48

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Escuela Nacional de Estudios Profesionales
"A R A G Ó N"

RECOPILACIÓN DE DATOS METEOROLÓGICOS POR MEDIO DE UN CONTROL DISTRIBUIDO, DESARROLLADO EN DELPHI, DE UN SISTEMA DE ADQUISICIÓN DE DATOS REMOTO, PARA EL INSTITUTO MEXICANO DEL PETRÓLEO

T E S I S

QUE PARA OBTENER EL TITUTLO DE:

INGENIERO EN COMPUTACION

PRESENTA:

OLIVER MOTA PEREZ

México, D.F. [redacted]

2001

Deseo dar las gracias a mis padres,
Helios y Francisca, por su apoyo
y paciencia y quiero dedicarles
este trabajo como muestra de
dicho agradecimiento

También dedico este trabajo a
mi hermana, Vivian.

Además deseo agradecer a
mis sinodales:
Ing. Roberto Cortes Buenrostro
Ing. Silvia Vega Muytoy
Ing. Oscar Álvarez Meléndez
Ing. Enrique García Guzmán
Ing. Juan Manuel López Carreto

Agradezco al Instituto Mexicano del Petróleo por las facilidades proporcionadas para realizar este trabajo.

En la Subdirección de Protección del IMP agradezco a:
Dr. Francisco Guzmán López-Figueroa

En la Gerencia Ciencias del Ambiente del IMP agradezco a:
Dra. Ma. Esther Ruiz Santoyo
Ing. Emmanuel González Ortiz

Y en especial a:
Lic. Paulo Cesar Mendoza García
Biol. José Manuel Moran Ortiz
por su asesoría en el desarrollo de este trabajo.

A mis amigos de siempre:
Ricardo Peña
Juan Carlos Rodríguez
Juan Carlos Tovar

Sin ustedes no lo hubiera logrado.

En la Universidad a:
Ing. Federique Jáuregui Renaud quien me recomendó para hacer posible este trabajo

Resumen

El presente trabajo fue para proporcionar al Instituto Mexicano del Petróleo, a PEMEX y a la sociedad en general una forma alternativa para comunicar los datos meteorológicos de un lugar remoto de forma directa aplicando una de las herramientas que pronto serán parte de nuestra vida misma, el Internet.

En este documento se presentaron los temas que sirvieron como plataforma para el desarrollo de un sistema aun más complejo y eficiente para proyectos futuros de sistemas de adquisición de datos distintas a las de la medición ambiental. No se menciona en este estudio de forma detallada el funcionamiento de cada parte de la Interfaz sino que son un preámbulo para el investigador o el estudiante interesado que quiera aprender mas allá de estos temas.

Indice de Materias

Resumen v

Indice de Materias vi

Lista de Figuras xii

Lista de Tablas xiv

Capítulo I

Introducción 1-1
1.1 Antecedentes[1] 1-2
1.2 Planteamientos del Problema 1-5
1.3 Objetivos 1-5
1.3.1 Objetivo General 1-5
1.3.2 Objetivos Particulares 1-5
1.4 Hipótesis 1-6
1.5 Importancia del Estudio 1-6
1.6 Limitaciones del Estudio 1-7

Capítulo II

Conceptos Básicos de Control[5] 2-1
2.1 Historia del Control 2-2
2.2 Las Variables Involucradas 2-3
2.3 Control Manual Común 2-4
2.4 Control de Retroalimentación 2-5
2.5 El control de procesos y Manejo de Procesos 2-6
Recapitulación 2-7

Capítulo III

Estructura Funcional del Control de Retroalimentación [5] 3-1
3.1 Un Ciclo de Control de Retroalimentación Sencillo 3-2
3.2 Diagramas de Bloque 3-3
3.3 El Esquema Funcional de un Ciclo de Retroalimentación 3-4
Recapitulación 3-6

Capítulo IV

Sensores y Sistemas de Transmisión [5] 4-1
4.1 El Sensor y el Transmisor 4-2
4.2 Selección de Unidades de Censado 4-3
4.3 Exactitud y Precisión 4-4
4.4 Sensibilidad y Repetitividad 4-5
4.5 Sistemas de transmisión 4-6
4.6 Sistema de transmisión eléctrica 4-7
4.7 Bus de campo digital 4-7
Recapitulación 4-8

Capítulo V

Mediciones Típicas 5-1
5.1 Sensores / transmisores de presión [6] 5-2
5.11 Barómetro [1] 5-3
5.3 Sensor/Transmisor de nivel [7] 5-5
5.4 Medición de la Temperatura usando Termocuplas [8] 5-7
5.6 Mediciones analíticas [5] 5-9
5.6.1 Medición pH 5-10
5.6.2 Cromatografía 5-11
5.6.3 Infrarrojos 5-11
5.6.4 Conductividad termal 5-11
5.6.5 Otros 5-11
Recapitulación 5-12

Capítulo VI

Arquitecturas de sistemas de control moderno 6-1
6.1 Componentes básicos [9] 6-2
6.1.1 Sensores y transmisores 6-2
6.1.2 Unidades de contacto 6-3
6.1.3 Elementos Finales de Control 6-3
6.1.4 Controladores 6-3
6.1.5 Desplegar / Registrando 6-4
6.2. Componentes del sistema [5] 6-4
6.2.1 Computadoras 6-4
6.2.2 Unidades de almacenamiento en masa 6-5
6.2.3 Software 6-5
6.2.4 Redes de trabajo de campo 6-6
6.5. Estructura del sistema de control [5] 6-6
6.6. Sistemas de control distribuido (DCS) [5] 6-7
Recapitulación 6-9

Capítulo VII

Desarrollo de Aplicaciones de Base de Datos en Delphi 7-1
7.1 Usando bases de datos [10] 7-2
7.1.1 Tipos de Bases de datos 7-3
7.1.1.1 Base de datos locales 7-3
7.1.1.2 Servidores de bases de datos remotos 7-4
7.1.2 Seguridad de base de datos 7-5
7.1.3. Transacciones 7-6
7.1.4 El Diccionario de Datos 7-7
7.1.5 Integridad referencial, almacenaje de procedimientos, y lanzadores (triggers) 7-7
7.2 Diseñando la interface usuario [11] 7-8
7.2.1 Desplegando un solo registro 7-9
7.2.2 Desplegando múltiples de registros 7-9
7.2.3 Analizando datos 7-11
7.2.4 Eligiendo que tipos de datos mostrar 7-11
7.2.5 Escribiendo reportes 7-13
Recapitulación 7-14

Capítulo VIII

Computación Distribuida 8-1
8.1 Fundamento Común [12] 8-2
8.1.1 Comunicación de la Red 8-2
8.1.2 Transmisión Síncrona y Asíncrona 8-3
8.1.3 Clientes, Servidores y Par (Peer) 8-4
8.1.4 APIs - Interfaces de Programación de Aplicación. 8-4
8.1.5 Interfaces de Terminal 8-5
8.1.6 Mensajes 8-6
8.1.7 RPC - Llamada de Procedimiento Remoto 8-7
8.2 Cliente/Servidor [12] 8-7
8.3 Middleware [12] 8-8
8.3.1 DCE - Ambiente de Computación Distribuida 8-8
8.3.2 Mensajería Formal 8-9
8.4 Objetos Distribuidos 8-10
8.4.1 JAVA RMI - Invocación de Método Remoto 8-11
8.4.2 DCOM - Modelo de Objeto de Componente Distribuido [15], [16] 8-11
8.4.3 CORBA - Arquitectura de Agente de Petición de Objeto Común [17], [18] 8-13
Recapitulación 8-14

Capítulo IX

Componentes de una Estación Meteorológica 9-1
9.1 Tipos de Monitoreo [19] 9-2
9.2 Partes que comprenden una Estación Meteorológica [19] 9-3
9.2.1 Información Meteorológica 9-3
9.2.2 Información Topográfica 9-3
9.2.3 Clasificación de uso del suelo [22] 9-3
9.2.4 Equipo [20] 9-4
9.2.5 Sensores Meteorológicos [1] 9-4
9.2.6 Modelos Climatológicos [1] 9-6
9.2.7 Gráficas para interpretación 9-7
Recapitulación 9-9

Capítulo X

Sistema de Comunicación Industrial 10-1
10.1 Material 10-2
10.1.1 Sensores con voltaje 0-10V 10-2
10.1.2 Chasis SCXI-1000 / Tarjeta DAQ-1200 10-2
10.1.3 NI-DAQ 6.2 10-3
10.1.4 NI-ComponentWorks 2.0 10-3
10.1.5 Personal Web Server 10-3
10.1.6 Internet explorer 10-3
10.1.7 Intranet / Internet 10-4
10.2 Procedimiento 10-4
10.2.1 Etapa I 10-4
Lógica de Trabajo "InterfazChaz" 10-6
Lógica de Trabajo "Interfaz" 10-7
10.2.2 Etapa II 10-8
Lógica de Trabajo "ClienteOCX" 10-8
Lógica de Trabajo "ServidorOCX" 10-9
10.2.3 Etapa III 10-11
10.2.4 Etapa IV 10-12

Capítulo XI

Manual de Operación 11-1
11.1 Descripción 11-2
11.1.1 Interfaz 11-2
11.1.2 InterfazChasis 11-3
11.1.3 ServidorSAD 11-3
11.1.4 ClienteSAD 11-3
11.2 Usando la Documentación 11-3
11.2.1 Para recuperar información 11-3
11.2.2 Para instalar, remover o cambiar propiedades de un sensor 11-4

11.2.3 Para cambiar el tiempo de muestreo 11-4
11.3 Requerimientos del Sistema 11-4
11.3.1 Requerimientos de Hardware 11-4
11.3.2 Requerimientos de Software 11-5
11.4 Instalando SADMeteoro® 11-6
11.4.1 Antes de Instalar 11-6
11.4.1.1 Configuración de Fecha en Computadoras (C y S) 11-7
Fecha 11-7
Hora 11-7
11.4.1.2 Instalación de software secundario (C - S) 11-7
11.4.1.3 Instalación de SADMeteoro® (S) 11-8
11.4.2 Durante la Instalación 11-8
11.4.3 Al Final de la Instalación 11-8
11.5 SADMeteoro® 11-9
11.5.1 Características del SADMeteoro® 11-9
11.5.1.1 Programa Interfaz 11-9
Carpeta Descripción 11-9
Carpeta Datos 11-9
Carpeta Funcionamiento 11-9
11.5.1.2 Programa InterfazChasis 11-9
11.5.1.3 Programa ServidorSAD 11-10
11.5.1.4 Programa ClienteSAD 11-10
Módulo 1 11-10
Módulo 2 11-10
Módulo 3 11-10
11.5.2 Recuperación y Graficación de Datos 11-10
11.5.2.1 Recuperar datos usando Interfaz 11-11
11.5.2.2 Recuperar datos usando Internet Explorer 11-13
11.5.2.3 Graficar los datos recuperados usando Interfaz 11-14
11.5.2.4 Graficar los datos recuperados usando Internet Explorer . 11-14
11.5.3 Modificación del tiempo de recopilación 11-14
11.5.3.1 Modificar el tiempo de muestreo 11-14
11.5.3.2 Activar o Desactivar el muestreo 11-15
11.5.4 Agregar, modificar y remover Sensores 11-15
11.5.4.1 Agregar un sensor 11-15
11.5.4.2 Modificar propiedades de sensores 11-16
11.5.4.3 Activar un sensor 11-16
11.5.4.4 Remover un sensor 11-16
11.5.4.5 Cancelar cambios 11-17
11.5.5 Descripción del Programa 11-17
11.5.5.1 Interfaz 11-17
Módulo 1 Interfaz 11-17
Módulo 2 Propiedades del Sensor 11-18
Módulo 3 Ayuda Online 11-19
11.5.5.2 InterfazChas 11-19
Pantalla de funcionamiento 11-20
Botón de activación 11-20
Pantalla de muestreo 11-20
Botón de Configuración 11-20

11.5.5.3 ServidorSAD 11-20
Botones Activa y Desactiva 11-20
11.5.5.4 ClienteSAD 11-20
Barra de Herramientas 11-21
Módulo de datos 11-21
Módulo de gráfica 11-22
Módulo de simulación 11-23
11.5.6 Problemas 11-23
11.5.5.1 Instalación 11-24
Desinstalar versiones anteriores 11-24
Carpeta de instalación 11-24
11.5.5.2 Probando tu Instalación 11-24
Para probar un sitio Web de su intranet 11-24
Para probar un sitio Web que no está conectado a una red 11-25
11.5.6.3 Problemas Técnicos o Comunes 11-25

Resultados A-1

Conclusiones B-1

Anexos C-1
ANEXO A - MICROSOFT® PERSONAL WEB SERVER© C-2
Instalar PWS C-2
Para instalar Personal Web Server C-3
ANEXO B - NI-DAQ 6.5.2© C-3
Errores con NI-DAQ C-3
Configuración del chasis SCXI-1000 y de la tarjeta SCXI-1200 ..... C-4
ANEXO C - BORLAND DATABASE ENGINE© C-4
ANEXO D - DCOM C-4
ANEXO E - MICROSOFT INTERNET EXPLORER 5.0 ..... C-5
ANEXO F - SOPORTE TÉCNICO C-5
ANEXO G - ComponentWorks 2.0 [20] C-6
ANEXO H - Chasis SCXI-1000 [20] C-6
ANEXO I - Tarjeta DAQ SCXI-1200 [20] C-6
ANEXO J - NI_DAQ 6.2 [20] C-6
ANEXO K - CODIGO INTERFAZCHAS C-7
ANEXO K - CODIGO INTERFAZ C-14
ANEXO K - CODIGO ServidorOCX C-22
ANEXO L - CODIGO ClienteOCX C-26
Información del Sistema C-46

Referencias Bibliográficas D-1

Glosario E-1

Lista de Figuras

Fig. 2.1 Las variables involucradas.
Fig. 2.2 Un sistema de calefacción para casa.
Fig. 2.3 Control manual tipico.
Fig. 2-4 Concepto de Control de Retroalimentación
Fig. 2-7 Control y Manejo de Proceso

Fig. 3-1 Un Ciclo Sencillo de Retroalimentación.
Fig. 3-2 Diagrama de Bloques para un Ciclo Simple
Fig. 3-3 El Esquema Funcional de un Ciclo de Retroalimentación.

Fig. 4.1. Exactitud y Precisión en una Medición de Temperatura
Fig. 4.2. Error Dinámico y Estático

Fig. 5.1 Los Tres Tipos de Referencias de Presión
Fig. 5.2 Barómetro de Mercurio
Fig. 5.3 Una Medición de Nivel de Presión Diferencial
Fig. 5.4 Un Sensor/Transmisor de Nivel Tipo-Desplazamiento
Fig. 5.5 Circuito de Termocupla
Fig. 5.6 Compensación Automática de Unión Fría
Fig. 5.7 Un Ensablaje Típico de un Termocupla
Fig. 5.8 Electrodos de Medición y Referencia de ph

Fig. 6.1. Posibles Organizaciones para un Sistema de Control
Fig. 6.2. Sistema de Control Distribuido (DCS)

Fig. 8.1 Comunicación de red de trabajo Continua.
Fig. 8.2 Arquitectura Middleware
Fig. 8.3 Arquitectura de Objetos Distribuidos.

Fig. 9.1 Un sistema DAQ Típico.
Fig. 9.2 Rosa de Vientos [21]
Fig. 9.3 Símbolos Meteorológicos [1]
Fig. 9.4 Mapa Meteorológico [1]

Fig. 10.1 Plan de Trabajo General
Fig. 10.2 Diagrama general de conexión de sensores.
Fig. 10.3 Diagrama de flujo general del programa "InterfazChaz".
Fig. 10.4 Diagrama de flujo general del programa "Interfaz".
Fig. 10.5 Diagrama de flujo general del programa "ClienteOCX".
Fig. 10.6 Diagrama de flujo general del programa "ServidorOCX".
Fig. 10.7 Diagrama electrónico para conectar sensor analógico.
Fig. 10.8 Diagrama electrónico para conectar sensor digital.

Fig. 11.1 Interfaz sin conexión
Fig. 11.2 Interfaz con propiedades de sensores

Fig. 11.3 Datos de un sensor
Fig. 11.4 Pagina principal del servidor Web personal.
Fig. 11.5 Componente ClienteOCX con datos recuperados
Fig. 11.6 Display de periodo de muestro de InterfazChaz
Fig. 11.7 Botón de activación de muestro del sistema
Fig. 11.8 Pantalla de captura del un sensor.
Fig. 11.9 Detalle sobre activar un sensor
Fig. 11.10 Barra de herramientas del componente ClienteOCX
Fig. 11.11 Detalle datos recuperados en ClienteOCX
Fig. 11.12 Detalle de datos graficados en el ClienteOCX.
Fig. 11.13 Detalle gráfica en ClienteOCX

Lista de Tablas

Tabla 5-1 Sensores de Nivel
Tabla 5-2 Los Rangos Útiles de Varios Sensores de Temperatura
Tabla 5-3 Los Tipos de Termocuplas mas Comunes

Tabla 11.1 Requerimientos de Hardware
Tabla 11.2 requerimientos de Software

CAPITULO I

Introducción

1.1 Antecedentes[1]

Los estudiosos de la antigua Grecia mostraban gran interés por la atmósfera. Ya en el año 400 a.C. Aristóteles escribió un tratado llamado Meteorológica, donde abordaba el "estudio de las cosas que han sido elevadas"; un tercio del tratado está dedicado a los fenómenos atmosféricos y el término meteorología deriva de su título. A lo largo de la historia, gran parte de los progresos realizados en el descubrimiento de leyes físicas y químicas se vio estimulado por la curiosidad que despertaban los fenómenos atmosféricos.

La predicción del tiempo ha desafiado al hombre desde los tiempos más remotos, y buena parte de la sabiduría acerca del mundo exhibida por los diferentes pueblos se ha identificado con la previsión del tiempo y los almanaques climatológicos. No obstante, no se avanzó gran cosa en este campo hasta el siglo XIX, cuando el desarrollo en los campos de la termodinámica y la aerodinámica suministraron una base teórica a la meteorología. Las mediciones exactas de las condiciones atmosféricas son también de la mayor importancia en el terreno de la meteorología, y los adelantos científicos se han visto potenciados por la invención de instrumentos apropiados de observación y por la organización de redes de observatorios meteorológicos para recoger datos. Los registros meteorológicos de localidades individuales se iniciaron en el siglo XIV, pero no se realizaron observaciones sistemáticas sobre áreas extensas hasta el siglo XVII. La lentitud de las comunicaciones también dificultaba el desarrollo de la predicción meteorológica, y sólo tras la invención del telégrafo a mediados del siglo XIX se hizo posible transmitir a un control central los datos correspondientes a todo un país para correlacionarlos a fin de hacer una predicción del clima.

Uno de los hitos más significativos en el desarrollo de la ciencia moderna de la meteorología se produjo en tiempos de la I Guerra Mundial, cuando un grupo de meteorólogos noruegos encabezado por Vilhelm Bjerknes realizó estudios intensivos sobre la naturaleza de los frentes y descubrió que la interacción entre masas de aire genera los ciclones, tormentas típicas del hemisferio norte. Los posteriores trabajos en el

campo de la meteorología se vieron auxiliados por la invención de aparatos como el rawinsonde o radiosonda, que hizo posible la investigación de las condiciones atmosféricas a altitudes muy elevadas. Después de la I Guerra Mundial, un matemático británico, Lewis Fry Richardson, realizó el primer intento significativo de obtener soluciones numéricas a las ecuaciones matemáticas para predecir elementos meteorológicos. Aunque sus intentos no tuvieron éxito en su época, contribuyeron a un progreso explosivo en la predicción meteorológica numérica de nuestros días.

Las observaciones hechas a nivel del suelo son más numerosas que las realizadas a altitudes superiores. Incluyen la medición de la presión atmosférica, la temperatura, la humedad, la dirección y velocidad del viento, la cantidad y altura de las nubes, la visibilidad y las precipitaciones (la cantidad de lluvia o nieve que haya caído).

La recopilación de datos sobre el clima se logra sobre todo por medio de la transmisión vía teletipo de mensajes codificados, a través de líneas terrestres y de la radio. Los circuitos nacionales de teletipos operan como líneas multiusuario, y los datos impresos por cualquier estación aparecen al mismo tiempo en todas las demás estaciones conectadas a la misma línea. Los datos recopilados a nivel nacional se intercambian a través de circuitos globales a larga distancia de alta velocidad, con lo que, en cerca de una hora, los informes sobre la superficie y las capas superiores de la atmósfera están disponibles en los centros regionales de muchos países. El sistema global de telecomunicaciones de la Organización Mundial de Meteorología actúa como centro de recepción y transmisión de los datos que proceden de las estaciones de superficie y los satélites meteorológicos, así como de los que proceden de barcos, aviones y radiosondas.

En el margen de dos horas desde la recogida de los datos, hay mapas climatológicos disponibles en los centros de previsión meteorológica. El uso del fax ha multiplicado la eficiencia de estos centros, ya que los mapas son trazados por analistas expertos y están al alcance de los meteorólogos de campo en mayor variedad y con mayor rapidez de lo que antes era posible, cuando se trazaban de forma local. Ciertos análisis de las

condiciones en la atmósfera superior son realizados de modo automático por medio de ordenadores o computadoras que, con periféricos adicionales, pueden traducir y almacenar la información codificada de las líneas de teletipo, realizar cálculos matemáticos y presentar los resultados en forma de líneas trazadas sobre mapas. Tales análisis se transmiten vía fax a las estaciones locales y son almacenados para su empleo en previsiones climatológicas numéricas.

La mejora en las observaciones de los vientos a gran altitud durante y después de la II Guerra Mundial suministró la base para la elaboración de nuevas teorías sobre la predicción del tiempo y reveló la necesidad de cambiar viejos conceptos generales sobre la circulación atmosférica. Durante este periodo las principales contribuciones a la ciencia meteorológica son del meteorólogo de origen sueco Carl-Gustav Rossby y sus colaboradores de Estados Unidos. Descubrieron la llamada corriente en chorro, una corriente de aire de alta velocidad que rodea el planeta a gran altitud. En 1950, gracias a las primeras computadoras, fue posible aplicar las teorías fundamentales de la termodinámica y la hidrodinámica al problema de la predicción climatológica, y en nuestros días las grandes computadoras sirven para generar previsiones en beneficio de la agricultura, la industria y los ciudadanos en general.

La recopilación de datos sobre el clima se logra sobre todo por medio de la transmisión vía teletipo de mensajes codificados, a través de líneas terrestres y de la radio. Los circuitos nacionales de teletipos operan como líneas multiusuario, y los datos impresos por cualquier estación aparecen al mismo tiempo en todas las demás estaciones conectadas a la misma línea. Los datos recopilados a nivel nacional se intercambian a través de circuitos globales a larga distancia de alta velocidad, con lo que, en cerca de una hora, los informes sobre la superficie y las capas superiores de la atmósfera están disponibles en los centros regionales de muchos países. El sistema global de telecomunicaciones de la Organización Mundial de Meteorología actúa como centro de recepción y transmisión de los datos que proceden de las estaciones de superficie y los satélites meteorológicos, así como de los que proceden de barcos, aviones y radiosondas.

Actualmente, una de las necesidades que el Instituto Mexicano del Petróleo presenta, es que sus sistemas de adquisición de datos puedan compartir la información de manera remota y directa. El instituto actualmente trabaja con un sistema de monitoreo meteorológico terrestre, el cual trabaja de forma similar a la Organización Mundial de Meteorología, donde los datos, provenientes de las estaciones de superficie que se encuentran instaladas dentro de las refinerías, son recopilados y transmitidos. Dichas estaciones cuentan con una torre que tiene instalados los sensores meteorológicos, y una unidad de recopilación de datos (datalogger), que realiza mediciones simultáneas de los sensores en determinado período de tiempo, y almacena esta información. La recuperación de datos que efectúa el Instituto se logra por medio de una recuperación manual, lo que por consecuencia su centro de recepción y trasmisión de datos analizados no puedan compartir esta información de forma inmediata con sus clientes.

1.2 PLANTEAMIENTOS DEL PROBLEMA

El propósito del presente trabajo fue proporcionar una nueva forma de recopilar y transmitir datos directamente de los sensores meteorológicos desde el lugar de origen con una conexión a una Intranet o a Internet.

1.3 OBJETIVOS

1.3.1 Objetivo General

Desarrollar una interfaz que permita consultar los datos de los sensores meteorológicos por medio de una intranet.

1.3.2 Objetivos Particulares

- Desarrollar una interfaz con conexión remota.
- Instalar tarjetas de adquisición de datos compatibles en una computadora para leer cualquier tipo de sensor meteorológico
- Crear un sitio WEB para la Interfaz

1.4 HIPÓTESIS

Si se logra desarrollar una interfaz que use una Intranet entonces se podrán consultar datos remotos de sensores meteorológicos desde cualquier computadora que use un navegador de internet comercial.

1.5 IMPORTANCIA DEL ESTUDIO

Con este trabajo de investigación se pretendió mejorar la distribución de la información meteorológica dentro y fuera del instituto, ocupando su Intranet. Además se propuso agregar una nueva forma recopilar datos meteorológicos a las técnicas existentes.

Esta interfaz proporciona una forma alterna de recuperar datos meteorológicos, pero la característica principal es que permite consultar los datos de los sensores desde cualquier lugar que tenga conexión a una red de trabajo.

Con esta aplicación se pretende proporcionar una solución a las refinerías de PEMEX para comunicar las estaciones meteorológicas con las que actualmente trabajan, con lo cual, se que podrá integrar la información ambiental a todas la áreas dentro de las instalaciones de la refinería y a su vez compartir esta información a otras empresas o a las comunidades o poblaciones aledañas o circundantes a las refinerías que tengan una conexión a Internet.

Lee Mikles opina que: "Las aplicaciones en internet ofrecen mejorar la flexibilidad y el flujo de información y se toman mejores decisiones para qué diseñadores y directores implanten herramientas" [2].

Sin embargo este tipo de comunicación no desplaza a los sistemas actuales, según dicen David Wofford, Adris Bartle y Jerry Diven, al afirmar que "Todos los informes pueden ser vistos en línea, ser enviados por telefax o ser impresos" [3].

A su vez comenta S. Zafar Kamal "Hacer que el información esté disponible en el lugar correcto, en el tiempo correcto, es la piedra angular de una manufactura integrada"[4].

En resumen, no importa como viaje información sino que ésta esté disponible a tiempo cuando se le necesite.

1.6 LIMITACIONES DEL ESTUDIO

A continuación se presentan las limitaciones del estudio:

- la interfaz trabaja sólo con base de datos PARADOX, sin embargo el sistema puede modificarse para que sea capaz de trabajar con otro tipo de bases de datos.
- la interfaz puede ser vista por MS-Internet Explorer 4.0 o superior por lo que no podrá ser visto con otros navegadores de Internet.
- la interfaz fue probada con sensores de dos marcas distintas, sin embargo es capaz de soportar cualquier otro tipo de sensor meteorológico, consulte la ficha técnica en los Anexos al final de este documento.
- la interfaz sólo fue probada en el sistema operativo Windows 98 y Windows 95 por lo que se recomienda no instalar la interfaz en otro sistema operativo.

CAPITULO II

Conceptos Básicos de Control [5]

Introducción

Este capítulo introduce a los conceptos básicos encontrados en un proceso de control; además presenta terminología básica.

2.1 Historia del Control

El primer uso bien definido del control de retroalimentación lo realizó la aplicación de Watt acerca de un regulador automático de la trayectoria en forma de arco (flyball governor) para la máquina de vapor en 1775. Como tema de interés, la mayoría de las aplicaciones siguientes y las investigaciones teóricas fueron asociadas con los reguladores automáticos y estos eran comúnmente utilizados en aplicaciones industriales.

Muchas nuevas tecnologías que han sido aplicadas al hardware de control de procesos, como el uso industrial de técnicas de automatización, han desarrollado y madurado en los últimos 50 años. Un ejemplo importante fue la aplicación de las capacidades de las computadoras digitales para el control de proceso; en este caso, el proceso de automatización recibió un fomento significativo y muy especial. Hoy día, muchas industrias asignan más del 10% de su gasto de capital de inversión para instrumentación y control. Su porcentaje se ha duplicado en los últimos 30 años y no muestra signos de disminuir.

La teoría fundamental del control automático también se ha desarrollado rápidamente, y ha sido creada una forma y un fundamento extenso de comprensión. Hoy día las aplicaciones están basadas en este fundamento. Sin embargo, la mayoría de los profesionales modernos encuentran dificultad, en la aplicación de teorías matemáticas bien-definidas del control de proceso automático. Mucha de esta dificultad, es completamente natural, pero gran parte de los problemas están en el hecho que casi siempre la enseñanza insuficiente está dirigida para ilustrar principios teóricos en aplicaciones industriales del diario.

2.2 Las Variables Involucradas

Para entender el control de proceso automático, es necesario primero tener en mente tres importantes términos asociados con cualquier proceso. Estos están ilustrados en la Fig. 2-1. Las *cantidades controladas* o *variables controladas* son aquellos cursos (flujos) o condiciones que el profesional desea controlar o mantener en un nivel deseado. Estas cantidades controladas o variables controladas pueden ser el índice de flujo, niveles, presiones, temperaturas, mezclas, etc. Para cada una de las variables controladas, el profesional también establece algún *valor deseado* o *setpoint* o *entrada de referencia.*

Para cada cantidad controlada, hay una *cantidad manipulada* o *variable manipulada* asociada. En el control de proceso esto es un curso que circula comúnmente, y en tales casos el índice de flujo del curso casi siempre es manipulada a través del uso de alguna válvula de control.

Los *disturbios* entran al proceso y tienden a manejar las cantidades controladas o variables controladas fuera de las condiciones deseadas o referencia o setpoint. La necesidad es entonces por un sistema de control automático para ajustar las cantidades manipuladas para que el valor de setpoint de la cantidad controlada sea conservada a pesar de los efectos de los disturbios. También, el setpoint puede ser cambiado y entonces las variables manipuladas necesitarán ser cambiadas para ajustar cantidades controladas, a su nuevo valor deseado.

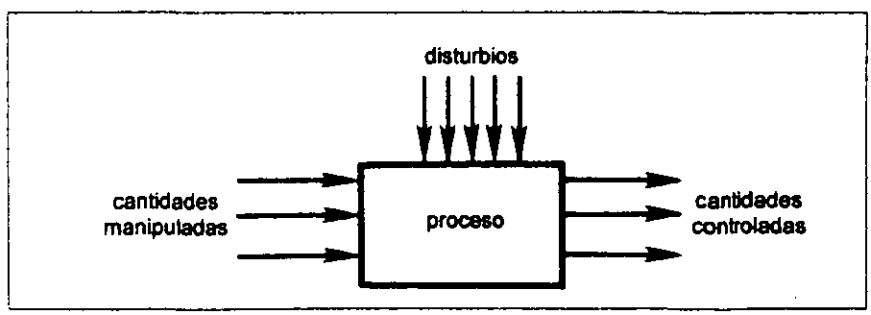


Fig. 2-1. Las variables involucradas

La Fig. 2-2 muestra un sistema típico de calefacción de casa. En tal sistema, la variable controlada es la temperatura del cuarto. (Actualmente, si deseas mantener una temperatura confortable en el cuarto, comúnmente controlas una variable que puede ser medida fácilmente, como la temperatura.) Un número de fuerzas provocan que la temperatura varíe, por ejemplo, la temperatura ambiental externa, el número de personas en el cuarto, el tipo de actividad que se lleva a cabo en el cuarto, etc. El sistema de control automático está designado para manipular el flujo de gasolina que va a la estufa a fin de mantener la temperatura del cuarto en el valor deseado o setpoint a pesar de los diversos disturbios.

2.3 Control Manual Común

Antes de estudiar el control de proceso automático, será de ayuda invertir un momento o dos para repasar una operación manual común. Esto es ilustrado en la Fig. 2-3, un proceso con una cantidad controlada. En la corriente que deja el proceso, existe un indicador que proporciona información al operador del valor de actual de la variable controlada. El operador es capaz de inspeccionar este indicador visualmente y, como resultado, manipular un flujo dentro del proceso para alcanzar algún valor deseado o

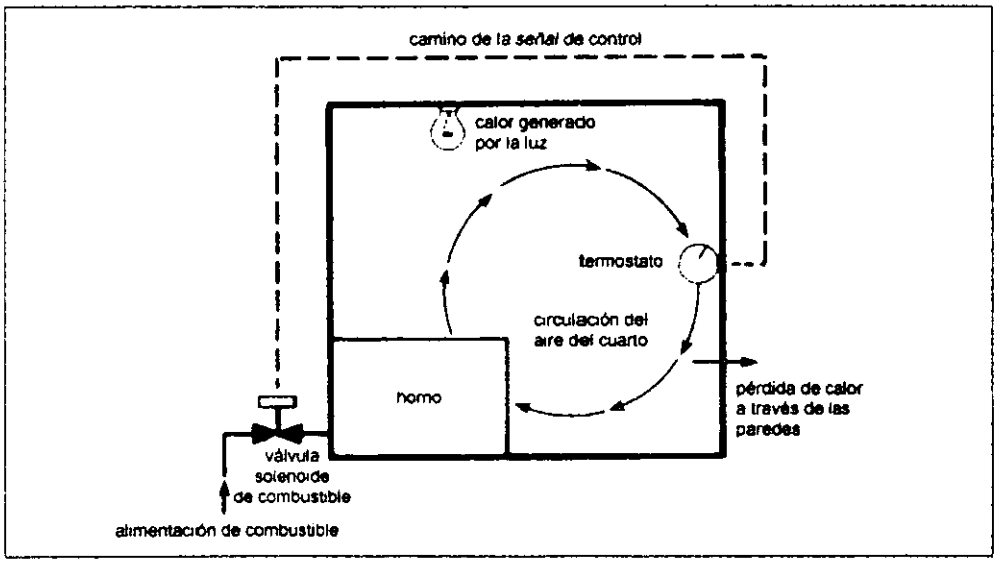


Fig. 2-2. Un sistema de calefacción para casa

setpoint de la variable controlada. El setpoint está, claro, en la mente del operador y el operador hace todas las decisiones. Los problemas inherentes en tal operación manual simple son obvios.

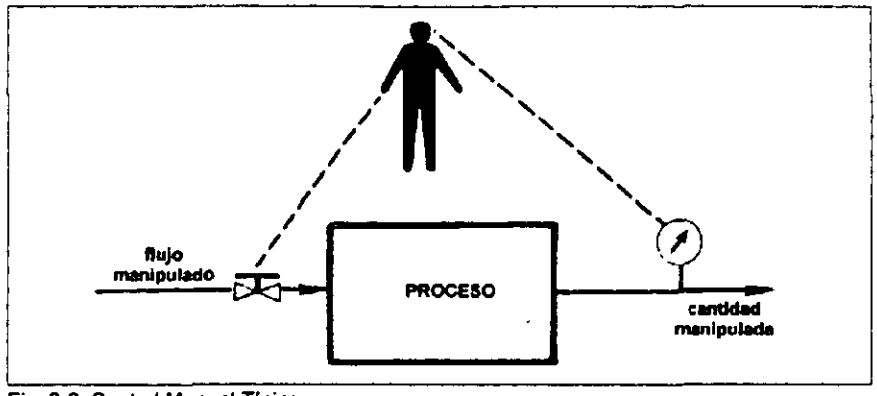


Fig. 2-3. Control Manual Típico

2.4 Control de Retroalimentación

La forma más simple para automatizar el control de un proceso es a través del control de retroalimentación convencional. Este concepto ampliamente usado es ilustrado en la Fig. 2-3. Los sensores o unidades de medición son instaladas para medir los valores actuales de las variables controladas. Estos valores actuales son entonces transmitidos al hardware de control de retroalimentación y este hardware hace una comparación entre los setpoints (o valores deseados) de las variables controladas y los valores medidos (o actuales) de las mismas variables. Basado en la diferencia (error) entre el valor actual y los valores deseados de las variables controladas, el hardware de control de retroalimentación calcula señales que reflejan los valores necesitados de las variables manipuladas. Estos son entonces transmitidos automáticamente para ajustar dispositivos (típicamente valores de control), los cuales manipulan entradas a los procesos.

La belleza del control de retroalimentación es que el diseñador no necesita conocer por adelantado cuales disturbios afectarán el proceso y, además, el diseñador no necesita conocer las relaciones cuantitativas específicas entre estos disturbios y sus efectos finales en las variables controladas. El hardware de control es usado en un formato estándar, y

todos los ciclos de control de retroalimentación tienden a reflejar la estructura conceptual general. Ilustrada en la Fig. 2-4. Para una extensión muy significativa, este patrón estándar existe a pesar de la naturaleza específica del proceso o de las variables controladas envueltas. El hardware particular en un ciclo y la conjunción particular de una pieza de hardware con otra, es una responsabilidad importante para el diseñador, pero toda la estrategia de control está específicamente definida. Tales estrategias de control de retroalimentación son las técnicas de control de proceso automático más sencillas que pueden desarrollarse, y este control de retroalimentación es común en la amplia mayoría de la tecnología de control para su uso en aplicaciones industriales hoy día.

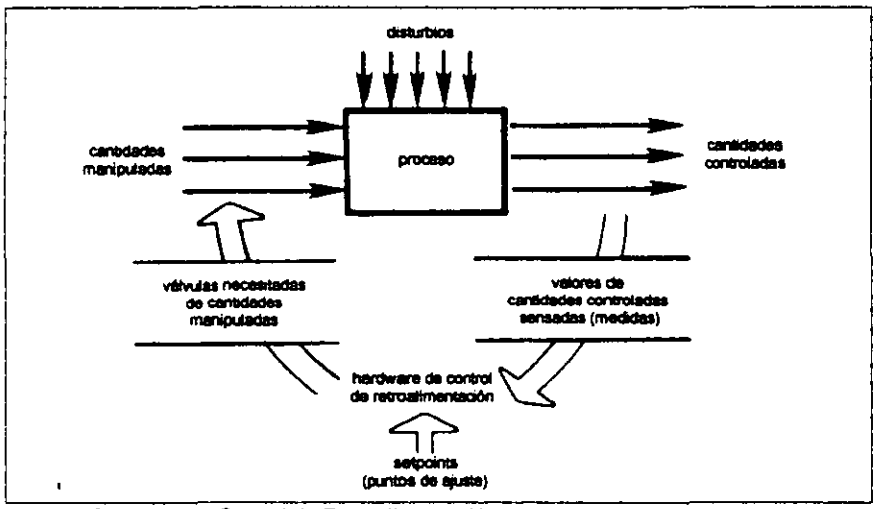


Fig. 2-4. Concepto de Control de Retroalimentación

2.5 El control de procesos y Manejo de Procesos

El proceso de automatización es usado para derivar el máximo de productividad del proceso. En el material presentado anteriormente en este capítulo, se ha hecho una suposición implícita que nosotros conocíamos los valores deseados (deseados para llevar a cabo el máximo de productividad) para las cantidad controladas. Una vez que estos valores deseados son conocidos, las técnicas son automatizadas para llevar a cabo y/o mantener estos valores deseados o setpoints. Con esta reflexión, sin embargo, se puede ver que algunas de las preguntas significativas asociadas con la productividad de un

proceso, son las preguntas que deben ser contestadas para determinar los valores deseados. Esto es básicamente la función de supervisión o de manejo y que casi siempre es dejado completamente al operador humano para que lo determine. Pero, en años recientes, con los avances significativos en proceso de automatización, muchas de estas funciones de supervisión o manejo se han vuelto automatizadas, y la habilidad para llevar a cabo soluciones tecnológicas y respuestas de hardware de dichas preguntas de manejo es una parte significativa de la escena moderna de control.

