UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Facultad de Ingeniería

Desarrollo e implementación de un interfaz usuario-maquina TOUCH-SCREEN para un sistema de monitoreo y control de granjas acuicolas.

Que para obtener el título de:

Ingeniero en Automatización.

Presenta:

Christian Reyes Gonzalez

M.-en C. Genaro Martín Soto Zarazúa

Querétaro, Qro., 15 de Octubre 2008

No. Adq [redacted]
No. Título
Clas TS
006.6784
R457d

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Dirección

ACUERDO 359/08

C. U. 17 de septiembre de 2008

C. CHRISTIAN REYES GONZÁLEZ
Pasante de Ingeniería en Automatización
(Sistemas mecatrónicos)
Presente.

Con relación a su oficio enviado al H. Consejo Académico de la Facultad en el que solicita titularse bajo la opción de tesis individual, me permito informarle que en la sesión ordinaria del 17 de septiembre del año en curso, este cuerpo colegiado acordó aceptar la opción de titulación por lo que deberá trabajar en el tema "**Desarrollo e implementación de una interfaz usuario-maquina TOUCH-SCREEN para un sistema de monitoreo y control de granjas acuícolas.**", bajo la dirección del M en C. Genaro Martín Soto Zarazúa

El Contenido aceptado por el H. Consejo Académico es el siguiente:

Resumen
Summary
Dedicatorias
Agradecimientos
Índice
Índice de cuadros
Índice de figuras
I. CAPITULO 1
Descripción del problema
Antecedentes y justificación
Objetivo general
Objetivos particulares
Hipótesis general
Hipótesis especificas
Axiomática
Metodología

Centro Universitario
de las Campanas
Santiago de Querétaro
México C.P. 76010
01 (442) 192 12 00
Ext. Fax. 6005

BIBLIOTECA CENTRAL, U.A.Q.

II. CAPITULO 2

Antecedentes

Evolución de la Interacción Hombre Máquina

Definición de Interacción Humano Máquina

Estilos de Interacción

Agentes de la interacción

Diseños de Interfaces

Análisis cualitativo de la interface

Análisis cuantitativo de la Interface

Análisis dirigido por la tecnología

III. METODOLOGIA

Panorama general

Primera versión

Segunda versión

Tercera versión

Control por tiempos

Control Fuzzy

Etapas complementarias

Etapa computacional

Etapa electrónica

Etapa de potencia

Control de etapa de potencia

Pruebas

Pruebas de hardware

Funcionalidad

Destrucción.

Máxima carga.

Compatibilidad.

Pruebas de software

Repetibilidad.

Respuesta.

entro Universitario
de las Campanas
iago de Querétaro
léxico C.P. 76010
1 (442) 192 12 00
Ext. Fax. 6005

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Dirección

Compatibilidad.

Entradas no validas.

Confiabilidad.

IV. RESULTADOS

Resultados control por tiempos

Respuesta control Fuzzy

Conclusiones de pruebas en Hardware y Software

Conclusiones finales

LITERATURA CONSULTADA

APENDICE

También hago de su conocimiento las disposiciones de nuestra Facultad, en el sentido que antes del Examen profesional deberá cumplir con los requisitos de nuestra legislación y deberá imprimir el presente oficio en todos los ejemplares de su tesis.

Atentamente
"EL INGENIO PARA CREAR NO PARA DESTRUIR"

DR. GILBERTO HERRERA RUIZ
Director
c.c.p. Archivo
*GHR/DHM.

DIRECCIÓN

Centro Universitario
o de las Campanas
tiago de Querétaro
México C.P. 76010
01 (442) 192 12 00
Ext. Fax. 6005

