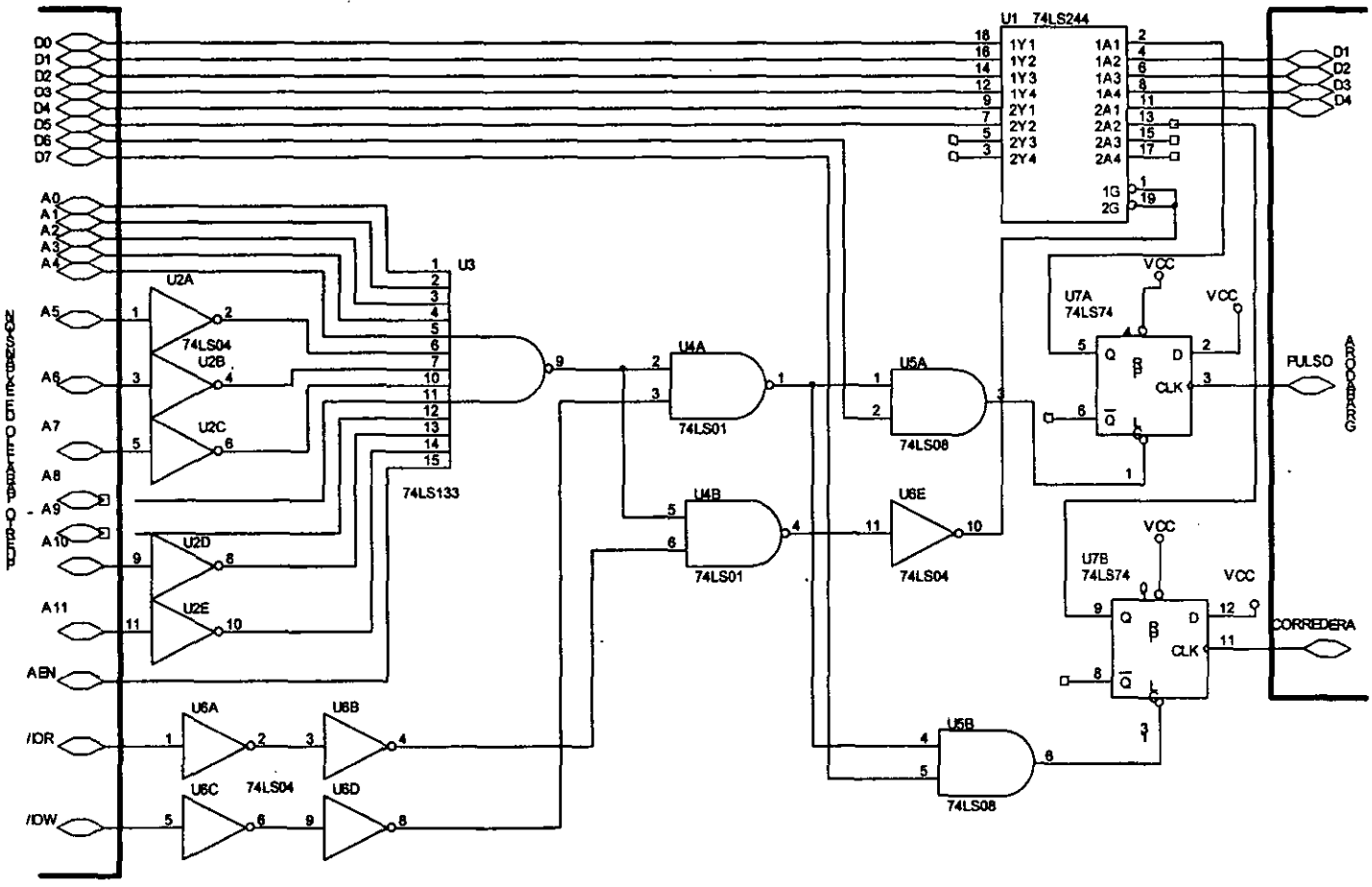

Fig. 12. Diagrama de bloques de la interfaz de la grabadora con el puerto de expansión

CAPÍTULO 5

DESARROLLO DEL PROGRAMA DE CONTROL

Propuesta para la interfaz gráfica

Dado el objetivo y las demás consideraciones ya vistas, para la realización del sistema se propuso el despliegue gráfico que se muestra en la figura 13.

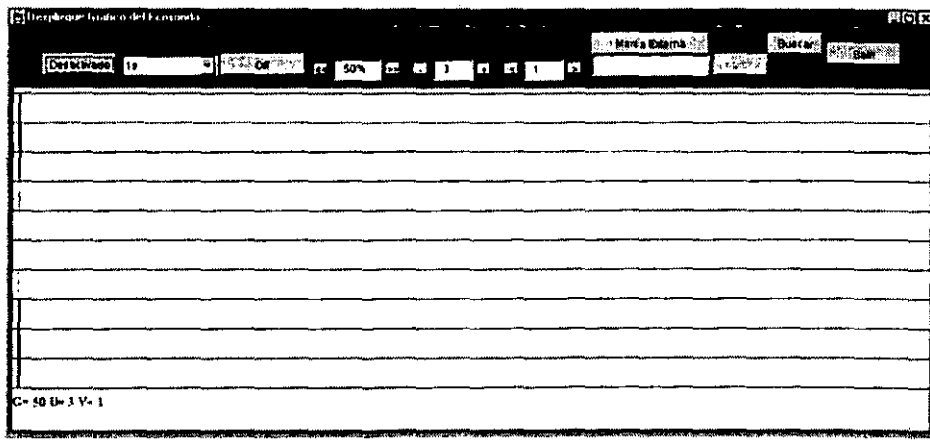

Fig. 13. Pantalla del despliegue gráfico propuesto

Con esta pantalla se puede ver más claramente el funcionamiento de la interfaz. Primeramente la pantalla se encuentra dividida en dos paneles. El panel superior es donde se encuentran los controles de la interfaz, que corresponden a la totalidad de las funciones necesarias. El segundo panel es la parte principal del despliegue de datos, es decir, aquí se despliegan los puntos correspondientes a los valores provenientes de la tarjeta. Así mismo, en este panel se tiene la opción de ver gráficas pregrabadas, esto es, el usuario podrá seleccionar un archivo de datos de alguna otra fecha y hora y éste será desplegado en dicho panel. Ahora se muestra con mayor detalle cada uno de los paneles y los requerimientos que cubren.

Panel de control

A partir de los controles de la grabadora, primeramente se hace un análisis para determinar la forma de cubrir cada uno de ellos. El aparato actual tiene dentro de sus controles los siguientes:

Control de velocidad en el avance del papel

Este control sirve para que el papel se desplace con mayor o menor velocidad, por lo que la imagen desplegada se verá más compacta o más alargada. Este control normalmente se encuentra en la menor velocidad de avance del papel, ya que este recurso es muy caro y mientras menos papel se utilice es mejor; además, los barridos se ven mucho mejor ya que se presenta una mejor delineación del contorno.

Por lo tanto, aunque este control se podría omitir, se dejó a final de cuentas, ya que la velocidad cero permite dejar de desplegar puntos en pantalla sin interrumpir el proceso de conversión. En el sistema actual esto es útil para que el papel deje de avanzar mientras el barco realiza alguna maniobra y no se desea apagar la grabadora, debido a que como se mencionó, la grabadora provee los pulsos periódicos para el disparo del transductor en el ecosonda.

Control de umbral

Este permite aclarar un poco las imágenes cuando se presentan demasiados puntos en zonas distintas al contorno de la superficie del fondo marino. Normalmente estos puntos no son de nuestro interés ya que se pueden deber al agitamiento del océano, algunos entes derivando en el mismo o trastornos en la señal capturada por el ecosonda. Estos puntos normalmente son de baja intensidad, por lo que al reducir el umbral desaparecen; por lo tanto, se decidió incluir este control.

Dado que tenemos puntos con valores entre 0 y 15, este umbral tendrá por lo tanto un rango entre 0 y 15, siendo el cero para cuando se desea que se vea reflejado en pantalla todo aquello que envíe el ecosonda y el quince para que la pantalla esté en blanco, de tal forma un valor intermedio filtrará todos los valores que estén por debajo o igual a ese

umbral y sólo permitirá pasar todos aquellos valores que estén por encima de dicho valor, obteniendo por consiguiente, una gráfica más limpia.

Control de ganancia

Este control sirve para que los tonos de gris se acerquen más hacia el negro, de tal forma que la gráfica tome mayor contraste entre el fondo blanco y los valores.

En la interfaz gráfica este valor es porcentual, por lo que los valores obtenidos son multiplicados por dicho porcentaje. Si el valor resultante sobrepasa el quince se acota dicho resultado a este número.

Control de la base de tiempo

Como ya se comentó, la grabadora es la fuente de pulsos para el ecosonda y estos pulsos son enviados periódicamente. El periodo es controlado a través de la grabadora, teniendo un rango que va de los 8 segundos para el periodo más largo y de 1/64 de segundo para el periodo más breve. Esta base de tiempo es seleccionada por el usuario, como respuesta los barridos cambian su frecuencia y la etapa de obtención de puntos requerida los entrega a mayor o menor velocidad.

Cabe hacer notar que este control de la interfaz gráfica y el de la grabadora deberán estar en el mismo valor, ya que la grabadora es quien en realidad genera el pulso, y el valor que se elige en la interfaz sirve de guía a la aplicación para mantenerse en sincronía con la grabadora.

Control de marca externa

Otra de las funciones es la de colocar una línea en la gráfica para marcar alguna zona importante, un inicio de transecto u otro acontecimiento que el usuario desee denotar.

Esta función es obtenida, en la interfaz gráfica, mediante un botón. Al presionar este botón se despliega una línea vertical en el panel superior. Dicha marca externa normalmente va acompañada de alguna anotación del investigador sobre el papel, en la grabadora actual. Se decidió agregar esta funcionalidad de colocar comentarios al momento de establecer una marca externa. Esto se logra mediante un campo de texto, que queda habilitado una vez que se presiona el botón de marca externa. Se introduce el comentario, y al presionar la tecla de *Enter*, dicho comentario se coloca en la gráfica al mismo tiempo que continúa el desplegado de datos.

Control de contraste

Como su nombre lo indica, esta perilla de la grabadora permite afinar un poco más los puntos desplegados en el papel; sin embargo, dado que con los controles de ganancia y de umbral se puede eliminar dicha función, se decidió omitir este control en la interfaz.

Control de resolución

Este control, que no se encuentra en la grabadora, se ideó primeramente para cubrir la necesidad de omitir las dos primeras franjas de datos que son inútiles al investigador. Dada esta necesidad, se optó por una solución que además entrega mayor funcionalidad al sistema actual. Ésta mejora consiste en un mecanismo de resolución o de *zoom* para que el usuario pueda agrandar un rango de profundidades deseado, y con ello tener una imagen más grande en donde se desee, y regresar posteriormente a la resolución principal. Esto a la vez de ser algo deseado por los investigadores, también cubre la posibilidad de poder desplegar sólo una pequeña parte de los puntos entregados por el ecosonda.

Con este control, el usuario puede seleccionar con el *mouse* o ratón una región del panel de desplegado. Esta región seleccionada será ahora el rango de profundidades en las cuales se tomaran y desplegarán los puntos en pantalla. Así, si suponemos que la gráfica sólo despliega 320 puntos, pero estos únicamente corresponden a la zona del fondo marino, se tendrá una resolución mucho mayor y muy cercana a la entregada por la grabadora original, a la vez de eliminar todo lo que está alrededor de la zona de interés, que puede ser molesto o distraiga el foco de las profundidades deseadas.

Control de dirección de barrido

En la grabadora actual, se tiene un interruptor que permite que se elabore la gráfica, ya sea en un sentido o en otro, esto es, con la superficie marina a mano izquierda, o bien, a mano derecha. Esta función no es utilizada en el aparato actual, además la interfaz gráfica es horizontal, por lo que da una mejor idea de cuál es la superficie y cuál es el fondo.

Control de dos desplegados

La grabadora tiene este mecanismo de impresión de puntos por duplicado, esto con el fin de tener dos canales de entrada de datos, por lo que se puede elegir desplegar un canal, el otro, o bien ambos canales a la vez. Cuando se elige desplegar ambos canales, se hace el despliegue de un canal en la mitad izquierda del papel y el otro canal se despliega su mitad derecha. Esto está pensado para poder desplegar la señal proveniente de dos ecosondas orientados en distintas direcciones, con el fin de obtener dos señales con datos distintos. No obstante esta función no es utilizada en el equipo actual, por lo que quedó descartada.

