UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

APLICACIÓN DE LA COMPUTACIÓN EN LABORATORIOS DE CIENCIAS

T E S I S

QUE PARA OBTENER SU TÍTULO DE:

MATEMÁTICO

P R E S E N T A :

JULIO GERARDO FLORES OLVERA

Tutor: M. en C. Jordi Iñaki Austrich Senosiain

2006

DATOS DEL JURADO

1.- Datos del alumno
Flores
Olvera
Julio Gerardo
58090200
Universidad Nacional Autónoma de México
Facultad de Ciencias
Matemáticas
077351780

2.- Datos del tutor
M. en C.
Jordi Iñaki
Austrich
Senosiain

3.- Datos del sinodal 1
M. en C.
José de Jesus
Galaviz
Casas

4.- Datos del sinodal 2
L. en C.C.
Francisco Lorenzo
Solsona
Cruz

5.- Datos del sinodal 3
L. en C.C.
Iván
Hernández
Serrano

6.- Datos del sinodal 4
M. en I.
María de Luz
Gasca
Soto

7.- Datos del trabajo escrito
Aplicación de la computación en laboratorios de ciencias
82 p
2006

INDICE

INTRODUCCIÓN

Durante los últimos veinte años, a partir de la comercialización que tienen las computadoras personales, hemos visto como su uso se ha extendido en todos los terrenos, desde los familiares, educativos y profesionales.

Estas computadoras, desde sus versiones más rudimentarias, han ofrecido a sus propietarios una serie de aplicaciones que le permiten efectuar tareas de un modo más simple, como son manejo de documentos y datos diversos, hasta llegar a aspectos más complejos como la simulación y los sistemas de control.

Los beneficios que ofrecen las computadoras personales, pueden ser ampliamente explotados en el campo de la educación, ya que con una orientación adecuada, facilitan el aprendizaje y permiten que tanto alumnos como profesores puedan tener un nuevo punto de vista sobre los diversos temas que forman las currículas.

En el campo de las ciencias, como la física, química y biología, es ampliamente conocido por todos que la experimentación es una parte fundamental para reforzar la teoría, por lo que estas nuevas herramientas de cómputo, son el complemento ideal para ampliar el alcance de dichas actividades.

Una ventaja de estas herramientas, es su capacidad para recolectar datos externos para su posterior análisis, cosa que se puede lograr con programas e interfaces comerciales que logran resultados extraordinarios, pero que por el otro lado, su costo suele ser muy elevado, quedando fuera del alcance de muchas instituciones educativas de nuestro país.

Es por esta razón que el presente trabajo, es una propuesta para desarrollar programas educativos orientados a las ciencias básicas, cuyo costo sea reducido y que aporte nuevas formas de observación de los fenómenos que ocurren en nuestro entorno.

1.- MOTIVACIÓN Y CONTEXTO

1.1.-PRESENTACIÓN

El objetivo de esta tesis, es documentar el desarrollo y la evolución de un proyecto que surge en 1985, para construir un sistema orientado al uso de computadoras en laboratorios de ciencias (física, química, biología, etcétera.), como una opción económica que ofrece la posibilidad de profundizar más en los temas de la currícula de la educación media y media superior.

El proyecto comienza en 1985 con la arquitectura disponible en esa época, la cual era muy limitada en recursos técnicos y comenzaba a tener difusión en el ambiente académico de México.

Por otra parte, el proyecto era una idea innovadora para observar fenómenos físicos desde otra perspectiva y así reforzar el aprendizaje de la materia.

A partir del primer producto, se han realizado adaptaciones conforme avanza la tecnología, pasando por los procesadores 8088 con no más de 640 KB en RAM, hasta los modernos equipos con procesadores PENTIUM con más de 250 MB en RAM, ambientes gráficos y herramientas de desarrollo como Visual Studio.

Lo relevante de ésta evolución, es que la esencia del producto no ha cambiado, ya que su propósito didáctico sigue vigente, por lo que solo ha tenido algunas mejoras en cuanto a alcances y presentación se refiere, pero en principio sigue operando tal y como fue concebido desde su inicio.

El desarrollo de este documento, está distribuido de tal forma, que primero se describe cómo surge cada concepto, posteriormente se hace una descripción breve de cómo se implementó en la primera versión, para finalizar con una descripción más detallada de cómo se adaptó en la última versión bajo una arquitectura de equipos con procesadores INTEL®, en ambiente Windows® y herramientas de VISUAL STUDIO®.

1.2.-ANTECEDENTES

En 1985, como parte del equipo de trabajo de la organización “ORT México”®, dedicada a la educación tecnológica y en especial al uso de la tecnología para apoyar la enseñanza, surge la idea de construir una aplicación capaz de proveer una herramienta que facilite la experimentación en los laboratorios de ciencias y que ofrezca además la posibilidad de realizar análisis de la información obtenida.

Su principal objetivo, es ofrecer una opción para apoyar la educación en los laboratorios de ciencias a nivel básico, medio y medio superior. Conocemos que en el mercado se ofrecen instrumentos muy precisos para medir diversos tipos de fenómenos físicos y químicos, pero debido a que sus costos son elevados, no siempre están al alcance de muchas de las instituciones educativas de nuestro país tanto públicas como privadas.

Es por ello que observamos que los métodos para la realización de diversos experimentos en los laboratorios de ciencias de muchas instituciones, siguen siendo los mismos procedimientos totalmente manuales que se han empleado por más de cuarenta años.

La metodología comúnmente empleada, consiste como primer paso, en tomar los datos provenientes de la realización de un experimento, por ejemplo, el cambio de temperatura de un determinado volumen de un líquido a lo largo de un periodo de tiempo, en donde de forma manual es recolectada esta información por los alumnos al momento de realizar las pruebas.

Como siguiente paso, se lleva a cabo la graficación y ajuste de estos datos en hojas milimétricas, también de forma manual para ser presentados para su interpretación.

Los procedimientos anteriores consumen la mayor parte del tiempo destinado a las prácticas de laboratorio, por lo que generalmente existen pocas posibilidades de realizar un análisis más a fondo que permita entender la naturaleza del fenómeno que se estudia; así mismo resulta difícil poder realizar varios experimentos que permitan comparar las variaciones de un fenómeno ante los cambios en los parámetros del experimento.

Además de lo anterior, toda la información recolectada, solo queda presente en papel, el cual generalmente es destruido, por lo que no existe un antecedente de experimentos previos que nos permitan formar una biblioteca que pueda ser usada en estudios posteriores.

Por otra parte, existen tipos de experimentos que resultan imposibles de realizar por estos medios tradicionales, como puede ser tomar muestras del cambio de temperatura cada décima de segundo, o medir el cambio de velocidad cada diez cm. de un objeto que se desplaza a lo largo de plano inclinado, cuya distancia sea de un metro.

Aunque a partir de los años ochenta, la tecnología informática comenzó a tener grandes avances en cuanto a poner estos recursos al alcance de cualquier persona y en especial a todas las instituciones educativas del país, aún hasta nuestros días, no ha alcanzado a cubrir todas sus áreas de enseñanza.

1.3.- ALTERNATIVAS

Dado que para poder realizar experimentos que van más allá de los limites de los sentidos humanos se requiere de instrumentos muy especializados, que generalmente tienen un costo muy elevado, hemos buscado aprovechar las capacidades de proceso de información de los equipos de cómputo que actualmente existen en el mercado, para simular instrumentos, que aún no siendo tan sofisticados, si ofrezcan la posibilidad de realizar muestreos automatizados.

Como una alternativa a mejorar la calidad de la educación científica en los niveles básicos, se propone el uso de microcomputadoras tomando en cuenta que éstas se han convertido en herramientas muy populares en cualquier organización, así como a nivel personal, por lo que pueden ser aprovechadas para que a muy bajo costo, se puedan desarrollar programas que efectúen esta función, utilizando componentes económicos de hardware, con el fin de optimizar y mejorar el rendimiento de la enseñanza de estos tópicos.

El proyecto propuesto, no solo tiene el propósito de desarrollar un sistema de instrumentación, sino también elaborar prácticas que permitan un estudio más a fondo de fenómenos como la termodinámica y la mecánica, mostrando al alumno la relación de los conceptos teóricos sobre la práctica.

Desde el inicio del proyecto en 1985, este sistema se replantea con cada cambio de tecnología, con la finalidad de tener siempre un producto vigente que permita obtener el mayor provecho de los recursos que ofrecen los equipos más modernos, pero sin perder de vista los objetivos originales, que principalmente están orientados a tener un producto económico que pueda estar al alcance de un mayor número de instituciones en nuestro país.

2.- FUNDAMENTOS TEÓRICOS

Esta sección describe los fundamentos que se plantearon para iniciar el diseño del producto, entre los cuales están los alcances para buscar la mejor tecnología que se pueda utilizar.

2.1.- SELECCIÓN DEL EQUIPO

En su primera etapa, la selección del equipo era muy relevante, si consideramos que el mercado ofrecía diversas arquitecturas, para los diversos ambientes que abarcaba el mercado del cómputo personal.

En 1985 la oferta de marcas en el mercado mexicano, que habían abarcado principalmente el campo de la educación, eran ATARI®, COMMODORE®, APPLE® y TANDY®, teniendo cada una de ellas una arquitectura, sistemas operativos y herramientas de programación propias y exclusivas.

El estudio, inició con una evaluación de todas estas marcas, identificando las siguientes premisas para elegir entre éstas, la más adecuada para el desarrollo del proyecto:

- ⇨ Debe ser una herramienta con capacidad de recolectar datos provenientes de diversas fuentes que puedan ser medidos (ejemplo: luz, temperatura, PH, entre otras).

- ⇨ La recolección de datos debe permitir ser programada a intervalos de tiempos muy cortos y precisos o muy amplios (ejemplo: una centesima de segundo, días, semanas).

- ⇨ Los datos recolectados deben ser presentados gráficamente, con la mayor resolución disponible para su interpretación.

- ⇨ Las gráficas resultantes deben permitir ser modeladas, ajustadas y comparadas contra modelos teóricos.

De acuerdo a estas condiciones, se buscó un equipo que ofreciera las siguientes características:

- ⇨ El equipo debe permitir la conexión directa de componentes eléctricos y electrónicos que permitan la recolección de datos, por medio de alguna de sus interfaces, que requiera el mínimo necesario de hardware adicional, a fin de mantener un costo reducido del producto.

- ⇨ Debe contar con la mejor resolución gráfica, para mostrar con el mayor detalle la información.

- ⇨ Debe contar con herramientas de desarrollo que permitan la construcción de módulos estructurados y rutinas de mucha precisión en los tiempos de ejecución.

- ⇨ El equipo a seleccionar, debe tener presencia en el mercado a fin de ofrecer una plataforma estable que mejore con el tiempo, considerando conveniente seleccionar la que sea utilizada en el ámbito académico, sobre todo, de educación básica, media y media superior.

Tomando en cuenta estas condiciones y considerando que en ese momento la mayor base instalada de equipos de cómputo en las instituciones educativas corresponde a la marca APPLE® con su modelo II, II+, IIe de 64 y 128 Kb en memoria se llegó a la conclusión que era la selección más adecuada.

Las características en hardware que hicieron apropiado a este equipo son:

- ⇨ Cuenta con cuatro puertos para juegos que permiten la lectura directa de componentes analógicos, permitiendo una conversión digital de hasta 16 bits.

- ⇨ Su interface gráfica para el modelo IIe, por medio de la tarjeta de expansión de 128 Kb, permite una resolución 160 x 320 pixeles con hasta ocho colores.

- ⇨ Su tiempo de procesador permite realizar lecturas de hasta una décima de segundo.

- ⇨ Su compilador de PASCAL y el Ensamblador con que cuenta son herramientas apropiadas para el desarrollo del proyecto.

Por otra parte, se necesitaba encontrar componentes electrónicos que permitieran la lectura de fenómenos externos, por lo que se entró en contacto con otras instituciones educativas que ya tenían experiencia en este campo como es el Technological Educational Research Center (TERC®) en Boston, Massachussets, donde han realizado proyectos similares con fines educativos.

Por medio de este instituto, fue posible recolectar literatura que describía algunas metodologías para este tipo de proyectos. También por medio de ellos fue posible entrar en contacto con proveedores de algunos componentes que en ese momento no era posible encontrar de forma comercial en México.

Al buscar el apoyo de APPLE® de México, ellos nos ofrecieron una amplia documentación sobre la forma de explotar algunos de sus puertos mediante programación de bajo nivel, bibliotecas de graficación por medio de ensamblador.

A finales de 1987, se obtuvo la primera versión del producto y se utilizó con éxito en los colegios asesorados por ORT®, sin embargo el mercado de microcomputadoras en México comenzó a cambiar, probocando que marcas como APPLE® con sus modelos IIe y IIc, finalmente desaparecieran, siendo desplazadas rápidamente por los equipos IBM-PC® y compatibles.

Esta arquitectura reemplazó a los equipos en todos los laboratorios de cómputo de las instituciones educativas y poco a poco, ganó el mercado de todas las instituciones del país, siendo además, la primera arquitectura que se implantó oficialmente en las instituciones públicas del país, donde este tipo de proyectos aún no habían tenido alcance.

Esto motivó a que ese año el proyecto se retomara para migrar el desarrollo original a esta nueva plataforma, por lo que fue necesario replantear las premisas originales, para determinar como adaptar el producto. De ello resultaron las siguientes conclusiones:

- Aunque no contaba con un puerto al que directamente se pudieran conectar sensores, si tenía la posibilidad de incorporar diversos tipos de interfaces de expansión para este propósito, ya sea por medio de una tarjeta de juegos o componentes conectados al puerto serie o paralelo.

- Su resolución gráfica era la mejor del mercado. Seguía siendo de 160 x 320, como las APPLE® con tarjeta de 128KB, pero ahora con un máximo de dieciseis colores.

- La capacidad de memoria RAM aumentaba. Existían configuraciones desde 64 hasta 640KB, lo que permitía realizar desarrollos cuyo diseño, no era tan dependiente de las limitantes de este recurso.

- Contaba con un reloj interno, que era ajeno al resto de los componentes del procesador, apoyando al desarrollo de algoritmos sencillos de toma de datos, que garantizaran su precisión en el tiempo de ejecución.

- La empresa Borland®, ofrecía su versión del compilador Turbo PASCAL®, que tenía un alto porcentaje de similitud con la sintaxis del PASCAL de APPLE®.

La diferencia en esta nueva etapa, fue la decisión de incorporar el uso de nuevas interfaces analógico/digitales, que aunque significaban un costo adicional al del software, también representó una mejora importante a la calidad del producto en ese momento.

En un inicio, se utilizó una interfase desarrollada por la organización Technological Educational Research Center (TERC®) de Boston, a un costo aproximado de $80 USD, que se conectaba al puerto serial.

Aunque esta interface dió buenos resultados, un año después, en 1989, se utilizó otra interface desarrollada por ORT® Israel con un costo de $100 USD, que por tener su propia tarjeta que se instalaba en las bahías de expansión, tenia mejor rendimiento que la anterior.

A partir de 1999, vuelve a evolucionar el producto, en esta ocasión, para incorporarlo a la nueva tecnología de ambientes gráficos y herramientas de programación orientada a objetos.

Como siempre, fue necesario volver a plantear las premisas, obteniendo estas conclusiones:

- A fin de reducir costos, se vuelven a utilizar tarjetas con puertos de juegos, con un costo de $10 USD. Aunque podían encontrarse interfaces de muy alta tecnología, a un costo de $300 a $500 USD, por el momento se eligió la opción más económica, a fin de no perder el objetivo de tener un producto al alcance de todo el mercado educativo.

- El ambiente gráfico de Windows® es excelente ya que ofrece resoluciones muy finas, más colores, y la incorporación de componentes para graficación, hacen muy sencilla su manipulación.

- Se elige el lenguaje de programación VISUAL BASIC®, que aunque representa un cambio que implica a volver a escribir el código, también es cierto que su programación es sencilla y sigue siendo estructurada, además que al incorporar el uso de API’s para llamadas a funciones de Windows de bajo nivel, se puede prescindir de la programación en ensamblador:

Hay que resaltar, que por el momento no se utilizó lo mejor de la tecnología en interfaces, quedando latente ésta posibilidad, la cual volverá a considerarse una vez que se vea el resultado de este producto.

2.2.- DESARROLLO DE CONCEPTOS GENERALES

Para el diseño y desarrollo de esta aplicación, hubo que evaluar muchos elementos de software y de hardware que permitiera obtener el mayor provecho de la arquitectura disponible en 1985.

2.2.1.- SELECCIÓN DE INTERFACES Y SENSORES

La selección de las interfaces y los sensores era un punto fundamental para el proyecto ya que ésta sería la fuente de entrada de la información con la que el resto de las opciones del sistema trabajaría.

Como ya se mencionó, en la primera versión para APPLE®, se aprovechó su puerto de juegos, que permite la conexión de un JOYSTICK, que internamente se compone de resistencias variables y botones encendido/apagado.

El puerto de juegos en la APPLE® IIe, se compone de una salida de alimentación de 5 voltios y cuatro entradas de tipo analógicas que alimentan a un condensador.

El método que emplea la computadora para leer el puerto se basa totalmente en software ya que lo que realiza es una cuenta del periodo que tarda en cargarse el condensador. Este periodo varía de acuerdo al valor de la resistencia del componente que tenga conectado.

El puerto de juegos cuenta también con dos entradas digitales que por medio de una localidad de memoria es posible conocer su estado de prendido o apagado, que corresponde a los botones del JOYSTICK.

En esta interfase era necesario conectar componentes electrónicos, cuya resistencia variará en función del cambio de un factor externo, como la temperatura, la luz, etcétera.

Para esta selección, se contactó nuevamente con la organización Technological Educational Research Center (TERC®) de Boston, que por su experiencia en el campo de la tecnología aplicada a la educación, está en contacto con proveedores que ofrecen lo último en este tipo de material.

Para la primera fase del proyecto se eligieron las termo-resistencias y los foto-transistores, que eran adecuados para las características del puerto.

Las funciones originales en Ensamblador y PASCAL para la lectura del puerto, permite una precisión de ocho bits (0 a 255) que corresponde a una resistencia de aproximadamente 100K Ohm. Este rango puede ser ampliado si se desarrolla en Ensamblador una versión modificada de estas rutinas, que permita un conteo hasta los 16 bits y así lograr lecturas de componentes de mayor valor de resistencia.

La desventaja de este puerto, fue que al depender totalmente de software para su lectura, requería que el algoritmo fuera constante en su tiempo de ejecución, lo que implicaba tener que considerar siempre el máximo periodo que pudiera leer, afectando las lecturas de menos de diez décimas de segundo.

En las nuevas interfaces desarrolladas a partir de 1989, la lectura de la resistencia que entregan los sensores como foto-resistencias y termo-resistencias, se realizan por medio del hardware de la tarjeta y los valores son entregados al programa, transformándolos en la unidad adecuada, por ejemplo: a grados centígrados para temperatura o LUMEN en intensidad luminosa.

En la nueva versión actual, se retoma el uso de puertos de juego con los mismos sensores de resistencia variable.