RESUMEN

El presente trabajo expone el desarrollo e implementación de la interfaz usuario maquina, haciendo uso de una pantalla táctil (Touch Screen) con la cual se pretende permitirle al usuario una manera más accesible para controlar el sistema acuícola. Este desarrollo interactúa con múltiples disciplinas, donde incluso es necesario tener un contacto frecuente con los operadores del sistema; debido a que las maquinas (computadoras) no demuestran por su forma la manera en que pueden ser utilizadas es necesario hacer adecuaciones e indicaciones de cómo y de qué manera hacer uso de ellas; es importante hacer notar que aunado a las distintas disciplinas inmiscuidas es también de vital importancia el uso de periféricos del sistema como las etapas de potencia, los actuadores y demás artefactos para que este sistema pueda desempeñar su función. El sistema se pretende entonces controlado únicamente por medio de una pantalla táctil y un tablero de potencia que a pesar de que el tablero incluye funciones manuales el sistema es por si totalmente manipulable a través de la ya mencionada pantalla. Se proponen dos soluciones dependiendo del uso que se le dé, una con un estricto control por tiempos y una mas donde se pueden programar condiciones que toma la computadora y con las cuales toma decisiones. El primero de ellos se limita únicamente a encender actuadores por un tiempo programado; el segundo utiliza una base de reglas a programar por el experto acuicultor con las cuales se raciona la cantidad de alimento.

(**Palabras clave**: acuicultura, control por tiempos, control fuzzy)

SUMMARY

This paper describes the development and implementation of the user interface machine, using a touch screen (Touch Screen) with which it is intended to allow the user to a more accessible way to control aquaculture system. This development interacts with multiple disciplines, even where it is necessary to have frequent contact with the operators of the system, because the machines (computers) do not show it by how it can be used is necessary to make adjustments and indications of how and how to use them, it is important to note that in addition to the various disciplines interfered is also of vital importance to the use of system peripherals such as the power stage, actuators and other devices for this system to play its role. The system is designed to then solely controlled by a touch screen and a power board that although the board includes manual functions if the system is totally manipulated by the aforementioned screen. Two solutions are proposed depending on the use given, with a strict control over time and a position where they can be programmed to take the computer and with which it makes decisions. The first of these is limited only to turn on actuators for a scheduled time, the second uses a base of rules to be programmed by the expert aquaculturist with which the amount of food rations.

(**Key words**: aquaculturist, scheduled time control, fuzzy control)

A todas las personas que me apoyaron y creyeron en mi
Padres, maestros, asesor y sinodales

AGRADECIMIENTOS

En primer lugar mi asesor Genaro Martin Soto Zarazua por la orientación y asesoría proporcionada. A mis padres por el apoyo económico y moral. Y a todos los maestros que me permitieron culminar mis estudios.

INDICE

INDICE DE CUADROS

INDICE DE FIGURAS

I. CAPITULO 1

Descripción del problema

Actualmente se encuentran sistemas para monitoreo y control de sistemas acuícolas comerciales y otros que han sido desarrollados por investigadores y estudiantes de la Universidad Autónoma de Querétaro (UAQ) y normalmente los métodos para introducir los parámetros del sistema requieren de un manual para su operación adecuada y el hecho de involucrar un manual implica conocimiento sobre control, tiempo para buscar, leer y entender el significado de cada uno de ellos y como es que estos parámetros modifican la dinámica del sistema lo que implica una limitante para el usuario con poco conocimiento ya que la entrada de parámetros puede complicarse. Los sistemas hasta ahora desarrollados, se consideran autónomos y en el caso de los sistemas de alimentación, se monitorea el sistema y de acuerdo a la información leída por los sensores (Temperatura, Oxigeno Disuelto y pH) el controlador varía la cantidad de alimento basándose en algoritmos de lógica difusa, los carbohidratos a suministrar para la neutralización del amonio vía el balance C/N y sí alguna perturbación afecta el sistema y es necesario cambiar algún parámetro el sistema no se puede pausar para analizar la documentación del sistema y pasado el tiempo aplicar una acción correctiva, estos sistemas aunque sofisticados y novedosos por las metodologías utilizadas para solucionar los problemas de acuicultura requieren de una interfaz accesible y amigable para los usuarios finales y de la misma manera hacer uso de tecnología de punta para mantenerse actualizado en los desarrollos tecnológicos hechos por la UAQ.