En un proceso particular, conforme el nivel de automatización es incrementado, la mayoría de los pasos iniciales envuelve el uso de control de proceso convencional (como el control de retroalimentación); pero conforme el nivel de automatización se incrementa, se asocia más y más la automatización con el manejo de procesos. Esto es ilustrado en la Fig. 2-7

La combinación de estos dos fenómenos - control de proceso y manejo de proceso- deben ser reflejados en nuestro completo entendimiento y apreciación, de la automatización del proceso.

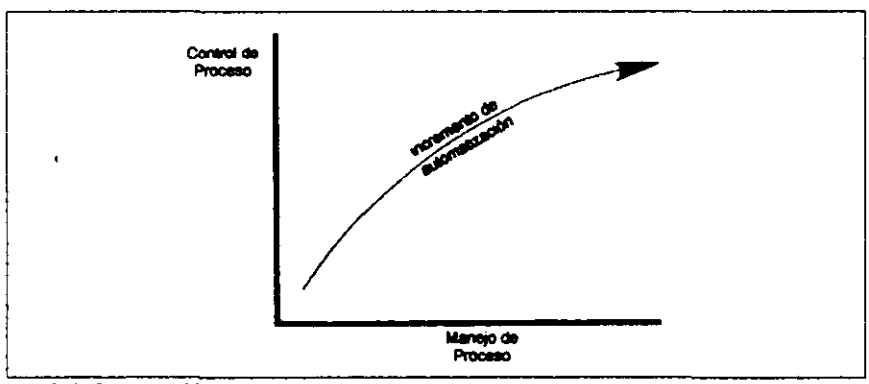


Fig. 2-7. Control y Manejo de Proceso

Recapitulación

Vistos y comprendidos los conceptos y nociones básicas de control, podremos detallar en el siguiente capítulo como se aplica este conocimiento en un sistema de control de retroalimentación.

■

CAPITULO III

Estructura Funcional del Control de Retroalimentación [5]

Introducción

El concepto general de control de retroalimentación fue presentado en el Capítulo 2. Ahora este concepto general es reducido a un perfil funcional para un ciclo de control de retroalimentación.

3.1 Un Ciclo de Control de Retroalimentación Sencillo

Cualquier proceso tendrá un número de variables controladas distintas y para cada variable controlada, existe una variable manipulada asociada que debe ser elegida. Discutido con anterioridad, solo se ha demostrado en términos explícitos y generalizados. Ahora una variable controlada específica es unida a través de un hardware de control de retroalimentación. Esto se realiza de forma ilustrativa en la Fig. 3.1

La variable controlada es censada o medida a través de instrumentación controlada y este valor censado de la variable controlada es entonces comparada con respecto al valor deseado de la variable controlada (el setpoint). La *diferencia* entre estas dos (el error) es usada como entrada al controlador de retroalimentación. Este controlador entonces calcula una señal para ajustar la variable manipulada. Desde que la variable manipulada es normalmente un flujo, la salida del controlador de retroalimentación comúnmente es una señal a la válvula de control, como es ilustrado en la Fig. 3-1. Mientras todo esto esta pasando en un modo continúo, los disturbios pueden entrar al proceso y atender el movimiento de la variable controlada en una dirección o en otra. La variable manipulada simple es usada para compensar todos los cambios producidos por los disturbios y, además, si hay cambios en el punto de ajuste (setpoint), la variable manipulada también es cambiada de acuerdo para producir los cambios necesarios en la variable controlada.

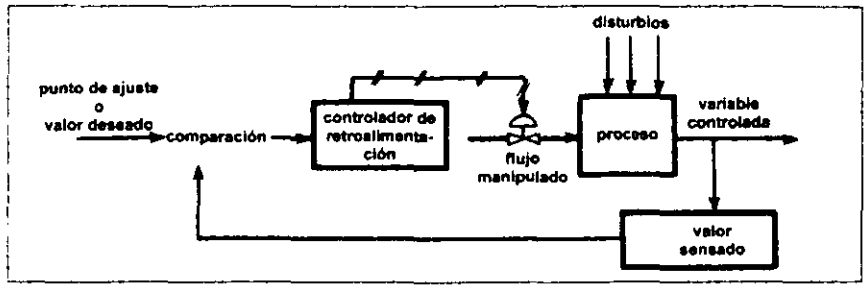


Fig. 3-1 Un Ciclo Sencillo de Retroalimentación.

En un sentido funcional, todas las operaciones de control de retroalimentación son ilustradas en la Fig. 3-1.

3.2 Diagramas de Bloque

Para tener una forma consistente de proporcionar representaciones pictóricas para los sistemas de control, es útil para tomar ventaja de los diagramas de bloque. Los diagramas de bloque son una herramienta gráfica simbólica, simple comúnmente usada en control automático.

Los diagramas de bloques tienen dos símbolos básicos; el primero es un círculo:

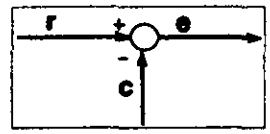
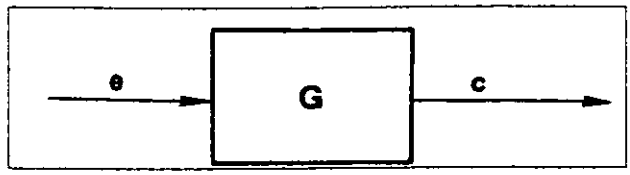


Fig a

Las flechas que entran al círculo y lo dejan no son vectores. Sin embargo, ellas representan variables y actualmente representan el flujo de la información. La cabeza de cada flecha tiene un signo algebraico asociado con él, ya sea mas o menos. El pequeño círculo es realmente una forma simple de representar una suma algebraica o substitución. El símbolo mostrado más arriba en el texto, representa la ecuación algebraica r - c = e

El otro símbolo del diagrama de bloques es, de hecho, un bloque con una flecha entrando y una flecha saliendo.



Fib b

Esta es la forma en la cual las operaciones algebraicas de multiplicación y división son representadas simbólicamente. La salida del diagrama es simplemente igual a lo que está encerrado dentro del bloque regulado en la entrada. El bloque mostrado en texto de la anterior representa la ecuación c = Ge

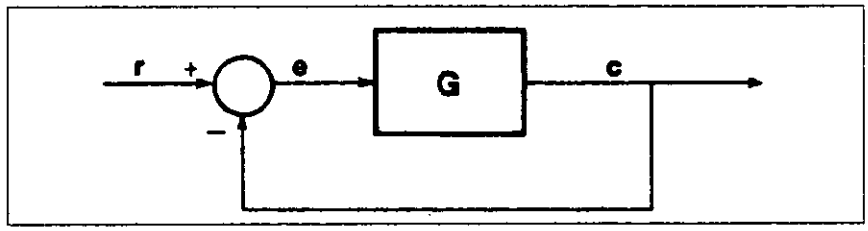


Fig. 3-2 Diagrama de Bloques para un Ciclo Simple

Los símbolos de diagrama pueden estar combinados en redes de trabajo. Lo que se muestra en la Fig. 3-2 es un diagrama de bloque de un ciclo de control de retroalimentación negativo muy simple; este representa una combinación de los dos símbolos mostrados anteriormente.

Los diagramas a bloques son usados de forma regular durante parte de éste capítulo para proporcionar presentaciones pictóricas de los principios y aplicaciones de control automático.

3.3 El Esquema Funcional de un Ciclo de Retroalimentación

El esquema general y estructura de un ciclo de control de retroalimentación simple necesita ser expandido para representar lo más cercano a la forma como es usado en la práctica, el control de retroalimentación. Tal esquema funcional es ilustrado en la Fig. 3-3. Básicamente, la Fig. 3-3 está separada en dos grandes partes: Los objetivos funcionales, están establecidos *dentro de la parte del controlador* y el balance del ciclo del proceso

está, claro, fuera de la parte del controlador.

Está claro, que debe haber medidas para que los operadores o algún hardware, proporcionen un "setpoint" al ciclo de control. Este "setpoint" es el valor deseado de la variable controlada y tendrá las mismas dimensiones que la variable controlada, p.e., si la variable controlada es galones por minuto, entonces el "setpoint" también será galones por minuto. Los elementos de entrada proporcionan una conversión funcional de "setpoint" de entrada al modo de operación del controlador, p.e., milivolts, miliampers, presión del aire psi, etc.

La variable de control es medida actualmente por un sensor y el valor medido de la variable controlada es transmitida de regreso a la parte del controlador. Dentro de la parte del controlador está el *comparador.* Esta importante unidad funcional actualmente compara -toma la diferencia algebraica entre- el valor del "setpoint" (después de la conversión por los elementos de entrada) y los valores de la variable transmitida de regreso a dentro de la parte del controlador para representar la variable controlada. El comparador, o *detector de error*, es común a todo el sistema de control de retroalimentación. Notese que éste es control de retroalimentación negativo, p.e., la señal que alimentó de regreso al comparador, es substraído desde la señal la cual indica el "setpoint". Todo el control de retroalimentación aplicado es control de retroalimentación negativo (control de retroalimentación positivo es inherentemente inestable).

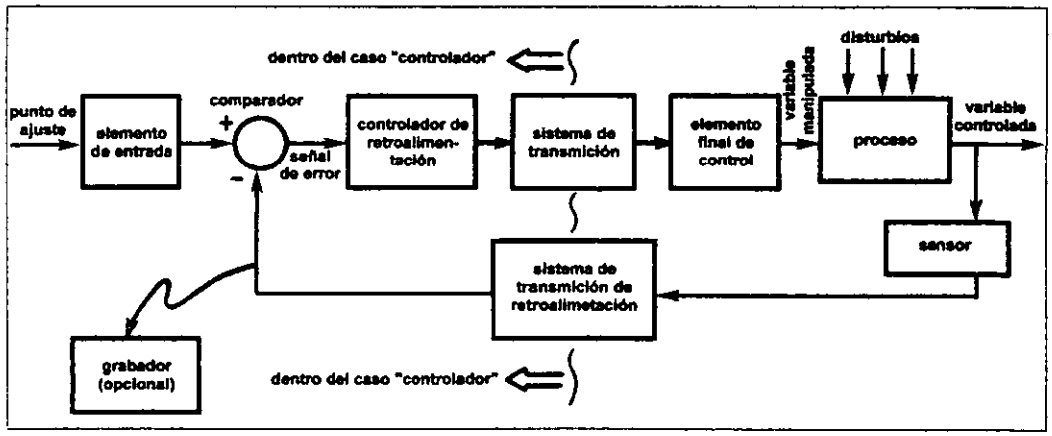


Fig. 3-3. El Esquema Funcional de un Ciclo de Retroalimentación.

La señal de error, la cual es la salida del comparador, se vuelve la entrada al controlador de retroalimentación. Basado en la señal de error, el controlador calcula una señal para el elemento final de control -el cual comúnmente es una válvula de control- y éste en los controles de cambio de la variable manipulada de entrada al proceso.

También ses muestra con el ciclo de control de retroalimentación de la Fig. 3-3 una grabadora (para la variable controlada) la cual es opcional.

En la práctica, mucha gente refiere a todos los elementos contenidos en la parte del controlador como si fuera el controlador de retroalimentación. Este uso, generalizado y amplio, además de común, no debe ser necesariamente preciso y no será usado en esta tesis. En lugar de ello, es mejor hacer distinciones funcionales a través de las diversas operaciones acerca del ciclo de control de retroalimentación y con la parte del controlador.

Recapitulación

Con lo antes expuesto se tiene el conocimiento concreto sobe un ciclo de control de retroalimentación donde se detallaron las partes que comprenden este ciclo para hacer una analogía entre un sistema real y un esquema matemático. En los siguientes capítulos analizaremos las partes que comprenden este ciclo.

■

CAPITULO IV

Sensores y Sistemas de Transmisión [5]

Introducción

La calidad del rendimiento de un sistema de control de retroalimentación es directamente dependiente en la calidad de medición de la variable controlada. Además, este valor medido debe ser transmitido al controlador en cierto modo de tiempo para que la acción correctiva pueda ser iniciada. También, la salida controlada debe ser transmitida hacia la válvula de control. El propósito de este capítulo es comprender el trabajo de estos elementos de medición y transmisión.

4.1 El Sensor y el Transmisor

Uno de los problemas más críticos en el diseño e instalación de un sistema de control de proceso de retroalimentación es la especificación de las unidades de censado que obtendrán una medición continua de la variable controlada. En un sentido de operación, esta unidad de medición no solo proporciona una medición de la variable controlada, sino que también produce un *cambio de la variable.* El cambio de variable toma lugar desde que la variable controlada misma no es la señal actual que se transmite de regreso al comparador. El transmisor produce una señal de salida cuyo valor, estado-estable tiene una relación predeterminada hacia la variable controlada.

Un transmisor no se requiere desde un punto de vista de la medición ni tampoco del de control. Básicamente, el transmisor sirve como una conveniencia operativa al hacer disponible en un lugar más centralizado los datos de medición de la variable controlada, p.e., en un cuarto de control remoto. Casi siempre, desde el punto de vista de hardware, la función de medición y la función de transmisión se incorporan a una sola pieza de hardware.

Las variables controladas principales en sistemas de control de proceso son, en orden descendente de la frecuencia de su concurrencia: *temperatura, presión, rango de flujo, composición*, y *niveles de líquidos.* Algunas de estas variables, como la presión, pueden medirse relativamente de forma directa mientras que otras, como la temperatura,

solo puede ser medido indirectamente.

Un término el cual también debe ser definido es *transductor*. Este es un término general para una unidad que recibe información en la forma de uno o más cantidades físicas, modifica la información o su forma o ambos, y manda una señal de salida. En un sentido, dependiendo que tipo de aplicación está envuelta, un transductor puede ser un elemento de medición primario (un sensor), un transmisor, un relevador, un convertidor, o alguna otra unidad.

Se aprecia que muchos de estos términos -sensor, transmisor, convertidor, transductor- tiene significados amplios y a veces se traslapan y, casi siempre, la instalación particular de piezas específicas de hardware dedicarán el término descriptivo apropiado. La confusión puede minimizar si el profesional se enfoca principalmente en el uso funcional involucrado.

4.2 Selección de Unidades de Censado

Deben ser consideradas muchas preguntas antes de ser elegida una propuesta específica de medición de la variable controlada, para un ciclo particular. No existen reglas difíciles ni rápidas para hacer dichas decisiones, pero hay un número de factores los cuales deben ser considerados:

A. ¿Cual es el rango normal en la cual la variable controlada puede variar; existen extremos para esto?
B. ¿Que exactitud, precisión y sensibilidad son necesarios? (Estos términos son definidos en detalle en la siguiente sección.)
C. ¿Que dinámicas de sensor son necesarias y disponibles?
D. ¿Que fluctuación es requerida?
E. ¿Cuales son los costos involucrados - no solo el costo de compra sino que también la instalación y costos de operación?
F. ¿Existen problemas espaciales de instalación, p.e. fluidos corrosivos, mezclas explosivas, restricción de tamaño y forma, preguntas de transmisión remota, etc.?

Obviamente, con esta larga lista de factores importantes involucrados, por ahora no se mencionará en detalle los aspectos de seleccionar sensores.

4.3 Exactitud y Precisión

La *exactitud* de una medición es el término usado para describir la cercanía con la cual la medición se acerca al valor verdadero de la variable a ser medida.. La *precisión* es la reproducción con la cual repetidas mediciones de la misma variable puede ser hecha bajo condiciones idénticas. En materia de control de proceso, la última característica es más importante que la exactitud, p.e. normalmente se desea medir una variable precisa que una que tenga un alto grado de exactitud absoluta. La distinción entre esas dos propiedades de medición es mostrada en la Fig. 4.1.

La curva punteada es una indicación de la temperatura actual de un fluido. La medición superior ilustra un instrumento preciso pero inexacto, mientras que la medición inferior muestra la medición dada por un instrumento impreciso pero más exacto. El primer instrumento tiene el mayor error, el último instrumento muestra el gran *desplazamiento.*

Los profesionales hacen una distinción entre dos tipos de exactitud: *estática* o *exactitud estado-estable* y *exactitud dinámica.* La exactitud estática es el acercamiento más cercano al valor verdadero de la variable, cuando ese valor verdadero es constante. Exactitud dinámica, por otra parte, es el acercamiento más cercano de la medición cuando el valor verdadero es cambiante. Estos términos son ilustrados en la Fig. 4.2. Claramente, el valor numérico de la precisión dinámica dependerá de la naturaleza del cambio dinámico hecho por el valor verdadero de la variable al ser medida. Además, las

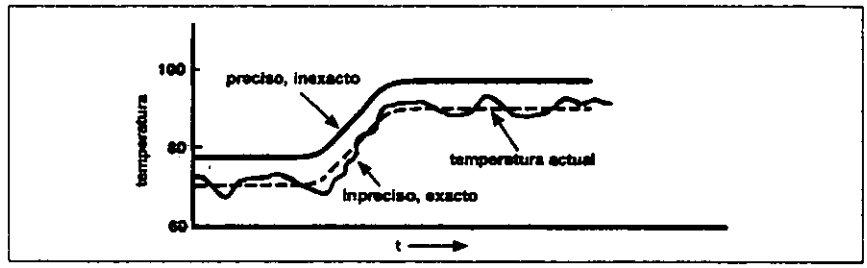


Fig. 4.1. Exactitud y Precisión en una Medición de Temperatura

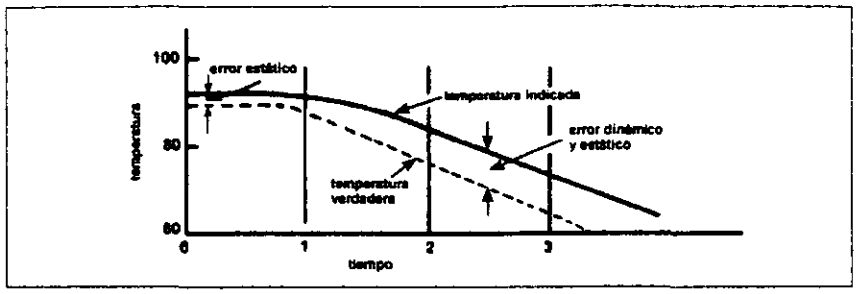


Fig. 4.2. Error Dinámico y Estático

propiedades del sistema de medición mismas tendrán un efecto. Para sistemas de control de proceso, un especificación práctica del tiempo de variación es el forzamiento tipo rampa, como se muestra en la Fig. 4.2, y una designación práctica de la exactitud dinámica es el *error dinámico* que resulta del forzamiento tipo rampa.

4.4 Sensibilidad y Repetitividad

La *sensibilidad* de una unidad de medición está definida por el promedio del cambio de señal de salida con respecto al cambio en una variable medida. Claramente, entre más grande el cambio de la señal de salida para un cambio de entrada dado, mayor la sensibilidad del elemento de medición. La sensibilidad es una relación estado-estable y, de hecho, es una *ganancia* estado-estable.

Existe otra forma de sensibilidad la cual es muy importante en sistemas de medición. Esta sensibilidad está definida como el cambio más pequeño en la variable medida la cual producirá un cambio en la señal medida desde el elemento censado. En muchos sistemas físicos, especialmente aquellos que contienen palancas, articulaciones y partes mecánicas, existe una tendencia para que esas partes móviles sean clavadas y tengan algún juego libre. El resultado es que señales de entrada pequeñas no pueden producir señales de salida detectables. Se necesitan instrumentos bien-diseñados y bien-construidos, para que la sensibilidad pueda ser mayor, y para que el sistema de control tenga la habilidad de respuesta a cambios pequeños en la variable controlada.

La repetitividad es la aproximación al entendimiento entre mediciones consecutivas de la misma variable de proceso. La repetitividad puede ser especificada como un porcentaje de toda la escala o un porcentaje de promedio.

4.5 Sistemas de transmisión

Cuando el sensor mide la variable controlada, el valor medido debe ser transmitido en alguna manera al controlador; esto puede ser a varios metros o a varios miles de metros. De forma similar, esto es necesario para obtener información preveniente del controlador hasta el elemento final de control; comúnmente, esto es una válvula de control.

Por años, muchos sistemas de transmisión de control de proceso usaron entubado neumático para transmitir información como una señal de presión de aire. Pocos de estos sistemas de transmisión neumáticos son usados hoy en día; pero existe una gran base de ellos instalados en plantas existentes. Estos sistemas neumáticos introducen un gran tiempo de retraso dentro de la dinámica de proceso del ciclo de control y, de esta manera, influye en la forma en la cual el ciclo trabaja- a veces una desventaja muy seria. Por todas estas razones, dedicaremos un poco de atención a los sistemas de transmisión neumática.

El medio de transmisión más común es el alambre de cobre, ya sea en la forma de par trenzado o en la forma de cable coaxial. El par trenzado ha sido usado grandemente por años y son los sistemas de transmisión común. Las tecnologías inalámbricas pueden ser útiles, especialmente en instalaciones móviles o temporales. El cable de fibra óptica también puede ser usado; pero es muy caro, y es difícil hacer múltiples empalmes.

Muchos sistemas de transmisión y de operación están disponibles, este documento solo se avocará a mencionar los de tipo eléctrica 4-20mA DC.

4.6 Sistema de transmisión eléctrica

Los sistemas de transmisión más comunes de señales eléctricas es 4-20 mA. Un par trenzado de alambre de cobre es usado para formar un ciclo de corriente dc. La corriente es preferida sobre el voltaje ya que es más inmune al ruido y requiere sólo de dos alambres. Existe un par trenzado individual de alambres para cada señal, y, para una planta de proceso de fluido, esto crea rápidamente la necesidad de miles de pares trenzados. Son muy caros para instalar y la instalación puede costar más tiempo que el proceso de control mismo.

El alambrado de campo debe también proporcionar el aislamiento galvánico de señales que provienen del equipo de proceso de tal forma que previene ciclos de tierra. Esto también es comúnmente instalado en lugares peligrosos que requieren medidas a prueba de explosión o seguridad intrínseca.

Es posible considera el uso de un sensor de salida directa como una señal, por ejemplo, use la salida mV de una termocupla de una señal directa. Los problemas de ruido, señales débiles, etcétera, normalmente conducen al uso de señales condicionadas en la forma de "transmisor" como asistente de transmisión.

Existen dos tipos de sistemas de transmisión: sistemas de dos-alambres y sistemas de cuatro-alambres. En el sistema de dos alambres, la corriente que alimenta el condicionador/transmisor también acarrea la señal. El tipo de cuatro alambres tiene alambres separados para la alimentación.

4.7 Bus de campo digital

Debido a los altos costos del alambrado de campo, el incremento de carga colocado en un sistema de transmisión, y la necesidad de transmitir señales digitales directamente, existe una clara necesidad de un estándar de bus de campo digital. Varios propietarios de bus de campo están actualmente en el mercado.

Las características de un bus de campo digital son tales que es muy seguro, y esto comúnmente es acompañado de redundancia. Esto también debe ser rápido, con respuesta de tiempos desde microsegundos hasta milisegundos. Esto también debe ser capaz de alimentar las unidades de campo a grandes distancias. Se han hecho propuestas para hacer un estándar, pero habrá que esperar antes de que sea aceptado como un estándar. Los vendedores ahora tienen sus propios estándares.

Recapitulación

Los sensores son una parte fundamental dentro de un sistema de control ya que de estos depende directamente el sistema donde están instalados, al igual que los sistemas de transmisión que transportan la información que dan los transductores. En este capítulo se distinguieron los tipos de sensores, al igual que los sistemas de transmisión, además del comportamientos que tienen al estar instalados y la calidad de medición que se requiere según sea su sistema

■

CAPITULO V

Mediciones Típicas

Introducción

Para que funcione un proceso de control o un sistema de adquisición de datos meteorológicos, se debe ser capaz de censar/medir las variables importantes del proceso. Este capítulo dará un breve estudio de algunos de los sensores más comunes usados para medir variables ambientales.

5.1 Sensores / transmisores de presión [6]

La presión es una de las mediciones más comunes y más importantes asociadas con el control de proceso. La mayoría de los meteorólogos están familiarizados con los elementos de medición de presión como nanómetro y barómetro. Estos elementos asumen que una persona debe estar ahí para leerla. Para que sea útil en un control de proceso, un elemento de medición de presión debe tener un transmisor de presión que producirá una señal de salida para transmitirlo, como puede ser una corriente eléctrica que es proporcional a la presión que va a ser medida. Comúnmente, un transmisor produce una salida de señal 4-20 mA, muy austera, y es adecuada para usarse en unidades inflamables o peligrosos.

La presión está definida como la fuerza por unidad de área; algunas unidades comunes son de libras por pulgada cuadrada, pulgadas de agua, pulgadas de mercurio, atmósferas, bars y torrs. La presión es siempre medida con respecto a estas tres

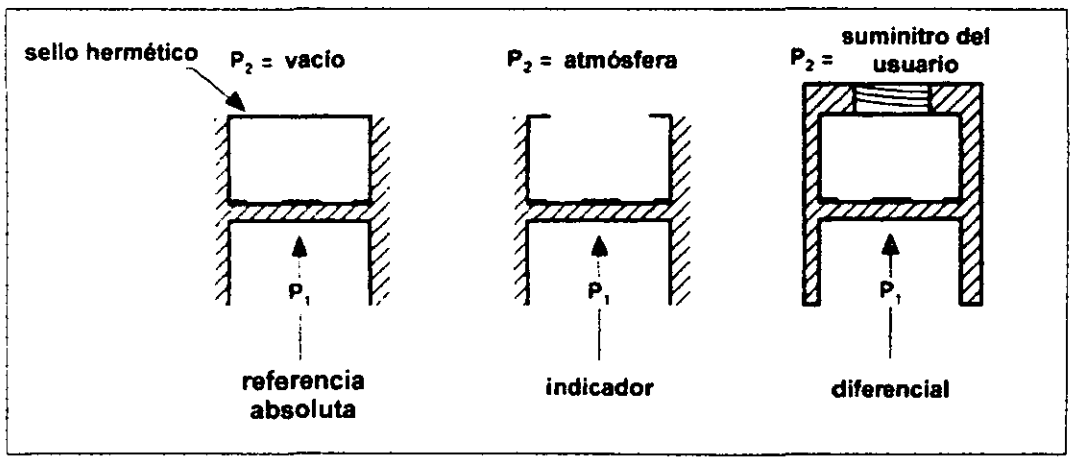


Fig. 5-1 Los Tres Tipos de Referencias de Presión

referencias comunes. Uno, si la referencia es un vacío, la presión medida es llamada precisión absoluta. Dos, si la referencia es la presión ambiental local, la presión medida es llamada presión manométrica. Tres, si la referencia es proporcionada por el usuario, la presión medida es llamada preción diferencial. La figura 5-1 ilustra estos tres tipos de referencias; P2 es la presión referencia.

Una de las formas más comunes para medir la presión es a través del uso de sensores de indicación de deformación como se muestra en la figura 5.1 b. La presión produce una deformación en el diafragma que causa un desplazamiento del diafragma, el cual es medido y traducido en una medición de presión. En la figura 5.1 b, el indicador de deformación, es una resistencia eléctrica cuya resistencia varía en función de la deformación. Cuatro de estos indicadores de deformación son delimitados a un diafragma metálico. Cuando la presión es aplicada, dos de estos cuatro indicadores de deformación estarán comprimidos (su resistencia disminuye) y dos están en tensión (su resistencia aumenta). Los cuatro detectores de deformación están conectados en un circuito de puente de Wheatstone para producir una señal proporcional eléctrica a la deformación/desplazamiento/presión.

La envoltura del transmisor contiene no sólo el indicador de deformación sino también una señal de condición, una señal portadora, y un conector eléctrico. La señal de condición es todo lo electrónico necesario para compensar, estabilizar, ajustar, identificar la señal. La fuente de presión es abierta a través de la señal de puerta, y el conector eléctrico es solo esto.

5.11 Barómetro [1]

Instrumento para medir la presión atmosférica, es decir, la fuerza por unidad de superficie ejercida por el peso de la atmósfera. Como en cualquier fluido esta fuerza se transmite por igual en todas las direcciones. La forma más fácil de medir la presión atmosférica es observar la altura de una columna de líquido cuyo peso compense exactamente el peso de la atmósfera. Un barómetro de agua sería demasiado alto para

resultar cómodo. El mercurio, sin embargo, es 13,6 veces más denso que el agua, y la columna de mercurio sostenida por la presión atmosférica normal tiene una altura de sólo 760 milímetros.

Un barómetro de mercurio ordinario está formado por un tubo de vidrio de unos 850 mm de altura, cerrado por el extremo superior y abierto por el inferior. Cuando el tubo se llena de mercurio y se coloca el extremo abierto en un recipiente lleno del mismo líquido, el nivel del tubo cae hasta una altura de unos 760 mm por encima del nivel del recipiente y deja un vacío casi perfecto en la parte superior del tubo. Las variaciones de la presión atmosférica hacen que el líquido del tubo suba o baje ligeramente; al nivel del mar no suele caer por debajo de los 737 mm ni subir más de 775 mm. Cuando el nivel de mercurio se lee con una escala graduada denominada nonius y se efectúan las correcciones oportunas según la altitud y la latitud (debido al cambio de la gravedad efectiva), la temperatura (debido a la dilatación o contracción del mercurio) y el diámetro del tubo (por los efectos de capilaridad), la lectura de un barómetro de mercurio puede tener una precisión de hasta 0,1 milímetros.

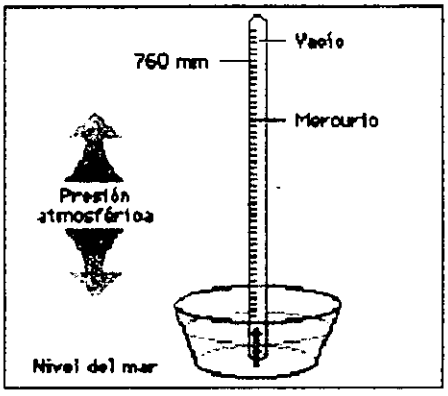


Fig. 5.2 Barómetro de Mercurio [1]

5.3 Sensor/Transmisor de nivel [7]

Los sensores pueden estar agrupados en siete categorías de acuerdo al principio de censado del nivel primario. Esto se muestra en la Tabla 5.1. La señal producida por una de estas mediciones deben ser traducidas a señales eléctricas, neumáticas, o señales digitales para usarse en el control de proceso.

El censado de nivel es más de forma intuitiva no-entendible que otras variables de proceso; por ello, le daremos al tema mostrado aquí una atención mínima. Unos cuantos ejemplos servirán para ilustrar.

La Fig. 5.3 muestra como un transmisor de presión diferencial puede ser usado para medir el nivel de un líquido bajo presión. Nota que la salida de un transmisor depende sólo del nivel y no de la presión estática en el tanque. Esto es un ejemplo de un elemento de nivel del tipo presión.

(A)	Varilla para medir profundidad	
(B)	Visión	Tubo Indicador Colador Flotador de cinta
(C)	Fuerza	Diafragma Báscula Bouyancy
(D)	Presión	Cabeza Hidrostática Chorro Continuo Presión Diferencial
(E)	Eléctrica	Sondas de Capacitancia Sensores de Conductancia Sensores de Resistencia
(F)	Detectores Ultrasónicos y Sónicos	
(G)	Otros	Detectores Infrarrojos Detectores de Microondas Detectores tipo Nuclear Detectores tipo Termal

Tabla 5-1. Sensores de Nivel [7]

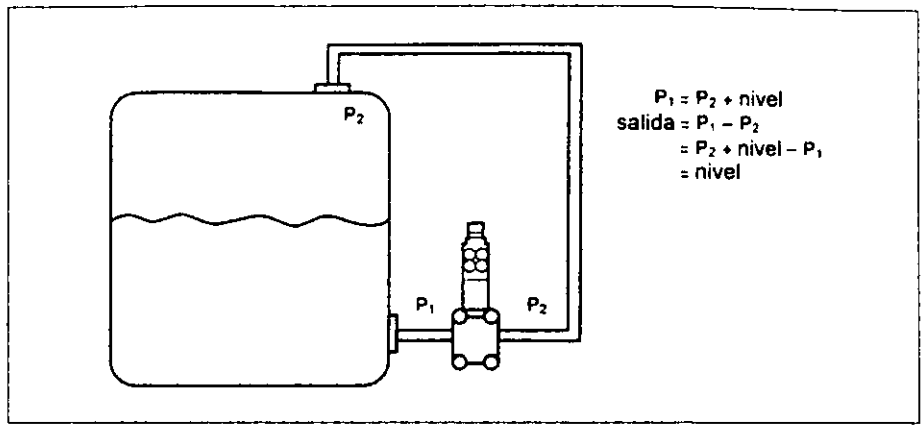


Fig. 5.3 Una Medición de Nivel de Presión Diferencial

Un sensor de nivel del tipo desplazado, y su transmisor son mostrados en la Fig. 5.4; éste es un ejemplo de unidad de nivel del tipo fuerza.

Las pruebas de capacitor son un ejemplo de detectores de nivel del tipo eléctrico. Una prueba es insertada dentro del tanque cuyo nivel va a ser determinado. La prueba es un capacitor variable; este comprende dos materiales de conducción separado por un

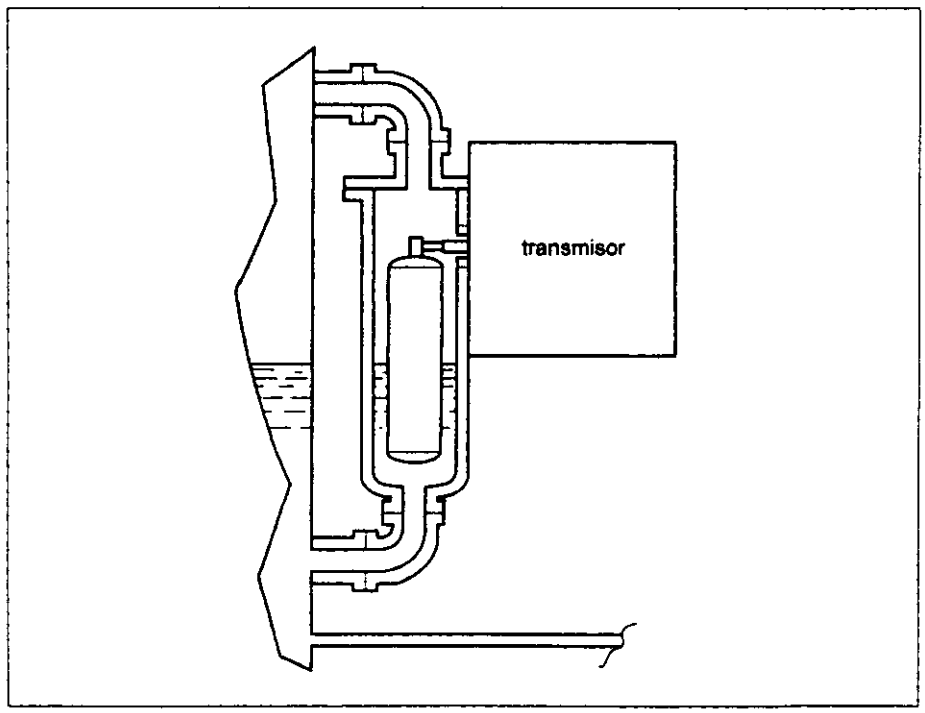


Fig. 5.4 Un Sensor/Transmisor de Nivel Tipo-Desplazamiento

aislante. La capacitancia del capacitor es determinada por el área de los conductores, la distancia entre ellos, y la constante dieléctrica del aislante. Comúnmente, la prueba es un electrodo, el tanque metálico es el otro. Conforme el nivel cambia, el liquido cubrirá o envolverá la prueba, con esto se cambia la capacitancia.

5.4 Medición de la Temperatura usando Termocuplas [8]

Los sensores de temperatura usados comúnmente son:

- Termocuplas,
- Termómetro de resistencia,
- Sistemas de llenado, y
- Pirómetros de radiación.

El rango de temperaturas en donde éstos diversos tensores pueden ser usados es mostrado en la Tabla 5-2.

Los elementos o unidades más comunes para la medición de temperatura es la

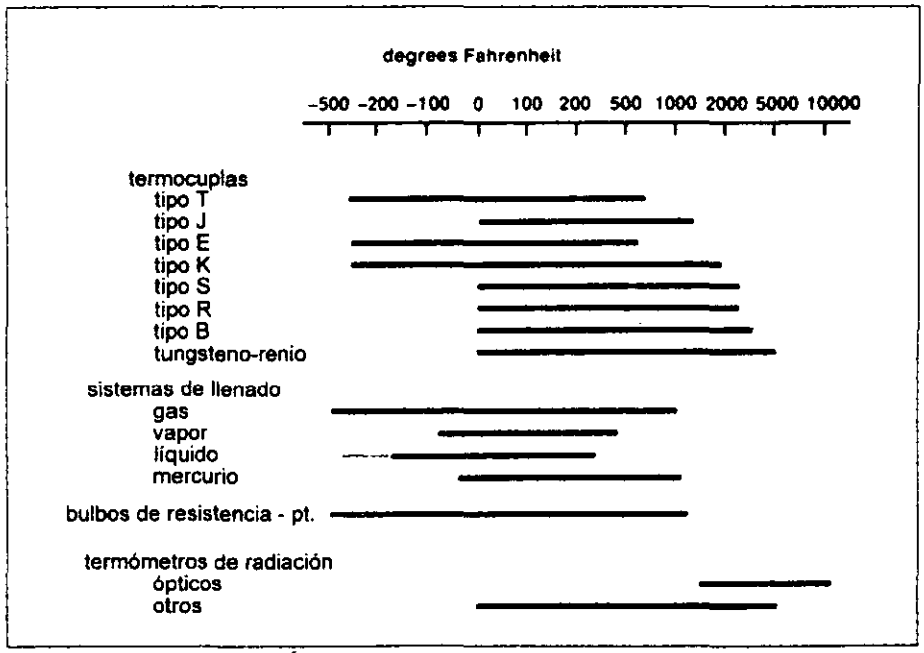


Tabla 5-2 Los Rangos Útiles de Varios Sensores de Temperatura

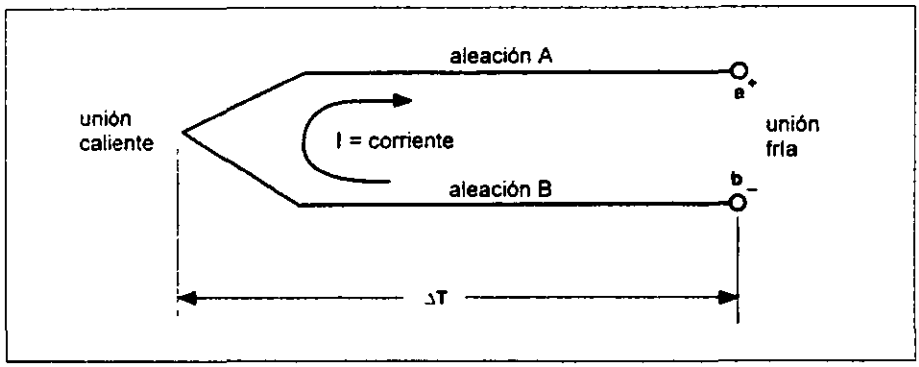


Fig. 5.5 Circuito de Termocupla

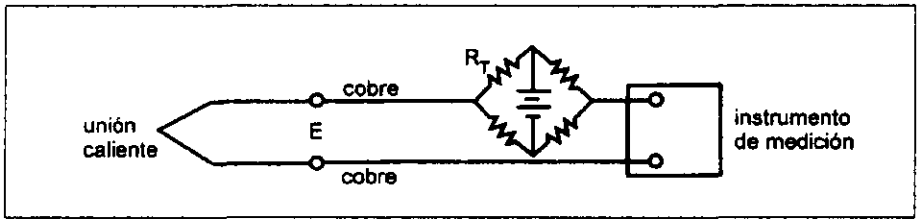


Fig. 5.6 Compensación Automática de Unión Fría

termocupla. Hay dos alambres de metales distintos, unidos en un extremo es llamado el "caliente" o, más apropiadamente, la unión de medición (vea la Fig. 5.5). En el extremo frío, el voltaje de circuito abierto, llamado el voltaje "Seedbeck", es medido. Este voltaje "Seedbeck" depende de la diferencia en temperatura entre las uniones caliente y fría, y el coeficiente "Seedbeck" de los dos metales. La corriente "Seedbeck" I continuará, si las terminales a y b están casi juntas.