Las lecturas se efectúan nuevamente con algoritmos similares a los de la primera versión, pero que en esta ocasión, la programación aprovecha los ciclos del reloj interno para tener la precisión necesaria en periodos muy cortos de muestreo (menores a un segundo). Esta nueva técnica en el algoritmo, elimina el uso de rutinas en Ensamblador, ya que a partir de los componentes de TIMER de Visual Basic y llamadas a API's del reloj interno, todo se realiza desde el lenguaje de alto nivel.
.

2.2.2.- METODOLOGÍA DE TOMA DE DATOS

El principal objetivo del proyecto es realizar muestreos contra el tiempo, cuyos periodos pudieran ir más allá del alcance del sentido humano, esto es, tomar muestras en periodos inferiores a un segundo, o muy largos como horas, días o semanas, así como en entornos con riesgo para la integridad humana, como lo sería una noche de invierno a menos de cero grados.

Analizando este requisito, se determinó que se necesitaba un método que garantice periodos exactos de tiempo para la toma de datos y su graficación.

Se analizaron las rutinas originales para la lectura de los puertos y se encontró que ésta no tiene un tiempo fijo de duración ya que terminaba cuándo el condensador se saturaba.

La rutina se modificó para que el periodo de conteo siempre fuera el valor máximo establecido y así mantener el tiempo constante.

Por otra parte, para llevar a cabo la graficación de cada medición al momento de ser tomada, se implementó un procedimiento de pre-cálculo de las coordenadas en base al sensor, almacenándolo en un arreglo en memoria que posteriormente era barrido en base a una búsqueda binaria. También aquí, este método, realizaba siempre el mismo número de lecturas para obtener un tiempo constante.

Para las termo-resistencias, su resolución se amplió de a dieciseis bits, logrando un intervalo de lectura de temperaturas desde 140 hasta -20 grados centígrados, mientras que con los foto-transistores la resolución necesaria para la obscuridad total no superaba los ocho bits.

Las opciones que se diseñaron para la toma de datos por medio de los sensores, tomando en cuenta la capacidad de velocidad y memoria disponible fueron:

- ⇨ Toma de datos en periodos predeterminados (de una décima de segundo a cuarenta y ocho minutos), pudiendo recolectar un máximo de cuatro gráficas de hasta sesenta datos cada una.

- ⇨ Simulación de instrumentos como termómetros y medidores de intensidad luminosa.

A partir de la integración de este proyecto en equipos PC, que inició con procesadores 8088, las capacidades de esta aplicación se incrementaron aprovechando el crecimiento de memoria (64KB contra 640KB), lo que permitió que el número de gráficas y su tamaño, se extendiera a 10 con un máximo de 1000 datos cada una.

En la nueva versión actual, aunque la arquitectura INTEL® ofrece mayores capacidades tanto de proceso como de almacenamiento, no se han hecho incrementos importantes en las capacidades del sistema, ya que se satisfacen ampliamente las necesidades didácticas para las que se diseñó.

2.2.3.- CONVERSIÓN DE VALORES Y CALIBRACIÓN DE SENSORES

El valor entregado por un sensor no corresponde a la unidad utilizada para la medición del fenómeno, requiriendo de fórmulas de conversión de valores.

En el caso de las termo-resistencias el fabricante provee la fórmula de conversión siguiente:

$$(B / Ln(A * X)) - 273.12$$

Donde:

A, B: constantes resultado de la calibración;
X: valor de la entrada del puerto de juegos.

El sensor de luz, cuenta con la fórmula:

$$I = A*B/(X+C)$$

Donde X = valor de la entrada analógica;
A, B: constantes resultado de la calibración.

Las constantes que se muestran en las fórmulas, son valores que se determinan por medio de una opción del sistema que calibra los sensores, ya que estos, no tienen una composición estable en su construcción, lo que implica una variación en su resistencia de acuerdo al valor que detectan.

2.2.4.- MANIPULACIÓN DE DATOS

Una vez que se han recolectado los datos, se debe contar con opciones que permitan su mejor análisis. Para ello, se estudió el tipo de manipulación que los datos requieren determinando las siguientes necesidades:

- Los datos obtenidos en los experimentos están sujetos a error por causa de factores externos no siempre controlables, por lo que en muchos casos, estos datos son alterados o eliminados para obtener una gráfica más aproximada al modelo teórico.

- En ocaciones, se debe revisar gráficamente áreas muy pequeñas del experimento realizado, con el propósito de captar detalles del fenómeno.

- Un método empleado para poder entender la naturaleza de un fenómeno, es en muchos casos, el sobreponer varias gráficas con el mismo experimento, en donde se han cambiado algunos parámetros, por lo que contar con una biblioteca de gráficas resulta de mucho valor.

- Todas las gráficas obtenidas por medio de métodos experimentales deben ser comparadas contra el modelo matemático que apoya la teoría, para así, entender las variaciones que causan los factores externos que están fuera de nuestro control, como son: la fricción del aire, la capacidad aislante de los materiales, etcétera.

Considerando los puntos anteriores, se determinó que el sistema debía cubrir los siguientes conceptos:

- Las gráficas presentadas podrán ser amplificadas mediante la modificación de las escalas en los ejes cartesianos, además de ser acotadas por sus valores máximo y mínimo.

- Los datos recolectados deben tener la posibilidad de ser editados manualmente para su ajuste o eliminación y así eliminar errores por causas externas.

- Las gráficas deben ser guardadas y leídas en dispositivos de almacenamiento como diskettes, disco duro y CD.

- Debe existir la posibilidad de introducir modelos matemáticos que permitan su comparación, contra las gráficas obtenidas en experimentos.

La construcción de estos conceptos, inicia en la primera versión con el uso de archivos planos en formato ASCII, situación que cambia en la versión actual, donde se usan estructuras de base de datos que permiten una mejor clasificación.

La descripción detallada del desarrollo de estas opciones se describe en capítulos posteriores.

2.3.- MATERIAL DE APOYO

Considerando que el objetivo de la aplicación es ofrecer un nuevo enfoque a las actividades en los laboratorios de ciencias, que nos permita reforzar al máximo los conceptos teóricos aprendidos en clase, es necesario el desarrollo de material didáctico de apoyo que oriente a los profesores en como aprovechar esta herramienta en su totalidad.

El material a desarrollar debe considerar lo siguiente:

- Debe buscar que el procedimiento de toma de datos, sea sólo una etapa más de la fase de experimentación, sin que ésta sea en sí misma la totalidad de la práctica.
- Las prácticas deben iniciar con un planteamiento de ideas acerca del comportamiento de un fenómeno, para posteriormente ser comprobadas o rechazadas por el resultado del experimento realizado.
- Deben permitir la comparación directa con los modelos teóricos, dando la posibilidad de deducir la razón de las posibles diferencias que existan durante la toma de datos del experimento.

- Al final del experimento, el alumno, tendrá los elementos suficientes para comprobar las ideas originalmente planteadas y podrá sacar sus propias conclusiones del fenómeno.

Debido a que el desarrollo de este tipo de material, requería de personas con experiencia en el campo de la docencia y una formación pedagógica en nuevas técnicas de enseñanza, fue necesario formar un equipo de trabajo que participara con los responsables del desarrollo del producto, para elaborar el material acorde a los objetivos planteados.

Como el objetivo de este documento se enfoca únicamente al desarrollo del producto de software, en capítulos posteriores se darán algunos ejemplos del material desarrollado, que muestra cómo se utilizó este producto.

3.-DISEÑO Y DESARROLLO DEL SISTEMA

Esta sección describe el diseño y desarrollo de la aplicación, detallando como han sido adaptados sus funciones y rutinas desde la primera versión, hasta las herramientas de VISUAL STUDIO® para la versión actual.

El producto en esencia es el mismo a lo largo del tiempo y sólo se han aprovechado las ventajas que ofrece ahora, el ambiente Windows® con las herramientas de desarrollo visual orientadas a objetos.

El diseño de la aplicación, consistió en construir módulos que contienen los algoritmos que materializan los conceptos que definen al producto, los cuales quedan de la siguiente manera:

- Incorporación de sensores y calibración.
- Algoritmos de muestreo para toma de datos.
- Manipulación de datos de muestreo.
- Manejo de presentación de gráficas.
- Aplicación de modelos matemáticos.
- Almacenamiento de información.
- Exportación e importación de datos.

3.1.- INCORPORACIÓN DE SENSORES Y CALIBRACIÓN

Esta sección describe la forma en que se seleccionaron los componentes electrónicos como foto-resistencias y termo-resistencias, al igual que las interfaces de hardware que requiere la computadora, además de su manejo dentro del sistema.

3.1.0.- HARDWARE E INTERFACES PARA SENSORES

La introducción, describe como este sistema obtiene muestreos de eventos externos por medio de la conexión de componentes electrónicos cuya resistencia varía en función de fenómenos como la temperatura y la intensidad luminosa, lo que nos lleva a utilizar interfaces que permitan a la computadora la lectura y conversión de estos valores en unidades conocidas como grados centígrados o LUMEN, para ser graficados y manipulados.

En el proyecto original, se usó el puerto de juegos que proveía APPLE® en su modelo IIe, diseñado para la conexión de palancas de juego o JOYSTICK, así como dos botones para detectar estados de encendido/apagado.

Las palancas de juego, son resistencias variables conocidas como potenciometros, que pueden ser sustituidas por resistencias variables ante un fenómeno, como son las termo-resistencias y foto-resistencias.

Cuando el proyecto evoluciona a los primeros equipos PC, se hizo un cambio importante en las interfaces, al sustituir los puertos de juego, por componentes comerciales, desarrollados en instituciones dedicadas a la tecnología orientada a la educación.

Estas interfaces, son eficaces para los propósitos del sistema, pero resultan costosas, ya que su precio fluctúa entre los $100 y $250 USD, además de ser dificil su adquisición en volúmenes pequeños, ya que implica tiempos de envío muy largos, cargos por fletes e impuestos aduanales, incrementando su costo al doble del valor original.

Por estos inconvenientes, en la versión actual, se vuelven a utilizar los puertos de juego, ya que uno de los objetivos, es ofrecer una herramienta al alcance de cualquier tipo de institución educativa del país.

En la actualidad, el puerto de juegos, no es un componente integrado a la configuración básica de una PC, pero adquirirlo en el mercado nacional, tiene un costo que va desde los $100 hasta los $250.

Para el desarrollo de este trabajo, se adquirió una tarjeta de sonido de mediana calidad que incluye el puerto de juegos, por un costo de $140.00 y cuya instalación y configuración es muy sencilla.

Para estas tarjetas, se necesitaba encontrar la forma de reconocerla desde un lenguaje de programación en ambiente Windows®, en nuestro caso particular, VISUAL BASIC®.

Desde las versiones de BASIC para sistema operativo MSDOS®, estas funciones de lectura de puertos de juegos, ya no son parte de la sintaxis habitual, pero existe una biblioteca que permite el acceso a ellos desde programas en VISUAL BASIC®.

La biblioteca que lo permite, se llama "`winmm.dll`" y se utiliza por medio de API’s que nos dan acceso a las siguientes dos funciones:

- joyGetDevCaps

Permite conocer si existe un puerto de juegos en la computadora y de ser así, nos proporciona información del mismo, sobre la siguiente estructura que recibe como parámetro:

```
Type JOYCAPS
  wMID        As Integer   ‘ No. de manufactura
  wPID        As Integer   ‘ ID del producto
  szPname     As String * 32    ‘ Nombre del producto
  wXmin       As Long   ‘ valor mínimo de posición X
  wXmax       As Long   ‘ valor máximo de posición X
  wYmin       As Long   ‘ valor mínimo de posición Y
  wYmax       As Long   ‘ valor máximo de posición Y
  wZmin       As Long   ‘ valor mínimo de posición Z
  wZmax       As Long   ‘ valor máximo de posición Z
  wNumButtons As Long   ‘ cantidad de botones
  wPeriodMin  As Long  ‘ intervalo mínimo de poleo de lectura
  wPeriodMAX  As Long  ‘ intervalo máximo de poleo de lectura
End Type
```

La sintaxis para declarar la función es la siguiente:

```
Declare Function joyGetDevCaps Lib "winmm.dll"_
  Alias "joyGetDevCapsA" (ByVal id As Long,_
  lpCaps As JOYCAPS, ByVal uSize As Long) As Long
```

- joyGetPos

Por medio de esta función, obtenemos el valor de la posición de los ejes **X** y **Y** de hasta dos palancas de juego conectadas al puerto.

Dichos valores, son colocados por la función dentro de la siguiente estructura que recibe como parámetro:

```
Type JOYINFO
  wXpos       As Long  ‘ Posición sobre el eje X
  wYpos       As Long  ‘ Posición sobre el eje Y
  wZpos       As Long  ‘ Posición sobre el eje Z
  wButtons    As Long  ‘ estado de los botones
End Type
```

La función recibe adicionalmente un valor entero que le indica si la lectura se hará de la primera palanca (`id=0`) o de la segunda (`id=1`).

Como cada eje representa la posición de una resistencia variable, con una sola tarjeta podemos conectar hasta cuatro sensores simultáneamente.

La sintaxis completa de la declaración de la función es:

```
Declare Function joyGetPos Lib "winmm.dll"_
   (ByVal uJoyID As Long, pji As JOYINFO) As Long
```

Resuelto el problema de conectar y reconocer los sensores en la computadora, el siguiente paso es convertir dichos valores de entrada en unidades de fenómenos conocidos.

Los valores que regresa el puerto, corresponden a un conteo que realiza la tarjeta, hasta que un capacitor alcanza su carga máxima, siendo necesario contar con funciones de conversión y calibración, las que se explican en las siguientes secciones.

3.1.1.- DEFINICIÓN DE ESTRUCTURAS DE SENSORES

Ya se mencionó que los sensores utilizados, son resistencias variables ante un fenómeno, por lo que cuando surge el proyecto, sólo se consideraron los de temperatura y luz, ya que existía suficiente documentación sobre su calibración y sus funciones de conversión.

En un principio, estas funciones eran parte del código del programa, pero conforme avanzó el proyecto, fueron apareciendo nuevos sensores en el mercado, como los de Ph, Presión, entre otros.

Como la gama es muy amplia, la versión actual, integra la posibilidad de agregar dinámicamente sensores, teniendo que registrar para ello datos como:

- Nombre del sensor.
- Función de transformación.
- Función inversa de transformación.
- Procedimiento de calibración.
- Rango mínimo y máximo.

Para lograr esta integración dinámicamente, se consideró la definición dentro del sistema de una estructura genérica que contenga todos los datos sobre cada sensor que se le ha incorporado.

A continuación, se muestra la estructura completa en el código de VISUAL BASIC® para almacenar esta información:

```
Type str_Sensor_rec
  G_Sensor   As String  'Tipo de Sensor
  G_Unidad   As String  'unidad de medida
  G_Tipo     As Long    ' 0=Fijo por sistema,
                        ' 1=Agregado por usuario
  G_MaxMed   As Long    'Máxima cantidad de mediciones
  G_MinInter As Double  'Mínimo Intervalo de medición
  G_MAX      As Double  'Valor Máximo en escala Y
  G_MIN      As Double  'Valor Mínimo en escala Y
  G_Fx       As String  'Función de transformación de datos
  G_FxInv    As String  'Función de transformación de datos
  G_DLLcall  As String  'Biblioteca dinámica con simulación
                        'de instrumento asociado al sensor
  G_DLLadj   As String  ' Biblioteca dinámica para calibrar
  xObj       As Object  'Objeto de biblioteca del sensor
End Type
```

En un módulo global se declara un arreglo para almacenar la información de hasta 50 sensores:

```
Global Sensor_rec(50)  As str_Sensor_rec
```

En la estructura, resaltan las variables `G_DLLcall, G_DLLadj` y `xObj`, que servirán para poder ligar dinámicamente al sistema, módulos externos con la definición de nuevos sensores, aprovechando la programación orientada a objetos de esta herramienta de desarrollo. La descripción detallada de esto, se verá en las siguientes secciones.

Finalmente, la información que se guarda en el arreglo de estructura de sensores, es escrita en una tabla de una base de datos en ACCESS®, que se utiliza también para realizar los cálculos requeridos para la preparación de los muestreos.

3.1.2.- CALIBRACIÓN DE SENSORES E INSTRUMENTACIÓN

Considerando que los sensores utilizados son componentes electrónicos que no son exactos en el rango de valor que entregan en función del fenómeno del cual dependen, es necesario calibrarlos por medio de una opción incorporada en la aplicación.

En la primera versión, sólo el sensor de temperatura requería de esta opción, por lo que dentro del código se diseñó esta vista.

Considerando que en la versión actual ya es posible incorporar dinámicamente nuevos sensores y que algunos de ellos pueden requerir de una opción de calibración, se aprovechó la facilidad que ofrece la programación orientada a objetos, para poder agregar en el sistema, aquellas ventanas especiales que requieran un nuevo sensor.

El modelo de programación orientada a objetos en VISUAL BASIC®, permite la construcción de proyectos que crean bibliotecas (DLL's), que por medio de las clases que los definen, pueden incorporarse dinámicamente a otras aplicaciones.

Para explicar esto detalladamente, veremos como ejemplo la clase que se creó para incorporar la calibración del sensor de temperatura en el sistema.

Aunque esta opción podría haber sido incorporada como una forma más del proyecto, se prefirió seguir el modelo que se diseñó para todo sensor, con la única diferencia de que se trata de una clase privada al proyecto.

Esta clase privada, es construida con el estándar que se define para cualquier sensor que se quiera incorporar de forma externa al sistema.

La clase mencionada, tiene el nombre de `LABClsTemp`, que consta de una colección de funciones privadas, así como propiedades y métodos que a continuación describiremos.

Las funciones de una clase que define un sensor son las de conversión de la entrada del puerto de juegos a la unidad deseada, así como a al inversa.

Para el caso de los sensores de temperatura y luz aquí descritos, las funciones y su código son los siguientes:

Funciones:

```
' ============================================================
' Función que regresa el valor convertido a la unidad de medida
' que corresponde al sensor (centígrados, LUMEN, etcétera)
'
'   xInput = Valor de la entrada del puerto de juegos
'     xIdx = Índice del sensor dentro de la clase
' ============================================================
Public Function ClsValor(ByVal xInput As Double,_
                         ByVal xIdx As Integer) As Double

  Select Case xIdx

    Case IdxTemp ‘ Sensor de temperatura
        ClsValor = (Cls_comMod(xIdx).B /_
          Log(Cls_comMod(xIdx).A * xInput))_
            - 273.12 ' TEMPERATURA }

    Case IdxLuz  ‘ Sensor de Luz
      ClsValor = (Cls_comMod(xIdx).B / (1# + xInput)) *_
                 Cls_comMod(xIdx).A -_
     ((Cls_comMod(xIdx).B * Cls_comMod(xIdx).A) / 65537#) 'Luz}

  End Select
```

```
End Function


' ============================================================
'Función que regresa el valor de la función inversa a la unidad de
medida
'que corresponde al sensor (centígrados, LUMEN, etcétera)
'
'   xValor = Valor de en la unidad de medida del sensor
'     xIdx = Índice del sensor dentro de la clase
' ============================================================
Public Function ClsPortInput(ByVal xValor As Double, ByVal xIdx As
Integer) As Long

  Select Case xIdx

    Case IdxTemp  ‘ Sensor de Temperatura
      ClsPortInput = Exp(Cls_comMod(xIdx).B /_
                 (xValor + 273.12))_
                 / Cls_comMod(xIdx).A 'TEMPERATURA^-1}
    Case IdxLuz   ‘ Sensor de Luz
      ClsPortInput = (Cls_comMod(xIdx).A * Cls_comMod(xIdx).B /_
       (xValor + ((Cls_comMod(xIdx).B * Cls_comMod(xIdx).A) /_
       65537#))) - 1

  End Select

End Function
```

Más adelante, se describe cómo se usa la función inversa para hacer precálculos que facilitan la graficación simultánea durante el proceso de muestreo.