Antecedentes y justificación

La historia de las interfaces de usuario puede ser dividida en las siguientes fases de acuerdo al tipo dominante de interfaz del usuario:

- Interfaz batch (1945-1968); donde el usuario programaba una tarea a la maquina y hasta que esta terminaba, podía entonces el usuario programar o desempeñar una nueva tarea.
- Interfaz de usuario de línea de comandos (1969-1980); El usuario entonces a través de una lista de instrucciones podía ordenar a la maquina el que se desempeñara tal o cual actividad.
- Interfaz grafica de usuario (1981-hasta la fecha); Por medio de una representación grafica el usuario interactúa con un sistema. Esta representación grafica es mas específicamente la posibilidad de representar en una pantalla no solo texto si no demás elementos que faciliten al usuario la interacción como iconos y un puntero.
- Interfaces tangibles (Ubicomp)
- Interfaz de usuario táctil (TUI)

Hoy en día los humanos interactúan mas con sistemas basados en tecnología computacional que con martillos y desarmadores. No siendo como las herramientas, la forma y controles de una computadora no comunican su propósito. La tarea de una interfaz humano maquina (HMI) es el hacer la función de una tecnología evidente. Así como un martillo bien diseñado se adapta a la mano del usuario y hace una tarea física fácil, una interfaz humano maquina bien diseñada debe cumplir con el concepto mental del usuario de la tarea que el o ella desean desempeñar.

En casi cualquier solución tecnológica, la efectividad del HMI puede predecir la acepción de la solución completa por los usuarios. Frecuentemente, tan lejos como los clientes se preocupen, el HMI es el producto – la experiencia del usuario con la interfaz es todavía mas importante que la arquitectura de los

mecanismos internos. Por ello en esta investigación se desarrolla una interfaz humano-maquina TOUCH-SCREEN de manera que el usuario no necesite un estudio exhaustivo del sistema para realizar correcciones justo a tiempo así evitando afectar la producción del sistema al monitorear las variables de interés ulteriormente mencionadas, tomando en cuenta la accesibilidad de su interfaz, intuitivamente permitirá la fácil configuración del sistema para un optimo y eficiente funcionamiento en cuanto al proceso de producción respecta. A pesar de que existen HMIs comerciales listas para conectarse solo ofrecen 2MB de memoria interna lo cual resulta insuficiente para guardar reportes de la respuesta del sistema además de tener pocas expectativas de actualización en cuanto a periféricos externos; así al tener un sistema base escalable se hace más posible el poder integrar futuras mejoras a sistemas en funcionamiento.

Objetivo general:

Desarrollar e implementar una interfaz grafica que ayudada por una pantalla de tacto, proporcione al usuario una manera más simple de interactuar con el sistema de control, y así evitar el empleo de un apuntador y un teclado.

Objetivo particulares

- Reducir la interfaz de usuario a una pantalla de tacto.

- Optimizar del código (mejora en velocidad), orientado a la respuesta del sistema de control.

- Aumentar la amigabilidad del sistema y su facilidad de uso.

Hipótesis

Con la implementación de la interfaz táctil de usuario el sistema se vuelve accesible para cualquier persona, al limitar completamente las opciones de entrada a aquellas que tengan verdadera significancía para el sistema.

Hipótesis particulares

- Al ser desempeñada la función del apuntador y teclado por la pantalla táctil, únicamente se utilizar la pantalla como interfaz.

- Al crear de manera dinámica los elementos del programa, este será ejecutado a mayor velocidad.

- Al hacer uso de un lenguaje de programación grafico orientado a objetos, el usuario tendrá mayor facilidad para hacer uso de el.

Metodología

- Estudio del sistema. Análisis de variables de entrada y salida necesarias para el control del sistema a través de una computadora personal, tipos (analógicas/digitales), y número de entradas/salidas.

- Identificar los puertos del computador más aptos para controlar el sistema, así como el diseño e implementación de circuitos de prueba o auxiliares al sistema.

- Descripción de algoritmo del programa, listar el procedimiento deseado del sistema y a partir de ello generar un algoritmo siguiendo la ruta mas adecuada y que implique menos consumo de memoria; el mencionado algoritmo será la base del programa.
- El Programa se diseñara con base en la forma más accesible para el usuario. Siempre teniendo en cuenta los requerimientos mínimos para el funcionamiento correcto de la rutina de control, evitando así la inclusión de código innecesario que puede poner en riego la velocidad o buen funcionamiento del sistema de control.

