ESCUELA SUPERIOR

POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD

"Sistemas Computarizado de Adquisición y Procesamiento de Datos para un Laboratorio de Calidad de Aguas"

Tesis de Grado

Previo a la obtención del Título de Ingeniero en Computación

Presentada por:

Johnny Macías Cisneros

GUAYAQUIL - ECUADOR

1994

AGRADECIMIENTO

Agradezco a Dios por todo lo que me ha permitido aprender en el transcurso del desarrollo de esta tesis.

Mi más sincero agradecimiento a mi Director de Tesis, Maestro y Amigo, el Ing. Sixto García Aguilar, por su constante colaboración y apoyo en el desarrollo de este trabajo.

DEDICATORIA

A MIS PADRES, quienes con el ejemplo me enseñaron el camino a seguir, con su apoyo me levantaron cuando estaba caído y por quienes he llegado a ser lo que he sido hasta hoy.

A MARTHA quien ha estado conmigo durante mis años de universidad apoyándome enteramente en el desarrollo de mis trabajos.

A MISHELLE y a todos aquellos amigos silenciosos que creyeron en mí, este trabajo tiene un poco de todos ellos.

ING. ARMANDO ALTAMIRANO

SUB-DECANO DE LA FACULTAD

DE INGENIERIA ELECTRICA

ING. SIXTO GARCIA A.

DIRECTOR DE TESIS

ING. JAIME PUENTE P.

MIEMBRO PRINCIPAL

DEL TRIBUNAL

ING. GUIDO CAICEDO R.

MIEMBRO PRINCIPAL

DEL TRIBUNAL

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL).

JOHNNY MIGUEL MACIAS CISNEROS

INDICE GENERAL

INTRODUCCION

Con el paso del tiempo y el avance de la tecnologia, el hombre tiende a buscar soluciones para facilitar y optimizar más su trabajo. Tomar las herramientas disponibles en su tiempo y adaptarlas para mejorar su condición de trabajo ha sido siempre la meta a seguir. Es por eso que he escogido como tema para mi trabajo de tesis de graduación el desarrollo de un sistema que propone la integración de aspectos Fisico-Quimicos, Electrónicos y de Análisis de Sistemas para la resolución de un problema planteado a nivel científico: El monitoreo y análisis de mediciones de parámetros fisicos en muestras de aguas.

Los laboratorios de calidad de aguas en los últimos años han tomado especial importancia en lo que a análisis de contaminación en agua se refiere, es por eso que el desarrollo de sistemas inteligentes que recojan datos de la muestra y los procesen es imperativo, de forma que la preparación y manipulación de parámetros sea realizada de la manera más eficiente y rápida posible, asi como la generación de reportes que indiquen, en tiempo real, los cambios en el comportamiento de nuestro objeto de estudio.

El sistema desarrollado actualmente cuenta con un sensor de temperatura y circuiteria necesaria para adaptar electrodos que midan los siguientes parámetros fisicos: PH, potencial Oxido-Reduccción y Conductividad.

Un programa diseñado para el efecto y cargado previamente lee los datos y permite al operador un completo análisis de los parámetros obtenidos, la posibilidad de guardarlos en un archivo y obtener gráficos de parámetros versus parámetros o parámetros versus tiempo. El sistema tendrá la portabilidad necesaria para funcionar en cualquier máquina de caracteristicas similares a la de prueba solamente conectando la circuiteria de adquisición de datos al puerto paralelo y cargando el respectivo programa.

Se espera probar que los sistemas de adquisición de datos por computador son mucho más eficientes, exactos y rentables que los métodos tradicionales de monitoreo.

Los resultados obtenidos a partir de esta tesis buscan despertar el interés de los Ingenieros en Computación en el estudio e implementación de sistemas computarizados no tradicionales. Este tipo de sistemas, a más de ser rentables, significan un beneficio tecnológico más importante para el desarrollo del pais que el aportado por la elaboración de aplicaciones comerciales tradicionales.

CAPITULO 1

GENERALIDADES

1.1 DEFINICION DEL PROBLEMA

El trabajo de graduación implementado tiene como objetivo primordial el desarrollo de un sistema computarizado aplicado a un campo puramente científico mas allá que comercial, se trata de aplicar la computación a una rama no tradicional de trabajo como son los sistemas de adquisición de datos en tiempo real.

Los sistemas de monitoreo de cualquier tipo se basan en la necesidad de un determinado usuario para obtener eficientemente diversos datos, los cuales dependerán de la naturaleza del fenómeno o proceso a estudiar, básicamente estos sistemas constan de una circuitería acondicionadora de señal, un convertidor analógico-digital y un programa que controla propiamente la recolección de datos, presentando al usuario las mediciones obtenidas cada intervalo de tiempo. En el mercado local es difícil por no decir imposible encontrar sistemas de este tipo, no así en los mercados extranjeros. Los precios de los sistemas de recolección de datos varían fuertemente

dependiendo del poder del software más que de la complejidad de la circuitería, siendo posible encontrar sistemas de propósito general que van desde los 500 hasta los 3000 dólares, sin contar con los costos de sensores. Esto que ocasionaría a un laboratorio un gasto aproximado de 2000 dólares, siempre y cuando el sistema sea comprado en el exterior (sin intermediarios).

Lo que se desea demostrar con este trabajo es que los profesionales nacionales estamos en capacidad de desarrollar sistemas de monitoreo y control para propósitos específicos, con una calidad mayor o igual a la de los productos extranjeros y a un precio significativamente menor. Esto a mediano plazo generaría un impulso en la producción de aplicaciones computarizadas no tradicionales en el medio.

He tomado como ejemplo para desarrollar mi trabajo, un sistema de adquisición de datos aplicado a los laboratorios de contaminación o plantas de tratamiento de aguas de residuo (SYSAQUA), el cual debe obtener mediciones de parámetros físicos del agua. He supuesto que en dicho laboratorio se tendrá cierta muestra de agua, la cual será objeto de estudio por medio de manipulaciones en el tiempo reaccionando con diversos agentes químicos. El programa será ajustado para recibir datos cada n minutos, generándose reportes en línea, gráficos en tiempo real, suavizaciones de gráficos, etc. y con lo cual se pueda formular predicciones en los cambios de comportamiento de la muestra.

En el caso de plantas o procesos industriales, la circuitería tendrá que ser mínimamente modificada, tomando en cuenta que los sensores se encuentran

alejados del computador, no obstante el programa será el mismo. Para ambos casos, laboratorios o procesos industriales, el operador podrá fijar valores críticos los cuales dispararán una alarma si es que el valor monitoreado sobrepasan los valores normales.

Observando ciertos programas de monitoreo, llegué a la conclusión de que la mayoria de ellos no cuentan con interfaces amigables para el usuario, convirtiéndose en un verdadero problema para operadores inexpertos; por lo cual el diseño de las pantallas se convirtió en uno de los aspectos más importante de mi trabajo; otra observación es que por lo general los programas de monitoreo no presentan opciones para análisis de resultados, por lo cual he decidido implementar opciones sencillas que de alguna manera superen estas limitaciones (gráficas y suavizaciones) logrando con esto facilidad en el manejo por parte del usuario, asi como también versatilidad y poder en el producto.

1.2 OBJETIVOS

- ✓ Diseñar e implementar un sistema de adquisición de datos en tiempo real que realice el monitoreo y análisis de parámetros físicos en muestras de aguas utilizando un PC.

- ✓ Implementar un programa que además de realizar el control de la adquisición de datos, presente al usuario una interfaz gráfica amigable y de

fácil uso, aún para aquellos operadores no familiarizados con el manejo de computadores.

- ✓ Evaluar la técnica de programación orientada a objetos en sistemas de adquisición de datos.

- ✓ Familiarizarse, diseñar e implementar la circuitería (hardware) para un sistema de adquisición de datos, tomando en cuenta acondicionamiento de señal, digitalización y cuantificación de la misma, así como la integración al computador.

- ✓ Demostrar la utilidad práctica del sistema en un laboratorio tomando en cuanta su eficiencia y costo versus los métodos tradicionales de medición.

1.3 DESCRIPCION GENERAL DEL SISTEMA ANALIZADOR DE AGUAS (SYSAQUA).

SYSAQUA es un sistema en desarrollado con el fin monitorear parámetros físicos en muestras de agua tales como PH, Temperatura, Conductividad, Salinidad, Potencial REDOX; Enviarlos al computador y permitir al usuario la manipulación y comparación de dichos parámetros, con el fin de obtener conclusiones que permitan identificar fallas y optimizar los sistemas de control de parámetros.

SYSAQUA encuentra su principal aplicación en las industrias que utilizan control de parámetros físicos del agua, tanto para la fabricación de elaborados como en el tratamiento de aguas residuales; actualmente el sistema está proyectado exclusivamente para el monitoreo, debido a los costos de los actuadores requeridos para tomar acción y controlar totalmente el tratamiento del agua; pero para el sistema, tanto en hardware como en software, será posible aumentar módulos que manejen controladores de elementos externos.

Un factor importante en este sistema será su portabilidad. Se lo podrá conectar a cualquier computador con procesador 80286 o mayor que cuente con una interfaz paralela del tipo Centronic's .

HARDWARE

La adquisición de datos se la realiza vía puerto paralelo del computador, al cual está conectado la circuitería que hace posible el acondicionamiento de la señal.

Esta circuitería está conformada por un acondicionador de señal y un módulo convertidor analógico/digital (ADC), éste módulo es multiplexado con el fin de poder discriminar la señal de entrada.

El acondicionador de señal se ocupa de minimizar el ruido inducido y amplificar el voltaje entregado por los sensores haciendo que éste llegue a un nivel adecuado para ingresar al ADC, el acondicionador es esencialmente formado por un amplificador seguidor de voltaje y un amplificador diferencial (amplificador de instrumentación).

La señal con los voltajes acondicionados son ingresados al ADC y por medio de una señal de control enviada desde el computador realiza la multiplexión de los datos.

Una vez convertido la señal de analógico a digital de 8 bits, es introducido a la interfaz paralela Centronic's.

SOFTWARE

El Software de Procesamiento de datos está diseñado para recoger los datos del modulo acondicionador de señal, manipularlos gráficamente y guardarlos en una base de datos,bajo formato LOTUS, para permitir futuras manipulaciones desde esa hoja electrónica.

La interfaz con el usuario, puramente gráfica, complementado con una pequeña ayuda en linea sobre tópicos especiales, brindará al operador facilidades para el manejo del sistema y una amplia apreciación de los gráficos generados a partir de las mediciones obtenidas.

El sistema de tipo Programable por el Usuario, podrá ser seteado para recoger mediciones cada cierto tiempo sin necesidad de control del operador. Contando con un sistema de alarma de parámetros detecta si lo monitoreado rebasa los limites establecidos previamente; además su porción residente permitirá trabajar en otras aplicaciones dentro del entorno del programa controlador mientras SYSAQUA monitorea los valores requeridos por usted.

Los gráficos generados para análisis de la muestra podrán ser del tipo parámetro versus parámetro, o parámetro versus tiempo.

REPORTES

Los reportes a generar para el usuario tendrán la siguiente información:

- ☑ Descripción de la Muestra a Analizar.
- ☑ Lista de Parámetros Programados.
- ☑ Intervalo de Tiempo para medición de Parámetros Especificados.
- ☑ Media de mediciones realizadas por Parámetro
- ☑ Mediciones registrados por hora y dia.
- ☑ Registro de Situaciones de Alarma.

Los Reportes podrán ser impresos o guardados en un archivo de disco para futura recuperación y análisis.

CAPITULO 2

DISEÑO DE HARDWARE PARA ADQUISICION DE DATOS

2.1 REQUERIMIENTOS FUNCIONALES DE HARDWARE

Los requerimientos de hardware para el sistema analizador de aguas (SYSAQUA) son básicamente similares a los de cualquier circuitería para adquisición de datos. He tratado de simplificar lo más posible los requerimientos, concentrándome en los aspectos principales de el monitoreo de señal y handshaking con el computador, puntualizando los requerimientos tenemos que:

- Las diversas señales provenientes de los transductores deben ser acopladas a un computador, usando el puerto paralelo del mismo.

- Realizar un acondicionamiento de la señal de entrada (desde los transductores) eliminando ruido y elevando el nivel de voltaje de mV a niveles de voltaje de hasta 5 voltios.

- Multiplexar de alguna manera las diversas señales y realizar una conversión analógica-digital controlada.

- La conversión analógica-digital deberá ser realizada con la mayor exactitud y rapidez posible.

- Conociendo que el puerto paralelo puede solamente recibir cuatro bits a la vez como entrada, implementar una circuiteria para multeplexación de los datos entregados al final del proceso de digitalización.

- Establecer señales entre el hardware y el computador que permitan hacer handshaking entre los dispositivos efectuando las conversiones de manera eficiente y sin errores.

- Considerar la existencia de varias tierras, una en el computador, distinta a la de la fuente alimentadora de la circuiteria lo cual hará que los niveles de voltaje lógicos difieran entre los dispositivos entorpeciendo la interacción.

- Asumir que los transductores se encuentra fisicamente cercanos a la circuiteria del sistema, lo cual no hace necesario consideraciones de atenuación de señal por transporte y pérdidas.

- Asumir que el sistema no está influenciado por fuentes de ruido externo considerables, lo que no hace necesario la utilización de filtros pasa altos o bajos.

2.2 DESCRIPCION DEL TRANSDUCTOR DE PRUEBA

Para probar la circuiteria de adquisición de datos utilicé un termistor, es decir, un transductor de temperatura con el cual monitoreo temperatura ambiente. Cabe anotar que por motivos de costo fue imposible acoplar los otros tres transductores con los cuales iba a contar el proyecto.

2.2.1 EL TERMISTOR

El Termistor es básicamente una resistencia que varía con el calor, esta propiedad es utilizada para obtener voltajes variables, los cuales al pasar por el ADC generarán señales digitales acordes con la temperatura medida.

Sabiendo que nuestros voltajes de referencia para la conversión analógica/digital van a ser +5V - 0V, entonces podemos diseñar un pequeño circuito, el cual transforme en voltajes dentro del rango establecido, la variación de temperatura registrada por el sensor. El circuito utilizado es el siguiente:

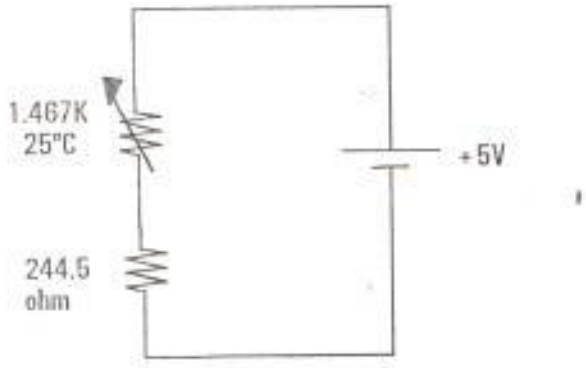


Fig. 2.1 Circuitería para el Termistor

Una aproximación de la curva Temperatura vs. Voltaje obtenida experimentalmente para este dispositivo en el circuito mostrado en la fig. 2.1 es:

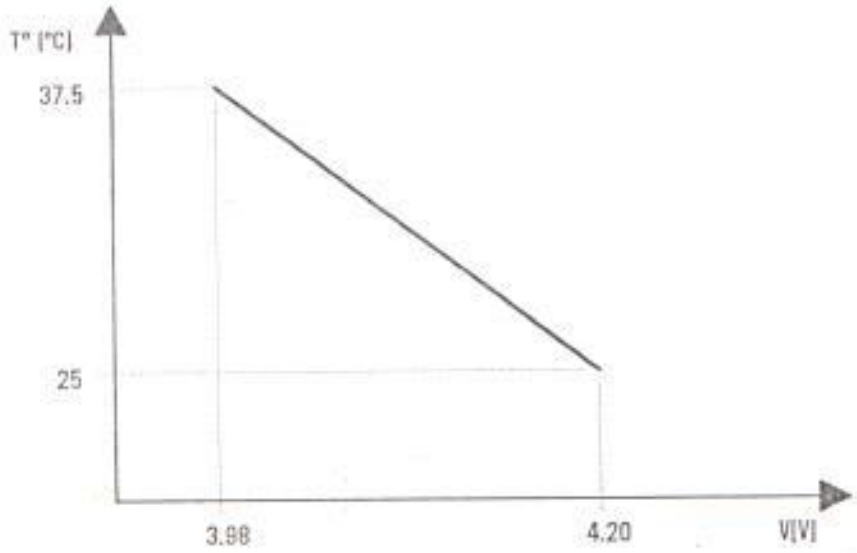


Fig. 2.2 Curva Temperatura vs. Voltaje

La cual obedece a la siguiente ecuación:

$$V(-56.81) + 263.6 = T^{\circ}$$

2.3 CONVERSION ANALOGICA DIGITAL

El sistema debe realizar conversiones de señales analógicas a digitales con la mayor exactitud posible y con una rapidez aceptable para que el concepto de medición en tiempo real no sea afectado, además tomando en cuenta que debemos muestrear cuatro es más conveniente usar un convertidor que a la vez ofrezca multiplexión de señales de entrada, por esto y bajo costo el convertidor elegido fue el ADC0808.

2.3.1 CARACTERISTICAS DEL ADC0808.

El ADC0808 es un dispositivo CMOS monolítico con un convertidor analógico-digital de 8 bits, con 8 canales de entrada multiplexado y compatible con lógica controladora de microprocesadores. Este convertidor utiliza la técnica de aproximaciones sucesivas para efectos de conversión. El multiplexor de 8 canales pueden accesar directamente a cualquiera de las 8 señales analógicas conectadas.

Este dispositivo elimina la necesidad de ajustes de cero y escala completa, esta provisto además de salidas de tres estados, este dispositivo ofrece alta velocidad, alta aproximación, dependencia mínima de temperatura, y consumo mínimo de poder.

Características:

- Resolución 8 bits
- Error ± 1/2 LSB y ± 1 MSB
- Tiempo de Conversión 100µS
- Alimentación +5Vdc
- Opera radiometricamente o con 5Vdc o voltaje de referencia ajustado
- Multiplexor de 8 canales
- Acoplamiento sencillo con microprocesadores
- No requiere de ajuste de cero o escala completa
- Rangos de temperatura -40°C a +85°C
- Consumo de potencia 15mW
- Salida de tres estados (latched TRI-STATE)

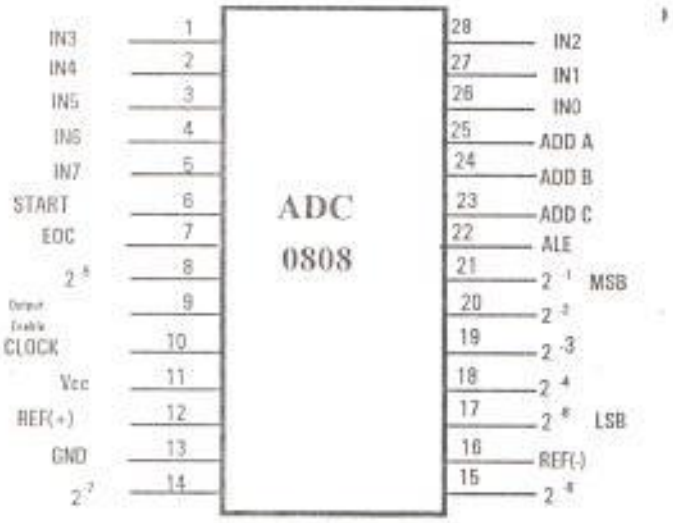


Fig 2.3 Diagrama de Conexión del ADC 0808

Diagramas de Tiempo:

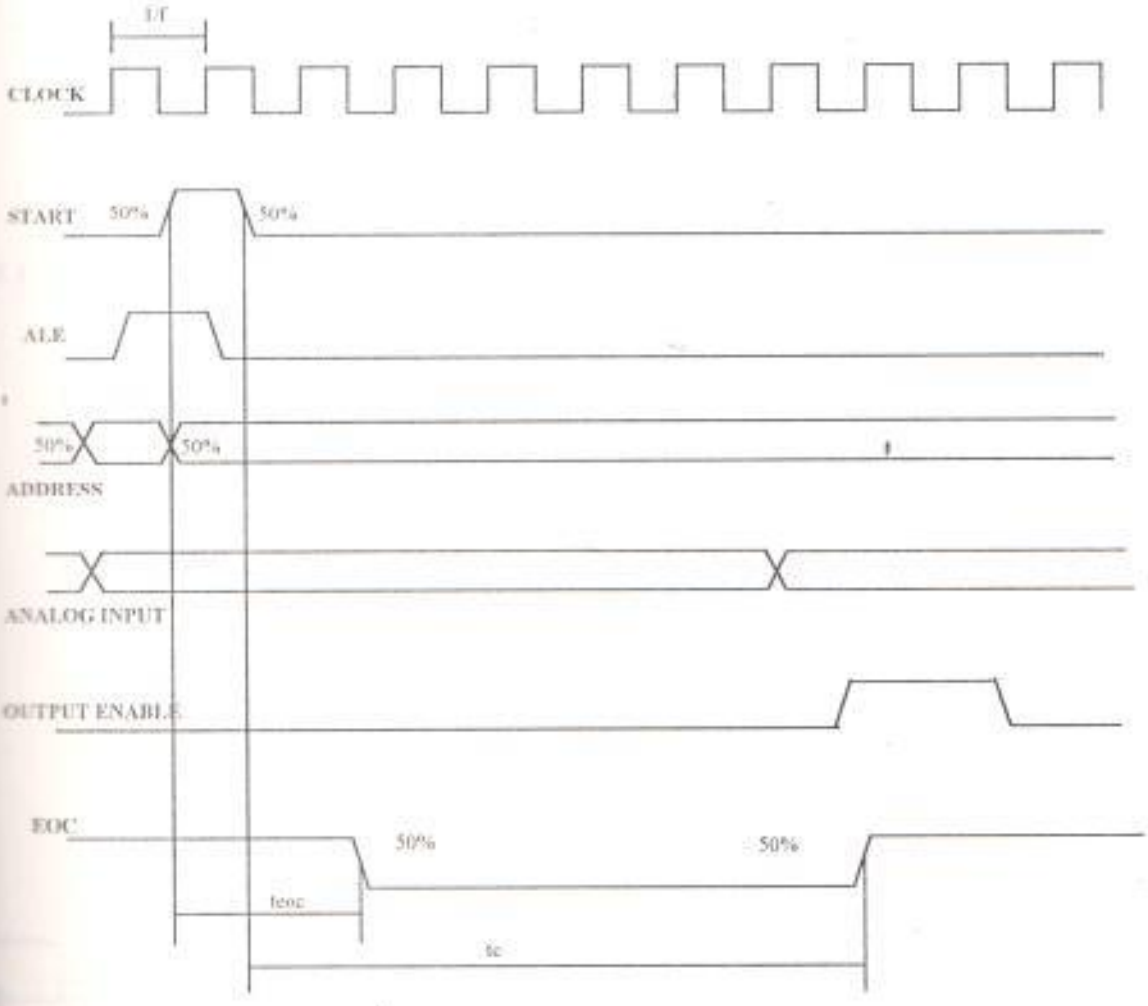


Fig. 2.4 Diagramas de Tiempo del ADC0808

2.4 DIAGRAMAS

En esta sección presentamos los diagramas de bloques y el diagrama detallado de la circuiteria de nuestro sistema de adquisición de datos de acuerdo a los requerimientos y asumsiones determinados anteriormente.