Métodos:

```
- Public Sub InitSensor(ByVal xBDdir As String,_
                        ByVal xREGISTRO As String)
```

Este método, se manda ejecutar después de crear el objeto y tiene dos propósitos:

El primero es dar a conocer a la clase, la ruta de la base de datos que contiene las tablas de sensores con valores calibrados, así como las tablas de trabajo para calcular las tablas de conversión de valores. Esto lo establecemos al pasar este valor en el parámetro `xBDdir`.

El segundo parámetro, es una cadena que identifica de forma única cada sensor.

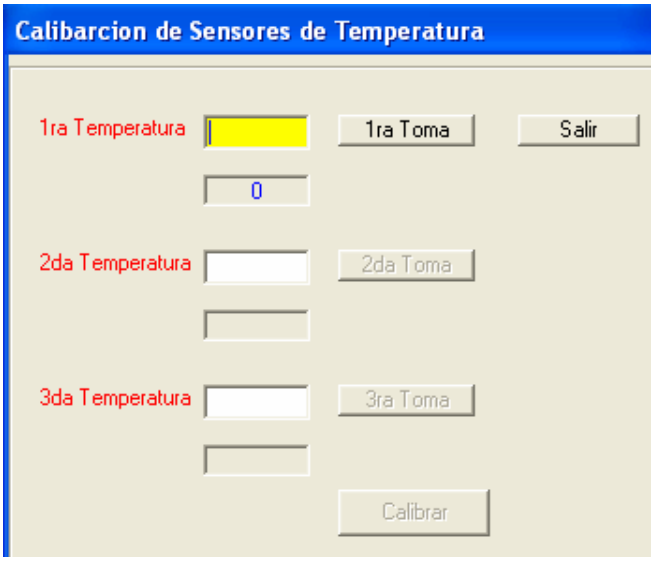
En este ejemplo, la clase contiene la definición de dos sensores, por lo que al crear el objeto, debemos pasar un valor para definir que tipo representa, siendo en este caso las siguientes que se obtienen de la tabla de sensores de la base de datos del sistema:

"LAB-00001-TMP" = Sensor de Temperatura
"LAB-00001-LUZ" = Sensor de Luz.

- `Public Sub Calibrar(xLbl As Object)`

El método calibrar, muestra una ventana que contiene el algoritmo que se usa para calibrar los sensores que define la clase. En el caso del sensor de temperatura, dicha calibración, se basa en la toma de tres valores externos, a los que el usuario relaciona con la temperatura, ajustando así las constantes de la función de conversión para obtener una medición exacta de la temperatura con ese sensor.

Como ya se mencionó que sólo el sensor de temperatura se calibra, el sistema determina esto al tener una cadena vacía asignada a la variable `G_DLLadj` de la estructura de sensores.



El parámetro que recibe este método, es un objeto tipo LABEL que corresponde a una etiqueta colocada en la ventana principal del sistema. Su objetivo es funcionar como un medio de comunicación con la clase para determinar cuando activar y desactivar el menú principal.

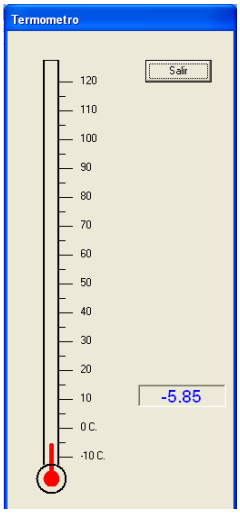
La razón de esto, es evitar que mientras está activa la ventana de calibración, no se tenga acceso al resto de las opciones de la ventana principal del sistema.

- `Public Sub Sensor(xLbl As Object)`

Otra facilidad incorporada en el sistema, es proveer al usuario de un instrumento similar a los usados en un laboratorio, como es el caso de un termómetro común de mercurio o de un fotómetro.

Con este tipo de opciones, los usuarios cuentan con esta instrumentación para efectos de realizar experimentos simples que sólo requiera la toma de unos cuantos datos o bien como ayuda para preparar un experimento más complejo.

Al igual que en el método de calibración, el parámetro que contiene, corresponde a la misma etiqueta que activa y desactiva el menú principal del sistema.



Esta ventana de instrumento, utiliza un objeto TIMER, para realizar un muestreo cada 500 milisegundos para tomar una muestra desde el puerto número uno, aplicar la función de conversión y graficar el valor obtenido.

A continuación se describe el código de este TIMER usado en la ventana del termómetro:

```
Private Sub Timer1_Timer()

  Dim X1 As Double
  Dim Y1 As Double
  Dim LonY As Long

' Lee el valor del Puerto de juegos
  Dame_input 1, X1

' convierte la entrada a grados centígrados
  Y1 = ClsValor(X1, IdxTemp)

' muestra el valor sobre un objeto LABEL
  txt_tmp.Caption = Format(Y1, "####.00")
```

```
‘ Ajusta el tamaño de un objeto LINE para
‘ simular el volumen del mercurio dentro
‘ de un termómetro de laboratorio
  LonY = Int(Dy * Y1) / 10

  Line3.Y1 = Line5(0).Y1 - LonY

End Sub
```

Propiedades:

- `Public Property Get/Let A() As Double`

- `Public Property Get/Let B() As Double`

- `Public Property Get/Let C() As Double`

- `Public Property Get/Let D() As Double`

Las cuatro propiedades, permiten asignar y obtener los valores de calibración de cada sensor, que pueden ser hasta cuatro (`A, B, C y D`).

- `Public xRegID as String`

La propiedad `xRegID`, regresa el valor del tipo de sensor que se definió en el método ‘`InitSensor`’.

- `Public Function Valor(ByVal xInput As Long) As Double`

- `Public Function PortInput(ByVal xValor As Double) As Long`

Con estas dos propiedades, obtenemos el valor convertido de la entrada del puerto de juegos (`Valor`), así como la función inverso, el valor del puerto de juegos equivalente al valor de una unidad de medida del sensor utilizado.

Hasta aquí hemos descrito las estructuras para almacenar sensores en el sistema (3.1.1) y el uso de clases para crear bibliotecas (DLL’s) que permiten su incorporación dinámica. Ahora pasamos a describir el algoritmo para leer la información registrada, en dicha estructura:

```
‘ Apertura de la tabla de B.D. que contiene la información
‘ de cada  sensor registrado en el sistema
Set tab_wrk = db.OpenRecordset("Select * from sensors”)
I = 0
K = 0
H = 0


Do While Not tab_wrk.EOF
```

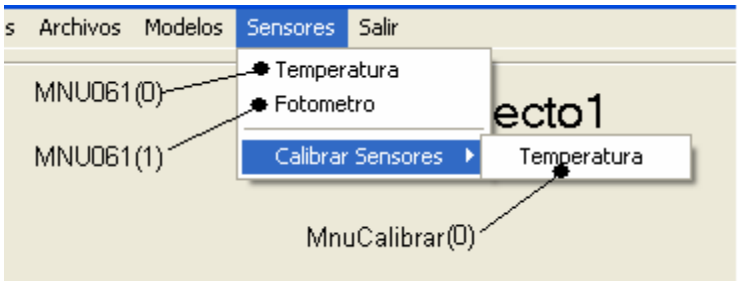
```
      Sensor_rec(I).G_Sensor = tab_wrk("Sensor")
      Sensor_rec(I).G_Unidad = tab_wrk("unidad")
      Sensor_rec(I).G_MaxMed = tab_wrk("MaxMed")
      Sensor_rec(I).G_MinInter = tab_wrk("MinInter")
      Sensor_rec(I).G_MIN = tab_wrk("Y_MIN")
      Sensor_rec(I).G_MAX = tab_wrk("Y_MAX")
      Sensor_rec(I).G_Fx = tab_wrk("Fx")
      Sensor_rec(I).G_FxInv = tab_wrk("FxInv")

' Crea el objeto de la clase con la definición de las
' características del sensor
      Sensor_rec(I).G_DLLcall = tab_wrk("DLLcall")
      Set Sensor_rec(I).xObj = CreateObject(tab_wrk("DLLcall"))
      Sensor_rec(I).G_DLLcall = ""

      K = K + 1
      LABMNUopc(K, 1) = I
      MNU061(K).Visible = True
      MNU061(K).Caption = Sensor_rec(I).G_Sensor

' Si el campo "DLLadj", no es una cadena vacía, la clase
' tiene definido un método de calibración del sensor
      If Not IsNull(tab_wrk("DLLadj")) Then
        Sensor_rec(I).G_DLLadj = tab_wrk("DLLadj")
      Else
        Sensor_rec(I).G_DLLadj = ""
      End If

      If Trim(Sensor_rec(I).G_DLLadj) <> "" Then
        H = H + 1
        LABMNUopc(H, 2) = I
        MNUCalibrar(H).Caption = Sensor_rec(I).G_Sensor
        MNUCalibrar(H).Visible = True
      End If

      Sensor_rec(I).xObj.InitSensor G_BDpath, tab_wrk("SensorID")
      Sensor_rec(I).xObj.A = tab_wrk("Val_A")
      Sensor_rec(I).xObj.B = tab_wrk("Val_B")
      Sensor_rec(I).xObj.C = tab_wrk("Val_C")
      Sensor_rec(I).xObj.D = tab_wrk("Val_D")
      I = I + 1
      tab_wrk.MoveNext
    Loop

    tab_wrk.Close

    Sensor_rec(I).G_Sensor = "*eor*"
```

La función del código anterior, es leer el contenido de la tabla "Sensores" de la base de datos del sistema, agregando estos valores a los campos de la estructura en memoria.

Podemos observar cómo casi al final del ciclo, se invoca por cada sensor el método "`InitSensor`", donde se define su tipo.

En este ciclo, resalta la sección donde se define y crea el objeto de la clase asociada al sensor (campo `G_DlLLcall`), así como la posible existencia de una ventana de calibración (campo `G_DLLadj<>""`).

Si un sensor tiene alguna de estas opciones definidas, se habilita un elemento del arreglo de objetos del menú "`MNU061`" para el instrumento, así como un objeto del arreglo del menú "`MNUCalibrar`" para la calibración.



Toda la información hasta aquí descrita, forma parte de la estructura básica requerida para lograr procesos de muestreo eficientes.

3.2.- CÓDIGO DE MUESTREO PARA TOMA DE DATOS

La toma de datos implica dos problemas a resolver; el primero es garantizar que los ciclos de tiempo sean constantes en todo momento y el segundo, poder graficar los datos de forma simultánea al muestreo, en el menor de los periodos que el equipo lo permita.

Para la primera versión, realizando pruebas, se determinó que si el desarrollo de estas rutinas se llevaba a cabo en PASCAL, no sería posible alcanzar el objetivo, ya que este lenguaje no garantiza un ciclo de tiempo constante en su proceso, además que por no ser un compilador que ejecuta lenguaje de máquina, sino el de un procesador virtual, los periodos de tiempo difícilmente podrían ser inferiores a medio segundo, por tal motivo, estos módulos fueron desarrollados en lenguaje Ensamblador.

El primer paso fue desarrollar la función de lectura de los puertos de juegos, buscando la forma de que cada muestra pudiera obtenerse en el menor tiempo posible, por lo que la base para este diseño fue la de un precálculo al seleccionar

el tipo de sensor y el periodo de muestreo, antes de iniciar el proceso de toma de lectura de datos.

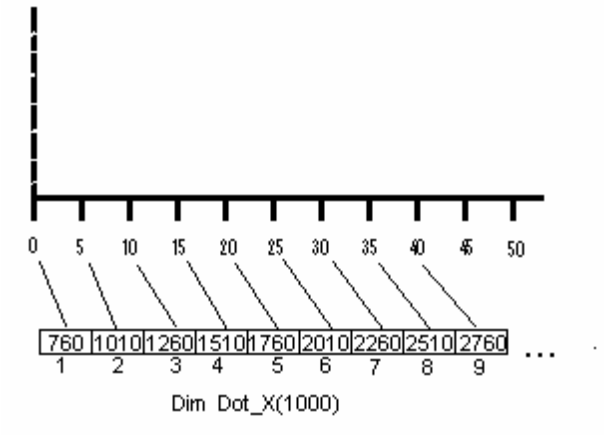
El primer precálculo consiste en determinar las coordenadas en pantalla para el eje **X**, de acuerdo con la escala que en ese momento existiera en los ejes cartesianos de la ventana principal. Estos datos quedan en un arreglo que al momento del experimento son recorridos por un apuntador.

En el diagrama que se muestra a continuación, se ejemplifica como es almacenada la información en la estructura antes mencionada.

En este caso, el muestreo se realiza cada 5 segundos, lo que se representa en la escala del eje **X** que aparece en la ventana del programa.

Cada escala que delimita un evento, tiene una separación de 250 puntos, siendo el extremo izquierdo la coordenada 760 de la ventana.

El valor inicial es almacenado en el elemento uno de la matriz "`Dot_X`", que tiene una dimensión de mil elementos.



Para completar la información en este arreglo, se utiliza el siguiente algoritmo:

```
Ini_X = 760          ' Coordenada X inicial
DeltaX = 250       ' Incremento en dots

For I=1 to  MAX_Eventos
     Dot_X(I) = DeltaX * I  + Ini_X
Next I
```

En este caso, la variable '`MAX_Eventos`' contiene el valor del número de eventos que el usuario define abarcará el muestreo, siendo mil el máximo aceptado.

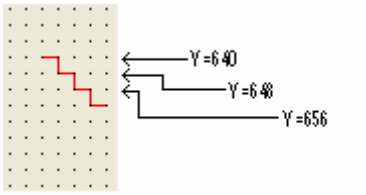
Un segundo precálculo se requiere para graficar de forma simultánea la toma de datos.

Este cálculo se basa en la escala representada en el eje **Y**, consistiendo en obtener una matriz de dos dimensiones, donde el primer elemento representa el rango de valores que entrega el puerto, mientras que el segundo valor corresponde a la coordenada dentro del eje **Y**.

Para obtener estas dos listas de valores, se realizan dos pasos:

El primero, corresponde a determinar la cantidad de puntos que forman el eje **Y**, que de acuerdo a la coordenada inicial y final consta de 6240 puntos.

Dicha cantidad, puede parecer demasiado amplia para armar una tabla, pero en realidad, no todas estas coordenadas son posicionables en un punto de la forma, por un objeto, como lo son las líneas que se trazan sobre una ventana de Visual Basic, tal como se puede ver en la siguiente figura:



El incremento para colocar un objeto es en múltiplos de ocho, lo que significa que al dividir la cantidad de puntos entre este factor, obtenemos sólo 780 puntos reales de graficación, siendo estos valores los que determinan el tamaño inicial de nuestra tabla.

El segundo paso, consiste en calcular que valor del sensor, corresponde a cada punto graficable, para lo cual se debe asignar un valor inicial (`Y_INI`) y un valor final (`Y_FIN`) en el eje **Y**.

Por ejemplo: si el sensor a utilizar es el de intensidad luminosa, asignamos un rango que va de 0 a 100 LUMEN, lo que nos da la siguiente relación:

```
  0 LUMEN   equivale a coordenada  Y=6240;
120 LUMEN   equivale a coordenada   Y=0.
```

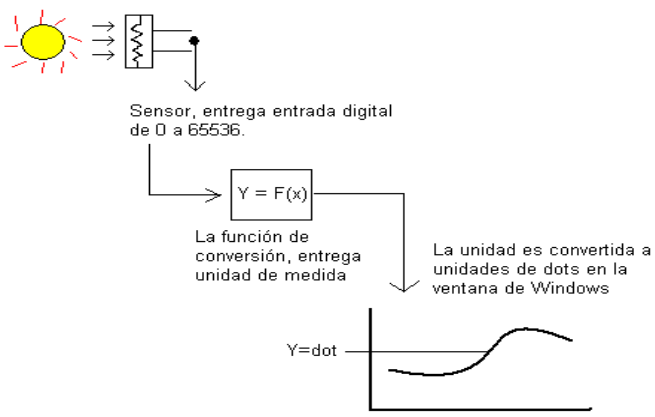
Y esto nos lleva a la siguiente ecuación lineal de transformación de valores en coordenadas de la ventana en el eje Y:

$$Y\_dot = 6240 - 52 * Y\_val$$

Despejando por la función de conversión, tenemos:

$$Y\_dot = 6240 - 52 * F(x)$$

Estas expresiones, las podemos representar en el siguiente diagrama:



Analizando la secuencia de pasos, es lógico que para cada punto que representa el eje **Y**, existe un valor asociado en la entrada del puerto donde conectamos el sensor, y esta relación la podemos construir en una tabla de un máximo de 780 elementos, con la que asociamos de forma inmediata una lectura del puerto con una coordenada en la ventana de Windows®.

Para construir la tabla con esta relación, necesitamos conocer la función inversa de la transformación de los valores del puerto, a las unidades del sensor, aplicando además, la función lineal de conversión de valores del puerto en coordenadas de pantalla.