Control de archivos

Una de las ventajas de la grabadora con respecto a la interfaz es la de poder conservar los valores pasados. Esto es, cuando el usuario desea repasar alguna zona antes recorrida, lo que tiene que hacer es buscar el rollo de papel adecuado, o si esta zona fue recorrida hace poco tiempo sólo tiene que ver el papel que cuelga de la grabadora e identificar la zona deseada.

En la interfaz gráfica, inevitablemente es necesario borrar la gráfica inmediata anterior para poder desplegar una nueva. Para resolver esta carencia se planeó lo siguiente:

- En primer lugar, se pensó en dividir la interfaz en dos paneles de despliegue, uno para el desplegado de los valores actuales, y el segundo para el desplegado de valores pasados. Cuando en el primer panel, el superior, se alcanza el final de la pantalla, automáticamente se envía esta gráfica al panel inferior, y al mismo tiempo, estos datos son enviados a un archivo para su almacenamiento en disco, mientras que el primer panel es limpiado para continuar el despliegue de los datos. La demanda excesiva de recursos hizo descartar esta idea.

- La segunda necesidad se cubre con el control de archivos. Éste consiste en un botón, situado en el panel de control, que permite seleccionar un archivo de datos pasados. Los datos contenidos en éste, son desplegados de tal forma que el usuario pueda visualizar nuevamente alguna zona de su interés. Estos archivos llevan como nombre la fecha y hora de su creación para facilitar su identificación. La inclusión de comentarios dentro de la gráfica hace necesario que al crear un archivo, se queden almacenados de igual forma los comentarios. De esta manera, al momento de recuperar una gráfica, ésta tendrá los comentarios y las marcas que en algún momento el investigador introdujo, y así se pueda imprimir dicha información.

Control de impresión

Este control fue implementado como parte de las últimas mejoras al sistema. Dado que en el sistema actual la gráfica es totalmente impresa, y al final de todas las campañas dichas impresiones son empacadas y descargadas por los investigadores, se implementó la opción de imprimir mediante impresora convencional los archivos generados durante los recorridos; una vez impresas las gráficas los archivos pueden ser borrados, esto permite mantener el espacio en disco original sin que se pierda la información obtenida. Obviamente, también se pueden copiar a disquetes flexibles dichos archivos para ser transportados. El sistema también se tendrá que instalar en toda computadora donde se deseen analizar las gráficas.

La impresión convencional es un poco tardada, por lo que es necesario suspender la conversión y desplegado de datos hasta que esta finaliza. Por lo tanto, el investigador puede deshabilitar esta opción durante el recorrido, y posteriormente imprimir los archivos generados, si así lo desea. El botón de "Activar/Desactivar" permite tomar la decisión de imprimir o guardar archivos en tiempo real[36]. Por otro lado, existe el botón de "Imprimir" el cual está relacionado con la recuperación de archivos; durante el despliegue en tiempo real este botón se encuentra inhabilitado; cuando se recupera un archivo, dicho botón queda habilitado, de tal forma, el usuario tiene la opción de imprimir la gráfica recuperada.

[36] En el modo de "Activar" se maneja la impresión en tiempo real, mientras que en el modo "Desactivar" los datos se almacenan en disco.

Panel de desplegado

Lo único que resta explicar de la interfaz es el panel de desplegado. El panel es la pantalla en la que se están desplegando valores en tiempo real. El desplegado es de arriba abajo y de izquierda a derecha, por lo que la superficie oceánica se despliega en la parte superior de dicho panel, mientras que el fondo submarino se despliega, según la escala y la profundidad, a la mitad o en la parte inferior del panel. Los puntos desplegados tienen distintos colores, en lugar de tonalidades de grises; no obstante se conserva el blanco como ausencia de valor o valor nulo y el negro como el máximo valor. Los colores seleccionados se muestran en la figura 14:

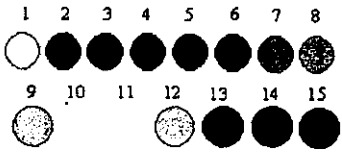

Fig. 14. Colores asignados de acuerdo a la intensidad del valor obtenido

Para desplegar valores pasados se tiene el mismo sentido de despliegue, con la diferencia de que éstos vienen de algún archivo seleccionado y el despliegue no está sincronizado, por lo que el desplegado es muy rápido. Al final de desplegar los valores, el sistema espera que el usuario presione la opción "Continuar" para reanudar el despliegue en tiempo real.

Interacción con los controles

Aquí se describe el funcionamiento conjunto de los distintos controles de la interfaz.

El cambio de la resolución o zoom, como ya se mencionó, permite al usuario ampliar el despliegue de cierta zona deseada. Este cambio de resolución tiene las siguientes repercusiones con los demás controles: si la resolución está activa, al cambiar la base de tiempo, aquélla se desactiva; de igual forma, al buscar un archivo el zoom se desactiva.

En cuanto al control de impresión: cuando se elige una base de tiempo determinada éste control se puede desactivar automáticamente, debido a que el sistema desactiva la impresión para bases de tiempo muy rápidas; no obstante, una vez que se elige una base de tiempo, se puede cambiar el estado del control de impresión. Otra consideración es que cuando se busca un archivo y se despliega en pantalla, el control de impresión automáticamente habilita la impresión para permitir que se impriman dichas gráficas.

El panel en el cual se despliegan las gráficas tiene una característica especial, a diferencia de cualquier aplicación que se maneje con ventanas. Esta característica consiste en que el repintado de dicho panel no recupera lo que se encuentra desplegado, es decir, si se arrastra o se coloca una ventana u otro objeto enfrente del panel, al momento de retirarlo todo aquello que se encontraba desplegado en el panel se pierde. Esto se debe a que el repintar un elemento gráfico tiene un gran consumo de recursos de la computadora, además de que es un evento que ocurre de manera asíncrona, lo que ocasionaría la pérdida de la

sincronía con la obtención de datos. Por esto, pantallas auxiliares como la de aceptación de la nueva resolución y el explorador de archivos se despliegan en la parte inferior de la pantalla evitando que se sobrepongan al panel de desplegado; inclusive con una resolución de 768 pixeles en la vertical, se llega a obstruir ligeramente la interfaz con el explorador de archivos; sin embargo, dicho evento no tiene mayor repercusión en el sistema, aunque se recomendaría una resolución superior. Cabe aclarar que únicamente el panel de desplegado tiene esta característica, por lo que el panel de control, por razones evidentes, tiene que repintarse cuando algún objeto se le ha sobrepuesto, cosa que interrumpe la operación normal del sistema, de tal forma que se recomienda no mover ni abrir aplicaciones distintas a las del sistema.

Cuando se recuperan gráficas desde archivos, sus valores se despliegan con los valores de ganancia, umbral y velocidad que se encuentren seleccionados al momento de abrir el archivo, dado que esta información no es almacenada al momento de crear el archivo. Esta situación puede originar que los comentarios recuperados de dicho archivo no concuerden con la ubicación del evento para el cual fueron generados (si es que el control de velocidad se encuentra en un valor distinto al del momento de crearse el archivo), ya que los comentarios guardan la posición horizontal en la cual fueron generados. No obstante, dado que la velocidad que se utiliza con más frecuencia es la de valor uno, es muy probable que no haya que ajustar este valor para que coincidan los eventos con los comentarios.

Primer propuesta del programa

Para cubrir el objetivo de desplegar la señal proveniente del ecosonda, se procedió a elaborar un programa en C que cubriría todos los requisitos antes mencionados.

Se cuenta con un programa de ejecución principal, que inicia todos los valores necesarios y manda llamar las distintas funciones, tales como, creación de la interfaz, conversión, desplegado de datos, modificación de los valores una vez que el usuario utiliza algún control, y trazo de la marca de la corredera.

En la parte principal del flujo del programa, primero se inician los valores de las opciones de segundos por barrido, ganancia, umbral y velocidad de avance del papel. Éstos son valores intermedios que permiten al usuario modificarlos hacia ambos extremos una vez que comienza la impresión de puntos.

Después comienza el ciclo principal. Éste se reinicia cada que existe el pulso externo proveniente de la grabadora. También dentro de este ciclo se hace un poleo (*polling*) de presión de teclas. Las teclas sirven para modificar las opciones y para la marca de evento.

En el intervalo comprendido entre dos pulsos, se realizan las conversiones, despliegue de resultados y colocación del enrejado, todo sincronizado por una variable que está en función de la base de tiempo elegida.

Cuando ocurre alguna presión del teclado, se interrumpe el ciclo principal hasta que se ajustan los valores a los deseados; se reinicia el ciclo con la llegada de un nuevo pulso de la grabadora. Esto mismo sucede cuando la tecla presionada es la que permite colocar la marca de algún evento.

La función de conversión se encarga del control de entradas y salidas del puerto de expansión. La señal analógica es convertida a 4 bits, por medio del convertidor analógico-

digital, y este resultado es leído por el puerto de datos. Este número es afectado posteriormente por los valores de ganancia y umbral deseados. Finalmente, cada conversión será un número decimal entre 0 y 15.

Este desarrollo comenzó a crear problemas y a mostrar ineficiencias en las siguientes cuestiones:

- Cuando ocurría una modificación de parámetros, a través del teclado, toda la ejecución se detiene. Dado que todo ocurre en el mismo hilo de ejecución y éste no continúa sino hasta que se terminaron de modificar los valores y se detecta otro pulso del ecosonda. Además, este hecho impide al usuario darse cuenta si sus modificaciones a las opciones son correctas, ya que tiene que hacer primero los cambios para después verlos reflejados en pantalla, es decir, no puede modificar y ver a la vez los cambios (tiempo real).

- Se tenía que validar la tecla presionada, dado que se pueden presionar otras teclas. Esto interrumpe momentáneamente el flujo del programa, sólo para determinar que la tecla era inválida y continuar la ejecución.

- La elaboración del enrejado se realizaba por cada barrido, cuestión que no es eficiente, por lo que en la siguiente versión primeramente se colocó el enrejado antes de comenzar a desplegar los valores.

- El manejo de la interfaz gráfica resultó algo hostil para el usuario, ya que tiene una pequeña ventana donde visualiza los valores de los parámetros.

Segunda aproximación del programa

Dadas las experiencias y necesidades detectadas a partir de la primer versión del programa en C, se procedió a realizar una segunda versión mucho más versátil y funcional realizada en Java, la interfaz de esta versión se muestra en la figura 15.

En un principio, este lenguaje era totalmente desconocido, e incluso se planteó la posibilidad de otros lenguajes tales como Visual Basic y C++. Las opciones anteriores fueron descartadas por las siguientes razones: Visual Basic es un lenguaje relativamente sencillo en cuanto a la elaboración y estructuración de su código, no obstante es un lenguaje a la vez limitado. Su lenguaje base es Basic, cuya sintaxis difiere de la de C, por lo que implicaba revisar dicha sintaxis y aprender a manejar las herramientas de creación en el entorno visual. La opción de C++ también fue considerada, pero su curva de aprendizaje es lenta. Por otro lado, Java es similar en algunos conceptos a C++, pero su aprendizaje es mucho más sencillo y rápido, ya que los conceptos de apuntadores, estructuras, asignación de memoria, entre otros, son transparentes al programador de Java.

Esta segunda versión permitía un enfoque mucho más sencillo, orientado a objetos, del sistema, por lo que se facilitó identificar los objetos necesarios para el desarrollo del sistema, sobre todo con la experiencia de la primera versión.

No obstante, esta aproximación incluye un programa en lenguaje C, que es el encargado de la comunicación a bajo nivel con la interfaz.