II. CAPITULO 2

Antecedentes

Los temas tratados en este apartado son en su mayoría bases sobre lo cual se fundamenta el desarrollo del proyecto; entre los más representativos se incluyen las HMIs, programación orientada a objetos, control difuso, acuicultura y diseño electrónico. La interfaz en desarrollo se enfoca específicamente al sistema de control de alimentadores para peces desarrollado por Soto-Zarazúa (2007) y el sistema para neutralización de amonio desarrollado por Olvera-Olvera en 2007 (Artículos en revisión). Al unir estos sistemas y con su nueva interfaz de usuario final se dará un valor agregado al sistema permitiendo al usuario contar con una solución efectiva, intuitiva y actualizada en cuanto a las herramientas tecnológicas de hoy en día; su principal ventaja será reflejada en la facilitación de uso para los encargados del manejo en granjas acuícolas y por consecuente en una mejor respuesta del sistema reflejada en la producción.

Evolución de la Interacción Humano Máquina

Mcluhan (1964) clasifica las eras de la humanidad, de tal manera que van a determinar lo que actualmente se conoce como la interacción hombre maquina o computadora. Tales eras son:

1. La era preliteraria.
2. La era de Gütemberg.
3. La era electrónica.

En la era preliteraria o tribal, este autor señala, que antes que la escritura se extendiera, la humanidad vivía en un espacio acústico, el espacio de la palabra hablada. Este espacio no tiene frontera, ni dirección, ni horizonte y esta cargado de emoción. Mientras que en la era de Gütemberg, con la invención de los tipos

móviles, se forzó al ser humano a tener una comprensión en forma lineal, uniforme, concatenada y continua. Así, debido a la escritura, se produjo una estructura que transformo al espacio en algo limitado, lineal, ordenado, estructurado y racional. La página escrita con sus bordes, márgenes y caracteres definidos en renglón tras renglón trajo una nueva forma de pensar el espacio.

Mcluhan sostenía que la movilidad del libro "fue como una bomba de hidrogeno" cuya consecuencia fue el surgimiento de un "entorno enteramente nuevo".

Apareció un nuevo ambiente: el espacio ilustrado, el espacio urbano. El pensamiento lineal, produjo además, "en la economía: la línea de montaje y la sociedad industrial"; "en física las visiones newtonianas y cartesianas del universo como un mecanismo en el que es posible localizar un suceso en el tiempo y el espacio". "En el arte la perspectiva"; "en la literatura la narración cronológica".

En la era electrónica Mcluhan pensó que la tecnología electrónica no depende de las palabras habladas y puesto que la computadora es la extensión del sistema nervioso central, dedujo que cabe la posibilidad de extender la conciencia. Edificando un camino hacia la comprensión y la unidad universal (hoy la Internet). El pensamiento de Mcluhan sobre los medios comienza con dos premisas, por un lado sostenía que "...somos lo que vemos"; y por el otro afirmaba que "...formamos nuestras herramientas y luego son estas las que nos forman". Mcluhan veía a los medios como agentes de posibilidades (herramientas) antes que de conciencia.

La mayoría de nosotros pensamos en los medios como fuentes que nos brindan noticias o información. Pero tenía su propia e ingeniosa definición de los medios. Para él cualquiera sea la tecnología, todo medio es una extensión de nuestro cuerpo, mente o ser, ejemplo: la ropa es una extensión de la piel, la casa es una extensión de los mecanismos de control de la temperatura corporal, el

estribo, la bicicleta y el automóvil son una extensión del pie humano, mientras que la computadora es una extensión de nuestro sistema nervioso central (cerebro). Este autor redefine los medios y en consecuencia se hace necesario redefinir el mensaje. Mcluhan (1964), parte de un nuevo concepto de "medio", estableciendo que "medio" es: toda prolongación de nuestro propio ser debido a cada nueva tecnología. Porque todos los medios de comunicación, desde el alfabeto fonético a la computadora, son extensiones del hombre que causan profundamente cambios duraderos en él y transforma su ambiente.