Como ya se mencionó anteriormente, el sensor de luz, tiene la siguiente función:

$$F(x) = Y = 100 /(X+1)$$

Despejando, tenemos que la función inversa esta dada por:

$$F^{-1}(y) = X = [100 / Y] - 1$$

Esta función, se usa ahora para obtener la tabla correspondiente a los valores de entrada del puerto de juegos, contra cada coordenada graficable del eje **Y**. A continuación, se muestra un ejemplo de la tabla que deseamos obtener:

	Y (LUMEN)	INPUT	DOT_Y
1	0.30000000	332	6240
2	0.45217949	220	6232
3	0.60435897	164	6224

4	0.75653846	131	6216
5	0.90871795	109	6208
6	1.06089744	93	6200
7	1.21307692	81	6192
8	1.36525641	72	6184
9	1.51743590	65	6176
10	1.66961538	59	6168
11	1.82179487	54	6160
12	1.97397436	50	6152
13	2.12615385	46	6144
14	2.27833333	43	6136
15	2.43051282	40	6128
16	2.58269231	38	6120
17	2.73487179	36	6112
18	2.88705128	34	6104
19	3.03923077	32	6096
20	3.19141026	30	6088
21	3.34358974	29	6080
22	3.49576923	28	6072
23	3.64794872	26	6064
24	3.80012821	25	6056

Para calcular la lista anterior, usamos la tabla TablaXY de la base de datos del sistema, a la que insertamos registros con el siguiente código:

```
' cálculo el rango en puntos de la longitud del eje Y
  YRange =Abs((GVew_rec(NGrph).YDot_Fin -_
           GVew_rec(NGrph).YDot_ini))
  Y_Lim = GVew_rec(NGrph).Y_Fin

  xsql = "Select * from TablaXY"

  Set tab_wrk = db.OpenRecordset(xsql)

  For I = 0 To 780

    tab_wrk.AddNew
    tab_wrk("ID") = I
    tab_wrk("Delta_DOT") = I * 8
    K = I * 8
' Cálculo de la unidad del sensor que corresponde
' a la coordenada en dot's de la pantalla
    X1 = GVew_rec(NGrph).Y_ini
    X2 = Y_Lim - GVew_rec(NGrph).Y_ini
    Y2 = X1 + (X2 * K / YRange)
    tab_wrk("Y") = Y2
' Obtenemos el valor de la entrada del puerto que corresponde
' a la unidad del sensor, por medio de la función inversa
' que esta definida en la propiedad PortInput del objeto
    tab_wrk("INPUT") = Sensor_rec(xSenIdx).xObj.PortInput(Y2)
    tab_wrk("DOT_Y") = GVew_rec(NGrph).YDot_ini - I * 8
```

```
      tab_wrk.Update

   Next I

   tab_wrk.Close
```

Después de construir la tabla, es leída en un arreglo en memoria, eliminando posibles valores de entrada duplicados (campo INPUT), seleccionando la coordenada promedio asociada y el valor promedio de la unidad del sensor que le corresponda:

```
   nPT = 1024 * (xIdx - 1)

   xsql ="SELECT INPUT, min(ID) AS xID, avg(dot_y) AS myDOT,"
   xsql = xsql &  avg(Y) as myVAL From TablaXY"
   xsql = xsql & "   Where Y>=Y_INI and Y<=Y_TOP"
   xsql = xsql & " GROUP BY PORT_INP"
   xsql = xsql & " ORDER BY PORT_INP"

   Set tab_wrk = db.OpenRecordset(xsql)
   J = 0

   For I = 1 To 1024
     If Not tab_wrk.EOF Then
       J = J + 1
       EjeY_Plot(nPT + I).ValReal = tab_wrk("INPUT ")
       EjeY_Plot(nPT + I).ValPlot = tab_wrk("myDOT")
       EjeY_Plot(nPT + I).ValFx = tab_wrk("myVAL")
       tab_wrk.MoveNext
     Else
       EjeY_Plot(nPT + I).ValReal = EjeY_Plot(nPT + J).ValReal
       EjeY_Plot(nPT + I).ValPlot = EjeY_Plot(nPT + J).ValPlot
       EjeY_Plot(nPT + I).ValFx = EjeY_Plot(nPT + J).ValFx
     End If
   Next I

   tab_wrk.Close
```

Como se puede observar en el código, la cantidad de valores obtenidos será menor o igual a 780, pero el arreglo se completa con el último registro, hasta obtener 1024 valores. La razón de esto, es que se usará una búsqueda binaria durante el muestreo, que se facilita si el tamaño del arreglo es potencia de dos.

La localización de la coordenada Y al momento de la medición, se realiza con un algoritmo de búsqueda binaria, que encuentra el valor en un máximo de diez intentos. A continuación se lista la función que realiza esta tarea:

```
Function Dame_Lectura(ByVal Nsen As Integer,_
                      ByVal xInit As Integer,_
```

```
                        xValor As Double) As Integer

  Dim I     As Integer
  Dim st    As Integer
  Dim myPT  As Integer
  Dim Xini  As Integer
  Dim nPT   As Integer
  Dim xInp  As Double

' --- Obtengo el dato del sensor desde
‘ --- el puerto de juegos (JoyStick)

   Dame_Input Nsen, xInp

' --- Método de búsqueda binaria ---

   I = 512
   nPT = xInit

   Do While I > 0

     myPT = (nPT + I)

     If EjeY_Plot(myPT).ValReal = xInp Then
       I = -1
     Else
       If EjeY_Plot(myPT).ValReal < xInp Then
         nPT = nPT + I
       End If
       I = Int(I / 2)
     End If

   Loop

   xValor = EjeY_Plot(myPT).ValFx

‘ Se regresa la coordenada “Y” de graficación
   Dame_Lectura = EjeY_Plot(myPT).ValPlot

End Function
```

Como se observa en el código, primero se obtiene el valor de entrada del puerto de juegos, para posteriormente ubicar dicho valor en una coordenada de la ventana de la gráfica, usando una búsqueda binaria, pero además, como contamos con el precálculo de los valores de conversión de la función del sensor (variable “`ValFx`”), guardamos este valor en la variable “`xValor`”, que entró como parámetro, lo que va acumulando en un arreglo los valores del muestreo.

Finalmente, cada valor leído es almacenado en un arreglo y es hasta el final del experimento, que son transformados en la escala que corresponde al tipo de sensor utilizado, siendo almacenados en estructuras diferentes que sirven para su posterior manipulación, en conjunto del resto de las gráficas obtenidas.

Este método funciona bien para mediciones mayores o iguales a una décima de segundo, pero para las inferiores, fue necesario suspender la graficación simultánea en pantalla, por el tiempo que ésta requería.

En el proyecto desarrollado con VISUAL BASIC®, junto con la integración de llamadas a funciones de bajo nivel al sistema operativo por medio de API's, permite implementar rutinas precisas en el manejo del tiempo que requieren para su ejecución.

La metodología actualmente usada, corresponde a la original, ya que resulta la más práctica para obtener graficación en tiempo real, pero con la diferencia de que se elimina el uso de lenguaje ensamblador, que aunque es rápido y preciso, su desarrollo es complejo ante la dificultad que representa la depuración de errores.

Además de simplificar el código, otra razón de eliminar el uso de ensamblador, obedece a que los equipos PC, cuentan con reloj interno que puede ser leído por medio de API's con una precisión de una milésima de segundo y de esta forma, ya no es necesario programar procesos precisos en su tiempo de ejecución que simulan un reloj interno.

Al igual que en el proyecto original, las rutinas de muestreo se dividen en dos tipos: La primera está diseñada para realizar muestreos mayores o iguales a un segundo, para la que se utilizó el componente TIMER de VISUAL BASIC®

Esta rutina es muy simple, ya que el intervalo del TIMER se programa en cincuenta milisegundos y en cada uno de los eventos que se ejecutan, al cumplirse este tiempo, se obtiene la Fecha/Hora para determinar si se ha alcanzado el periodo de espera para la toma del siguiente dato, teniendo esto un margen de error no mayor al 5% por toma.

El código del TIMER para este tipo de muestreos se lista a continuación:

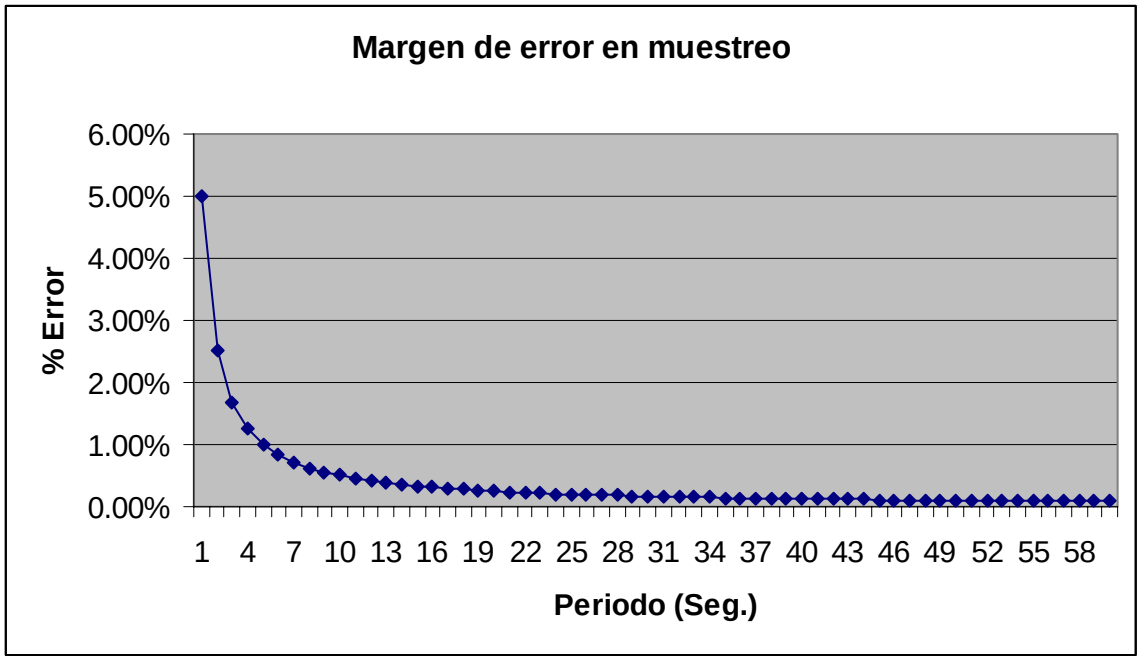
```
Private Sub Timer1_Timer()

' Timer diseñado para mediciones mayores o iguales a 1 segundo

      ‘ Obtiene la fecha/hora del momento actual
  V_Now = Now

 ‘ determina si la diferencia contra la fecha/hora de la muestra
```

```
 ‘ anterior a alcanzado el periodo establecido para el muestreo
  If DateDiff("s", V_Tme, V_Now) >= xPer Then
    V_Tme = V_Now
    Select Case V_Modo

      Case 1  ' Muestreo solo del sensor en Puerto 1
        PuntoXY1(V_pt).Top = Dame_Lectura(Sen_X, 0,_
                 EjeY_rec(V_pt1).ValReal)
        PuntoXY1(V_pt).Visible = True

      Case 2  ' Muestreo solo del sensor en Puerto 2
        PuntoXY2(V_pt).Top = Dame_Lectura(Sen_Y, 1024,_
                EjeY_rec(V_pt2).ValReal)
        PuntoXY2(V_pt).Visible = True

      Case 3  Muestreo con sensores en Puerto 1 y 2
        PuntoXY1(V_pt).Top = Dame_Lectura(Sen_X, 0,_
                EjeY_rec(V_pt1).ValReal)
        PuntoXY1(V_pt).Visible = True

        PuntoXY2(V_pt).Top = Dame_Lectura(Sen_Y, 1024,_
                EjeY_rec(V_pt2).ValReal)
        PuntoXY2(V_pt).Visible = True

    End Select

    V_pt = V_pt + 1
    V_pt1 = V_pt1 + 1
    V_pt2 = V_pt2 + 1
    If V_pt > V_per Then
      Terminar 1, Timer1
    End If
  End If

End Sub
```

El margen de error de este método, va en disminución conforme el periodo del muestreo es mayor, además que podemos considerarlo no significativo, tomando en cuenta que en este tipo de experimentos, lo importante es obtener el número de muestras esperadas en el tiempo total determinado.

Por ejemplo: obtener 60 muestras cada segundo, implica tener un periodo total de un minuto.

Como el algoritmo se basa en tomar medidas cada vez que el reloj indica que ha variado el tiempo en un segundo, contra la hora del muestreo anterior, si consideramos que el TIMER muestrea cada 50 milisegundos, el error en el periodo será de (+-)0.05 segundos, lo cual sería también el máximo error posible con respecto a la duración total del muestreo.



El segundo método para realizar mediciones, está diseñado para periodos menores a un segundo y mayores o iguales a una décima. En esta caso, ya no se usa un objeto TIMER, sino que se realiza un ciclo, en el que se está obteniendo la diferencia de milisegundos que han pasado por medio del llamado a la función de sistema `GetTickCount()`, que regresa el número de milisegundos que han trascurrido desde el boot del equipo hasta el momento.

Cuando la diferencia que se obtiene con esta función es igual al número de milisegundos establecidos en el periodo de medición, se dispara la toma del dato y la graficación del mismo. El siguiente código, ejemplifica este método:

```
Do Until V_Cta = 0

    DoEvents
    V_tme1 = GetTickCount

    If (V_tme1 - V_tme0) >= DecPer Then
           <código de toma de datos y graficación >
    End if



    V_tme0 = V_tme0 + DecPer ' V_tme1

    V_Cta = V_Cta – 1

Loop
```

El tercer caso de muestreo, es para mediciones inferiores a una a cinco décima de segundo, que es el mínimo periodo que se logró controlar usando sólo programación VISUAL BASIC®.

En este rango, no se realiza la graficación simultanea, por lo que el algoritmo es también un ciclo que mide la diferencia de tiempo con la función "`GetTickCount`", pero sólo se lleva a cabo la toma del dato, mostrando la gráfica hasta el momento de completar el muestreo.

3.3.- MANIPULACIÓN DE DATOS DE MUESTREO

Los datos recolectados en los muestreos, son almacenados en estructuras de memoria que permiten su manipulación mediante la presentación gráfica en pantalla.

La primera versión, presentaba limitaciones muy grandes en este aspecto, ya que aunque los equipos APPLE® IIe contaban con 128 KB de memoria, sólo se podía direccionar por medio de PASCAL los primeros 64KB, de los cuales, un alto porcentaje era ocupado por la máquina virtual y el código del programa objeto.

Lo anterior, limitó a que sólo se pudiera reservar el espacio suficiente para cuatro gráficas de cien datos cada una.

En la actualidad, con los alcances de memoria RAM de los equipos INTEL® (desde 64MB hasta más de 8GB), este recurso ya no resulta un problema, sin embargo, dado que el alcance didáctico no ha variado, sólo se expandió el almacenamiento a diez gráficas con un máximo de mil datos cada una.

Las estructuras mencionadas, sirven para guardar datos obtenidos por muestreo, así como el cálculo de gráficas resultado de un modelo matemático y de aplicar métodos de ajuste a los datos para suavizar su curvatura.

La primera estructura es un arreglo de diez elementos, en donde se almacena la información general que describe a una gráfica:

```
Type str_Graph_rec
  G_Nombre   As String
  G_Fecha    As Date    'Fecha de la muestra
  G_Sensor   As String  'Tipo de Sensor
  G_SenIdx  As Integer 'Apuntador a arreglo de sensores
```

```
  G_NumMed   As Long
  G_Interval As Double
  G_MAX      As Double  'Valor Máximo de la muestra
  G_MIN      As Double  'Valor Mínimo de la muestra
  G_AVG      As Double  'Valor Promedio de la muestra
  G_STD      As Double  'Desviación estándar
  G_PtDat    As Integer 'Pointer a estructura de datos
  G_Fx       As String  'Función asociada
  G_flg      As Integer '-1=disponible, 0=sin cambios,
                        ‘ 1=modificada, 100=borrar
  M_DES      As Double  'Pendiente Recta mínimos cuadrados
  B_DES      As Double  'Ordenada Origen Recta mínimos cuadrados
  NSpline    As Integer '>0:No. de Gráfica de ajuste, 0=Gráfica
                        ' no ajustada, -1:Gráfica spline
  Dy_Dx      As Integer 'Número de derivadas de Fx
End Type
```

La segunda estructura es un arreglo de 12000 elementos, en donde se almacenan los valores que resultaron de la muestra:

```
Type str_Eje_rec
   ValReal   As Double   'Valor resultado de la calibración
   ValPlot   As Long     'Coordenada en dots sobre el eje Y
   xVisible  As Boolean  ' TRUE= el valor es visible en la ventana
End Type

Global EjeX_rec(12000)  As str_Eje_rec
Global EjeY_rec(12000)  As str_Eje_rec
```

De estos 12000 valores, se destinan los primeros 10000 a almacenar hasta 1000 puntos para un máximo de diez gráficas concurrentes en memoria, mientras que los restantes 2000, son áreas de trabajo para almacenar temporalmente los datos que recolecta el proceso de muestreo, que pueden ser hasta dos simultáneamente.

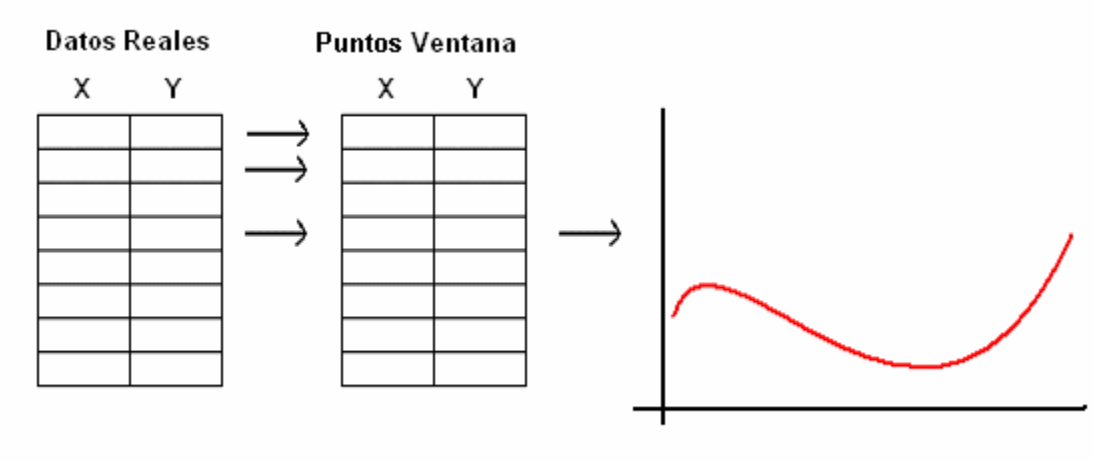
3.4.- MANIPULACIÓN DE GRÁFICAS

Para que se tenga la posibilidad de observar los datos de un muestreo desde distintos ángulos, además de ajustar valores para eliminar factores externos al fenómeno que se ha medido, el sistema incorpora diversas opciones para manipular las gráficas sobre los ejes cartesianos.

Estas opciones, en la primera versión, se desarrollaron con programación convencional en PASCAL, en las que cada vez que cambia algún valor en los ejes o en alguna gráfica, todas las se vuelven a calcular punto por punto conforme se trazan nuevamente en pantalla.

Este método, dado la velocidad del procesador, resultó lento cuando se mostraban en la pantalla el máximo de gráficas permitido.

Para solventar el problema, se definieron arreglos paralelos a los de los datos reales que almacenan las coordenadas de graficación. También recordemos que la estructura que define a una gráfica, incluye un campo que representaba una bandera para indicar cuando la gráfica ha sufrido cambios en alguno de sus datos y requiere un recálculo para su graficación.



Las rutinas originales se realizaron en Ensamblador, obteniendo una mejora del 80% en el tiempo de ejecución, pero en la actualidad, considerando la velocidad que ofrecen los procesadores INTEL®, la graficación se programó en VISUAL BASIC®, ya que sus componentes de gráficas están optimizados en sus algoritmos para el trazo en pantalla, obteniendo una velocidad de reproducción óptima.

Para hacer más ágiles las funciones de graficación, el sistema usa la siguiente estructura que lleva el control de los planos cartesianos que aparecen en la ventana principal:

```
Type str_Graph_View_rec
  Titulo   As String ‘ Encabezado del gráfico
  TituloX  As String ‘ Título de la unidad de eje X
  TituloY  As String ‘ Título de la unidad de eje Y
  X_ini    As Double ‘ Valor mínimo en eje X
  X_Fin    As Double ‘ Valor máximo en eje X
  Y_ini    As Double ' Valor mínimo en eje Y
  Y_Fin    As Double ‘ Valor máximo en eje X



 ‘ Coordenada (x,y) inferior izquierda de la ventana
  XDot_ini    As Long ‘
  YDot_ini    As Long ‘

‘ Coordenada (x,y) superior derecha de la ventana
  XDot_Fin    As Long ‘
  YDot_Fin    As Long ‘
  GrphTpe  As Integer ' No. de tipo de gráfica
```

```
  NumGrph  As Integer ' No. de gráficas activas

‘ Factores de la transformación lineal de coordenadas
 ‘ reales a dots de los ejes en la ventana principal.
  XDelta   As Double
  YDelta   As Double
  XDelPnt  As Double
  YDelPnt  As Double
  Xmax     As Double
  Ymax     As Double
End Type

Global GVew_rec(2)           As str_Graph_View_rec
```

Con esta estructura se declara un arreglo de dos elementos, de los cuales, el primero corresponde al plano cartesiano principal, mientras que el segundo, es utilizado como área de trabajo para llevar a cabo los muestreos.