El voltaje "Seedbeck" puede ser usado para indicar la temperatura solo si la temperatura de la unión fría se conoce o si cualquier cambio en la temperatura de la unión fría es compensada. En un circuito, esto puede ser hecho como se muestra en la Fig. 5.6, donde se proporciona un resistor RT de compensación de unión fría. Nota también en la Fig. 5-6 que el alambre de cobre es usado como alambre "extensión" para el par de termocupla.

El tipo de termocupla más común se dan en la Tabla 5-3. Un ensamblaje de termocupla industrial típico es mostrado en la Fig. 5-7.

Tipo de Termocupla	Rengo de Temperatura	Límites de Error	
		Estándar	Premium
T	0-350°C, 0-660°F	± 1°C or ± 0.75%	± 5°C or 0.4%
	200-0°C, 300°-32°F	± 1°C or ± 1.5%	—
J	0-750°C, 0-1400°F	± 2.2°C or ± 0.75%	± 1.1°C or ± 0.4%
E	0-900°C, 0-1650°F	± 1.7°C or ± 0.5%	± 1°C or ± 0.4%
	-200°-0°C, 300°-32°F	± 1.7°C or ± 1%	—
K	0-1250°C, 0-2300°F	± 2.2°C or ± .75%	± 1.1% or ± 0.4%
	- 200°-0°C, -300°-32°F	± 2.2°C or ± 2%	—
R,S	0-1450°C, 32-2700°F	± 1.5°C or ± .25%	± 0.6°C or ± 0.1%
B	800-1700°C, 1500-3100°F	± 0.5%	—
Tungsten-rhenium 24 ga.**	0-2930°C, 32-4200°F	± 13.3°C or 1%	—

* Cualquiera que sea mayor
** Pequeños indicadores tienen límites menores de temperatura

Tabla 5-3 Los Tipos de Termocuplas Más Comunes

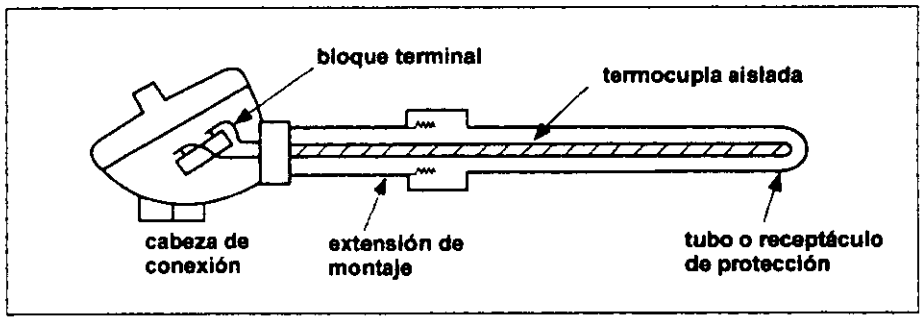


Fig. 5-7 Un ensamblaje Típico de una Termocupla

5.6 Mediciones analíticas [5]

Los sensores analíticos - comúnmente llamados *analizadores* - son usados para medir composiciones, usualmente de una solución de algún tipo. La atención está en la medición de la concentración. La mayoría de estas mediciones son inferenciales, por ejemplo, una medición de alguna propiedad es usada inferencialmente para medir la concentración de algún componente en la mezcla. Existe una gran variedad de dichos sensores y veremos algunos de ellos.

5.6.1 Medición pH

pH es una medición de una solución si es ácida o es base; este mide el "potencial" de los iones de hidrógeno. El agua pura, absolutamente neutral, tiene un pH de 7.0; menor a 7.0 es ácido; mayor que 7.0 es base. La escala corre desde cero a 14. Matemáticamente, es definida como:

$$pH = pH - \log CH$$

Donde CH = concentración de iones de hidrógeno en gm-moles/litro.

El pH se mede al establecer una célula galvánica, la cual desarrolla un potencial eléctrico desde una reacción química en los electrodos.

Se usan dos electrodos, cada uno tiene una mitad de célula galvánica interna; un electrodo es el electrodo de medición y el otro es el electrodo de referencia (vea la figura 5.8). Cuando estos dos electrodos son sumergidos en la solución cuyo pH va hacer medido, ellos pueden ser conectados externamente desde un circuito cerrado; una

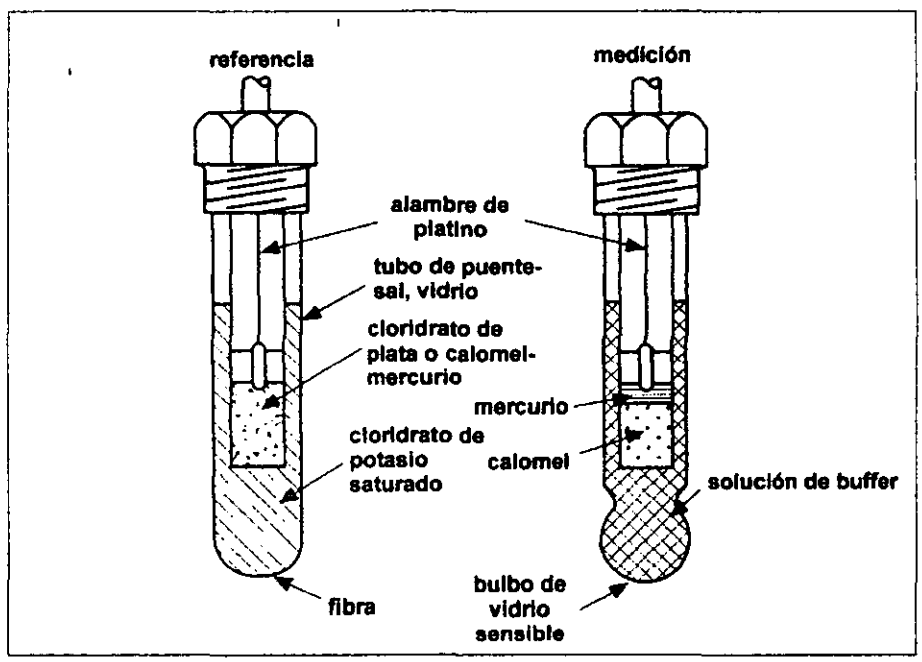


Fig. 5.8 Electrodos de Medición y Referencia de ph

corriente fluirá y puede ser medido para indicar el pH.

5.6.2 Cromatografía

La Cromatografía es especialmente útil para medir la concentración de componentes químicos en una muestra de múltiples-componentes. Comúnmente, la muestra se evapora si ésta no se encuentra ya en ese estado; entonces, el vapor es sacado en un flujo de gas portador inerte, el cual fluye a través de un tubo pequeño llenado con material absorbente. Por medio de repetidas absorciones y desabsorciones, los componentes de muestra emergerán del tubo en diferentes tiempos. Un sensor detecta cada componente conforme emerge del tubo. Además de la cromatografía de gas, es posible hacer cromatografía líquida.

5.6.3 Infrarrojos

Usado para medir concentración las técnicas de el uso de infrarrojos son basadas en el hecho que las moléculas químicas diferentes absorben diferentes combinaciones de frecuencias de radiación infrarroja. Los patrones de frecuencia absorbidas identifican la composición molecular y la cantidad absorbida puede ser usada para medir la concentración. Es útil para gases y líquidos.

5.6.4 Conductividad termal

Esta técnica es útil para gases y vapores. Está basado en el hecho que las diferentes sustancias tienen diferentes habilidades para conducir el calor. Aplicando el principio para medir, la conductividad termal no es medida; en vez de esto, la conductividad es comparada con el aire y medida con una condición específica.

5.6.5 Otros

La lista de analizadores es casi infinita, por ejemplo, conductividad eléctrica, analizadores de oxígeno, detectores de ionización de flama, analizadores de dióxido de carbono, espectrómetros de masa, etcétera. Los que discutimos aquí son sólo para propósitos ilustrativos. El lector serio ciertamente necesita explorar lo antes mencionado

con mayor profundidad.

Recapitulación

En este capítulo pudimos analizar el funcionamiento de algunos sensores que censan fenómenos físicos que son usados, (algunos, dependiendo del tipo de análisis ambiental que se vaya a realizar), en un sistema meteorológico. Sin embargo se hace mención de otros sensores que pueden ser útiles para el lector si busca información sobre otros tipos de sensores más sofisticados o que requiera de mediciones para otros fenómenos.

CAPITULO VI

Arquitecturas de sistemas de control moderno

Introducción

Esta unidad se enfoca en los componentes de ciclos de control individual y como estos elementos son examinados en sistemas de control mayores. La estructura de estos sistemas mayores es ilustrado y se mencionarán algunas de sus ventajas y desventajas.

6.1 Componentes básicos [9]

6.1.1 Sensores y transmisores

La temperatura, el flujo, la presión, y nivel son las cuatro variables más medidas en los sistemas de control de proceso; pero muchas, muchas otras variables como el pH, la velocidad, la humedad, la posición, la composición, etcétera pueden también ser medidas. Los sensores que hacen estas mediciones pueden ser simples o pueden ser muy complejas.

Estos sensores están conectados desde su lugar de campo hasta el lugar de control-registro-despliegue que pueden estar a un lado o remotamente. La señal transmitida puede tomar muchas formas diferentes; pero hoy día 4-20 mA es el estándar más común. La mayoría de las variables tienen su propia conexión de alambre de par trenzado única; pero las señales pueden ser multiplexadas para salvar costos de alambrado.

Algunos sensores son inherentemente digitales, y la tendencia es proporcionar más y más sensores con capacidades de salida digital. También, muchos sensores son construidos "inteligentes" con un microprocesador interno para mejorar las mediciones para los sistemas de control. Las salidas de los sensores digitales pueden ser convertidos de regreso a análogas para transmitirlas en un alambre 4-20 mA, pero existe una pérdida de precisión y de repetición. Esto pone presión en la necesidad para buses de campo digitales, ahora ellos se están empezando a instalar.

6.1.2 Unidades de contacto

Señales on-off son una parte virtual de cualquier instalación de control. Ellos se necesitan para indicar el proceso o la condición del equipo, por ejemplo, indicadores de posición de válvulas, unidades de seguridad, condicionadores de iniciado de motor, switches de nivel, etcétera. Tales señales son usadas, entre otros propósitos, para indicar la condición, abrir y cerrar válvulas, iniciar y apagar equipo.

6.1.3 Elementos Finales de Control

Las válvulas de control casi siempre son los elementos finales de control en un ciclo. Estas válvulas comúnmente tienen un actuador neumático para posicionar la válvula. Las señales provenientes del controlador que van al elemento final de control son comúnmente 4-20 mA y estas deben convertirse a una presión de aire, comúnmente 3-15 psig, para operar el actuador. Los posicionadores de válvula tienen la capacidad I / P. También es posible tener posicionadores de válvula que aceptan una señal directamente, y las tendencias son muy parecidas a éstas.

Existen otros tipos de elementos finales de control además de las válvulas de control. Los ejemplos incluyen unidades de velocidad variable para bombas y motores, unidades on-off, y otros. Las conversiones de señales apropiadas se necesitan para convertir la señal que se va a transmitir del controlador a la forma en que la necesita un actuador específico.

6.1.4 Controladores

Los controladores analógicos, tanto neumáticos o electrónicos, fueron las unidades de control estándar por años. Muchos de estos controladores analógicos tuvieron interfaces digitales que permitieron cambiar ("switching") de manual a automático, ajustar el "setpoint" o ajustar la señal de salida de la válvula. Virtualmente en todos los casos, la puesta a punto del controlador fue hecha a mano.

La mayoría de los controladores están basados en un microprocesador. Muchos están diseñados para manipular un ciclo simple, mientras que otros pueden manipular arreglos

multi-ciclos (vea la sección 6-6). Ellos proporcionan un control del tipo PID, pero ya están disponibles algoritmos mucho más avanzados. Muchos están provistos de puesta a punto automático. Las interfaces para otro equipo de sistema de control están provistos de interfaces digitales, que trabajan comúnmente en conjunto con un bus.

6.1.5 Desplegar / Registrando

Los sistemas viejos de control analógicos están orientados a grandes paneles para desplegar y registrar datos. Típicamente, esto está en un cuarto de control separado, el cual también contiene percheros de terminado para alambrado de campo y tal vez una consola computacional para propósitos relacionados con el control. A veces, los desplegados de datos y el equipo de registro están compartidos a través de muchos ciclos; en sistemas viejos, cada ciclo podía tener el propio.

Los sistemas novedosos están orientados a la forma digital y usan tubos de rayos catódicos o pantallas de cristal líquidos para desplegar la información. Las gráficas usadas para facilitar este desplegado pueden ser muy impresionantes. Las entradas del operador con el sistema es a través del teclado, lápices ópticos, o paneles sensibles ("toch panels o touch screen"). Estas estaciones operador pueden estar localizadas de forma central o ellas pueden estar distribuidas a través de la planta vía LAN o Intranets.

6.2. Componentes del sistema [5]

6.2.1 Computadoras

En los años de mil novecientos veinte, un sistema de máquinas tuvieron comúnmente un enorme motor elevado, en una parte, y un gran eje de dirección del tamaño del cuarto. Las máquinas individuales usaron una banda de dirección para tomar su poder del eje. Hoy día, cada máquina individual no sólo tiene su propio motor, pero virtualmente todas las máquinas tienen pequeños motores para proporcionar poder donde se necesite.

Cuando las computadoras llegaron al principio en el control de proceso,

comúnmente existió una máquina central que hacía todo. Hoy día, como en el sistema de máquinas, donde el poder de la computación se necesite, se instala una computadora de cualquier tipo. Se conectan pequeñas computadoras de propósito especial con grandes máquinas en una red de trabajo de área local (LAN)

Los controladores lógicos programables (PLCs) se usan de forma considerable, en conjunto con las buscadores personales (PCs) o computadoras asignadas a funciones especiales.

6.2.2 Unidades de almacenamiento en masa

Además de el almacenaje normal que necesitan las computadoras, existe la necesidad del almacenamiento en masa. Este almacenamiento en masa, se necesita para archivar la operación de la planta, para almacenar datos y hojas de flujo de diseño de la planta, para contener instrucciones, para el entrenamiento del operador en destreza y procedimientos, para mantener instrucciones, procedimientos y fórmulas de cooperación, para estudios de diseño y/o problemas de funcionamiento, etcétera.

La necesidad para almacenar siempre crece; los costos disminuyen conforme la tecnología se vuelve disponible. El almacenamiento más común es el de disco electromecánico, además de los sistemas láser óptico. Se buscan nuevos tipos de almacenaje como el de acceso en paralelo, un ejemplo son los hologramas.

6.2.3 Software

El software de operación del sistema lo proporciona el vendedor. Le permite al usuario implementar fácilmente estrategias de control; pero el software mismo no puede ser (y no debe ser) modificado por el usuario. El vendedor normalmente proporciona un lenguaje de programación que permite al usuario escribir aplicaciones de control avanzadas específicas. Además, los paquetes de Software especiales, como el control de proceso estadístico, "inteligencia artificial", y paquetes de optimización están disponibles. El software de base de datos también está disponible.

6.2.4 Redes de trabajo de campo

Las transmisiones de datos de campo casi siempre están basados en alambres de cobre par-trenzado, usando una señal 4-20mA, los pares de alambres individuales casi siempre son usados para cada variable. El poder para operar las unidades de campo son comúnmente proporcionados a través de este mismo par. Como se puede esperar, esta estructura conduce al uso literal de miles de pares de alambres trenzado. Este alambrado comúnmente cuesta más que el mismo equipo de control. El costo se hace cada vez mayor debido a la necesidad de que sea a prueba de explosión o de forma intrínseca, instalación segura y/o la necesidad de prevenir las regreciones de tierra.

La situación descrita arriba, de forma obvia, ilustra las necesidades de un bus de campo digital estándar para transmitir señales. Algunos bus de vendedores específicos ya están disponibles; pero el estándar ya está disponible desde 1998. Este bus es conocido como bus de campo.

6.5. Estructura del sistema de control [5]

La arquitectura distribuida es actualmente un estructura de uso muy popular para los sistemas de control. El nombre implica que el control de proceso y el manejo de proceso actuales son funciones que, de hecho, están distribuidos a través de la planta. Ellos no están concentrados en un lugar geográfico específico llamado cuarto de control, tampoco están concentrados inherente-mente en una sola pieza de hardware. Desde que existe la necesidad para el manejo y el control distribuido, es lógico distribuir el hardware y la capacidad para completar estas funciones.

Enfocándonos en ésta estructura, abre nuestro pensar en varios tipos de estructuras arquitectónicas de control que son posibles. La figura de 6.1 muestra cinco estructuras diferentes para establecer un sistema que complete el control del proceso. La figura 6-3a muestra un sistema jerárquico que fue la naturaleza de la mayoría de los sistemas de control digital. La figura 6-3b muestra una red de trabajo general, y, cuando se compara con un sistema jerárquico o una organización especializada similar, dicha red

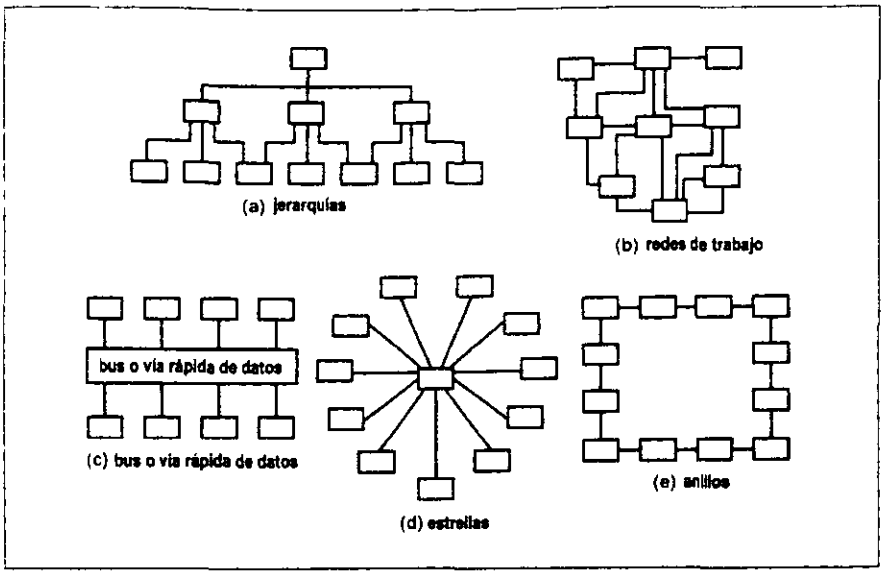


Fig. 6.1. Posibles Organizaciones para un Sistema de Control

de trabajo general permite comunicaciones redundantes y confiables a través de los diferentes componentes. Sin embargo, los elevados costos de dicho sistemas, los hacen poco atractivos para el uso normal.

No haremos comentarios especiales acerca de los sistemas de estrella o de anillo ya que éstos no se usan mucho. El sistema "highway" y bus mostrados en la Fig 6.1c, son los sistemas más populares para implementar un control distribuido. Uno puede argumentar que el sistema está enormemente centralizado con respecto al sistema highway; pero este hace que el sistema sea seguro.

6.6. Sistemas de control distribuido (DCS) [5]

Está claro que el diseño arquitectónico se ha llevado lejos de ser arreglos de ciclo individuales simples para el control de proceso y se ha movido hacia los sistemas distribuidos interconectados con comunicaciones de interconexión elaboradas. La idea del sistemas distribuidos es ilustrado en la figura 6.2. En el control distribuido, los controladores de retroalimentación individual para cada ciclo de proceso son removidos de lugar de cuarto de control y colocados acerca de los sensores de campo y/o los actuadores. Cuando esto se hace, claro, existe la necesidad de ligar las comunicaciones,

por ejemplo, un bus digital y un highway de datos que conectan los controladores individuales con los operadores, computadoras, consolas, y las pantallas. El ciclo de control se vuelve físicamente más pequeño y, por ello, menos vulnerable al ruido o al daño. El enlace de comunicación se puede perder; pero, básicamente, esto representa una pérdida de la inteligencia del operador ya que los controladores de campo individuales continúan operando localmente. Los sistemas de este tipo puede ser implementados tanto con controladores digitales como analógicos y aún con controladores neumáticos; pero hoy día la mayoría del enfoque del hardware digital prevalece y continuará así en el futuro.

Con este enfoque, el operador en el cuarto de control central tiene acceso a todos los datos de los controladores tales como valores de ajuste, mediciones de variables de proceso, niveles de señal de salida del controlador, etcétera. Las pantallas sofisticadas también están disponibles y las funciones de supervisión o manejo -con toda la potencia

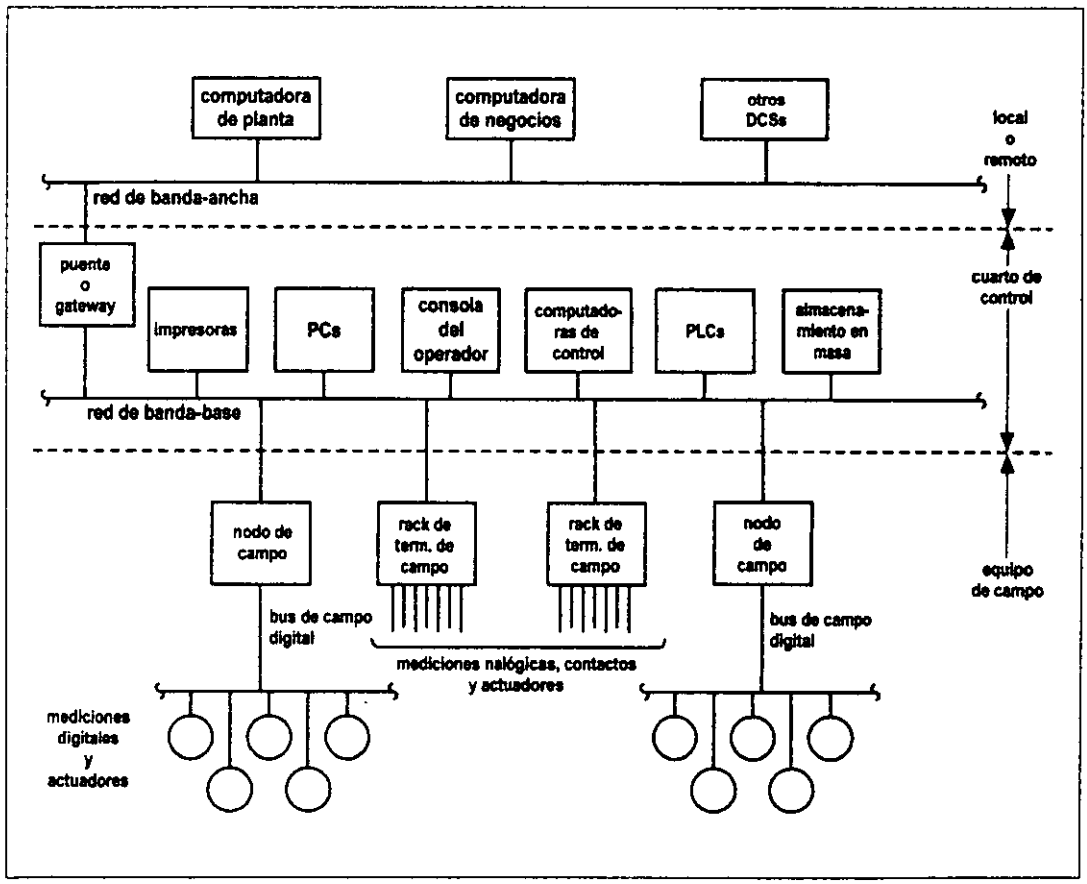


Fig. 6.2. Sistema de Control Distribuido (DCS)

del información de manejo avanzado, optimización, y supervisión- son implementadas fácilmente con el cuarto de control mismo.

Los sistemas de control distribuidos inherentemente tienen nodos de campo y perchas de terminado, con lo que reducen el costo elevado del alambrado de campo y permite la trasmisión de señal más precisas. Desde que muchos sensores ahora contienen microprocesadores, las señales digitales pueden ser transmitidas sin el uso del 4-20 mA dc de trasmisión digital. En dichos sistemas las unidades de campo se necesitan para la conversión de analógico a digital.

En los DCs, el software de E/S del operador, los algoritmos de control básico, y el software de operación del sistema están anexas en los diversos elementos del sistema. El software para manipular, ordenar, y desplegar datos también se proporciona, así como un lenguaje de aplicación de alto nivel para establecer funciones de control avanzado y manejo del proceso.

Recapitulación

Se vio en el capítulo 3 la forma clásica para representar un ciclo de control, sin embargo existen mediciones mas complejas que requieren de otro tipo de control. Y es aquí donde se comentó que tan complejo puede ser un sistema real, pero además se mostraron algunas soluciones, arquitecturas y elementos que pueden aplicarse para mantener el ciclo bajo control, donde se mencionan de forma general algunos sistemas computacionales que son indispensables para el control de un ciclo.

■

CAPITULO VII

Desarrollo de Aplicaciones de Base de Datos en Delphi

Introducción

Las aplicaciones de bases de datos permiten a los usuarios interactuar con la información que está almacenada en bases de datos. Las bases de datos proveen una estructura para la información, y permite ser compartida entre diferentes aplicaciones.

Delphi provee soporte para aplicaciones de bases de datos relacionales. Las bases de datos relacionales organizan la información en tablas, las cuales contienen filas (registros) y columnas (campos). Estas tablas pueden ser manipuladas por simples operaciones conocidas como cálculo relacional.

Cuando se diseña una aplicación de base de datos, debe entenderse como están estructurados los datos. Basados en esa estructura, se puede entonces diseñar una interfaz para desplegar datos al usuario y permitir al usuario a entrar nueva información o modificar datos existentes.

Este capítulo introduce algunas consideraciones comunes para diseñar aplicaciones de bases de datos y las decisiones envueltas en diseñar una interface de usuario.

7.1 Usando bases de datos [10]

Los componentes en la página Data Access -Acceso de datos- en la paleta Component le permite a la aplicación leer desde y escribir a bases de datos. Estos componentes usan la Máquina de Bases de Datos de Borland (Borland Database Engine - BDE) para accesar información de las bases de datos la cual hace disponible para los controles de datos en interfase del usuario.

Dependiendo de la versión de Delphi, la BDE incluye controladores para diferentes tipos de bases de datos. Mientras todos estos tipos de bases de datos contienen tablas las cuales almacenan información, diferentes tipos soportan características adicionales como:

- Seguridad de base de datos.
- Transacciones
- Diccionario de datos
- Integridad referencial, almacenaje de procedimientos, y lanzadores (triggers)

7.1.1 Tipos de Bases de datos

Pueden conectarse diferentes tipos de bases de datos, dependiendo que tipo de controladores tenga instalados con la "BDE". Todas las versiones de Delphi incluyen controladores para bases de datos locales. En suma, si se tiene la versión "Client/Server" o "Enterprise", pueden usarse los controladores instalados con SQL Links -ligas de SQL- para comunicarse con servidores de bases de datos remotas.

Elegir que tipo de base de datos a usar depende de varios factores. Los del usuario datos pueden estar ya almacenados en una base de datos existente. Si se están creando las tablas de información que la aplicación usa, pueden querer considerarse las siguientes preguntas:

- ¿Cuántos datos contendrán las tablas?
- ¿Cuántos usuarios estarán compartiendo estas tablas?
- ¿Cuántos tipos de desempeños (velocidad) se requiere de la base de datos?

7.1.1.1 Base de datos locales

Las bases de datos locales residen en el disco local o en un área de red local. Ellas tienen las APIs propietarios para acceder los datos. Comúnmente, éstas están dedicadas para un solo sistema. Cuando están compartidas por varios usuarios, usan un mecanismo de aseguramiento basado en archivo. Debido a esto, ellas a veces son llamadas bases de datos basados en archivo.

Las bases de datos locales pueden ser más rápidas que los servidores de bases de datos, ya que ellas residen casi siempre en el mismo sistema como una

aplicación de base de datos.

Debido a que ellas están basadas en archivo, las bases de datos locales son más limitadas que los servidores de base de datos en la cantidad de datos que éstas pueden almacenar. Por eso, si en la decisión se usa una base de datos local, debe considerarse cuántos datos, las tablas están esperando mantener.

Las aplicaciones que usan bases de datos locales son llamadas aplicaciones grada-simple ya que la aplicación y la base de datos comparten un solo sistema de archivo. Ejemplos de bases de datos locales incluyen Paradox, dBASE, FoxPro, y Acces.

7.1.1.2 Servidores de bases de datos remotos

Los servidores de bases de datos remotos usualmente residen en una máquina remota. Ellos usan Lenguaje Estructurado de Pregunta ("Structured Query Language - SQL") para permitir a los clientes acceder a los datos. Debido a esto, ellos son a veces llamadas servidores SQL. (Otro nombre es sistema de Manejo de Bases de Datos Remoto - "Remote DataBase Management System- RDBMS".) Además de los comandos comunes que hacen SQL, la mayoría de los servidores de base de datos remotas soportan un "dialecto" único de SQL.

Los servidores de Base de datos remoto están diseñados para ser accesados por muchos usuarios al mismo tiempo. En vez de el sistema cerrado basado en archivo, como aquellos empleados por bases de datos locales, ellos proveen más soporte multi-usuario sofisticado, basado en transacciones.

Los servidores de base de datos remotos contienen más datos que las bases de datos locales. A veces, los datos de un servidor de base de datos remoto no siempre residen en una sola máquina, pero son distribuidos en otro distintos servidores.

Las aplicaciones que usan servidores de base de datos remoto son llamados aplicaciones de dos-gradas (two-tiered application) o aplicaciones multi-gradas (multi-tiered-application) ya que las aplicaciones y las bases de datos operan en sistemas independientes (o gradas). Ejemplos de servidores SQL incluyen a Interbase, Oracle, Sybase, Informix, Microsoft SQL server, DB2.

Nota Debe tenerse la versión "Client/Server" o "Enterprise" para escribir aplicaciones que usan servidores de base de datos remotos.

7.1.2 Seguridad de base de datos

Las bases de datos a menudo contienen información sensible. Diferentes bases de datos proveen esquemas de seguridad para proteger esa información. Algunas bases de datos, como "Paradox" y "dBASE", solo proveen seguridad en la tabla o nivel de campo. Cuando los usuarios tratan de accesar las tablas protegidas, requieren proveer un "password". Una vez que los usuarios han sido autenticados, ellos pueden ver solo esos campos (columnas) para los cuales ellos tienen permiso.

La mayoría de los servidores "SQL" requieren un "password" y un nombre de usuario para usar todo el servidor de base de datos. Una vez que el usuario a entrado en la base de datos, ese nombre de usuario y "password" determinan que tablas pueden ser usadas.

Cuando se diseña una aplicación de base de datos, debe considerarse que tipo de autenticación es requerida por el servidor de base de datos. Si no se quiere que los usuarios tengan que proveer un "password", debe usarse una base de datos que no requiera de uno o se debe proveer el "password" y el nombre del usuario al servidor programablemente. Cuando se provee el "password" programablemente, debe tenerse cuidado que la seguridad no podrá ser violada al leer el "password" desde la aplicación.

Si se requiere que el usuario proporcione el "password", debe considerarse que el

"password" es requerido. Si no estás usando una base de datos local pero intentas escalar a un servidor "SQL" más grande posteriormente, se requerirá la invocación del "password" antes de accesar la tabla, aunque no se requiera hasta entonces.

Si la aplicación requiere múltiples "passwords" debido a que deben entrar a varios sistemas protegidos o bases de datos, puede hacerse que tus usuarios proporcionen un solo "password" maestro el cual es usado para accesar una tabla que requiera de "password" por los sistemas protegidos. La aplicación entonces proporciona "password" programablemente, sin requerir que el usuario proporcione múltiples "passwords".

En aplicaciones multi-gradas, podría requerirse de usar diferentes modelos de seguridad completo. Puede usarse "CORBA" o "MTS" para controlar el acceso a gradas intermedias y permitir a las gradas intermedias manipular todos los detalles de entrada a servidores de base de datos.

7.1.3. Transacciones

Una transacción es un grupo de acciones que deben ser llevadas a cabo todas con éxito en uno o más tablas en una base de datos antes de que ellas sean entregadas (sean permanentes). Si cualquiera de las acciones en el grupo falla, entonces todas las acciones son devueltas (deshacer).

Las transacciones protegen contra fallas de hardware que ocurren en la mitad del comando de base de datos o un conjunto de comandos. Ellas también forman la base del control de la concurrencia multi-usuario en servidores "SQL". Cuando cada usuario interactúa con la base de datos solo a través de transacciones, un comando de usuario no puede romper la unidad de transacción de otro usuario. En cambio, el servidor "SQL" pone en un horario las transacciones que vienen entrando, las cuales o son un éxito total o una falla total.

Aunque el soporte de la transacción no es parte de la mayoría de las bases de

datos, los controladores "BDE" proveen un soporte limitado de transacciones para algunas de esas bases de datos. Para los servidores "SQL" y las bases de datos compiladas-ODBC, el soporte de transacción de la base de datos es provista por la misma base de datos. En aplicaciones multi-gradas, puedes crear transacciones que incluyen acciones otras coma las operaciones de base de datos o el giro (span) de múltiples bases de datos.

7.1.4 El Diccionario de Datos

No importa que tipo de base de datos se use, la aplicación tiene acceso al Diccionario de Datos. El Diccionario de Datos provee una área de almacenaje que se puede personalizar, independiente de las aplicaciones, donde se pueda crear conjuntos de atributos de capos extendidos que describen el contenido y apariencia de los datos.

Por ejemplo, si se desarrollan aplicaciones financieras frecuentemente, se puede crear un número de conjuntos de atributos de campo especializado que describa diferentes formatos de desplegado para el dinero. Cuando son creados conjuntos de datos (datasets) para una aplicación en el momento de diseño, en vez de usar el "Object Inspector" para establecer los campos de dinero en cada database a mano, se puede asociar esos campos con un conjunto de atributos de campo extendido en el diccionario de datos. Usando el diccionario de datos se asegura una apariencia de datos consistente dentro y a través de la aplicación que estas creando.

En un ambiente "Client/Server", el Diccionario de Datos puede residir en un servidor remoto para la repartición adicional de la información.

Una interface de programación para el Diccionario de Datos está disponible en la unidad drinft (localizado en la librería lib).

7.1.5 Integridad referencial, almacenaje de procedimientos, y lanzadores (triggers)

Todas las bases de datos relacionales tienen ciertas características en común que permiten a las aplicaciones almacenar y manipular datos. Además, las bases de datos

casi siempre proveen otras características específicas de base de datos que pueden ser útiles para asegurar la relación consistente entre tablas en una base de datos. Esto incluye:

- **Integridad Referencial.** La integridad referencial provee un mecanismo que previene a las relaciones "master/detail" entre tablas de ser rotas. Cuando el usuario trata de borrar un campo en una tabla maestra (master table) la que produciría un archivo de registros detallados solitarios, la integridad referencial previene el borrado o borrado automático de los registros detallados solitarios.

- **Procedimientos Almacenados.** Los procedimientos almacenados son conjuntos de sentencias que son nombradas y almacenadas en un servidor "SQL". Los procedimientos almacenados usualmente desarrollan tareas relacionadas con el servidor, y regresa conjuntos de registros (datasets).

- **Gatillos (Triggers).** Los gatillos son un conjunto de sentencias "SQL" que son ejecutadas automáticamente en respuesta a un comando en particular.

7.2 Diseñando la interface usuario [11]

La página "Data Controls" de la paleta "Component" provee un conjunto de controles de datos que representan un dato de los campos en un registro de base de datos, y permiten a los usuarios editar los datos y pegar los cambios de regreso a la base de datos. Usando controles de datos, puede construirse una interface usuario (UI) de la aplicación de base de datos para que la información sea visible y accesible a los usuarios

Los controles de datos obtienen los datos de y los mandan a un componente de datos fuente, *TDataSource*. Un componente de datos fuente actúa como un conducto entre la interface y el componente dataset el cual representa un conjunto de información desde las tablas en una base de datos. Varios controles de datos en una forma pueden compartir un fuente simple de datos, en tal caso el desplegado en cada control está

sincronizado de tal forma que los usuarios se desplazan a través de los registros, el valor correspondiente en los campos para los registros actuales se despliega en cada control. Una aplicación de componentes de datos fuentes usualmente residen en un módulo de datos, separado de los controles de datos en las formas.

Los controles de datos que se agreguen a la interface usuario depende en que tipos de datos se están desplegando (texto plano, texto con formato, gráficas, elementos de multimedia, etc.). Además, elección de controles es determinado por como quiere organizarse la información y como (o si) se quiera permitir a los usuarios navegar a través de los registros de los "datasets" y agregar o editar datos.

Las secciones siguientes introducen los componentes que pueden usarse para varios tipos de interface usuario.

7.2.1 Desplegando un solo registro

En muchas aplicaciones, solo se quiere proveer información acerca de un solo registro de datos a la vez. Por ejemplo, una aplicación de una orden de entrada puede desplegar solo la información acerca de una sola orden sin indicar que otras órdenes actualmente se están entrando. Esta información probablemente viene de un solo registro en una dataset de ordenes.

Las aplicaciones que despliegan un solo registro, son usualmente sencillas de leer y entender, ya que toda la información de la base de datos es acerca de la misma cosa (en el caso previo, la misma orden). Los controles de datos en estas interfaces de usuario representan un solo campo del registro de la base de datos. La página "Data Controls" de la paleta "Component" provee una gran variedad de controles que representan diferentes tipos de campos.

7.2.2 Desplegando múltiples de registros

Algunas veces quieres desplegar muchos registros en la misma forma. Por ejemplo,

un aplicación de facturación puede mostrar todos las órdenes hechas por un solo cliente de la tienda en la misma forma.

Para desplegar múltiples registros, usa el control de reja. Los controles de reja proveen un multi-campo, una vista multi-registro de datos que pueden hacer tu aplicación de interface de usuario más comprometido y efectivo.