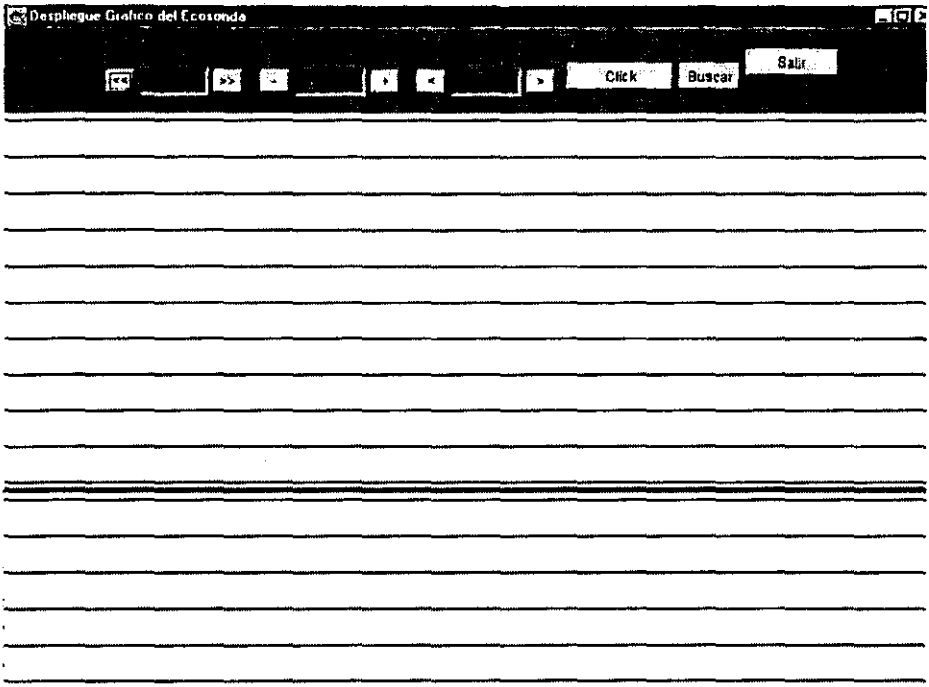

Fig. 15. Primera interfaz gráfica funcional

Esta aproximación a su vez tuvo distintas versiones, por lo que en lo siguiente de este documento el primer programa en C será señalado como la primer aproximación y el programa en Java será la segunda aproximación, desprendiéndose de ésta última las versiones que se comentarán en la descripción siguiente.

Descripción del código en C de la segunda aproximación

Este programa es el encargado de leer y escribir a la tarjeta de expansión del puerto paralelo. La tarjeta tiene asignada la dirección de memoria 0x31F; este programa hace uso de esta dirección para ambas operaciones. La clase **Conversion** es la encargada de ejecutar dicho programa, que al invocarlo, envía como argumentos de la función **main()**, del programa en C, la cota superior e inferior que se están desplegando. Por ejemplo, cuando se está manejando la pantalla completa, se envían los valores máximo y mínimo del panel. Al efectuar un cambio en la resolución, se envían los nuevos valores. Otro parámetro que también se envía es un valor en relación con la base de tiempo que se está utilizando.

El código primeramente espera el pulso proveniente de la grabadora, para garantizar la sincronía; una vez que lo recibe, comienza a leer datos hasta juntar el máximo de datos permisibles en un renglón; entonces termina su ejecución y regresa el control al código Java. También sirve de monitor constante para la señal proveniente de la corredera, ya que

cuando ocurre ésta, se incorpora a los datos un valor que permite identificar, al momento del despliegue, que es necesario poner una línea vertical.

Primera versión

Fue la primera versión con la que se comenzaron las pruebas en el barco. Una vez que se tenían la grabadora y el ecosonda físicamente, se procedió a probar el programa y a comenzar las mejoras del mismo. En el periodo del 14 de marzo al 24 de marzo de 1999 se realizaron 7 versiones, que fueron las creadas durante la campaña oceanográfica. Posteriormente se creó una última versión con mejoras adicionales determinadas por la experiencia en el barco.

Primeramente la pantalla se encuentra dividida en tres paneles, el superior es donde se encuentran los controles de la interfaz; estos sustituyen casi la totalidad de las funciones necesarias para el despliegue de los puntos en la grabadora actual. El segundo panel es la parte principal del despliegue de datos, es decir, aquí es donde se despliegan los puntos correspondientes a los valores tomados de la tarjeta. En el tercer panel, o panel inferior, se pueden ver gráficas pasadas, esto es, el usuario puede seleccionar un archivo de datos de alguna otra fecha u hora y éste será desplegado en dicho panel inferior.

La clase **Conversion** es un hilo de ejecución independiente; es la encargada de comunicarse con la tarjeta de expansión. Esta comunicación se realiza a través del programa en C llamado **adcint.c**, por lo que la clase manda llamar a su archivo ejecutable. Este archivo en C envía la serie de pulsos necesarios para que el convertidor capture la señal analógica por una de sus entradas, la convierta y finalmente la entregue nuevamente a este programa. Este programa durante su ejecución envía a la clase **Conversion** un resultado y finalmente envía un valor de retorno para indicar si su terminación fue correcta o incorrecta.

En esta primer versión la clase **Conversion** ejecuta el código de C para un solo dato, no obstante esto hace demasiado lenta la ejecución del código a tal grado que la mayoría de las bases de tiempo no pueden ser satisfechas. En versiones posteriores, se determinó que el programa en C debería ir capturando los datos hasta que se llenara un renglón, esto es, que el programa comience el control del ADC, obtenga los datos necesarios para llenar un renglón y termine su ejecución. Se decidió que un renglón sea el límite de datos que regresa el programa en C, ya que al aumentar esta cantidad de datos, el despliegue en la pantalla se demoraría demasiado, cuestión que podría resultar inadecuada para el investigador.

La clase **Pulso**, que después de la segunda versión fue eliminada, es un hilo independiente que se encarga de llamar a un programa en C: **pulso.c**. Este último únicamente sirve de monitor del pulso del ecosonda e informa al resto del sistema de este evento; no obstante, resulta demasiada carga para el procesador el manejo de un hilo más. Por otro lado, esta función puede ser hecha desde el programa **adcint.c** encargado de la conversión de datos.

Segunda versión

En esta versión las principales mejoras fueron en cuanto a la velocidad de conversión. Se elimina el uso de la clase **Pulso** y por consiguiente del programa en C **pulso.c**, ya que el

manejo de tantos hilos hacía muy lento al sistema; además, el desplegado de puntos se llevaba a cabo en intervalos poco periódicos. Conviene hacer notar que no hay un monitor del pulso en esta versión. Se hicieron pruebas con otro programa en C considerando la posibilidad de un monitor del pulso haciendo uso del puerto paralelo, junto con algunos circuitos de apoyo.

Otra de las mejoras fue en cuanto al uso más eficiente del código para el despliegue de la gráfica, ya que anteriormente se empleaban un par de métodos, uno para el desplegado en la parte inferior y otro para la parte superior; este par de métodos se sustituyeron por uno solo el cual recibe como parámetro extra un tipo primitivo **boolean** que le indica si los datos enviados serán desplegados arriba (true) o abajo (false). Por lo anterior, en esta versión todavía se mantiene el desplegado de dos paneles.

Dado que en esta versión se tuvieron problemas con la señal proveniente del ecosonda, todavía no fue posible ver el funcionamiento pleno del programa ni posibles fallas en su implementación.

Tercera versión

Esta fue la primera versión de la cual se obtuvieron datos, tanto en archivos como en pantalla, ver figura 16. Estos datos no fueron tomados propiamente del ecosonda, sino que el convertidor únicamente fue conectado a un voltaje constante. Se utilizó el pulso como señal entrante para la obtención de estos datos. Entre las principales mejoras, a partir de la observación de su funcionamiento, se tuvieron las siguientes:

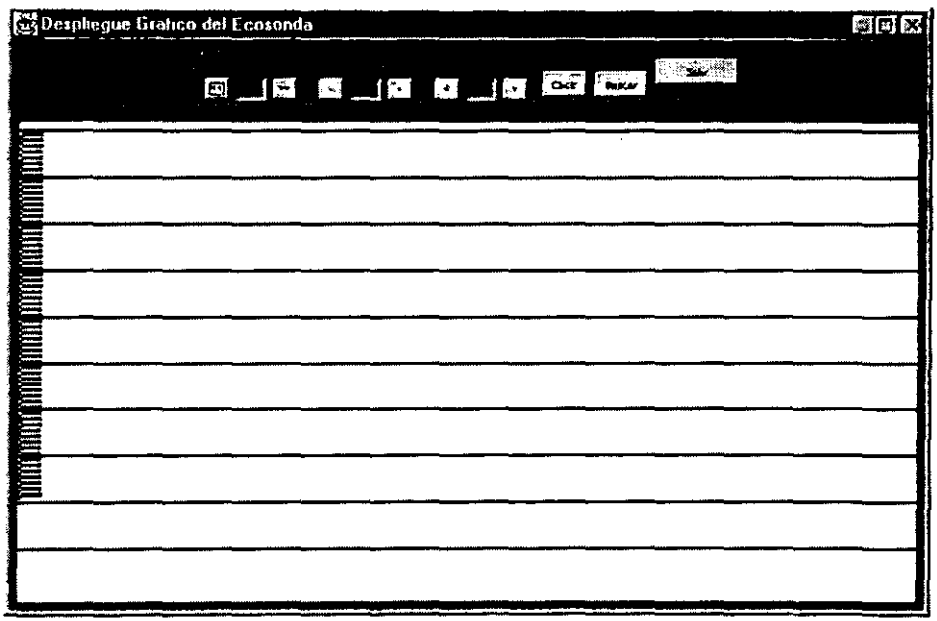

Fig. 16. Interfaz de la tercera versión

a) El manejo del pulso proveniente del ecosonda se implementó nuevamente en esta versión, sólo que ahora dicho manejo se hizo en el programa **adcint.c** que es llamado por medio de la clase **Conversión** y no a través de otro hilo como sucedía en la primera versión.

b) Se detectó una sobrecarga de memoria, debido a un dato de tipo Vector en la clase **Conversion** que crecía indebidamente. Se trató de simplificar al máximo la clase **Conversion**, principalmente con la mejora del método **runCommand**. En el evento del botón "Buscar" de la interfaz, se realizaban algunas acciones antes de suspender el hilo de conversión; esto hacia muy lento al sistema, por lo que primero se suspendió el hilo de conversión antes de ejecutar las acciones correspondientes a la búsqueda y recuperación de archivos.

c) Hay que mencionar que en esta versión se pudo ver que el sistema consume demasiados recursos de la computadora, ya que se notan interrupciones durante el desplegado en curso. Para los fines del desplegado esto no interfiere demasiado con la gráfica plasmada en el panel. Cuando se llena el panel superior, se hace todo el manejo para el desplegado de esa gráfica en la parte inferior y también para el guardado de dichos datos en un archivo. Durante este tiempo se pierden algunos datos, que para fines prácticos, es imperceptible el número de renglones que se dejan de desplegar; no obstante se optó en el diseño por desaparecer el panel inferior y ganar con ello mayor tiempo del sistema dedicado a la conversión de datos.

d) También se pudo comprobar el éxito en cuanto a la reducción del tamaño de los archivos generados. Por ejemplo, un archivo de una pantalla que ocupa, sin comprimir, aproximadamente 13,000 bytes, queda reducido a uno de 300 bytes aproximadamente.

e) Se mejoró un poco también el tamaño de los botones desplegados, reduciéndolos, de manera que el panel superior fuera más pequeño.

Cuarta versión

En esta versión se dirigen los datos a un directorio denominado datos, es decir, cada archivo creado a partir de la información de la gráfica se almacena con su nombre correspondiente y en la ruta especificada en la clase **ManejoArchivo**.

Quinta versión

En esta versión se despliegan un mayor número de puntos, pero dada la resolución del monitor utilizado en el crucero, se tuvo que reducir de dos pantallas a una sola. Con este cambio se pueden desplegar alrededor de 320 puntos. Cabe hacer notar que la funcionalidad del botón de búsqueda de archivos no se modificó, aún sabiendo que presionar este botón llena la pantalla inferior con el contenido de un archivo; únicamente se evitó utilizar el control durante esta etapa de pruebas.

También aquí se observó que con las distintas bases de tiempo se perdía la imagen en pantalla, dado que la señal analógica proveniente del ecosonda varía en amplitud dependiendo de la escala seleccionada, por lo que se necesitaba un ajuste previo de la señal. Este ajuste tiene que ser variable dependiendo de la base de tiempo elegida, y este tipo de circuito no se tenía contemplado.

Sexta versión

Con el fin de observar el comportamiento del sistema utilizando distintas bases de tiempo, se descartó la utilización del convertidor de la tarjeta de expansión. Se decidió implementar esta última versión durante el crucero. Esta versión toma los datos digitales directamente de la grabadora. Los datos son tomados vía puerto paralelo LPT1 con ayuda de algunos circuitos básicos. El programa **adcint.c** se encarga de tomar el mayor número de puntos entre pulsos del ecosonda; después se toman 320 de estos puntos, que es la resolución más alta manejada.