En la estructura, destacan los últimos seis elementos, con los que se obtienen los factores para transformar los valores reales de las gráficas, en puntos de la ventana dentro del plano cartesiano.

Los factores se calculan en cada cambio del rango de los ejes, por medio de las ecuaciones de la siguiente rutina:

```
Private Sub Calc_Factores(ByVal NGvew As Integer,_
                          GVew_rec As str_Graph_View_rec)
 With GVew_rec

  .Xmax = .X_ini + 12# * ((.X_Fin - .X_ini) / 10#)
  .Ymax = .Y_ini + 12# * ((.Y_Fin - .Y_ini) / 10#)

  .XDelta = .X_Fin - .X_ini
  .YDelta = .Y_Fin - .Y_ini
  .XDelPnt = .XDot_Fin - .XDot_ini
  .YDelPnt = Abs(.YDot_Fin - .YDot_ini)

 End With

End Sub
```

Usando estos factores, la siguiente rutina calcula la transformación lineal de los valores de los muestreos en coordenadas de pantalla, además de calcular información adicional, como el valor máximo, mínimo, promedio, desviación estándar, además de la pendiente y la ordenada al origen de la recta de mínimos cuadrados asociada al muestreo:

```
‘ Los primeros valores son definido en un inicio como
‘ el máximo y el mínimo
```

```
  Graph_rec(xIdx).G_MIN = EjeY_rec(PT + 1).ValReal
  Graph_rec(xIdx).G_MAX = EjeY_rec(PT + 1).ValReal
  YSum = 0 ‘ variable para obtener la suma de los valores

  For I = 1 To Graph_rec(xIdx).G_NumMed

    MyX = EjeX_rec(PT + I).ValReal
    MyY = EjeY_rec(PT + I).ValReal

    YSum = YSum + MyY

 ‘ Se determinan los valores máximo y mínimo de la muestra
    If MyY > Graph_rec(xIdx).G_MAX Then
       Graph_rec(xIdx).G_MAX = MyY
    End If

    If MyY < Graph_rec(xIdx).G_MIN Then
       Graph_rec(xIdx).G_MIN = MyY
    End If

‘ Los puntos son calculados si están dentro del cuadrante
‘ que define el plano cartesiano de la ventana.
    If EnRango(NGvew,MyX) Then

      If (GVew_rec(NGvew).Y_ini > MyY) Then
         MyY = GVew_rec(NGvew).Y_ini
      End If

      If (MyY > GVew_rec(NGvew).Ymax) Then
        MyY = GVew_rec(NGvew).Ymax
      End If

      Xres = GVew_rec(NGvew).XDelPnt * ((MyX -_
           GVew_rec(NGvew).X_ini) / GVew_rec(NGvew).XDelta)
      YRes = GVew_rec(NGvew).YDelPnt * ((MyY -_
            GVew_rec(NGvew).Y_ini) / GVew_rec(NGvew).YDelta)

      EjeX_rec(PT + I).ValPlot = GVew_rec(NGvew).XDot_ini +_
            Int(Xres)
      EjeY_rec(PT + I).ValPlot = GVew_rec(NGvew).YDot_ini -_
            Int(YRes)
      EjeX_rec(PT + I).xVisible = True
      EjeY_rec(PT + I).xVisible = True
    Else
      EjeX_rec(PT + I).xVisible = False
      EjeY_rec(PT + I).xVisible = False
    End If

  Next I

' --- Obtiene el promedio
  Graph_rec(xIdx).G_AVG = YSum / Graph_rec(xIdx).G_NumMed

' ----  Obtiene la Desviación Standard = SQR( SUM( (Pi-AVG)^2 ))
  Graph_rec(xIdx).G_STD = 0
```

```
  MySum = 0

  For I = 1 To Graph_rec(xIdx).G_NumMed
    MyY = EjeY_rec(PT + I).ValReal - Graph_rec(xIdx).G_AVG
    MyY = MyY * MyY
    MySum = MySum + MyY
  Next I

  Graph_rec(xIdx).G_STD = Sqr(MySum)
‘ -- Obtiene factores M Y B de la recta de
‘ -- mínimos cuadrados (solo muestreos) -------

  If Graph_rec(xIdx).G_Fx = "" Then
    DESVIACION xIdx
  End If

End Sub
```

Con los puntos calculados, las gráficas son trazadas utilizando objetos LINE, de los cuales existen diez (`Graf01, .. , Graf10`), que son declarados cada uno como un arreglo de objetos, que incrementan su dimensión en forma dinámica cuando se crea la gráfica.

Las siguientes dos rutinas, muestran como son creados y colocados en la ventana estos objetos:

```
Sub Arma_Grafica(ByVal xIdx As Integer, ByVal NGvew As Integer,_
                 ByVal CrtLine As Boolean)

‘ Si la bandera de graficar vale -1, entonces, el elemento
‘ del arreglo de gráficas, no está asignado a un muestreo


  If Graph_rec(xIdx).G_flg = -1 Then
     Exit Sub
  End If



  For I = 1 To Graph_rec(xIdx).G_NumMed – 1
    Select Case xIdx
      Case 1
        If CrtLine = True Then
          Load Graf01(I) ‘Crea un objeto LINE
        End If
        Arma_Linea Graf01(I), I, xIdx
           .
           .
           .
      Case 10
        If CrtLine = True Then
```

```
          Graf10(I) 'Crea un objeto LINE
        End If
        Arma_Linea Graf10(I), I, xIdx
    End Select

  Next I

End Sub


' Rutina para colocar el objeto LINE en la ventana
Private Sub Arma_Linea(xLinea As Line, ByVal NumPto As Integer,_
                       ByVal NumGraf As Integer)
  With xLinea
    .Visible = EjeX_rec(MAXDATOS * (NumGraf - 1) +_
               NumPto).xVisible
    .Visible = xLinea.Visible And EjeX_rec(MAXDATOS *_
               (NumGraf - 1) + NumPto + 1).xVisible
    .Tag = Format(NumPto)
  ' Asigna las coordenadas a los extremos de la línea,
  ' utilizando el punto actual y el siguiente
    .X1 = EjeX_rec(MAXDATOS * (NumGraf - 1) + NumPto).ValPlot
    .X2 = EjeX_rec(MAXDATOS * (NumGraf - 1) + NumPto + 1).ValPlot
    .Y1 = EjeY_rec(MAXDATOS * (NumGraf - 1) + NumPto).ValPlot
    .Y2 = EjeY_rec(MAXDATOS * (NumGraf - 1) + NumPto + 1).ValPlot
  End with

End Sub
```

Este conjunto de rutinas, facilita la implementación de las siguientes opciones del sistema desarrolladas para el manejo de gráficas:

Escalas

Esta opción, permite al usuario ajustar los rangos de los ejes cartesianos de la ventana del sistema, así como los títulos que desea aparezcan en los ejes, así como en el encabezado.

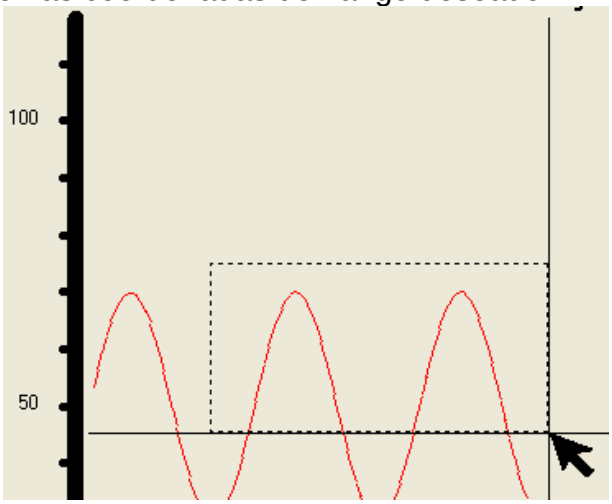
En esta ventana, el usuario, escribe en los campos correspondientes los cambios que desea efectuar sobre el rango del plano cartesiano.

Al oprimir el botón 'Aceptar', se validan dichos rangos y se llaman a las funciones de cambio de escalas y cálculo de los puntos de las gráficas de los muestreos.

Zoom

La opción de Zoom, ajusta la escala de los ejes cartesianos a un cuadrante que el usuario define con el trazo del Mouse.

Aunque el resultado es similar al de la opción "Escalas", con ésta es posible lograr un ajuste a un espacio específico del cuadrante de acuerdo a lo que ve el usuario, sin tener que conocer las coordenadas del rango deseado.



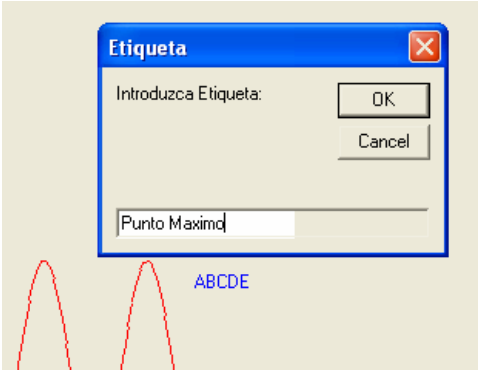
Su funcionamiento, se basa en marcar la esquina superior izquierda, arrastrar el Mouse con el botón izquierdo oprimido, liberándolo cuando se apunta a la esquina inferior derecha, lo que determina el rango de las nuevas coordenadas del plano cartesiano.

Etiquetas

Pensando en que las gráficas que aparecen en la ventana principal, puedan contener información que describan su trazo como puntos relevantes del

muestreo, se incluyó la opción de colocar etiquetas cuyo contenido lo escribe directamente el usuario.

Al activar la opción con el Mouse, el usuario indica donde colocará la etiqueta al oprimir el Mouse, lo que provoca que a continuación, se despliegue una ventana para capturar su contenido



Si se desea eliminar una de ellas, basta con apuntar con el Mouse y dar un doble click.

Ajustar

Esta opción, obtiene el valor máximo y mínimo de todas las gráficas que están en la ventana, ajustando el rango de los ejes cartesianos a estos.

Para agilizar este cálculo, el sistema utiliza las variables `G_MIN` y `G_MAX` de la estructura de gráficas, lo que le permite comparar un máximo de veinte valores para obtener los nuevos límites del eje **Y**.

El código de la rutina que realiza este cálculo es el siguiente:

```
Private Sub MNU022_Click()

  Dim myMin As Double, myMax As Double
  Dim myMinX As Double, myMaxX As Double

   For I = 1 To G_MaxGraph
  ‘ Determina si hay una gráfica asociada a este elemento
  ‘ de la matriz de gráficas

     If Graph_rec(I).G_flg >= 0 Then
        K = K + 1
        If K = 1 Then
          myMin = Graph_rec(I).G_MIN
          myMax = Graph_rec(I).G_MAX
        Else
          If myMin > Graph_rec(I).G_MIN Then
```

```
            myMin = Graph_rec(I).G_MIN
          End If
          If myMax < Graph_rec(I).G_MAX Then
            myMax = Graph_rec(I).G_MAX
          End If
        End If
      End If
    Next I

    If K > 0 Then   'Hay gráficas para ajustar
      If vbyes = MsgBox("Confirme ajustar ejes”) Then
        myMin = Int(myMin)     ' Ajusta al entero inferior
        myMax = Int(myMax + 1) ' Ajusta al entero superior

        GVew_rec(0).Y_ini = myMin
        GVew_rec(0).Y_Fin = myMax
   ‘ Cambia rango del eje Y y calcula factores de
   ‘ transformación de los puntos
        Chg_Scale 0, True
        CalcAll_Graph 0, True
      End If
    End If
End Sub
```

Información

El objetivo de esta opción, es mostrar datos cuantitativos y estadísticos de cada muestreo que se grafica en la ventana principal.

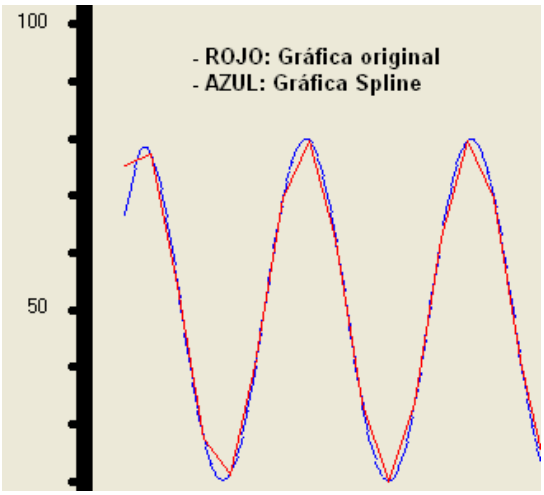
La ventana de esta opción, contiene una lista de las gráficas que están presentes. Al elegir una de ellas, se muestran los siguientes datos:

- Valor mínimo y máximo de la muestra.
- Color de las líneas de la gráfica.
- Sensor o función asociada.
- Número de mediciones.
- Intervalo de medición.
- Valor promedio.
- Valor de la desviación estándar.

Adicionalmente, tiene un botón que permite crear una nueva gráfica que aplica el método b-spline, que crea una curva más suave que la original.

Esta nueva gráfica se construye calculando diez veces más puntos que la gráfica original, teniendo como máximo mil puntos.

Las rutinas utilizadas, se obtuvieron de las bibliotecas que provee Turbo-PASCAL® de Borland® para equipos PC, siendo adaptadas al lenguaje VISUAL BASIC® de la versión actual.



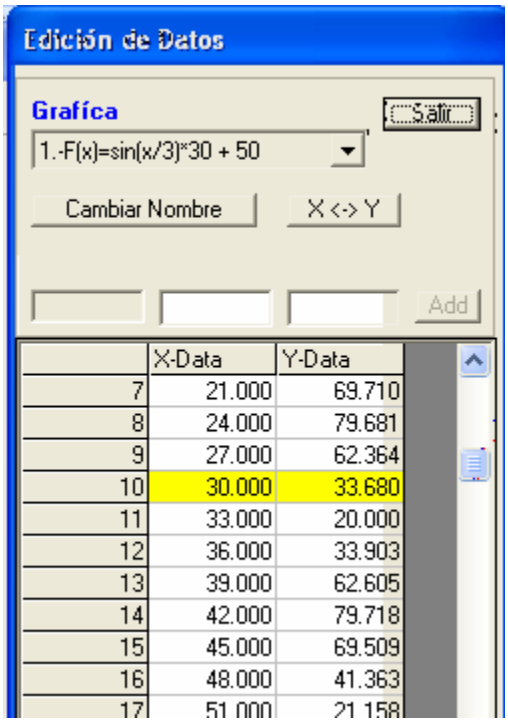
Editar

Con esta opción, el usuario hace ajustes manuales a los datos que recolectó con un muestreo.

Este tipo de ajustes, son una práctica común, ya que muchos experimentos se ven afectados por fenómenos que están ocurriendo alrededor del evento que estamos midiendo, tales como cambio en la temperatura del ambiente, tiempos de reacción de los sensores, etcétera.

La opción, presenta una ventana donde el usuario selecciona la coordenada que desea modificar, oprime doble clic sobre ella y es llevada a un grupo de campos en la parte superior que le permite hacer el cambio.

Cuando el cambio es aceptado con el botón “Add”, se sustituye el valor en la lista original y es marcado en color amarillo para resaltarlo.



Borrar

Como la cantidad máxima de gráficas que se pueden presentar simultáneamente en la ventana principal es de diez, el sistema cuenta con una opción que le permite al usuario eliminar cualquiera de ellas.

El procedimiento consiste en colocar una marca especial en el campo G_FLG de la matriz de la estructura de control de gráficas, para que una rutina revise y ponga a este elemento en estado disponible y descargue de memoria los objetos LINE que la trazan en la ventana.

A continuación se muestra el código de la rutina que realiza esta función:

```
Sub Borra_Grafica(ByVal xALL As Boolean)

 For K = 1 To G_MaxGraph

' Si el campo G_flg es igual a 100 o es una gráfica
' activa (G_flg>=0) y el parámetro xALL es TRUE,
' entonces, la gráfica es eliminada.
  If Graph_rec(K).G_flg = 100_
       Or (Graph_rec(K).G_flg >= 0 And xALL = True)_
  Then

    For I = 1 To Graph_rec(K).G_NumMed - 1

      Select Case K
```

```
        Case 1

    ‘ descarga de memoria el objeto LINE
        Unload Graf01(I)
               .
               .
               .
        Case 10
          Unload Graf10(I)
      End Select
    Next I

  ‘ Si hay una gráfica Spline asociada, se borra
    Borra_SplineRef K

  ' Se marca el espacio de la gráfica como disponible
    Graph_rec(K).G_flg = -1

  End If

 Next K

End Sub
```

3.5.- MODELOS MATEMÁTICOS

La posibilidad de introducir modelos matemáticos que permitieran comparar la definición teórica de un fenómeno, contra el resultado de un experimento, resulta fundamental para consolidar los conocimientos adquiridos en clase.

Por esta razón, al sistema se le incorporan varias opciones para el manejo de modelos matemáticos que pueden ser comparados con los muestreos obtenidos.

3.5.1.- DEFINICIÓN DE FUNCIONES MATEMÁTICAS

El sistema cuenta con un módulo para captura de funciones matemáticas que pueden ser graficadas y comparadas con las resultantes de los muestreos, permitiendo obtener desviaciones estándar, recta de mínimos cuadrados y otros valores estadísticos.

Dado que el sistema se diseñó para utilizar cualquier tipo de sensores, los modelos no podían estar predefinidos, teniendo que ser posible capturarlos y validarlos en sintaxis para posteriormente almacenarla en un tipo de estructura que facilitara su graficación.

Como parte del manejo de estas funciones, se incluyen algunas funciones trigonométricas, logaritmos y raíz cuadrada.

Para implantar de esta característica del sistema, en la primera versión, se desarrolló un módulo de análisis semántico y sintáctico que entregaba la expresión reducida a notación prefija. Esta salida, era evaluada con una rutina basada en el manejo de una pila.

El uso de estas rutinas, resultó muy eficiente, pero tenía algunas limitantes, por lo que para la versión actual, se buscó mejorar el alcance del analizador, pero al mismo tiempo reduciendo la complejidad del código utilizado.

La solución a esto, fue calcular estas funciones dentro de sentencias SQL que son evaluadas por el motor de ACCESS®.

Esto simplifica significativamente el módulo de evaluación de expresiones algebraicas, ya que la ejecución de una sentencia SQL, incluye la evaluación sintáctico/semántica, así como la evaluación de la misma, todo esto, en tiempo de ejecución.