2.4.1 DIAGRAMA DE BLOQUES

El hardware del sistema de adquisición de datos esta compuesto por los siguientes módulos:

- Módulo Acondicionador de Señal
- Módulo Convertidor Analógico-Digital
- Módulo Multiplexor de Resultados

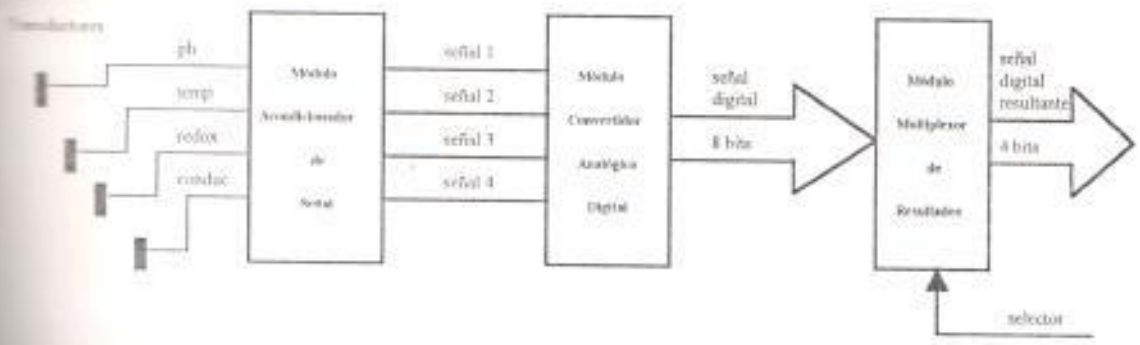


Fig.2.5 Diagrama de Bloques de Circuitería de Adquisición de Datos.

2.4.2 DESCRIPCION DE MÓDULOS

2.4.2.1 Módulo Acondicionador de Señal

El Módulo Acondicionador de Señal es aquel encargado de recoger la señal analógica de los sensores (niveles de voltaje) y amplificarlos a niveles de voltaje acorde con los voltajes de referencia del Módulo Convertidor Analógico Digital.

Esta formado por cuatro estructuras del siguiente tipo:

Un bloque amplificador seguidor de voltaje

Un bloque de amplificadores diferenciales

Los dos bloques acoplados forman un bloque de amplificadores de instrumentación, los cuales son amplificadores diferenciales optimizados con impedancia de entrada y CMRR altas. Un amplificador de instrumentación generalmente se usa en aplicaciones en las cuales un voltaje diferencial pequeño y un voltaje en modo común grande son las entradas, el seguidor de voltaje produce una impedancia de entrada muy alta.

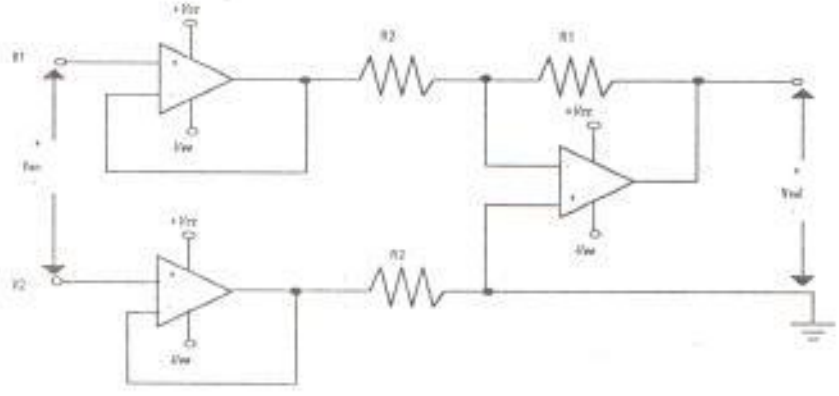


Fig. 2.6 Diagrama de Amplificador Instrumental

En este caso he utilizado OPAMPs en integrados UA741CP e integrados UA747CN conectados de la siguiente manera:

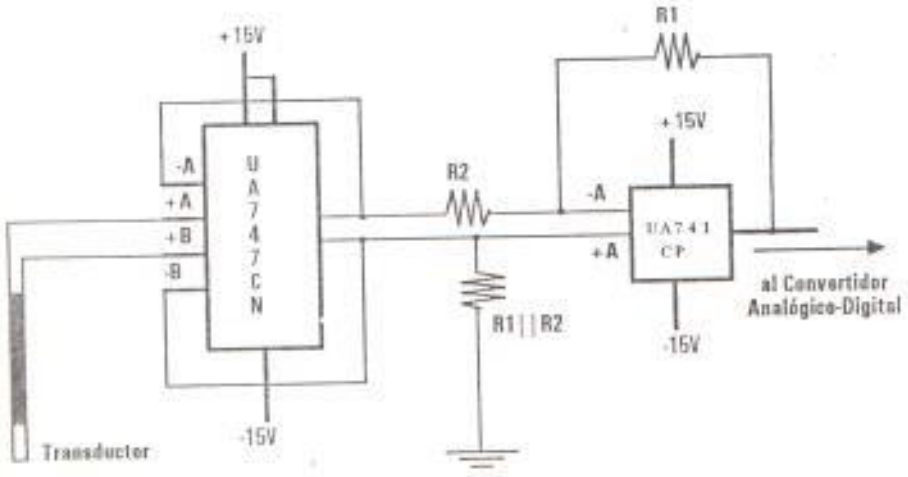


Fig. 2.7 Diagrama del Bloque Acondicionador de Señal

Las resistencias R1 y R2 varían de acuerdo a las ganancias requeridas en los bloques amplificadores de instrumentación de acuerdo a la respuesta de los transductores (niveles de voltaje de salida), así tenemos:

Sensor	R1 KΩ	R2 KΩ	R1||R2 KΩ	B
PH	625	1	1	1

Tabla 2-1. Valores de R1, R2 y ganancias

2.4.2.2 Módulo Convertidor Analógico-Digital

Las conexiones de las señales acondicionadas al convertidor analógico digital, son gobernadas por las entradas ALE, START, y por las direcciones C,B,A, las cuales vienen del computador. La salida EOC (fin de conversión) es monitoreada también por el computador, los 8 bits de salida están conectadas al módulo multiplexor de bits resultantes. El reloj conectado al ADC tiene una frecuencia de 1.5 KHz, es necesario tomar en cuenta la velocidad del computador con relación a la velocidad de conversión del ADC, la velocidad de conversión es dada por 8 pulsos de reloj, o sea mas o menos 2.6 seg., muy lento para el computador de prueba por lo cual el programa debe contemplar un pequeño retardo para dar paso a una efectiva captura de la señal, especialmente la salida EOC, una discusión mas profunda se expone en el capitulo de diseño de módulos.

Un diagrama de las conexiones del módulo convertidor Analógico-Digital se expone a continuación.

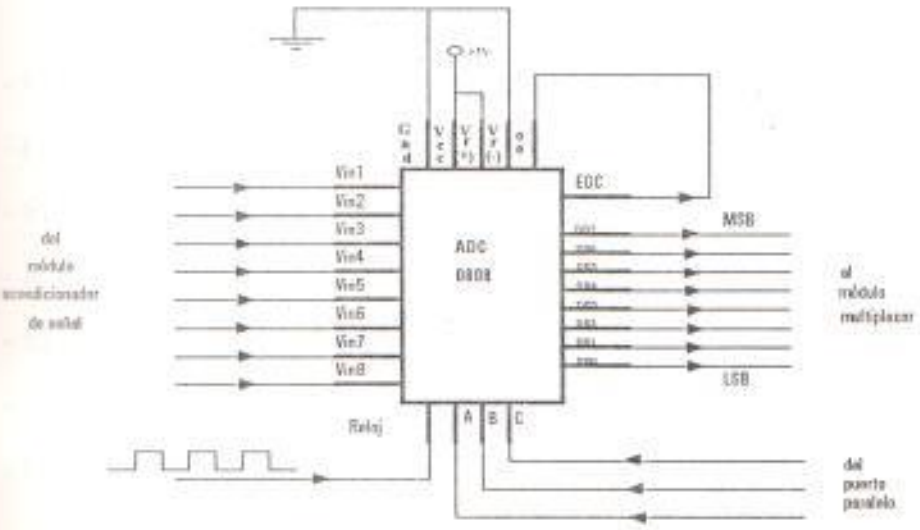


Fig. 2.8 Diagrama del Módulo Convertidor Analógico Digital

La señal de salida EOC sirve de entrada para OE (output enable) lo que permite que los datos digitalizados se posicionen en la salida del convertidor apenas éste haya terminado la conversión.

El voltaje de referencia positiva está dado a +5V, el voltaje de referencia negativa a tierra +0V, es decir los voltajes de referencia positiva y negativa son tomados directamente de la fuente de alimentación externa y para una correcta conversión los niveles de voltaje de las señales provenientes del módulo acondicionador deben estar entre este rango (+5 - 0 V).

2.4.2.3 Módulo Multiplexor de Resultados

El puerto paralelo del computador sólo acepta 4 bits de entrada, en vista de eso, para no perder resolución en los datos, y aprovechar los 8 bits entregados por el módulo convertidor analógico-digital, he implementado un módulo que entrega los 8 bits en dos grupos de cuatro, interactuando con el CPU, por medio de la siguiente mecánica:

- El módulo convertidor analógico digital genera los 8 bits de salida.
- Los 8 bits son recogidos por el módulo multiplexor.
- El CPU manda una señal de selección SLCT 0 al módulo.
- El módulo multiplexor entrega los cuatro bits menos significativos al CPU.
- El CPU recoge los cuatro bits y manda señal de selección SLCT 1.
- El módulo multiplexor entrega los cuatro bits más significativos.
- El CPU recibe los cuatro bits más significativos.

Considero que los 8 bits entregados por el módulo convertidor analógico-digital permanecen estables al menos hasta que el CPU reciba toda la información.

A continuación se presenta el diagrama del módulo multiplexor:

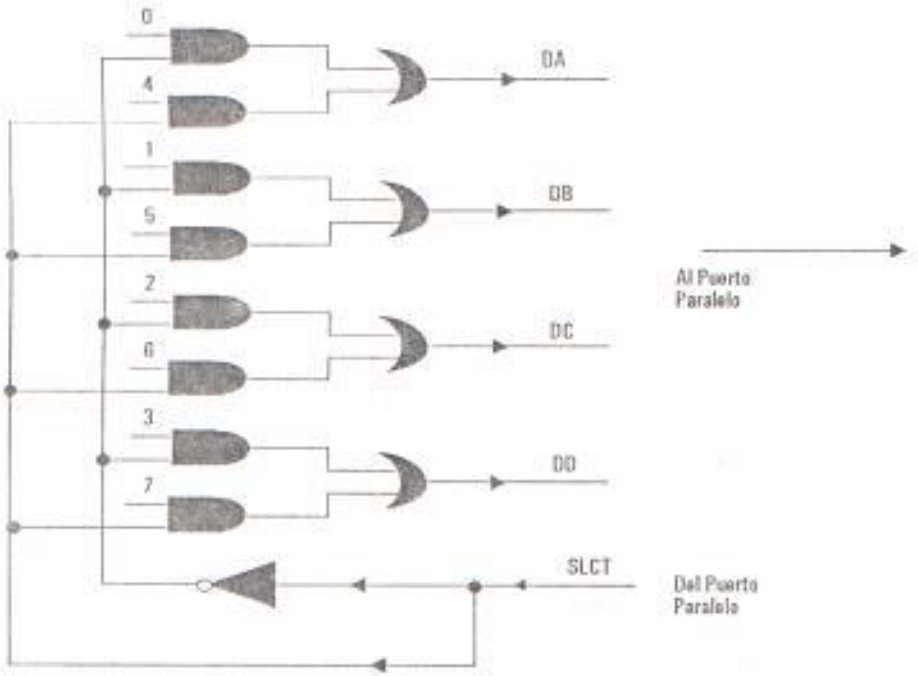


Fig 2.9 Diagrama del Módulo Multiplexor de Señal Resultado

0,1,2,3,4,5,6,7, son las señales digitales provenientes del módulo convertidor, Las señales DA, DB, DC, DD son los cuatro bits mas o menos significativos dependiendo si la señal activada desde el puerto paralelo es SLCT para el más significativo o SLCT negado para el menos significativo.

CAPITULO 3

CONEXIÓN CIRCUITERIA/PUERTO PARALELO

3.1 CONSIDERACIONES

El Hardware del sistema de adquisición de datos se comunica con el CPU a través del puerto paralelo, se tomó la decisión de muestrear datos por este puerto atendiendo las siguientes consideraciones:

- Se elige comunicación tipo paralela y no serial basado en la rapidez de este tipo de comunicación con el mundo externo.

- La rapidez lograda por la transmisión de datos en conjunto de bits ofrecida por el puerto paralelo es muy importante para lograr muestreos de datos en tiempo real.

- Se elige que la circuiteria totalmente externa al computador solamente conectada a través del puerto paralelo porque esto representa una manera

fácil y rápida para comunicarse con el CPU. Usando el puerto paralelo externo la instalación del sistema se vuelve sencilla, considerando que no hay que instalar tarjetas internas ni hay que hacer cambios en configuración.

- Se supone que la muestra se encuentra cerca al computador, por lo que la principal limitación del uso de puertos paralelos, elevado costo en cableado, no es relevante en nuestro caso.

- Aún si la muestra se encontrara distante, como en el caso de monitoreo de plantas y procesos industriales, se podría trasladar la señal analógica hacia el dispositivo convertidor analógico-digital, el cual se encuentra junto al computador conectado al puerto paralelo, realizándose la digitalización en un lugar físicamente cercano al computador, no en el lugar de monitoreo. Con esto queda descartado la distancia como un factor de consideración para las conexiones entre la circuitería externa y el CPU.

- Otra ventaja que ofrece el puerto paralelo programable es que los bytes de entrada, salida y control se mapean en posiciones diferentes de memoria, facilitando la manipulación de los mismos de forma independiente.

- El puerto paralelo programable tipo CENTRONIC´S fue el utilizado considerando su amplia popularidad entre los PC. La modularidad entonces queda así asegurada.

recolección de datos del dispositivo. La solución a este problema se lo encontró aterrizando todo el sistema a una tierra común para evitar así lazos de tierra, la tierra común utilizada para las pruebas fue un escritorio metálico dando excelentes resultados.

El siguiente diagrama ilustra en forma general las conexiones entre la circuitería de recolección de datos y el CPU (ver Fig 3.2).

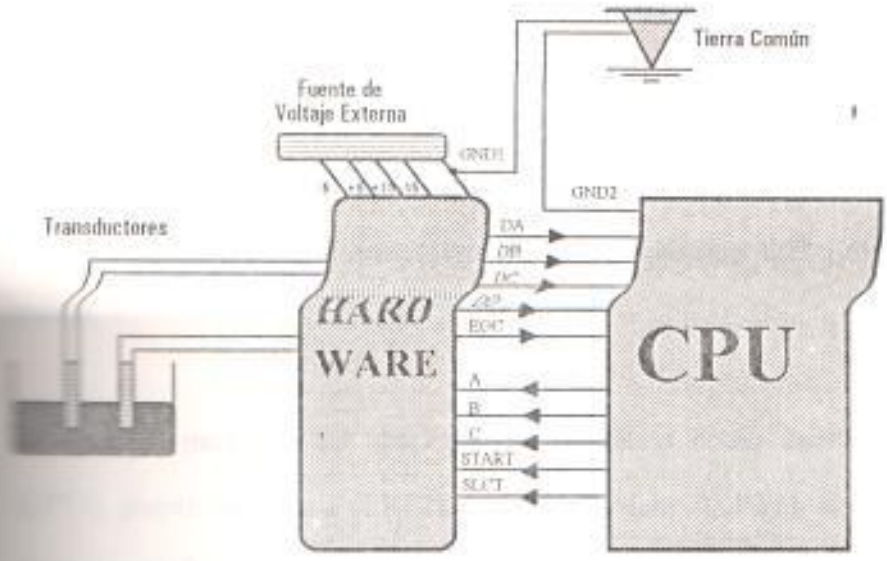


Fig. 3.2 Conexión a Tierra del Sistema.

PROGRAMA DE PRUEBA

Para probar el handsahking entre la circuitería y el CPU se implementó un programa de prueba, el cual a parte de recibir los bits de datos y enviar los bits

Para la instalación de nuestro sistema lo único necesario sería un puerto de paralelo libre (LPT1 preferentemente), la circuirtería de adquisición de datos, sensores, el programa controlador y un computador.

3.2 EL PUERTO PARALELO

Cada computador tiene un puerto paralelo programable pre-instalado el cual hace posible añadir varios dispositivos que aceptan 8 bits de datos paralelos a niveles de voltaje estándares TTL. Este puerto puede funcionar como vía de entrada, vía de salida o vía de entrada/salida, generalmente sirve como medio de conexión para las impresoras. Un conector del tipo D25 provee acceso al puerto.

Los puertos paralelos son mapeados en memoria como puerto paralelo 1 (LPT1), puerto paralelo 2 (LPT2) o puerto paralelo 3(LPT3), siendo LPT1 el puerto predefinido. Los puertos ocupan diversas direcciones de memoria predefinidas (tabla 3.1).

Puerto ID	Dirección de Datos	Dirección de Status	Dirección de Control
LPT1	03BC	03BD	03BE
LPT2	0378	0379	037A
LPT3	0278	0279	027A

Tabla 3.1. Asignaciones de Dirección de Puertos Paralelos

El puerto paralelo usado (LPT1) definido para entrada/salida envia 8 bits de datos (D0-D7), 8 bits para control y recibe 8 bits de estatus, todos estos bytes están mapeados en posiciones de memoria consecutivas (tabla 3.1).

En nuestro caso solamente utilizamos 5 de 8 bits de salida y 5 bits de entrada los cuales están mapeados en la dirección de bits de status; no utilizamos los bits de control como se puede ver en la tabla 3.2.

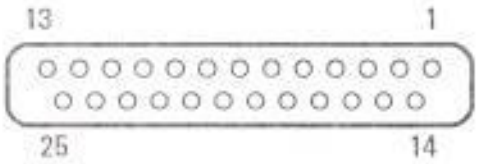


Fig.3.1. Conector tipo D25

Pin No.	I/O	Señal	Pin No.	I/O	Señal
1	I/O	NC	14	O	NC
2	I/O	C	15	I	DA
3	I/O	B	16	O	NC
4	I/O	A	17	O	NC
5	I/O	START	18	NA	Ground
6	I/O	SLCT	19	NA	Ground
7	I/O	NC	20	NA	Ground
8	I/O	NC	21	NA	Ground
9	I/O	NC	22	NA	Ground
10	I	DD	23	NA	Ground
11	I	EOC	24	NA	Ground
12	I	DC	25	NA	Ground
13	I	DB			

Tabla 3.2. Señales conectados a los pines

Es de anotar que de los 8 bits de estatus, 3 de ellos no están físicamente disponibles, pero deben ser considerados porque modifican los valores de los parámetros en la recolección de datos, ver tabla 3.3.

Byte Recibido Puerto 03BC		Byte Enviado Puerto 03BD	
BIT No.	Señal	BIT No.	Señal
7	- EOC	7	0
6	DD	6	0
5	DC	5	0
4	DB	4	SLCT
3	DA	3	START
2	1	2	C
1	1	1	B
0	1	0	A

Tabla 3.3 Detalle de Bytes Recibidos y Enviados

Durante la implementación de las conexiones entre la circuitería de adquisición de datos y el puerto paralelo surgieron problemas referentes a los niveles de voltaje presentes en el computador y en la circuitería por ser las tierras de ambos dispositivos diferentes, existiendo entre ellas una diferencia de potencial de 325 mV.