Propuesta final del programa

Dado que al inicio de este trabajo no se tenían conocimientos de la programación orientada a objetos, se fueron realizando prototipos que fueron mejorandose con el tiempo, conservando algunas de las ideas y códigos iniciales, pero cambiando grandemente en otros aspectos. A continuación se muestra una explicación de la última versión, después de haber pasado por varias versiones, y una última etapa de optimización, que fue cuando ya se tenía un manejo considerable de Java.

Algunas nomenclaturas que se siguieron son las siguientes: las clases mostradas con la fuente Arial 10 negritas, por ejemplo, **Sonda** son propias del sistema creado; el resto de clases mencionadas forman parte de la versión estándar de Java JDK (Java Development Kit). Los métodos se muestran con la fuente Times 10 negritas, por ejemplo, **getX()**; en algunos se destaca el argumento o argumentos que reciben, si fuese importante señalarlos. Finalmente, los objetos se señalan con la fuente Arial 10 (sonda).

La parte fundamental del sistema consiste en cuatro clases de Java. En primera instancia, se cuenta con la clase **Sonda** que es la que se encarga de iniciar los objetos necesarios para el funcionamiento del sistema. Es una aplicación (en inglés denominada como *Stand alone*) que tiene un método **main** encargado de iniciar todo el sistema. Dichos objetos son instancias de algunas de las clases que incluye el sistema. Una de éstas es la clase **Interfaz** que despliega la interfaz gráfica y realiza el control de todos los botones. **Interfaz** es la parte fundamental del hilo principal de ejecución, tiene una herencia de la clase Frame. Otra instancia es la clase encargada de conectarse con el programa en C que obtiene los datos, recibe el nombre de **Conversion**, esta tiene una herencia directa[37] de la clase Thread, por lo que tiene las características de un hilo de ejecución encargado de la conversión.

[37] La herencia directa se logra utilizando la palabra clave **extends**, por lo que se dice que una clase *extiende* de otra cuando se refiere a una herencia de este tipo.

La clase Interfaz

La clase **Interfaz** es la encargada de crear los dos paneles ya descritos anteriormente, con todos los controles necesarios. Dentro de esta clase se crea una instancia de un objeto del tipo **Lienzo**, que es lienzoSuperior[38], que se encarga de desplegar los valores dentro de sí mismo. **Lienzo** es el otro hilo de ejecución, extiende de un componente gráfico Canvas, de igual forma implementa[39] la interfaz Runnable.

Como ya se comentó los controles fueron aumentando conforme se fue avanzando y depurando el sistema. Estos controles son los de Ganancia, Umbral, Velocidad, Salir, Marca externa, Búsqueda y apertura de archivos, control de impresión y guardado y cambio de la base de tiempo.

En **Interfaz** se tienen estos controles, así como la funcionalidad cuándo dichos controles son activados. Se aumentan o decrementan las variables encargadas de la ganancia, umbral y velocidad, en tiempo real, dentro de su rango permisible: la ganancia tiene un rango de valores de 0 a 15, el umbral tiene un rango porcentual del 0 al 100%, con incrementos/decrementos del 10%, y finalmente la velocidad toma valores entre 0 y 5.

El control de marca externa únicamente manda llamar al método **marcaExterna** del objeto lienzoSuperior, el cual dibuja una línea vertical negra; se detienen tanto el hilo de conversión como el de despliegue, para agilizar la operación.

Al ser accionado el control de abrir archivos, 'Buscar', primeramente detiene ambos hilos, después crea una instancia de **MiFileDia** que se encarga de abrir el sistema de archivos, en el directorio donde se encuentran los archivos comprimidos. Una vez que el usuario selecciona el archivo adecuado, a través de una instancia de **DriverDes**, se descomprime el archivo y se recupera el vector de valores que es pasado como argumento al método **grafica** del objeto lienzoSuperior. Una vez que termina la ejecución de este método, el sistema se detiene con el fin de permitir el análisis del despliegue recuperado. Para reanudar el despliegue en tiempo real, el usuario debe presionar el mismo botón, de búsqueda, sólo que éste ahora tendrá la etiqueta de 'Continuar'. En caso de que el usuario presione la opción de buscar archivos, pero decida cancelar dicha acción, usando el botón 'Cancelar' de la ventana de búsqueda de archivos, de todas formas tendrá que reactivar la conversión en tiempo real, presionando el mismo botón de 'Continuar'.

El control de Cambio de la base de tiempo es una lista desplegable (en inglés *Drop-Down list* o *Choice*), que muestra las posibles bases de tiempo a seleccionar. Primeramente, en cuanto se presiona el botón de la lista desplegable se suspenden los hilos de ejecución; una vez que una de estas bases es seleccionada, se cambia el atributo 'tiempo' de la clase **Sinc**, de tal forma que el hilo de conversión se vea afectado por dicho cambio y tome un nuevo intervalo para la obtención de los valores; posteriormente se reinician los hilos. Es importante hacer notar que sólo la primera vez que se presiona el botón de la lista se acciona un evento, por lo que es necesario que una vez que el usuario accione dicho botón, tenga que seleccionar algún elemento de la misma (lo que desencadena otro evento), ya que de lo contrario no continúa la conversión en tiempo real.

---

[38] Este nombre lo toma desde la segunda aproximación, donde efectivamente existían tanto un lienzo inferior, como uno superior.

[39] Se utilizará el término *implementa* para denotar el mecanismo de herencia indirecta, es decir, cuando una clase utiliza la palabra clave *implements* y adquiere las características de una **interfaz**; distinto a cuando una clase extiende de otra.

El control de impresión se lleva a cabo en el área del panel de control donde se encuentra la etiqueta 'Imprimir' y el botón que cambia su etiqueta de 'Activado' a 'Desactivado', esto es, cuando se presiona el botón éste cambia su estado; por ejemplo si la impresión está desactivada, la etiqueta 'Desactivado' estará presente en el botón, y para activar esta funcionalidad se deberá presionar dicho botón, lo que inmediatamente hará que cambie la etiqueta a 'Activado'. Para esta función se manejan un par de banderas (tipos de datos boolean) denominadas *imprimir* y *guardado*, de tal forma que cuando está desactivada la impresión la bandera de *imprimir* está en 'false' y la de *guardado* en 'true', y viceversa, cuando está activada la impresión, la bandera *imprimir* estará en 'true' y la de *guardado* en 'false'. Estas banderas son tomadas por la clase **Lienzo** para permitir el guardado o la impresión de los datos, una vez que se llenó el panel. Por el momento se manejan estas banderas como excluyentes, es decir, si guardan en disco los datos, estos no se imprimen, y viceversa. Por último esta clase cuenta con el método **prepara** el cual prepara el trabajo de impresión, teniendo involucrado un objeto de la clase **PrintPanel** que es la que se encarga de efectuar la impresión principalmente a través de su método **print**. Este método es llamado por la clase **Lienzo** cuando se ha llenado el panel.

La clase Lienzo

La clase encargada del desplegado de los valores provenientes del ecosonda fue denominada como **Lienzo**. Esta clase es en sí la generadora de los objetos en los cuales se pueden desplegar los valores; esta clase extiende de Canvas, que es una clase de Java que permite dibujar elementos con completa libertad. Tiene el método **run()**, el cual es heredado de la clase Thread, que permite que la máquina virtual de Java lo reconozca como hilo de ejecución. Este método permite desplegar los puntos de acuerdo al Vector de valores (de tipo Integer) tomado de la clase **Sinc**. Este vector es actualizado por la clase **Conversion**.

Para el despliegue de los valores, se toman de uno en uno, y se ajustan de acuerdo a los valores de Ganancia, Umbral y Velocidad vigentes; el número que resulta de este ajuste se remplaza por su color correspondiente, utilizando el método **generaColor**, y finalmente se despliega en las coordenadas correspondientes.

El método **run** lleva el incremento de la coordenada vertical, y una vez que el barrido del Vector llega a su punto final, se invoca al método **controlaGrafica**, el cual ajusta las coordenadas para iniciar un nuevo barrido. También detiene los hilos y manda a ejecutar el guardado de la pantalla o la impresión de la misma. Este método también identifica cuándo se llena el panel, en cuyo caso se crea una instancia de la clase **DriverManArch**. Esta toma todos los valores de dicho panel y los almacena en un archivo, el cual a su vez es comprimido y guardado en el directorio asignado.

También es ésta la clase que se encarga de colocar los márgenes, así como de controlar el largo y ancho de los puntos; también maneja el enrejado del lienzo, es decir, coloca las diez líneas divisorias a lo largo del lienzo; esto es realizado con el método **creaRejas**, además en el método **run** se tiene una pequeña rutina que evita que los datos se desplieguen sobre el enrejado.

Esta clase cuenta además con el método **grafica(Vector v,boolean superior)** el cual es recibe un vector de datos recuperado desde un archivo; este método también hace uso de **controlaGrafica** para el avance de coordenadas, y una vez que se llega al final del vector, únicamente se elimina este y se concluye el llamado al método.

La clase **Lienzo** es responsable de escuchar los eventos del mouse que ocurren dentro de sí, es decir, cuando el usuario ejecuta acciones con el mouse dentro del panel. Esto desencadena eventos que esta clase atiende; estos están orientados a cubrir la función de 'Zoom' o de aumento de resolución. Cuando el usuario presiona el mouse y sin soltarlo lo arrastra diagonalmente, se forma un rectángulo entre la coordenada donde fue presionado y la coordenada donde es liberado; este rectángulo será la base para tomar la nueva resolución; una vez liberado el mouse aparecerá una ventana que consulta si se desea proceder al cambio de resolución, ver figura 17. Esencialmente, la parte importante del rectángulo es su alto, es decir sus coordenadas verticales. Estas permiten que el programa **adcint.c** realice conversiones únicamente dentro de ese rango. Por ejemplo, si una resolución normal presenta 320 puntos para cubrir toda la vertical del panel de despliegue, cuando se realiza un Zoom, el usuario traza con el mouse una diagonal; supongamos que obtenemos un rectángulo con coordenada vertical superior de 120, por ejemplo, y de coordenada vertical inferior de 180; entonces, se realizarán 320 conversiones entre dichas coordenadas y se desplegarán en todo el panel. Por lo tanto, la zona seleccionada será ampliada hasta que se presione el botón de restaurar, lo que regresa la conversión a la predeterminada; un ejemplo de cambio de resolución se muestra en la figura 18.

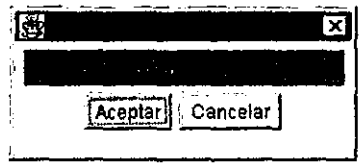

Fig. 17. Ventana previa a cambiar la resolución

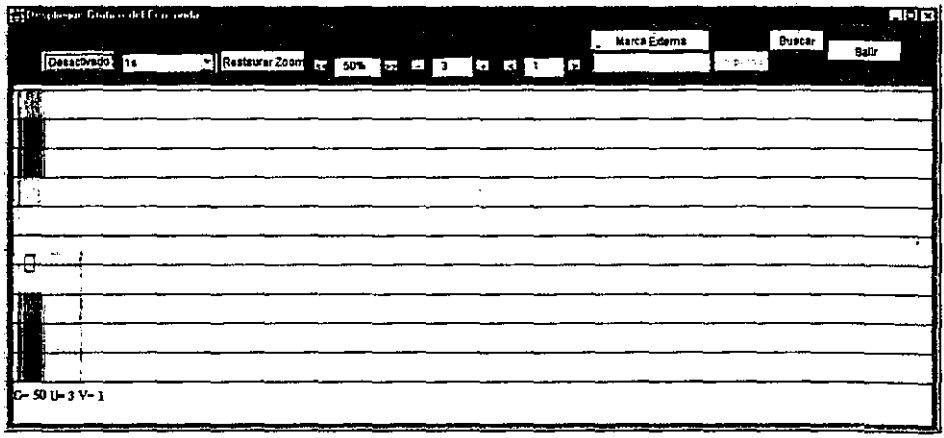

Fig. 18. Ejemplo de cambio de resolución

La clase Conversión

La clase **Conversion** es un hilo de ejecución independiente, que se encarga de obtener los datos del exterior. La obtención de los datos en un principio, se manejo utilizando la tarjeta de expansión con el ADC0809, con el fin de tomar la señal analógica proveniente del ecosonda, y evitar el uso de la grabadora en ese aspecto. La segunda forma de obtención de datos es a través de la grabadora y sus salidas digitales. Estas entregan la representación binaria en cuatro dígitos de cada una de los datos por desplegar. Finalmente, la clase **Conversion** tiene una simulación de datos, que no es otra cosa que un método que obtiene datos aleatorios. Esta última es la que se describe a continuación.