Esta ejecución de sentencias SQL, se realiza sobre la tabla de trabajo "FxWork" de la base de datos del sistema, cuya estructura es la siguiente:

```
Create table FxWork(
  ID  Integer,     ‘ consecutivo entero
  X   double,      ‘ Valor de la coordenada x
  Y   double,      ‘ Valor de la coordenada Y, resultado de F(x)
  Mx double, ‘ Pendiente de la recta de mínimos cuadrados
  Bx double  ‘ Ordenada origen de la recta de mínimos cuadrados
)
```

Con esta tabla, es posible construir sentencias SQL, que contenga expresiones del tipo Y = F(x).

Para explicar a detalle este método, veamos un ejemplo para obtener los datos de la gráfica Y= SIN(X/10)*20 + 50 en un rango de 0 <= x <= 20.

El primer paso, es cargar en la tabla tantos registros como puntos deseamos para la gráfica. Para este ejemplo calcularemos veinte puntos (x en { 1,2,3..,20}), lo que hacemos con esta rutina:

```
TotX = 0

Dx = Val(txt_Dx.Text)  ‘para el ejemplo, Dx=1

Nmed = Val(txt_Nmed.Text) ‘para el ejemplo Nmed = 20

Set tab_wrk = db.OpenRecordset(“Select * from FxWork”)
```

```
For I = 1 To Nmed
  TotX = TotX + Dx
  tab_wrk.AddNew
  tab_wrk("ID") = I
  tab_wrk("X") = TotX
  tab_wrk("Y") = 0
  tab_wrk.Update
Next I

tab_wrk.Close
```

El código, crea estos registros de la siguiente tabla:

ID	X	Y
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0
9	9	0
10	10	0
11	11	0
12	12	0
13	13	0
14	14	0
15	15	0
16	16	0
17	17	0
18	18	0
19	19	0
20	20	0

El siguiente paso, es construir la sentencia SQL “`Update`”, que actualice el campo “Y” con la función escrita por el usuario en el campo texto “`txt_Fx`”. A continuación, se describe el código que efectúa esta tarea:

```
xsql = "Update FxWork set Y=" & txt_Fx.Text

On Error Resume Next

  Err.Clear
```

```
db.Execute xsql

If Err.Number <> 0 Then
  MsgBox "Error en definición de Función:" & Err.Description
  On Error GoTo 0
  Exit Sub
Else
  On Error GoTo 0
  Crea_Grafica V_Idx
  MsgBox "Gráfica generada correctamente"
End If
```

Lo que nos dará como resultado la siguiente tabla:

ID	X	Y
1	1	***51.9967***
2	2	***53.9734***
3	3	***55.9104***
4	4	***57.7884***
5	5	***59.5885***
6	6	***61.2928***
7	7	***62.8844***
8	8	***64.3471***
9	9	***65.6665***
10	10	***66.8294***
11	11	***67.8241***
12	12	***68.6408***
13	13	***69.2712***
14	14	***69.7090***
15	15	***69.9499***
16	16	***69.9915***
17	17	***69.8333***
18	18	***69.4770***
19	19	***68.9260***
20	20	***68.1859***

Ahora, sólo resta abrir la tabla con un recordset para obtener las coordenadas (x,y) que se grafican en la ventana de la aplicación.

Si la función F(x) , introducida por el usuario, estuviera mal escrita, la ejecución de la sentencia "`Update`", arrojaría un error que será desplegado en un mensaje para que la función sea corregida.

3.5.2.- MÉTODOS NUMÉRICOS PARA CÁLCULO DE DERIVADAS

Hemos descrito como definir modelos comparando las gráficas que resultan de un muestreo, con el fin de conocer la diferencia que existe entre un fenómeno real y el modelo teórico que lo fundamenta.

Para conocer otros aspectos del fenómeno real, relacionados con la velocidad de cambio que sufre a lo largo del tiempo del muestreo, es importante calcular la derivada del modelo.

En la implementación de esta función, se utilizó un algoritmo sencillo de cálculo numérico de la derivada, considerando que en una gran medida, los modelos utilizados corresponden a funciones suaves y continuas.

Por medio de este cálculo, podemos encontrar el valor de la derivada en un punto, así como graficar la función resultante de la misma.

Primero describiremos como se obtiene la derivada en un punto de la gráfica. Tomemos como ejemplo la siguiente función:

$$F(x) = X*X/10 + 20$$

Esta función cuadrática, cuya curva es una parábola, tiene como derivada:

$$F'(x) = X / 5$$

Ahora, si deseamos obtener el valor de la derivada, para X = 5, tenemos:

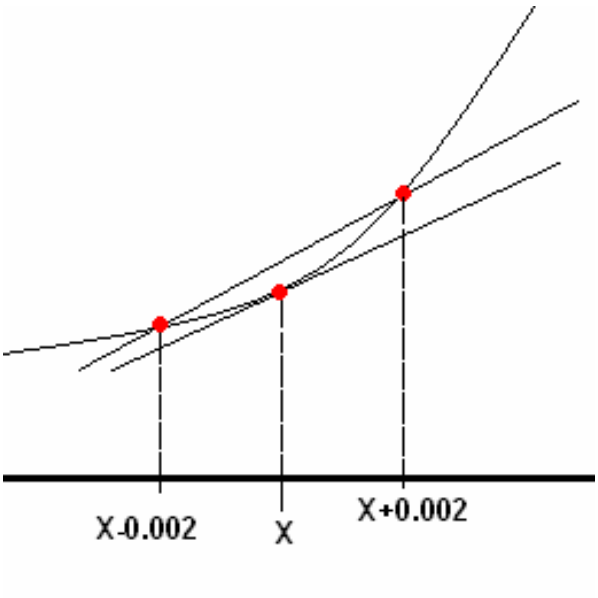
$$F'(5) = 5/5 = 1$$

Este resultado, representa la tangente de la recta que toca a la gráfica en el punto indicado y corresponde a la dirección en que está orientada la función en ese punto.

Considerando esto, nuestro algoritmo se basa en obtener una recta cuya pendiente sea lo más cercana posible al valor de la derivada. Para ello, obtenemos la recta que pasa por los puntos X1= 5 – 0.002 y X2 = 5 + 0.002, efectuando los siguientes cálculos:

$$F(X1) = (4.998)*(4.998) /10 + 20 = 22.4980$$
$$F(X2) = (5.002)*(5.002)/10 + 20 = 22.5020$$

En el siguiente diagrama, podemos observar como los cálculos efectuados, son una aproximación a la recta asociada con el valor de la derivada del punto que nos interesa:



Calculamos la pendiente entre estos dos puntos:

(F(X1) – F(X2))/(X2 – X1) = (22.5020 – 22.4980)/(5.002 – 4.998)

= 0.004 / 0.004 = 1

Como se observa, la recta obtenida entre los puntos X1 y X2, es paralela a la derivada de la función. En este caso, la siguiente tabla muestra como el resultado de ambas pendientes en un rango de 0 a 10, es siempre el mismo:

X	F'(x)	X1	X2	F(x2) – F(x1)/(x2-x1)
0.000	***0.000***	-0.002	0.002	***0.000***
1.000	***0.200***	0.998	1.002	***0.200***
2.000	***0.400***	1.998	2.002	***0.400***
3.000	***0.600***	2.998	3.002	***0.600***
4.000	***0.800***	3.998	4.002	***0.800***
5.000	***1.000***	4.998	5.002	***1.000***
6.000	***1.200***	5.998	6.002	***1.200***
7.000	***1.400***	6.998	7.002	***1.400***
8.000	***1.600***	7.998	8.002	***1.600***
9.000	***1.800***	8.998	9.002	***1.800***
10.000	***2.000***	9.998	10.002	***2.000***

Hay que resaltar que esta relación de igualdad, no siempre se tendrá para cualquier función, ya que en muchas encontraremos diferencia, pero para efectos prácticos, esta desviación no será significativa. El mismo método, se emplea para calcular la gráfica de la derivada.

A continuación, se muestra el código de la rutina que efectúa estos cálculos, que usa las tablas de la base de datos, para evaluar la función que introdujo el usuario:

```
Private Sub DERIVA_Grph(ByVal MyGph As Integer,_
                        ByVal SI As Boolean)
'MyGph: Numero de gráfica a derivar
  Dim X1 As Double, Y1 As Double
  Dim X0 As Double, Y0 As Double
  Dim X As Double, Y As Double, R As Double
  Dim M As Double, Dx  As Double
  Dim st As Integer, X2 As Integer, Y2 As Integer, I As Integer
  Dim PT As Integer, N As Integer
  Dim Ava   As Double, DAva As Double
  '
  PT = MAXDATOS * (MyGph - 1)
  db.Execute ("Delete from FxWork")
  Set tab_wrk = db.OpenRecordset("Select * from FxWork")
  Dx = Dx * 0.004
  '
 With tab_wrk

‘inserta N registros en la tabla de trabajo FxWrk, que
‘corresponden a los puntos que tiene la gráfica de la función
‘usando a los campos A y B como áreas e paso para X1 y X2
  For I = 1 To Graph_rec(MyGph).G_NumMed
    .AddNew
    .Fields("ID") = I
    .Fields("A") = EjeX_rec(PT + I).ValReal + 0.002
    .Fields("B") = EjeX_rec(PT + I).ValReal - 0.002
    .Fields("D") = Dx) ' DX= (X2 – X1)
    .Update
  Next I
  '
  .Close
'
 End With
'
 With db
' ---- Obtiene Y1 = F(X1)
  .Execute "Update FxWork set X=A"
  .Execute "Update FxWork set Y=" & Graph_rec(MyGph).G_Fx
'
' --- Obtiene Y0 = F(X2)
  .Execute "Update FxWork set X=B"
  .Execute "Update FxWork set Fx=" & Graph_rec(MyGph).G_Fx
' --- Obtiene la pendiente de función en punto específico
  .Execute "Update FxWork set Y=(Y-Fx)/D"
```

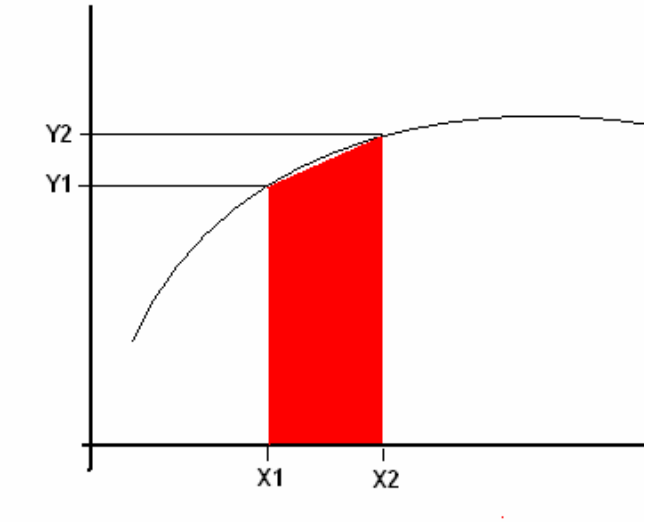
```
 End With
‘ --- Lee el resultado y guarda los valores de la nueva gráfica
  xsql ="Select * from FxWork order by id"
  Set tab_wrk = db.OpenRecordset(xsql)

 With tab_wrk
  For I = 1 To Graph_rec(MyGph).G_NumMed
    EjeY_rec(PT + I).ValReal = .Fields("Y")
    .MoveNext
  Next I
  .Close
 End With

End Sub
```

3.5.3.- MÉTODOS NUMÉRICOS PARA CÁLCULO DE INTEGRALES

Al igual que la derivada, la integral de una función, tiene un significado útil en los modelos de fenómenos físicos y químicos entre otros.

El sistema, cuenta con una opción para calcular la integral definida entre un rango que establece el usuario.

Este cálculo se lleva a cabo con un método numérico, basado en el área del tetraedro que se forma entre dos puntos de la curva de una función, como se muestra en el siguiente diagrama:



Sabemos que este método es menos exacto que otros que se pueden implementar con el apoyo de derivadas o de curvas de grado N, pero para fines prácticos del sistema, éste resultó bastante eficiente.

La rutina de este algoritmo, realiza cuarenta divisiones en el rango que indica el usuario, apoyándose en las tablas de la base de datos, para realizar el cálculo de estas áreas, sombreando adicionalmente la parte entre la curva y el eje **X**, para efectos didácticos.

A continuación se describe el código utilizado:

```
Private Sub INTEGRA(ByVal MyGph As Integer,_
      ByVal LimA As Double, ByVal LimB As Double,_
      ByVal N_INT As Integer, ByVal N As Integer,_
      ByVal SI As Boolean)

Dim Dx As Double, R As Double, R1 As Double
Dim A As Double, B As Double, PtoRes As Double
Dim X1 As Long, Y1 As Long
Dim X2 As Long, Y2 As Long
Dim FC As Double, I As Integer, st As Integer
Dim A1 As Double, K As Integer, PT As Integer

  A = LimA
  B = LimB

  PT = MAXDATOS * (MyGph - 1)
  Graph_rec(MyGph).Dy_Dx = -10

‘ Obtiene el ancho de cada porción a calcular
  Dx = (B - A) / N
‘ Determina el signo del resultado
  If A < B Then
    FC = 1
  Else
    R = A
    A = B
    B = R
    Dx = -Dx
    FC = -1
  End If

  A1 = A

  K = 1

  db.Execute "Delete from FxWork"

‘ Inserta en la tabla tantos registros como
‘ secciones de área a calcular
  Set tab_wrk = db.OpenRecordset("Select * from FxWork")

  Do While (A1 < B)
    K = K + 1
    tab_wrk.AddNew
    tab_wrk("ID") = K
    tab_wrk("A") = A1
    tab_wrk("B") = A1 + Dx
```

```
    tab_wrk.Update
    A1 = A1 + Dx
  Loop

  tab_wrk.Close

'-- Calcula Y1
  db.Execute "Update FxWork set X=A"
  db.Execute ("Update FxWork set Y=" & Graph_rec(MyGph).G_Fx)

'-- Calcula Y2
  db.Execute "Update FxWork set  X=B"
  db.Execute ("Update FxWork set Fx=" & Graph_rec(MyGph).G_Fx)

  V_RES = 0
  R = 0
  st = 0
  I = 0
‘ Recupera los datos calculados y obtiene el área
  Set tab_wrk = db.OpenRecordset("Select * from FxWork")

  Do While (Not tab_wrk.EOF) And (st = 0)

    A = tab_wrk("A")
    B = tab_wrk("B")
    R = tab_wrk("Y")
    R1 = tab_wrk("Fx")

    PtoRes = (R + R1) * Dx / 2#
    V_RES = V_RES + PtoRes

    I = I + 1
‘ Calcula las coordenadas en la ventana.
    Calc_PtoXY A, R, 0, X1, Y1
    Calc_PtoXY B, R1, 0, X2, Y2

‘Crea el objeto shape que simula el área entre el eje X
‘y la curva de la función
   On Error Resume Next
    Load Lab_MNU.Shape2(I)
   On Error Goto 0
    flg_alt = I
    On Error GoTo 0
    Lab_MNU.Shape2(I).Top = Y1
    Lab_MNU.Shape2(I).Height = Abs(GVew_rec(0).YDot_ini - Y1)
    Lab_MNU.Shape2(I).Width = X2 - X1
    Lab_MNU.Shape2(I).Left = X1
    Lab_MNU.Shape2(I).Visible = True
    DoEvents
    tab_wrk.MoveNext

  Loop

  tab_wrk.Close
‘Deposita el resultado en la variable que recibe como parámetro
```

```
  V_RES = V_RES * FC

End Function
```

3.6.- ALMACENAMIENTO DE INFORMACIÓN

Guardar la información obtenida de los muestreos, se consideró relevante para efectos didácticos, por lo que desde la primera versión del sistema, exiten opciones para guardar estos datos en distintos tipos de archivo, así como la posibilidad de exportarlos a otras aplicaciones.

3.6.1.- CONCEPTO DE PROYECTO Y SU ESTRUCTURA DE BASE DE DATOS

En la primera versión, las gráficas eran almacenadas en archivos planos que podían ser transportadas a otros equipos para ser leídos por la misma aplicación, así como su exportación a la base de datos de APPLEWorks, producto muy popular para el manejo de información en ese tipo de equipos.

En la versión actual, la información es almacenada en bases de datos ACCESS® que pueden ser migradas con mucha facilidad a productos como MS-OFFICE® y a otros manejadores de bases de datos como SQLserver®, ORACLE®, entre otros.

La elección de ACCESS®, se tomó considerando que es un formato manejado directamente por los drivers de MSJET® incorporados en Windows®, eliminando esto la necesidad de usar algún otro programa servidor para el manejo de las bases de datos como sería SQLsever®.

Esta nueva forma de guardar datos, permite no tener límites en la cantidad de datos a almacenar, ya que el mismo motor de bases de datos es quien de controla las estructuras que contienen los datos, lo que además, facilita su explotación por medio de herramientas como SQL.

El modelo usado para la base de datos, consiste en agrupar más de una gráfica en un conjunto al que denominamos ‘proyecto’, que por lo general, serán las gráficas de una práctica o de un grupo.

En este nuevo modelo, se ha diseñado una estructura de base de datos relacional que mantiene una integridad referencial entre tablas, para garantizar la consistencia de los datos.

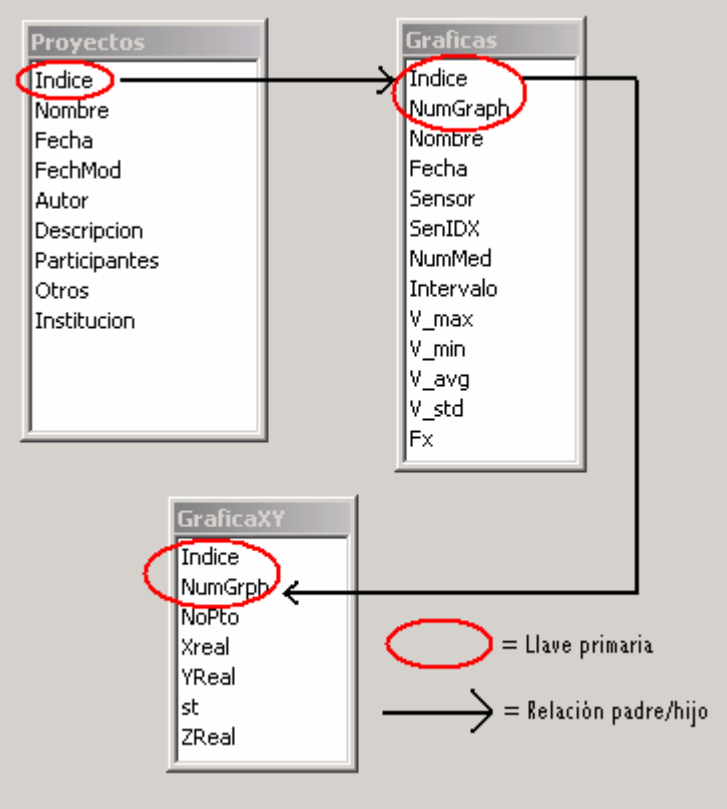
La estructura consta de las siguientes tres tablas:

- Proyectos: Contiene la descripción de los datos que identifican a un grupo de gráficas, conteniendo datos generales como nombre de la práctica fecha de realización y cantidad de gráficas asociadas.

- Gráficas: Almacena la información que describe a cada gráfica como es el sensor utilizado, valores máximo, mínimo y promedio; cantidad de puntos, entre otros.