La diferencia de potencial existente entre ambas tierras hacia que los unos y ceros lógicos de la circuitería fueran recibidos como unos lógicos y niveles indeterminados por el CPU causando problemas en el handshaking y

de control realiza una conversión de los dos grupos de cuatro bits recibidos a un formato de 8 bits.

Existieron dos problemas en el handshaking entre computador y circuiteria, el primero fue que el computador ejecutaba las instrucciones de forma más rápida que el tiempo de la circuiteria para cambiar las señales lógicas, por lo que fue necesario poner un retardo (delay (100)) de 100 ms para asegurarnos que los datos recogidos eran los actuales y no los anteriores o de transición.

El segundo problema, tal vez el más difícil de resolver fue el de convertir dos datos de 4 bits en uno solo de 8 bits, tomando en cuenta que los datos recogidos eran los cuatro bits menos significativos y los cuatro bits más significativos del mismo resultado. Para resolver esto se desarrolló una fórmula a partir de varios datos muestreados, la cual se presenta a continuación:

$$Valor\ Total = b1/8 + b2*2$$

donde:

b1= Conjunto de bits menos significativos

b2= Conjunto de bits mas significativos

No hay que olvidar que los datos muestreados son recibidos como enteros de la forma:

$$b_{1,2} = 2^{D7} + 2^{D6} + 2^{D5} + 2^{D4}$$

siendo DD el MSB y DA el LSB.

Ejemplo:

Si b1=32 entonces b1(binario)=0010, refiriéndonos a la tabla 3.3, datos ingresados por el puerto paralelo, dirección de status, tenemos:

$$b1 = 0010 = 32$$
$$b1 = 0*2^3 + 0*2^4 + 1*2^5 + 0*2^6 = 32$$

$$b2 = 1001 = 72$$
$$b2 = 1*2^3 + 0*2^4 + 0*2^5 + 1*2^6 = 72$$

Si sumamos algebraicamente b1 y b2 para sacar un total;

Total = 32 + 72 =104

lo que no coincide con el total de los dos grupos de bits juntos y ordenados;

Total = 00101001 =148

pero aplicando la fórmula tenemos que:

$$Total = 32/8 + 72*2 = 4 + 144 = 148$$

lo cual nos da un resultado válido.

3.3.1 Algoritmo de comunicación con el puerto.

El algoritmo de comunicación con la circuitería de adquisición de datos es el siguiente:

- Envío dirección de parámetro a medir
- Envío señal de START al puerto
- Envío señal de 0 al puerto
- retraso 100ms
- Espero a que termine la conversión (senso EOC)
- Mando señal de SLCT=0
- Recibo los cuatro bits menos significativos y los almaceno
- Recibo los cuatro siguientes bits
- Calculo el total y almaceno

No hay que olvidar que las direcciones de los puertos están dadas en números Hexadecimales.

El programa de prueba está codificado en C, ya que este lenguaje presenta instrucciones que permiten enviar y recibir datos del puerto con notable facilidad. **Inport(), Outport()** las que tienen como parámetros la dirección del puerto, definidas en el encabezado **DOS.H**.

3.3.2 Codificación

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

void  enviar(int parametro)
{
 int port=0x03BC;
 outport(port,parametro);
}


int recibir()
{
 int port=0x03BD;
 int resultado=0;

 resultado=inportb(port);
 return(resultado);
}

main()
{
 int num;
 int res=0;
 int b1=0;
```

```
int b2=0;
int total=0;

clrscr();
for (num=0;num<4;num++)
{
 res=0;
 enviar(num);                    // envio direccion de parametro a medir
 enviar(num+8);                  // envio senial de start con la direccion
 enviar(0);
 delay(100);
 while (res<128)                 // senso el end of convertion
 res=recibir();
 enviar(0);                      // mando senial de select 0 (4 primeros datos)
 b1=recibir();                   // recibo los cuatro primeros bits
 if (b1>127) b1=b1-128;          // elimino el EOC
 b1=b1-7;                        // b1 primer dato puro
 enviar(16);
 b2=recibir();                   // recibo los siguientes cuatro bits
 if (b2>127) b2=b2-128;
 b2=b2-7;                        // b2 segundo dato puro
 total=(b1/8)+(b2*2);            // valor total de dato recibido
 printf("\n Datos a: %d    Dato b: %d    Total: %d \n",b1,b2,total);
 }
}
```

CAPITULO 4

ANALISIS Y DISEÑO DEL PROGRAMA DE MONITOREO

4.1 REQUERIMIENTOS GENERALES DEL SISTEMA

Para establecer los requerimientos generales del sistema fue necesario entrevistarme con varios profesionales involucrados en el trabajo en laboratorios de calidad de aguas tales como biólogos, ambientalistas, asistentes de laboratorio. De tales conversaciones extraigo los requerimientos de un sistema ideal para sus aplicaciones, los cuales en si son los requerimientos para cualquier sistema general de adquisición de datos. Tal vez la principal inquietud que reflejaban los entrevistados era la necesidad de elaborar un sistema de monitoreo en linea, de fácil manejo, barato y con una buena interfaz sistema-usuario, siendo éstos las principales metas de mi programa.

Los requerimientos generales del sistema son:

- Implementar una interfaz amigable con el usuario, preferentemente gráfica.
- Efectuar mediciones de parámetros en tiempo real.

- El sistema debe tener opciones de mediciones programadas de parámetros cada cierto tiempo.
- Se debe permitir al usuario parar el monitoreo en cualquier instante.
- Se debe permitir resetear el sistema sin salir del mismo.
- Presentar información en línea de los datos recopilados hasta el momento (reportes y medidores gráficos).
- Establecer niveles críticos de parámetros según la muestra monitoreada.
- Permitir la elaboración en línea de cuadros de análisis parámetro vs parámetro o parámetros vs tiempo.
- El sistema debe presentar opciones de suavización de gráficos obtenidos para así poder hacer proyecciones de comportamiento entre puntos o futuras.
- El sistema debe implementar opciones de impresión de reportes
- Los datos recopilados deben ser almacenados de tal forma que puedan ser recuperados por otros programas, hojas electrónicas preferentemente.
- Las estadísticas obtenidas deben contemplar niveles máximo medidos, niveles mínimos, valores medios.
- El sistema debe permitir grabar los datos recogidos y recuperar archivos de muestreo anteriores.

Aprovechando la oportunidad del diseño e implementación del software de control de monitoreo para el sistema SYSAQUA utilizaremos el método de programación orientado a objetos (OOP), por lo tanto el análisis, diseño e implementación del programa será basado en esta técnica de programación.

Del resultado de nuestro trabajo evaluaremos también el método de programación antes mencionado.

4.2 DISEÑO DEL SISTEMA

Atendiendo los requerimientos planteados de nuestro sistema, siguiendo la técnica de programación orientado a objetos, y tomando en cuenta como principal requerimiento una interfaz amigable para el usuario, se han definido las siguientes clases básicas.

- Clase Punto
- Clase Pantalla
- Clase Ventana
- Clase Parámetro
- Clase Grafico
- Clase Boton
- Clase Mouse

A partir estas clases y aprovechando las dos caracteristicas principales de programación orientada a objetos (encapsulación y herencia) se derivan varias

sub clases y objetos que heredan las funciones y variables de las clases padres, el siguiente gráfico de jerarquías ilustra los niveles de clases básicas y clases derivadas definidos para el sistema.

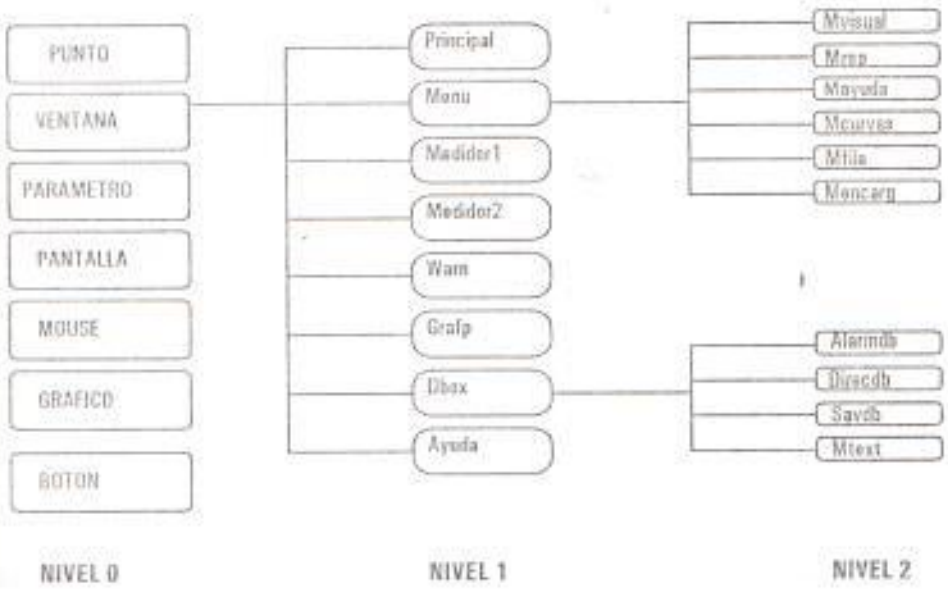


Fig. 4.1 Diagrama Jerárquico de Clases Utilizadas

4.2.1 Descripción de Clases.

La siguiente descripción de clases muestra el objetivo de la estructura, su constructor, su destructor, variables utilizadas y las descripciones de funciones implementadas para cada una de ellas.

En vista de que algunas clases no utilizan punteros ni localidades de memoria especiales y sabiendo que las clases solamente tienen vigencia dependiendo del

alcance (scope) determinado según el lenguaje de programación, existirán clases en las que no se consideren destructores.

4.2.1.1 Clase Punto.

Objetivo:

El objetivo de la clase Punto es de implementar una estructura que maneje coordenadas de pantallas tipo gráfico sin importar los tipos de tarjetas (resoluciones) de éstas, esto es, generando puntos referenciales que bien pueden funcionar para monitores tipo Hercules como VGA obteniendo un máximo de modularidad en la interfaz con el usuario. Además provee funciones de álgebra de coordenadas para facilitar el manejo de gráficos generados.

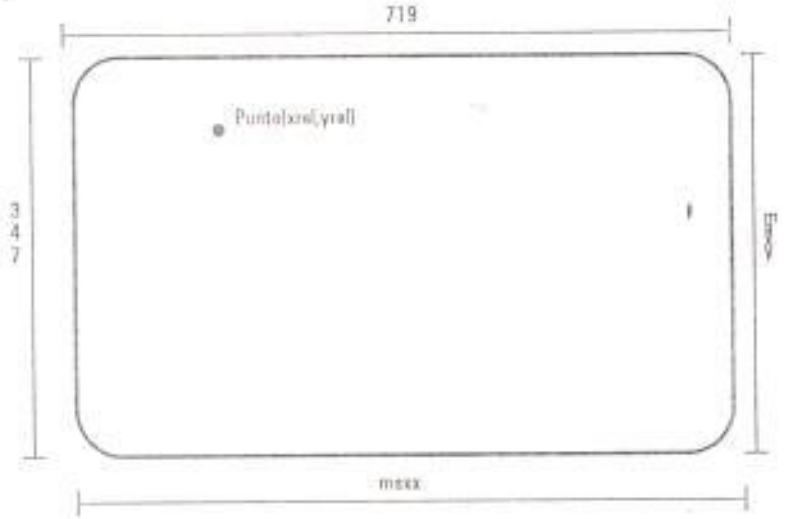


Fig. 4.2 Coordenadas Absolutas y Relativas respecto a la pantalla

Supongamos que en la pantalla dibujada (Fig. 4.2), tenemos un punto definido por sus valores relativos. La pantalla tiene como máximos relativos 719x347, entonces tendríamos los siguientes pares de valores:

xreal = xrel x maxx/719

yreal = yrel x maxy/347

xrelativo = xrel

yrelativo = yrel

Variables:

Las variables definidas para esta clase son las siguientes:

Variable	Tipo	Propósito
xval	entero	Coord. X real.
yval	entero	Coord. Y real.
xrel	entero	Coord. X relativa.
yrel	entero	Coord. Y relativa.

Las coordenadas reales se refieren a los valores de x y y verdaderos de acuerdo a la resolución provista por el monitor en uso, los valores relativos son las transformaciones de los valores reales respecto a las coordenadas del monitor tipo Hércules (719 x 347 pixels), se escogió el Hércules como referencia por ser en una máquina de ese tipo donde se empezó a desarrollar el proyecto.

Constructores:

Tenemos dos tipos de inicializadores, uno con parámetros y otro sin parámetros.

Punto()

Inicializador sin parámetros, inicializa las variables a cero.

Punto(x,y)

Inicializa automáticamente las variables según los valores pasados, tómese en cuenta que los valores de x, y son valores tomando como base la resolución (719 x 347) no tomando en cuenta la resolución real del monitor sea cual fuere.

xval=x*factorx

yval=y*factory

xrel=x

yrel=y

Destructor: Ninguno.

Funciones:

Factorx():

Función de tipo real, divide el máximo valor de x obtenido según el tipo de tarjeta gráfica, para el valor relativo 719 y retorna el factor de conversión para x.

Factor y():

Función de tipo real, divide el máximo valor de y obtenido según el tipo de tarjeta gráfica, para el valor relativo 347 y retorna el factor de conversión para y.

X():

Función que retorna el valor relativo de x.

Y():

Función que retorna el valor relativo de y.

xr():

Función de tipo real, retorna el verdadero valor de x, según la tarjeta gráfica usada.

yr():

Función de tipo real, retorna el verdadero valor de y, según la tarjeta gráfica utilizada.

operador+Punto p:

Función de tipo Punto, el cual retorna la suma de dos coordenadas de puntos en sus valores relativos.

operador-Punto p:

Función de tipo Punto, el cual retorna la resta de dos coordenadas de puntos en sus valores relativos.

4.2.1.2 Clase Pantalla

Objetivo:

Pantalla es una clase diseñada para manejar todos los recursos relacionados con pantallas gráficas tales como inicialización, cerrado de pantalla gráfica, colores, lineas, llenados, bordes, etc.

Variables:

Variable	Tipo	Propósito
maxX	entero	Guarda máximo valor de x
maxY	entero	Guarda máximo valor de y

Constructores:

Inicializar():

Inicializar guarda la inicialización de modo gráfico, detectando, mandando error y terminando el programa si existe problemas en la inicialización del modo.

Destructor:

Finalizar():

Cierra el modo gráfico y vuelve a modo texto la pantalla inicializada.

Funciones:

limpiar():

Función que no retorna valores, limpia la pantalla gráfica.

borde(Punto a, Punto b):

Dibuja un rectángulo tomando como limite superior las coordenadas de a y como limite inferior las coordenadas de b. La función no retorna valores.

rectangulo(entero fill, entero c, Punto a, Punto b):

Dibuja un rectángulo sin marco, lo llena con el patrón fill y el color c, en los limites entre a y b.

colorfondo():

Función que retorna el color de fondo actualmente utilizado.

colorfondo(entero cf):

Función que no retorna valores. Setea el color de fondo que se utilizará en el gráfico.

colordibujo():

Función que retorna el color que actualmente se utiliza en el dibujo.

colordibujo(entero dib):

Funcion que no retorna valores. Setea el color del dibujo a utilizarse por medio de la variable dib.

linea(Punto a, Punto b):

Función que no retorna variables. Dibuja una linea desde las coordenadas a hasta la coordenada b.

mover(Punto a):

Función que no retorna valores. Posiciona un pixel en las coordenadas indicadas por a.

lineaa(Punto a):

Dibuja una linea desde la posición actual del cursor hasta la posición indicada por las coordenadas del Punto a. No retorna valores.

visual(Punto a, Punto b):

Función que define un segmento de trabajo en la pantalla limitado por los puntos a y b. No retorna valores.

ponerpunto(Punto a, entero color):

Función que dibuja un punto de color determinado en las coordenadas dadas por a. No retorna valores.

texto (Punto pos, string texto):

Función que posiciona el string texto en las coordenadas dadas por pos. No retorna valores.

arco(Punto a, entero sa, entero ea, entero radio)

Función que dibuja un arco centrado en las coordenadas a, desde los ángulos sa a ea, de radio radio. No retorna valores.

llenar(Punto a, entero color)

Función que llena un área limitada por líneas de color.

4.2.1.3 Clase Mouse

Objetivo: Esta clase se encarga del manejo del ratón.

Variables:

Variable	Tipo	Propósito
globax	entero	Propósito General
globbx	entero	Propósito General
globcx	entero	Propósito General
globdx	entero	Propósito General

Estas variables de propósito general cambian de acuerdo a las funciones que se esté aplicando.

Constructores:

Mouse():

Este inicializador define un punto en las coordenadas inferior izquierda de la pantalla, encera las variables globales, reconoce y activa el driver del raton.

Destructor:

No tiene destructores.

Funciones:

accion(entero a, entero b, entero c, entero d):

Esta función es privada de Mouse, maneja las interrupción 0x33 de acuerdo a las variables a,b,c,d. El resultado de la interrupción sea cual fuere se guardan en las variables globax, globbx, globcx, globdx. Esta función no retorna valores.

display():

Esta función permite hacer un display del cursor del ratón en modo gráfico invocando a acción(1,0,0,0). No retorna valores.

esconder():

Esta función esconde el cursor del ratón, invoca a acción(2,0,0,0). No retorna valores.

coord():

Función de tipo Punto, retorna las coordenadas del cursor. Invoca a la función accion(3,0,0,0). Las coordenadas son almacenadas en las variables globcx y globdx, con estos valores transformados la función retorna una variable c de tipo punto.

tecla():

Función de tipo Punto, retorna las coordenadas donde se aplastó la tecla izquierda del ratón.

4.2.1.4 Clase Ventana

Objetivo:

La clase Ventana tiene como objetivo crear, restaurar y manejar sectores de pantalla definidos, en otras palabras areas de trabajo.

Variables:

Variable	Tipo	Propósito
s	Punto	Coordenada superior izquierda del sector
i	Punto	Coordenada inferior derecha del sector
colorf	entero	Color de fondo del sector
colord	entero	Color de dibujo para el sector

img	Puntero	Puntero a imagen del sector
tipo	caracter	Tipo de Ventana

Las ventanas pueden ser de dos tipos: Medidores ('M') o Ventanas ('V').

Los Medidores son sectores que para crearse no necesitan ser guardar el contenido previo de el sector a ocupar, en otras palabras permanecen fijos en pantalla, por lo tanto el puntero a imagen img no guarda ningun valor. Las ventanas son sectores que por su caracteristica temporal si necesitan guardar el contenido previo de pantalla y posteriormente restaurarlo, por lo que en este caso img si guardará un valor.

Constructores:

Ventana(Punto vs, Punto vi, entero vcolorf, entero vcolord, caracter tip) :

El constructor de ventana inicializa las variables caracteristicas de la clase, asignando:

s = vs

i = vi

colorf = vcolorf

colord = vcolord

img = NULL

tipo = tip

Ventana() :

El tipo de constructor sin parámetros es usado en situaciones en las cuales se quiere crear una ventana de tipo global, la cual será inicializada posteriormente dentro de alguna función local. No inicializa variables.

Destructores:

Cerrar() :

Cerrar es el destructor de la clase, su función es liberar el puntero img, disponer de su dirección de memoria. Es preciso notar que un factor crítico en el uso de interfaces puramente gráficas es el manejo de memoria, ya que las diferentes ventanas u objetos gráficos ocupan determinadas cantidades de memoria. Si la memoria ocupada no es liberada luego de usarse provoca con facilidad desbordamiento de la pila y por lo tanto consecuencias impredecibles en la corrida del sistema.

Funciones:

Abrir() :

Función que abre una ventana cuadrada con límites especificados por las variables de ventana s.i. Se verifica el tipo de ventana, si no es medidor entonces se guarda la porción correspondiente para restaurarla luego, a continuación se crea la ventana. La ventana abierta consta de un rectángulo lleno con el color de fondo bordeado de un marco del color del dibujo.

Esta función no retorna valores.

Abrirell(entero patron, entero color) :

Función que abre una ventana eliptica con limites especificados por las variables de ventana s,i. Se verifica el tipo de ventana, si no es medidor entonces se guarda la porción correspondiente para restaurarla luego, a continuación se crea la ventana. La ventana abierta consta de una elipse llena con el color de fondo bordeado de un marco del color del dibujo, esta elipse encierra a su vez una más pequeña llena con color y fondo especificados en los parámetros de la función.

Esta función no retorna valores.

Guardar() :

Guarda una porción de pantalla de acuerdo a las coordenadas de pantalla s,i. El contenido de pantalla guardado es asignado al puntero img. Esta función no retorna valores.

Imprimir(Punto a, string texto) :

Función que imprime un mensaje especificado por texto en las coordenadas a de la pantalla. Esta función no retorna valores.

Mensaje (string texto) :

Función que imprime un mensaje texto en las coordenadas relativas prefijadas (60,313). No retorna valores.

Mensaje1 (string texto) :

Función que imprime un mensaje texto en la última linea de la ventana definida. No retorna valores.

4.2.1.5 Clase Boton

Objetivo:

Esta clase tiene como objetivo dibujar el botón correspondiente a determinada función del programa. La figura del botón está especificada en un archivo .DIB.

Variables:

Variable	Tipo	Propósito
mbs	Punto	Posición superior izquierda del botón

Constructores:

boton(Punto in) :

Este constructor inicializa la variable mbs con el valor in.

boton() :

Este constructor no inicializa variables, su propósito es la de crear un botón global que se inicializará dentro de una función cualquiera.

Destructores:

Ninguno.

Funciones:

crear(string archivo) :

Función que guarda los atributos previos de pantalla (color de dibujo y de fondo) e invoca a Dibujar para la creación del botón. No retorna valores.

Dibujar(string archivo) :

Función que lee el archivo especificado por el parámetro y dibuja pixel por pixel el botón. No retorna valores.

4.2.1.6 Clase Parametro

Objetivo:

Esta clase define a cada parámetro por definir como un objeto. Cada parámetro tendrá funciones implícitas para captura de datos, control de activación de alarmas, valores medios, máximos y mínimos, control de valores críticos, número de mediciones obtenidas. En resumen esta clase maneja independientemente todo lo relacionado a la captura y procesamiento de parámetros.

Cabe anotar que los datos recogidos son guardados en estructuras dinámicas (arreglos dinámicos de punteros) permitiendo una eficiente utilización de localidades y montos de memoria. Además cada parámetro tiene un código numérico por el cual se puede referenciar.

Variables:

Es necesario definir la estructura datos;