Esta clase extiende de Thread, por lo que cuenta con su método **run**; en dicho método, se encuentra la rutina de obtención de datos, que no es otra cosa que obtener datos aleatorios espaciados por un cierto intervalo de tiempo. Estos datos se van insertando en un Vector; una vez que se alcanzan los 320 datos, se ingresa un dato extra, un 200, que indica el retorno de línea. Cabe hacer notar que la clase Vector sólo permite que le sean introducidos objetos y no tipos de datos primitivos; entonces, antes de ser introducido un dato entero, se le hace una conversión a tu tipo de dato equivalente en objeto, que es la clase Integer (la clase **Lienzo** realiza el proceso inverso al recuperar los datos del vector). Una vez que dicho vector se encuentra lleno, es enviado a la clase **Sinc**, a través de su método **grafica(Vector valores)**, y se encienden un par de banderas en la misma clase **Sinc**, *graficando* y *vectorListo*. Posteriormente, se entra a un ciclo que verifica la primera bandera, mientras ésta continúe en *true*, esto indica que la clase **Lienzo** sigue haciendo uso del vector enviado. Mientras que la segunda bandera si está encendida, permite que la clase **Lienzo**, de manera análoga, salga de su ciclo monitor, y tome el vector recién depositado en **Sinc**. Una vez que **Lienzo** toma el vector, entonces apaga la variable *graficando*, por lo que mientras se despliegan los valores en pantalla, se están convirtiendo otro conjunto de datos. Este par de variables son las que se encargan de tener el control de la sincronía entre los hilos de conversión y de despliegue. Cuando ambos ciclos son detenidos por el hilo principal, no se pierde la sincronía una vez que son reanudados, dado que estas variables no cambian su estado. Es por esto que reside una instancia de la clase **Sinc** tanto en **Conversion** como en **Lienzo**.

Cabe señalar que antes de utilizar el par de banderas descritas anteriormente, se intentó hacer uso de un par de técnicas propias del manejo de hilos de ejecución que tiene Java. La primera con el modelo de espera y notificación, mediante métodos propios de la clase Object; la segunda técnica con los métodos de la clase Thread: **suspend**, **resume** y **yield**; no obstante, tras varias pruebas, estas técnicas no dieron resultado.

Clases encargadas del manejo de archivos

La clase **DriverDes** es la clase encargada de manipular y controlar todas las clases concernientes al proceso de descompresión de archivos. Este proceso se lleva a cabo cuando el usuario ha seleccionado un archivo para ser desplegado en pantalla.

Esta clase tiene el método **rutina(String nombreArchivo)**, que recibe el nombre del archivo en cuestión y regresa un vector con los datos contenidos en ese archivo. El proceso se hace de la forma siguiente: esta clase crea un objeto de la clase **DescompresorArchi**, para ejecutar el método **descomprime(String nombreArchivoZip)**, este método crea un archivo del

mismo nombre, pero sin extensión zip y regresa una cadena con el nombre de dicho archivo. Posteriormente, a través de una instancia de la clase **ArchivoVector** y con el nombre del archivo pasado como parámetro, se ejecuta el método **guardarEnVector(String nombreArchivo)** para obtener el Vector de valores.

La clase **DriverManArch** es la clase encargada del control y manipulación de las clases concernientes al proceso de compresión de archivos. Como ya se explicó anteriormente, la compresión de archivos se lleva a cabo cuando el panel se llena completamente de datos y es necesario que sea borrado para seguir desplegando datos. Entonces, cuando se detecta esta condición en la clase **Lienzo**, se manda a llamar al método, denominado como **metodo**, a través de una instancia de la clase **DriverManArch**; el argumento de dicho método es un vector que contiene todos los valores de esa pantalla. Dentro del método de esta clase se utiliza un objeto del tipo **ManejoArchivo**, para llamar al método **guardarArchivo(Vector vector)** que se encarga de almacenar toda la información del Vector en un archivo. El nombre de este archivo es tomado de una clase generadora de nombres denominada **NombreDeArchivos**; el resultado de ejecutar el método **guardarArchivo** es la generación de un archivo con los datos de la pantalla y el retorno de dicho método es el nombre del archivo; este nombre es pasado como argumento a la clase **ComprimeArchi**, la que finalmente comprime y almacena el archivo en el directorio de almacenamiento.

La clase **ComprimeArchi** es la encargada de la compresión de los datos, y su método es denominado **comprime**; este recibe el nombre del archivo que contiene los datos, abre un flujo de lectura hacia dicho archivo, y a su vez crea un flujo de escritura a un nuevo archivo con el mismo nombre pero con extensión zip; este flujo de escritura además de enviar los datos a su nuevo destino también los comprime. Además de los datos, primeramente es pasado a ese archivo comprimido el nombre original del archivo sin comprimir.

La clase **DescompresorArchi** es la encargada de la descompresión de los datos, y su método es denominado descomprime; éste recibe el nombre del archivo con extensión zip, abre un flujo de lectura desde ese archivo, extrae los datos y los pasa al archivo sin extensión zip. El nombre del archivo es tomado como el primer elemento recuperado de dicho archivo. Finalmente, este método regresa la cadena con el nombre del archivo.

Clase **ArchivoVector**. Esta clase es la encargada de recuperar los datos desde un archivo. Cuando el usuario oprime el botón de Buscar y después selecciona un archivo, este nombre del archivo es pasado al método **guardarEnVector** de esta clase, la cual se encarga de abrir un flujo de lectura hacia ese archivo. Todos los datos son cargados en un Vector, el cual es regresado por el método. De tal forma, la clase que lo ejecuta obtiene todos los datos correspondientes a la pantalla desplegada en esa fecha.

La clase **MiFileDia**, a través del método **selección**, es la encargada de crear una instancia del componente FileDialog de Java. FileDialog muestra el sistema de archivos de la computadora en donde se encuentra corriendo el sistema, en específico, se abre el directorio destinado para el almacenamiento de los archivos de datos, ver figura 19. Una vez que el usuario selecciona un archivo, el nombre de éste es enviado como valor de regreso del método **seleccion**.

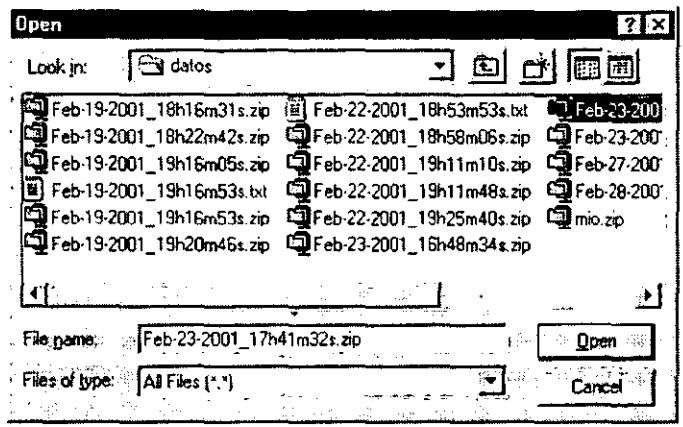

Fig. 19. Ventana de selección del archivo a desplegar

La clase Sinc

La clase **Sinc**, cuenta con tres atributos importantes, un vector, que es el medio por el cual se transmiten los datos de la clase **Conversión** a la clase **Lienzo**, y dos valores que mantienen la sincronización, que son 'graficando' y 'vectorListo'; ambos atributos son de tipo boolean, y son estáticos, es decir, de clase, de tal forma que aunque exista una instancia de esta clase, tanto en **Conversión** como en **Lienzo** cuando alguna de estas modifica uno de estos atributos, la otra clase verá reflejados los cambios. Para acceder a estos dos últimos atributos, se tienen los metodos **setXXX()** y **getXXX()**, donde 'XXX' es el nombre del atributo de los antes mencionados; el primero se encarga de establecer un nuevo valor al atributo y el segundo regresa dicho atributo. Estos métodos están declarados como públicos y sincronizados (synchronized) de tal forma que cuando se intenta acceder a uno de los atributos y éste está siendo utilizado por otro método, dicha llamada espera a que termine el primer proceso para después poder acceder al atributo.

Para el vector, los métodos que lo modifican u obtienen, para dar una mayor idea, se denominan **grafica(Vector v)** y **convierte()**, siendo el primero el llamado por la clase **Conversión** y el segundo por la clase **Lienzo**. Éste último regresa el vector con los valores a desplegar.

Esta clase, también se encarga de manejar el valor que toma la base de tiempo. La clase **Lienzo** es capaz de modificarlo y la clase **Conversión** constantemente está consultando este valor para conocer la tasa de conversión. Los métodos encargados de modificar este atributo de tiempo son **setTiempo** y **getTiempo**. Cabe aclarar que el tiempo se maneja con dos atributos, ya que en la conversión se maneja como un entero, pero en la interfaz se recoge una cadena de caracteres. Evidentemente estos dos métodos también están sincronizados.

La clase PrintPanel

La clase **PrintPanel**, es la clase fundamental para el manejo de la impresión. Extiende del componente gráfico Canvas e implementa la interfaz Printable de Java. La primer clase le permite sobreescribir métodos como **paint** y **update**, además permite que sea incorporada en un segundo panel inferior de la clase "Interfaz", el cual no es visible pero es fundamental agregarlo para la impresión. La interfaz Printable permite sobreescribir el método **print(Graphics g, PageFormat pf, int page)** que es indirectamente llamado desde **Interfaz** cuando se desea imprimir. El método **prepara** de la clase **Interfaz**, a través de una instancia de **PrintJob**, indica al mecanismo de impresión cuál es el componente a imprimir y el formato que llevará la página. Al ejecutarse el proceso de impresión, es llamado el controlador de la impresora, por lo que aparecerá la pantalla de impresión de éste, la cual permite controlar el número de copias, la posición del papel y demás propiedades de la impresora; ver figura 20.

Esta clase también tiene métodos casi idénticos a los que tiene **Lienzo** para desplegar los valores, tales como **grafica, controlaGrafica** y **creaRejas**, con la diferencia de que se hace uso de un objeto de tipo Graphics2D; ésta clase es más sofisticada que la clase Graphics que se maneja en la clase **Lienzo**. Se menciona esta clase como más sofisticada ya que es incorporada, al JDK, después que la segunda y sus métodos están inclinados a la filosofía de orientación a objetos. Sin embargo, su uso básico no es distinto al utilizado. Desafortunadamente, el mecanismo de impresión necesariamente hace uso de un objeto de este tipo y no del tipo Graphics que se emplea en el resto del desarrollo.

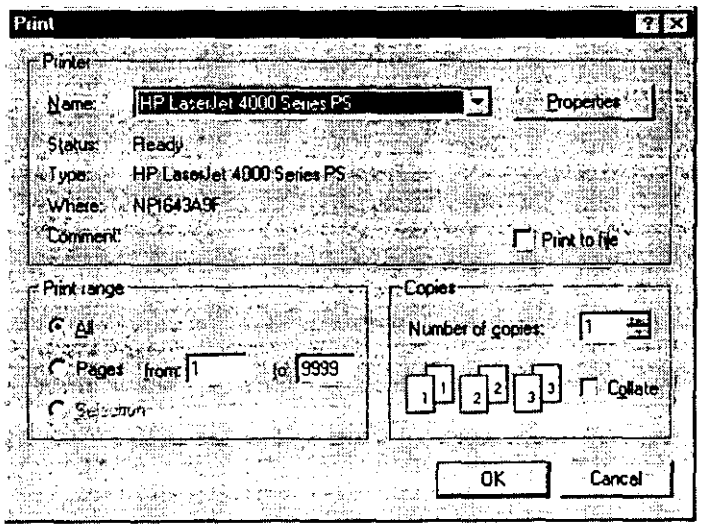

Fig. 20. El manejo de impresión hace uso de las facilidades de impresión del sistema operativo

Como se puede ver en la figura anterior, el controlador de impresión depende principalmente del modelo de impresora configurada en la computadora personal, por lo

que dicho controlador de impresión presentará mayor o menor acoplamiento con la función implementada en Java.

Clases auxiliares

La clase **Boton** es una clase interna de la clase **Interfaz**. Esta clase extiende del componente gráfico Button, y lo único que hace es sobreescribir el método **getPreferredSize()** con el fin de controlar el tamaño del componente. En esencia, se despliega un botón ligeramente más pequeño que una instancia un objeto de tipo Button.