- GráficaXY: Contiene los valores de las coordenadas X y Y de cada gráfica.

Las tablas mencionadas, mantienen una relación de integridad referencial, del tipo padre/hijo en relación a sus llaves primarias y foráneas, como lo muestra el siguiente diagrama:



En el diagrama podremos observar que los registros de la tabla ‘Proyectos’, se identifican de forma única por un número consecutivo (campo ’Índice’).

Así mismo, el diagrama muestra como cada registro de la tabla ‘Proyectos’, está relacionado con al menos un registro de la tabla ‘Gráficas’, la cual se identifica por el número de proyecto (índice) al que pertenece y un número consecutivo dentro de la tabla (NumGrph).

De la misma manera, cada gráfica, está asociada con la serie de coordenadas que la definen en el plano cartesiano, que están almacenadas en la tabla ‘GraficaXY’ y la llave primaria de esta tabla es el número de proyecto (índice), el número de gráfica (NumGrph) y un consecutivo que establece la secuencia de los puntos (NoPto).

La ventaja de ofrecer de esta forma la información al usuario final, es que puede utilizarla para explotarla a su gusto, ya que puede combinar datos provenientes de diversos muestreos, modelar los datos en paquetes de graficación más avanzada, etcétera.

3.6.2.- EXPORTAR E IMPORTAR DATOS DE PROYECTOS

Originalmente, el formato usado por la aplicación para guardar datos, llevaba implícito la posibilidad de exportación a otros paquetes.

El primer paquete al que se exportó información fue 'APPLE Works'®, que integraba una hoja de cálculo, un procesador de palabra y una base de datos, a la que era posible integrar datos provenientes de archivos planos con registros cuyos campos vinieran separados por comas.

Al migrar a equipos PC, esta estructura de archivos planos siguió siendo práctica para integrar datos en paquetes de esa época como LOTUS123® y DBASE®.

Esta opción, construía un archivo con la estructura de DBASE®, la cual era leída directamente por este paquete y resultaba un formato compatible para integrarlo en la mayoría de los programas comerciales de la época.

En la actualidad, el nuevo formato son las bases de datos de ACCESS®, las que podemos considerar un formato de fácil integración a otros paquetes como MS-OFFICE®, ya que incorpora la opción de importarlos.

Sin embargo, para esta versión, se buscó que el proceso sea más sencillo para que el usuario no necesite entiender conceptos de base de datos para pasar su información a otras aplicaciones.

Por esta razón, se incorpora un módulo específicamente para la migración de datos.

Esta opción, permite establecer el nombre del archivo a crear, así como el caracter delimitador de cada columna.

Los caracteres delimitadores que se pueden elegir, son: el tabulador (TAB), que es el más común para incorporar datos a hojas de cálculo como EXCEL®. También se puede elegir la coma y el PIPE (“,” y “|”), los cuales son los más apropiados para exportar datos a cualquier base de datos u hoja de cálculo.

4.- DESARROLLO DE MATERIAL DIDÁCTICO DE APOYO

Uno de los objetivos de este proyecto es cambiar la perspectiva de los alumnos al comparar un modelo teórico, contra el resultado del muestreo de un fenómeno, ya sea físico, químico o de cualquier otra área de la currícula.

Por esta razón, fue necesario diseñar prácticas que reflejen esta idea y así aprovechar al máximo los beneficios del producto.

Para esta tarea, se integró un equipo de trabajo con personal docente que tuviera amplia experiencia en la enseñanza, así como un extenso conocimiento de las ciencias para diseñar las prácticas con el enfoque propuesto.

Es así como se incorporan diversos profesionales del campo de la física, la química, además de gente con experiencia en nuevas técnicas de la enseñanza.

Para lograr este material, hubo un arduo debate de diversos temas de la currícula, como el concepto de calor, temperatura y luz, entre otros.

4.1.-OBJETIVO DE LAS PRÁCTICAS

El producto de software desarrollado no es ningún sustituto de los programas académicos, sólo es una herramienta para apoyar el desarrollo de los actualmente implementados, sobre todo a nivel de educación media y media superior, en donde se integra a materias como química, física y biología.

El alcance del producto, no solo pretende apoyar a estos niveles, ya que también puede ser usado en la educación pre-escolar y básica, en donde el objetivo es desarrollar las habilidades para entender el entorno que nos rodea, comprendiendo de manera detallada, el desarrollo de algunos fenómenos relacionados con los cambios atmosféricos y el cambio de estado de sustancias comunes como el agua.

Las prácticas desarrolladas en base a estos objetivos, fueron aplicadas con gran éxito en las pruebas piloto que se hicieron en varias escuelas públicas y privadas del Distrito Federal, demostrando que son de gran utilidad para apoyar la comprensión de estos fenómenos.

Originalmente las prácticas sólo se diseñaron para los sensores de luz y temperatura, pero se pueden diseñar para otros como pH, conductividad y movimiento.

En la actualidad, existe bastante material de referencia que se puede obtener de Internet o en libros que se adquieren de empresas que han desarrollado productos similares, pero también es importante mencionar que esta es una herramienta que permite desarrollar paralelamente la creatividad de los profesores, estimulándolos a que diseñen sus propias prácticas en base a sus necesidades y adaptándose al entorno social al que pertenecen.

En la siguiente sección, se describen algunas prácticas tipo, que ejemplifican el alcance del producto.

Es importante resaltar, que en estas prácticas, se buscó emplear materiales de fácil adquisición, económicos y de bajo riesgo para la integridad física de los alumnos.

4.2.-EJEMPLO DE PRÁCTICAS TIPO

Ya que este documento no pretende abarcar mas allá de lo refierente al diseño y desarrollo del software de este proyecto, en esta sección, sólo se describen cinco prácticas, de las cuales, dos están orientadas al nivel pre-escolar y básico, mientras que las restantes, aplican para educación media y media superior, abarcando temas de física y química, utilizando los sensores de temperatura y luz.

Estas prácticas, no son las originalmente diseñadas para el producto, pero se obtuvieron de la página de Internet de la empresa Vernier® , donde desarrollan hardware y software similar al expuesto en esta tesis.

Los textos fuerón traducidos del inglés al español, conservando su formato original.

Las prácticas presentadas en la sección de anexos son:

- Tocando las cosas. Comparando aislamiento térmico.
- Estados de cambio del agua.
- Evaporación y atracción intermolecular.
- Determinando el radio molecular en una reacción química.
- Reflexión y absorción de la luz.

CONCLUSIONES

Como se mencionó al inicio, el proyecto evoluciona desde su primera versión en APPLE en el año de 1987, hasta la actual en arquitectura INTEL.

El sistema se utilizó en diversos ambientes educativos del Distrito Federal, como el 'Colegio Hebreo Tarbut'®, El 'Colegio Hebreo Sefaradí'®, además de algunas instituciones públicas como la Secundaria Albert Einstein.

La herramienta, demostró que es un excelente auxiliar para que los alumnos comprendan la naturaleza de algunos fenómenos tan comunes, como es la sensación de frío y calor y como se relaciona esto con los métodos cuantitativos para su medición.

Su manejo siempre resultó muy intuitivo, requiriendo de sólo algunas instrucciones para que tanto maestros como alumnos, pudieran entender todas las funciones con las que cuenta este programa, ya que desde sus primeras versiones, se manejó un ambiente gráfico a base de ventanas como a las que estamos acostumbrados a ver en el ambiente de Windows® y que resulta muy amigable para el usuario.

Por otra parte, aunque se sabe que las posibilidades del programa podrían crecer ampliamente si se empleara hardware más sofisticado, esto significaría incrementar el costo del mismo, limitando esto el alcance que se busca de poder ser adquirido por cualquier institución educativa del país.

Una versión básica de este programa con los sensores de temperatura y luz, puede ofrecerse a un costo muy reducido, sin embargo, si se desea incorporar otros sensores, hay que considerar que algunos de ellos pueden tener costos muy elevados, como sería el caso de los de pH o electromagnetismo, pero en nuestra experiencia, observamos que con solo los sensores básicos, se puede cubrir una amplia gama de temas de la currícula de las materias de ciencias.

Respecto a la organización que originalmente promovió el desarrollo de este programa, esta abandonó el desarrollo del proyecto desde la primera versión para arquitectura PC, pero actualmente, se tiene el interés de volver a promover este proyecto, buscando adaptarlo a las futuras arquitecturas de computadoras, en donde se espera poder realizar muestreos más rápidos, así como explotar al máximo las capacidades para presentar gráficas con mejor presentación.

Se tienen muchas ideas para expandir el producto, entre ellas, aprovechar la incorporación de estos equipos en ambientes de redes locales, lo que permitiría construir versiones a nivel servidor, que puedan estar atendiendo de forma simultánea solicitudes de muestreo de varias estaciones cliente.

Anexo I
Prácticas de Laboratorio

Práctica 1: Tocando las cosas
Comparando aislamiento térmico

Cuándo tienes una bebida caliente, ¿la colocas en un vaso de unicel para mantenerla tibia?. SI hace frío en el exterior, ¿te colocas un sombrero para mantenerte tibio?. En ambos casos, el vaso de unicel y el sombrero son tipos de materiales aislantes que conservan el calor.

Otros materiales como el metal, son buenos conductores, o sea que transfieren el calor de una fuente muy rápido. Tu puedes experimentar la conductividad del metal, si tocas una cuchara que ha estado dentro de un plato con sopa caliente o cuando te sientas en una banca de metal que ha estado expuesta al sol.

Objetivos:

Comparar los cambios de temperatura, cuando a dos diferentes tipos de recipiente, se les añade agua fría o caliente.

Determinar que materiales son mejores conductores y cuales mejores aislantes.

Materiales:

Computadora con programa LabWin

Al menos 1 sensor de temperatura

2 tasas o vasos de distintos materiales (unicel, cerámica, papel o plástico)

Masking tape y un marcador resistente al agua

Procedimiento:

1. Predice cual de los recipientes será el mejor aislante y márcalo con el número 1. De la misma forma, marca el mejor conductor con el número 2

2. Conecta el sensor de temperatura a la interfase del sistema en la computadora.

3. Escribe una hipótesis de lo que ocurrirá con el agua en cada recipiente. Por ejemplo, "Si pongo sopa caliente en el mejor conductor, creo que se enfriará muy rápido". "Si coloco la sopa caliente en el mejor aislante la temperatura no descenderá".

Hipótesis:

Si coloco agua fría en el recipiente que es el mejor conductor, Yo pienso que la temperatura

_____

Si coloco agua fría en el recipiente que es el mejor aislante, Yo pienso que la temperatura

_____

4. Llena el recipiente 1 hasta la mitad con agua fría y coloca el sensor de temperatura dentro de él, evitando tocar el fondo.

5. En la computadora, prepara tu experimento para tomar 60 muestras cada 10 segundos y comienza a tomar los datos.

6. De la gráfica resultante, anota el valor máximo y mínimo que se registro. Considera que los primeros datos de la gráfica, son para estabilizar al sensor, por lo que no debes tomarlos en cuenta.

Realiza este mismo experimento para el recipiente dos.

Análisis de los datos

Tabla de Datos				
Vaso	Tipo de material	Temperatura Inicial	Temperatura Final	Cambio de Temperatura
1		°C	°C	°C
2		°C	°C	°C

En la tabla donde anotaste los resultados, obtén la diferencia entre la temperatura inicial y la final, determinando si esta aumentó o disminuyó, Anota el resultado en la tabla.

¿Tu hipótesis sobre los cambios de temperatura, fueron correctos? _____

En esta actividad, ¿cuál fue el mejor aislante? _____
y cual el mejor conductor:

_____

Si tu quieres mantener una taza de chocolate caliente, ¿cuál material usarías? _____

¿Por qué? _____

Si tu quieres mantener una bebida fría, ¿cuál material usarías? _____

¿Por qué? _____

Práctica 2: Estados de cambio del Agua

Como se sabe, si ponemos hielo en una bebida, esto hará que su temperatura baje. Al mismo tiempo que el hielo comienza a enfriar la bebida, este comienza a disolverse en el líquido.

Considerando lo anterior, ¿Qué crees que ocurra si colocas un hielo en agua caliente?. ¿El hielo se devolverá en el agua en un rango de tiempo diferente al que tomaría poniéndolo en agua fría?.

En esta actividad, descubriremos el comportamiento del hielo al disolverse en agua.

OBJETIVO:

Aprender como varia el tiempo de disolución de hielo en diferentes ambientes.

Aprender las propiedades del agua en estado sólido y líquido.

MATERIALES:

Computadora con programa LabWin

1 sensor de temperatura

2 recipientes. Pueden ser tazas de cerámica o vasos de unicel

Cubos de hielo

Agua a temperatura ambiente

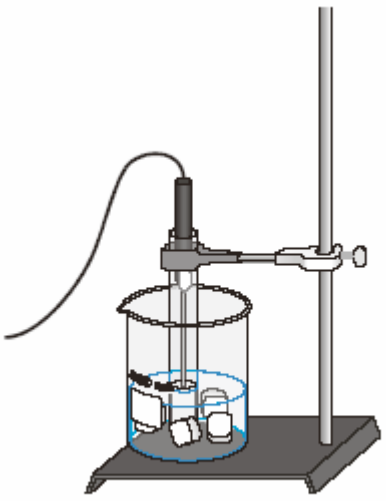


Figure 1

PROCEDIMIENTO

- Escribe tu hipótesis de que ocurrirá con el hielo y la temperatura del agua cuando colocas el hielo dentro del agua a temperatura ambiente:

  Si coloco el hielo en el agua, _____

  .____

  Y la temperatura del agua _____

  .____

- Conecta el sensor de temperatura a la interfase del programa LabWin
- Toma uno de los recipientes y llena la mitad con agua.
- En el segundo recipiente, coloca dos cubos de hielo.
- Introduce el sensor de temperatura en el recipiente con agua, asegurándote de no tocar el fondo con él.
- En el programa LabWin, programa un muestreo de 500 mediciones cada 10 segundos. Inicia el muestreo.
- Agrega un cubo de hielo al agua. Procura que el sensor, se mantenga solo tocando el agua.
- Observa el comportamiento de la gráfica, mientras se disuelve el hielo. Escribe algunos comentarios de lo que observas.
- Cuando se haya disuelto el primer hielo, agrega un segundo cubo al agua. Escribe un comentario de lo que sucede ahora.
- Cuando el segundo cubo de hielo se ha disuelto totalmente, detén el muestreo y salva la gráfica.

ANALISIS DE DATOS:

¿Tu hipótesis fue correcta?, explica por que si o porque no: _____

._____

._____

Anota en la tabla inferior, la temperatura del agua al colocar el primer hielo. Utiliza el campo de "Temperatura Inicial".

Anota en el cuadro "Temperatura Final", la temperatura del agua al disolverse el primer hielo.

Anota de igual manera, los datos de temperatura inicial y final del segundo hielo.

Sobre los campos "Cambio de Temperatura", anota el valor que resulta de restar la temperatura inicial de la final.

Tabla de Datos			
Cubo de Hielo	Temperatura Inicial	Temperatura Final	Cambio de Temperatura
1	°C	°C	°C
2	°C	°C	°C

Observa la grafica e indica que cubo de hielo se disolvió más rápido:_____

¿Por qué uno de los hielos tardó más en disolverse? _____

._____

Práctica 3: Evaporación y atracción intramolecular

En este experimento, los sensores de temperatura son colocados en diversos líquidos. La evaporación ocurre cuando el sensor es removido del líquido.

La evaporación es un proceso endotérmico, cuyo resultado, es un descenso en la temperatura.

La magnitud de este descenso de temperatura, al igual que la viscosidad y la temperatura de ebullición, se relacionada con la atracción intermolecular.

En este experimento, estudiaremos los cambios de temperatura producidos por la evaporación de diversos líquidos y su relación con las fuerzas intermoleculares de atracción. También usaremos el resultado para predecir los cambios de otros líquidos.

En este experimento usaremos dos tipos de componentes orgánicos, alcoholes y alkanes. Los dos alkanes son n-pentane (C_5H_{12}) y n-hexane (C_6H_{14}) . Adicionalmente al carbón e hidrogeno, los alcoholes contienen el grupo funcional –OH; El alcohol metilico (CH_3OH) y el alcohol etílico (C_2H_5OH) serán los dos alcoholes empleados en este experimento, examinando la estructura molecular de ambos para predecir el valor de su fuerza intermolecular.

OBJETIVOS:

- Estudiar los cambios de temperatura causados por la evaporación de diversos líquidos.
- Relacionarlos cambios de temperatura con las fuerzas intermoleculares de atracción.

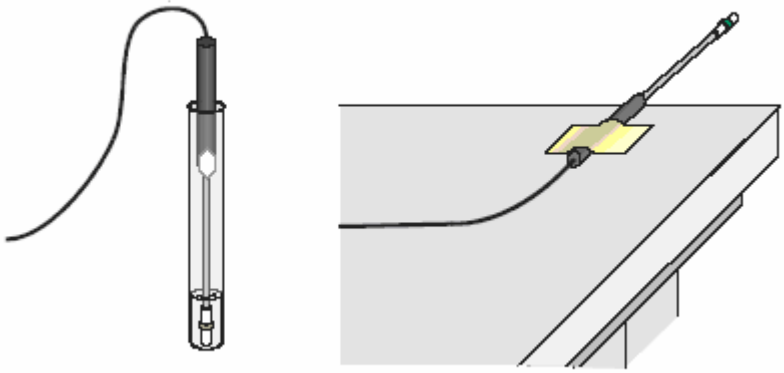


Figura 1

MATERIALES:

Computadora
Programa LabWin
2 sensores de temperatura
6 piezas de filtro de papel (2.5 x 2.5 cm)
2 ligas pequeñas
Masking tape
Alcohol metilico
Alcohol etílico
l-propanol
l-butanol
n-pentane
n-hexane

EJERCICIO:

Antes de iniciar el experimento, realice estas actividades previas:

Dibujar la estructura de la formula de la molécula de cada compuesto y determinar el peso molecular de cada uno.

RECOMENDACIONES:

Los compuestos utilizados son tóxicos e inflamables, evite su inhalación, así como contacto con la piel, ojos y ropa.

De preferencia, se recomienda el uso de lentes protectores (goggles) en todo momento.

PROCEDIMIENTO:

- Conecte los dos sensores de temperatura a la interfase de la computadora y seleccione la opción de 'Medir' del menú principal del programa.
- Envuelva el sensor 1 y 2 en las piezas de papel filtro, sujetándolas con las ligas, como se muestra en la figura 1.
- Coloque el sensor 1 en el recipiente del alcohol etílico y el sensor 2 en el de l-propanol. Asegúrese de sumergir totalmente las piezas de papel filtro.
- Tenga listas dos piezas de masking tape de aproximadamente 10 cm cada una.
- Después de mantener por al menos 30 segundos los sensores sumergidos en las sustancias, de inicio al muestreo por el programa y después de 15 segundos, retire simultáneamente ambos sensores del liquido y sujételos con las tiras de masking tape sobre el borde una mesa, sin que las puntas toquen la superficie (ver figura 1), manteniendo una separación aproximada de 5 cm. Del borde.
- Cuando la gráfica muestre que ambos sensores alcanzaron la temperatura mínima y esta comience a incrementarse, detenga el muestreo y regrese al menú principal para ver dichas gráficas.
- Utilizando la opción de "Gráficas", "Información", obtenga los valores mínimo y máximo de cada gráfica, calculando la diferencia entre ambos valores (Dt), que nos indicará la diferencia de temperatura durante el proceso de evaporación.
- Retire el papel de ambos sensores
- Basado en el resultado del primer experimento, estime el valor de Dt para el resto de las sustancias. Para probar estas estimaciones, repita los pasos 2 a 8 de este procedimiento.