```
struct datos
{
entero dp
entero alarma
puntero a datos sig
}
```

Esta estructura forma la celda básica del arreglo dinámico de valores para la clase parámetro, donde:

dp = valor del dato recogido

alarma = bandera de alarma para ese dato

sig = enlace a próxima celda tipo dato

El resto de las variables se definen como sigue:

Variables	Tipo	Propósito
codigo	entero	Codigo de parámetro
med	real	Valor medio
maximo	entero	Valor máximo
minimo	entero	Valor minimo
critico	entero	Valor critico

numero	entero	Número máximo de datos
ifalarm	entero	Condición de alarma
datacab	puntero a datos	Cabeza de arreglo dinámico
datapiv	puntero a datos	Variable pivote de arreglo dinámico
dataaux	puntero a datos	Variable auxiliar de arreglo dinámico

Constructores:

parametro(int code) :

Este constructor inicializa las variables de la siguiente manera:

codigo = code

med = maximo = minimo = critico = numero = 0

datacab = dataaux = datapiv = NULL

Destructores:

kput() :

Dispone de los punteros definidos y los pone a NULL cuando no son útiles.

Funciones:

inicializador() :

Reinicializa las variables cuando sea necesario cargar nuevos valores para los mismos parámetros sin necesidad de definir los objetos otra vez. No retorna valores.

captura(entero a) :

Captura el dato a, llena la estructura datos y envia a la función aniade para que el elemento sea anexado al arreglo. Además discrimina los valores máximos y mínimos, los valores medios y activa la alarma siempre y cuando el valor a sobrepase el valor crítico. Esta función no retorna valores.

captura(entero mp, entero mip, real medp, entero nump) :

Función que por medio de los parámetros pasados determina, respectivamente entero num máximo, mínimo, med, numero respectivamente. No retorna valores.

alarma() :

Función que setea a ifalarm=1. No retorna valores.

noalarma() :

Función que setea a ifalarm=0. No retorna valores.

entrega() :

Función de tipo puntero a la estructura datos, retorna un puntero a la variable datacab de la clase.

falarm() :

Función de tipo entero que retorna el valor de ifalarm;

flagnum() :

Función de tipo entero que retorna el valor de número.

max() :

Función tipo entero que retorna el valor máximo recibido.

min() :

Función tipo entero que retorna el valor mínimo recibido.

media() :

Función tipo entero que retorna el valor medio recibido.

crit(int c) :

Función que setea la variable crítico del parámetro. No retorna valores.

aniade(puntero a datos pter, entero i) :

Función que recibe un puntero a datos para ser anexado al arreglo dinámico, discrimina si es el primer elemento o elementos posteriores, según eso inicializa datacab o datapiv. No retorna valores.

recibe(puntero a datos pter) :

Recibe un puntero a la cabeza de un arreglo dinámico de elementos tipo datos, seteando de esta manera a datacab. No retorna valores.

medida() :

Función tipo entero que retorna el último valor medido del parámetro.

4.1.2.7 Clase Grafico

Objetivo:

La Clase Grafico tiene como objetivo principal el manejo de objetos que permitan dibujar gráficos en linea parámetro vs parámetro, elaborar suavizaciones y determinar ecuaciones suavizadas generadas desde el gráfico inicial.

Contiene funciones que permiten mejorar la presentación de los gráficos y un sistema de dibujo el cual ordena los datos y pone los puntos automáticamente escalados.

Variables:

Hemos definido una estructura que guardará los valores de los parámetros a graficar:

```
estructura dat
{
entero x
entero y
}
```

Los elementos de la estructura son los valores del parámetro 1 (x) y del parámetro 2 (y).

Las demás variables están especificadas como sigue:

Variable	Tipo	Propósito
codigo	entero	Código del gráfico
p1	entero	Código del parámetro 1
p2	entero	Código del parámtero 2
titulo	string	Titulo del Gráfico
textox	string	Texto del eje X
textoy	string	Texto del eje Y
coef	arreglo real	Coeficientes de la ecuación suavizada
intervalox	real	Intervalo de valores en X (escala x)
intervaloy	real	Intervalo de valores en Y (escala Y)
val	entero	Número de valores muestreados
dato	arreglo dat	Guarda los valores de los parámetros

Constructor:

grafico(entero a, entero b, entero code) :

Inicializador de variables:

codigo = code

p1 = a

p2 = b

grafico() :

Inicializador de variables codigo, p1 y p2 a 0.

Destructor:

Ninguno.

Funciones:

texto() :

Función que pide el texto del gráfico (titulo, textox, textoy). No retorna valores.

valores() :

Función que llena el arreglo dato con la información de los parámetros especificados por p1 y p2; x = p1, y = p2. No retorna valores.

senalar(Punto a) :

Función que señala con un circulo las coordenadas de los puntos a graficar a=Punto(x,y). No retorna valores.

dibujar() :

Función que dibuja el gráfico p1 vs p2. No retorna parámetros.

gtext() :

Función que imprime los titulos principales, etiqueta del eje X, etiqueta del eje Y en el dibujo. No retorna parámetros.

nombre() :

Función de tipo string que retorna la etiqueta de tipo **"(parametro1) vs (parametro2)"**.

ordenar()

Función que ordena el arreglo dato de acuerdo a los elementos x. No retorna valores.

La suavización del gráfico es realizada por el método de ***Interpolación y Polinomio de Lagrange*** con el cual es posible encontrar un polinomio aproximante especificando algunos puntos en el plano por los cuales deben pasar.

El **polinomio interpolante de Lagrange** se define por medio del siguiente teorema:

Si xo,x1,.....,x_n son (n+1) números diferentes y f es una función cuyos valores están dados por esos puntos, entonces existe un único polinomio P de grado a lo más n con la propiedad de que:

$$f(x_k) = P(x_k) \qquad \text{para cada} k = 0,1,2,3,4,\ldots\ldots,n.$$

El polinomio está dado por:

$$P(x) = f(x_0)L_{n,0}(x) + \ldots\ldots\ldots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x),$$

donde:

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\ldots\ldots(x-x_{k-1})(x-x_{k+1})\ldots\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} = \prod_{i=0, i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)}$$

para cada k=0,1,2,3,...,n

Elaborando un algoritmo para este método de interpolación tenemos:

- Determinar el valor n
- Lazo hasta la repetición número n
- Determinar el valor de k
- Calcular el numerador de $L_{n,k}$
- Guardar los coeficientes del numerador en *arreglo1*
- Calcular el denominador de $L_{n,k}$
- Dividir coeficientes del *arreglo1* para el denominador obtenido
- Multiplicar los elementos del *arreglo1* por *f(k)*
- Sumar el valor final de *arreglo1* a *coef*

Por la complejidad de cada una de las tareas que debe efectuar el algoritmo, se lo ha dividido en las siguientes funciones propias de la clase Grafico.

coeficientes()

Función que asigna los coeficientes obtenidos por medio de la interpolación al arreglo coef. No retorna valores.

denominador(entero num)

Función que determina el denominador de $L_{n,k}$, divide los elementos de arreglo1 para el valor obtenido. Posteriormente estos valores son multiplicados por los valores Y, *f(x)* de las coordenadas. No retorna valores.

numerador(entero num, entero num1) :

Función que calcula los coeficientes del numerador de $L_{n,k}$ y los asigna a arreglo1. No retorna valores.

l(entero n) :

Función que calcula el polinomio $L_{n,k}$. Invoca a numerador, denominador y coeficientes. No retorna valores.

suavizar() :

Función que determina el valor n, realiza el lazo *0 a n* y obtiene los coeficientes suavizados, invoca dentro del lazo a la función l(entero n). No retorna valores.

scoef() :

Funcion que muestra los coeficientes de los polinomios obtenidos por la interpolación de puntos. No retorna valores.

sgraf() :

Función que grafica el polinomio obtenido de la interpolación de puntos. No retorna valores.

4.2.1.8 Clase Trian

Objetivo:

Esta clase implementa una alarma visual y sonora.

Variables:

Variable	Tipo	Propósito
base	Punto	Coordenadas del centro de la alarma visual
r	entero	Radio del botón de alarma
tf	entero	Color de fondo de la alarma

Constructores:

Trian(Punto c, entero a) :

Inicializa las variables base, r, tf=GREEN.

Trian() :

Constructor que no inicializa variables, declara objetos globales que se inicializarán posteriormente.

Destructores: Ninguno.

Funciones:

dibujo(int a) :

Función que dibuja la alarma con el color especificado en a. No retorna valores.

blink() :

Función que maneja el destello de la alarma y produce el sonido de alerta. No retorna valores.

4.2.1.9 Clase medidor1 (Clase derivada de clase Ventana)

Objetivo:

Esta clase implementa un medidor gráfico de barras.

Variables:

Variable	Tipo	Propósito
divisiones	entero	Número de divisiones del medidor
colorf2	entero	Color de fondo 2
mcold	entero	Color de dibujo
ms	Punto	Punto interno superior
mi	Punto	Punto interno inferior
ms1	Punto	Punto externo superior
mi1	Punto	Punto externo inferior
nombre	string	Nombre del medidor

Constructores:

medidor1(string nom, entero div, entero colf, entero colf2, Punto in, Punto f) :

Inicializa una Ventana de coordenadas in,f, de color de fondo colf, color de dibujo cold y de tipo 'm'.

Inicializa las variables divisiones=div, colorf2=colf2, mcold=cold, ms1=in, mi1=f, nombre=nom, ms=ms1+Punto(6,6), mi=mi1-Punto(6,10).

medidor1() :

Inicializa una ventana con parámetros 0 y divisiones=0, el objeto definido por este constructor se definirá con valores reales posteriormente.

Destructores:

Ninguno.

Funciones:

dibescalas() :

Dibuja un medidor gráfico de barras con el número de escalas indicado por num. No retorna valores.

marcar(entero max) :

Marca en el medidor gráfico hasta un valor definido por el parámetro max. No retorna valores.

rayar(entero num, entero max) :

Raya las divisiones del medidor num divisiones de máximo de divisiones. No retorna valores.

4.2.1.10Clase **medidor2 (Clase derivada de clase Ventana)**

Objetivo:

Esta clase implementa un medidor tipo digital.

Variables:

Variables	Tipo	Propósito
m2s	Punto	Punto superior externo
m2i	Punto	Punto inferior externo
m2as	Punto	Punto superior interno
m2ai	Punto	Punto inferior interno
colorf2	entero	Color de fondo externo
colorf1	entero	Color de fondo interno
mcold	entero	Color de dibujo
texto	string	Nombre del medidor

Constructores:

medidor2(entero colf, entero colf2, entero cold, Punto in, Punto fin) :

Inicializa una Ventana de coordenadas in,fin, de color de fondo colf, color de dibujo cold y de tipo 'm'.

Inicializa las variables m2s=in, m2i=fin, colorf2=colf2, mcold=cold, colorf1=colf, m2as=m2s+Punto(3,3), m2ai=m2i-Punto(3,3).

medidor2() :

Inicializa una ventana con parámetros 0, el objeto definido por este constructor se definirá con valores reales posteriormente.

Destructores:

Ninguno

Funciones:

crear(string nombre) :

Función que dibuja el medidor digital e imprime el nombre del medidor. No retorna valores.

actualizar(string texto) :

Función que actualiza los valores del display digital dado en texto. No retorna valores.

4.2.1.11Clase Menu (Clase derivada de clase Ventana)

Objetivo:

Esta clase implementa un menú de opciones, permite la selección de las mismas y su posterior ejecución.

Variables:

Entre las variables de esta clase se define una estructura llamada item, la cual guarda el nombre de las opciones y su posición en la ventana de menú. Su descripción es como sigue.

```
estructura item
{
Punto posición
string nombre
}
```

Variables	Tipo	Propósito
menmax	entero	Número máximo de opciones
mencolorf	entero	Color para resaltar las opciones
mencolord	entero	Color de las letras
mens	Punto	Límite superior del menú
meni	Punto	Límite inferior del menú
lista	arreglo de item	Nombre y posición de las opciones

Constructores:

menu(Punto mns, Punto mni, entero mncolorf, entero mncolord, entero max) :

Inicializa una Ventana de coordenadas mns,mni, color de fondo mncolorf, color de dibujo mncolord y de tipo 'v'.

Inicializa las variables: ens = mns; meni = mni; menmax = max; mencolorf = mncolorf; mencolord = mncolord.

Destructores:

fin() :

Destruye el objeto Menu por medio del destructor de Ventana Cerrar().

Funciones:

ejecutar(entero opcion):

Función de tipo virtual, se encarga de ejecutar la opción escogida en activar. No retorna valores.

poner(Punto p, string texto) :

Función que imprime el parámetro texto en la posición definida por p. No retorna valores.

piloto() :

Función que maneja a las demás funciones en la creación, activación y ejecución de las opciones del menú. No retorna valores.

crearmenu() :

Función que crea la ventana de menú e imprime la lista de opciones (variable lista). No retorna valores.

cambiar(entero xmen, entero ymen) :

Función por medio de la cual se deja de sombrear la posición establecida por ymen y sombrea la posición indicada por xmen; esto lo hace invocando a marcaritem(entero, entero). No retorna valores.

marcaritem(entero pitem, entero coloritem) :

Función que pinta la opción especificada por el parámetro pitem con el color establecido por el parámetro coloritem. Si coloritem=BLACK entonces se sombrea la posición, si coloritem=WHITE deja de sombrearse la opción. No retorna valores.

activar(entero mx) :

Función que retorna un valor entero. El algoritmo de esta función hace que el programa se sumerja en un lazo en el cual se van monitoreando las teclas presionadas, si la tecla es ENTER la función retorna la posición de la opción escogida, si la tecla es ESC retorna el valor de 100.

4.1.12Clase Dbox (Clase derivada de clase Ventana)

Objetivo:

Esta clase implementa una ventana temporal tipo dialogo (Dialog Box).

Variables:

Se necesitan dos tipos de estructura: item1 para las etiquetas de las opciones y la estructura opt para el contenido de las etiquetas.