De igual forma a la anterior, la clase **Pantalla** es una clase interna que extiende de Panel con el único fin de hacer un panel de un tamaño deseado. Dicho tamaño se controla a través del constructor que recibe un par de enteros que son enviados como parámetros al método **getPreferredSize()**.

La clase interna **MioPanel** también extiende de Panel, pero la finalidad de ésta es definir un acomodo o 'Layout' de sus componentes internos en una columna y dos renglones; este acomodo es utilizado en la distribución de los controles de umbral, ganancia, velocidad, marca externa, apertura de archivos y el control para salir del programa.

RESULTADOS

Tarjeta con el convertidor ADC

Fue de gran utilidad emplear la primera tarjeta de expansión del puerto paralelo durante el crucero. Se pudo tener una visión más clara del sistema actual, es decir, el tipo de señales que maneja la grabadora. Con la señal analógica se tuvieron malas experiencias, ya que aunque se lograron obtener resultados, estos fueron muy limitados, dada la naturaleza de la señal, la cual era bipolar y de magnitud y frecuencia variables, lo que hacía muy complicada su conversión. En resumen, se lograron obtener un par de gráficas a una base de tiempo determinada; al momento de cambiar la base de tiempo los resultados fueron desfavorables.

Utilización del puerto paralelo de la PC

Esta segunda prueba tuvo un éxito mayor. La parte de la conversión de la señal se dejó completamente a la grabadora, mientras que el sistema únicamente se ocupaba de tomar los datos digitalizados y desplegarlos. Fue posible desplegar gráficas a distintas bases de tiempo. Estas pruebas permitieron identificar las fallas y requerimientos del programa. Muchas de las mejoras en la parte de optimización se hicieron con base en estos resultados. Como ya se comentó, esta obtención de datos no es factible en el sistema final, ya que es necesario disponer del puerto paralelo para el manejo de la impresión.

Tarjeta final

Esta tarjeta de expansión de puerto paralelo, satisface la necesidad de dejar disponible el puerto de la impresora y contiene todos los circuitos necesarios para la captura de los datos, así como las señales del pulso de la grabadora y de la corredera. Esta tarjeta no ha sido probada con la grabadora, dado que su creación fue posterior a la experiencia del crucero; no obstante todas las necesidades encontradas en las pruebas iniciales han sido satisfechas en esta tarjeta.

Los lenguajes de programación

Se decidió desarrollar el sistema con una mezcla de lenguaje Java con lenguaje C, aprovechando las ventajas de ambos. La interacción se encuentra en una clase de Java que manda ejecutar el código previamente compilado y hecho ejecutable con C. Éste último se encarga de gobernar la interfaz con el ecosonda, obtiene los datos que son enviados al programa en Java, y después termina su ejecución. Con las técnicas antes mencionadas se pretende reducir al máximo las limitaciones que tiene Java.

Programa inicial

Al principio del crucero, se tenía un programa experimental que tenía contempladas las necesidades aparentes, identificadas del hecho de estudiar los documentos de la grabadora. No obstante, tenía por definir muchas otras necesidades y demás cuestiones, que no se podían determinar sino hasta el momento de estar en contacto físico con el aparato. Desde un principio este programa cumplía con su objetivo principal, que era desplegar en pantalla una señal similar a la desplegada en la grabadora; no obstante hacían falta muchas mejoras para trazar eficientemente dichas gráficas.

Programa final

Se puede considerar que con este trabajo se llegó a un programa que cubre con los objetivos establecidos, manejando casi todas las bases de tiempo del sistema actual, gracias a su manejo de hilos de ejecución y a su interacción con la parte física. Permite el almacenamiento de resultados, tanto en papel como en un medio magnético. Se eliminan de la grabadora los controles no utilizados, y se agregan controles que tienen funciones que hacen que el sistema sea fácil de usar y que sea visto como deseable para sustituir el grabado en papel. Dado que en ocasiones, aunque un sistema sea muy bueno si no presenta ventajas evidentes, el usuario final puede decidir fácilmente no utilizarlo, uno de los objetivos de este trabajo fue precisamente entregar un desarrollo competitivo con el actual, y que satisfaga necesidades no cubiertas.

CONCLUSIONES

Con base en los resultados obtenidos, se puede decir que los objetivos quedan casi en su totalidad cubiertos, lo que hace pensar que este sistema bien puede sustituir a la grabadora actual. Existen algunos puntos sobre los que se podría trabajar, y que permitirían cumplir plenamente con los objetivos establecidos. A continuación se describen algunos de los puntos a implementar o mejorar.

Durante el desarrollo y pruebas de este sistema se pudo ver que el sistema actual tiene características difíciles de igualar, pero a la vez carencias que cubrir. La grabadora actual contempla un rango de bases de tiempo difícil de igualar con recursos de hardware de computadoras personales. Es por ello que en los periodos de tiempo más pequeños, este desarrollo no resulta competitivo; no obstante mientras, los avances del hardware de computadoras continúen, esta limitación podrá ser cubierta.

Por otro lado, en cuanto al manejo de impresión, la técnica utilizada resulta poco satisfactoria, dado que se utilizan clases que están poco definidas y en constante cambio. Se sugiere para futuras mejoras implementar una técnica de impresión más eficiente. La causa de la lentitud en la impresión de los datos está identificada y radica en el hecho de que se envían a impresión los datos uno por uno, mientras que la forma en que se debería hacer, es tomar la gráfica como un objeto, sin importar su naturaleza, e imprimirlo como una imagen.

En cuanto a la capacidad de desplegado en pantalla, como se comentó oportunamente, la grabadora despliega una cantidad de puntos difícil de cubrir con un monitor con resolución convencional; no obstante, de ser deseado, se podría aumentar la capacidad del programa para desplegar un mayor número de puntos, aunque en principio, se consideran suficientes los que despliega actualmente.

Otro punto que también se había considerado, es en relación con la capacidad de despliegue de datos en sentido horizontal, es decir, se sugiere omitir todo un conjunto de datos, entre un periodo y otro, y únicamente tomar datos cada tercer periodo o incluso mas,

esto reduciría necesidades de los recursos del sistema, sin afectar visiblemente los resultados entregados.

El alcance de este sistema incluye exclusivamente sustituir el despliegue de la grabadora. No obstante, la otra función de la grabadora, que es proporcionar el pulso al transductor, se conserva intacta. Es por esta razón que la intención de utilizar el ADC fue descartada después de las primeras pruebas. Si en un futuro se desease sustituir completamente la grabadora, sería necesario tener osciladores, de amplitud y frecuencia variables para el ecosonda, mientras que para el convertidor se necesitaría un banco de amplificadores, para alternar entre ellos cada que haya un cambio en la base de tiempo, ya que la amplitud de la señal analógica del ecosonda es distinta para cada base de tiempo

REFERENCIAS

1. Transductores Hidroacústicos, Reportes de Desarrollo
Haro, Arturo; González, Arturo; Hernández, Jaime; Gómez, Humberto
Volumen 4, No. 9
Octubre de 1994
IIMAS, UNAM
Cap. 1 al 4 y apéndice D

2. Key Java Advanced Tips and Techniques
John Hunt and Alex McManus
Ed. Springer
Gran Bretaña, 1998

3. TTL Logic Data Book, Texas Instruments, 1988

4. Java en pocas palabras, 2ª Edición
Flanagan David
Ed. O'reilly
Cap. 3

5. Java in a nutshell, 3rd Edition
Flanagan David
Ed. O'reilly
Cap. 4

6. Electricidad y magnetismo

Jaramillo, Gabriel
Ed. Trillas
México D.F.
Pp. 512-514

7. http://www.doc.ic.ac.uk/~ih/doc/par/ "Interfacing to the IBM PC parallel printer port"

8. http://www.lvr.com/parprtib.htm "Parallel Port Complete" by Jan Axelson

9. http://www.national.com/pf/AD/ADC0809.html "Product Folder ADC0809"

10. http://www.edoceramic.com/NavDucArr.htm "EDO Coorporation, Naval Transducers and Arrays"

11. http://www.maxim-ic.com/1st_pages/ENGJR33.htm "Pipelines ADCs come of age"

12. http://www.techfest.com/hardware/bus.html "Bus and I/O Standards"

13. http://java.sun.com/products/jdk/1.1/docs/tooldocs/solaris/javadoc.html "javadoc – The Java API documentation"

14. http://www.rae.es/NIVEL1/buscon/ntlle.HTML "Diccionario de la lengua española; Real Academia Española"

15. http://www.dwheeler.com/java-imp.html "Alternative Java Implementations"

16. http://java.sun.com/products/hotspot/ "Java HotSpot Technology"

17. http://java.sun.com/docs/books/tutorial/post1.0/preview/performance.html "Java Tutorial; Performance Enhacements"

18. The Complete Java Certification Study Guide
Simon Roberts, Philip Reynolds
Ed. Sybex, Inc.
1 de Julio de 2001

19. Java Threads
Oaks, Scott; Wong Henry
Ed. O'reilly
Enero de 1997
California, Estados Unidos.

ANEXOS

Documentación del código fuente en Java

La documentación del código fuente hecho en Java fue generada utilizando **javadoc**, que es una herramienta incluida en el JDK. Ésta realiza una interpretación de las declaraciones y de los comentarios en un conjunto de archivos fuente, y produce un conjunto correspondiente de páginas HTML que describen las clases públicas y privadas, clases internas, interfaces, constructores, métodos y campos.

Esta herramienta se puede ejecutar en paquetes completos, archivos individuales o ambos. En el primer caso, se utilizan como argumento del comando **javadoc** una serie de nombres de paquetes; en el segundo caso se utilizan una serie de nombres de los archivos fuente. Para una descripción detallada del uso de esta herramienta, se recomienda recurrir a la referencia número 13 de este texto.

El comando que fue utilizado para elaborar dicha documentación es el siguiente:

```
javadoc -version -author -sourcepath . *.java -d ./documentacion
```

donde:

-version, es un parámetro que indica que se debe generar la versión indicada en los códigos fuente.

-author, es un parámetro que indica que se debe agregar el autor indicado en los códigos fuente.

-sourcepath, indica que a continuación se indica el directorio donde están los archivos fuente; para este caso, el comando es ejecutado desde el directorio donde se encuentran dichos archivos, por lo que a continuación sólo se escribe ".".

*.java, indica cuáles son los archivos fuente a documentar, que en este caso son todos.
-d, indica que a continuación se detalla el directorio donde se debe colocar la documentación.
./documentacion, es un subdirectorio del directorio donde se encuentran los archivos fuente, y es el lugar donde se colocará la documentación.

Documentación generada

A continuación se muestra un ejemplo de la documentación que es creada al ejecutar este comando.

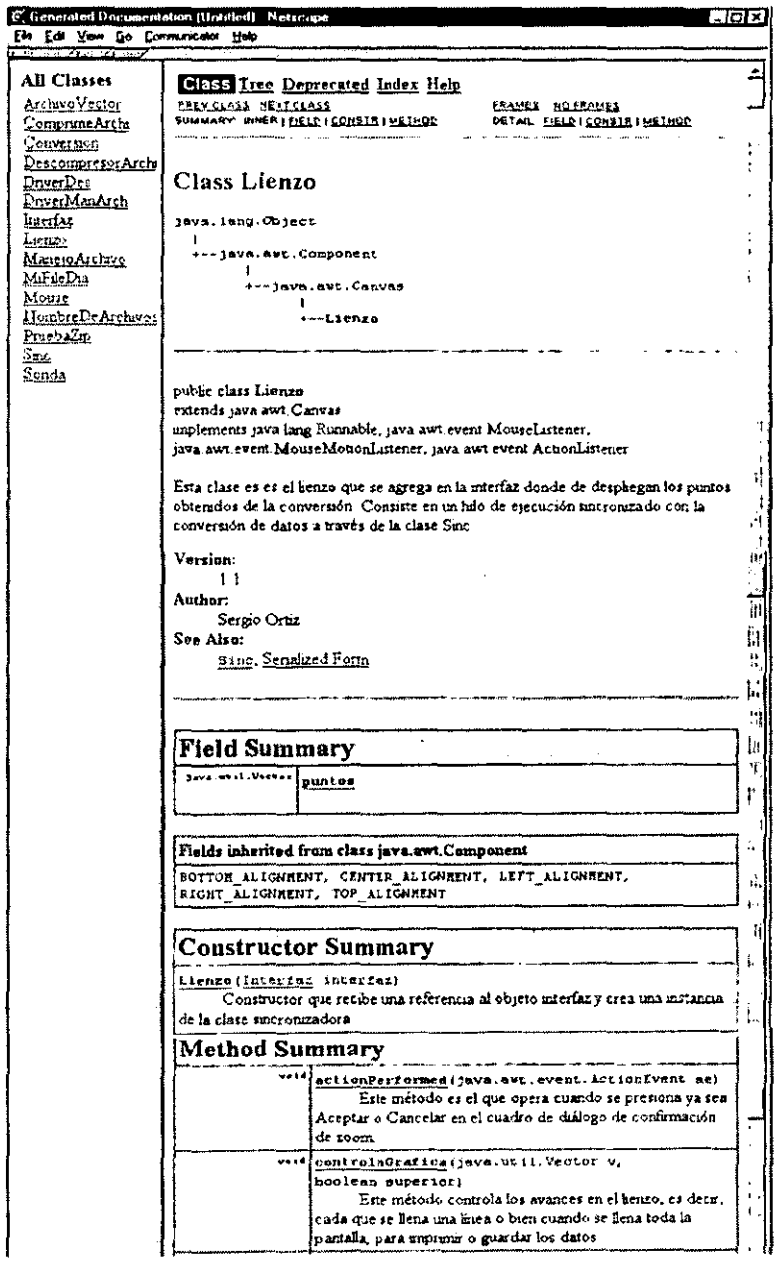

Fig 21. Muestra de la documentación generada con **javadoc**.