Se querrá diseñar una interface que despliegue ambos campos desde un solo registro y rejas que representan múltiples registros. Hay dos modelos que combinan esos dos acercamientos:

- **Formas maestro-detallado:** Puede representarse la información de una tabla maestra y una tabla detallada al incluir ambos controles que despliegan un solo campo y controles de reja. Por ejemplo, se puede desplegar información acerca de un solo cliente de la tienda con una reja detallada que despliega las ordenes para ese cliente de la tienda.

- **Formas perfora-abajo:** En una que despliega múltiples registros, puede incluirse controles de campo simples que despliegan una información detallada solo del registro actual. Este acercamiento es particularmente útil cunado los registros incluyen memos largos o información gráfica. Como el usuario se desplaza a través de los registros de la 'reja', el 'memo' o las 'gráficas' se actualizan para representar los valores del registro actual. Establecer esto es muy fácil. La sincronización entre dos de los desplegados es automático si el control de la reja y el memo o la gráfica comparten la misma fuente de datos.

Nota No es generalmente una buena idea combinar esos dos enfoques en una sola forma. Ya que el resultado a veces es efectivo, es usualmente confuso para los usuarios entender las relaciones de datos.

7.2.3 Analizando datos

Las aplicaciones de base de datos no representan la información de la base de datos directamente del usuario. En cambio, ellas analizan y resumen información de las bases de datos para que los usuarios puedan dibujar las conclusiones de los datos.

El componente "TDBChart" en la página de "Data Controls" de la paleta "Component" permite presentar la información de las bases de datos en un formato gráfico que concede a los usuarios graficar rápidamente la información de la base de datos importada.

Además, las versiones "Client/Server" incluye una página de "Decision Cube" en la paleta "Component". Este contiene seis componentes que te permiten desarrollar análisis de datos y tabulaciones en datos cuando se construye una aplicación de soporte de decisión. Para más información acerca del uso de componentes "Decision Cube"[1].

7.2.4 Eligiendo que tipos de datos mostrar

Casi siempre, los datos que se quiere que aparezcan en la aplicación de base de datos no corresponden exactamente a los datos en una bases de datos simple. Querrá usarse solo el subconjunto de campos o un subconjunto de registros en una tabla. Querrá combinarse la información de más de una tabla en una vista simple.

Los datos disponibles para la aplicación de base de datos es controlada solo por elección propia en el componente de dataset. Los "datasets" resumen las propiedades y métodos de la tabla de base de datos, para que no se tenga que hacer más alteraciones dependiendo en si el dato está almacenado en la tabla de base de datos o deriva de uno o más tablas en la base de datos. Para mayor información en los métodos y propiedades comunes de bases de datos[2].

---

[1] Vea el capítulo 14 del manual de usuario de Delphi.

[2] Vea el capítulo 18, "Entendiendo los datasets" del manual de usuario de Delphi.

La aplicación puede contener más de un dataset. Cada "dataset" representa una tabla lógica. Al usar "datasets", la lógica de tu aplicación es almacenada temporalmente de ser restructurada por las tablas físicas en tus bases de datos. Puede que se necesite alterarse el tipo de componente dataset, o la forma que especifica los datos que contiene, pero el resto de la interface usuario puede continuar el trabajo sin alteración.

Puede usarse cualquiera de los siguientes tipos de dataset:

- **Componentes "Table" -Tabla-:** Las tablas corresponden directamente tablas fundamentales en la base de datos. Puede ajustarse que campos aparecen (incluyendo campos agregados de visualización y campos calculados) al usar componentes de campo persistentes. Pueden limitarse los registros que parecen usar rangos o filtros.

- **Componentes "query" -consulta-:** Las consultas proveen la mayoría de los mecanismos para especificar que aparece en un dataset. Pueden combinarse los datos de múltiples tablas usando uniones, y limita los campos y registros que aparecen basados en cualquier criterio que se exprese en "SQL".

- **Procedimientos almacenados:** Los procedimientos almacenados son conjuntos de sentencias "SQL" que son nombradas o almacenadas en un servidor "SQL". Si el servidor de base de datos define un procedimiento remoto que regresa un dataset que se quiere, puede usarse un componente "procedure" almacenado.

- **"Datasets" cliente:** Los "datasets" cliente almacenan en un lugar temporal los registros del dataset lógico en memoria. Debido a esto, ellos solo pueden mantener un número limitado de registros. Se puede construir pequeñas aplicaciones usando "datasets" cliente ya que ellos no requieren de la "Borland Database Engine", solo "DBClient.DLL". Los "datasets" cliente se popularizan con los datos en una de dos formas: desde una aplicación servidor o desde un dato archivo-débil almacenado

en disco. Cuando se usa un dataset cliente para representar datos archivo-débil, debe crearse la tabla fundamental en de forma programada.

- **"Datasets" anidados:** Los "datasets" anidados representan los registros en un conjunto detallado anidado de Oracle8. Delphi no te permite crear tablas Oracle8 con campos "datasets" anidados, pero se pueden editar y desplegar datos desde los campos dataset existentes usando "datasets" anidados. El dataset anidado obtiene sus datos desde un componente de campo dataset en un dataset el cual contiene datos Oracle8.

- **"Datasets" personalizados:** Pueden crearse descendientes propios personalizados de *TDataSet* para representar un cuerpo de datos que fabriques o acceses en el código que se escriba. Escribiendo "datasets" personalizados permite la flexibilidad de manejar los datos usando cualquier método que se elija, mientras se permita usar los controles de datos VCL para construir interfaces de usuario.

7.2.5 Escribiendo reportes

Si se quiere permitir a los usuarios imprimir información de base de datos desde los dataset en la aplicación, pueden usarse los componentes de reporte en la página "QReport" de la paleta "Component". Usando esos componentes, pueden construirse visualmente reportes agrupados para representar y resumir la información de las tablas de base de datos. Pueden agregarse resúmenes para agrupar encabezados y pies de página para analizar los datos basados en criterios agrupados.

Para empezar un reporte para la aplicación, se elige el icono "QuickReport" desde la caja de diálogo de "New Items". Elíjase "File|New" desde el menú principal, y diríjase a la página etiquetada "Bussines". De doble-clic al icono "QuickReport" Wizard para lanzar la ayuda.

Recapitulación

En este capítulo se abordaron los temas, puntos, y consideraciones generales para poder realizar una interfaz con la base de datos según el usuario lo solicite, además de las características que debe cumplir la base de datos para que el usuario tenga una vista sobre los datos con los que esta trabajando. Con este conocimiento como trabajar con bases de datos y como crear interfaces, en el próximo capítulo se abordará el tema de la distribución de interfaces dentro de Intranet.

■

CAPITULO VIII

Computación Distribuida

Introducción

El surgimiento de las estaciones de trabajo conectadas en red y la caída de los mainframes centralizados se debe al cambio dramático de la tecnología de la información en las últimas dos décadas. Este empuje ha puesto más vigor de proceso en las manos de usuarios finales y en los recursos de hardware distribuido a través de la empresa. No existirá mas dominio de los pisos elevados y centros de datos, el poder del proceso ahora reside en las computadoras personales, en los servidores de grupo de trabajo, y en las minicomputadoras. Este empuje se vio envuelto en hardware; el desafío actual es desarrollar la infraestructura del software para hacer uso de estos recursos distribuidos.

Conforme las redes se han vuelto más prevalecientes de los recursos de computación, el concepto de el procesamiento relacionado distribuido entre recursos múltiples se ha vuelto de forma creciente, viable y deseable. Al paso de los años, varios métodos han evolucionado para permitir esta distribución, alineando la distribución de datos sencillos a sistemas avanzados que soportan una multitud de servicios. Este capítulo presenta un repaso de los significados usados para habilitar la distribución del trabajo de computo, cubriendo conceptos centrales e implementaciones populares de esos conceptos. El objetivo es educar a la audiencia a las tecnologías disponibles además de sus fortalezas y debilidades.

8.1 Fundamento Común [12]

8.1.1 Comunicación de la Red

El fundamento de todas las arquitecturas de computación distribuida es la noción de la comunicación entre computadoras. Aunque básico, esto muestra que tan cerca, de forma conceptual, algunas arquitecturas de distribución común están fundamentados en sus propias instalaciones de comunicación fundamentales. La combinación de hardware y el software de nivel de sistema que permite comunicar a las computadoras, se refiere comúnmente a la capa de transporte. Cuando varias computadoras están conectados a otra a través de un misma capa, ellas se pueden considerar una red de trabajo de

computadoras.

Parecido a la pieza de información que puede ser envuelto, tratado y enviado por el servicio postal, las redes operan generalmente en paquetes, los cuales son análogos al paquete que se envía a través del correo. Como un paquete de correo, el paquete de red tiene direcciones 'proveniente (from)' y 'hacia (to)' y contiene alguna información, como un mensaje. Al igual que un mensaje de correo, el receptor puede o no elegir o reconocer la aceptación del paquete.

Si el menajo de correo o red excede ciertos límites de tamaño, puede que necesite romperse en partes separadas y ser reconstruidos al llegar a su destino. Sin embargo, estos paquetes separados de forma física pueden ser tratados solo como paquetes lógicos. La capa de transporte, la semántica de direccionamiento, la secuenciación de paquetes, el formato de datos y un host de otros componentes definidos hacen un protocolo de comunicación. Estos protocolos predefinidos son lo que permiten a los sistemas de computadoras interpretar adecuadamente los paquetes recibidos provenientes de otros sistemas.

8.1.2 Transmisión Síncrona y Asíncrona

Parecido un paquete de correo, el interés del remitente varía en la recepción subsecuente del paquete y las acciones tomadas en respuesta al mismo. Hay casos donde el remitente no está interesado en cuando, o si tal vez, el paquete ha llegado a su destino. Existen otros casos donde el remitente quiere confirmar que el paquete ha llegado, pero no se necesita dicha información para continuar su tarea. Existen también casos donde el remitente no puede continuar a menos que este reciba una respuesta del destinatario.

Los modo de transmisión síncrono de operación es aquel donde: el remitente necesita una respuesta que venga del destinatario antes que pueda continuar. Las formas de operación donde el remitente no requiere una respuesta proveniente del destinatario,

por lo menos no antes de que puede continuar, se considera asíncrono. Esta distinción es generalmente uno de los factores que determinan la conveniencia del protocolo de comunicación dado a una tarea dada.

8.1.3 Clientes, Servidores y Par (Peer)

Los términos *Cliente/Servidor* y *Par-a-Par (Peer-to-Peer)* se han asociado a cualidades específicas de la computación distribuida. De hecho, los clientes, los servidores y pares son solo roles ejercidos por los participantes en un protocolo de comunicación. Estos roles pueden ser cambiados constantemente dentro de una sesión de comunicación. Notese que esos participantes son en estos momentos hilos de ejecución, los cuales pueden existir en el mismo sistema o aún dentro del mismo proceso como el hilo de ejecución con el que se están comunicando.

Cuando un hilo de ejecución abre un canal de comunicación y espera a que otra hilo lo contacte, este se considera generalmente un *servidor*. El hilo que inicia la comunicación al contactar el servidor es considerado generalmente un *cliente*. *Par (Peer)* es un término general usado para referirse a un hilo que es capaz de actuar tanto como cliente como servidor.

8.1.4 APIs - Interfaces de Programación de Aplicación.

Las instalaciones de comunicaciones de base son proporcionadas generalmente por los APIs de solicitud del sistema operativo (OS) y de la red. Estos grupos son funciones llamadas por un programa para lograr la transmisión y la recepción actuales de bytes de datos entre los sistemas. En general, esos componentes de nivel-bajo proporcionan una abstracción limitada de la sesión subyacente de la comunicación, dejando a los comunicantes proporcionar los servicios lógicos como el direccionamiento y la conversión de datos.

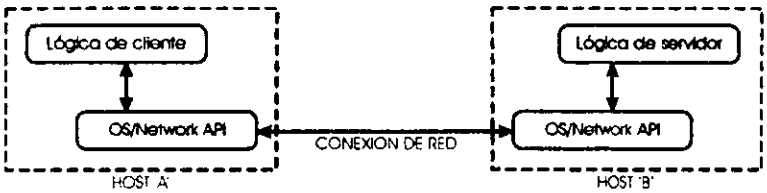


Fig. 8.1 Comunicación de red de trabajo Continua.

8.1.5 Interfaces de Terminal

La forma más vieja de computación distribuida no es reconocida generalmente como tal - entrar en un sistema host de una terminal tonta o de un emulador de terminal que se ejecuta en una estación de trabajo. Tal vez no de la manera elegante, este método ha probado ser muy efectivo.

Existen un número de protocolos para este tipo de comunicación, entre el "Telnet", "rsh" y el "rexec". El concepto e implementación son simples; el cliente actúa muy parecido a una terminal conectada directamente, pero con algunas facilidades adicionales permitiendole comunicarse a través de una conexión remota. Cada vez que una tecla es presionada, el cliente manda un paquete que contiene un código que identifica la llave al servidor. El servidor, en turno, manda de regreso paquetes que contienen datos para que se desplieguen por el cliente. Mientras que las interfaces están limitadas generalmente a texto, las aplicaciones basadas-en-servidor son capaces de usar llaves de color y extendidas para mejorar la funcionalidad de la interfaz del cliente.

Entre los beneficios de las interfaces de terminal está el hecho que en muchos casos ellos no requieren que la aplicación sea escrita usando un API de comunicación, permitiendo a los programas, que fueron escritos sin considerar la distribución, que sean usados de forma remota sin modificarlos.

8.1.6 Mensajes

En la evolución de la computación distribuida vino el concepto del mensaje, un paquete de datos que es etiquetado con la información que contiene. Esto permite a una capa de procesado intermedio en el servidor enviar por una ruta (to rout) el mensaje, o el dato que contiene, al receptor apropiado para él.

Los sistemas de mensaje pueden operar de forma natural en una arquitectura asíncrona. Ya que la comunicación basada-en-mensaje está bien adaptada al encaminamiento (routing) intermedio, estas características pueden ser combinadas para proporcionar un nivel de abstracción a la estructura misma de la comunicación. Los mensajes pueden ser depositados en una lista por el servidor/encaminador, del que ellos fueron recuperados y desempeñados por uno o más procesadores lógicos.

Estos procesadores no pueden responder a todos los mensajes o pueden responder directamente al cliente. Sin embargo, para mantener la abstracción, ellos pueden mandar un mensaje de regreso al servidor, a través de otra lista, para que sea encaminado de regreso al cliente. Sin embargo, para mantener la abstracción, ellos pueden mandar un mensaje de regreso al servidor, a través de otra lista, para que puedan ser encaminado de regreso al cliente.

Las arquitecturas basadas-en-mensaje también son capaces de operar de forma síncrona. Generalmente en este modo, el servidor/encaminador pasa el mensaje al procesador, el cual pasa una respuesta de regreso al servidor a ser regresado al cliente. Otro modo híbrido está disponible, sin embargo, en el cual el servidor se comporta de forma asíncrona como se describió arriba y solo el cliente actúa de forma síncrona. Esta combinación de conductas permiten al servidor ganar la eficiencia de la operación asíncrona mientras que el cliente se beneficia de la simplicidad de proceder y la seguridad del procesado asíncrono.

Esta arquitectura básica de la comunicación cliente con un servidor el cual

despacha mensajes basados en su contenido serán vistos para que sea la base de muchos modelos siguientes de distribución.

8.1.7 RPC - Llamada de Procedimiento Remoto

El concepto detrás de las RPCs es simple - para hacer que aparente ser una llamada de procedimiento normal dentro de un proceso y que su ejecución la lleve a cabo dentro de otros procesos, posiblemente en un sistema remoto. Varias implementaciones de protocolos RPC se han desarrollado con la meta en común de reducir la complejidad de comunicación entre procesos a través de ocultar la implementación.

El concepto de la base de los mecanismos RPC es el de serializar datos de la llamada de la función en una corriente secuencial y reconstruirlos al final de la recepción de la conexión. Esta conducta toma lugar de forma síncrona, reflejando la semántica de la programación de procedimientos tradicional. El proceso cliente RPC hace una llamada a lo que aparenta ser una función estándar, conocida como talón (stub). Sin embargo, en vez de ejecutarse de forma local, los parámetros pasados a la función son empaquetados y transmitidos a un ambiente de ejecución remota, donde ellos son pasados a una implementación real de la función. En la implementación de la función de la ejecución, su valor de regreso es serializado y devuelto al stub cliente, el cual lo regresa al que llamo.

8.2 Cliente/Servidor [12]

Como se mencionó arriba, los términos *cliente* y *servidor* raramente se refieren al rol desempeñado por los participantes en una sesión de comunicación. Sin embargo, el término *Cliente/Servidor* se ha vuelto común en su uso describiendo niveles altos, aunque similar en forma conceptual, de arquitectura. La interpretación común de los términos denota un sistema donde el procesado significante se hace en el *cliente*, el cual también somete operaciones al *servidor* para la ejecución. En este tipo de arquitectura la operación asíncrona se asume generalmente, en donde el cliente espera la confirmación que la operación ha estado realizando antes de proceder.

8.3 Middleware [12]

A diferencia de las arquitecturas cliente/servidor donde el cliente se identifica y se comunica directamente con el servidor, el concepto de 'middleware' asume una capa funcional entre el cliente y el servidor. Esta capa puede proporcionar servicios a los comunicantes tales como resolución de lugar y el alias, semántica de autenticación y transacción. Otras conductas asociadas con el middleware incluyen sincronización de tiempo y transacción entre formatos de datos.

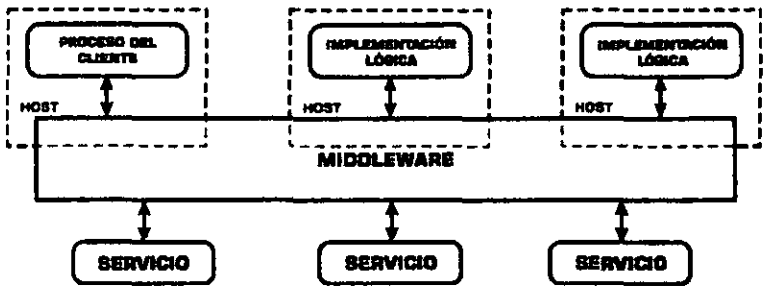


Fig 8.2 Middleware

Esta capa adicional permite a los clientes para interactuar con una abstracción genérica de un servidor en lugar con un host específico y/o proceso. Varios servicios se proporcionan a través de capas separadas, empañando la distinción entre los servicios proporcionado por el middleware y funcionalidad añadido por el servidor. Estas abstracciones permiten a las aplicaciones ser desarrollados por un API estandarizado sin el conocimiento del lugar o implementación de funcionalidades externas. Esta ocultación de implementación es uno de los pros del modelo middleware, aunque hace difícil para el cliente determinar que rendimiento puede esperar de cualquier implementación lógica dada.

8.3.1 DCE - Ambiente de Computación Distribuida

"OSF[1] DCE formaliza muchos de los conceptos descritos donde con un grupo de especificaciones relacionadas" [13]. La especificación DCE RPC está entre las mas

[1] La Fundación de Software Abierto (OSF - Open Software Foundation) se ha cambiado el nombre a The Open Group (OG), sin embargo DCE es una marca registrada, y esta como "OSF DCE"

implementadas ampliamente en la industria, proporcionando conducta consistente a través de ambientes de ejecución heterogéneas. La arquitectura DCE también define la capa, el tiempo, autenticación y seguridad, directorio y nombrado de servicios.

Debido a que DCE está soportado por un consorcio industrial que comprende muchos de los grandes vendedores de sistemas operativos, sus estándares gozan de soporte muy difundido a través de las grandes plataformas de computación. La base de la funcionalidad de DCE está incluido en casi todas las variantes de UNIX, y como los sistemas operativos para PC se han vuelto más avanzadas, el soporte de servicio de base DCE se han vuelto más comunes en ellos. Estos estándares están basados en métodos de programación de procedimiento en el lenguaje de programación 'C', sin embargo, limitando su aplicabilidad a distribuciones multilenguajes y orientado a objetos.

8.3.2 Mensajería Formal

En su superficie, las arquitecturas de mensajería formal tales como las MQSeries de IBM y la MSMQ de Microsoft aparentan ser mucho mas a una estructura de mensaje enlistado descrito anteriormente. Debajo de esta superficie, sin embargo, difieren de forma enorme en su implementación.

Para proporcionar la entrega confiable de mensajes asíncronos, un modelo "almacena y transmite" es usado en donde un mensaje a ser enviado por un proceso pasa de forma síncrona a una capa middleware que almacena el mensaje, y cualquier información de direccionamiento que pueda tener, a un mecanismo de almacenaje persistente antes de regresar el control al proceso de envío. Una vez que el mensaje ha sido almacenado en esta manera, el middleware puede usar una variedad de métodos para tratar de obtener el mensaje a su recibidor destinado, mientras el remitente continua su proceso.

La realidad de la arquitectura viene del concepto que el poseedor del mensaje actual del mensaje no destruya su copia persistente del mensaje a menos que este haya

recibido la confirmación del recibidor subsecuente que el mensaje ha sido almacenado de forma segura por él. Debido a que cada liga en la cadena de comunicación almacena el mensaje hasta que esta conozca que ha sido enviado exitosamente, el remitente, original, puede proceder con su procesado seguro que su mensaje pasará a su destino. Debido a la naturaleza asíncrona de esta arquitectura, el remitente debe solicitar una confirmación de recibido del mensaje (o una confirmación debe haber una acción específica al recibir el mensaje particular) si este necesita conocer cuando llegó el mensaje u otros detalles de su manejo.

8.4 Objetos Distribuidos

Las arquitecturas de distribución de objetos se construyen en el concepto de middleware al encapsular los datos en objetos dentro de interfaces funcionales. Parecido a los APIs de procedimiento bien-diseñado, los detalles de la implementación están ocultos del usuario del objeto. Sin embargo, a diferencia de los APIs, las arquitecturas de objeto limitan el acceso para invocar los métodos definidos por el objeto. Además, los métodos son invocados en los objetos directamente, vía referencias a los objetos, eliminando la necesidad de instancias locales de los objetos.

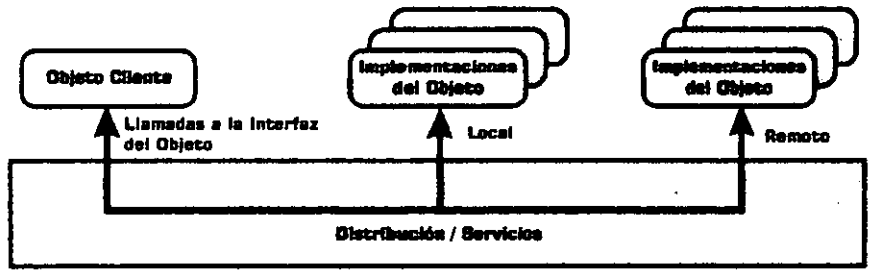


Fig. 8.3 Arquitectura de Objetos Distribuidos.

Este ocultamiento casi-completo de la implementación permite a las arquitecturas de objetos distribuidos soportar lugar, plataforma y transparencia de lenguaje de programación. Sin embargo, dicha transparencia no esta fuera de sus costos, la cual a incitado a los programadores de algunas arquitecturas de objetos distribuidos a renunciar una cierta neutralidad a cambio de mejoras percibidas en el funcionamiento y aplicabilidad a las tareas específicas y/o facilidad de empleo.

8.4.1 JAVA RMI - Invocación de Método Remoto

Java de Sun, aunque tiene poco en la industria de la computación, ha ganado una gran aceptación debido a su diseño neutralidad de plataforma, neutralidad y orientado a objetos. Java fue diseñado desde sus inicios como un ambiente de ejecución completa, en vez de como cualquier otro lenguaje de programación, por eso este es capaz de proporcionar una interfaz consistente y abstracta sin hacer caso de la plataforma fundamental.

Esta independencia de plataforma se consigue a través del uso de la Máquina Virtual de Java (JVM) que emula una plataforma de computación. El JVM se proporciona por cada combinación actual de hardware y sistema operativo en el cual Java va a correr. Debido a que todos los programas aparentan, en el nivel de aplicación, estar corriendo en la misma plataforma de computo, la comunicación entre las aplicaciones Java se hace más fácil.

"Java RMI proporciona una arquitectura de leguaje-específico que permite a las aplicaciones distribuidas Java-a-Java ser construidos fácilmente. La ventaja principal al usar Java RMI cuando se diseña un sistema distribuido Java puro es que el modelo de objetos Java puede tomar ventaja cada vez que sea posible. Claro, esto impide el uso de Java RMI en un ambiente multilenguaje. La independencia de plataforma inherente de Java, sin embargo, aún permite la distribución en ambientes heterogéneos." [14]

8.4.2 DCOM - Modelo de Objeto de Componente Distribuido [15], [16]

El protocolo fundamental de distribución de objetos de Microsoft DCOM, una extensión de la arquitectura de integración COM de Microsoft, lo que permite la integración entre los objetos, ejecutandose en hosts separados en la red.

COM empieza como una forma que permite a los clientes ligar a implementaciones de objetos de forma dinámica; p.e. en el tiempo-de-ejecución, incorporarlos a espacio de dirección simple. Las implementaciones fueron empacadas en Librerías de Enlace

Dinámico (DLLs). COM es esencialmente un esquema de integración, adoptando la estructura de tablas de función virtual C++ para la compatibilidad binaria. Estas tablas de función virtual, comúnmente conocidas como 'vtables', consiste de una tabla de dirección de función (en términos de lenguaje C) o el equivalente. Las interfaces COM son presentadas a los clientes como punteros hacia las vtables, así, de ese modo ocultando los detalles de la implementación. Esto hace a los binarios COM independientemente reemplazables, conforme ellos implementen las misma interfaces a sus predecesores.

Al tomar el camino de la menor resistencia y adoptando el modelo vtable de C++ de una integración binaria, COM realizó los binarios reemplazables y la eficiencia de proceso de una invocación de métodos de proceso, equivalente a la llamada de la función virtual de C++. Estos beneficios vienen naturalmente con algún costo. Debido a que una interfaz es presentada como puntero a un solo vtable, las interfaces no pueden ser definidas usando múltiples herencias. Esto puede implicar múltiples vtables y con ellos múltiples punteros por interfaz. Además, debido a que no hay un servicio intermediario para despachar llamadas de funciones, cualquier lenguaje de programación además de C++ debe ir entre contorciones para trabajar con COM.

Para poder direccionar la creciente necesidad de la distribución de los objetos a través de múltiples hosts (p.e. múltiples direcciones físicas), Microsoft desarrollo DCOM como una extensión de COM. Como una extensión en vez de una arquitectura separada, DCOM insertó un interfaz stub entre la llamada de la aplicación y la implementación actual de la interfaz. De esta manera la arquitectura reconstruye de forma firme un modelo basado en RPC, aunque la implementación siga basada en un esquema de integración binaria, en vez de un modelo más abstracto.

8.4.3 CORBA - Arquitectura de Agente de Petición de Objeto Común [17], [18]

CORBA es un estándar que lo sostiene el Grupo de Manejo de Objeto (OMG) para la distribución de objetos a través de redes heterogéneas. Diseñado como una infraestructura de plataforma neutral para la comunicación inter-objetos, este a ganado una aceptación en todas partes.

CORBA permite usar a las aplicaciones una interfaz común, definidas en un Lenguaje de Definición de Interfaz (Interface Definition Language - IDL), a través de múltiples plataformas y herramientas de desarrollo. EL IDL de OMG está diseñado para que sea una plataforma y un lenguaje neutral; los datos y las llamadas a las conversiones de formato son manejadas transparentemente por el ORB. Todas las interfases de los objetos CORBA, y los tipos de datos usados en aquellas interfaces, están especificadas en el IDL. Esta definición común permite a las aplicaciones operar en objetos sin que importe la forma en como el objeto es implementado.

Visto por el cliente, un objeto CORBA es completamente opaco, en que la implementación del objeto y su localización es desconocida por la aplicación que lo usa. Generalmente, un cliente CORBA sólo sabrá encontrar o crear los objetos que éste necesite a través de las interfaces de objetos bien conocidos como los mecanismos de búsqueda y fábricas. Es probable que el cliente no sepa donde o como estos objetos bien conocidos están implementados, pero serán capaces de localizarlos por el nombre a través de el Servicio de Nombres de CORBA (CORBA Naming Service).

Es el caso de cualquiera de estos objetos CORBA, el cliente sólo sabe que interfaz pública del objeto es, y puede con ello a acceder a la funcionalidad del objeto. CORBA también proporciona algunas capacidades de identificación e invocación de interfaz de objeto a través de su Depósito de Interfaz (Interfece Repository - IR) e Interfaz de Invocación Dinámica (Dynamic Invocation Interface - DII). Mientras éstos tienen el potencial que permiten (casi siempre) una configuración completa en tiempo de ejecución del acceso a los objetos CORBA, en la práctica pueden existir muy pocos casos donde

dichas capacidades son laboriosos debido a los problemas semánticos. Indice

Recapitulación

Posterior a la aparición de internet y poco después que las Intranets fueron adoptadas por las empresas para el manejo de su información, nació la necesidad de que los sistemas cliente/servidor convencionales evolucionarán, hasta las técnicas actuales de distribución de objetos. Con esta técnica se pudo distribuir la interfaz con la que se pueden hacer consultas remotas. En los dos próximos capítulos se detallará una aplicación particular para las técnicas mencionadas en capítulos anteriores.

CAPITULO IX

Componentes de una Estación Meteorológica

Introducción

Son de gran importancia los criterios para el diseño de redes de monitoreo ambiental ya que deben tomarse diversos juicios de acuerdo a los objetivos que busca el científico o el ingeniero ambiental, además de las características físicas y urbanas del área, la localización y la cantidad de estaciones donde van a instalarse, el tipo de sistema de adquisición de datos, el sistema de comunicación los períodos de muestreo, los métodos de análisis, los sistemas de información geográfica, herramientas de computación y otros tipos de métodos auxiliares.

9.1 Tipos de Monitoreo [19]

Existen varios tipos de muestreos ambientales los cuales determinarán los elementos necesarios para una estación meteorológica, estos monitoreos pueden ser de tipo:

- atmosféricos
- pronósticos
- apoyo al tráfico aéreo
- servicios de agricultura
- servicios de hidrología
- Observaciones terrestres
- Observaciones en la atmósfera superior

En esta tesis basó en mediciones terrestres a nivel de suelo, por lo que se omitieron ciertos componentes que realizan otro tipo de observaciones (p.e. observaciones superiores a nivel atmósfera).

Antes de comenzar a ensamblar una estación meteorológica, se requiere de información geográfica y estadística para determinar el equipo necesario.

9.2 Partes que comprenden una Estación Meteorológica [19]

9.2.1 Información Meteorológica

Se debe saber el tipo y cantidad de información disponible que proporcionan otras organizaciones o empresas dedicadas al mismo ramo, ya que generalmente los datos meteorológicos son generados con propósitos diferentes como se mencionó anteriormente. Por lo que se requiere complementar la información que otros grupos han realizado con la que se piensa recuperar en el lugar donde se decidió la instalación de la estación.

Los servicios meteorológicos locales (si existen) tienen por lo común información general acerca de las condiciones climáticas del área. Es común medir la dirección y la velocidad del viento, las variaciones de temperatura con respecto a la hora, el día y la estación del año. Otros datos se pueden encontrar como la precipitación pluvial, períodos de insolación, humedad relativa y absoluta, entro otros. Estos serán una buena referencia para saber los factores que se piensan mediar para conseguir los sensores.

9.2.2 Información Topográfica

La topografía es un factor importante para seleccionar el sitio de muestreo por su efecto en los fenómenos de vientos locales y de estabilidad . Por ejemplo, la colocación de una estación en un terreno ondulado, presentará fuertes variaciones en la medición de fenómenos de viento en comparación a una superficie plana.

9.2.3 Clasificación de uso del suelo [22]

Es importante clasificar el uso de suelo de acuerdo a las actividades que se realizan. Una forma de realizar esta clasificación es por medio de la cartografía local, regional, o satelital las cuales nos proporcionan las características de la zona, es decir, la ubicación de zonas residenciales, comerciales, industriales, su densidad y el tipo de construcción, así como zonas ecológicas o verdes, u otro tipo de zona abierta. Es de enorme importancia contar con este tipo de información cartográfica para determinar las áreas de muestreo y saber que fenómenos medir y como van a ser afectados.

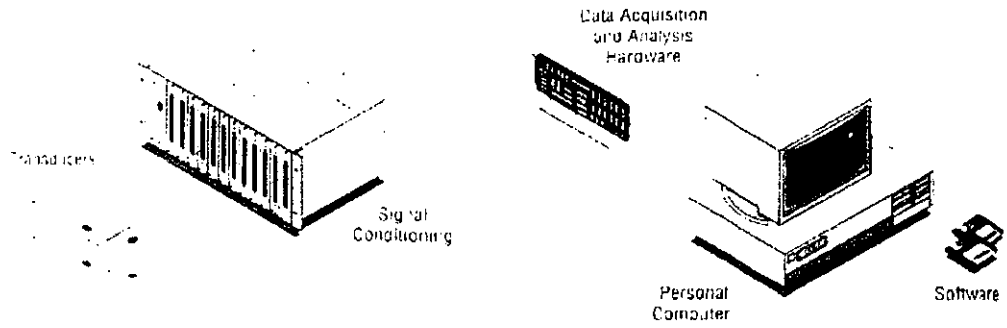


Fig. 9.1 Un sistema DAQ Típico.

9.2.4 Equipo [20]

Hoy en día, la mayoría de los científicos e ingenieros están usando computadoras personales con buses de expansión parar la adquisición de datos. Para obtener resultados apropiados de un sistema meteorología basado en una computadora, depende en cada uno de estos elementos de sistema (ver Fig 9.1).

- Computadora Personal
- Sensores (Transductores)
- Hardware de adquisición de datos
- Software

9.2.5 Sensores Meteorológicos [1]

Las observaciones hechas a nivel del suelo son más numerosas que las realizadas a altitudes superiores. Incluyen la medición de la presión atmosférica, la temperatura, la humedad, la dirección y velocidad del viento, la cantidad y altura de las nubes, la visibilidad y las precipitaciones (la cantidad de lluvia o nieve que haya caído).

Para la medición de la presión atmosférica se utiliza el *barómetro de mercurio.* Los barómetros aneroides, aunque menos precisos, son también útiles, en especial a bordo de los barcos y cuando se usan junto con un mecanismo de registro llamado barógrafo para registrar las tendencias barométricas a lo largo de un cierto periodo de tiempo. Todas

las lecturas barométricas empleadas en los trabajos meteorológicos se corrigen para compensar las variaciones debidas a la temperatura y la altitud de cada estación, con el fin de que las lecturas obtenidas en distintos lugares sean directamente comparables.

Para la observación de la temperatura se emplean muchos tipos diferentes de *termómetros*. En la mayor parte de los casos, un sensor comercial que abarque un rango habitual de temperaturas es más que suficiente. Es importante situarlo de modo que queden minimizados los efectos de los rayos solares durante el día y la pérdida de calor por radiación durante la noche, para obtener así valores representativos de la temperatura del aire en la zona a medir.

El instrumento que se utiliza más a menudo en los observatorios meteorológicos es el *higrómetro*. Una variante especial de este sensor, conocido como psicrómetro, consiste en dos termómetros: uno mide la temperatura con el bulbo seco y el otro con el bulbo húmedo. Un dispositivo más reciente para medir la humedad se basa en el hecho de que ciertas sustancias experimentan cambios en su resistencia eléctrica en función de los cambios de humedad. Los instrumentos que hacen uso de este principio suelen usarse en el radiosonda o rawisonde, dispositivo empleado para el sondeo atmosférico a grandes altitudes.

El instrumento más utilizado para medir la dirección del viento es la *veleta* común, que indica de dónde procede el viento y está conectada a un dial o a una serie de conmutadores electrónicos que encienden pequeñas bombillas (focos) en la estación de observación para indicarlo. La velocidad del viento se mide por medio de un anemómetro, un instrumento que consiste en tres o cuatro semiesferas huecas montadas sobre un eje vertical. El anemómetro gira a mayor velocidad cuanto mayor sea la velocidad del viento, y se emplea algún tipo de dispositivo para contar el número de revoluciones y calcular así su velocidad.

Las precipitaciones se miden mediante el *pluviómetro* o un *nivómetro*. El primero

es un cilindro vertical abierto en su parte superior que permite la entrada de la lluvia y calibrado en milímetros o pulgadas, de modo que se pueda medir la profundidad total de la lluvia caída. El nivómetro es también un cilindro que se hinca en la nieve para obtener una muestra. Después se funde ésta y se mide en términos de profundidad equivalente de agua, permitiendo con ello que su medición sea compatible con la de las precipitaciones. Las mediciones de la profundidad de la nieve caída se efectúan con una regla similar a las reglas comunes.

Los recientes avances producidos en el campo de la electrónica han ido acompañados de un desarrollo que del uso de instrumentos meteorológicos electrónicos. Uno de estos instrumentos es el *radar meteorológico*, que hace posible la detección de huracanes, tornados y otras tormentas fuertes a distancias de varios miles de kilómetros. Para tales fines, se usan las ondas de radar reflejadas por las precipitaciones asociadas con las alteraciones, que sirven para trazar su curso. Otros instrumentos meteorológicos electrónicos incluyen: el empleado para medir la altura de las nubes y el que se usa para medir el efecto total del humo, la niebla y otras limitaciones a la visibilidad. Ambos instrumentos suministran importantes mediciones para el despegue y aterrizaje de los aviones.

9.2.6 Modelos Climatológicos [1]

Los principios de las ecuaciones que gobiernan las condiciones físicas de la atmósfera se conocen desde hace mucho tiempo, pero sólo en fechas recientes se han desarrollado computadoras con suficiente potencia y rapidez. El mayor centro de ejecución de modelos climatológicos es el European Centre for Medium-Range Weather Forecasting (Centro europeo para la previsión meteorológica a plazo medio), situado en Bracknell, Inglaterra. La atmósfera es demasiado grande y compleja como para predecir con exactitud su comportamiento, incluso con los equipos más poderosos, pero es posible construir análogos matemáticos, o modelos, bastante realistas. En el modelo más simple sólo se predicen las condiciones a un único nivel. Es posible efectuar descripciones más realistas de la atmósfera empleando al mismo tiempo un gran número de niveles, y en el

modelo más sofisticado que se emplea hoy se usan nueve niveles. Las ecuaciones son tales que pueden calcularse los cambios en las propiedades atmosféricas a cada nivel para un breve plazo de tiempo tan sólo 10 minutos después de realizadas las observaciones. Las previsiones son después sustituidas por los datos iniciales observados, y el proceso se repite para sucesivos intervalos de tiempo hasta llegar a un plazo total de 72 horas. Los resultados así obtenidos para las 12, 24, 36, 48 y 72 horas posteriores a la hora inicial son trazados de modo automático sobre mapas que reflejan las condiciones previstas en los diversos niveles, y estos son transmitidos vía facsímil a las estaciones y otros usuarios del servicio.

9.2.7 Gráficas para interpretación

Los procedimientos descritos más arriba se realizan de modo automático, pero las previsiones resultantes requieren gran habilidad interpretativa. El clima se ve afectado en gran medida por condiciones locales que no pueden incluirse en los modelos. Además, los modelos no son representaciones perfectas de la atmósfera, y los meteorólogos experimentados prefieren en ocasiones no confiar en los resultados de los equipos, o pueden introducir en ellos modificaciones basadas en su propia experiencia.