Semejante extensión es una intensificación, una ampliación de un órgano, sentido o función, y siempre que esto tiene lugar, parece que el sistema nervioso central no toma conocimiento consiente de lo que esta pasándole a él, ya que este autor observa, que este efecto es similar a la falta de conciencia de los peces con relación al agua en la que se desplazan. Por otra parte Mcluhan (1967) afirma que el medio es el mensaje, siendo así, nos obliga a evaluar lo que entendemos tanto por medio como por mensaje. Hemos visto como redefinió el significado del sentido común. Lo mismo realiza con el significado de mensaje, Mcluhan explica que si solo entendemos por mensaje la información o contenido, dejamos de lado una de las características mas importantes de los medios: su poder para modificar el curso y funcionamiento de las relaciones y actividades humanas, dado que el mensaje de un medio es todo cambio que ese medio provoca es la sociedades o culturas.

Determinado así que la computadora, es un medio que permitirá la extensión de nuestro sistema nervioso central. Mcluhan derivaba las consecuencias de la tecnología al orden social y natural del hombre. Con la posibilidad de extender nuestra conciencia, va más allá de los conceptos contemporáneos.

Definición de Interacción Humano Máquina

La palabra interacción, sencillamente, refiere a un sistema previo que puede organizarse de manera formal o informal y por ello, se menciona que la interacción, en dicho sistema, realiza procesos de intercambio en sentido amplio.

En el tema que nos ocupa, la Interacción, es un término que se refiere a una relación dada entre el ser humano o la persona y la máquina a través de una interfase. Nuestra definición esta configurada en la comprensión que lleva al ser humano a realizar una extensión de sus capacidades. Por la extensión de nuestras capacidades por medio de las máquinas, se entiende las ventajas que dan al ser humano para realizar otras tareas concomitantes, dejando las rutinas o de tipo autómata a las maquinas.

Además por la extensión se comprende la posibilidad de realizar tareas que comprendas a las máquinas como interfase para la comunicación directa o indirecta con otros seres humanos. En esta relación de hombres o personas y máquinas se comprende que las interacciones en si, se relacionan con los procesos internos automáticos del ser humano. Estos procesos internos son rutinas de procesamientos de la información, así las máquinas llevan en si algoritmos que procuran mejorar el desempeño de la persona y aumentar su inteligencia, como asimismo sus niveles de conciencia, dado que las personas utiliza las máquinas para su uso personal.

La Interacción Humano Computadora (del inglés HCI Human Computer Interaction) es uno de los temas principales de este apartado. No obstante ello; el termino máquina, para este trabajo conceptual es mucho mas amplio e incluye además de las computadoras u ordenadores a otros dispositivos (robots, automóviles, aparatos de investigación, diagnostico y tratamientos médicos, etc.). Otra definición plantea que la Interacción Humano Máquina es: "el intercambio de símbolos entre dos o mas partes, asignando a los participantes en el proceso comunicativo, los significados a esos símbolos", Booth (1989; Pág. 46) o la de

H74377
BIBLIOTECA CENTRAL, U.A.Q.

Johnson (1992), el cual considera la interacción hombre máquina: "... como el estudio de la interacción entre la gente, los ordenadores y las tareas. Esto principalmente relacionado con la comprensión de cómo la gente y los ordenadores pueden interactivamente realizar tareas, y como tales sistemas interactivos son diseñados."

De esta manera decimos que la HMI (del inglés Human Machine Interface, Interfaz Humano Maquina), tiene como finalidad estudiar:

1. El Hardware, el software ambos en función de la interacción.
2. Como lo establecen Lewis y Reiman (1993): el diseño está en relación al usuario y no a la maquina (User-centred design).
3. Los modelos mentales de los usuarios con relación al modelo del sistema.
4. Las tareas realizables por el sistema y su adaptación a las necesidades del usuario.
5. El impacto en las organizaciones

Por ello, para analizar la interacción con los sistemas, tanto estructural, como funcionalmente hace falta:

1°. Comprender los factores relacionados con los mecanismos psicológicos, ergonómicos, de las organizaciones y su modo de estructuración en relación a como se desarrollan tareas y se hace uso de las computadoras, de modo que con ello se infiere un método para:

2°. Desarrollar herramientas y técnicas que colaboren con los diseñadores para conseguir que los sistemas computacionales sean los correctos para las necesidades que se desean realizar, y de este modo:

3°. La interacción humano-computadora sea eficiente, efectiva, segura, tanto a nivel de la persona como del grupo implicado (Preece, 1994).