```
estructura item
{
Punto posicion
string nombre
}

estructura opt
{
Punto opcpos
caracter tipo
entero lon
string contenido
}
```

Variable	Tipo	Propósito
dbmax	entero	Máximo número de opciones
dbs	Punto	Limite superior de Ventana
dbi	Punto	Limite inferior de Ventana
titulo	string	Titulo del Dbox
lista	arreglo tipo item1	Lista de etiquetas de opciones de Dbox
opciones	arreglo tipo opt	Lista de contenido de opciones de Dbox

Constructores:

dbox(Punto ds, Punto di, string tit, entero dmax) :

Constructor que define un objeto Ventana de coordenadas ds,di, color de fondo Gris Claro, color de dibujo Negro y tipo 'v'.

Inicializa las variables:

dbs = ds

dbi = di

dbmax = dmax

titulo = tit

lista se inicializa con valores nulos para sus elementos.

Destructores:

cerrardb() :

Cierra el dbox por medio del destructor de ventana Cerrar().

Funciones:

creardb() :

Dibuja el dialog box, imprime las etiquetas y el contenido de las opciones, finalmente activa el dialog box (permite el llenado de datos). No retorna valores.

activardb() :

Función que permite navegar e ingresar datos en el dialog box. No retorna valores.

posdb(entero opcion, entero opcant) :

Función que dibuja el cursor dentro del dbox de acuerdo a las posiciones definidas por el parámetro opcion y borra el cursor de la posición anterior definida por opcant. No retorna valores.

datadb(caracter c, entero nop) :

Función que recoge los datos de las opciones y las presenta en pantalla. Valida también que los datos digitados estén en concordancia con el tipo definido de la opción, el parámetro nop indica el número de opción a tipear. Retorna 1 si la tecla tipeada fue la última permitida por la extensión de la opción o 0 si se aplastó un ENTER.

ponerdatodb(entero popdb) :

Función que imprime el contenido de una opción en junto a su respectiva etiqueta en el dbox. No retorna valores.

limpiarproc(entero popi) :

Función encargada de limpiar la opción del dbox para permitir el ingreso de un nuevo dato. No retorna valores.

4.2.1.13Clase Warn (Clase derivada de clase Ventana)

Objetivo:

La clase Warn fue implementada con el propósito de contar con una ventana temporal que muestre un mensaje de advertencia para el usuario, este mensaje desaparecerá de pantalla solamente cuando el operador tenga pleno conocimiento del mismo presionando una tecla de ok.

Variables:

Variable	Tipo	Propósito
texto	string	Mensaje a presentar
num	entero	Número de ventana

Constructores:

warn(entero i) :

Inicializa una Ventana de coordenadas predefinidas (150,74); (550,200); color de fondo Gris Claro; color de dibujo Negro; de tipo ventana temporal 'v'.

Inicializa las variables:

texto = nada

mnm = i

Destructor:

Utiliza el destructor de Ventana Cerrar()

Funciones:

warnver(string texto) :

Crea la ventana de advertencia con el mensaje texto, espera a que el usuario presione la tecla de ok para después cerrarla. No retorna valores.

4.2.1.14 Clase Ayuda (Clase derivada de clase Ventana)

Objetivo:

La clase Ayuda implementa una ventana temporal en la cual se imprime el texto explicativo correspondiente a la sección del programa donde nos encontremos.

Variables:

Variable	Tipo	Propósito
estado	entero	Número de ventana de ayuda.

Constructor:

ayuda(entero stat)

Inicializa una ventana de coordenadas predefinidas (150,74); (550,270); color de fondo Gris Claro, color de dibujo Negro, tipo ventana temporal 'v'.

Inicializa la variable estado=stat.

Destructor:

Utiliza el destructor de Ventana Cerrar().

Funciones:

ver()

Función que crea la pantalla de ayuda, hace el display del mensaje almacenado en un determinado archivo (Ayuda.hlp) y espera que el usuario presione cualquier tecla. La variable stat indica en que parte del archivo esta el mensaje buscado y el carácter @ nos indica el fin del mensaje. No retorna valores.

1.15Clase grafp (Clase derivada de clase Ventana)

Objetivo:

Esta clase fue diseñada para implementar la pantalla base que servirá para la presentación de curvas del sistema. Maneja las diferentes opciones Lista de gráficos , Suavizar gráficos, Texto y Salir.

Variables:

Variables	Tipo	Propósito
ps	Punto	Limite superior de pantalla
pi	Punto	Limite inferior de pantalla

Constructores:

grafp(Punto prs, Punto pri, entero prcolorf, entero prcolord) :

Función que define una Ventana de coordenadas prs, pri; color de fondo prcolorf, color de dibujo pprcolord, y tipo permanente 'm'. Se define como permanente porque las limitaciones de memoria impiden guardar el contenido previo de pantalla, o sea impide definirla como temporal.

Inicializa las variables ps=prs y pi=pri.

Destructor:

Ninguno. Utiliza el mismo destructor de Ventana, Cerrar().

Funciones:

crearpantalla(entero t) :

Función encargada de la definición de curvas y de dibujar la pantalla de gráficos con sus diferentes opciones. El parámetro t indica si se ha definido previamente las curvas, si no se han definido obliga al operador a definirlas, caso contrario permite la entrada a la selección de opciones.

gmen(Punto a, string texto, entero marca) :

Función diseñada para dibujar los botones del menú principal, crea los botones en las coordenadas dadas por a, imprime el parámetro texto, marca=1 indica botón presionado, marca=0 indica botón normal. No retorna valores.

marcarpant(entero a, entero color) :

Función que marca los diferentes botones del menú, el parámetro a define el texto y su posición, el parámetro color indica si el botón es aplastado. Los parámetros anteriormente definidos pasan al invocar a bmenu(Punto a, texto, color). No retorna valores.

manejar() :

Función que maneja la navegación entre los botones del menú y la ejecución de las opciones en un lazo que se rompe solo con la opción Salir. Recoge la tecla presionada y produce la sensación de movimiento en el menú pull-down del sistema por medio de la función marcarpant(entero a, entero color), si se presiona ENTER manda a ejecutar la opción elegida invocando a Ejec1(int op) o Ejec2(int op), si se trata de la opción Salir sale del lazo y termina la función.

En realidad es esta función la que controla la ejecución del programa en sí. No retorna valores

Ejec1(entero a) :

Función que es llamada por manejar(). Se encarga de ejecutar las opciones relacionadas con el menú de Lista(a=1) y Suavizar(a=2). No retorna valores.

Ejec2(entero a) :

Función invocada por manejar(), se encarga de ejecutar las opciones relacionadas con el menú de Texto(a=3) y Salir(a=4). No retorna valores.

- He puesto dos funciones separadas para ejecutar las opciones de la pantalla curvas, debido a que cada una de estas define a su vez un conjunto de objetos tipo menú, dbox o grafico. Si se definen los objetos usados para la ejecución en una sola función se produce un STACK OVERFLOW debido al alto consumo de memoria de cada uno de los objetos por lo tanto me veo en la necesidad de distribuirlos en dos funciones.

4.2.1.16 Clase Principal (Clase derivada de clase Ventana)

Objetivo:

Esta clase implementa una pantalla base en la cual van a estar los medidores, alarmas, además maneja todas las opciones principales y las ejecuta. Es de notar la función manejar() la cual coloca al sistema en un lazo que se rompe al seleccionar la opción Salir, terminando el programa.

Variables:

Variables	Tipo	Propósito
ps, pi	Punto	Límites de la pantalla
ph,phm	medidor1	Medidor de barra
tiempo,tiempom,temp, temp, redox, redoxm, conduct, conductm, salinidad, salinidadm	medidor2	Medidor Digital
tph,ttemp,tredox,tcon,tsali	Trian	Alarmas
nomarchivo	string	Nombre del Archivo

Constructores:

Principal (Punto prs,Punto pri, entero pcolorf, entero pcolord)

Este constructor inicializa una Ventana de dimensiones prs, pri, con colr de fondo prcolorf y color de dibujo prcolord. Además inicializa las variables ps y pi

Destructores:

Ninguno propio, usa los destructores de Ventana.

Funciones:

reinicializar()

Función que reinicializa los medidores 1 y 2, los parámetros definidos, la variable texto global y deja a Principal listo para redibujar la pantalla. No retorna valores.

inicializar() :

Función que define e inicializa los medidores, parámetros variables globales.

No retorna valores.

dibujarmedidores() :

Función utilizada para dibujar medidores y botones en la pantalla principal. No retorna valores.

crearpantalla() :

Función encargada de crear la pantalla principal. Invoca a Abrir(), bmenu(), dibujarmedidores() y marcarpant(). No retorna valores.

bmenu(Punto a, string texto, entero pa) :

Función diseñada para dibujar los botones del menú principal, crea los botones en las coordenadas dadas por a, imprime el parámetro texto, pa=1 indica botón presionado, pa=0 indica botón normal. No retorna valores.

marcarpant(entero a, entero color) :

Función que marca los diferentes botones del menú, el parámetro a define el texto y su posición, el parámetro color indica si el botón es aplastado. Los parámetros anteriormente definidos pasan al invocar a bmenu(Punto a, texto, color). No retorna valores.

actmed() :

Función que actualiza los medidores de la pantalla de acuerdo a los valores muestreados por los parámetros. No retorna valores.

manejar():

Función que maneja la navegación entre los botones del menú y la ejecución de las opciones en un lazo que se rompe solo con la opción Salir. Recoge la tecla presionada y produce la sensación de movimiento en el menú pull-down del sistema por medio de la función marcarpant(entero a, entero color), si se presiona ENTER manda a ejecutar la opción elegida invocando a Ejec1(int op) o Ejec2(int op), si se trata de la opción Salir sale del lazo y termina la función. En realidad es esta función la que controla la ejecución del programa en si. No retorna valores.

veralarma():

Función de tipo entero que prueba si es que el valor de algún parámetro ha sobrepasado el nivel crítico y si es así dispara la correspondiente alarma. Retorna uno si se ha activado una alarma, en caso contrario retorna cero.

Ejec1(entero a):

Función que es llamada por manejar(). Se encarga de ejecutar las opciones relacionadas con el menú de Archivo(a=1) y Curvas(a=2). No retorna valores.

Ejec2(entero a):

Función invocada por manejar(), se encarga de ejecutar las opciones relacionadas con el menú de Reportes (a=3), Monitoreo(a=4), Ayuda(a=5) y Salir(a=6). No retorna valores.

- He puesto dos funciones separadas para ejecutar las opciones del menú, debido a que cada una de estas define a su vez un conjunto de objetos tipo

menú o dbox principalmente. Si defino todos los objetos usados para ejecución en una sola función se produce un STACK OVERFLOW debido al alto consumo de memoria de cada uno de los objetos definidos, por lo tanto me vi en la necesidad de distribuirlos en dos funciones.

7Clase Mfile (Clase derivada de clase Menu)

Objetivo:

Clase que define el menú de Archivos con su respectivas opciones, las maneja y las ejecuta. La ejecución la realiza por medio de la función tipo virtual ejecutar(opción).

Variables:

Variable	Tipo	Propósito
num	entero	Número de ventana
nombarch	string	Nombre del archivo de trabajo

Constructores:

mfile(entero n)

Inicializador de la clase mfile, define un menú con las siguientes caracteristicas: Limites superior e inferior (70,24), (250,80); color de fondo Gris Claro, color de dibujo Negro, número de opciones 4.

Inicializa a la variable num con el valor n.

ructor:

iza el mismo destructor que la clase Menu.

nciones:

pilot() :

nción de entrada para mvisual, invoca a mostrar() y a piloto() de menu, etorna un valor entero que indica si es que se abortó cualquiera de las peraciones indicadas por el menú.

nostrar() :

unción que define las posiciones y etiquetas de las diferentes opciones resentes en el menú. No retorna valores.

jecutar (entero opcion) :

Función virtual que manda a ejecutar la opción elegida por el usuario invocando a ej1(), ej2(), ej3(), ej4() dependiendo del parámetro opción. No retorna valores.

ej1() :

Función que ejecuta la primera opción del menú (Grabar), relacionada con grabar un archivo. No retorna valores.

ej2() :

Función que ejecuta la segunda opción del menú (Abrir), relacionada con recuperar un archivo. No retorna valores.

ej3()

Función que ejecuta la tercera opción del menú (Nuevo), relacionada con desechar los datos anteriores y dar paso a nueva información. No retorna valores.

ej4()

Función que ejecuta la cuarta opción del menú (Observaciones), relacionada con escribir comentarios relacionados con la muestra y el archivo. No retorna valores.

nombre()

Función de tipo caracter, retorna el string guardado por nombarch, en otras palabras el nombre del archivo de trabajo.

4.2.1.18Clase Mcurvas (Clase derivada de Clase Menu)

Objetivo

Clase que define el menú de graficación de curvas con su respectivas opciones, las cuales maneja y ejecuta. La ejecución es realizada por medio de la función tipo virtual ejecutar(opción).

Variables:

Variable	*Tipo*	*Propósito*
num	entero	Número de Menu

Constructor:

mcurvas(entero n) :

Inicializador de la clase mcurvas, define un menú con las siguientes características: Límites superior e inferior (200,24), (420,60), color de fondo Gris Claro, color de dibujo Negro, número de opciones 2.

Inicializa a la variable num con el valor n.

Destructor:

Utiliza el mismo destructor que la clase Menu.

Funciones:

mostrar() :

Función que define las posiciones y etiquetas de las diferentes opciones presentes en el menú, invoca además a piloto. Retorna la opción escogida por el usuario.

ejecutar (entero opcion) :

Función virtual que manda a ejecutar la opción elegida por el usuario invocando a ej() dependiendo del parámetro opción. No retorna valores.

ej() :

Función que permite definir gráficos si no han sido pre-definidos, si se intenta graficar sin existir gráficos predefinidos aparece un mensaje de advertencia. No retorna valores.

2.1.19Clase Mayuda (Clase derivada de clase Menu)

Objetivo:

Clase que define el menú de Ayuda con su respectivas opciones, las maneja y las ejecuta. La ejecución la realiza por medio de la función tipo virtual ejecutar(opción).

Variables:

Variable	Tipo	Propósito
num	entero	Número de ventana

Constructores:

mayuda(entero n)

Inicializador de la clase mfile, define un menú con las siguientes características: Límites superior e inferior (400,24), (600,80); color de fondo Gris Claro, color de dibujo Negro, número de opciones 3.

Inicializa a la variable num con el valor n.

Destructor:

Utiliza el mismo destructor que la clase Menu.

Funciones:

mostrar()

Función que define las posiciones y etiquetas de las diferentes opciones presentes en el menú. No retorna valores.

ejecutar (entero opcion)

Función virtual que manda a ejecutar la opción elegida por el usuario invocando a ayuej(), dependiendo del parámetro opción. No retorna valores.

ayucj()

Función que pone una pantalla de ayuda (de clase ayuda) y la cierra. No retorna valores.

4.2.1.20Clase Mrep (Clase derivada de clase Menu)

Objetivo:

Clase que define el menú de Reportes con su respectivas opciones, las maneja y las ejecuta. La ejecución la realiza por medio de la función tipo virtual ejecutar(opción).

Variables:

Variable	Tipo	Propósito
num	entero	Número de ventana
hinic	entero	Hora inicial de muestreo

minic	entero	Minuto inicial de muestreo
dmin	entero	Intervalo de muestreo

Constructores:

nrep(entero n) :

Inicializador de la clase nrep, define un menú con las siguientes características: Límites superior e inferior (300,24), (450,60); color de fondo Gris Claro, color de dibujo Negro, número de opciones 2.

Inicializa a la variable num con el valor n.

Destructor:

Utiliza el mismo destructor que la clase Menu.

Funciones:

mostrar() :

Función que define las posiciones y etiquetas de las diferentes opciones presentes en el menú. Activa al menú por medio de la función piloto. Retorna el valor de la opción escogida.

hora() :

Función que retorna la hora del sistema en caracteres.

fecha() :

Función que retorna la fecha del sistema en caracteres.

shora() :

Función que retorna el intervalo de horas entre muestreo en caracteres.

ihora() :

Función que convierte hora y minutos iniciales a enteros.

ejecutar (entero opcion) :

Función virtual que manda a ejecutar la opción elegida por el usuario invocando a ej1(), ej2(), dependiendo del parámetro opción. No retorna valores.

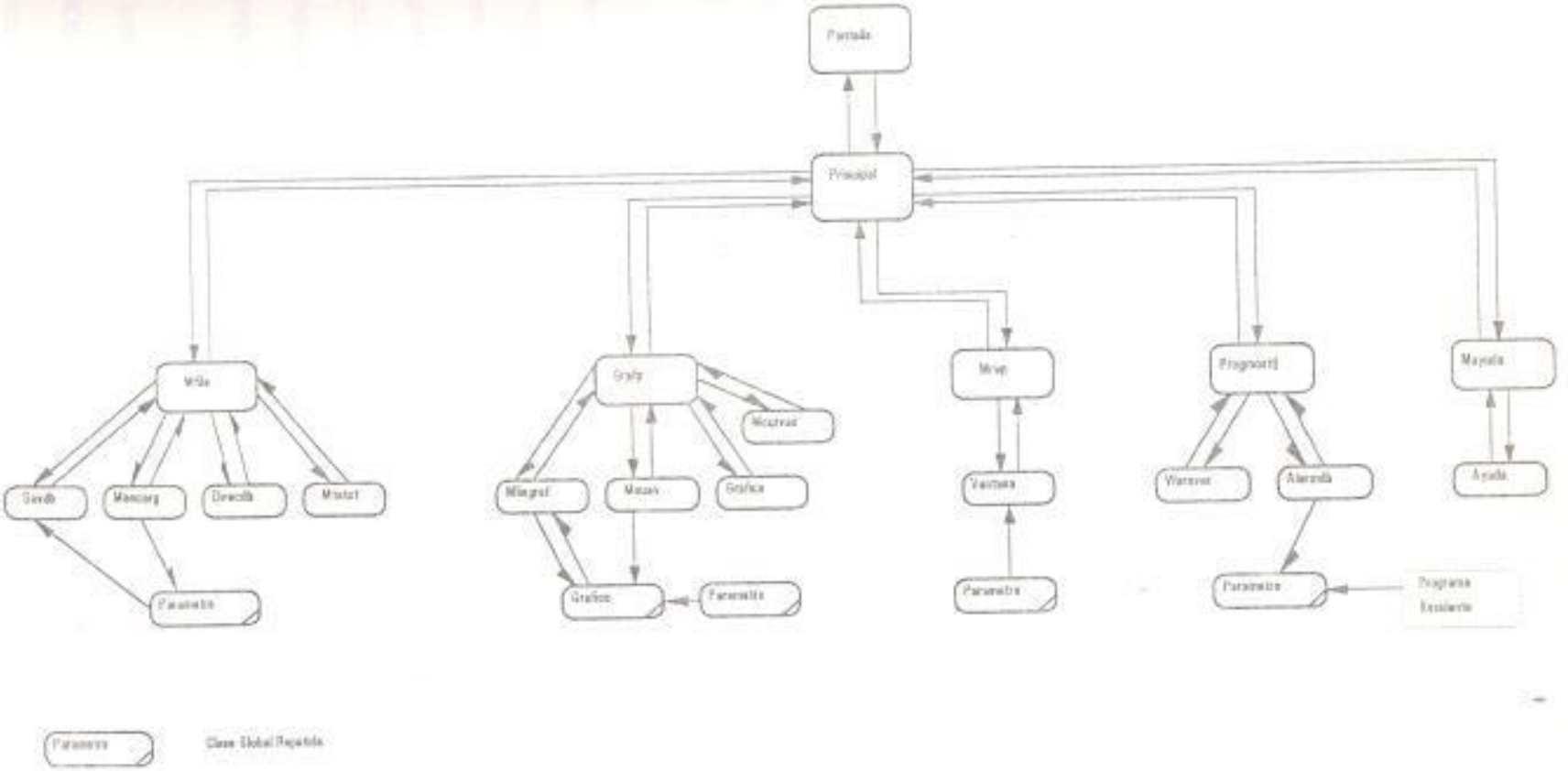
ej1() :

Función que ejecuta la impresión de reportes por pantalla. No retorna valores.

ej2() :

Función que ejecuta la salida de reportes por impresora. No retorna valores.

4.3 ESQUEMA DE INVOCACION DE CLASES, OBJETOS Y SUBPROGRAMAS



4.4 MONITOREO RESIDENTE DE DATOS.

Una de las principales características del programa es su capacidad de recoger datos automáticamente cada cierto tiempo pre-programado, esta característica lo hace superior a muchos otros programas de adquisición de datos en los cuales el operador debe aplastar cierta tecla al momento de hacer la recolección del parámetro.

La capacidad antes mencionada es lograda por medio de la activación de un programa residente en memoria, el cual fue desarrollado a partir del redireccionamiento del vector de interrupción 1CH correspondiente al tic del reloj interno.

Uno de los problemas afrontados durante la implementación del programa residente fue el darnos cuenta que la interrupción 1CH es demasiado rápida para que cada vez que se active (18.2 veces por segundo) permita la ejecución de un pequeño programa, por lo cual es necesario hacer que la rutina asociada a esta interrupción sea lo más pequeña posible, además con esto nos aseguramos que el sistema en conjunto no tenga retardos de ningún tipo.

También es de anotar que se debe considerar que el sistema operativo DOS no presenta facilidades de reentrancia para las invocaciones recursivas a programas residentes, esto es otro de los limitantes para que un programa pequeño pueda correr efectivamente mediante una activación de 18.2 veces por segundo.

Aun así es imposible tener una rutina de adquisición de datos lo suficientemente pequeña para asociarla con el tic del reloj, por lo cual la recolección de datos propiamente lo hace la parte no residente del sistema. La parte residente se limita a detectar si es tiempo o no de muestrear el dato. Si detecta que es tiempo de recoger datos entonces le comunica a la parte no residente para que lo haga.

4.4.1 Algoritmo Residente

A continuación se explica detalladamente el funcionamiento de la parte residente y su relación con la parte no residente.

Antes que nada es necesario precisar que existen dos variables localizadas en posiciones fijas de memoria, la primera está en la localidad 0x0040 offset 0x0110, indica cual es el intervalo de tiempo en el cual se debe hacer el muestreo. La segunda localizada en 0x0040 offset 0x0100 indica a la parte no residente si se debe o no muestrear.

Es importante notar una tercera variable declarada externa pero local para la rutina residente la cual lleva en cuenta el número de invocaciones a la rutina residente. Gracias a esta variable se puede saber si se ha cumplido o no con el intervalo de tiempo para muestrear comparándola con la variable de intervalo.

Algoritmo residente:

- Recuperar la dirección de memoria 0x0040 offset 0x0110, el intervalo de tiempo
- Asignar el intervalo de tiempo a una variable
- Incrementar en uno el contador de tiempo
- Comparar si el contador de tiempo es igual a intervalo de tiempo
- Si es igual guarde en la dirección de memoria 0x0040 offset 0x0100 un 1, poner en cero el contador de tiempo

4.2 Comportamiento de la parte no residente:

El siguiente algoritmo está inmerso en todos aquellos lazos o estados del programa que signifiquen retardos en tiempo, por ejemplo la espera para que se digite una tecla en las clases principal, menú, dbox, etc.

- Lazo principal
- Lazo secundario hasta que se digite una tecla
- Obtenga el valor de muestreo desde la dirección de memoria
- Si el valor de muestreo es uno realice recolección de datos y ponga en la dirección de memoria 0x0040 offset 0x0100 un cero.

Dependiendo de donde se encuentre, este pequeño algoritmo ejecutará operaciones secundarias como por ejemplo la actualización de los medidores en la pantalla principal.

4.4.3 Consideraciones de lenguaje para la rutina residente:

Considere, después de hacer varias pruebas entre programas residentes escritos en Asembler y C++, que C++ presenta una manera más fácil y rápida para su desarrollo, claro que sin la eficiencia que provee Asembler, pero para mi caso la eficiencia y rapidez ofrecida por C++ era perfecta.

La rutina residente se encuentra separada del programa principal, por lo cual también fue compilada como un ejecutable independiente; este pequeño programa redirecciona la rutina asociada con la interrupción 1CH a mi rutina, es de notar que la parte residente solamente esta formado por 5 líneas, las cuales representan un tiempo de ejecución que no retarda la ejecución de otros programas en el mismo entorno DOS. Esta aclaración la hago porque una vez cargado el programa residente, este permanece en memoria (aunque en estado inactivo) hasta que se resetee la máquina.

A continuación se presenta la codificación del programa residente implementado para el sistema.

```
#include<stdio.h>
#include <dos.h>
#define C 10h
int muestrita=0;
int flag=1;
```

```
int limite=0;

void interrupt muestrear(void)
{
 asm push ds;
 limite=peek(0x0040,0x0110);
 if (limite!=0) muestrita++;
 if (muestrita==limite)
 {
  muestrita=0;
  poke(0x0040,0x0100,1);
 }
 asm pop ds;
}


  void instalar ()
  {
   union REGS  Inr,Outr;
   struct SREGS  segreg;
   char far*    mifunc;

   mifunc=  (char far*) muestrear;
   Inr.h.ah=0x25;
   Inr.h.al=0x1C;
   segreg.ds=FP_SEG(mifunc);
```