Sin embargo para poder realizar una interpretación, los meteorólogos se basan en ciertas simbología y gráficas para representar un fenómeno climáticos. A continuación se presentarán algunas gráficas:

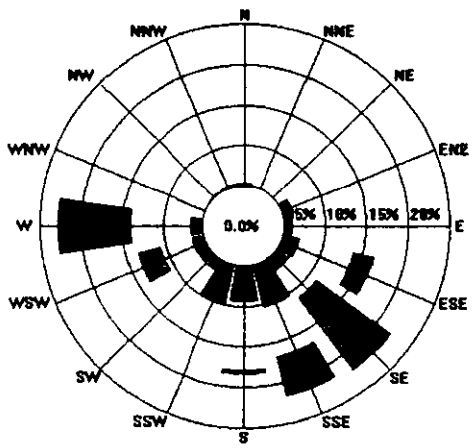


Fig. 9.2 Rosa de Vientos [21]

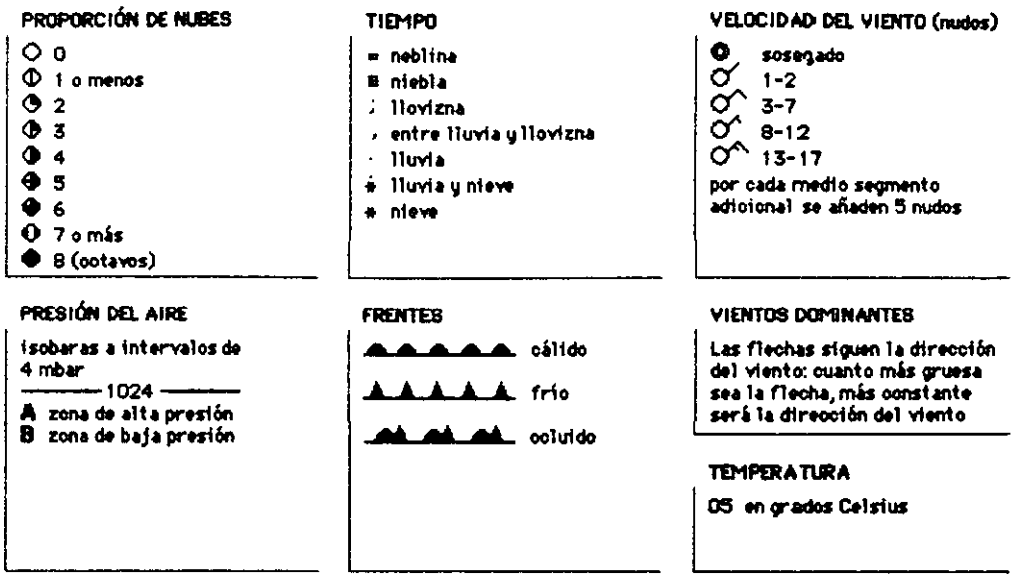


Fig. 9.3 Símbolos Meteorológicos [1]

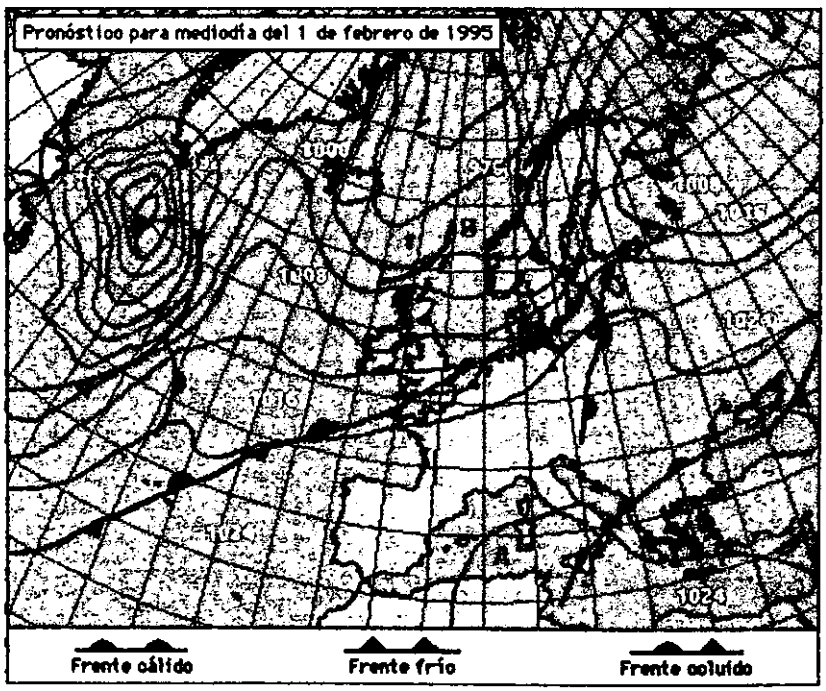


Fig. 9.4 Mapa Meteorológico [1]

Recapitulación

En este capítulo se mencionaron de forma general los elementos que pueden comprender un sistema de adquisición de datos meteorológico sin entrar en detalle, ya que un sistema de este tipo puede variar dependiendo las necesidades que el usuario tenga o el fin que tenga una estación meteorológica. Sin embargo, se mostraron las características principales que debe tener un sistema de adquisición de datos antes de realizar la compra o instalación de equipo o software para su estación.

CAPITULO X

Sistema de Comunicación Industrial

Introducción

A continuación se expondrán los materiales y procedimientos utilizados para desarrollar la interfaz que se propuso en el objetivo de esta tesis.

10.1 Material

10.1.1 Sensores con voltaje 0-10V

Los sensores con los que se hicieron las pruebas, fueron para medir las siguiente variables ambientales:

- Velocidad de viento [22]
- Dirección de viento [22]
- Temperatura [22]
- Presión Barométrica [22]
- Radiación solar [22]

Los sensores que se usaron fueron de la marca MetOne Inc. Para mayor detalle consulte el Anexo para descripción de los mismos.

10.1.2 Chasis SCXI-1000 / Tarjeta DAQ-1200

El chasís SCXI-1000 es de gran resistencia y de bajo ruido, en él, se pueden instalar tarjetas especiales para la adquisición de datos. Con este modelo, su forma de alimentación es a base de una alimentación AC, por lo que puede instalarse en cualquier lugar que proporciona una alimentación 127V AC. [20]

La tarjeta DAQ-1200 es un módulo de múltiples tareas analógicas, digitales que puede adquirir datos de cualquier asesor que trabaje con señales analógicas o digitales. La tarjeta puede conectarse directamente al puerto paralelo de la impresora a la computadora.la característica principal de esta tarjeta es que soporta hasta ocho entradas analógicas con un rendimiento de 100 kS/s a 12 bits, que soportan entradas programables para 0 - 10 V (unipolar) o ± 5 V(bipolar). [20]

10.1.3 NI-DAQ 6.2

NI-DAQ es un programa que se encarga de controlar y de reconocer el hardware que se mencionó en el inciso anterior, programando los componentes específicos de una computadora tales como los controladores DMA, servicio de interrupción, genera una interfaz entre los sistemas operativos y maneja los datos adquiridos. Además NI-DAQ proporciona una interfaz entre el ambiente desarrollo (LabView, C++, Delphi o Visual Basic) y la aplicación entre el equipo. [20]

10.1.4 NI-ComponentWorks 2.0

NI-Component Works es una colección de controles ActiveX de 32-bits diseñados para crear sistemas de instrumentación virtual. Al combinarlo con una herramienta que desarrolló estándar como Delphi, Visual Basic, Visual C++, proporcionan una solución de desarrollo completo para aplicaciones de automatización y medición basadas en computadora. Usando las librerías de estos componentes que puedes comunicar directamente a los equipos de adquisición de datos. [20]

10.1.5 Personal Web Server

Personal Web Server es un servidor Web personal que acelera y simplifica la configuración de un sitio Web, desde crear automáticamente una página principal personalizada hasta arrastrar y colocar publicaciones de documentos, además es una aplicación que puede usarse para compartir rápidamente documentos en su formato nativo, para convertir documentos al formato HTML o alojar páginas que contengan componentes adicionales como Applets o componentes ActiveX y entonces usar PWS para compartirlos entre diferentes sistemas operativos. [25]

10.1.6 Internet explorer

Internet Explorer es un programa que permite visualizar paginas Web en su computadora, este navegador de internet permite visualizar componentes ActiveX, lo que permitirá ver la interfaz gráfica (ClienteOCX) que fue descrita anteriormente. [25]

10.1.7 Intranet / Internet

Internet e Intranet son el medio de comunicación que utiliza el sistema SADMeteoro para consultar los datos meteorológicos remotos.

10.2 Procedimiento

Para desarrollar este sistema se elaboró un plan de trabajo (Fig 10.2) donde se contemplan 4 etapas. Esas se detallarán a continuación:

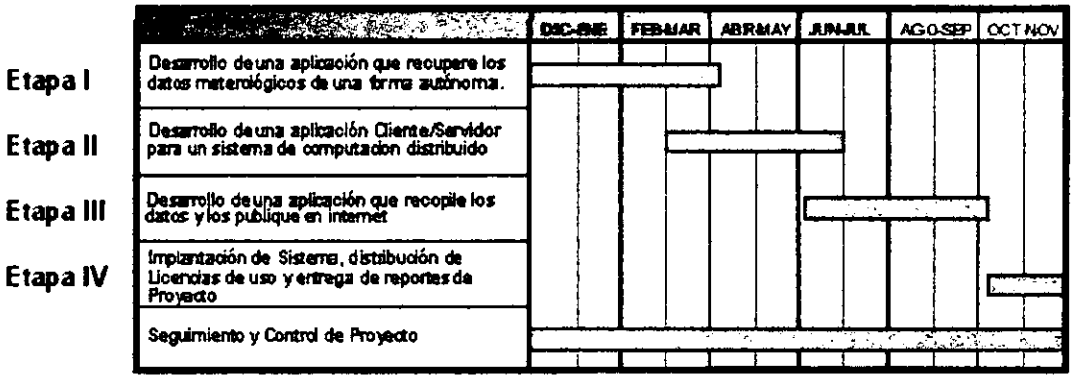


Fig. 10.1 Plan de Trabajo General

10.2.1 Etapa I

El estudio se realizo dentro del laboratorio de calibración que se encuentra ubicado en el edificio mixto "José López Portillo y Weber", el cual tiene instalado el servicio de Intranet del Instituto.

Se analizó la ficha técnica de cada sensor, así como de la tarjeta de adquisición de datos para determinar si entre estos elementos había compatibilidad. Posteriormente se ensambló la tarjeta SCXI-1200 dentro del chasis SCXI-1000 [26], y se conectó en la computadora. Los sensores se conectaron a la tarjeta de adquisición de datos usando un dispositivo secundario de conexión, vea el diagrama general de conexión en la fig. 10.1

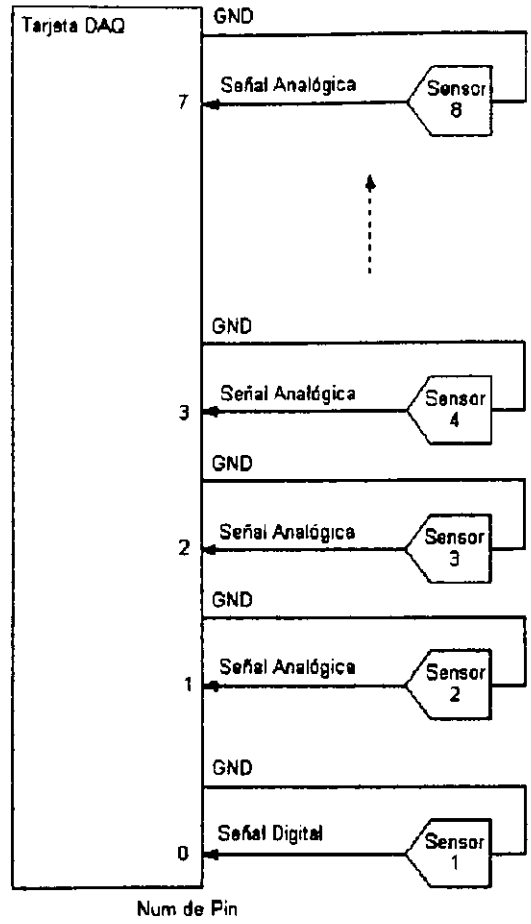


Fig. 10.2 Diagrama general de conexión de sensores.

En la computadora se instaló el software de configuración NI-DAQ 6.2 para dar de alta y configurar la tarjeta SXCI-1200. Posteriormente se instaló el compilador Delphi 4.0 C/S, además los componentes de adquisición de datos ComponentWorks 2.0. Posteriormente se empezó a diseñar 4 distintos programas, los cuales cumplen con diversas tareas pero dependen unas de otras para el funcionamiento del sistema.

Se desarrollaron dos aplicaciones prototipo que sirvieron de muestra para programar posteriormente el módulo "*InterfazChas*", el cual se encarga de leer la información de los sensores en un determinado lapso de tiempo. A su vez se creó simultáneamente el programa "*Interfaz*" el cual se encarga de modificar las características

de cada uno de los sensores.

Lógica de Trabajo "InterfazChaz"

Al activarse la opción de recuperar datos el programa se conectará primero al componente de adquisición de datos [23], este se encarga buscar la tarjeta SCXI-1200 por medio del programa de configuración *NI-DAQ* y le comanda leer la información del sensor. Cada determinado lapso de tiempo, el programa InterfazChaz muestrea los sensores y almacena los datos en su respectivo base de datos, convirtiendolos de voltajes a valores ambientales reconocibles. En caso de que el sistema se interrumpa, se podrá modificar el tiempo de muestreo. Al final de este capítulo, en la figura 10.3 se puede ver el esquema general de funcionamiento del programa "InterfazChaz".

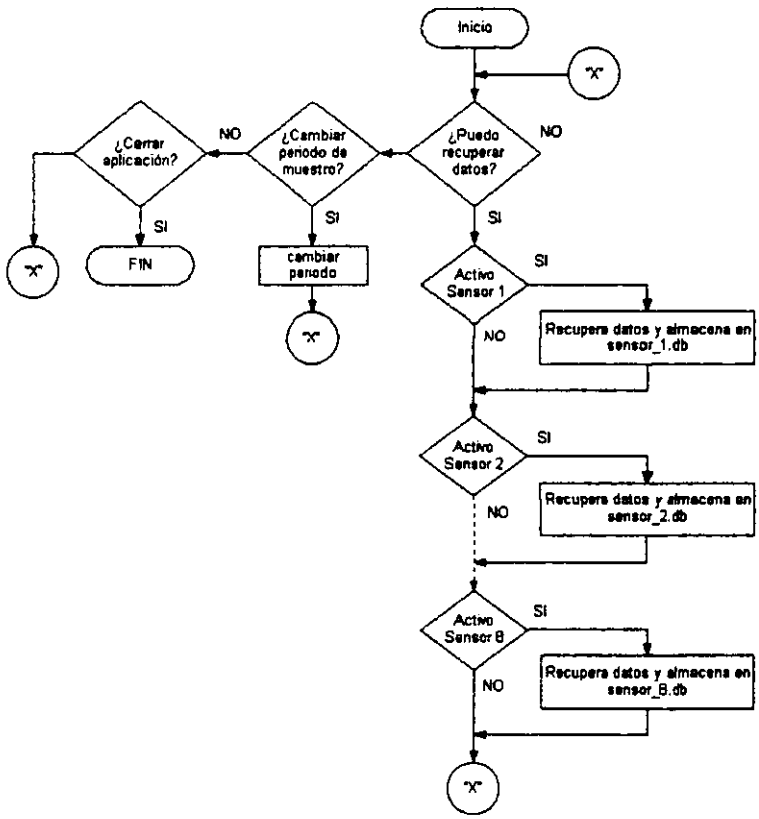


Fig. 10.3 Diagrama de flujo general del programa "InterfazChaz".

Lógica de Trabajo "Interfaz"

Interfaz no es un programa que necesite estar activo siempre, es una interfaz de captura que sirve para cambiar las propiedades de cada sensores y los almacena en la base de datos *Sensor.db*. Además, permite visualizar los datos ya almacenados por el programa *InterfazChas*, pero no permite la modificación de la información meteorológica.

El programa hace una conexión directa al archivo "sesnor.db" para mostrar las propiedades de la estación; en una ventana muestra los sensores y sus atributos, donde pueden modificarse los datos. Cuando se cambian los valores automáticamente se actualizan los datos en las bases si el usuario lo desea. La otra función de este programa es abrir la base de un sensor en particular solo para observar los datos almacenados.

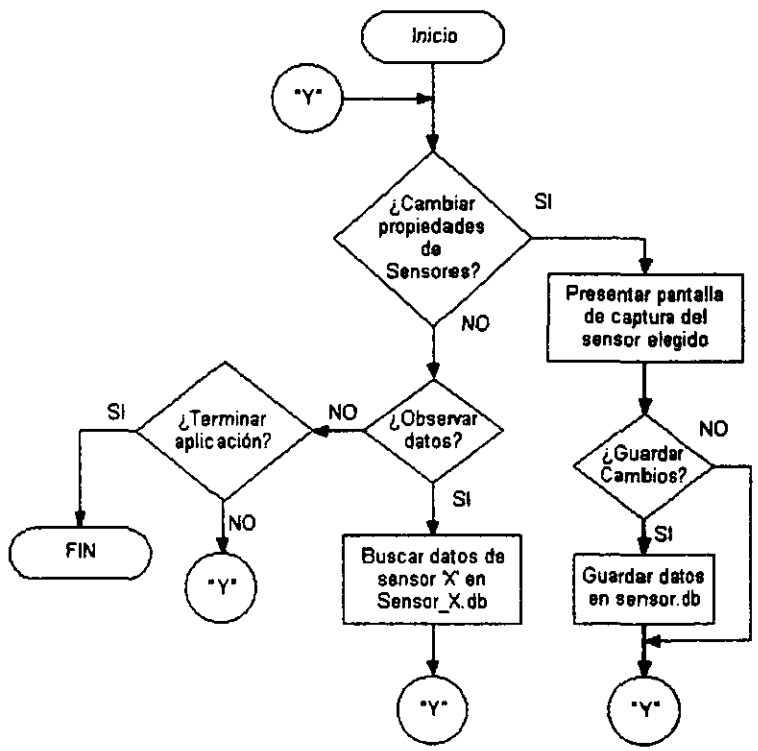


Fig. 10.4 Diagrama de flujo general del programa "Interfaz".

10.2.2 Etapa II

Se programaron los otros dos módulos, que se encargarán de distribuir la información a través de una Intranet.

En primer lugar se activó el "Personal Web Server" para que pueda hospedar la página que contiene el componente ClienteOCX, el cual es la interfaz gráfica que se distribuirá en la Intranet. El clienteOCX contiene varias funciones que permiten la consulta, presentación y graficación de los datos.

Lógica de Trabajo "ClienteOCX"

Este componente ActiveX, esta dividido en dos módulos: Comunicación al *ServidorOCX*, y manejo de datos. El primer módulo interno se encarga de solicitar los datos que el usuario solicite, este agente de solicitud viaja por la Intranet buscando el ServidorOCX y al encontrarlo deposita su petición. Si la petición es válida regresará la información solicitada y la presentará en el componente ClienteOCX que se encuentra dentro del "Internet Explorer". La información se clasifica dentro de una base de datos que el mismo componente crea en la máquina cliente, con permiso del usuario.

Una vez recibidos los datos y clasificados, el sistema cierra las comunicaciones y el siguiente módulo interno se encarga de manipular los datos, en la computadora cliente. Las funciones que existen son la de presentación y graficación de datos, donde el usuario puede interactuar con los datos. Al final de este capítulo, en la figura 10.5 se puede ver el esquema general de funcionamiento del programa *ClienteOCX*.

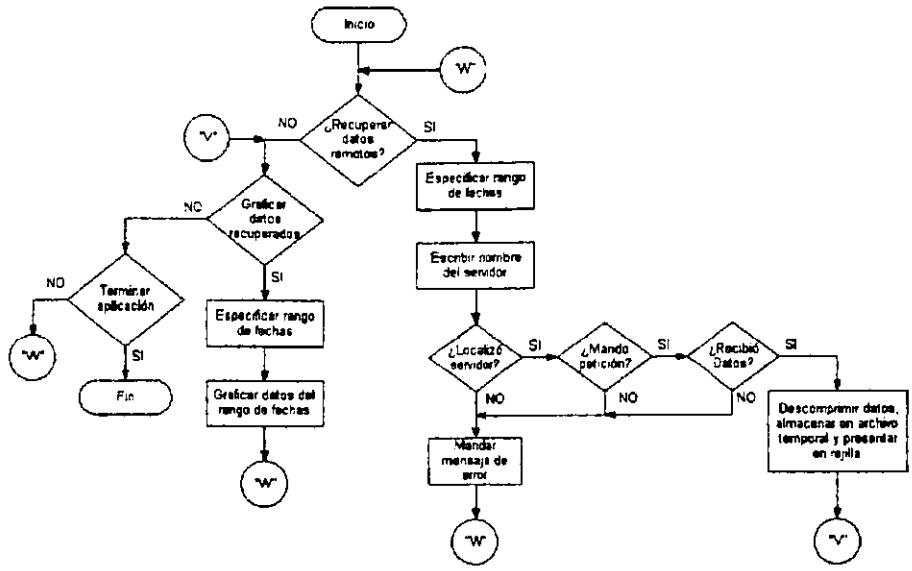


Fig. 10.5 Diagrama de flujo general del programa "ClienteOCX"

Lógica de Trabajo "ServidorOCX"

Esta aplicación, que siempre está activa, presta el servicio de recepción de solicitudes y responde a los clientes enviando la información clasificada pero codificada en paquetes, según sean las necesidades de información del cliente.

El servidor en espera, al momento de recibir la petición del cliente este guarda la petición momentáneamente. Empieza a leer la solicitud y conforme encuentra los datos los almacena en un paquete general. Una vez que termina de leer la solicitud, envía el paquete de datos por la Intranet hasta el cliente que solicito la información. El paquete no requiere de confirmación de recepción del remitente ya que de esto se encarga el protocolo TCP/IP.

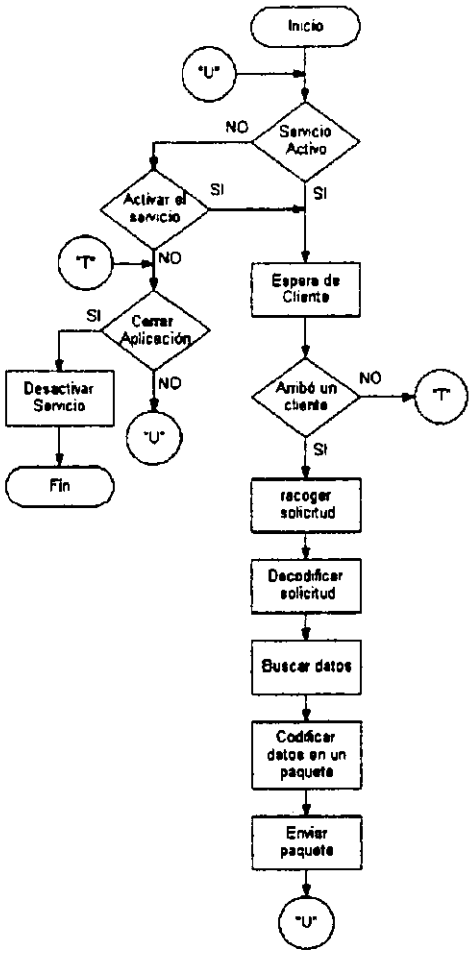


Fig. 10.6 Diagrama de flujo general del programa "ServidorOCX"

10.2.3 Etapa III

Las 4 aplicaciones antes descritas (InterfazChas, Interfaz, ServidorOCX y ClienteOCX) se ensamblaron en una computadora personal para que ésta fuera el instrumento de recopilación y distribución de datos meteorológicos en la Intranet.

Se realizó una prueba de funcionamiento, donde el sisema se dejó trabajando 24 horas, durante dos semanas para ver su conducta en operación critica, es decir, observar si existen anomalías al hacer trabajar de forma continua el sistema durante un prolongado tiempo sin operación manual.

Para llevar a cabo la prueba se necesitó de sensores de Velocidad de Viento, Dirección de Viento, Presión Barométrica, Temperatura y Radiación Solar, además de una computadora con los módulos de adquisición de datos (Chasis SCXI-1000 y la rajeta DAQ SCXI-1200) y los módulos de aplicación (InterfazChaz, Interfaz, ServidorOCX, ClienteOCX). Como referencia para el funcionamiento de los sensores este nuevo sistema fue colocado junto a otros sistemas comerciales de adquisición de datos meteorológicos.

El sistema para transportarlo fue desmontado del laboratorio, empacado y trasladado a la parte superior del edificio (azotea) del edificio de "Sismología" dentro del Instituto Mexicano del Petróleo. El equipo de desempacó e instaló en una caseta especial para sistemas de adquisición ambientales del mismo Instituto. Se instalaron los sensores a una torre meteorológica y se conectaron a la tarjeta SCXI-1200. Vea los diagramas de conexión del sensor de velocidad de viento en la fig. 10.7 y en la fig 10.8 para los otros sensores analógicos. El sistema sea activó y se dejó trabajando. Se hicieron revisiones para ver el funcionamiento del sistema durante cuatro días.

Transcurrido el tiempo de prueba el sistema fue llevado nuevamente al Laboratorio donde se hicieron las pruebas finales para comprobar si el sistema

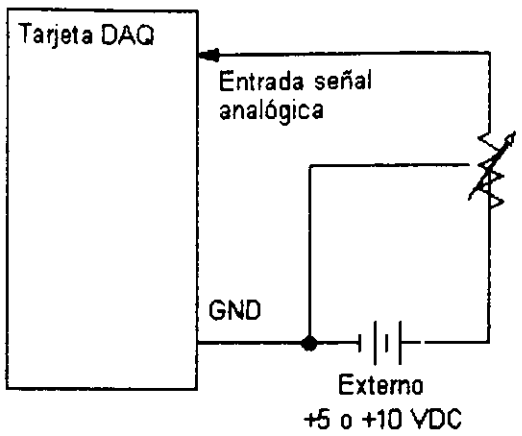


Fig. 10.7 Diagrama electrónico para conectar sensor analógico

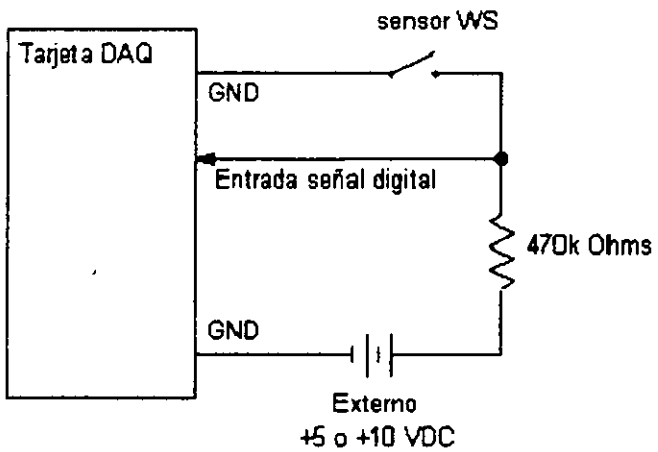


Fig. 10.8 Diagrama electrónico para conectar sensor digital

10.2.4 Etapa IV

Se elaboró la documentación de manejo que acompañará a este sistema, además de este impreso.

■

CAPITULO XI

Manual de Operación

SADMETEORO® es un conjunto de programas de software meteorológico gráfico. Este integra un formato de uso fácil para comunicar, coleccionar y reportar datos meteorológicos. El programa Interfaz simplifica la configuración de parámetros e identificación de la estación. Usa el programa INTERFAZCHASIS para que tu computadora sea el instrumento de recopilación de datos. Y usa CLIENTESAD para generar reportes en cualquier parte del la red de trabajo (o Internet).

11.1 Descripción

Este capítulo de esta tesis te presenta el SADMETEORO y te instruye rápidamente para su manejo. Te enseñara a usar las tareas más comunes con algunas recomendaciones para ahorrar tiempo. Son cuatro programas principales: **INTERFAZ, INTERFAZCHASIS, SERVIDORSAD, CLIENTESAD**. Existen 5 programas adicionales que se requieren para que funcione el sistema completo, vea el punto 11.3.1 para mas detalles. Cada programa puede ejecutarse de forma independiente o se ejecuta en segundo plano, lo que permite al usuario aprovechar los recursos de su computadora en otras tareas, sin tener que interrumpir el proceso que realiza SADMETEORO®.

Nota: Dependiendo la tarea que vaya a realizar la computadora algunos de estos programas no será necesaria su instalación o su ejecución permanente.

11.1.1 INTERFAZ

Interfaz es un programa que permite la configuración de la estación. Es un programa elemental ya que con él se interactúa de forma directa con las propiedades de cada sensor, así como las bases de datos que almacenan la información que el programa INTERFAZCHAS recopiló de los sensores. El programa presenta una interfaz amigable que permite la rápida configuración de todo el sistema. Contiene 3 carpetas, y cada una de ellas dan un resumen de las propiedades de la estación. Una de ellas te permite la visualización de datos en tiempo real. (El tiempo de muestreo depende de la configuración que tenga el programa INTERFAZCHASIS).

11.1.2 INTERFAZCHASIS

Permite la lectura, recopilación y almacenamiento automática de datos meteorológicos que proporcionan los sensores meteorológicos. Además, interfazchas es el enlace entre otros programas que dependen directamente de lo que este realice. Este programa realzará el muestreo de los sensores de acuerdo al tiempo que el usuario establezca.

11.1.3 SERVIDORSAD

ServidorSAD es un servidor de bases de datos que se usa para proporcionar el servicio de distribución de información meteorológica a clientes que lo soliciten en una Intranet (o Internet si el administrador de la empresa permite la publicación).

11.1.4 CLIENTESAD

Cliente OCX es un programa cliente que sirve como interfaz remota de las bases de datos que la computadora servidor se encarga de recopilar y almacenar. Cliente OCX además permite la graficación de los datos, por lo que se podrán hacer impresiones de la información que el usuario solicitó.

11.2 Usando la Documentación

Este punto proporciona un resumen general de la documentación, la cual está organizada de acorde a las necesidades primarias para configurar, recuperar, generar reportes, o transmitir datos por la red de la empresa (o Internet si la empresa lo permite) con SADMETEORO®.

11.2.1 Para recuperar información

La siguiente sección proporciona información que permite recuperar, graficar y generar un reporte de la información meteorológica almacenada tiempo atrás. Esta sección tiene la los elementos necesarios para realizar las estas tareas.

Vea la sección 11.5.2 "Recuperación y Graficación de Datos".

11.2.2 Para instalar, remover o cambiar propiedades de un sensor

Si se quiere modificar, instalar o remover un sensor de la estación, usando el programa INTERFAZ DE SADMETEORO, este te permitirá hacerlos rápidamente.

Vea la sección 11.5.4 "Agregando, removiendo y modificando Sensores"

11.2.3 Para cambiar el tiempo de muestreo

Se puede cambiar el tiempo que el programa InterfazChasis de SADMETEORO® usa para que el muestreo varíe según las necesidades del usuario y la capacidad de almacenamiento que contenga la computadora.

Vea la sección, 11.5.3 "Modificación del tiempo de recopilación"

11.3 Requerimientos del Sistema

Para instalar y ejecutar SADMETEORO®, la computadora debe tener ciertos requerimientos de hardware y de software. Este punto detalla los requerimientos, e incluye una tabla de uso fácil que puedes usar para reconozcas tu sistema.

11.3.1 Requerimientos de Hardware

Si planeas usar el equipo como solo recuperador de datos solitario no será necesaria la conexión a red. Si planeas implementar todo el sistema, sin importar el software que vayas a instalar en tu computadora, será necesario que tengas una conexión para red de trabajo. Además, el hardware debe tener las siguientes especificaciones:

Requerimientos de Hardware y Recomendaciones para:

SERVIDOR Y CLIENTE		
Componente Hardware	**Mínimo**	**Recomendación**
Procesador	66 Mhz 486	166 Mhz Pentium® o PII®
RAM	16 MB	32 a 64 MB
Espacio libre en disco duro	10 MB	100MB
Monitor	VGA (800-600)	Super VGA (1024-768)
Tarjeta de Video	800x600 (256 Colores)	1024x768 (16-bit)

Tabla 11.1 Requerimientos de Hardware

Hardware Adicional Requerido.

- Chasis SCXI-1000, marca: National Instruments®
- Tarjeta DAQ-1200, marca: National Instruments®
- Fuente de alimentación 12 VDC, 1Amp

11.3.2 Requerimientos de Software

El software tiene varios módulos, los cuales realizan tareas específicas por lo que dependiendo del tipo de el fin que vaya a realizar la computadora será necesaria la instalación de dichos módulos. A continuación se muestra una tabla de fácil lectura que te permite elegir el software que desees instalar:

Software Necesario			
Mínimo	**Recomendación**	**Cliente**	**Servidor**
S.O. Windows 95B© o superior	Windows 98© o Windows 2000©	●	●
Microsoft® Internet Explorer 4.0	Microsoft® Internet Explorer 5.0	●	●
DCOM95® para Windows 95	DCOM98© para Windows 98©	●	●
Borland DataBase Engine®	Borland DataBase Engine	●	●
NI-DAQ® 6.5.1 for Windows 95/98 and Windows NT 4.0.	NI-DAQ 6.5.1 for Windows 95/98© and Windows NT 4.0©	○	●
Microsoft® Personal WebServer© V4.0	Microsoft® Personal WebServer© V4.0	○	●

●- Requiere instalarse. ○ - No requiere instalarse

Tabla 11.2 requerimientos de Software

11.4 Instalando SADMeteoro®

En esta sección de explicarán los pasos necesarios para la instalación de SADMETEORO®. Según el software que se necesite se seguirán los siguientes pasos.

11.4.1 Antes de Instalar

El sistema SADMETEORO® requiere de cierto software para que este funcione junto con el hardware que lo acompaña. De lo contrario aparecerán mensajes de error que serán propios de SADMETEORO® o del hardware.

Nota: Si se especifica la letra "C"el punto deberá aplicarse al cliente, de igual manera con el servidor, se usará la letra "S".

11.4.1.1 Configuración de Fecha en Computadoras (C y S)

Antes de instalar o ejecutar tu aplicación, tanto el cliente como el servidor, por primera vez asegura que la "Configuración Regional" de tu computadora trabaje con las siguientes propiedades:

Fecha

En "Configuración Regional" del "Panel de Control" da clic en la lengüeta "Fecha" y asegura que:

- "Formato de Fecha Corta" sea: `dd/MM/aaaa`
- "Separador de Fecha" sea: `/`

Hora

En "Configuración Regional" del "Panel de Control" da clic en la lengüeta "Hora" y asegura que:

- "Formato de Hora"sea: `hh:mm:ss tt`
- "Separador de hora" sea: `:`
- "Símbolo a.m."sea: `a.m.`
- "Símbolo p.m."sea: `p.m.`

11.4.1.2 Instalación de software secundario (C-S)

Antes de instalar SADMETEORO® deberás instalar el software adicional que será necesario para el funcionamiento del sistema.

- Para instalar Microsoft® Web Server vea APÉNDICE A - MICROSOFT PERSONAL WEB SERVER (S)
- Para instalar y configurar su hardware vea APÉNDICE B - NI-DAQ 6.5.2 (S)
- Para instalar el controlador de bases de datos vea APÉNDICE C - BORLAND DATABASE ENGINE (C-S)
- Para instalar DCOM vea APÉNDICE D - DCOM (C-S)
- Internet Explorer 4.0 (se recomienda la versión 5.0 ya que en esta se

realizaron las pruebas de diseño).[C-S]

11.4.1.3 Instalación de SADMeteoro®[S]

Una vez que instaló y configuró el software secundario necesario en la computadora donde va a trabajar el sistema, ahora podrá instalar SADMETEORO.

Siga los siguientes pasos:

- Dentro del CD-ROM vaya al directorio: `\instalacion`
- De doble-clic a: `setup.exe`
- Siga los pasos

11.4.2 Durante la Instalación

En esta etapa deberás seguir todos los puntos que se te piden. Se recomienda ampliamente dejar los valores por omisión.

11.4.3 Al Final de la Instalación

Se recomienda que cada vez que se instale un programa nuevo se reinicie la computadora para que las nuevas configuraciones tengan efecto. Una vez instalado y configurado todo el software Se habrán creado varios directorios y accesos directos que te permitirán identificar los distintos programas que se instalaron. En caso de que no apareciera el acceso directo del programa que instalaste, por favor, instalalo de nuevo.

Para el sistema SADMETEORO® se habrá creado un grupo de programas llamado IMP-SADMETEORO, donde se encontraran 3 accesos directos a los programas p' . :ipales: INTERFAZ, INTERFAZCHASIS y SERVIDORSAD. El cuarto programa CLIENTESAD no tendrá acceso directo ya que solo se podrá ver por medio del Internet Explorer (previamente configurado).

11.5 SADMeteoro®

11.5.1 Características del SADMeteoro®

El sistema consiste de cuatro programas separados: INTERFAZ, INTERFAZCHASIS, SERVIDORSAD, y CLIENTESAD.

11.5.1.1 Programa Interfaz

Interfaz se ocupa para comunicar al usuario con la computadora. Es el programa que se encarga de configurar toda la estación. Además visualiza datos de forma general, en tiempo real, para ver el funcionamiento del sistema.

Carpeta Descripción

Muestra las propiedades de cada uno de los sensores. Cuando se conecta este programa a las bases de datos aparecen las propiedades mostradas en una tabla.

Carpeta Datos

En esta carpeta puedes ver los datos que han sido recopilados de cada uno de los sensores. En si lo que se observará serán los voltajes leídos de los sensores.

Nota: Con el programa "CLIENTESAD" se verán los valores transformados.

Carpeta Funcionamiento

En este programa es donde se pueden ver los datos que son recopilados, en tiempo real, por el programa InterfazChasis y depositados en las bases de datos.

11.5.1.2 Programa InterfazChasis

INTERFAZCHASIS se usa para comunicar la computadora con los sensores. Es el enlace que mide los voltajes de los sensores que el hardware adicional (SCXI) mide y establece el tiempo de muestreo de los fenómenos naturales. Además es un tablero que muestra el funcionamiento de los sensores.

11.5.1.3 Programa ServidorSAD

SERVIDORSAD se encarga de establecer la comunicación entre el sistema y el programa ClientSAD, ya que se encarga de leer las peticiones de este y manda un paquete como respuesta a su solicitud.

11.5.1.4 Programa ClienteSAD

CLIENTESAD solicita, gráfica y reporta datos meteorológicos. Cualquier medición hecha por INTERFAZCHASIS aquí se podrá representar en una gráfica escalar o angular. Establece el nexo con un servidor local o remoto (SERVIDORSAD), y realiza solicitudes que el usuario necesite solo con el uso del ratón. Existen 3 módulos desmontables en CLIENTESAD:

Módulo 1

Se encarga de visualizar los datos que serán o fueron solicitados por el usuario. Los datos se solicitaran dentro de un rango específico de fechas.