Por otra parte, debido a que una entre muchas otras de las funciones de la Interacción Humano Computadora tenga por fin ultimo la búsqueda de información,

Cobo (2006) comenta que "las brújulas indican que los estudios de Interacción Humano Máquina [Human Computer Interaction (HCI)], pronto serán remplazados por la Interacción Humanó Información [Human Information Interaction (HII)]". Las investigaciones en 'Human Information Interaction' sin duda se han inspirado en 'Human Computer Interaction'. Desde esta nueva perspectiva, la complejidad de la interacción con la información no solo se expresa a través de interfaces que permiten al usuario un alto control de la funcionalidad de un dispositivo, sino que lo que se busca priorizares el acceso a mapas de información. El que existan interfaces cada vez más transparentes permite que los usuarios puedan acceder a información multimedia a través de varios dispositivos e interfaces. El énfasis cambia de la interfaz a la información. La búsqueda de información, una vez más se acerca a la usabilidad. Cada día crece (y se adopta mas) la cultura de la usabilidad. Esta puede encontrarse en conceptos como: ergonomía, interfaz humano máquina [Human Machine Interface, HMI], interacción humano computadora [Human Computer Interaction, HCI o Computer Human Interaction, CHI], diseño centrado en el usuario [User Centered Design, UCD], factores humanos [Human Factors, HF], diseño de interfaz usuario [User Interface Design, UID], por nombrar algunas.

Siguiendo el tema, podemos observar que la interacción es multimodal (Lóres, 2006), por ello tenemos que:

1. La mayoría de los sistemas actuales interaccionan a través de un teclado, una pantalla y un ratón.
2. Cada uno de estos dispositivos se puede considerar canales de comunicación del sistema y corresponden con ciertos canales de comunicación humanas (tacto, vista,...).
3. Una interacción es multimodad cuando usa múltiples canales de comunicación simultáneamente.
4. Cada canal del usuario se puede considerar una modalidad diferente de interacción. Los sistemas actuales tienden a tener múltiples canales de comunicación de entrada/salida. Los seres humanos procesan la

información simultáneamente por varios canales. Por ejemplo podemos ajustar el movimiento de un ratón mediante la voz.

Debido a estas modalidades podemos hablar de interacción dimensional dado por el concepto de dimensionalismo estructural humano.

Estilos de Interacción

Preece (1994) dice que: "Estilos de interacción" es un término genérico que se utiliza para agrupar las diferentes maneras en que los usuarios se comunican o interaccionan con el ordenador o computadora. Los estilos de interacción más importantes son:

1. La interfaz/interfase por línea de órdenes.
2. Menús y formularios.
3. Manipulación directa.
4. Interacción asistida (agentes/asistentes)

A continuación se explica en cada uno de ellos:

1. Es una manera de dar instrucciones directamente al ordenador. Pueden tener la forma de teclas de función (F1 al F12), un carácter, abreviaciones cortas, palabras enteras o una combinación de las 2 primeras.