```
Inr.x.dx=FP_OFF(mifunc);
intdosx(&Inr,&Outr,&segreg);
Inr.x.ax=0x3100;
Inr.x.dx=0x500;
intdos(&Inr,&Outr);
flag=0;
}

main()
{
 instalar();
 }
```

CAPITULO 5

MANUAL DEL USUARIO

1 INTRODUCCION

Bienvenido a SYSAQUA, un sistema especialmente diseñado para obtener información en linea del comportamiento de parámetros fisicos en muestras de agua, por medio del constante monitoreo y de herramientas de análisis provistas en el programa controlador.

Dependiendo de los requerimientos del usuario final, este sistema puede ser util en el control de procesos de producción donde las caracteristicas del agua utilizada es vital, o en el simple estudio del comportamiento de las reacciones de determinados liquidos con respecto a elementos externos, v.g. análisis de comportamiento en laboratorios de calidad de agua.

El paquete está formado por un diskette conteniendo el programa controlador, un módulo electrónico externo, un cable de conexión, una fuente de poder.

> **Nota:** Los sensores no vienen incluidos en el paquete.

2 HARDWARE REQUERIDO

Para la instalación del sistema se recomienda un computador de las siguientes características:

ITEM	Características
Procesador	286 o superior
Memoria RAM	640 Kb. o mayor
Velocidad	16 MHz o mayor
Monitor	a colores CGA o superior
Puerto Paralelo	Centronic's
Espacio en Disco Duro	1 MB como mínimo

Tabla 5.1 Características Mínimas de Hardware

5.3 INSTALACIÓN DEL SISTEMA

El programa controlador del sistema se encuentra en un disco de 3½ pulgadas. Para instalar el programa se debe copiar todos los archivos del disco a un subdirectorio llamado C:\SYSAQUA. Una vez copiado los archivos, se puede dar por terminada la instalación del programa.

El siguiente paso para poner en funcionamiento el sistema es la conexión de la circuitería al puerto paralelo, asegúrese de desconectar previamente alguna dispositivo externo conectado.

Para ejecutar el programa siga los siguientes pasos:

- ☑ Cámbiese al directorio C:\SYSAQUA
- ☑ Digite SYSAQUA
- ☑ Presione ENTER

Lo primero que usted verá es la pantalla de presentación (fig 5.1), para salir de ella y continuar con la ejecución del programa presione cualquier tecla.

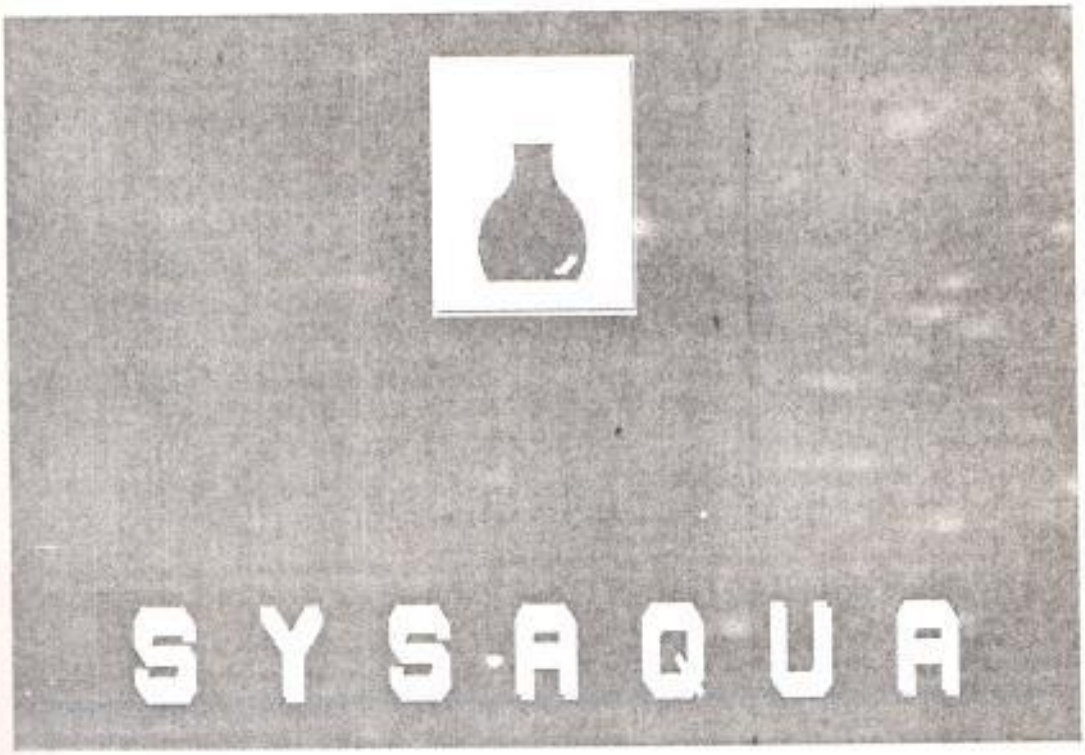


Fig. 5.1 Pantalla de Presentación del Sistema

Una vez que desaparece la pantalla de presentación aparece la pantalla principal indicando que el sistema está en pleno funcionamiento.

LA PANTALLA PRINCIPAL

La pantalla principal es la puerta de entrada al manejo del sistema, para facilidad de manejo y visualización por parte del usuario se la ha dividido en varios sectores:

- ☑ Sector de Menu
- ☑ Sector de Botones
- ☑ Nombre de Archivo de Trabajo
- ☑ Sector de Medidores y Alarmas.
- ☑ Sector de Mensajes

Todos estos sectores facilmente distinguibles se muestran en la figura 5.2.

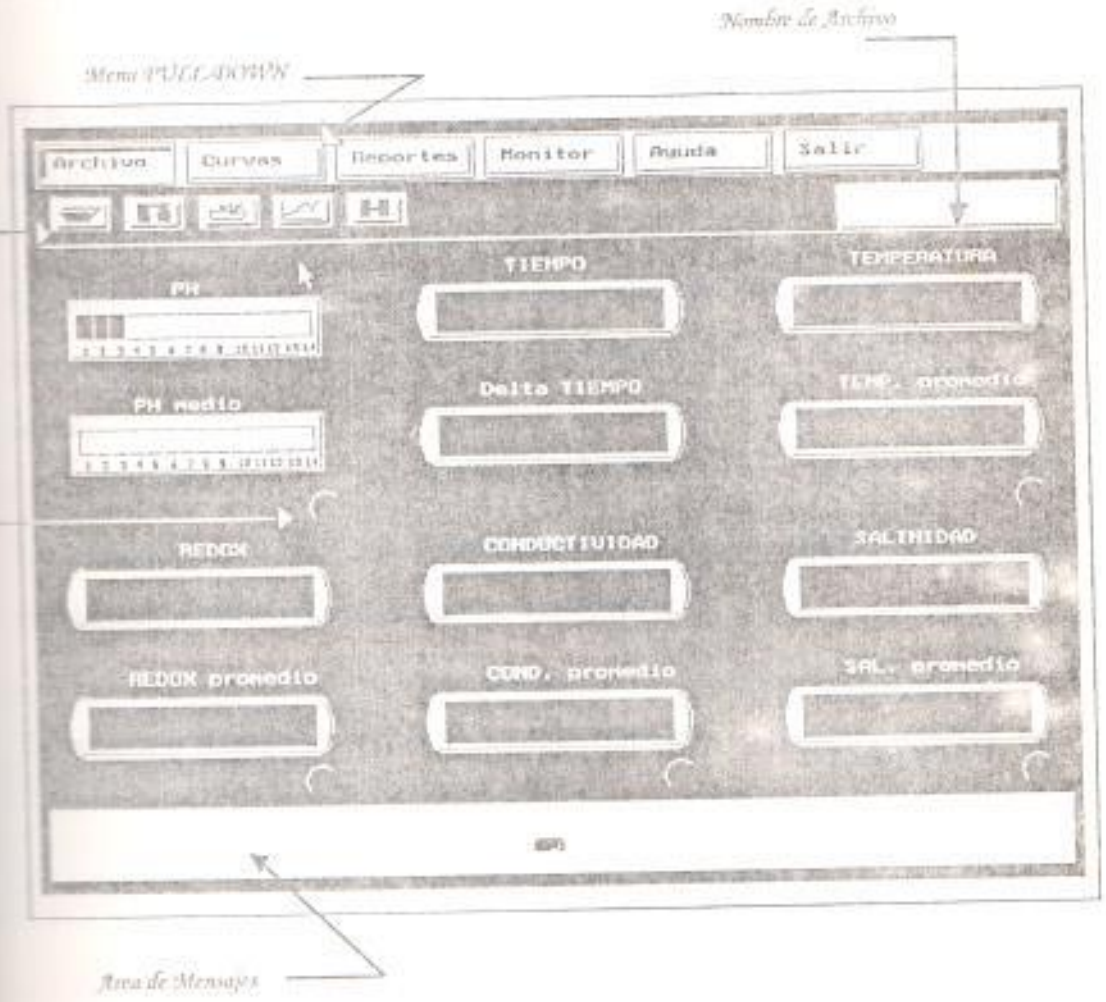


Fig.5.2 Pantalla Principal del Sistema

Se puede ver en la esquina superior izquierda el cursor del ratón, el cual se habilita automáticamente al aparecer la pantalla principal.

A continuación explicamos detalladamente cada uno de los elementos de esta pantalla.

5.4.1 Menues y Dboxes.

El sector de Menú está compuesto por un menú de tipo pull-down, el cual permite setear y manejar todas las opciones del sistema.

La navegación a través del menú se la puede realizar de dos maneras: La primera usando las teclas de movimiento del cursor ←→ y la segunda usando el ratón.

Las opciones son elegidas presionando **ENTER** o el botón izquierdo si se está usando el ratón sobre el comando correspondiente.

El menú seleccionado presenta el siguiente formato:

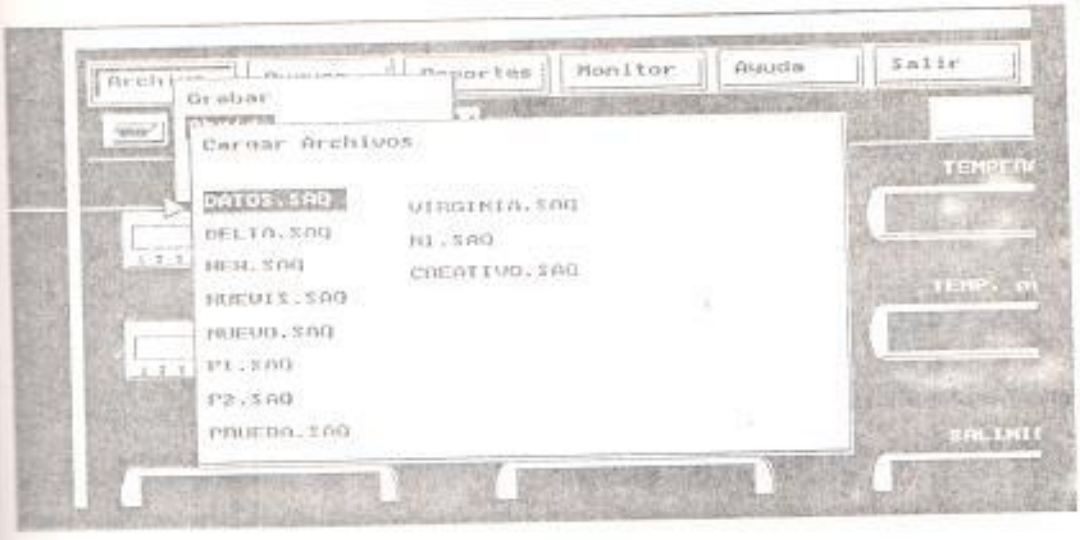


Fig. 5.3 Ejemplo de Selección de Menú

Una vez con la ventana correspondiente podemos navegar en ella usando las teclas ↑ y ↓. La selección de la opción se la realiza por medio de la tecla ENTER. Para salir de la ventana de menú activada se presiona la tecla ESC.

Es probable que como consecuencia de la elección de una determinada opción se active una Ventana de Diálogo (Dialog Box), la cual tiene el formato de la figura 5.4:

Fig. 5.4 Ejemplo de Dbox.

Esta ventana sirve para ingresar datos con formatos determinado al programa. Naturalmente estos datos son validados para evitar inconsistencias en el funcionamiento del programa.

La navegación a través del Dbox se la realiza por medio de la tecla ENTER, la mencionada tecla tiene doble función: Acepta el dato ingresado y permite al usuario cambiarse de opción si es que no se desea ingresar o modificar el dato pre-ingresado. Presionando ESC desactivamos el Dbox.

Una vez que ha escogido la opción a completar usted puede ingresar la información necesaria, las teclas **DEL** y **BACKSPACE** borran el contenido de la opción directamente dejándola limpia, lista para el ingreso de un nuevo valor o dato.

ENTER como ya se mencionó acepta el valor ingresado. Al ingresar el valor se efectúa la validación, si el dato ingresado no corresponde al tipo esperado entonces el programa procede a borrar su valor, dando la oportunidad de ingresar un valor consistente.

5.4.2 Botones

Los Botones son instrumentos muy útiles para la ejecución rápida y fácil de opciones (Fig. 5.5), se los selecciona posicionándonos con el ratón sobre ellos y aplastando el botón izquierdo, luego de eso y dependiendo

la opción elegida aparecerá una pantalla temporal de tipo menu, dialog box, ventana de advertencia, o ayuda.

Fig. 5.5 Botones del Sistema

Los botones disponibles en el sistema son

- Botón para Apertura de Archivos
- Botón para Grabar Archivos
- Botón para Impresión de Reportes
- Botón para Graficación de Curvas
- Botón para Ayuda

5.3 La tecla F1 (ayuda)

F1 es la tecla que nos permite acceder directamente a la ayuda del sistema desde el lugar en el que nos encontremos. Presenta una ventana temporal con la descripción general del lugar en el que nos encontremos, un ejemplo es la ayuda de la pantalla principal.

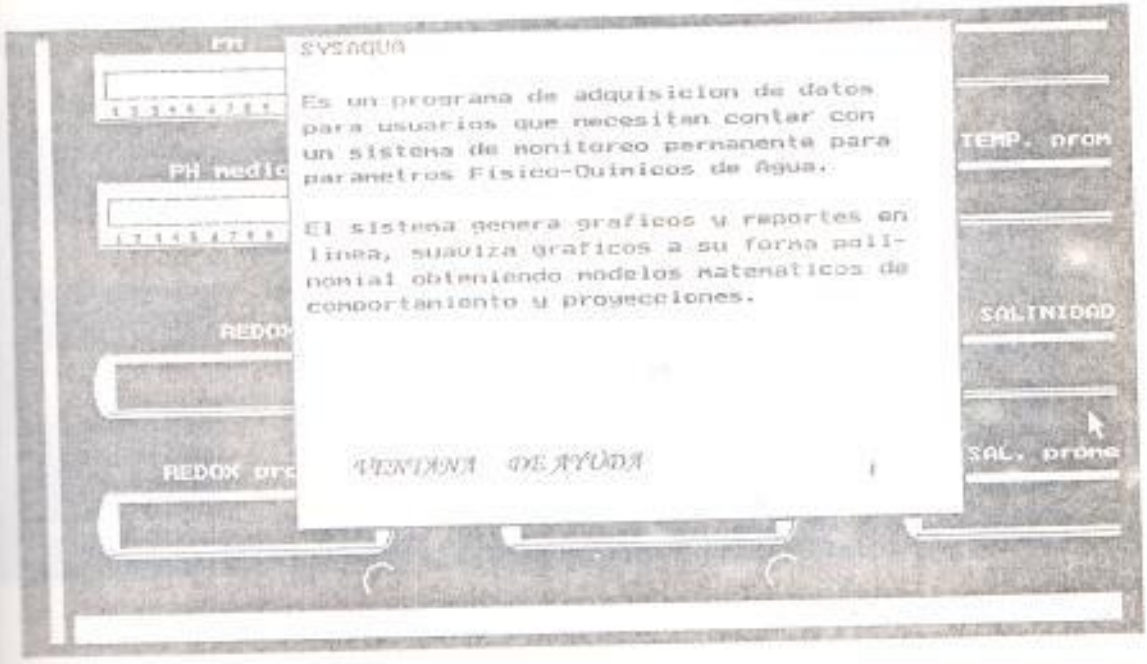


Fig. 5.6 Ejemplo de Ventana de Ayuda

Salimos de la ventana de ayuda aplastando cualquier tecla.

5.4.4 Ventanas de Advertencia

Cuando es necesario imprimir algún mensaje importante se genera una ventana temporal de advertencia, por medio de la cual el programa se asegura que el usuario quede enterado de cualquier novedad. Un ejemplo de esto es la ventana de advertencia que aparece cuando queremos salir del programa estando aún en modo residente (ver Fig 5.7).

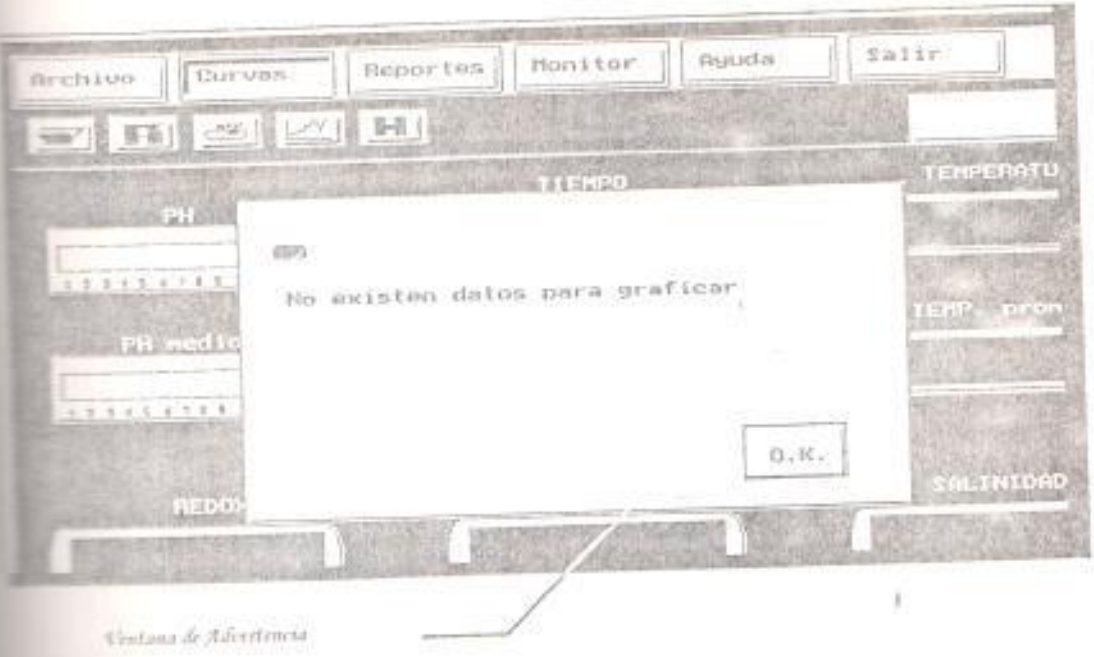


Fig. 5.7 Ejemplo de Ventana de Advertencia

Usted puede salir de la ventana de advertencia presionando cualquier tecla.

5.4.5 Medidores

Los medidores son ventanas permanentes a través de las cuales los datos monitoreados son impresos, existen dos clases diferenciadas según su uso específico. Así tenemos:

Medidores Gráficos de Barra: Usados para parámetros con niveles superiores e inferiores relativamente pequeños, en este caso para medir PH (Fig. 5.8).

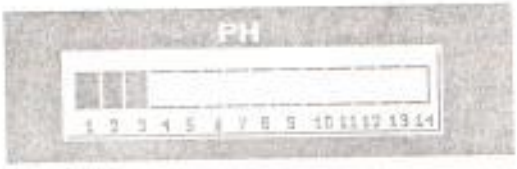


Fig. 5.8 Medidor Gráfico de Barra

Medidores Digitales: Usados para las mediciones de parámetros con límites superiores altos o mediciones que requieran alta precisión (Fig. 5.9). En nuestro caso el resto de parámetros.

Fig 5.9. Medidor Digital

En el sector de medidores encontramos dos medidores del mismo tipo para cada parámetro. Los medidores tienen nombres en la parte superior indicando la medición actual y los valores promedios de la muestra tomando en cuenta hasta el último dato recogido.

5.4.6 Sistema de Alarmas

El sistema de alarmas del programa está implementado por medio los botones situados en la parte inferior izquierda de los medidores, en condiciones normales son silenciosos y de color verde, pero cuando el dato medido sobrepasa su valor crítico entonces se activan coloreándose de forma intermitente y emitiendo un pitido.

Fig. 5.10 Localización de Alarmas

Puede darse el caso de más de una alarma activada a la vez, las alarmas se desactivan cuando el dato vuelve a su estado original.

5.5 MENU DE ARCHIVOS

Este menú concentra todos aquellos comandos que tienen que ver con el manejo de Archivos. Tenemos las siguientes opciones:

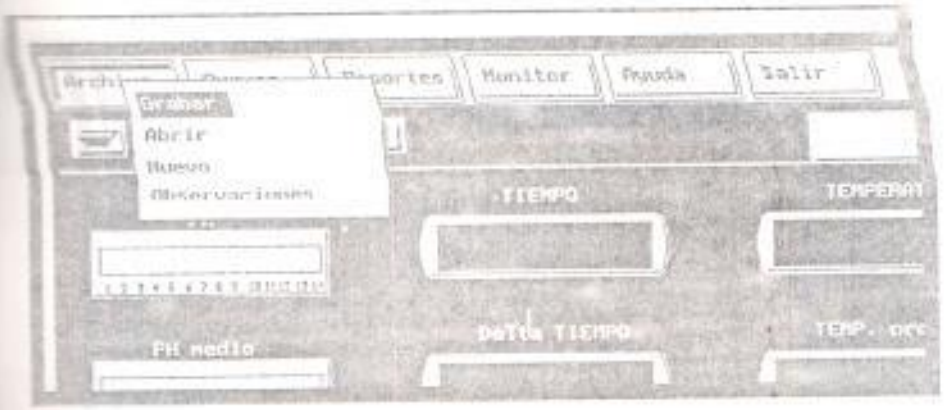


Fig. 5.11 Menú de Archivos

Los Archivos de trabajo de SYSAQUA tienen extensión .SAQ por definición.

5.5.1 Abriendo archivos

Para abrir un archivo existente nos posicionamos en la opción de Abrir. Al abrir un archivo existente se carga en el programa los valores de los parámetros leídos en el archivo y se imprimen en pantalla los valores promedios de dichos parámetros. Los pasos para Abrir un archivo existente se resumen en los siguientes:

☑ Posiciónese en la opción de Abrir y presionar ENTER.

☑ Llene el Dialog Box con el directorio en el que se van a buscar los archivos.

☑ De la lista de archivos existentes seleccione el archivo deseado y presione ENTER.

Fig. 5.12 Ejemplo de Selección de Archivos

Siguiendo estos pasos se carga el archivo de trabajo. Si los pasos fueron llevados a cabo correctamente, el nombre del archivo seleccionado debe aparecer en el área correspondiente a nombre de archivo de trabajo.

Una vez que el archivo ha sido cargado podemos manejar completamente las opciones del menú principal por ejemplo las opciones de gráfico y de reportes.

5.5.2 Grabando su trabajo

Usted puede grabar el trabajo cargado actualmente ya sea por el comando Abrir o por monitoreo de muestras, con el comando Grabar.

Los pasos para guardar su trabajo son los siguientes:

- ☑ Asegúrese que su programa no esté en modo residente
- ☑ Asegúrese que existan datos para grabar
- ☑ Invoque el comando Abrir del menú de Archivo
- ☑ Llene la ventana de observaciones si desea
- ☑ En el Dbox presentado, escriba la ruta y el nombre del nuevo archivo
- ☑ Presione ENTER, el nombre del archivo aparecerá en el área correspondiente a archivo de trabajo.

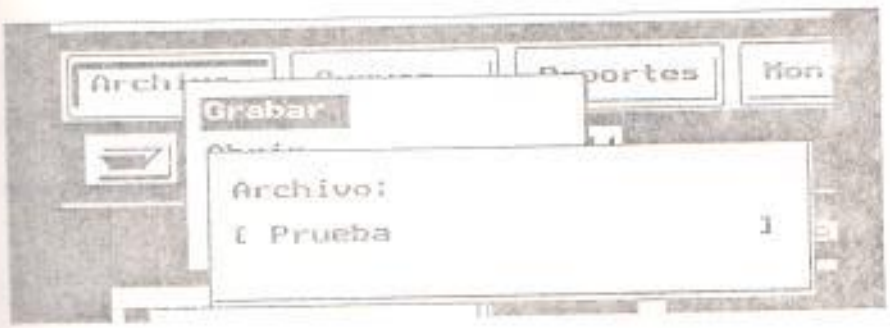


Fig. 5.13 Ejemplo de Grabación de Datos Muestreados

- Si el nombre del archivo está repetido, el sistema le indicará el problema y le pedirá confirmación para actualizar el archivo existente.

5.3 Comenzando un nuevo trabajo

Una vez grabado sus datos entonces está listo para comenzar un nuevo trabajo de monitoreo o de visualización de archivos existentes.

El comando Nuevo del menú de Archivos le permite a usted desechar los datos anteriores para dar paso a la utilización de nuevos valores. Basta con presionar ENTER en el comando Nuevo para que se enceren los medidores y desaparezca el nombre del archivo de trabajo.

Nota: Si usted decide comenzar un nuevo trabajo debe tomar en cuenta que todos los datos anteriores de su sistema se perderán irremediablemente.

5.4 Añadiendo observaciones a la muestra

Siempre es útil tener comentarios adicionales con respecto a la muestra que está siendo monitoreada que en cierto momento nos ayuden a