Módulo 2

Se encarga de graficar los datos solicitados por el módulo anterior. Se puede realizar un reporte impreso de la gráfica. Esta gráfica puede realizar acercamientos para ver con detalle cada punto de muestreo.

Módulo 3

Se encarga de presentar los análisis mas comunes de los datos que se visualizan con el Módulo 2, es decir, analiza el valor medio, máximo, mínimo. Además, cuenta con una brújula que simula las condiciones de dirección de viento durante ese periodo.

11.5.2 Recuperación y Graficación de Datos

SADMETEORO realiza estas dos tareas de forma rápida y simple usando dos programas del sistema: Interfaz e Internet Explorer (CLIENTESAD).

11.5.2.1 Recuperar datos usando Interfaz

a) Abra su aplicación Interfaz: Inicio >> Programas >> SADMETEORO >> Interfaz y aparecerá una ventana como ésta.

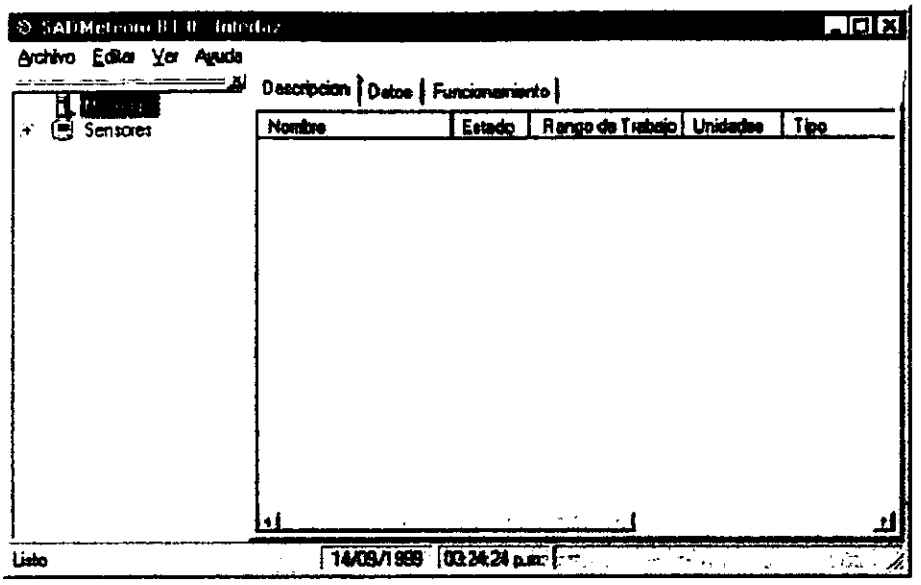


Fig. 11.1 Interfaz sin conexión

b) Presione Conectar y aparecerán las propiedades de cada sensor en la parte derecha.

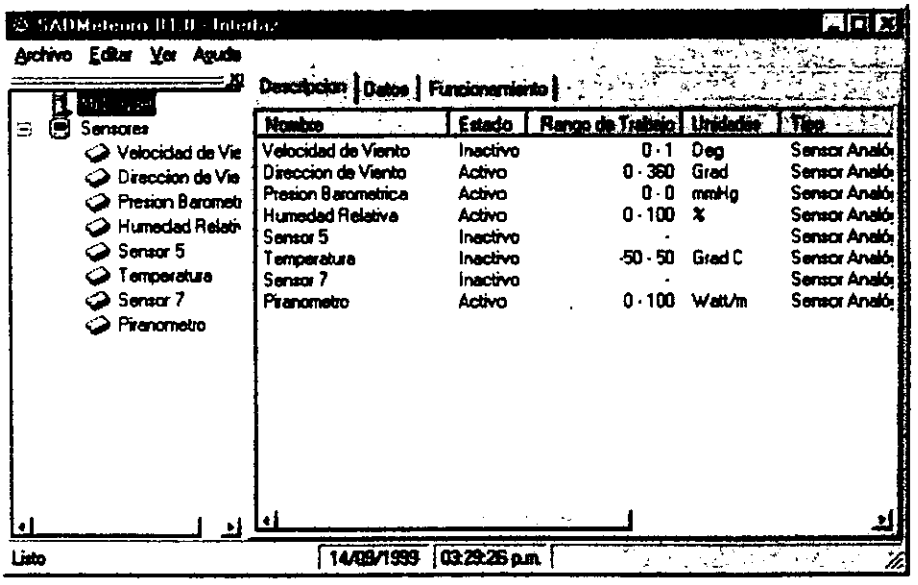


Fig. 11.2 Interfaz con propiedades de sensores

Nota: Si no aparece el nombre de los sensores por favor vaya al punto 11.5.4.2 "Modificar propiedades de sensores".

c) En la parte izquierda de la pantalla expanda el directorio sensores, seleccione un sensor, vaya a la carpeta "Datos", seleccione un rango de fechas específicas y presione el botón "Busca" y aparecerá una pantalla parecida a esta.

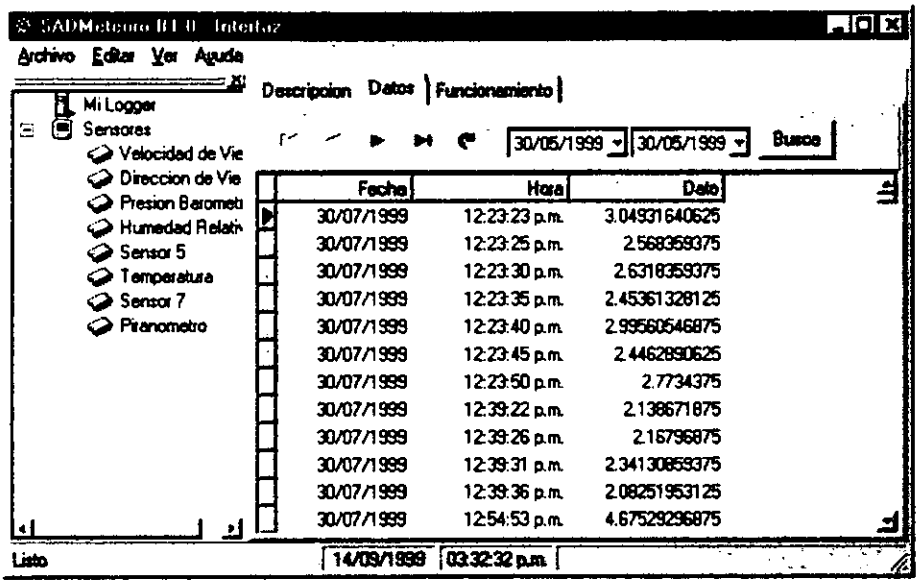


Fig. 11.3 Datos de un sensor

Nota: En caso de no obtener datos, ve a la sección 11.5.6.4 "Problemas Comunes"

11.5.2.2 Recuperar datos usando Internet Explorer

a) Abra su aplicación Internet Explorer: Inicio >> Programas >> Internet Explorer. Escriba la dirección siguiente: http://nombre_servidor/ y aparecerá una ventana como esta:

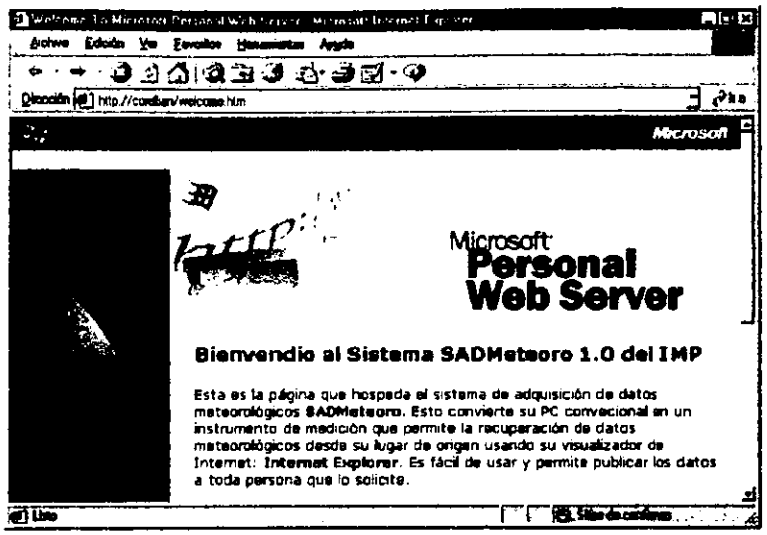


Fig. 11.4 Pagina principal del servidor Web personal.

b) Dentro de la página web llame la pagina con el programa y aparecerá el CLIENTESAD. Asegurando que el SEVIDORSAD esta trabajando y está activo,

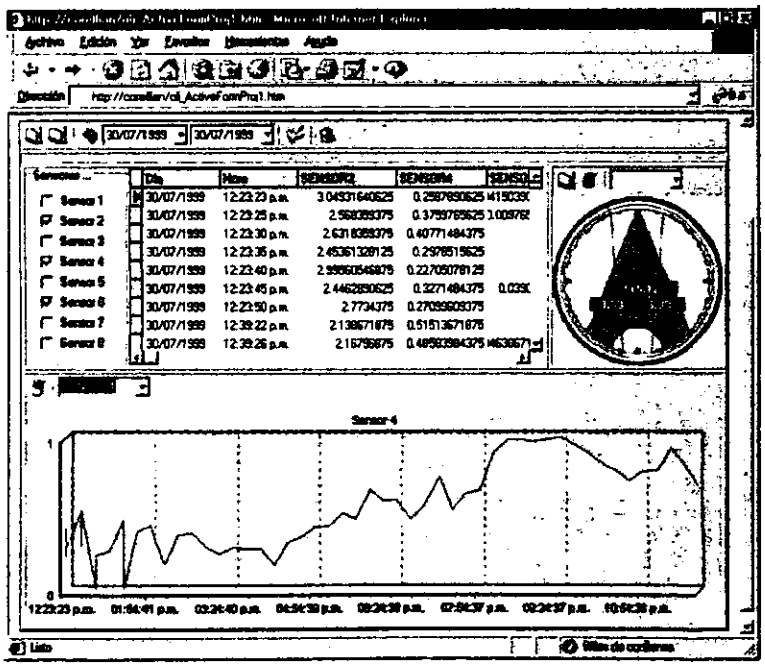


Fig. 11.5 Componente ClienteOCX con datos recuperados

presione el botón de conexión remota (1), seleccione un rango de fechas (2), y presione el botón se solicitud (3).

11.5.2.3 Graficar los datos recuperados usando Interfaz

Una vez conectado a las bases de datos, se podrá obtener una gráfica con movimiento. Vaya a la carpeta "Funcionamiento", active el botón "Mirar Sensor" y tendrá una gráfica que se irá actualizando conforme el programa InterfazChasis vaya depositando la información de los sensores en las bases de datos.

11.5.2.4 Graficar los datos recuperados usando Internet Explorer

Una vez recuperados los datos con su CLIENTESAD, ahora podrá graficar y generar un reporte con el Módulo 2 (Módulo que esta en la parte Derecha). Elija un sensor dentro del módulo, clic en el botón de dibujo y tendrá la gráfica. Clic en el botón de reporte y obtendrá un reporte detallado de la gráfica según sean sus necesidades.

11.5.3 Modificación del tiempo de recopilación

El sistema trabaja según el tiempo de muestreo que se le asigne. Es decir, si se le asigna 5 minutos de muestreo, el sistema recuperará datos cada 5 minutos.

11.5.3.1 Modificar el tiempo de muestreo

Para cambiar el tiempo o la frecuencia de muestreo clic-derecho en el pantalla digital verde y elije del menu el periodo que desee. Como observación entre más corto sea el tiempo de muestreo, se incrementarán las bases de datos más rápido por lo que el espacio en el disco duro se ira terminando conforme pase mayor tiempo. Sin embargo, las bases de datos ocupan, por muy saturadas que estas estén, un espacio pequeño.

Fig. 11.6 Display de periodo de muestro de InterfazChaz

11.5.3.2 Activar o Desactivar el muestreo

El sistema puede estar en dos estados: "Activo" o "Stand By...", pero nunca estará apagado. Este solo se apagará si se cierra la ventana. Para cambiar el estado clic en el botón cuadrado.

Fig. 11.7 Botón de activación de muestro del sistema

11.5.4 Agregar, modificar y remover Sensores

Tres actividades principales se realizan para con los sensores, en esta estación.

11.5.4.1 Agregar un sensor

1. En la parte derecha de la pantalla donde aparecen las características generales del sensor clic-derecho en el nombre del sensor (por e.j. Sensor 1). Elija "Propiedades" y aparecerá una ventana como la de abajo:

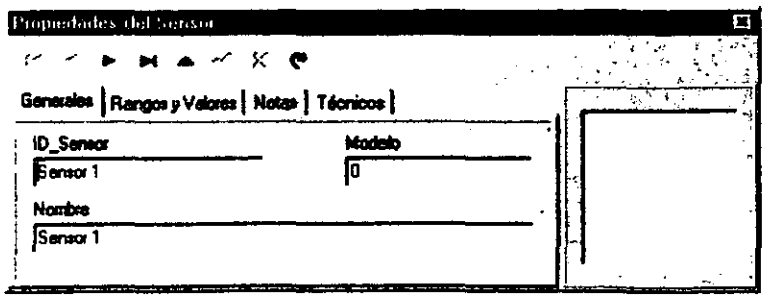


Fig. 11.8 Pantalla de captura del un sensor.

2. Escriba cada una de las propiedades del sensor, según los parámetros que

proporcionó el fabricante del sensor. Para no omitir ninguna propiedad configure las cuatro carpetas.

3. Una vez terminada la configuración, presione el botón para guardar los cambios y cierre la ventana.

11.5.4.2 Modificar propiedades de sensores

1. Siga los pasos 1 y 2 del punto 11.5.4.1 pero ahora elija el sensor que desee y cambie los valores anteriores por los nuevos..

2. Una vez terminada la configuración, presione el botón para guardar los cambios y cierre la ventana.

11.5.4.3 Activar un sensor

Puede que se necesite retirar un sensor por cualquier motivo del sistema, (p.e. mantenimiento, calibración, etc.) , y se desea mantener la configuración, además de que el sistema siga funcionando. De clic en el lugar donde se muestra en la figura de abajo para activar o desactivar el sensor. ('☒' Activado, '☐' Desactivado)

11.5.4.4 Remover un sensor

Para remover o eliminar un sensor junto con sus propiedades siga el paso 1 del punto 11.5.4.1 y elija del menu eliminar.

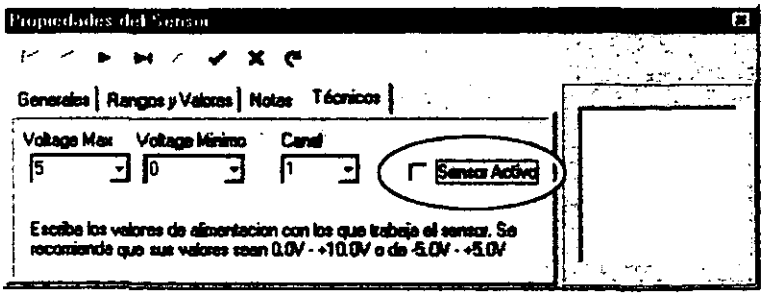


Fig. 11.9 Detalle sobre activar un sensor

11.5.4.5 Cancelar cambios

Si tuvo un error al copiar las propiedades o quiera mantener la configuración anterior de clic en el botón: ×|. Sin embargo, si dio primero clic en el botón ✓| no podrán revertirse los cambios.

11.5.5 Descripción del Programa

En esta sección del capítulo de la tesis, se describirán todas las funciones de cada programa (INTERFAZ, INTERFAZCHAS, SERVIDORSAD y CLIENTESAD) que integran el sistema SADMETEORO®.

11.5.5.1 Interfaz

Datos interfaz esta dividido en 3 Módulos

Módulo 1 Interfaz

Aquí es donde se resume todas las características y propiedades del sistema SADMETEORO, esta se divide en:

1) Menu de Comandos

Archivo > **Salir** - Cierra la aplicación.

Editar > **Copy** - Copia cualquier selección al portapapeles de la computadora.

Editar > **Cut** - Corta cualquier selección al portapapeles de la computadora.

Editar > **Paste** - Pega el contenido del portapapeles a un lugar específico elegido por el usuario.

Ver > **Arbol** - Cuando se haya cerrado el "Árbol de Sensores" con este comando volverá a aparecer.

Ayuda > **Ayuda Online** - Muestra una ventana con lo necesario para solicitar soporte técnico calificado.

2) Árbol de Sensores

Este es lugar donde se muestran en forma jerárquica los sensores

disponibles instalados para este sistema.

3) Carpeta Descripción

En esta carpeta se muestran todas las características o propiedades que tiene cada sensor, muestra sus estado y una breve descripción de estos.

4) Datos

Aquí se podrán visualizar los datos que estuvo recopilando un sensor específico. Se le puede especificar una fecha específica que muestre los datos.

5) Funcionamiento

Esta carpeta nos muestra una pantalla que al activarla nos muestra los datos que está recopilando en ese instante el programa InterfazChasis.

Módulo 2 Propiedades del Sensor

Esta ventana aparecerá cuando se de clic-derecho y se elige propiedades en cualquier sensor dentro de la Carpeta Descripción. En ella se encontrarán:

1) Botones de Comando

Con estos botones se puede navegar entre los sensores, mostrando las características de cada uno de ellos, pero por omisión mostrará el sensor que se eligió. Se describirán de izquierda a derecha:

Primero - Se mueve hasta el primer sensor.

Anterior - Se mueve al sensor anterior.

Siguiente - Se mueve al siguiente sensor

Ultimo - Se mueve al último sensor

Editar - Se activa el modo de edición de las propiedades del sensor

Guardar - Se guarda los cambios que se hicieron al sensor

Cancelar - Se eliminan los cambios hechos al censor mueve al sensor (Si se di clic al botón guardar antes este botón no funcionará)

Refrescar - Se actualizan las bases de datos sensor (su uso es solo para mantenimiento)

2) Carpeta Generales

Aquí se escribirán el número de identificación (ID_Sensor) del sensor, su modelo (Modelo) y su nombre informal (Nombre)

3) Carpeta Rangos y Valores

Escriba los valores y unidades con los cuales trabaja el sensor para la interpretación de datos.

4) Carpeta Notas

Escriba cualquier nota referente al sensor.

5) Carpeta Ténicos

Escriba los valores de alimentación con los que trabaja el sensor. Se recomienda que sus valores sean 0.0V - +10.0V o de -5.0V - +5.0V

6) Grafico

Si cuenta con una gráfica o una imagen que represente a su sensor de clic-derecho y cargue su imagen favorita.

Módulo 3 Ayuda Online

Mostrará una ventana con la información necesaria para proporcionar soporte técnico cuando Ud. lo necesite.

11.5.5.2 InterfazChas

En este programa se hará el muestreo de los sensores instalados, clasificará

los datos y los colocará en las bases de datos correspondientes. Este sistema depende de la configuración que se le haya dado en el programa INTERFAZ. Los elementos que contiene son:

Pantalla de funcionamiento

En ella se muestran indicadores (leds) de color verde los cuales muestran si un sensor está activo o no, y muestran el valor numérico leído de los sensores.

Botón de activación

Este botón permite el funcionamiento o interrupción del funcionamiento del sistema de adquisición de datos.

Pantalla de muestreo

Es el que sirve como indicador de la frecuencia con la cual está trabajando el sistema actualmente, además al dar clic-derecho en el aparece un menú con varios tiempos de muestreo con los cuales trabaja el sistema.

Botón de Configuración

Es un botón de colo amarillo que al dar clic-derecho manada llamar a la aplicación Interfaz para que pueda configurar las propiedades del sistema.

11.5.5.3 ServidorSAD

Es módulo del sistema es el que se encargará de proporcionar servicio a los usuarios que soliciten datos por medio del CLIENTESAD. Este sistema tiene los siguientes elementos:

Botones Activa y Desactiva

Estos botones son los que activan o interrumpen el servicio.

11.5.5.4 ClienteSAD

Este módulo activo es el que se encargará de mostrar, graficar y generar reportes de los sensores según las fechas que se le establezcan. Este programa

esta dividido en:

Barra de Herramientas

Fig. 11.10 Barra de herramientas del componente ClienteOCX

Con está barra se realizarán las tareas más comunes e indispensables para la recopilación de datos. Se comentarán de Izquierda a derecha:

1) Conectar - Este botón establecerá la conexión con el programa SERVIDORSAD para poder realizar las transacciones.

2) Desconectar - Este botón desconectará con el programa SERVIDORSAD, pero podrán realizarse tareas con los demás módulos. Se recomienda su uso cuando salga del programa.

3) Buscar - Este botón se encargará de realizar las transacciones con el SERVIDORSAD y las necesidades del usuario. Se encargará de solicitar datos meteorológicos a partir del rango de fechas que se establezca.

4) Fecha Inicial / Fecha Final - Estos menús que despliegan calendarios, son el rango de fechas que usuario podrá mover según así lo desee. Se recomienda que el rengo de días sea pequeño (5 o menos), ya que el tiempo de recuperación variará con respecto a la velocidad de transmisión de la red y de la velocidad de procesamiento de la computadora.

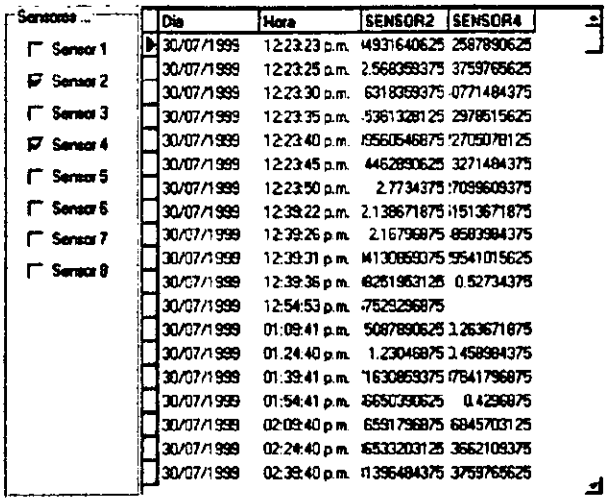
5) Reporte - Este botón realizará una impresión de lo que se muestra en el programa CLIENTESAD.

6) Limpiar Bases Temporales - Este botón se usa para mantenimiento de las bases de datos. Con él se borran las bases de datos temporales que se generan cada vez que se solicitan datos. Se recomienda usarlos después que termine de trabajar con el programa.

Módulo de datos

En este módulo se presentan los datos meteorológicos recuperados de la

estación en una tabla. Con ellos se pueden trabajar para realizar los reportes, gráficas o una simulación.



Dia	Hora	SENSOR2	SENSOR4
30/07/1999	12:23:23 p.m.	4931640625	2587890625
30/07/1999	12:23:25 p.m.	2.568359375	3759765625
30/07/1999	12:23:30 p.m.	6318359375	-0771484375
30/07/1999	12:23:35 p.m.	-5361328125	2978515625
30/07/1999	12:23:40 p.m.	-9560546875	-2705078125
30/07/1999	12:23:45 p.m.	4462890625	3271484375
30/07/1999	12:23:50 p.m.	2.7734375	-7099609375
30/07/1999	12:39:22 p.m.	2.138671875	-1513671875
30/07/1999	12:39:26 p.m.	2.16796875	-8583984375
30/07/1999	12:39:31 p.m.	-4130859375	9541015625
30/07/1999	12:39:36 p.m.	-8251953125	0.52734375
30/07/1999	12:54:53 p.m.	-7529296875	
30/07/1999	01:09:41 p.m.	5087890625	1.263671875
30/07/1999	01:24:40 p.m.	1.23046875	1.458984375
30/07/1999	01:39:41 p.m.	-1630859375	-7841796875
30/07/1999	01:54:41 p.m.	-8650390625	0.4296875
30/07/1999	02:09:40 p.m.	6591796875	6845703125
30/07/1999	02:24:40 p.m.	-6533203125	3662109375
30/07/1999	02:39:40 p.m.	-1396484375	3759765625

Fig. 11.11 Detalle datos recuperados en ClienteOCX

En la gráfica se muestran los datos de los sensores que el usuario eligió. No es necesario la recuperación total ya que tomaría bastante tiempo la recuperación de datos.

Módulo de gráfica

Aquí se podrán realizar gráficas de los datos recuperados del tipo escalar. Se elige un sensor del menú y se gráfica presionando el botón de pintado, ya que este módulo depende de las fechas que el usuario haya establecido en la barra de herramientas.

Módulo de simulación

Este es un lugar donde se podrá realizar la simulación del comportamiento de datos de acuerdo al rango de tiempo que el usuario establezca en la barra de herramientas.

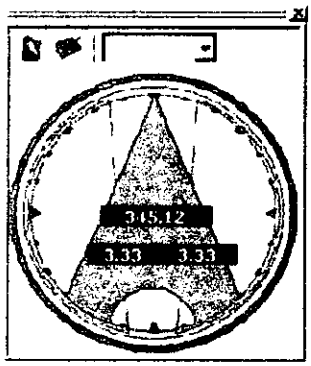


Fig. 11.13 Detalle gráfica en ClienteOCX

Elija el sensor del menú, presione el botón de simulación y verá una animación de lo que ocurrió durante el rango de fechas que se estableció en la barra de herramientas.

11.5.6 Problemas

Si tiene dificultades con su sistema SADMETEORO, utilice las secciones siguientes de este tema para solucionar el problema.

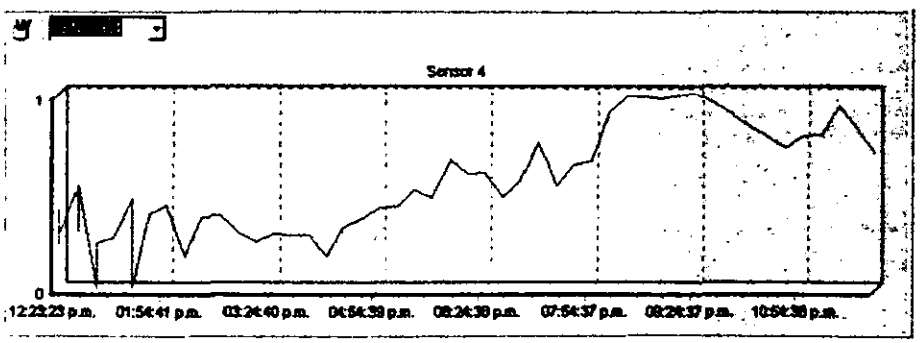


Fig. 11.12 Detalle de datos graficados en el ClienteOCX.

11.5.5.1 Instalación

Si tiene problemas al instalar SADMETERO o de funcionamiento del programa tras la instalación, consulte los elementos siguientes.

Desinstalar versiones anteriores

Esta versión realiza automáticamente la actualización a PWS versión 1.0. Si tiene PWS 4.0 versión Alpha o Beta 2, debe desinstalarla antes de instalar esta versión. Realice la desinstalación mediante la opción Quitar todo del programa de instalación incluido en el disco compacto de la versión anterior. Si la opción Quitar todo no está disponible, utilice Agregar o quitar y desactive todas las casillas de verificación.

Carpeta de instalación

De manera predeterminada, Personal Web Server se instala en C:\Windows\System\Inetsrv. No puede cambiar la carpeta predeterminada de instalación. El directorio de publicación se instala, de manera predeterminada, en C:\Inetpub\Wwwroot. Si cambia el directorio predeterminado de publicación, asegúrese de que escribe un nombre de ruta completo, incluyendo la letra de unidad. El programa de instalación puede malinterpretar las rutas relativas y las que no contienen una letra de unidad.

11.5.5.2 Probando tu Instalación

Puede probar la instalación si utiliza Internet Explorer para ver los archivos de su directorio particular.

Para probar un sitio Web de su intranet

1. Compruebe que el equipo tiene una conexión activa de red y que el servicio Servidor WINS (u otro método de resolución de nombres) está funcionando.
2. Inicie un explorador de Web, como Internet Explorer.
3. Escriba el Localizador de recursos universal (dirección URL) para el

directorio particular del nuevo servidor y presione ENTRAR.

La dirección URL es "http://" seguida del nombre de su servidor en la red de Windows y la ruta del archivo que desea ver (incluya las barras diagonales). Por ejemplo, si su sitio Web está registrado en el servidor WINS como "Admin1" y desea ver el archivo Homepage.htm en la raíz del directorio particular, debe escribir en la Barra de direcciones del explorador http://admin1/homepage.htm y, después, presionar ENTRAR. Se mostrará la página principal en la pantalla.

Para probar un sitio Web que no está conectado a una red

1. Utilice el Explorador de Windows para buscar el archivo Hosts.sam en la carpeta \Windows y ábralo con un editor de textos, como Bloc de notas.
2. Bajo la línea 127.0.0.1 localhost, escriba 127.0.0.1 nombre_de_su_equipo, donde nombre_de_su_equipo es el nombre de equipo mostrado en la página de propiedades Identificación en el subprograma Red del Panel de control.
3. Guarde el archivo con el nombre Hosts, sin extensión.
4. Abra Internet Explorer, haga clic en Ver, Opciones, Conexión y seleccione Usar un servidor proxy para acceder a Internet.
5. Haga clic en Opciones avanzadas y agregue el nombre de su equipo a la lista Excepciones.
6. Escriba localhost o el nombre del equipo en la Barra de direcciones del explorador y presione ENTRAR.

Nota También puede utilizar la dirección IP 127.0.0.1 para llegar al sitio Web.

11.5.6.3 Problemas Técnicos o Comunes

Para analizar y corregir problemas comunes revise el siguiente documento:

a. En el CD-ROM diríjase al directorio `\Documentos\`

b. De doble-clic al documento `preguntas.htm`

■

Resultados

A continuación de describirán los resultados obtenidos por etapa:

Etapa 1

Los sensores son compatibles con la tarjeta DAQ SCXI-1200, ya que se alimentan con voltajes de 5 o 10 VDC, sin embargo el sensor de Temperatura necesita de una alimentación de 10 VDC ya que no presenta respuesta cuando se le suministra 5 VDC.

Es muy importante revisar siempre la ficha técnica del sensor ya que el sistema solo funcionará con señales de 5 ó 10 VDC, por tal motivo si se excede este margen puede dañarse irreversiblemente su equipo de adquisición de datos.

El sensor de Velocidad de Viento en su funcionamiento presenta dos estados: Abierto y Cerrado. Este sensor presenta ruido en la señal cuando está en el estado de abierto por lo que requirió de una instalación especial[1]

El sistema actual solo soporta un sensor digital por lo que se recomienda especial cuidado al instalarlo, consulte un técnico electrónico calificado para que lo asesore, de lo contario se provocará un corto circuito que dañará su tarjeta de adquisición de datos irreversiblemente. Vea el diagrama siguiente para su conexión:

Siga cuidadosamente las instrucciones de conexión de su sensor, así como el manual de usuario de la tarjeta DAQ, donde se detalla los 8 pines analógicos que utiliza este sistema para la recuperación de datos. Se pide tenga especial cuidado en distinguir una tierra física (GND) y una tierra lógica de lo contrario el sistema se dañará permanentemente.

Si se desea dar mantenimiento o desea mejorar los módulos debe ser una persona capacitada para hacer los ajustes necesarios, de lo contrario, su equipo puede resultar

---

[1] Vea la gráfica 10.7 del capítulo 10.

seriamente dañado.

La aplicación InterfazChas recupera los datos de los sensores basandose en las especificaciones técnicas del sensor, las cuales se encuentran dentro de la base de datos sensor.db. La información que se recupera se almacena en la base de datos sensor_1.db, sensor_2.db, sensor_3.db, sensor_4.db, sensor_5.db, sensor_7.db, sensor_8.db.

La aplicación Interfaz modifica la información de la base sensor.db si el usuario así lo desea. Además permite ver los datos recuperados de la base de datos sensor_x.db, dentro de un rango de fechas.

Es importante que cualquier alteración manual a estas bases de datos provocará el mal funcionamiento del sistema y como resultado puede perder la información irremediablemente.

Etapa 2

Al fabricar el sistema de adquisición de datos se notó que la tecnológica ActiveX es una buena herramienta si se trabaja en un ambiente único como Windows 98 o 95. Esta ventaja también tiene su impedimento, ya que estos objetos no se pueden distribuir a otros sistemas operativos. Por lo que se recomienda a la persona que este interesada en desarrollar tecnologías alternas o similares a esta, que revise otras tecnologías como son los *Applets* de *Java*, si desea que su sistema funcione en sistemas como UNIX, LINUX, o MacOS.

Otro punto que hay que observar es la seguridad al distribuir objetos en una Intranet o Internet. Cada vez que se desarrolle una aplicación o un componente distribuida siempre debe ir acompañado de una ficha de autenticación con fecha de caducidad para que el usuario pueda confiar en los sistemas que vaya a utilizar.

Además de la seguridad también hay notar que el sistema es muy exigente con

respecto a recursos; conforme más servicios se conecten, mucho más carga en la red habrá. Para mayor referencia consulte al administrador de su Intranet o si uno es el administrador piense que la tecnología de Internet es usada por mas gente y entre más gente esté conectada solicitando información más lento su sistema funcionará.

Cuando se pensó en el tipo de base de datos que se iba a usar, se consideró que la seguridad de esta base no sería tan crítica como una base de datos bancaria, sin embrago, el sistema puede mejorarse usando el tipo de base de datos que más convenga, e inclusive puede establecer niveles de seguridad. En resumen, se pueden hacer las bases de datos tan complejas como se deseen, solo se tendrán que modificar los módulos para que acepten esas nuevas tecnologías de seguridad en bases de datos.

Al momento de planear el sistema de comunicación entre el ClienteOCX y el ServidorOCX, se consideró una técnica simple de comunicación. Esta consiste en enviar cadenas codificadas de información. A continuación se muestra el resultado que producen el cliente y el servidor:

El cliente cuando manda una solicitud viaja de la siguiente manera:

"01~01/06/1999~07/31/1999#13#1002~01/06/1999~07/31/1999"

donde esta cadena está en dos solicitudes (2 sensores) dividida cada solicitud en 3 grupos:

a) 2 caracteres: sensor

b) 10 caracteres: fecha de inicio de búsqueda y

c) 10 caracteres: fecha de fin de búsqueda.

d) '~' : que es un separador

e) '#13#10' : que es un espacio y un retorno de carro en código ASCII entre solicitud y solicitud

El servidor cuando manda la respuesta a la solicitud lo envía de la siguiente manera: "01~12/07/1999~10:32:37 a.m.~275.23#13#10"

donde esta cadena es un dato de un sensor con una fecha y hora específica, donde:

a) 2 caracteres: sensor

b) 10 caracteres: fecha dentro del rango especificado por el usuario

c) 13 caracteres: hora específica en que el dato fue capturado por el sistema

d) caracteres restantes posterior al separador: Dato específico del sensor

e) '~' : separador

f) '#13#10' : que es un espacio y un retorno de carro en código ASCII entre dato y dato

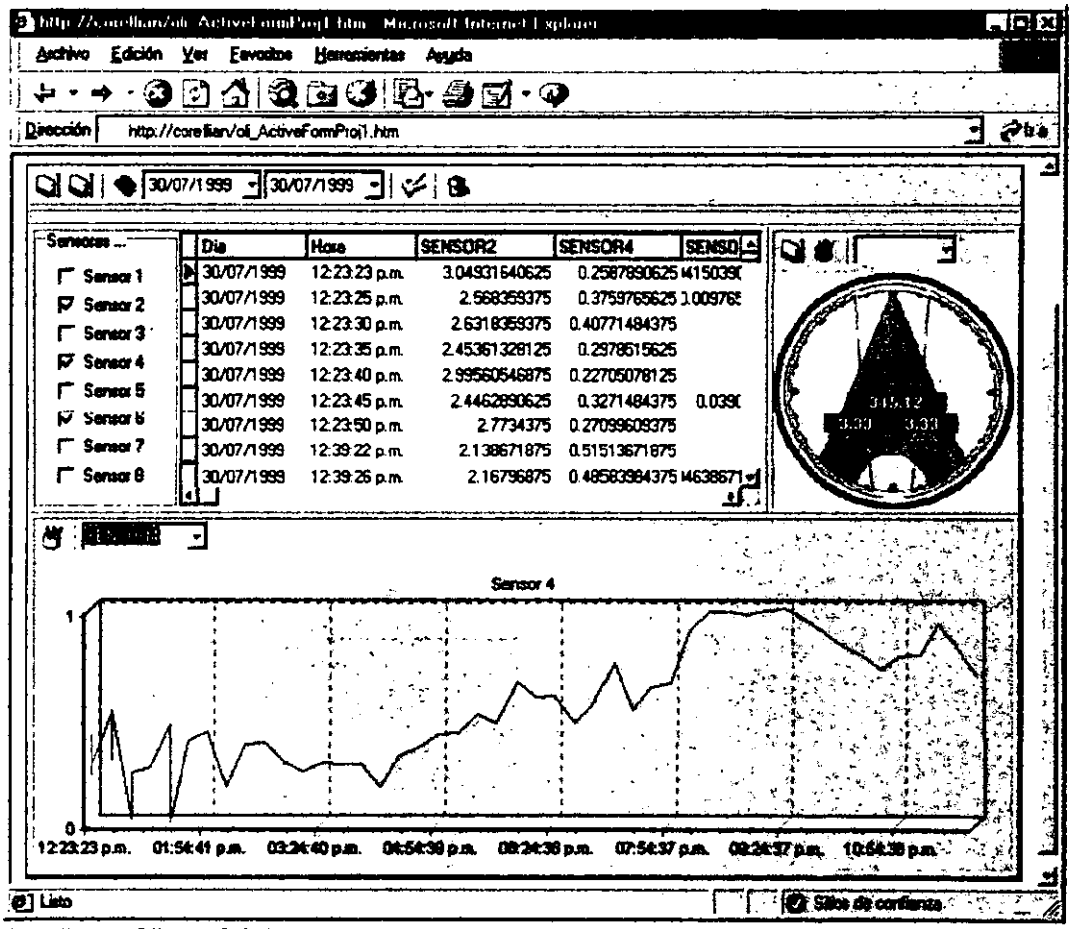
Por lo que se recomienda que el usuario que se encargue de continuar el trabajo use una nueva metodología de seguridad como el uso de encriptamiento de datos, si se piensa usar un medio de comunicación inseguro como Internet.

Con respecto al Internet Explorer 4.0 o superior, este tiene una propiedad que tienen todos los navegadores de internet, almacena en el disco duro las páginas y el contenido de estas en una zona llamada 'cache'. El problema que existe con esta *memoria caché* es que almacena también los componentes ActiveX por lo que se recomienda al usuario o al programador que quiera desarrollar nuevas versiones de estos módulos que cada vez que haga una solicitud de componente ActiveX, limpie esta memoria, consulte su manual de usuario para saber como hacerlo.

Etapa 3

El sistema es capaz de transportase de forma similar a los sistemas actuales de recuperación. Solo se recomienda que su transportación sea igual de cuidadosa que con los otros sistemas de adquisición de datos.

Una vez terminada la prueba de recuperación de datos, el componente presentó la siguiente pantalla con la información recuperada:



Detalle de ClienteOCX con datos recuperados.

Se recomienda que cuando se soliciten datos, sea el rango de fechas menor o igual a 7 dias y un sensor a la vez ya que entre más datos se soliciten mayor será el tiempo de espera.