Para ver esta documentación es necesario abrir el archivo **index.html**, generado por la herramienta, con un navegador o *browser*; el único requerimiento para el navegador es que soporte el uso de *Cascade Style Sheets* u hojas de estilo, que son modificaciones a las etiquetas de HTML para dar cierto diseño a la documentación (estas hojas de estilo también son generadas automáticamente por la herramienta).

Se puede ver en la figura 21, que la distribución que tiene la documentación es con base en recuadros; en el recuadro izquierdo se muestran todas las clases que están documentadas; en el recuadro derecho se encuentra un menú con cinco opciones: *Class*, que es la opción resaltada cuando se selecciona una de las clases del lado izquierdo, muestra la jerarquía de la clase, además, muestra su declaración acompañada de la explicación documentada, después viene información del autor, la versión, y clases relacionadas, posteriormente, vienen documentados los atributos de la clase y aquellos heredados, luego, el o los constructores y un resumen de los métodos con su descripción, y finalmente, un detalle de los métodos de la clase; *Tree*, muestra en el recuadro izquierdo todas las clases, pero con su jerarquía de clases (tal como se muestra en la jerarquía mostrada en la figura 21, para la clase Lienzo, pero sin mayor detalle); *Deprecated* muestra los métodos y clases cuyo uso ya no es recomendado; *Index*, muestra a manera de índice alfabético todos los métodos de todas las clases documentadas; y finalmente, *Help*, que es una ayuda de cómo utilizar la documentación.

Esta documentación es demasiado extensa y completa, por lo que deberá ir acompañando a los archivos fuente al momento de la instalación.

Curva de magnetización

Si se desean obtener las características de un material ferromagnético dado, se toma una muestra del mismo y se construye una gráfica de la magnitud de la inducción magnética B en el material, para diversos valores de la intensidad magnética H. De la gráfica así construida, se podrán obtener los valores de la permeabilidad que se desee, como el cociente B/H.

El procedimiento usual para la obtención de la información necesaria para dibujar la gráfica B contra H, es construir un pequeño núcleo toroidal sobre el cual se devanan dos bobinas, como se muestra en la figura 22. Este método aprovecha el hecho de que para un devanado uniforme los campos son aproximadamente constantes en toda la sección transversal del toroide.

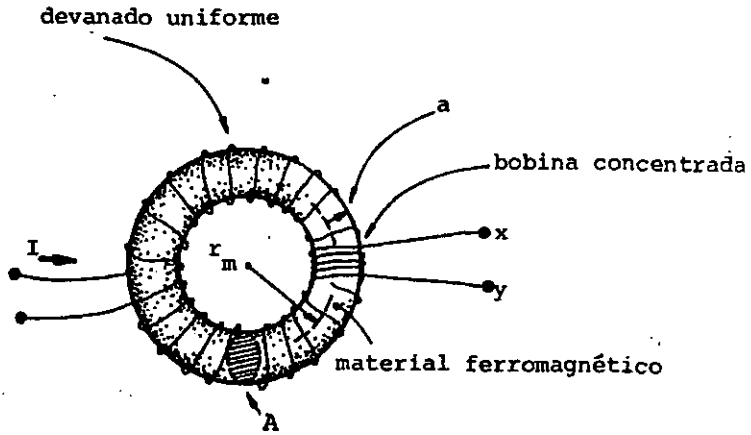

Fig. 22. Experimento de Rowland para obtener la curva de magnetización

Por medio de la ley de Faraday es posible determinar las variaciones del flujo magnético ϕ_b, midiendo la diferencia de potencial inducida en las terminales x-y.

Para una variación conocida de I es posible determinar ϕ_b, y como B es aproximadamente constante en toda la sección, su valor se obtiene mediante la relación

$$B = \phi_b / A$$

La figura siguiente muestra una curva típica obtenida con una muestra originalmente sin magnetización, o simplemente curvas de magnetización.

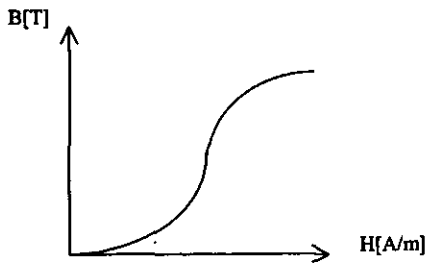

Fig. 23. Curva de magnetización típica para un material ferromagnético.

Técnicas de conversión

Desde mediados de los años 70, los convertidores analógico-digital monolíticos (ADC) han empleado técnicas de integración, de aproximaciones sucesivas, y de conversión tipo "flash". En los años 80 los diseños delta-sigma extendieron las alternativas. Recientemente, apareció una nueva clase conocida como "pipeline" o entubamiento.

ADC de conversión directa

La conversión directa es una de las más rápidas de las anteriormente mencionadas; también es conocida como conversión tipo "flash". Los ADC basados en esta arquitectura son extremadamente rápidos y realizan directamente su conversión de multiples bits, pero requieren de un diseño analógico intenso para manejar el gran número de comparadores y voltajes de referencia requeridos. Como se muestra en la figura 24, un convertidor con resolución de N bits tiene 2^N-1 comparadores conectados en paralelo, con sus voltajes de referencia definidos por una red de resistores y espaciados $V_{REF}/2^N$.

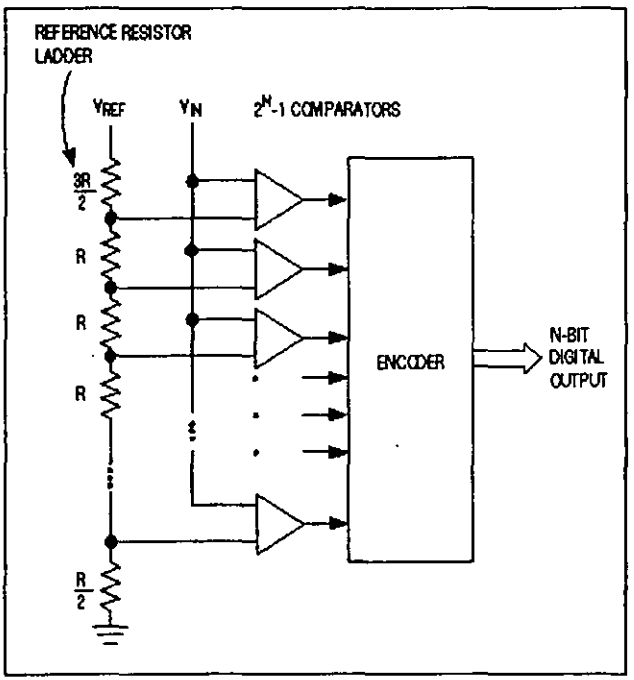

Fig. 24. ADC basado en la arquitectura de conversión directa, incluye un banco de 2^N-1 comparadores y una red de resistencias divisoras para los voltajes de referencia

Un cambio en el voltaje de entrada normalmente causa un cambio de estado en la salida de más de uno de los comparadores. Estos cambios en las salidas son combinados con una unidad de lógica de decodificación que produce una salida del convertidor de N bits en paralelo. A pesar de que estos convertidores son los más rápidos disponibles, su resolución está limitada por el tamaño del arreglo de circuitos, por la excesiva capacitancia de entrada y consumo de potencia provocado por el gran número de comparadores utilizados. La estructura repetitiva demanda una precisión idéntica de las secciones de los comparadores, debido a que cualquier diferencia puede causar un error estático como un voltaje de compensación excesivo en la entrada.

Los ADC tipo flash son propensos a errores esporádicos y salidas erráticas conocidas como "sparkle codes", las cuales tienen dos causas principales:

- metaestabilidad en los comparadores (estado de aletargamiento);
- burbujas debido a la temperatura

Los retardos desiguales de los comparadores pueden convertir un 1 lógico en 0 ó viceversa, causando la aparición de "burbujas" que no ocurrirían bajo una temperatura adecuada. Debido a que la unidad de decodificación no puede detectar este error, se genera una salida fuera de secuencia que aparece como un destello "spark" en la salida.

El tamaño de uno de estos convertidores de 8 bits es casi 7 veces mayor a su equivalente ADC con técnica tipo "pipeline". Al mismo tiempo que la capacitancia de entrada puede ser de hasta 6 veces mayor y su consumo de potencia del doble que el ADC tipo "pipeline".

ADC de aproximaciones sucesivas

La técnica de conversión basada en un registro de aproximaciones sucesivas (SAR[40]), también conocida como bits ponderados, o en inglés bit-weighing, emplea un comparador para comparar el voltaje de entrada con la salida de un convertidor digital-analógico de N bits (DAC). Utilizando la salida del DAC como referencia o realimentción, este proceso se aproxima al resultado final como la suma de N pasos ponderados, en el cual cada paso es una conversión para un bit.

En el primer paso se almacena el bit más significativo del DAC en el SAR, y en el siguiente paso compara este valor (MSB) con la entrada. La salida del comparador (alta o baja) es alimentada al DAC como una corrección antes de que se realice la siguiente comparación. Utilizando un circuito reloj de control, el SAR continúa esta ponderación y corrimiento hasta que se complete el paso del bit menos significativo (LSB), el cual produce una salida del DAC dentro de un rango de error de ±1/2LSB. Así como se va determinando cada bit, este es retenido en el SAR como parte de la salida del ADC.

[40] Del inglés "succesive-approximation register" se toma el acrónimo SAR.

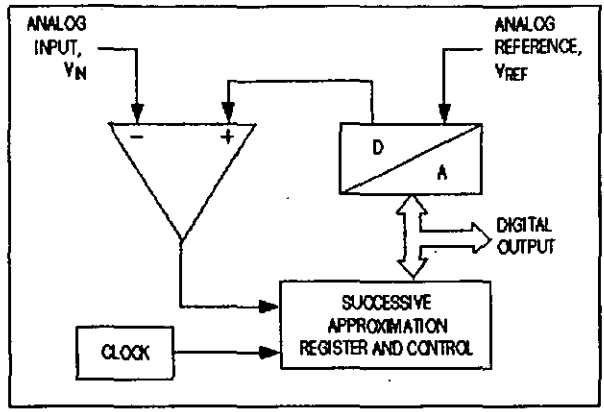

Fig. 25. ADC de aproximaciones sucesivas, consiste en un DAC, un comparador, un registro de aproximaciones sucesivas, un reloj y lógica de control

Estos convertidores toman muestras a una tasa de 1 Mega muestra por segundo, manejan un suministro de corriente relativamente bajo y tienen un bajo costo de producción, no obstante su diseño analógico es intenso. Comparado con la arquitectura de convertidores tipo "pipeline", los ADC de aproximaciones sucesivas proporcionan un bajo ancho de banda de entrada.

ADC con técnica de integración

El ADC con técnica de integración, también llamado dual o de pendiente múltiple, es uno de los más populares. El clásico convertidor dual tiene dos secciones principales: un circuito que adquiere y digitaliza la señal, produciendo un intervalo de tiempo o una secuencia de pulsos; y un contador que traduce este resultado en el valor digital de salida.