despejar dudas, es por esto que el programa implementa el comando Observaciones.

Este comando presenta un Dbox del siguiente tipo:

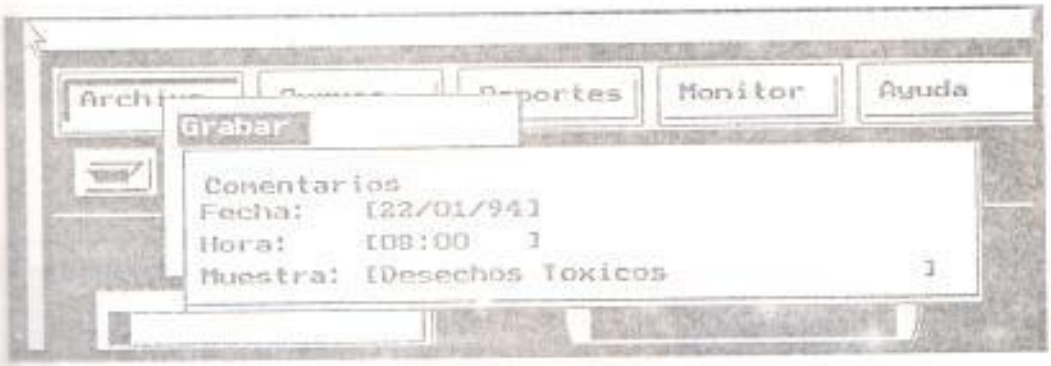


Fig. 5.14 Ejemplo de Dbox de Comentarios

Como puede ver tenemos campos para fecha de muestreo, hora de muestreo y comentarios de la muestra, estos datos serán guardados en su oportunidad en el archivo .SAQ correspondiente.

5.6 MENU DE CURVAS

Se ha implementado la graficación de parámetros como una herramienta de ayuda en el análisis del comportamiento de la muestra monitoreada evitando en lo posible la necesidad de traspasar los datos a una hoja

electrónica. Detalles más avanzados acerca del manejo de gráficos se dan en la sección 5.7 La Pantalla de Gráficos.

La ventana de curvas presenta dos opciones principales para el procesamiento de graficación de parámetros: Definición de gráficos y Graficación.

Fig. 5.15 Menú de Curvas

> Esta opción es válida solamente si existen datos para graficar.

5.6.1 Definiendo Curvas

Usted puede definir gráficos rectangulares, parámetro versus parámetros usando cualquier clase de datos recolectados. Para definir gráficos X,Y debe seguir los pasos abajo indicados, note que los parámetros serán seleccionados en base a números.

☑ Presione ENTER sobre la opción Definición

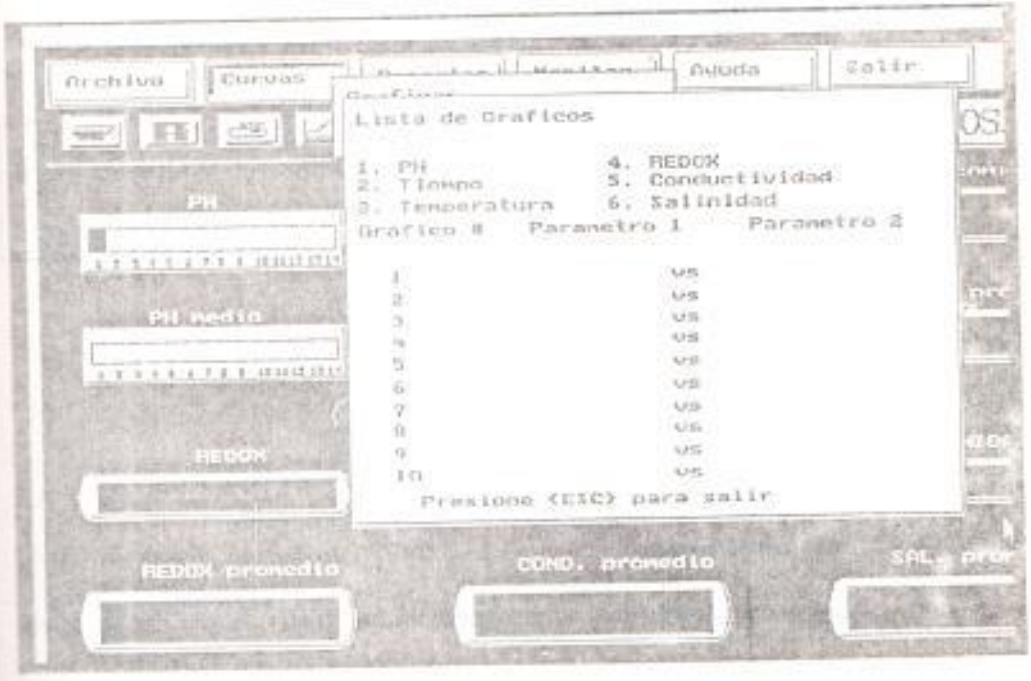


Fig. 5.16 Definición de Curvas del Sistema

☑ Digite el número del primer parámetro X, se dará cuenta que el cursor automáticamente se posiciona en el siguiente espacio

☑ Digite el número del segundo parámetro Y

☑ Repita hasta un máximo de 10 definiciones

☑ Presione ESC para salir del Dbox

> El programa permite hasta un máximo de 10 gráficos definidos. Cada vez que usted define nuevos gráficos, la definición anterior es descartada

5.6.2 Graficación

Al seleccionar el comando Graficar usted ingresará a la Pantalla de Gráficos (ver sección 5.7).

> El programa impedirá el acceso a Graficar si no existen gráficos predefinidos.

5.7 LA PANTALLA DE GRÁFICOS

Usted entra a la pantalla de gráficos por medio del comando Graficar del menú de Curvas, esta pantalla presenta un menú pull-down con comandos para la manipulación y el análisis de los diversos gráficos predefinidos.

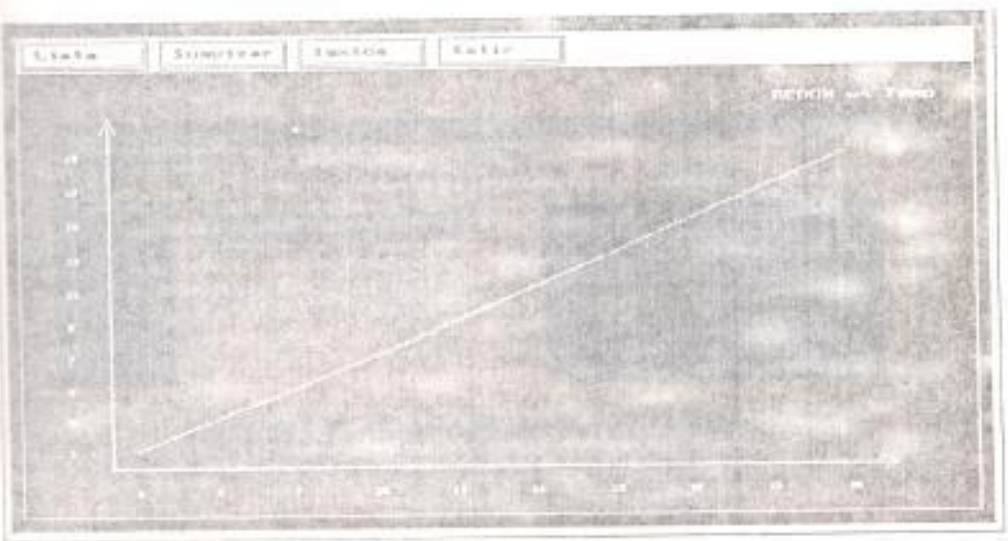


Fig. 5.17 Pantalla de Gráficos

Los comandos presentados en esta pantalla son invocados de la misma manera como los de la pantalla principal (ver sección 5.4 La Pantalla Principal).

5.7.1 Lista de gráficos predefinidos

Podemos visualizar cualquier gráfico predefinido seleccionándolo de la lista presentada por el programa. Una vez seleccionado se graficará una recta X,Y con sus respectivos ejes.

Para llamar a uno de los gráficos predefinidos seguimos los siguientes pasos:

- ☑ Seleccione el comando Lista.
- ☑ De la lista presentada seleccione el gráfico de interés

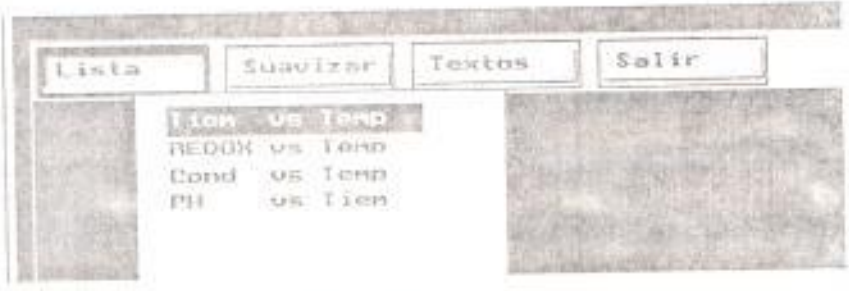


Fig. 5.18. Ejemplo de Elección de Gráfico

Puede notar que la escala usada es automática, fijada de acuerdo a los valores máximos de cada uno de los parámetros ordenados. Cada

coordenada de la gráfica está señalada por círculos. El título provisional del gráfico aparecerá en la parte superior derecha de la pantalla (ver figura 5.19).

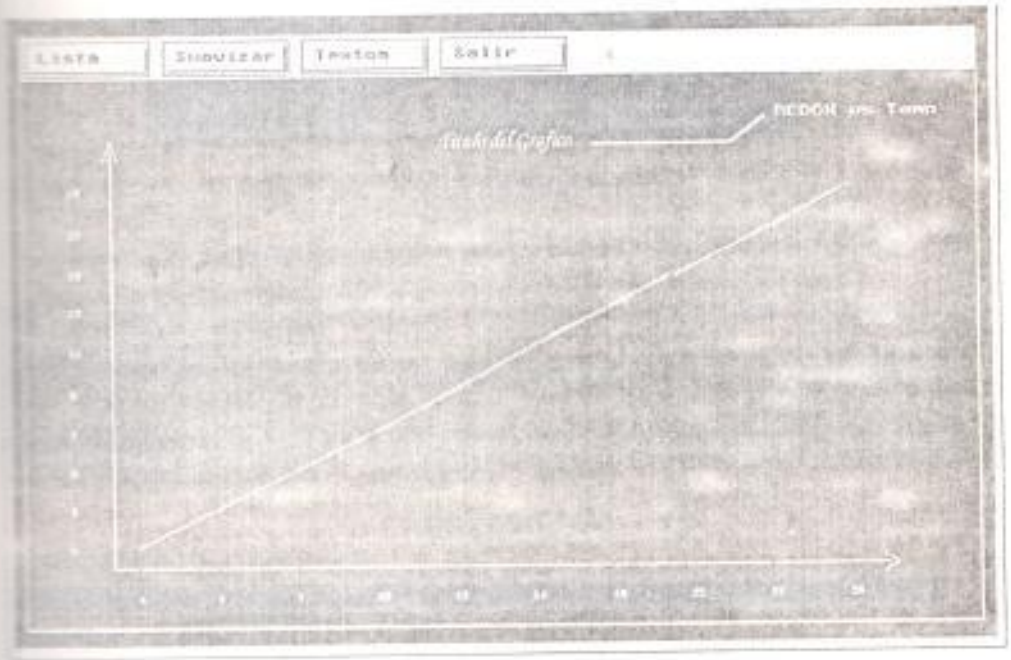


Fig. 5.19 Ejemplo de Gráfico Seleccionado

Suavización de gráficos

Usted puede aprovechar esta característica del sistema para hacer análisis más profundos y proyecciones válidas del comportamiento de la muestra monitoreada. El programa utiliza el algoritmo de Interpolación y polinomio de Lagrange en grupos de 5 elementos o menos, es decir que para un arreglo de 13 datos tendremos dos polinomios de orden 4 y uno de orden 2. Esto lo hace con el fin de obtener aproximaciones más precisas con menos cantidades decimales involucradas.

Fig. 5.20 Menú de Suavización

Presionando la opción Coeficientes puede usted obtener los coeficientes de los polinomios de orden 4 obtenidos para los diversos intervalos de la curva graficada (Fig 5.21).

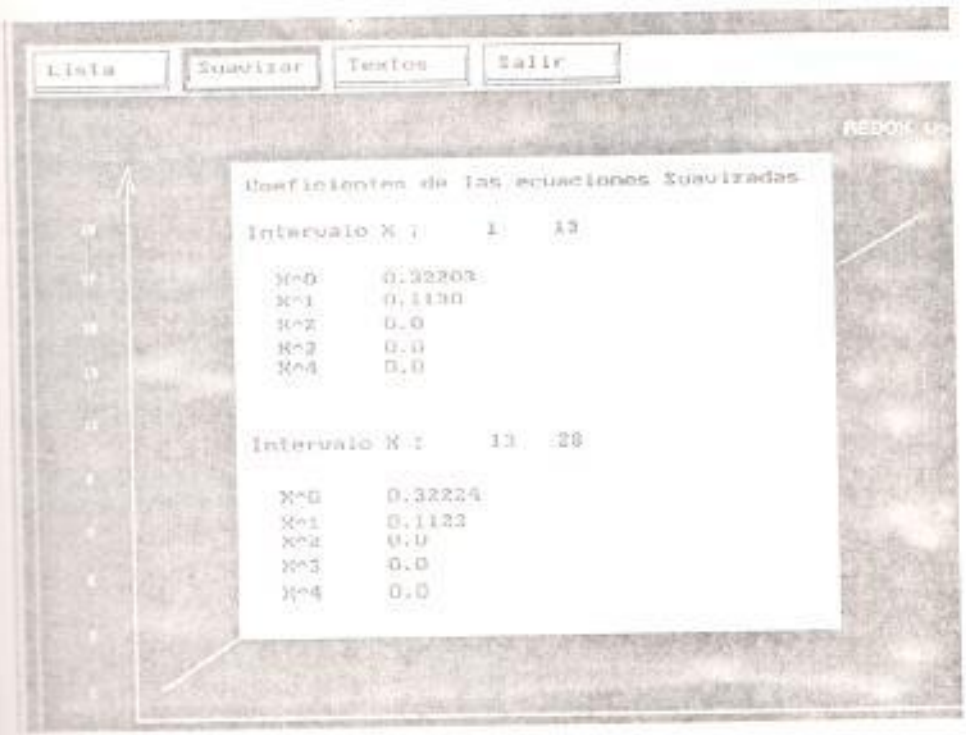


Fig. 5.21 Coeficientes de los Polinomios Suavizados

5.7.3 Mejorando la presentación de su gráfico

La presentación del gráfico puede ser mejorada por medio de la inserción de texto ya sea para el título, eje X, eje Y. Esto lo logra seleccionando la opción texto del menú de la pantalla de gráficos (fig. 5.22).

Fig. 5.22 Opción Texto

Al seleccionar la opción Texto aparecerá un Dbox en el cual usted puede especificar los diversos títulos a colocar en el gráfico (Fig.5.22).

Luego debe seleccionar nuevamente el gráfico usando el comando Lista permitiendo la actualización del mismo (Fig. 5.23).

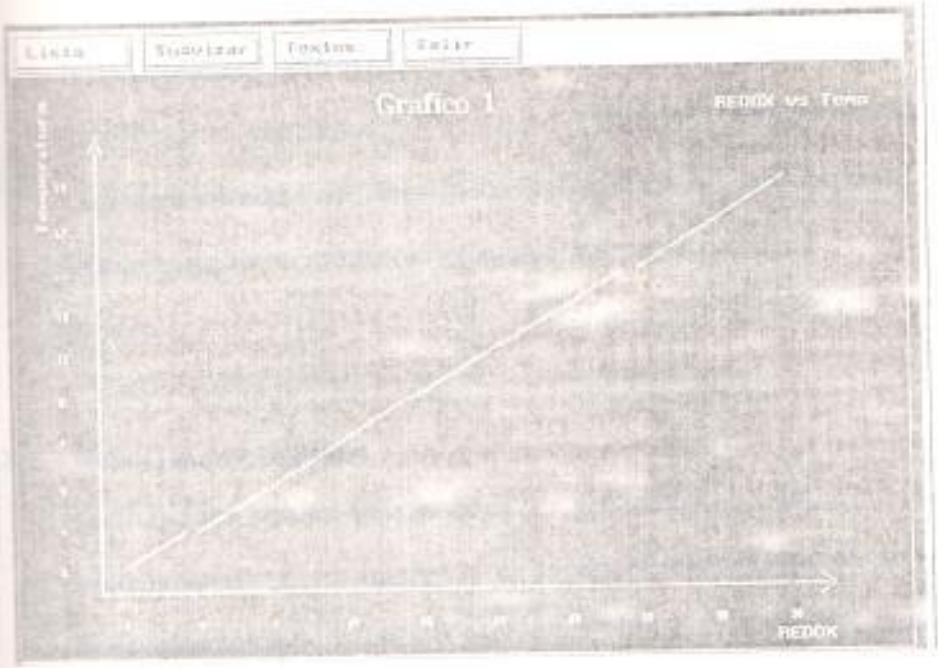


Fig. 5.23 Gráfico con Textos Definidos

5.7.4 Impresión de Gráficos

Usted puede imprimir el gráfico tal cual aparece en pantalla con sólo escoger la opción Imprimir del menú que aparece en la pantalla gráfica.

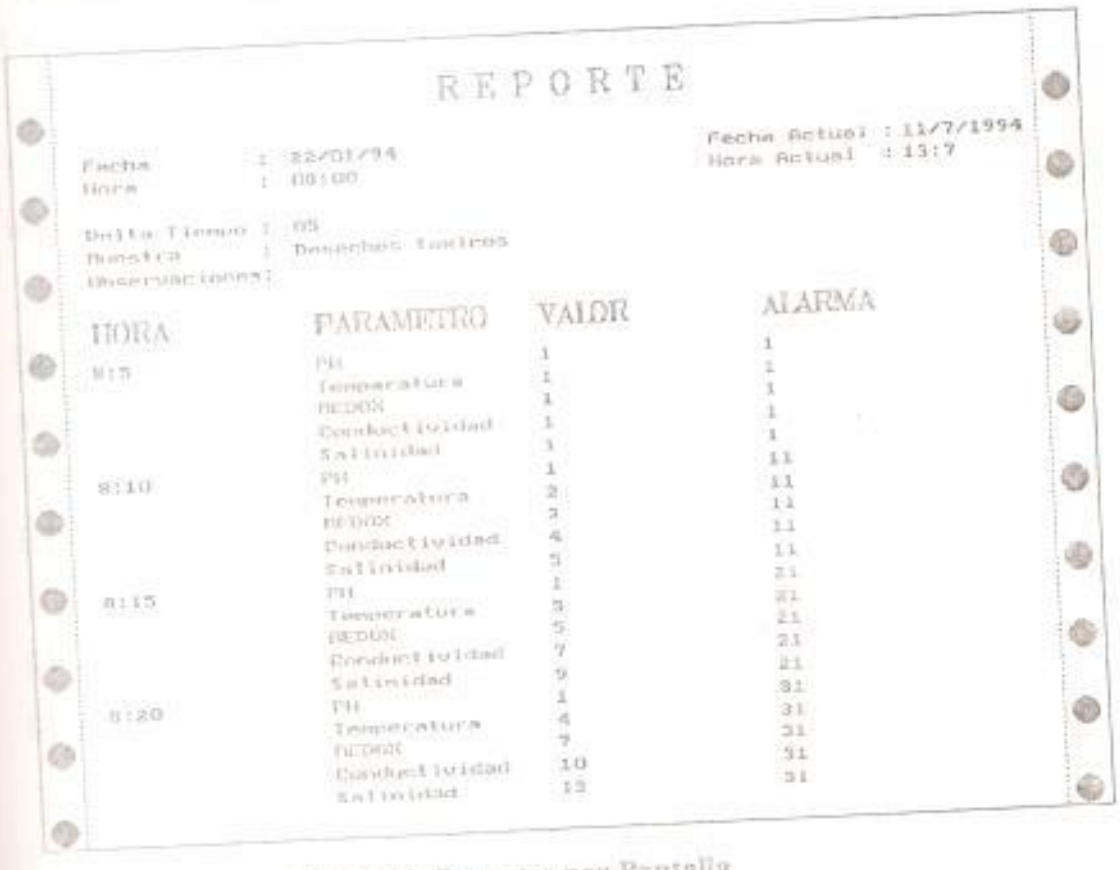
> Debe estar seguro que su impresora esté activa, de otra manera aparecerá un mensaje de advertencia.

5.5 GENERACION DE REPORTES

Los Reportes generados por el sistema (siempre y cuando existan datos cargados) pueden ser mostrados en pantalla y por impresora, esta selección la toma usando el comando Reportes de menú de la pantalla principal.

5.5.1 Reportes por pantalla

Los reportes por pantalla pueden ser generados en linea aún cuando el sistema se encuentre operando en modo residente (reportes en tiempo real) seleccionando la opción pantalla del Menú de Reportes, con lo cual obtenemos una pantalla del siguiente tipo:



REPORTE

Fecha Actual : 11/7/1994
Hora Actual : 13:7

Fecha : 22/01/94
Hora : 08:00

Delta Tiempo : 05
Planta : Desechos Toxicos
Observaciones:

HORA	PARAMETRO	VALOR	ALARMA
8:5	PH	1	1
	Temperatura	1	1
	REDOX	1	1
	Conductividad	1	1
	Salinidad	1	11
8:10	PH	1	11
	Temperatura	2	11
	REDOX	3	11
	Conductividad	4	11
	Salinidad	5	21
8:15	PH	1	21
	Temperatura	3	21
	REDOX	5	21
	Conductividad	7	21
	Salinidad	9	31
8:20	PH	1	31
	Temperatura	4	31
	REDOX	7	31
	Conductividad	10	31
	Salinidad	13	

Fig. 5.24 Reportes por Pantalla

El reporte por pantalla muestra los siguientes datos:

- ☑ Fecha de muestreo, hora de inicio de muestreo intervalo de muestreo, observaciones, fechas y hora actuales.
- ☑ Las horas en las cuales fueron medidos los parámetros
- ☑ Los valores registrados de cada uno de ellos
- ☑ Ausencia o Existencia de condiciones de Alarma.

Usted además puede navegar por la pantalla de reportes utilizando las teclas PgUp, PgDn o ESC para salir.

5.8.2 Cómo obtener un reporte impreso

Los reportes por impresora no se los puede obtener en línea a menos que usted cuente con dos puertos paralelos.

Para que pueda generar un reporte impreso debe seguir los siguientes pasos:

- ☑ Asegúrese que la impresora esté debidamente conectada, caso contrario saldrá un mensaje de advertencia.
- ☑ Active en el Menú de Reportes la opción de Impresora.
- ☑ Cambie el papel cuando el programa así lo exija, por definición se imprimen un máximo de 55 líneas por página.

2.8.3 Exportando los datos en formato texto

Si usted desea hacer un análisis más exahustivo que el que realiza SYSAQUA, puede exportar su reporte en formato texto para que pueda importarlo sin problemas en cualquier hoja electrónica o procesador de palabras. Para exportar un reporte usted debe hacer lo siguiente:

- ☑ Active el Menú de Reportes
- ☑ Escoja la opción Archivo
- ☑ Digite el nombre del archivo texto destino con la extensión deseada

> ✍ Si el nombre del archivo digitado corresponde a uno existente el programa emitirá un mensaje pidiendo confirmación para su reemplazo o anulación de la operación.

2.9 MONITOREO RESIDENTE

Para activar la recolección de datos de la muestra usted debe de programar el monitoreo residente siguiendo los siguientes pasos:

- ☑ Asegúrese que la circuitería esté conectada al puerto paralelo del computador.
- ☑ Seleccione la opción Monitoreo del menú pull-down en la pantalla principal.

☑ Establezca los niveles críticos de los parámetros

☑ Defina el intervalo de tiempo de muestreo en minutos

5.9.1 Definiendo valores críticos

Al seleccionar Monitoreo del menú pull-down se activará la siguiente pantalla temporal de tipo Dbox:

Fig. 5.25 Selección de Valores Críticos e Intervalo de Tiempo de Muestreo

En esta pantalla usted debe seleccionar los valores críticos de los parámetros respetando las unidades especificadas. Una vez que ha

determinado los valores críticos debe establecer el intervalo de tiempo en minutos en el cual la muestra va a ser monitoreada.

Al salir de esta pantalla temporal el sistema inmediatamente empezará con el monitoreo de datos (el sistema entra en Modo Residente), el cual durará hasta que usted detenga el trabajo. Note que al entrar el programa a Modo Residente se enciende una señal MR en la parte superior izquierda de su pantalla indicando que el sistema está capturando datos *(Fig. 5.26)*.

Fig. 5.26

Para abortar la instalación del Modo Residente debe poner ESC antes de definir los valores críticos, si hace esto se activará el mensaje de advertencia "Modo Residente Abortado".

> Mientras el programa se encuentre en Modo Residente no podrá abrir nuevos archivos, imprimir informes o salir del programa. Todas las opciones restantes se encuentran plenamente disponibles.

5.3.2 Parando el Trabajo

Si usted desea detener el trabajo de monitoreo lo único que debe hacer es seleccionar la opción Monitoreo del menú pull-down, al realizar esto usted verá un mensaje que dice "Usted está saliendo del Modo Residente" e inmediatamente la señal MR se borrará.

> Una vez detenido el trabajo no podrá empezar continuar con el monitoreo en el mismo ambiente de trabajo, así que se sugiere grabar lo anterior, escoger la opción Nuevo del Menú de Archivos y empezar a monitorear otra vez.

CAPITULO 6

PRUEBAS DEL SISTEMA

El sistema fue probado en tres fases, tomando en cuenta especialmente condiciones extremas, estas fases se enumeran como sigue:

- ☑ Pruebas para Hardware
- ☑ Pruebas de Software
- ☑ Pruebas de Hardware/Software

PRUEBAS DE HARDWARE

La circuitería de adquisición de datos fue probada por etapas, según era armada. Primero se construyó el módulo acondicionador de señal, luego el módulo convertidor analógico digital y por último el módulo multiplexor de resultados.

4.3.1 MATERIALES USADOS

Los equipos usados para las pruebas de circuitería fueron:

- 2 Multímetros
- 1 Osciloscopio
- 1 Fuente de 5V
- 1 Punta lógica

4.3.2 METODOLOGIA

- Para las pruebas al módulo acondicionador de señal se utilizaron señales de prueba provenientes de un circuito alimentado con +5V del cual se obtenía una señal de entrada en mV (hasta 400 mv) lograda por la variación de un reóstato y parecida a las señales típicas de los sensores. Esta señal era amplificada y medida finalmente por un multímetro.