Etapa 4

Se publicaron la documentación necesaria para la operación del sistema. Sin embargo, se recomienda al usuario que revise los Anexos para configurar u operar el software adicional que trabaja con este sistema.

Recomendaciones Finales

Se recomienda, como todo sistema computacional, un respaldo de la información en un lugar seguro. El directorio que contiene los datos se puede copiar como cualquier otro archivo dentro del ambiente Windows 98 o 95. Consulte su manual de usuario para mayores detalles.

No se recomienda instalar el sistema en el ambiente Windows NT, ya que el sistema de adquisición de datos, un error interno de configuración. Este problema ya fue notificado a National Instruments, Inc. y ya trabajan en una solución. Sin embargo el componente ClienteOCX si puede verse para hacer consultas.

También no se recomienda instalar el sistema en el ambiente Windows 2000 ya que la arquitectura de trabajo es igual a Windows NT, por lo que puede presentar el mismo tipo de error por parte del equipo.

Antes de instalar y configurar el sistema se recomienda tener conocimiento sobre manipulación del software secundario, de lo contrario, el sistema no podrá funcionar correctamente.

Conclusiones

Al realizar este proyecto que incluyen varios conceptos teóricos, se puede notar en primera instancia que no están relacionados entre si, pero es todo lo contrario. Es una agrupación de elementos que permiten un vínculo tecnológico de correlación que trabajando en conjunto y realizan una tarea que hasta hace algún tiempo sin la ayuda de la computadora sería imposible.

A pesar que algunas tecnologías aun son experimentales o de reciente estandarización como la distribución de objetos dentro de una Intranet, estos ya pueden aplicarse sin dificultad a una empresa tan grande como PEMEX. Al distribuir objetos permite la re-utilización de tecnología que tienen ya implantada, como es el caso de sensores y su Intranet.

Al terminar este proyecto, se pudo afirmar que el Internet será o en algunos casos ya es una herramienta de uso indispensable,, como el teléfono o el fax. Con esta forma de comunicación hace que un equipo tan sencillo como una estación meteorológica, pueda compartir su información, a quien lo desee, cuando se necesite en un periodo de tiempo reducido.

Para finalizar, basta decir que los objetivos planteados al principio de este documento se cumplieron con éxito, y que el sistema es funcional. A pesar que siempre habrá personal resistente a la evolución de los sistemas, y que rompe con esquemas anteriores de trabajo, la solución propuesta en esta tesis profesional es acertada. Y para el problema de recopilación de datos remota, este sistema fue aceptado por unanimidad por el Instituto Mexicano del Petróleo.

Anexos

ANEXO A - MICROSOFT® PERSONAL WEB SERVER©

Microsoft® Personal Web Server (PWS) 4.0 es la respuesta a sus necesidades de compartir su información personal y desarrollo de Web. PWS es un servidor Web personal que acelera y simplifica la configuración de un sitio Web, desde crear automáticamente una página principal personalizada hasta arrastrar y colocar publicaciones de documentos.

En una intranet corporativa, Personal Web Server puede usarse para compartir rápidamente documentos en su formato nativo o para convertir documentos al formato HTML y entonces usar PWS para compartirlos entre diferentes sistemas operativos.

Como Personal Web Server es compatible con Páginas Active Server, puede usarse como plataforma de desarrollo y pruebas para sitios Web. Es útil para crear un sitio en la oficina o en casa y probarlo mediante Personal Web Server antes de alojarlo en el servidor corporativo o en un proveedor de servicios Internet.

Instalar PWS

En Windows 98, PWS ofrece las tres opciones de instalación siguientes:

- Instalación mínima Los componentes mínimos necesarios para ejecutar PWS.
- Instalación típica Las opciones mínimas, con funcionalidades y documentación adicionales.
- Instalación personalizada Muestra todos los componentes posibles como opciones, con todas las opciones incluidas en las instalaciones mínima y típica preseleccionadas.

Para instalar Personal Web Server

- En el CD-ROM diríjase al directorio: `\PWS`
- De doble-clic al programa: `Setup.exe`
- Siga los pasos
- Reinicie el equipo siguiendo el proceso de instalación. Si está realizando la instalación a través de Internet, seleccione la plataforma y siga las instrucciones de la pantalla.
- Para utilizar PWS en un equipo sin conexión a una red, debe cambiar a Hosts el nombre del archivo Hosts.sam ubicado en el directorio \Windows.

ANEXO B - NI-DAQ 6.5.2©

NI-DAQ es un programa que controla el hardware adicional para hardware o instrumentos de National Instrument®. Siga detalladamente los pasos para su instalación, configuración y puesta en marcha.

- En el CD-ROM diríjase al directorio: `\NIDAQ\BASE`
- De doble-clic al programa: `install.exe`
- Siga los pasos
- Reinicie el equipo siguiendo el proceso de instalación.

Nota: Para mayor documentación sobre este programa vea el archivo: leame.htm en el diectorio \NIDAQ

Errores con NI-DAQ

Consulte la guía que proporciona NI-DAQ para corregir errores. O consulte la página http://www.natinst.com para una consulta personalizada del error que marca solo este programa.

Configuración del chasis SCXI-1000 y de la tarjeta SCXI-1200

Para configurar el chasis y la tarjeta revise el siguiente documento:

- En el CD-ROM diríjase al directorio `\Documentos\`
- De doble-clic al documento `lpt1200.htm`

ANEXO C - BORLAND DATABASE ENGINE©

Borland DataBase Engine (BDE) es un programa que controla las bases de datos que manejan el formato `*.db`. Para instalarlo, solo debe seguir los siguientes pasos:

- En el CD-ROM vaya al directorio: `\BDE`
- De doble-click al programa: `install.exe`
- Siga los pasos
- Reinicie el equipo siguiendo el proceso de instalación.

ANEXO D - DCOM

DCOM© es el software necesario que se usará para la transmisión de datos en la red de trabajo, Intranet o Internet.

Según sea su sistema operativo deberá ocupar el DCOM necesario:

Windows 95©

- Vaya al directorio: \DCOM
- De doble clic en el programa: DCOM95.EXE
- Siga los pasos
- Reinicie el equipo siguiendo el proceso de instalación.

Windows 98©

- Vaya al directorio: \DCOM
- De doble clic en el programa: DCOM98.EXE
- Siga los pasos

- Reinicie el equipo siguiendo el proceso de instalación.

ANEXO E - MICROSOFT INTERNET EXPLORER 5.0

Este programa o "browser" comercial será el que permitirá usar el módulo CLIENTESAD para ello requiere la siguiente configuración:

- Elija "Opciones de Internet" del menu "Herramientas".
- En la carpeta de "Seguridad" elija "Intranet Local"
- De clic en "Sitios" >> "Opciones Avanzadas"
- Agrega: `http://nombre_servidor`

  Donde "`nombre_servidor`" es el nombre de tu Servidor Web Personal (Personal Web Server - ver Anexo A).

ANEXO F - SOPORTE TÉCNICO

Póngase en contacto con el Servicio Técnico si tiene alguna pregunta específica sobre cómo utilizar el programa que ha adquirido.

Antes de llamar al Servicio Técnico, por favor rellene la hoja de Información del Sistema que encontrará al final de esta sección. Tener esta información preparada, le permitirá ahorrar tiempo y conseguir una mejor asistencia.

Este servicio es exclusivo para usuarios que previamente hayan llenado y enviado la tarjeta de registro.

Servicio OnLine

e-mail

omota@www.imp.mx

omota@mpsnet.com.mx

Pagina Web

http://corellian

ANEXO G - ComponentWorks 2.0 [20]

http://www.ni.com

ANEXO H - Chasis SCXI-1000 [20]

http://www.ni.com

ANEXO I - Tarjeta DAQ SCXI-1200 [20]

http://www.ni.com

ANEXO J - NI_DAQ 6.2 [20]

http://www.ni.com

ANEXO K - CODIGO INTERFAZCHAS

```
unit U_Interfazchas;
{ Autor: Oliver Mota Perez; Copyright 1999}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  CWUIControlsLib_TLB, OleCtrls, CWDAQControlsLib_TLB, ComCtrls, StdCtrls,
  Db, DBTables, ExtCtrls,fmxutils, Menus, Math;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    StatusBar1: TStatusBar;
    CWButton1: TCWButton;
    CWButton2: TCWButton;
    CWButton3: TCWButton;
    CWButton4: TCWButton;
    CWButton5: TCWButton;
    CWButton6: TCWButton;
    CWButton7: TCWButton;
    CWButton8: TCWButton;
    Table1: TTable;
    Timer1: TTimer;
    CWAIPoint1: TCWAIPoint;
    Table1ID_Sensor: TStringField;
    Table1Canal: TFloatField;
    Table1Nombre: TStringField;
    Table1Modelo: TStringField;
    Table1Activo: TBooleanField;
    Table1Vol_Max: TFloatField;
    Table1Vol_Min: TFloatField;
    Table1Rango_Max: TFloatField;
    Table1Rango_Min: TFloatField;
    T_Canal1: TTable;
    T_Canal2: TTable;
    T_Canal3: TTable;
    T_Canal4: TTable;
    T_Canal5: TTable;
    T_Canal6: TTable;
    T_Canal7: TTable;
    T_Canal8: TTable;
    CWButton9: TCWButton;
    LabelStandby: TLabel;
    LabelActivo: TLabel;
    Table1Muestreo: TFloatField;
    PopupMenu1: TPopupMenu;
    N1Minuto1: TMenuItem;
    N5Segs1: TMenuItem;
    N15Segs1: TMenuItem;
    N30Segs1: TMenuItem;
    N5Minutos1: TMenuItem;
    N15Minutos1: TMenuItem;
    N1Hora1: TMenuItem;
    N1: TMenuItem;
    N30Minutos1: TMenuItem;
```

```
    N2: TMenuItem;
    Instantaneo1: TMenuItem;
    N3: TMenuItem;
    Label5: TLabel;
    Image3: TImage;
    PopupMenu2: TPopupMenu;
    Ejecutar1: TMenuItem;
    Image2: TImage;
    CWAI1: TCWAI;
    Memo1: TMemo;
    CWDIO1: TCWDIO;
{}  function AnalizaVelo(dummy:boolean):real;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure CWButton9Click(Sender: TObject);
    procedure N5Segs1Click(Sender: TObject);
    procedure Image3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image3MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormDblClick(Sender: TObject);
    procedure Ejecutar1Click(Sender: TObject);
    procedure CWAI1AcquiredData(Sender: TObject; var ScaledData,
      BinaryCodes: OleVariant);
  private
    { Private declarations }
  public
    procedure ActivaCanales;
    { Public declarations }
  end;

var
  Form1: TForm1;
  VeloVientoData : real;

implementation

{$R *.DFM}

{yc}
function TForm1.AnalizaVelo(dummy:boolean):real;
var
  I,Revol : word;
  RevolReal : real;
  YY : string[2];
begin
   Revol := 0;

   for I := 0 to 999 do
   begin
     YY := memo1.Lines[I];
     if I <> 0 then
     begin
       if (YY = '0') and (memo1.Lines[I-1] = '5') then Revol := Revol + 1;
       if (yy = '5') and (memo1.Lines[I-1] = '0') then Revol := Revol + 1;
```

```
      end;
    end;{for}
    revolreal := Revol div 4;
    result := (60*revolreal)/16.767+1;
end;

{computadora}
procedure Tform1.ActivaCanales;
{Activa los sensores necesarios y cierra los que no se usen}
begin
  T_Canal1.Close;
  T_Canal2.Close;
  T_Canal3.Close;
  T_Canal4.Close;
  T_Canal5.Close;
  T_Canal6.Close;
  T_Canal7.Close;
  T_Canal8.Close;
  while not Table1.Eof do
  begin
    if Table1.fields[4].value then
      case Table1.Fields[1].value of
        1: begin CWButton1.Value := true; T_Canal1.Open; end;
        2: begin CWButton2.Value := true; T_Canal2.Open; end;
        3: begin CWButton3.Value := true; T_Canal3.Open; end;
        4: begin CWButton4.Value := true; T_Canal4.Open; end;
        5: begin CWButton5.Value := true; T_Canal5.Open; end;
        6: begin CWButton6.Value := true; T_Canal6.Open; end;
        7: begin CWButton7.Value := true; T_Canal7.Open; end;
        8: begin CWButton8.Value := true; T_Canal8.Open; end;
      end
    else
      case Table1.Fields[1].value of
        1: CWButton1.Value := false;
        2: CWButton2.Value := false;
        3: CWButton3.Value := false;
        4: CWButton4.Value := false;
        5: CWButton5.Value := false;
        6: CWButton6.Value := false;
        7: CWButton7.Value := false;
        8: CWButton8.Value := false;
      end;
    Table1.Next;
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
{ recupera datos del chasis y los pone en las bases de datos`}
var
   volts : OleVariant;
   voltio : real;
   error : integer;
begin
    StatusBar1.Panels[8].text := 'Recuperando...';
    {canal_1}
    if CWButton2.value then begin
      CWAIPoint1.Channels.RemoveAll;
```

```
      CWAIPoint1.Channels.Add('1',0,0,cwaiDefaultInputMode,
      cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := volts;
      Statusbar1.panels[1].text := '    '+floattostr(voltio);
      T_Canal2.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_2}
    if CWButton3.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('2', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := volts;
      Statusbar1.panels[2].text := '    '+floattostr(voltio);
      T_Canal3.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_3}
    if CWButton4.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('3', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := Volts;
      Statusbar1.panels[3].text := '    '+floattostr(voltio);
      T_Canal4.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_4}
    if CWButton5.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('4', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := volts;
      Statusbar1.panels[4].text := '    '+floattostr(voltio);
      T_Canal5.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_5}
    if CWButton6.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('5', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := Volts;
      Statusbar1.panels[5].text := '    '+floattostr(voltio);
      T_Canal6.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_6}
    if CWButton7.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('6', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := volts;
      Statusbar1.panels[6].text := '    '+floattostr(voltio);
      T_Canal7.appendRecord([nil,Date,Time,voltio]);
    end;
```

```
    {canal_7}
    if CWButton8.value then begin
      CWAIPoint1.Channels.RemoveAll;
      CWAIPoint1.Channels.Add('7', 0, 0, cwaiDefaultInputMode,
cwaiDefaultCoupling);
      CWAIPoint1.SingleRead(volts,1000);
      voltio := Volts;
      Statusbar1.panels[7].text :=  '    '+floattostr(voltio);
      T_Canal8.appendRecord([nil,Date,Time,voltio]);
    end;
    {canal_0}
    if CWButton1.value then begin
      Error := CWDIO1.SingleWrite(IntPower(2,7));
      if error <> 0 then Form1.caption := intToStr(Error);
      Try
        CWAI1.Configure;
        CWAI1.Start;
        {el analisis y colocacion en BD esta en el metodo AcquiredData del
CWAI1}
      Finally
        VeloVientoData := analizaVelo(true);
        Statusbar1.panels[0].text := '    '+FloatToStr(VeloVientoData);
        T_Canal1.appendRecord([nil,Date,Time,VeloVientoData]);
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Table1.Open;
  Table1.first;
  ActivaCanales;
  Table1.Close;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  T_canal1.Close;
  T_canal2.Close;
  T_canal3.Close;
  T_canal4.Close;
  T_canal5.Close;
  T_canal6.Close;
  T_canal7.Close;
  T_canal8.Close;
end;

procedure TForm1.CWButton9Click(Sender: TObject);
begin
  if (CWButton9.value) then
  begin
    {Como el valor de muestreo es igual para todos los sensores se toma el
     muestreo del primer sensor}
    Table1.Close;
    Table1.Open;
    Table1.first;
    Timer1.interval := Table1Muestreo.AsInteger*1000;
```

```
    Table1.Close;
    Timer1.Enabled := true;
    StatusBar1.Panels[8].text := 'Recuperando...';
    LabelStandBy.font.Color := ClMaroon;
    LabelActivo.font.color := clLime;
  end
  else
  begin
    Timer1.Enabled := false;
    StatusBar1.Panels[8].text := 'Stand by ...';
    LabelStandBy.font.Color := ClRed;
    LabelActivo.font.color := clgreen;
  end;
end;

procedure TForm1.N5Segs1Click(Sender: TObject);
{se establece el valor de muestreo a todos los sensores.
 Todos los sensores tienen el mismo valor de muestreo}
var
  J : word;
begin
  if not (CWButton9.Value) then
  begin
    j := 5;
    if sender = N5Segs1 then begin j := 5;
Image2.Picture.LoadFromFile('t5seg.bmp') end;
    if sender = N15Segs1 then begin j := 15;
Image2.Picture.LoadFromFile('t15seg.bmp') end;
    if sender = N30Segs1    then begin j := 30;
Image2.Picture.LoadFromFile('t30seg.bmp') end;
    if sender = N1Minuto1   then begin j := 60;
Image2.Picture.LoadFromFile('t1Min.bmp') end;
    if sender = N5Minutos1  then begin j := 300;
Image2.Picture.LoadFromFile('t5Min.bmp') end;
    if sender = N15Minutos1 then begin j := 900;
Image2.Picture.LoadFromFile('t15Min.bmp') end;
    if sender = N30Minutos1 then begin j := 1800;
Image2.Picture.LoadFromFile('t30Min.bmp') end;
    if sender = N1Hora1     then begin j := 3600;
Image2.Picture.LoadFromFile('t60Min.bmp') end;
    if sender = Instantaneo1 then begin j := 1;
Image2.Picture.LoadFromFile('t00seg.bmp') end;

    Table1.Close; Table1.Open;
    Table1.First;
    while not (table1.Eof) do
    begin
      Table1.Edit;
      Table1Muestreo.AsInteger := j;
      Table1.post;
      Table1.Next;
    end; {while}
    table1.close;
  end {if}
  else
    MessageDlg('Pon en "StandBy" el sistema para efectuar
```

```
Cambios',mtwarning,[mbOK],0);
end;

procedure TForm1.Image3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  Image3.Picture.LoadFromFile('Setupdown.bmp');
end;

procedure TForm1.Image3MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  Image3.Picture.LoadFromFile('Setup.bmp');
end;

procedure TForm1.FormDblClick(Sender: TObject);
begin
   Application.Minimize;
end;

procedure TForm1.Ejecutar1Click(Sender: TObject);
begin
  executefile('P_interfaz.exe','','c:\scada\Interface',SW_SHOW);
end;

procedure TForm1.CWAI1AcquiredData(Sender: TObject; var ScaledData,
  BinaryCodes: OleVariant);
var
  I : integer;
begin
  Memo1.Lines.Clear;
  for I := 0 to 999 do
    Memo1.Lines.add(IntToStr(ScaledData[I]));
  CWAI1.Stop;
end;

end.
```

ANEXO K - CODIGO INTERFAZ

```
unit U_principal;
{ Autor: Oliver Mota Perez; Copyright 1999}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, ImgList, StdActns, ActnList, Menus, ToolWin, StdCtrls,
  Buttons, OleCtrls, Db, DBActns, DBTables, Mask, DBCtrls, Grids, DBGrids,
  U_ModuloDatos, CWUIControlsLib_TLB;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    StatusBar1: TStatusBar;
    Timer1: TTimer;
    MainMenu1: TMainMenu;
    Archivo1: TMenuItem;
    Editar1: TMenuItem;
    Ver1: TMenuItem;
    Ayuda1: TMenuItem;
    Acercade1: TMenuItem;
    Cut1: TMenuItem;
    Copy1: TMenuItem;
    Paste1: TMenuItem;
    ImageList1: TImageList;
    Panel2: TPanel;
    Splitter1: TSplitter;
    Panel3: TPanel;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet3: TTabSheet;
    Panel4: TPanel;
    Arbol1: TMenuItem;
    ListView2: TListView;
    Salir1: TMenuItem;
    TreeView1: TTreeView;
    GroupBox2: TGroupBox;
    ImageList2: TImageList;
    ActionList1: TActionList;
    EditCopy1: TEditCopy;
    EditCut1: TEditCut;
    EditPaste1: TEditPaste;
    Datos: TTabSheet;
    PopupMenu2: TPopupMenu;
    PopupMenu3: TPopupMenu;
    DBGrid1: TDBGrid;
    Panel5: TPanel;
    DBNavigator1: TDBNavigator;
    Button1: TButton;
    DateTimePicker1: TDateTimePicker;
    DateTimePicker2: TDateTimePicker;
    ToolBar2: TToolBar;
    BotonConectar: TToolButton;
    propiedades1: TMenuItem;
```

```
    CWGraph1: TCWGraph;
    Panel6: TPanel;
    CWButton1: TCWButton;
    Label1: TLabel;
    Timer2: TTimer;
    N5: TMenuItem;
    Eliminar1: TMenuItem;
    ImageList3: TImageList;
    propiedades2: TMenuItem;
    N1: TMenuItem;
    Eliminar2: TMenuItem;
    N3: TMenuItem;
    DatosHistoricos1: TMenuItem;
    DatosInstantaneos1: TMenuItem;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Panel3UnDock(Sender: TObject; Client: TControl;
      NewTarget: TWinControl; var Allow: Boolean);
    procedure Panel3DockDrop(Sender: TObject; Source: TDragDockObject; X,
      Y: Integer);
    procedure Arbol1Click(Sender: TObject);
    procedure Salir1Click(Sender: TObject);
    procedure TreeView1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure BotonConectarClick(Sender: TObject);
    procedure ListView2SelectItem(Sender: TObject; Item: TListItem;
      Selected: Boolean);
    procedure propiedades1Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure CWButton1Click(Sender: TObject);
    procedure Eliminar1Click(Sender: TObject);
    procedure DatosHistoricos1Click(Sender: TObject);
    procedure DatosInstantaneos1Click(Sender: TObject);
    procedure Acercade1Click(Sender: TObject);
    procedure propiedades2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Global_ScaledData,Global_Respaldo : variant;

implementation

uses U_propiedades, Uayuda;

{$R *.DFM}

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  StatusBar1.Panels[2].text := TimetoStr(time);
  StatusBar1.Panels[1].text := DatetoStr(date);
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   if Panel4.ManualDock(Panel3,Panel4,alnone) then
     panel1.caption :='';
   GroupBox2.Caption := '';
   Button1.Enabled := false;
end;

procedure TForm1.Panel3UnDock(Sender: TObject; Client: TControl;
  NewTarget: TWinControl; var Allow: Boolean);
begin
  Panel3.Width:=0;
end;

procedure TForm1.Panel3DockDrop(Sender: TObject; Source: TDragDockObject;
  X, Y: Integer);
begin
  Panel3.autosize := true;
  Panel3.autosize := false;
end;

procedure TForm1.Arbol1Click(Sender: TObject);
begin
  Panel4.Show;
  Panel4.width:=180;
  Panel3.autosize := true;
  Panel3.autosize := false;
end;

procedure TForm1.Salir1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.TreeView1Click(Sender: TObject);
begin
  Button1.Enabled := false;
  CWButton1.Enabled := false;
  TreeView1.PopupMenu := nil;
  ModuloDatos1.TB_Datos.Close;
  if treeView1.Items.Item[0].selected then
    BotonConectar.click;

  if TreeView1.Selected.Parent.Index = 1 then
  begin
    PageControl1.ActivePage := TabSheet1;
    TreeView1.PopupMenu := PopupMenu2;
    case TreeView1.Selected.Index of
      0: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S1.db';
          ModuloDatos1.TB_Datos.Open;
         end;
      1: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
```

```
          ModuloDatos1.TB_Datos.TableName := 'Dato_S2.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      2: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S3.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      3: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S4.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      4: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S5.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      5: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S6.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      6: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S7.db';
          ModuloDatos1.TB_Datos.Open;
        end;
      7: begin
          GroupBox2.Caption := TreeView1.Selected.Text;
          Button1.Enabled := true; CWButton1.Enabled := true;
          ModuloDatos1.TB_Datos.TableName := 'Dato_S8.db';
          ModuloDatos1.TB_Datos.Open;
        end;
    end; {case}
  end; {if pariente 2}

  if TreeView1.Selected.Parent.Index = 1 then
  begin
    TreeView1.PopupMenu := PopupMenu2;
    case TreeView1.Selected.Index of
      0: {Elegir sensor con ttable};
      1: {Elegir sensor con ttable};
    end; {case}
  end; {if}
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ModuloDatos1.TB_Datos.Disablecontrols;
  Try
    ModuloDatos1.TB_Datos.IndexName := 'Sec_Fecha';
```

```
    ModuloDatos1.TB_Datos.SetRangeStart;
    ModuloDatos1.TB_Datos.Fields[1].value := DateTimePicker1.date;

    ModuloDatos1.TB_Datos.SetRangeEnd;
    ModuloDatos1.TB_Datos.Fields[1].value := DateTimePicker2.date;

    ModuloDatos1.TB_Datos.ApplyRange;
  finally
    ModuloDatos1.TB_Datos.EnableControls;
  end;
end;

procedure TForm1.BotonConectarClick(Sender: TObject);
{ Leer de la base de datos sensor.db la informacion de los sensores y
mostrarla
  en la pantalla del usuario. }
var
 i : word;
begin
  ModuloDatos1.TB_Sensores.close;
  ModuloDatos1.DS_Sensores.Enabled := false;
  try
    ModuloDatos1.TB_Sensores.open;
    ModuloDatos1.TB_Sensores.first;
    i := 2;
    ListView2.Items.Clear;
    while not ModuloDatos1.TB_Sensores.eof do
    begin
      TreeView1.Items[i].text := ModuloDatos1.TB_SensoresNombre.asString;
      ListView2.Items.Add;
      ListView2.Items.item[i-2].caption :=
      ModuloDatos1.TB_Sensores.Fields[2].asString;
      if (ModuloDatos1.TB_SensoresActivo.value) then
        ListView2.Items.item[i-2].subitems.Add('Activo')
      else
        ListView2.Items.item[i-2].subitems.Add('Inactivo');
      ListView2.Items.item[i-2].subitems.Add(
      ModuloDatos1.TB_SensoresRango_Min.asString +' - '+
      ModuloDatos1.TB_SensoresRango_Max.asString);

      ListView2.Items.item[i-2].subitems.Add(
      ModuloDatos1.TB_SensoresUnidades.asstring);
      ListView2.Items.item[i-2].subitems.Add('Sensor Analogico');
      ListView2.Items.item[i-2].subitems.Add(
      ModuloDatos1.TB_SensoresVol_Min.asString + 'V' +' -
      +'+ModuloDatos1.TB_SensoresVol_Max.asString + 'V');
      i := i + 1;
      ModuloDatos1.TB_Sensores.next;
    end; {while}
  finally
    ModuloDatos1.TB_Sensores.close;
  end;
end;

procedure TForm1.ListView2SelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
```

```
  if selected then
    listview2.popupmenu := popupmenu3
  else
    listview2.popupmenu := nil;
end;

procedure TForm1.propiedades1Click(Sender: TObject);
begin
  ModuloDatos1.TB_Sensores.close;
  ModuloDatos1.TB_Sensores.open;
  {analiza que item fue elegido}
  if ListView2.Selected.Index = 0 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 1 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.Next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 2 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 3 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 4 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 5 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 6 then
```

```
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  if ListView2.Selected.Index = 7 then
  begin
    ModuloDatos1.TB_Sensores.first;
    ModuloDatos1.DS_Sensores.Enabled := true;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;ModuloDatos1.TB_Sensores.next;
    ModuloDatos1.TB_Sensores.next;
    Form2.Showmodal;
  end;
  BotonConectar.click; {vuelve a reconstruir la pantalla ListView2}
  ModuloDatos1.TB_Sensores.close; {cierra la TB_Sensores}
end;

procedure TForm1.Timer2Timer(Sender: TObject);
var
  I : word;
begin
  if CWButton1.value = true then
  begin
    ModuloDatos1.TB_Datos.Close;
    ModuloDatos1.TB_Datos.Open;
    ModuloDatos1.TB_Datos.Last;
    for I := 0 to 68 do
    begin
      Global_Respaldo[I]:=Global_ScaledData[I+1];
      {Global_ScaledData[I]:=Global_Respaldo[I];}
    end;
    for I := 0 to 68 do
    begin
      Global_ScaledData[I]:=Global_Respaldo[I];
    end;
    Global_ScaledData[69] := ModuloDatos1.TB_Datos.Fields[3].Asfloat;
    CWGraph1.PlotY(Global_ScaledData,0,1,TRUE);
  end;
end;

procedure TForm1.CWButton1Click(Sender: TObject);
var
  I : word;
begin
  Global_ScaledData := VarArrayCreate([0,69], varVariant);
  Global_Respaldo := VarArrayCreate([0,68], varVariant);
  for I := 0 to 69 do Global_ScaledData[I] := 0.0;
  for I := 0 to 68 do Global_Respaldo[I] := 0.0;
end;

procedure TForm1.DatosHistoricos1Click(Sender: TObject);
```

```
begin
  PageControl1.ActivePage := Datos;
end;

procedure TForm1.DatosInstantaneos1Click(Sender: TObject);
begin
  PageControl1.ActivePage := TabSheet3;
end;

procedure TForm1.Acercadel1Click(Sender: TObject);
begin
  FAyudaOnline.showmodal;
end;

procedure TForm1.propiedades2Click(Sender: TObject);
begin
  PageControl1.ActivePage := Datos;
end;

end.
```

ANEXO K - CODIGO ServidorOCX

```
unit UServidor;

{ Antes de ejecutar tu aplicacion asegura que la fecha de tu computadora
  trabaje con 4 digitos el a±o.
  1)  En 'Configuracion Regional' del 'Panel de Control' da clic en la
lengueta
  'Fecha' y en la parte de 'Formato de Fecha Corta' asegura que tenga
  el formato:   dd/MM/aaaa
  y asegura que tenga el separador:   /
  2)  En 'Configuracion Regional' del 'Panel de Control' da clic en la
lengueta
  'Hora' y en la parte de 'Formato de Hora' asegura que tenga
  el formato:    hh:mm:ss tt
  separador de hora:    :
  simbolo a.m.:   a.m.
  simbolo p.m.:   p.m.

  Da clic en Aceptar.
  Autor: Oliver Mota Perez; Copyright 1999}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Db, DBTables, ScktComp, StdCtrls, ComCtrls, Grids, DBGrids;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    ServerSocket1: TServerSocket;
    MemoRecibe: TMemo;
    Query1: TQuery;
    Button1: TButton;
    Button2: TButton;
    StatusBar1: TStatusBar;
    procedure ServerSocket1ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure AnalisaPeticion(peticion:string);
    procedure BuscaDatosYEnvia(Sensor,Inicio,Fin,BaseDeDatos:string);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
```

```
procedure TForm1.BuscaDatosYEnvia(Sensor,Inicio,Fin,BaseDeDatos:string);
{Busca los datos y los envia al cliente conforme los encuentra}
var
  tempo : string;
  consulta : string;
  mensaje : string;
begin
  Consulta := '';
  Consulta := 'SELECT Fecha, Hora, Dato FROM "'+ BaseDeDatos +'" Dato_s1 WHERE
Fecha BETWEEN "'+Inicio+'" AND "'+Fin+'"';
{               SELECT Fecha, Hora, Dato FROM "Dato_S1.DB"        Dato_s1 WHERE
Fecha BETWEEN "06/16/1999" AND "06/16/1999"
}
  Query1.Close;
  Query1.SQL.Clear;
  Query1.SQL.Add(Consulta);
  Query1.Open;
  Query1.First;
  while not Query1.Eof do
  begin
    Tempo := Sensor+'~'+Query1.FieldByName('Fecha').AsString + '~';
    Tempo := Tempo + Query1.FieldByName('Hora').AsString + '~';
    Tempo := Tempo + Query1.FieldByName('Dato').AsString;
    Mensaje:= Tempo + #13#10;
    ServerSocket1.Socket.Connections[0].SendText(Mensaje);
    Query1.Next;
  end;
  Query1.Close;
end;

procedure Tform1.AnalisaPeticion(peticion:string);
var
  I : Word;
  Cadena: string[100];
  ID : string[2];
  FechaInicio,FechaFin :string[10];
  diaI,MesI,AnioI,diaF,MesF,AnioF : string[5];

begin
{01~01/06/1999~07/31/1999
02~01/06/1999~07/31/1999
03~01/06/1999~07/31/1999
04~01/06/1999~07/31/1999
}
    I := 0;
    While not (I > MemoRecibe.Lines.Count -1) do
    begin
      ID := ''; Cadena := ''; FechaInicio :=''; FechaFin :='';
      diaI:= '';MesI:= '';AnioI:= '';diaF:= '';MesF:= '';AnioF:= '';
      Cadena := MemoRecibe.Lines[I];
      ID := Cadena[1]+cadena[2];

      DiaI:= Cadena[4]+cadena[5]; MesI:= Cadena[7]+Cadena[8];
      AnioI:=Cadena[10]+Cadena[11]+Cadena[12]+Cadena[13];
      DiaF:= Cadena[15]+cadena[16]; MesF:= Cadena[18]+Cadena[19];
      AnioF:=Cadena[21]+Cadena[22]+Cadena[23]+Cadena[24];
```

```
{--analisis y envio}
if ID = '01' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S1.DB');
end;
{--}
if ID = '02' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S2.DB');
end;
{--}
if ID = '03' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S3.DB');
end;
{--}
if ID = '04' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S4.DB');
end;
{--}
if ID = '05' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S5.DB');
end;
{--}
if ID = '06' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S6.DB');
end;
{--}
if ID = '07' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S7.DB');
end;
{--}
if ID = '08' then
begin
  FechaInicio := MesI +'/'+ DiaI +'/'+ AnioI;
  FechaFin := MesF +'/'+ DiaF +'/'+ AnioF;
  BuscaDatosYEnvia(ID,FechaInicio,FechaFin,'Dato_S8.DB');
end;
```

```
      {--fin de analisis y envio}
      I := I + 1;

    end;{while}
    ServerSocket1.Socket.Connections[0].SendText('FIN');
end;

procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  RecepcionTemporal : string;
begin
  MemoRecibe.Lines.clear;
  RecepcionTemporal := Socket.ReceiveText;
  {el cliente pide: '01~12/07/1999~12/07/1999'}
  MemoRecibe.Lines.Add(RecepcionTemporal);
  AnalisaPeticion('Dummy');
end; {procedure}

procedure TForm1.Button1Click(Sender: TObject);
begin
  ServerSocket1.Active := true;
  StatusBar1.Panels[0].Text := 'Activado y Trabajando...';
  Button1.enabled := false;
  Button2.Enabled := true;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  ServerSocket1.Active := false;
  StatusBar1.Panels[0].Text := 'Desactivado.';
  Button1.enabled := true;
  Button2.Enabled := false;
end;

end.
```