PROCESANDO LOS DATOS

- Dos de los líquidos, n-pentane y l-butanol, tiene pesos moleculares similares, pero valores Dt significativamente diferentes. Explique esta diferencia, basado en sus fuerzas intermoleculares.
- Describa cual alcohol tiene la mayor fuerza intermolecular de atracción y cual la menor. Utilice los resultados de este experimento.

- Describa cual alkane tiene la mayor fuerza intermolecular de atracción y cual la menor. Utilice los resultados de este experimento.

- Dibuje la gráfica de las cuatro Dt contra sus respectivos pesos moleculares.

PRE-LABORATORIO

Substancia	Formula	Formula Estructural	Peso Molecular
ethanol	C_2H_5OH		
1-propanol	C_3H_7OH		
1-butanol	C_4H_9OH		
n-pentane	C_5H_{12}		
methanol	CH_3OH		
n-hexane	C_6H_{14}		

TABLA DE DATOS

Substancia	t_1 (°C)	t_2 (°C)	Δt (t_1–t_2) (°C)
ethanol			
1-propanol			
1-butanol			
n-pentane			
methanol			
n-hexane			

Predicción Δt (°C)	Explicación

Practica 4: Determinando el radio molecular en una reacción química

El balance de una ecuación de una reacción quima nos da el radio molecular de los reactantes y el producto como coeficientes. Cuando alguna de las formulas químicas es desconocida, un experimento nos puede conducir a ayudarnos a determinar el radio molecular.

Este experimento usa dos substancias comunes como reactantes: un ión hipoclorito (OCl) y un tiosulfato ($S_2O_3^{2}$). En la reacción, el hipoclorito oxida el tiosulfato de acuerdo a la siguiente ecuación desbalanceada e incompleta que se muestra a continuación:

A OCl + B $S_2O_3^{2}$ --> producto

Los coeficientes A y B para los reactantes, es posible identificarlos sin conocer el producto de la reacción. El proceso para determinarlo se denomina 'variación continua'. Se pueden preparar una serie de mezclas de los dos reactantes y cada mezcla. Cada mezcla tendrá la misma cantidad de volumen total y el mismo número de moles de reactantes. La reacción es exotérmica, generando la máxima energía, cuando la reacción se consume completamente al hipoclorito y al tiosulfato. Tu podrás establecer los coeficientes usando estas mezclas y por consecuencia el radio molecular para la reacción.

OBJETIVOS:

Medir el cambio de entalpía de una serie de reacciones.
Establecer la estoiquiometría de una reacción de oxido-reducción en la cual los reactantes son conocidos, pero su producto no.

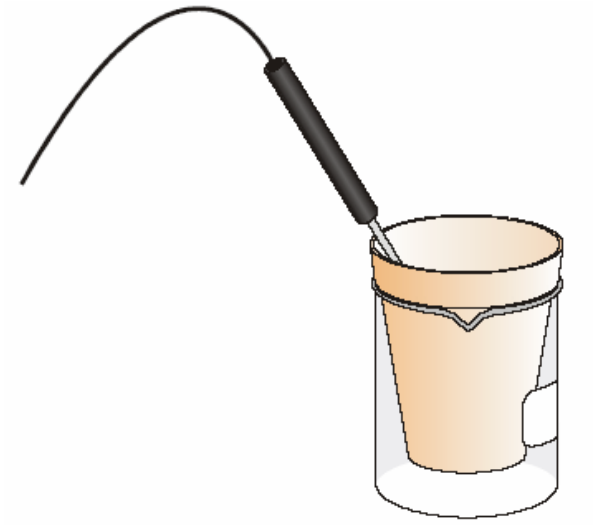


Figura 1

MATERIALES

Programa LabWin	2 vasos de 50mL
Sensor de temperatura	Solución de 0.50 M de NaOCl
2 vasos de 1 mL.	Solución de 0.50 M de $Na_2S_2O_3$ en 0.2 M NaOH
2 vasos de 25 L	Vasos de unicel

PROCEDIMIENTO:

1.- Selecciona un área bien ventilada para el experimento. Realízalo con el apoyo de un profesor de la materia y sigue sus indicaciones de seguridad. Utiliza lentes protectores durante todo el experimento.

2.- Conecta el sensor de temperatura a la interfase de la computadora.

3.- Agregue 25.0 mL de la solución NaCl en un vaso de unicel. Para evitar que se voltee y derrame, coloque el vaso dentro de uno de loa vasos graduados como se muestra en la figura 1.

4.- Introduzca la punta del sensor de temperatura dentro del vaso con la solución NaCl.

5.- En otro vaso, agregue 25 mL de la solución $Na_2S_2O_3$. No los mezcle hasta el paso en que esto se indique.

6.- Programe el sistema para realizar un muestro de 48 mediciones cada 5 segundos. Inicie el muestreo y después de algunos segundos, agregue la segunda solución, en el vaso que contiene el sensor.

7.- Terminado el muestreo, salve la gráfica resultante.

8.- Utilizando la opción de 'Gráficas', 'Información', obtén las temperaturas máxima y mínima de la reacción. Usa la tabla que se anexa para anotar los resultados.

Con estos datos crea una gráfica en Excel, donde el eje X corresponda a volumen en mL, mientras que el eje Y, corresponda a la diferencia entre la temperatura mínima y máxima.

9.- Sigue las indicaciones de tu instructor para desechar la sustancia resultante de la primera reacción y vuelve a repetir en dos ocasiones más el experimento con volúmenes de 35 mL y 50 mL de cada sustancia en cada nuevo experimento.

TABLA DE DATOS

Volumen OCl^- (mL)	Volumen $S_2O_3^{2-}$ (mL)	Cambio de Temperatura (°C)

ANALISIS DE DATOS

1.- Determina el valor completo del radio molecular de los dos reactantes. Usa la información de la gráfica en Excel.

2.- ¿El radio molecular de los reactantes?. Determina si esto es importante o necesario para el éxito del experimento.

3.- ¿Cuál de las soluciones fue el reactante limitante en cada prueba?.

4.- Encuentra el balance de la ecuación química de la reacción de ambas substancias. ¿Los radios moleculares que determinaste en tu experimento, concuerdan con los coeficientes de la ecuación de los dos reactantes?. Explica esto, sobre todo si no concuerdan.

Práctica 5: Reflexión y Absorción de la luz

Habrás notado que en un día caluroso, la ropa de color claro es más fresca que la de color oscuro. El color y la textura, influyen en que tanta energía radiada por el sol, es absorbida o reflejada.

Cada color refleja una determinada cantidad de luz, mientras que el resto, es absorbida como energía calorífica. La cantidad de luz reflejada, es llamada valor de reflexión de luz del color, siendo este valor muy bajo para colores oscuros y alto para los más claros.

En cuanto a la energía absorbida, los claros retienen poca, mientras que los oscuros mucha más.

Todo lo anterior, influye en las decisiones de la gente, como en la elección de colores claros para autos y casas en climas tibios y soleados.

En este experimento, investigaremos la relación entre el porcentaje de reflexión de varios colores y el cambio de temperatura producido por la absorción de energía. Para ello, mediremos la cantidad de la energía reflejada por papel de varios colores usando el sensor de luz y al mismo tiempo mediremos el cambio de temperatura que produce.

OBJETIVO:

- Usar el sensor de luz para determinar el factor de reflexión de los colores.
- Usar el sensor de temperatura para medir la absorción de energía proveniente de la luz.

MATERIALES:

Programa LabWIN
Sensor de Luz y Temperatura
4 cm de popotes para beber
Lámpara con foco de 150 W
Papel aluminio
Pinza de soporte de laboratorio
Papel blanco
Papel negro
2 piezas de papel de colores
soporte de laboratorio
Masking Tape y regla de 30 cm

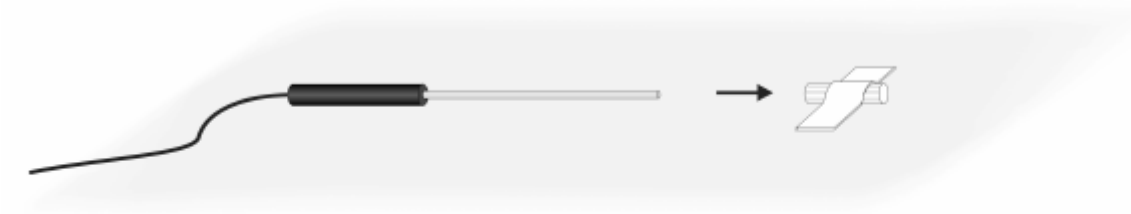


Figura 1

PROCEDIMIENTO

1. Prepare los materiales como a continuación se describe:

   - Pegue con masking tape el popote para beber, en la superficie de la mesa, como se muestra en la figura 1.
   - Inserte el sensor de temperatura en el popote, tan hondo como sea posible, pero que permanezca en su interior.
   - Coloque la hoja de papel blanco encima del sensor de temperatura
   - Usando el soporte de laboratorio con su pinza, sujete el sensor de luz a una distancia de 5 cm. de la hoja de papel, como se muestra en la figura 2
   - Coloque la lámpara a 10 cm. sobre la hoja de papel.
   - Mantenga las luces del laboratorio apagadas.

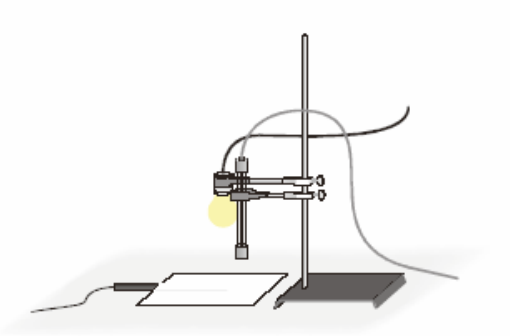


Figura 2

2. Programe en el sistema un muestreo de 120 mediciones cada 5 segundos; inicie el experimento hasta que concluya el período definido.

3. Vuelva a repetir el paso anterior, primero para la hoja de aluminio, continué con el papel negro y finalmente con las hojas de color.

DATOS

Color	Blanco	Negro	Aluminio	_____	_____
Temperatura Inicial (°C)					
Temperatura Final (°C)					
Cambio en Temperatura (°C)					
Valor de Reflexión (lux)					
Porcentaje de Reflexión	%	%	100 %	%	%

PROCESAMIENTO DE DATOS

1. Resta la temperatura final a la inicial, para determinar el cambio por cada color.

$$\% \text{ de Reflexión } = \frac{\text{Valor de reflexión del papel}}{\text{Valor de reflexión del aluminio}} \times 100$$

2. Anota cual es el color con más incremento de temperatura

3. Anota cual es el color con menor incremento de temperatura

4. Calcula el porcentaje de reflexión de cada color. Usa la siguiente formula

5. Anota cual es el color con mayor reflexión de luz

6. Anota cual es el color con menor reflexión de luz

7. Describe que relación hay entre el factor de reflexión y el cambio de temperatura.

8. Con base a estos resultados, describe cuales crees que serán las superficies del planeta con mayor reflexión de la luz solar.

9. ¿Crees que nuestro planeta tiene un alto grado de reflexión?. Explica ampliamente tu respuesta.

Glosario

ACCESS: Aplicación desarrollada por Microsoft®, como motor de base de datos relacionales.

ACTIVEX: Término colectivo aplicado a la familia de tecnologías de software de socios de negocio de Microsoft®, que permiten comunicar y compartir datos entre aplicaciones Windows®.

API: Acrónimo en inglés de “Interface de Programación de Aplicaciones”. Método para utilizar los recursos construidos bajo el sistema operativo Microsoft-Windows®, como parte de una aplicación de Visual Basic entre otras. El API, especifica la sintaxis necesaria en VISUAL BASIC® para hacer llamados a bibliotecas dinámicas (DLL’s) en Windows®.

ARREGLO: Una estructura lineal de datos que consiste de una secuencia numerada de valores numéricos, caracter o referencia a objetos.

ASCII: Acrónimo en inglés de “Código Estándar Americano para Intercambio de Información”. Es un modelo que representa una secuencia ordenada de números asociada con las letras del alfabeto, números y otros caracteres.

B-SPLINE: Método numérico para crear gráficas con curvatura suave, en base a un conjunto de puntos del plano cartesiano.

DBASE®: Software de manejo de bases de datos desarrollado por la empresa Ashton-Tate en los años 80’s, para ser utilizado en computadoras con sistema operativo MS-DOS®.

DLL: Abreviatura en inglés de “Bibliotecas Dinámicamente Ligadas”. Son archivos que se cargan en memoria cuando son requeridos por una aplicación, siendo éstos, recursos compartidos en Windows®. Cuando una DLL se encuentra en memoria, ésta puede ser utilizada por más de una aplicación.

ENSAMBLADOR: Lenguaje de programación de bajo nivel, utilizado para escribir programas a nivel de las instrucciones básicas que reconoce un procesador. Estos programas son “ensamblados”, para ser traducidos a lo que se conoce como “código de máquina” que son una serie de valores en binario que representan instrucciones para el procesador de una computadora.

EXCEL®: Aplicación comercial desarrollada por Microsoft®, como una herramienta conocida como “hoja de cálculo”, que se utiliza bajo ambiente Windows®.

FOTO-RESISTENCIA: Componente electrónico de resistencia variable en función de la intensidad luminosa que recibe.

INTEL®: Empresa dedicada a la fabricación de componentes electrónicos de estado sólido, quién en 1979 desarrolla el procesador 8086®, el cual es seleccionado por IBM® para la primera computadora personal conocida como PC.

JOYSTICK: Dispositivo que se conecta a una computadora por medio de un puerto de juegos. Este dispositivo, también es conocido como “palanca de juegos”, por estar orientado principalmente para usarse con este tipo de programas de computadora.

LOTUS123®: Aplicación comercial, que es una de las primeras herramientas conocidas como “hoja de cálculo”. En la época de las computadoras con sistema operativo MS-DOS®; fue una de las aplicaciones más populares de su tipo.

LINE: Objeto ActiveX incluido en la aplicación Visual Basic®, para trazar líneas dentro de una ventana de un programa desarrollado con este lenguaje. Estos objetos, se pueden desplegar en diversas posiciones, así como diversos colores y anchos de línea.

LUMEN: Unidad de medición de intensidad luminosa, utilizada en las teorías físicas del comportamiento de este fenómeno.

TERC®: Abreviatura en inglés de “Centro de Investigación en Tecnología Educativa”, ubicado en la ciudad de Boston, Massachusetts, USA. Este instituto desarrolla diversos proyectos educativos apoyados en la tecnología.

MS-DOS®: Sistema operativo desarrollado por Microsoft® utilizado en las computadoras IBM-PC®.

MSJET®: Nombre del motor de base de datos construido dentro de VISUAL BASIC®. Es un recurso compartido por más de 20 aplicaciones, entre ellas Word® ACCESS®, Excel®, PowerPoint®, etcétera. MSJET® es un motor totalmente relacional y soporta su propia sintaxis SQL, tablas, querys y otras características de bases de datos.

ORT®: Acrónimo de “Organización para la Rehabilitación del Trabajo”. Organismo fundado en Rusia en el año 1880, cuyo objetivo, era capacitar a los inmigrantes del campo para trabajar con la tecnología de la nueva era industrial. Actualmente su sede está en Israel y tiene presencia en todos los continentes, difundiendo la educación tecnológica.

PASCAL: Lenguaje de programación estructurada de 3ra generación, desarrollado originalmente por Niklaus Wirth.

PENTIUM®: Familia de procesadores o CPU’s, desarrollados por la empresa INTEL. Actualmente es la familia con mayor presencia en los equipos conocidos como PC.

PIXEL: Unidad mínima de graficación en una pantalla de ambiente gráfico desplegada en un monitor de computadora. Cada PIXEL, se identifica por una coordenada de dos dimensiones que representa el desplazamiento horizontal y vertical con respecto a la esquina superior izquierda de una pantalla.

PUERTO DE JUEGOS: componente de hardware en una computadora, destinada para la conexión de dispositivos usados principalmente para controlar programas de juegos.

QUERY: representa una instrucción para extraer o manipular datos almacenados en una base de datos. Un Query puede representarse en varias formas dentro de una aplicación Visual Basic®, pero la más común es como una sentencia escrita en SQL.

RAM: Abreviatura en inglés de “Memoria de Acceso Aleatorio”. Representa la cantidad de memoria no permanente que tiene instalada por HARDWARE una computadora.

RECORDSET: Objeto utilizado en la programación con bases de datos, cuya función es ser un apuntador a un conjunto de registros de datos, que están a disposición del programa que los obtiene por medio de una consulta.

SQL: Acrónimo en inglés de “Lenguaje de Consulta Estructurada”. Representa la sintaxis de sentencias escritas en un lenguaje semejante al inglés, que le es enviado a un motor de base de datos para la ejecución de una operación de lectura o modificación a los datos en una o varias tablas, dentro de una base de datos.

TERMO-RESISTENCIA: Componente electrónico de resistencia variable en función de la temperatura que registra a su alrededor.

TIMER: Componente ActiveX incluido en la aplicación Visual Basic®, que permite programar la ejecución de rutinas que se ejecutan cada determinado periodo de milisegundos.

VISUAL BASIC®: Nombre de la aplicación desarrollada por Microsoft® para el desarrollo de programas de propósito general en ambiente Windows®.

VISUAL STUDIO®: Nombre de la suite de productos de desarrollo de programas, creada por Microsoft®. Este producto, incluye lenguajes de programación como VISUAL BASIC®, VISUAL C++®, VISUAL FOXPRO®, entre otros.

WINDOWS®: Producto de ambiente gráfico desarrollado por Microsoft® para ser ejecutado en computadoras conocidas como PC.

BIBLIOGRAFÍA:

Norton, Peter. “Guide to Visual Basic 6”.
Sams Publishing, USA, 1998.

Roman, Steven. “Win32 API Programming with Visual Basic”.
O’Reilly, USA, 2000.

Courant R., John F..”Introducción al Cálculo y al Análisis Matemático Vol. I”.
LIMUSA, México, 1979.

Wexler, Charles.”Geometría Analítica. Un enfoque vectorial”.
Montaner y Simon, S.A., España, 1968.

Gutiérrez, Carlos.”Física 2”.
LAROUSSE, México, 1998.

Sears, Francis W.; Zemansky, Mark W..”Física”.
Aguilar S.A., España, 1966.

REFERENCIAS DE INTERNET

Vernier Internacional
www.vernier-intl.com
Empresa dedicada al desarrollo de software educativo.
Consultada en Enero 2005.

epanorama
www.epanorama.net
Página con documentación relacionada con los puertos de juegos de una PC
Consultada en Enero 2005.

Southwest
www.southwest.com
Página con documentación del uso de API’s para manejo de los puertos de una PC desde lenguajes como Visual Basic®
Consultada en Enero 2005.