- Para probar el módulo convertidor analógico/digital se usó además de dos señales provenientes del módulo acondicionador de señal, una tercera proveniente íntegramente de la circuitería elaborada para el termistor. Las señales de control para el ADC fueron enteramente manuales con excepción de la señal de reloj (ver figura 2.8).

- El módulo multiplexor de resultados fue probado con las señales resultantes del módulo convertidor analógico/digital y la señal de SLCT de manera manual.

RESULTADOS

El código teórico es obtenido mediante la fórmula

$$Código = \frac{Vin * 255}{Vref[+] - Vref[-]}$$

Voltaje de Entrada [V]	Código Obtenido	Código Teórico	% Error
1.35	69	68.85	0.21
1.43	73	72.93	0.095
1.45	74	73.95	0.067
1.647	84	83.997	0.0035

Tabla 6.1 Resultados de Pruebas al Módulo Convertidor Analógico Digital

Con lo cual vemos errores mínimos entre los valores teóricos y los obtenidos, concluyendo que el módulo convertidor analógico digital trabaja eficientemente.

El módulo multiplexor elige entre los cuatro datos menos o mas significativos dependiendo de SLCT 0 o 1 respectivamente.

Código de Salida Original	Código con SLCT 1	Código con SLCT 0
70	0100	0110
74	0100	0101
84	0101	0100

Tabla 6.2 Resultados del Módulo Multiplexor

PRUEBAS DE SOFTWARE

De igual manera que el Hardware, el Software fue probado por fases, a medida que sus diversos componentes iban acoplándose, un tópico interesante consisten las pruebas del modo residente del sistema, el cual fue realizado sin la intervención de la circuitería de adquisición de datos.

La metodología presentada corresponde a una de las múltiples pruebas de software a las que fue sometido el programa.

METODOLOGIA

- Se estableció los niveles críticos para los diferentes parámetros con el fin de probar las alarmas con un intervalo de tiempo para muestreo fue de 7 minutos. Es necesario tomar en cuenta que el intervalo de

muestreo se lo especifica en minutos pero para los casos de prueba se lo ha bajado a segundos, con el fin de obtener mas mediciones en cortos lapsos y probar algunas situaciones extremas con relación a los tiempos de muestreo.

- Mientras el programa estaba en modo residente se definieron gráficos PH vs.Tiempo y REDOX vs. Tiempo y se procedió a graficarlos.

- Se obtuvo suavizaciones de los gráficos definidos.

- Se paró el monitoreo para grabar los datos y obtener reportes.

RESULTADOS

De los pasos anteriores se obtuvo el siguiente reporte impreso de resultados.

REPORTES POR PANTALLA E IMPRESORA

REPORTE

Fecha :
Hora :05:12
Delta Tiempo :7
Muestra :
Observaciones:

HORA	PARAMETRO	VALOR	ALARMA
13:7	PH	14	1
	Temperatura	51.5	0
	REDOX	187.5	1
	Conductividad	97.5	0
	Salinidad	62.4	0
13:14	PH	14	1
	Temperatura	63.5	0
	REDOX	247.5	1
	Conductividad	128.7	1
	Salinidad	86.4	0
13:21	PH	14	1
	Temperatura	26	0
	REDOX	60	0
	Conductividad	31.2	0
	Salinidad	11.4	0
13:28	PH	14	1
	Temperatura	38	0
	REDOX	120	1
	Conductividad	62.4	0
	Salinidad	35.4	0
13:35	PH	3	0
	Temperatura	4.5	0
	REDOX	7.5	0
	Conductividad	3.9	0
	Salinidad	0.3	0
13:42	PH	14	1
	Temperatura	62	0
	REDOX	240	1
	Conductividad	124.8	1
	Salinidad	83.4	0

13:49	PH	14	1
	Temperatura	24	0
	REDOX	50	0
	Conductividad	26	0
	Salinidad	7.4	0
13:56	PH	14	1
	Temperatura	36	0
	REDOX	110	1
	Conductividad	57.2	0
	Salinidad	31.4	0
14:3	PH	14	1
	Temperatura	48	0
	REDOX	170	1
	Conductividad	88.4	0
	Salinidad	55.4	0
14:10	PH	14	1
	Temperatura	33.5	0
	REDOX	97.5	0
	Conductividad	50.7	0
	Salinidad	26.4	0
14:17	PH	14	1
	Temperatura	38.5	0
	REDOX	122.5	1
	Conductividad	63.7	0
	Salinidad	36.4	0
14:24	PH	14	1
	Temperatura	50.5	0
	REDOX	182.5	1
	Conductividad	94.9	0
	Salinidad	60.4	0
14:31	PH	14	1
	Temperatura	62.5	0
	REDOX	242.5	1
	Conductividad	126.1	1
	Salinidad	84.4	0
14:38	PH	14	1
	Temperatura	24.5	0
	REDOX	52.5	0
	Conductividad	27.3	0
	Salinidad	8.4	0

14:45	PH	14	1
	Temperatura	36.5	0
	REDOX	112.5	1
	Conductividad	58.5	0
	Salinidad	32.4	0
14:52	PH	14	1
	Temperatura	48.5	0
	REDOX	172.5	1
	Conductividad	89.7	0
	Salinidad	56.4	0
14:59	PH	14	1
	Temperatura	60.5	0
	REDOX	232.5	1
	Conductividad	120.9	1
	Salinidad	80.4	0
15:6	PH	14	1
	Temperatura	23	0
	REDOX	45	0
	Conductividad	23.4	0
	Salinidad	5.4	0
15:13	PH	14	1
	Temperatura	35	0
	REDOX	105	1
	Conductividad	54.6	0
	Salinidad	29.4	0
15:20	PH	14	1
	Temperatura	47	0
	REDOX	165	1
	Conductividad	85.8	0
	Salinidad	53.4	0
15:27	PH	14	1
	Temperatura	59	0
	REDOX	225	1
	Conductividad	117	1
	Salinidad	77.4	0
15:34	PH	14	1
	Temperatura	21	0
	REDOX	35	0
	Conductividad	18.2	0
	Salinidad	1.4	0

15:41	PH	14	1
	Temperatura	33	0
	REDOX	95	0
	Conductividad	49.4	0
	Salinidad	25.4	0
15:48	PH	14	1
	Temperatura	45	0
	REDOX	155	1
	Conductividad	80.6	0
	Salinidad	49.4	0
15:55	PH	14	1
	Temperatura	23.5	0
	REDOX	47.5	0
	Conductividad	24.7	0
	Salinidad	6.4	0
16:2	PH	14	1
	Temperatura	35.5	0
	REDOX	107.5	1
	Conductividad	55.9	0
	Salinidad	30.4	0
16:9	PH	14	1
	Temperatura	47.5	0
	REDOX	167.5	1
	Conductividad	87.1	0
	Salinidad	54.4	0
16:16	PH	14	1
	Temperatura	59.5	0
	REDOX	227.5	1
	Conductividad	118.3	1
	Salinidad	78.4	0
16:23	PH	14	1
	Temperatura	21.5	0
	REDOX	37.5	0
	Conductividad	19.5	0
	Salinidad	2.4	0
16:30	PH	14	1
	Temperatura	33.5	0
	REDOX	97.5	0
	Conductividad	50.7	0
	Salinidad	26.4	0

16:37	PH	14	1
	Temperatura	45.5	0
	REDOX	157.5	1
	Conductividad	81.9	0
	Salinidad	50.4	0
16:44	PH	14	1
	Temperatura	63.5	0
	REDOX	247.5	1
	Conductividad	128.7	1
	Salinidad	86.4	0
16:51	PH	14	1
	Temperatura	56	0
	REDOX	210	1
	Conductividad	109.2	1
	Salinidad	71.4	0
16:58	PH	8	0
	Temperatura	12	0
	REDOX	20	0
	Conductividad	10.4	0
	Salinidad	0.8	0
17:5	PH	14	1
	Temperatura	57	0
	REDOX	215	1
	Conductividad	111.8	1
	Salinidad	73.4	0
17:12	PH	14	1
	Temperatura	58.5	0
	REDOX	222.5	1
	Conductividad	115.7	1
	Salinidad	76.4	0
17:19	PH	13	1
	Temperatura	19.5	0
	REDOX	32.5	0
	Conductividad	16.9	0
	Salinidad	1.3	0
17:26	PH	14	1
	Temperatura	59.5	0
	REDOX	227.5	1
	Conductividad	118.3	1
	Salinidad	78.4	0

17:33	PH	14	1
	Temperatura	21.5	0
	REDOX	37.5	0
	Conductividad	19.5	0
	Salinidad	2.4	0
17:40	PH	14	1
	Temperatura	33.5	0
	REDOX	97.5	0
	Conductividad	50.7	0
	Salinidad	26.4	0
17:47	PH	14	1
	Temperatura	45.5	0
	REDOX	157.5	1
	Conductividad	81.9	0
	Salinidad	50.4	0
17:54	PH	14	1
	Temperatura	57.5	0
	REDOX	217.5	1
	Conductividad	113.1	1
	Salinidad	74.4	0
18:1	PH	11	0
	Temperatura	16.5	0
	REDOX	27.5	0
	Conductividad	14.3	0
	Salinidad	1.1	0
18:8	PH	14	1
	Temperatura	31.5	0
	REDOX	87.5	0
	Conductividad	45.5	0
	Salinidad	22.4	0
18:15	PH	14	1
	Temperatura	43.5	0
	REDOX	147.5	1
	Conductividad	76.7	0
	Salinidad	46.4	0
18:22	PH	14	1
	Temperatura	56	0
	REDOX	210	1
	Conductividad	109.2	1
	Salinidad	71.4	0

18:27	PH	8	0
	Temperatura	12	0
	REDOX	20	0
	Conductividad	10,4	0
	Salinidad	0.8	0
18:36	PH	14	1
	Temperatura	30	0
	REDOX	80	0
	Conductividad	41,6	0
	Salinidad	19.4	0
18:43	PH	14	1
	Temperatura	58	0
	REDOX	220	1
	Conductividad	114.4	1
	Salinidad	75.4	0
18:50	PH	12	1
	Temperatura	18	0
	REDOX	50	0
	Conductividad	15,6	0
	Salinidad	1.2	0
18:57	PH	14	1
	Temperatura	32	0
	REDOX	90	0
	Conductividad	46.8	0
	Salinidad	23.4	0
19:4	PH	14	1
	Temperatura	44	0
	REDOX	150	1
	Conductividad	78	0
	Salinidad	47.4	0
19:11	PH	14	1
	Temperatura	56.5	0
	REDOX	212.5	1
	Conductividad	110.5	1
	Salinidad	72.4	0
19:18	PH	9	0
	Temperatura	13.5	0
	REDOX	22.5	0
	Conductividad	11.7	0
	Salinidad	0.9	0

19:25	PH	14	1
	Temperatura	30.5	0
	REDOX	82.5	0
	Conductividad	42.9	0
	Salinidad	20.4	0
19:32	PH	14	1
	Temperatura	42.5	0
	REDOX	142.5	1
	Conductividad	74.1	0
	Salinidad	44.4	0
19:39	PH	14	1
	Temperatura	54.5	0
	REDOX	202.5	1
	Conductividad	105.3	1
	Salinidad	68.4	0
19:46	PH	5	0
	Temperatura	7.5	0
	REDOX	12.5	0
	Conductividad	6.5	0
	Salinidad	0.5	0
19:53	PH	14	1
	Temperatura	28.5	0
	REDOX	72.5	0
	Conductividad	37.7	0
	Salinidad	16.4	0

CONCLUSIONES

1. La culminación de este trabajo ha probado la capacidad del ingeniero politécnico para estudiar y combinar áreas diversas de la ciencia y unificarlas en una aplicación de su especialidad.

2. Se ha demostrado que en nuestro medio es altamente posible la producción de sistemas de adquisición de datos eficientes, a bajos precios y con calidad igual a los sistemas extranjeros.

3. El sistema actualmente implementado, si bien es cierto, no cumple con todas las exigencias de hardware en un principio propuestas (por falta de dinero no se pudo instalar sensores de PH, REDOX y salinidad), compensa esas deficiencias con el enriquecimiento de las características del programa, las cuales superan las originalmente planteadas en la presentación del proyecto.

4. El presente trabajo ha permitido evaluar una de las técnicas de programación poco conocida en nuestro medio pero de gran aceptación en otros países, la programación orientada a objetos (OOP). He encontrado que este tipo de programación simplifica en gran manera la codificación del programa, por medio de la derivación de clases y aminora grandemente las líneas de código por medio de la no repetición de funciones utilizadas para lograr efectos similares, un claro ejemplo de esto se ve en la construcción de menúes.

RECOMENDACIONES

1. Es recomendable realizar una versión mejorada del sistema bajo entorno WINDOWS, con lo cual el programa ganaría una mayor facilidad en su uso, además se obtendría un sistema que pueda correr completamente en background.

2. Se recomienda difundir la Programación Orientada a Objetos como técnica de programación base en los diversos proyectos a lo largo de la carrera, sustituyendo así el uso de programación meramente estructurada.

3. Es recomendable que la Facultad de Ingeniería Eléctrica dentro del plan de prestación de servicios contemple la necesidad de formar un equipo de programadores para producción de paquetes de software de cualquier tipo a pedido. Lo cual estoy seguro tendrá gran aceptación por parte de las

empresas privadas redundando en beneficios económicos para la institución y generación de trabajo a tiempo parcial para los estudiantes.

BIBLIOGRAFIA

...ERS. IBM PS/2 a Reference Guide, Mc Graw Hill, New York, 1989.

...LEY J.C. Transducers for Microprocessor Systems, Macmillan Publishers ...D, Hampshire, 1985.

...ELLEY, POHL. Lenguaje C Introducción a la Programación, Addison-Wesley ...oamericana, México, 1984.

...ALVINO. Principios de Electrónica. Mc Graw Hill, México, 1989.

...ITCHEL ED. Secrets of the Borland C++ Masters, SAMS Publishing, Indiana, ...

...ATIONAL SEMICONDUCTOR. Data Conversion/Adquisition Databook, ...tional Semiconductor, 1984.

...LAZAR ANGEL. Tesis de Grado "Sistema de Supervisión y Administración ...Computarizado para un Laboratorio de Acuacultura", ESPOL, Guayaquil, 1989.

...RGENT, SHOEMAKER. Personal Computer from the Inside Out, Addison-Wesley Publishing Company, 1984.