ANEXO L - CODIGO ClienteOCX

```
unit Oli_ActiveFormImpl1;

{ Autor: Oliver Mota Perez; Copyright 1999}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ActiveX, AxCtrls, Oli_ActiveFormProj1_TLB, ScktComp, Db, DBTables,
  ExtCtrls, StdCtrls, OleCtrls, CW3DGraphLib_TLB, Grids, DBGrids, ComCtrls,
  ToolWin, CWUIControlsLib_TLB, jpeg, ImgList, TeEngine, Series, TeeProcs,
  Chart, DBChart, Menus;

type
  TOli_ActiveFormX = class(TActiveForm, IOli_ActiveFormX)
    Panel4: TPanel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    ToolBar3: TToolBar;
    ToolButton7: TToolButton;
    ToolButton8: TToolButton;
    ToolButton9: TToolButton;
    ToolButton10: TToolButton;
    DateTimePicker1: TDateTimePicker;
    DateTimePicker2: TDateTimePicker;
    Query1: TQuery;
    Table1: TTable;
    DataSource1: TDataSource;
    ClientSocket1: TClientSocket;
    Panel1: TPanel;
    Panel2: TPanel;
    DBGrid1: TDBGrid;
    Memo2: TMemo;
    GroupBox1: TGroupBox;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    CheckBox6: TCheckBox;
    CheckBox7: TCheckBox;
    CheckBox8: TCheckBox;
    Memo1: TMemo;
    ImageList1: TImageList;
    ToolB_Vaciar: TToolButton;
    ToolButton12: TToolButton;
    ProgressBar1: TProgressBar;
    Query2: TQuery;
    ToolButton3: TToolButton;
    ToolButton11: TToolButton;
    Panel6: TPanel;
    Image1: TImage;
    ToolBar2: TToolBar;
    ToolButton4: TToolButton;
```

```
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ComboBox3: TComboBox;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Panel5: TPanel;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ComboS_1: TComboBox;
    DBChart1: TDBChart;
    PopupMenu1: TPopupMenu;
    N3D1: TMenuItem;
    N2: TMenuItem;
    IncluirenReporte1: TMenuItem;
    Etiquetas1: TMenuItem;
    Series1: TFastLineSeries;
{}  procedure Manda_Peticion;
{}  procedure ColocaEnArchivo_DB(Dia_Real,Hora_Real:TDateTime; Sensor:string;
DatoReal:Real);
    procedure ToolButton7Click(Sender: TObject);
    procedure ToolButton8Click(Sender: TObject);
    procedure ToolButton10Click(Sender: TObject);
    procedure ClientSocket1Error(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure CheckBox3Click(Sender: TObject);
    procedure CheckBox4Click(Sender: TObject);
    procedure CheckBox5Click(Sender: TObject);
    procedure CheckBox6Click(Sender: TObject);
    procedure CheckBox7Click(Sender: TObject);
    procedure CheckBox8Click(Sender: TObject);
    procedure ToolButton1Click(Sender: TObject);
    procedure ToolButton4Click(Sender: TObject);
    procedure ToolB_VaciarClick(Sender: TObject);
    procedure ClientSocket1Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure Memo2Change(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure N3D1Click(Sender: TObject);
    procedure IncluirenReporte1Click(Sender: TObject);
    procedure Etiquetas1Click(Sender: TObject);
    procedure ToolButton11Click(Sender: TObject);
  private
    { Private declarations }
    FEvents: IOli_ActiveFormXEvents;
    procedure ActivateEvent(Sender: TObject);
    procedure ClickEvent(Sender: TObject);
    procedure CreateEvent(Sender: TObject);
    procedure DblClickEvent(Sender: TObject);
    procedure DeactivateEvent(Sender: TObject);
    procedure DestroyEvent(Sender: TObject);
    procedure KeyPressEvent(Sender: TObject; var Key: Char);
    procedure PaintEvent(Sender: TObject);
  protected
```

```
    { Protected declarations }
    procedure DefinePropertyPages(DefinePropertyPage: TDefinePropertyPage);
override;
    procedure EventSinkChanged(const EventSink: IUnknown); override;
    while Get_Active: WordBool; safecall;
    while Get_AutoScroll: WordBool; safecall;
    while Get_AutoSize: WordBool; safecall;
    while Get_AxBorderStyle: TxActiveFormBorderStyle; safecall;
    while Get_BiDiMode: TxBiDiMode; safecall;
    while Get_Caption: WideString; safecall;
    while Get_Color: OLE_COLOR; safecall;
    while Get_Cursor: Smallint; safecall;
    while Get_DoubleBuffered: WordBool; safecall;
    while Get_DropTarget: WordBool; safecall;
    while Get_Enabled: WordBool; safecall;
    while Get_Font: IFontDisp; safecall;
    while Get_HelpFile: WideString; safecall;
    while Get_KeyPreview: WordBool; safecall;
    while Get_PixelsPerInch: Integer; safecall;
    while Get_PrintScale: TxPrintScale; safecall;
    while Get_Scaled: WordBool; safecall;
    while Get_Visible: WordBool; safecall;
    procedure _Set_Font(const Value: IFontDisp); safecall;
    procedure Set_AutoScroll(Value: WordBool); safecall;
    procedure Set_AutoSize(Value: WordBool); safecall;
    procedure Set_AxBorderStyle(Value: TxActiveFormBorderStyle); safecall;
    procedure Set_BiDiMode(Value: TxBiDiMode); safecall;
    procedure Set_Caption(const Value: WideString); safecall;
    procedure Set_Color(Value: OLE_COLOR); safecall;
    procedure Set_Cursor(Value: Smallint); safecall;
    procedure Set_DoubleBuffered(Value: WordBool); safecall;
    procedure Set_DropTarget(Value: WordBool); safecall;
    procedure Set_Enabled(Value: WordBool); safecall;
    procedure Set_Font(var Value: IFontDisp); safecall;
    procedure Set_HelpFile(const Value: WideString); safecall;
    procedure Set_KeyPreview(Value: WordBool); safecall;
    procedure Set_PixelsPerInch(Value: Integer); safecall;
    procedure Set_PrintScale(Value: TxPrintScale); safecall;
    procedure Set_Scaled(Value: WordBool); safecall;
    procedure Set_Visible(Value: WordBool); safecall;
  public
    { Public declarations }
    procedure Initialize; override;
  end;

implementation

uses ComObj, ComServ;

{$R *.DFM}

{ TOli_ActiveFormX }

procedure TOli_ActiveFormX.DefinePropertyPages(DefinePropertyPage:
TDefinePropertyPage);
```

```
begin
  { Define property pages here.  Property pages are defined by calling
    DefinePropertyPage with the class id of the page.  For example,
      DefinePropertyPage(Class_Oli_ActiveFormXPage); }
end;

procedure TOli_ActiveFormX.EventSinkChanged(const EventSink: IUnknown);
begin
  FEvents := EventSink as IOli_ActiveFormXEvents;
end;

procedure TOli_ActiveFormX.Initialize;
begin
  inherited Initialize;
  OnActivate := ActivateEvent;
  OnClick := ClickEvent;
  OnCreate := CreateEvent;
  OnDblClick := DblClickEvent;
  OnDeactivate := DeactivateEvent;
  OnDestroy := DestroyEvent;
  OnKeyPress := KeyPressEvent;
  OnPaint := PaintEvent;
end;

while TOli_ActiveFormX.Get_Active: WordBool;
begin
  Result := Active;
end;

while TOli_ActiveFormX.Get_AutoScroll: WordBool;
begin
  Result := AutoScroll;
end;

while TOli_ActiveFormX.Get_AutoSize: WordBool;
begin
  Result := AutoSize;
end;

while TOli_ActiveFormX.Get_AxBorderStyle: TxActiveFormBorderStyle;
begin
  Result := Ord(AxBorderStyle);
end;

while TOli_ActiveFormX.Get_BiDiMode: TxBiDiMode;
begin
  Result := Ord(BiDiMode);
end;

while TOli_ActiveFormX.Get_Caption: WideString;
begin
  Result := WideString(Caption);
end;

while TOli_ActiveFormX.Get_Color: OLE_COLOR;
begin
```

```
  Result := OLE_COLOR(Color);
end;

while TOli_ActiveFormX.Get_Cursor: Smallint;
begin
  Result := Smallint(Cursor);
end;

while TOli_ActiveFormX.Get_DoubleBuffered: WordBool;
begin
  Result := DoubleBuffered;
end;

while TOli_ActiveFormX.Get_DropTarget: WordBool;
begin
  Result := DropTarget;
end;

while TOli_ActiveFormX.Get_Enabled: WordBool;
begin
  Result := Enabled;
end;

while TOli_ActiveFormX.Get_Font: IFontDisp;
begin
  GetOleFont(Font, Result);
end;

while TOli_ActiveFormX.Get_HelpFile: WideString;
begin
  Result := WideString(HelpFile);
end;

while TOli_ActiveFormX.Get_KeyPreview: WordBool;
begin
  Result := KeyPreview;
end;

while TOli_ActiveFormX.Get_PixelsPerInch: Integer;
begin
  Result := PixelsPerInch;
end;

while TOli_ActiveFormX.Get_PrintScale: TxPrintScale;
begin
  Result := Ord(PrintScale);
end;

while TOli_ActiveFormX.Get_Scaled: WordBool;
begin
  Result := Scaled;
end;

while TOli_ActiveFormX.Get_Visible: WordBool;
begin
  Result := Visible;
```

```
end;

procedure TOli_ActiveFormX._Set_Font(const Value: IFontDisp);
begin
  SetOleFont(Font, Value);
end;

procedure TOli_ActiveFormX.Set_AutoScroll(Value: WordBool);
begin
  AutoScroll := Value;
end;

procedure TOli_ActiveFormX.Set_AutoSize(Value: WordBool);
begin
  AutoSize := Value;
end;

procedure TOli_ActiveFormX.Set_AxBorderStyle(
  Value: TxActiveFormBorderStyle);
begin
  AxBorderStyle := TActiveFormBorderStyle(Value);
end;

procedure TOli_ActiveFormX.Set_BiDiMode(Value: TxBiDiMode);
begin
  BiDiMode := TBiDiMode(Value);
end;

procedure TOli_ActiveFormX.Set_Caption(const Value: WideString);
begin
  Caption := TCaption(Value);
end;

procedure TOli_ActiveFormX.Set_Color(Value: OLE_COLOR);
begin
  Color := TColor(Value);
end;

procedure TOli_ActiveFormX.Set_Cursor(Value: Smallint);
begin
  Cursor := TCursor(Value);
end;

procedure TOli_ActiveFormX.Set_DoubleBuffered(Value: WordBool);
begin
  DoubleBuffered := Value;
end;

procedure TOli_ActiveFormX.Set_DropTarget(Value: WordBool);
begin
  DropTarget := Value;
end;

procedure TOli_ActiveFormX.Set_Enabled(Value: WordBool);
begin
  Enabled := Value;
```

```
end;

procedure TOli_ActiveFormX.Set_Font(var Value: IFontDisp);
begin
  SetOleFont(Font, Value);
end;

procedure TOli_ActiveFormX.Set_HelpFile(const Value: WideString);
begin
  HelpFile := String(Value);
end;

procedure TOli_ActiveFormX.Set_KeyPreview(Value: WordBool);
begin
  KeyPreview := Value;
end;

procedure TOli_ActiveFormX.Set_PixelsPerInch(Value: Integer);
begin
  PixelsPerInch := Value;
end;

procedure TOli_ActiveFormX.Set_PrintScale(Value: TxPrintScale);
begin
  PrintScale := TPrintScale(Value);
end;

procedure TOli_ActiveFormX.Set_Scaled(Value: WordBool);
begin
  Scaled := Value;
end;

procedure TOli_ActiveFormX.Set_Visible(Value: WordBool);
begin
  Visible := Value;
end;

procedure TOli_ActiveFormX.ActivateEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnActivate;
end;

procedure TOli_ActiveFormX.ClickEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnClick;
end;

procedure TOli_ActiveFormX.CreateEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnCreate;
end;

procedure TOli_ActiveFormX.DblClickEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnDblClick;
end;
```

```
procedure TOli_ActiveFormX.DeactivateEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnDeactivate;
end;

procedure TOli_ActiveFormX.DestroyEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnDestroy;
end;

procedure TOli_ActiveFormX.KeyPressEvent(Sender: TObject; var Key: Char);
var
  TempKey: Smallint;
begin
  TempKey := Smallint(Key);
  if FEvents <> nil then FEvents.OnKeyPress(TempKey);
  Key := Char(TempKey);
end;

procedure TOli_ActiveFormX.PaintEvent(Sender: TObject);
begin
  if FEvents <> nil then FEvents.OnPaint;
end;

{por parte del usuario}

procedure TOli_ActiveFormX.Manda_Peticion;
var
  I,Max : Word;
  Mensaje : String;
begin
  I := 0;Max := Memo1.Lines.Count - 1;Mensaje := '';
  while not (I > Max) do
  begin
    Mensaje := Mensaje + Memo1.Lines[I] + #13#10;
    I := I + 1;
  end;
  if (MessageDlg('Quiere enviar la solicitud?',mtWarning, [mbYes,mbNo],0)) =
mrYes then
    ClientSocket1.Socket.SendText(Mensaje);
end;

procedure
TOli_ActiveFormX.ColocaEnArchivo_DB(Dia_Real,Hora_Real:TDateTime;Sensor:string
; DatoReal:Real);
begin
   if Sensor = '01' then
   begin
     if Table1.FindKey([Dia_Real]) then
     begin
        Table1.IndexName := 'Sec_Hora';
        if Table1.FindKey([Hora_Real]) then
        begin
          Table1.edit;

Table1.SetFields([nil,nil,nil,DatoReal,nil,nil,nil,nil,nil,nil,nil]);
```

```
          Table1.Post;
        end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,DatoReal,nil,nil,nil,nil,nil,nil,n
il]);
     end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,DatoReal,nil,nil,nil,nil,nil,nil,n
il]);
   end;

   if Sensor = '02' then
   begin
     if Table1.FindKey([Dia_Real]) then
     begin
        Table1.IndexName := 'Sec_Hora';
        if Table1.FindKey([Hora_Real]) then
        begin
          Table1.edit;

Table1.SetFields([nil,nil,nil,nil,DatoReal,nil,nil,nil,nil,nil,nil]);
          Table1.Post;
        end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,DatoReal,nil,nil,nil,nil,nil,n
il]);
     end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,DatoReal,nil,nil,nil,nil,nil,n
il]);
   end; {Sensor 02}

  if Sensor = '03' then
  begin
     if Table1.FindKey([Dia_Real]) then
     begin
        Table1.IndexName := 'Sec_Hora';
        if Table1.FindKey([Hora_Real]) then
        begin
          Table1.edit;

Table1.SetFields([nil,nil,nil,nil,nil,DatoReal,nil,nil,nil,nil,nil]);
          Table1.Post;
        end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,DatoReal,nil,nil,nil,nil,n
il]);
     end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,DatoReal,nil,nil,nil,nil,n
il]);
  end; {Sensor 03}

  if Sensor = '04' then
  begin
    if Table1.FindKey([Dia_Real]) then
    begin
       Table1.IndexName := 'Sec_Hora';
       if Table1.FindKey([Hora_Real]) then
       begin
          Table1.edit;
```

```
Table1.SetFields([nil,nil,nil,nil,nil,nil,DatoReal,nil,nil,nil,nil]);
          Table1.Post;
        end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,DatoReal,nil,nil,nil,n
il]);
    end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,DatoReal,nil,nil,nil,n
il]);
  end; {sensor 04}

  if Sensor = '05' then
  begin
    if Table1.FindKey([Dia_Real]) then
    begin
       Table1.IndexName := 'Sec_Hora';
       if Table1.FindKey([Hora_Real]) then
       begin
         Table1.edit;
         Table1.SetFields([nil,nil,nil,nil,nil,nil,nil,DatoReal,nil,nil,nil]);
         Table1.Post;
       end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,DatoReal,nil,nil,n
il]);
    end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,DatoReal,nil,nil,n
il]);
  end; {sensor 05}

  if Sensor = '06' then
  begin
    if Table1.FindKey([Dia_Real]) then
    begin
       Table1.IndexName := 'Sec_Hora';
       if Table1.FindKey([Hora_Real]) then
       begin
         Table1.edit;
         Table1.SetFields([nil,nil,nil,nil,nil,nil,nil,nil,DatoReal,nil,nil]);
         Table1.Post;
       end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,DatoReal,nil,n
il]);
    end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,DatoReal,nil,n
il]);
  end; {sensor 06}

  if Sensor = '07' then
  begin
    if Table1.FindKey([Dia_Real]) then
    begin
       Table1.IndexName := 'Sec_Hora';
       if Table1.FindKey([Hora_Real]) then
       begin
         Table1.edit;
         Table1.SetFields([nil,nil,nil,nil,nil,nil,nil,nil,nil,DatoReal,nil]);
         Table1.Post;
```

```
      end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,nil,DatoReal,n
il]);
    end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,nil,DatoReal,n
il]);
  end; {sensor 07}

  if Sensor = '08' then
  begin
    if Table1.FindKey([Dia_Real]) then
    begin
       Table1.IndexName := ''ec_Hora';
       if Table1.FindKey([Hora_Real]) then
       begin
         Table1.edit;
         Table1.SetFields([nil,nil,nil,nil,nil,nil,nil,nil,nil,nil,DatoReal]);
         Table1.Post;
       end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,nil,nil,DatoRe
al]);
    end else
Table1.AppendRecord([nil,Dia_Real,Hora_Real,nil,nil,nil,nil,nil,nil,nil,DatoRe
al]);
  end; {sensor 8}

end;

procedure TOli_ActiveFormX.ToolButton7Click(Sender: TObject);
begin
  ClientSocket1.Host := InputBox('Conectar al servidor', 'Escriba el nombre
del servidor', 'CORELLIAN');
  ClientSocket1.Active := true;
  ToolButton10.Enabled := true;
end;

procedure TOli_ActiveFormX.ToolButton8Click(Sender: TObject);
begin
  ClientSocket1.Active := false;
  ToolButton10.Enabled := false;
end;

procedure TOli_ActiveFormX.ToolButton10Click(Sender: TObject);
{Por medio del SQL se crea una nueva tabla la cual contendra
temporalmente los datos para mostrarlos en la DBGrid
Por medio del socket se traeran los datos necesarios}
var
  CreaTabla,tiempo_InicionFin : string;
begin
  try
    Table1.Close;
    Memo1.Clear;
    Memo2.clear;
    tiempo_InicionFin := DateToStr(DateTimePicker1.Date) +'~' +
DateToStr(DateTimePicker2.Date);
    Table1.TableName := 'c:\windows\temp\SADTempo.db';
```

```
    Table1.IndexName := 'Sec_Dia';
    Table1.Open;
    if CheckBox1.Checked then Memo1.Lines.Add('01'+'~'+tiempo_InicionFin);
    if CheckBox2.Checked then Memo1.Lines.Add('02'+'~'+tiempo_InicionFin);
    if CheckBox3.Checked then Memo1.Lines.Add('03'+'~'+tiempo_InicionFin);
    if CheckBox4.Checked then Memo1.Lines.Add('04'+'~'+tiempo_InicionFin);
    if CheckBox5.Checked then Memo1.Lines.Add('05'+'~'+tiempo_InicionFin);
    if CheckBox6.Checked then Memo1.Lines.Add('06'+'~'+tiempo_InicionFin);
    if CheckBox7.Checked then Memo1.Lines.Add('07'+'~'+tiempo_InicionFin);
    if CheckBox8.Checked then Memo1.Lines.Add('08'+'~'+tiempo_InicionFin);
    Manda_Peticion;
    Table1.Close;
  except
    on EDBEngineError do
    begin
      MessageDlg('Se creara una tabla de trabajo.', mtWarning , [mbOK], 0);
      CreaTabla := 'CREATE TABLE "c:\windows\temp\SADTempo.db"'+
                   '('+#13#10+
                   'IDENT AUTOINC,' +#13#10+
                   'DIA DATE,'+#13#10+
                   'HORA TIME,'+#13#10+
                   'SENSOR1 NUMERIC(10,2),'+#13#10+
                   'SENSOR2 NUMERIC(10,2),'+#13#10+
                   'SENSOR3 NUMERIC(10,2),'+#13#10+
                   'SENSOR4 NUMERIC(10,2),'+#13#10+
                   'SENSOR5 NUMERIC(10,2),'+#13#10+
                   'SENSOR6 NUMERIC(10,2),'+#13#10+
                   'SENSOR7 NUMERIC(10,2),'+#13#10+
                   'SENSOR8 NUMERIC(10,2),'+#13#10+
                   'PRIMARY KEY(IDENT) '+#13#10+
                   ')' ;
      Query1.Close;
      Query1.SQL.Clear;
      Query1.SQL.Add(CreaTabla);
      Query1.ExecSQL;
      Query1.Close;
      CreaTabla := 'CREATE INDEX Sec_Dia ON "c:\windows\temp\SADTempo.db"
(DIA)';
      Query1.SQL.Clear;
      Query1.SQL.Add(CreaTabla);
      Query1.ExecSQL;
      Query1.Close;
      CreaTabla := 'CREATE INDEX Sec_Hora ON "c:\windows\temp\SADTempo.db"
(HORA)';
      Query1.SQL.Clear;
      Query1.SQL.Add(CreaTabla);
      Query1.ExecSQL;
      Query1.Close;
      CreaTabla := 'CREATE INDEX Sec_HorayDia ON "c:\windows\temp\SADTempo.db"
(DIA, HORA)';
      Query1.SQL.Clear;
      Query1.SQL.Add(CreaTabla);
      Query1.ExecSQL;
      MessageDlg('Tabla creada',mtInformation, [mbOK], 0);
      Table1.TableName := 'c:\windows\temp\SADTempo.db';
      Table1.IndexName := 'Sec_Dia';
      Table1.Close;
```

```
      Table1.Open;
      if CheckBox1.Checked then Memo1.Lines.Add('01'+'~'+tiempo_InicionFin);
      if CheckBox2.Checked then Memo1.Lines.Add('02'+'~'+tiempo_InicionFin);
      if CheckBox3.Checked then Memo1.Lines.Add('03'+'~'+tiempo_InicionFin);
      if CheckBox4.Checked then Memo1.Lines.Add('04'+'~'+tiempo_InicionFin);
      if CheckBox5.Checked then Memo1.Lines.Add('05'+'~'+tiempo_InicionFin);
      if CheckBox6.Checked then Memo1.Lines.Add('06'+'~'+tiempo_InicionFin);
      if CheckBox7.Checked then Memo1.Lines.Add('07'+'~'+tiempo_InicionFin);
      if CheckBox8.Checked then Memo1.Lines.Add('08'+'~'+tiempo_InicionFin);
      Manda_Peticion;
      Table1.Close;
    end; {EDBEngineError}
  end;{try}
end;

procedure TOli_ActiveFormX.ClientSocket1Error(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  MessageDlg('El servidor esta apagado.',mtinformation,[mbOK],0);
end;

procedure TOli_ActiveFormX.CheckBox1Click(Sender: TObject);
begin
  if checkBox1.Checked = true then
  begin
    DBGrid1.Columns.Items[2].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[2].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox2Click(Sender: TObject);
begin
  if checkBox2.Checked = true then
  begin
    DBGrid1.Columns.Items[3].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[3].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox3Click(Sender: TObject);
begin
  if checkBox3.Checked = true then
  begin
```

```
    DBGrid1.Columns.Items[4].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[4].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox4Click(Sender: TObject);
begin
  if checkBox4.Checked = true then
  begin
    DBGrid1.Columns.Items[5].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[5].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox5Click(Sender: TObject);
begin
  if checkBox5.Checked = true then
  begin
    DBGrid1.Columns.Items[6].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[6].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox6Click(Sender: TObject);
begin
  if checkBox6.Checked = true then
  begin
    DBGrid1.Columns.Items[7].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[7].visible := false;
end;
```

```
procedure TOli_ActiveFormX.CheckBox7Click(Sender: TObject);
begin
  if checkBox7.Checked = true then
  begin
    DBGrid1.Columns.Items[8].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[8].visible := false;
end;

procedure TOli_ActiveFormX.CheckBox8Click(Sender: TObject);
begin
  if checkBox8.Checked = true then
  begin
    DBGrid1.Columns.Items[9].visible := true;
    Try
      Table1.Open;
    except
      MessageDlg('La base de datos esta vacÝa. Solicite datos a su preveedor',
mtWarning, [mbOK],0);
    end;
  end
  else DBGrid1.Columns.Items[9].visible := false;
end;

procedure TOli_ActiveFormX.ToolButton1Click(Sender: TObject);
{Se encarga de graficar los datos que el usuario necesita}
var
  CreaTabla : string;
  inicio, fin : string;
begin
  Query2.SQL.Clear;
  Query2.Close;
  Series1.Active := false;
  Inicio := FormatDateTime('mm/dd/yyyy',DateTimePicker1.Date);
  Fin    := FormatDateTime('mm/dd/yyyy',DateTimePicker2.Date);
  Case ComboS_1.ItemIndex of
    0 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR1 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR1';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox1.caption);
        end;
    1 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR2 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
```

```
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR2';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox2.caption);
        end;
    2 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR3 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR3';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox3.caption);
        end;
    3 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR4 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR4';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox4.caption);
        end;
    4 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR5 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR5';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox5.caption);
        end;
    5 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR6 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR6';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox6.caption);
        end;
```

```
    6 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR7 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR7';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox7.caption);
        end;
    7 : begin
          CreaTabla := 'SELECT DIA, HORA, SENSOR8 FROM
"c:\windows\temp\SADTempo.db" SADTempo WHERE DIA BETWEEN "'+Inicio+'" AND
"'+Fin+'"';
          Query2.SQL.Clear;
          Query2.SQL.Add(CreaTabla);
          Series1.YValues.ValueSource := 'SENSOR8';
          Series1.Active := true;
          Query2.Active := true;
          DBchart1.Title.Text.clear;
          DBchart1.Title.Text.Add(CheckBox8.caption);
        end;
  end; {case}
end;

procedure TOli_ActiveFormX.ToolButton4Click(Sender: TObject);
begin
  {Realiza una animaci¾n sobre los datos recuperados}
  MessageDlg ('No es posible ejecutar esta funcion, en esta versi¾n de Prueba.
',mtInformation,[mbOK],0);
end;

procedure TOli_ActiveFormX.ToolB_VaciarClick(Sender: TObject);
begin
  Table1.Close;
  if MessageDlg('Esta a punto de limpiar los datos respalados en su base de
datos temporal. ¬Quiere Eliminarlos?',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    try
      Table1.DeleteTable;
    except
      on exception do
      messageDlg('Ya fueron eliminados los datos
datos.',mtInformation,[mbOK],0);
    end;
  end;
end;

procedure TOli_ActiveFormX.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
{ Cuando lea lo que envia el servidor lo decodificara y lo pondra
  en la base de datos temporal. 'c:\windows\temp\SADTempo.db'}
var
  InfoRecibida :string;
```

```
begin
  InfoRecibida := Socket.ReceiveText;
  {el servidor envia: '01~12/07/1999~10:32:37 a.m.~275.23'
                      ||||||||||||||||||||||||||||||||||
                      00000000011111111122222222233333
                      12345678901234567890123456789012345}
  memo2.lines.add(Inforecibida);

end; {onread socket1}

procedure TOli_ActiveFormX.Memo2Change(Sender: TObject);
var
  Infodetalle : string[100];
  DiaReal, HoraReal : TDateTime;
  Dia, hora : string[15];
  Dato : string;
  Elsensor : string[2];
  I,J,Max,LargodeInfo : word;
  ElDatoReal : real;
begin
  if (memo2.Lines.Strings[memo2.Lines.Count-1] = 'FIN') then
  begin
    messagedlg('Informacion solicitada: Recibida.
Decodificando...',mtinformation, [mbOK],0);
    I := 0;Max := Memo2.Lines.Count - 1;
    ProgressBar1.Max := Max;
    ProgressBar1.Position := 0;
    while not (I > Max) do
    begin
      Dia := ''; Hora := ''; Dato:='';
      InfoDetalle := Memo2.Lines[I];
      LargoDeInfo := length(InfoDetalle);
      if not (LargoDeInfo < 29) then
      begin
        ElSensor := InfoDetalle[1]+InfoDetalle[2];
        for J := 4 to 13 do Dia := Dia + InfoDetalle[J];
        for J := 15 to 27 do Hora := Hora + InfoDetalle[J];
        for J := 29 to (LargoDeInfo) do Dato := Dato + InfoDetalle[J];
        DiaReal := StrToDate(Dia);
        HoraReal := StrToTime(Hora);
        ElDatoReal := StrToFloat(dato);
        Table1.Close;
        Table1.TableName := 'c:\windows\temp\SADTempo.db';
        Table1.IndexName := 'Sec_Dia';
        Table1.Open;
        Table1.First;
        ColocaEnArchivo_DB(DiaReal,HoraReal,ElSensor,ElDatoReal);
      end;
      I := I + 1;
      ProgressBar1.Position := I;
    end; {while}
    {Table1.Close;}
    Table1.IndexName := 'Sec_HorayDia';
    ToolButton8.Click;
  end; {if fin}
end;
```

```
procedure TOli_ActiveFormX.FormCreate(Sender: TObject);
begin
  {if Panel2.ManualDock(Panel1,Panel2,alclient) then
     panel1.caption :='';

  if Panel5.ManualDock(Panel1,Panel5,albottom) then
     panel1.caption :='';

  if Panel6.ManualDock(Panel1,panel6,alright) then
     panel1.caption :='';}
end;

procedure TOli_ActiveFormX.N3D1Click(Sender: TObject);
begin
  if (N3D1.Checked) then
  begin
    N3D1.Checked := false;
    DBChart1.view3D := false;
  end
  else
  begin
    N3D1.Checked := true;
    DBChart1.view3D := true;
  end;
end;

procedure TOli_ActiveFormX.IncluirenReporte1Click(Sender: TObject);
begin
  if Incluirenreporte1.Checked then
  begin
    Incluirenreporte1.Checked := false;
  end
  else
  begin
    Incluirenreporte1.Checked := true;
  end;
end;

procedure TOli_ActiveFormX.Etiquetas1Click(Sender: TObject);
begin
    if (Etiquetas1.Checked) then
  begin
    Etiquetas1.Checked := false;
    Series1.Marks.Visible := false;
  end
  else
  begin
    Etiquetas1.Checked := true;
    Series1.Marks.Visible := true;
  end;
end;

procedure TOli_ActiveFormX.ToolButton11Click(Sender: TObject);
begin
  DBChart1.CopyToClipboardBitmap;
  MessageDlg('Llamada al *.dll',mtinformation,[mbOK],0);
```

```
end;

initialization
  TActiveFormFactory.Create(
    ComServer,
    TActiveFormControl,
    TOli_ActiveFormX,
    Class_Oli_ActiveFormX,
    1,
    '',
    OLEMISC_SIMPLEFRAME or OLEMISC_ACTSLIKELABEL,
    tmApartment);
end.
```

Información del Sistema

Programa

Nombre del Producto: _____
Versión: _____
Número de serie (si lo hubiera): _____

Hardware

Computadora: _____
Modelo: _____
Tamaño en RAM (en MB): _____

Sistema Operativo

□ Windows Versión: _____
□ Windows NT Versión: _____

Otras características

Incluya el nombre del fabricante, nombre del producto y número de modelo para hardware, si no lo conoce.

Disco Duro: _____
Capacidad (en MB): _____
Tipo □ SCXI □ ESDI □ RLL □MFM
□ IDE □Otro

Tarjeta de Video/Monitor: _____
Impresora: _____
Modem: Velocidad (en baudios):
Tarjeta de Red: _____
Hardware Adicional: _____
Otro Software: _____

Referencias Bibliográficas

[1] Enciclopedia, "Microsoft Encarta 99", Microsoft Corp., 1999

[2] Lee Mikles, "Internet impect felt on plant floor", InTech, p 46, June 1997, ISA Publication.

[3] David Wofford, Adris Bertle, and Jerry Diven, "Enhance SCADA with common databases", InTech, p. 49, December 1997, ISA Publication.

[4] S. Zafar Kamal, "Integrates enterprise proves key to flexible manufacturing", InTech, p.42, July 1998, ISA Publication.

[5] Paul W. Murril, "FUNDAMENTALS OF PROCESS CONTROL THEORY: SECOND EDITION", Instrument Society of America, Abril 1993

[6] Boulden, Geoffrey B., "Take the Pressure Out of Pressure-Sensor Selection" Chemical Engineering Progress", p. 70, November, 1990.

[7] Cho, Chun H., "Measurement and Control Liquid Level, ISA, 1982.

[8] Magison, E. C., "Temperature Measurement in Industry", ISA, 1984.

[9] Gunkler, Albert A., and Bernard, John W., "Computer Control Strategies for the Fluid Process Industries", ISA, 1990.

[10] "BORLAND DELPHI 4, FOR WINDOWS 95 AND WINDOWS NT", INPRISE Corp. 1998.

[11] Steve Teixeira & Xavier Pacheco. "BORLAND DELPHI 4, DEVELOPER'S GUIDE". SAMS Publishing. 1998

[12] Ted Burghart, "DISTRIBUTED COMPUTING OVERVIEW", QUOIN Inc., Junio 1998.

[13] R. Salz; "DCE 1.2 Contents Overview", Open Group RFC 63.3 (The Open Group, October 1996)

[14] "Java Remote Method Invocation" (Sun Microsystems, Inc., December 1997)

[15] "The Component Object Model Specification" (Microsoft Corporation, Digital Equipment Corporation, October 1995)

[16] N. Brown, C. Kindel; "Distributed Component Object Model Protocol" (Microsoft Corporation, January, 1998)

[17] "The Common Object Request Broker: Architecture and Specification, Version 2.1" (Object Management Group, et al, August 1997)

[18] "CORBAservices: Common Object Services Specification" (Object Management Group, et al, July 1997)

[19] Curso Teórico, "Introducción al Diseño de Redes de Monitoreo Atmosférico", Departamento del Distrito Federal, p.p. 10-15, México,

[20] "MEASURE AND AUTOMATION CATALOGUE 1999", National Instruments, 1999, DAQ Tutorial & DAQ Driver Software. http://www.ni.com

[21] "Reportes ambientales", Instituto Mexicano del Petróleo

[22] "Atlas de la Ciudad de México", Departamento del Distrito Federal. p.p. 37-40, Mexico

[22] "Manual de Instalación de Estación Meteorológica", MetOne Inc.

[23] "COMPONENTWORKS: GETTING RESULTS WITH CW", National Instruments, April 1998.

[24] "WHAT IS". www.whatis.com 1999

[25] "Referencia de Productos Microsoft", Microsoft, http://www.micorsoft.com

[26] "SCXI-1200 Manual de usuario", National Instruments, http://www.ni.com

Glosario

Ancho de banda (bandwith) - El ancho de banda (bandwith) es una medición de la capacidad de información de un enlace de comunicación. Ethernet, por ejemplo, tiene una ancho de banda (bantwith) de 10Mps. Se dice de aquellos usuarios de estaciones de trabajo o de redes que utilizar la red (network) de un modo intenso, que utilizan un *ancho de banda (bandwith) alto.*

Bit - Digito Binario. Es el elemento más pequeño de información de la computadora. Un bit es un único dígito en número binario (0 o 1). Los grupos de bits forman unidades más grandes de datos en los sistemas de computadora - siendo el byte (ocho bits) el más conocido de estos.

Bps - bits por segundo. bps se utiliza como medición de trasmisión de datos en un sistema de comunicaciones.

Byte - es una unidad de datos equivalente a ocho dígitos binarios (bits). Un byte contienen el equivalente a un carácter único, como puede ser una letra del alfabeto (a) o un signo de y (&). Las medidas de los dispositivos de almacenamiento, tales como discos y bases de datos, se dan en bytes.

Cliente - Es un sistema de computadora o de estación de trabajo que solicita un servicio o los contenidos de un fichero desde un servidor (server) de ficheros.

Clientes/servidor (client/server) - Es una arquitectura de red (network) por la que se divide un sistema en dos partes: el cliente (terminal frontal) es un computador solicitante (normalmente una PC), y el servidor (server) (terminal final) es una computadora suministrador. Ambos términos se puede aplicar a los dispositivos de hardware o a los programas de software.

Concentrador (hub) - Es un dispositivo que se utiliza en una red (network) y que sirve de emplazamiento central para comentar las estaciones de trabajo entre sí. El término *hub*

se utiliza a menudo como término genérico para determinar a un componente de equipo de red (network).

Conmutador (switch) - Un conmutador (switch) es parecido a un puente (bridge), aunque suele ser más rápido. El término *conmutación* proviene de la industria de las telecomunicaciones en donde originalmente se dominaba *conmutadores mecánicos* a los dispositivos que encaminaban llamadas telefónicas. Un conmutador (switch) segmenta una red (network) en dominios más pequeños, y proporciona un mayor porcentaje de ancho de banda (bandwith) acabe estación final (un estación final es, por ejemplo, una PC).

Encaminador (router) - Los encaminadores (routers) facilitar un enlace en redes separadas geográficamente. Una interconexión de redes basada en encaminamientos consiste en muchas subredes lógicas diferentes. Los puentes (bridges) y conmutadores (switches) conectan estas subredes, y su función es mejorar el rendimiento de la red manteniendo el tráfico dentro de los segmentos.

Ethernet - Xerox corporation inventó Ethernet, y lo desarrolló conjuntamente con Intel, y Digital Equipment Corporation (DEC), y es una tecnología utilizada extensamente en las LANs. Las redes Ethernet utilizar el protocolo CSMA/CD y funcionan en varios cables a una velocidad de 10Mps; son utilizadas, por ejemplo, por protocolos TCP/IP y XNS.

LAN - red de área local. Es una renta dispositivos conectados (como son computadoras, impresoras que, servidores (servers) y concentradores (hubs)) que cobrar una área geográfica relativamente pequeña abre paréntesis generalmente no más grande que una planta de un edificio). LANs se caracterizan por transmisiones de alta velocidad en cortas distancias. Ethernet, FDDI y Token Ring son tecnologías ampliamente utilizadas en la configuración de LANs.

Interconexión de redes - es un conjunto de redes (que puede ser de tipo diferentes) que están interconectadas por medio de encaminadores (routers), pasarelas (gateways), no otro dispositivo, para que de este modo puedan funcionar como una sola gran red (network).

Internet - El Internet es una red (network) global compuesta por redes gubernamentales, académicas, comerciales, militares y corporativas que abarcan todo el mundo. El internet fue desarrollado originalmente por el ejército norteamericano, y poco después se popularizó el investigación académica y comercial. Los usuarios que tienen acceso al Internet pueden leer descargar datos, virtualmente acerca de cualquier tema, desde cualquier parte del mundo.

Intranet - Los Intranets son redes privadas internas, utilizados por compañías e instituciones académicas alrededor del mundo. El público del exterior no puede exceder a estas Intranets que sirven como bases de datos de información en el mismo formato que utiliza la World Wide Web.

Kbps - Kilobits por segundo. Es la medida de la velocidad de transferencia que datos en un sistema de comunicaciones. Un kilobit equivale a 1000 bits.

Mbps - Megabits por segundo (no confundir con megabytes por segundo - m*B*ps). Es la medida de la velocidad de transferencia de datos en un sistema de comunicación. Un megabit es un millón de bits. Diez megabits por segundo (10Mbps) equivale a 10 millones de impulsos transmitidos por segundo en un sistema de comunicaciones.

Muro corta fuego (firewall) - Un muro corto fuego (firewall) en una red (network) es un nodo (node) configurado como una barrera para impedir que el tráfico cruce de un segmento a otro. Los muros corta fuego se utilizan para mejorar el tráfico, y a modo de medidas de seguridad, y pueden hacer las veces de una barrera entre público conectado y redes privadas. Se puede implementar un muro corta fuego (firewall) en un encaminador

(router) o puede ser un dispositivo de red especial para este propósito.

Nodo (node) - Es un dispositivo conectado a una red (network), como puede ser una computadora o un servidor (server)

Paquete - Un paquete es una unidad de información que es enviar entre sí las estaciones de trabajo y otro equipo a través de la red (network). Cuando se enviar información desde una computadora (como puede ser un fichero de texto), se transmite través de la red (network) como una serie de paquetes. Un paquete consisten un conjunto de bits que formar un solo bloque de datos, y que contiene un canal transversal formado por información de control como es el emisor, el receptor, y datos de control de errores, además del mismo mensaje.

Protocolo - En una red (network), un protocolo es un conjunto formal de normas y convenciones desarrollado por organismos reguladores internacionales que deciden como intercambian datos los distintos aparatos una red. Un protocolo de Fidel formato, la sincronización, el control y la secuencia de datos en una red.

Red (network) - Es un grupo de dispositivos, como por ejemplo las computadoras, las impresoras, los concentradores (hubs), los conmutadores (switches), y otros componentes de hardware, que están conectados entre sí y pueden comunicarse. Las redes varían en tamaño: unas pueden abarcar una sola oficina, y otras abarcar todo el mundo.

Red Cliente/Servidor (Network Client/Server) - Es una estructura de red (network) de área local (LAN) en la que los recursos de red (network) están centralizádos y controlados desde uno o más servidores (servers). Las estaciones de trabajo individuales o *clientes* (como son las Pcs) deben solicitar que los servicios a través del/los servidor/es.

Red de igual a igual (peer to peer) - Una red de igual a igual en la que las estaciones de trabajo (como las Pcs) pueden compartir información y los recursos de todos ellos, sin tener que depender de un servidor (server) central.. Por ejemplo, en una red de cuatro PCs y un concentrador (hub), es posible permitir el acceso a los ficheros de las cuatro Pcs desde cualquier otra PC.

Servidor (server) - Es una computadora o dispositivo especializado en una red (network) que comparte usuarios múltiples. Un servidor (server) facilita a los usuarios que el acceso a servicios de red (network) compartidos, tales como ficheros el computadora e impresoras.

Segmento - Un segmento es un grupo de dispositivos tales como PCs, servidores (servers) o impresoras están conectados entre sí por componentes de un equipo de red (network). En los segmentos Ethernet, las computadoras que se puede conectar entre sí por medio de concentradores (hubs), y las señales que se difunden en dicho segmento se oyen en todas las estaciones a él conectadas. Se pueden enviar paquetes entre dos segmentos si están interconectado por medio de un paquete (bridges) o un encaminador (router). Los segmentos de red conectados por puentes (bridges) o encaminador (router), forman interconexiones de redes. A los segmentos se les llama a menudo subredes.

TCP/IP - Protocolo de control de transmisión/Protocolo de Internet. Este es el nombre de dos de los protocolos más conocidos desarrollados por el Departamento de Defensa de los Estados Unidos en la década de los años 70, para permitir la comunicación entre el equipo de proveedores diferentes. Originalmente era un estándar de UNIX, pero ahora TCP/IP está soportaba en casi todas las plataformas, y es el protocolo del Internet. IP representa el esquema por el cual los dispositivos (ambos con direcciones IP) se comunican entre sí. TCP gestiona el flujo de paquetes IP, y garantiza que los paquetes estar libres de errores y llegan a su destino correctamente.

Token Ring - Desarrollado por IBM, Token Ring ofrecer un método para conectar dispositivos en una LAN (red de área local). Por ello, Token Ring ofrece el mismo servicio que Ethernet, pero llevado a cabo de un modo diferente: una señal electrónica (un paquete de datos) se pasa a través de esta estación en un anillo. La mayoría de los negocios pequeños instalan redes Ethernet, porque en comparación, son más sencillas que Token Ring.

Tráfico - Movimiento de paquetes de datos en una red (network)

WAN - Red de área extendida.

Web - ver WWW

WWW - World Wide Web (o Web) es un servicio de Internet que permite acceder fácilmente a la información en servidores (servers) por todo el mundo. Los examinadores de Web tales como Netscape Navigator e Internet Explorer, permiten a los usuarios que "examinar la Web" para acceder a esta información. Los documentos WWW se estructura utilizando HTML (HyperText Mark-up Language) y pueden incorporar aplicaciones JAVA, Javascript.