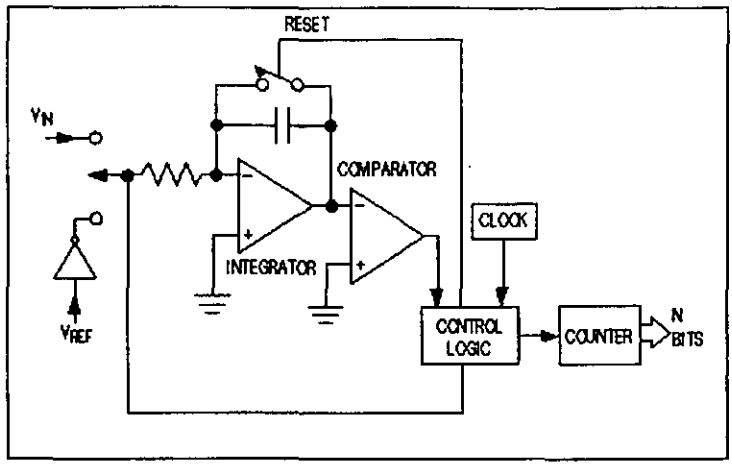

Fig. 26. Una de las técnicas de conversión más lentas pero más simples emplea un integrador que se carga con el voltaje de entrada y se descarga con un voltaje de referencia de polaridad opuesta

El convertidor de doble pendiente utiliza un integrador analógico con entradas conmutadas, un comparador y una unidad de conteo. El voltaje de entrada es integrado por un intervalo de tiempo fijo (T_{CHARGE}) que usualmente corresponde al conteo máximo de la unidad de conteo interno. Al final de este intervalo, el dispositivo comienza nuevamente a contar y aplica una referencia de polaridad inversa (negativa) a la entrada del integrador. Con esta señal aplicada de polaridad opuesta, el integrador tiene una pendiente negativa hasta que su salida alcanza el cero, lo cual detiene el contador y borra el estado del integrador.

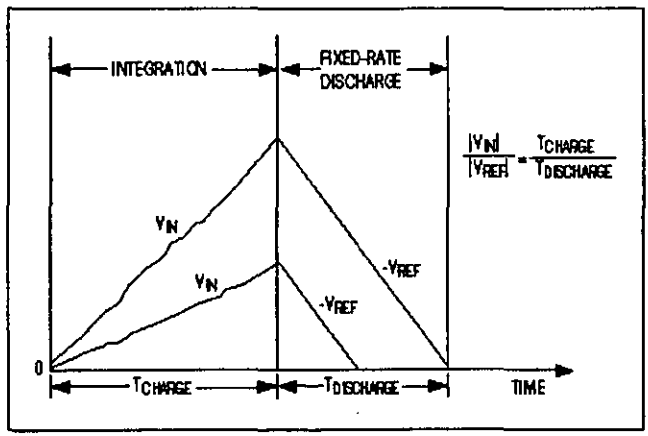

Fig. 27. Estas gráficas de voltaje ilustran la relación de tiempos para un ADC de integración con doble pendiente

La carga adquirida por el capacitor del integrador durante el primer intervalo de integración ($T_{CHARGE}/|V_{IN}|$) debe ser igual al que se utiliza durante el segundo intervalo de integración con pendiente negativa ($T_{DISCHARGE}/|V_{REF}|$). Entonces, la salida binaria es proporcional a la relación de estos intervalos de tiempo con la cuenta total. $T_{DISCHARGE}$ al final del segundo intervalo corresponde al valor digital de salida del ADC.

$$\frac{|V_{IN}|}{|V_{REF}|} = \frac{T_{CHARGE}}{T_{DISCHARGE}}$$

El sistema puede anular cualquier voltaje remanente durante la conversión, iniciando un ciclo de calibración dentro del convertidor. Comparado con los convertidores de tipo "pipeline", los de técnica de integración son extremadamente lentos con ancho de banda reducido. Pero su ventaja es el rechazo al ruido de alta frecuencia y su uso con bajas frecuencias fijas de 50 o 60 Hz los hacen útiles en entornos industriales y en aquéllos para los que no sea necesaria una alta tasa de conversión.

ADC tipo sigma delta (Σ-Δ)

Los convertidores sigma delta (Σ-Δ), también llamados convertidores de sobremuestreo, consisten en un modulador Σ-Δ seguido por un filtro digital. El modulador, cuya arquitectura es similar al ADC de doble pendiente, incluye un integrador y un comparador con un ciclo de retroalimentación que contiene un DAC de un bit. Este DAC interno es simplemente un interruptor que conecta la entrada del comparador con una referencia de voltaje positiva-negativa. Este ADC también incluye una unidad de reloj que proporciona los tiempos adecuados para el modulador y el filtro digital.

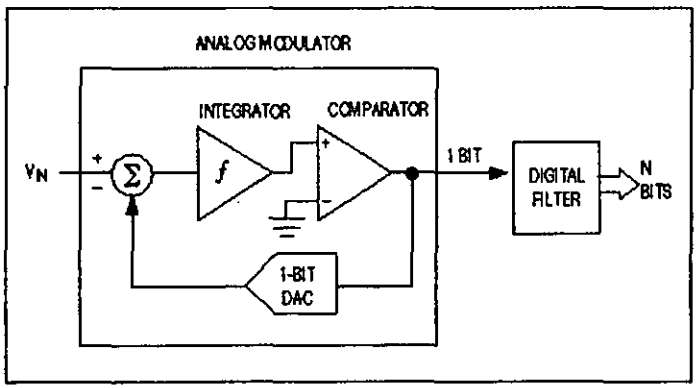

Fig. 28. Los dos bloques principales de un convertidor sigma-delta con el modulador y el filtro digital

Las señales con ancho de banda pequeño aplicadas como entrada a un ADC Σ-Δ son cuantizadas con una resolución muy baja (1-bit), pero con una frecuencia de alta tasa de muestreo, como 2 MHz o mayor. Combinado con un filtro digital, este sobremuestreo reduce su tasa de muestreo a 8 kHz, e incrementa la resolución del ADC a 16 bits. Aunque son más lentos que los de técnica "pipeline" y manejan señales de entrada con un estrecho ancho de banda, estos convertidores presentan tres grandes ventajas:

- Una conversión de bajo costo y alta eficiencia
- Un filtrado digital al momento de la conversión
- Compatibles con DSP para integración con sistemas

ADC tipo "pipeline"

Este tipo de convertidores, también conocidos como "subranging quantiziers" por su ponderación por subrangos, consisten de numerosas etapas consecutivas; cada una contiene un circuito conocido como *track/hold*[41] (T/H), un ADC y un DAC de baja resolución y un circuito sumador que incluye un amplificador entre las etapas para proporcionar ganancia.

Las aplicaciones comunes para estos convertidores son los sistemas de comunicación y los de adquisición de datos.

Las conversiones de N bits son rápidas y precisas, utilizando al menos dos etapas sucesivas. Primero, se ejecuta una rápida conversión de M bits. Luego, usando un DAC con precisión mínima de N bits, se convierte el resultado a uno de los 2^M niveles analógicos y es comparado con la entrada. Finalmente, la diferencia es convertida con un convertidor de alta precisión tipo "flash", de K bits, y las dos o más etapas de salida son combinadas; ver figura 29.

[41] Se podría sustituir este término por algo similar a "persecución y retención", no obstante, para evitar ambigüedades se decide dejar el término en inglés.

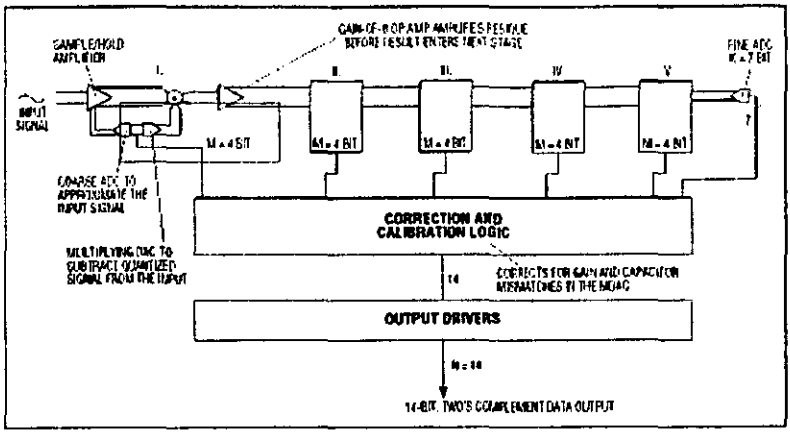

Fig. 29[42]. Este diagrama funcional muestra la lógica de corrección de error y calibración de un ADC tipo "pipeline" de 5 etapas

La mayoría de convertidores ADC tipo "pipeline" incluyen circuitería digital de corrección de errores que opera entre las etapas. Algunos convertidores tienen una unidad de calibración que compensa efectos no deseados como variaciones de temperatura, o desacoplamiento capacitivo en el DAC que realiza la multiplicación.

[42] Ver la referencia número 11.

Cuadro comparativo

A continuación se muestra un cuadro con las distintas técnicas de conversión, considerando las resoluciones soportadas, su rango de velocidades y otras características.

Arquitectura	Resolución	Velocidad	Ventajas/Desventajas
Flash	8 bits	250 Msps[43] – 1 Gsps	+ Extremadamente rápido + Ancho de banda de entrada elevado - Consumo de potencia elevado - Gran tamaño físico - Gran impedancia de entrada - Alto costo - Sparkle codes[44]
SAR	10 bits - 16 bits	76 ksps – 250 ksps	+ Alta resolución y precisión + Bajo consumo de potencia + Pocos componentes externos - Ancho de banda de entrada estrecho - Tasa de muestreo limitada - V_N debe permanecer constante durante la conversión[45]
Integradores	> 18 bits	< 50 ksps	+ Alta resolución + Bajo consumo de corriente + Rechazo al ruido excelente - Baja velocidad
Sigma/Delta	> 16 bits	> 200 ksps	+ Alta resolución + Alto ancho de banda + Filtrado digital dentro del circuito - Tasa de muestreo limitada
Pipeline	12 bits – 16 bits	1 Msps – 80 Msps	+ Tasa de conversión elevada + Bajo consumo de potencia + Corrección digital de errores y auto calibración - Requiere 50% del ciclo típico de trabajo - Requiere una frecuencia de reloj muy baja

Tabla 2. Cuadro comparativo de las técnicas de conversión usadas en los convertidores ADC

[43] Msps unidad de mega muestras por segundo (mega samples per second), de igual forma Gsps y ksps son giga muestras por segundo y kilo muestras por segundo.
[44] Ver la descripción de los convertidores tipo flash.
[45] Ver la descripción de los convertidores de aproximaciones sucesivas.

Decibeles

El fondo que rodea al término *decibel* tiene su origen en el hecho establecido de que la potencia y los niveles de sonido están relacionados con la base logarítmica. Esto es, un incremento en el nivel de potencia, digamos de 4 a 16 W, no corresponde a un incremento del nivel de audio por un factor de 16/4=4. Sólo se incrementará por un factor de 2, que se obtiene como el exponente que se requiere aplicar a la potencia inicial 4 de manera que 4^2=16. Para un cambio de 4 a 64 W, el nivel de audio se incrementará por un factor de 3, debido a que 4^3=64. En forma logarítmica, la relación puede escribirse como $\log_4 64=3$.

Para efectos de estandarización, se definió al *bel* (B) mediante la siguiente ecuación que relaciona los niveles de potencia P_1 y P_2:

$$G = \log_{10} \frac{P_2}{P_1} [B] \qquad (1)$$

Sin embargo, se encontró que el *bel* era una unidad de medición demasiado grande para propósitos prácticos, por lo que se definió el decibel (dB), de forma que 10 dB = 1 B. Por tanto,

$$G_{dB} = 10 \log_{10} \frac{P_2}{P_1} [dB] \qquad (2)$$

Existe una segunda ecuación para los decibeles que se aplica frecuentemente. Si se tiene una red de circuitos con voltaje de entrada V_i igual a algún valor V_1, la potencia de entrada sería $P_1=V_1^2/R_i$, donde R_i es la resistencia de entrada de este sistema. Si V_i debiera aumentarse (o disminuirse) a algún otro nivel, V_2, entonces la potencia resultante sería $P_2=V_2^2/R_i$. Si sustituimos en la ecuación 2 para determinar la relación resultante en decibeles entre los niveles de voltaje:

$$G_{dB} = 20 \log_{10} \frac{V_2}{V_1} [dB] \qquad (3)$$