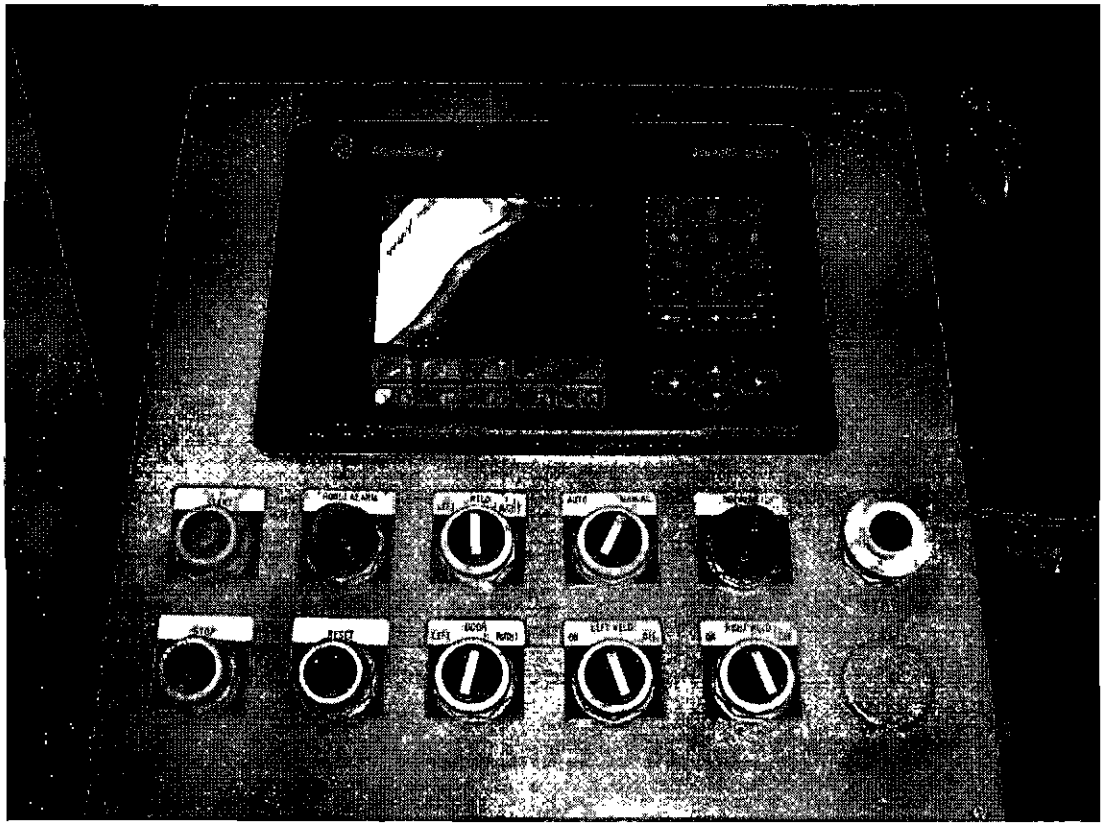


Figura 1. Ejemplo de interfaz con funciones directas.

2. Un menú es un conjunto de opciones visualizadas en la pantalla, que se pueden seleccionar y la selección de una de ellas o más supone la ejecución de una orden subyacente y normalmente un cambio en el estado de la interfaz. A diferencia de la interfaz por línea de órdenes, los usuarios tienen la ventaja de no tener que recordar ni palabras, ni sintaxis, siempre y cuando los textos que acompañan a los menús sean significativos, lo que no siempre se produce.

Uno de los problemas que tienen los menús es que ocupan mucho espacio en la interfase, dicho problema se trata de resolver con los menús desplegables o menús pop-up.

3. El término manipulación directa describe sistemas que tienen las siguientes características:

a) Representación continúa de los objetos y de las acciones de interés.

b) Cambio de una sintaxis de comandos compleja por la manipulación de objetos y acciones.

c) Acciones rápidas, incrementales y reversibles que provocan un efecto visible inmediatamente en el objeto seleccionado.

La interfaz WIMP: WIMP quiere decir Ventanas, Iconos, Menús y Apuntadores (Windows, Icons, Menus and Pointers).

4. Utiliza la metáfora del asistente personal o agente que colabora con el usuario en el mismo ambiente de trabajo. Así el usuario en vez de dirigir la interacción, trabaja en un entorno cooperativo. Por todo ello, el usuario y los agentes o asistentes se comunican, controlan eventos y realizan tareas.

Agentes de la interacción

Según Henry Lieberman es un programa que puede ser considerado por el usuario como un asistente o programa que le ayuda y no se le considere una herramienta desde el punto de vista de una interfaz de manipulación directa. El agente de la interfase lee la entrada que el usuario presenta en la interfase y puede hacer cambios en los objetos que el usuario ve en la pantalla, aunque no necesariamente una a una con las acciones del usuario.

El agente puede observar muchas interacciones del usuario antes de hacer una acción o con una sola interacción puede lanzar una serie de acciones y actuar en determinados periodos de tiempo que hemos fijado. Las características de los agentes son:

Autonomía. Trabajan en segundo plano y si no se les pide explícitamente observan al usuario y a las fuentes de información accesibles.

Inteligencia. Actúan por su propia iniciativa y pueden trabajar en entornos heterogéneos adaptándose a múltiples situaciones (no necesariamente utilizan la misma estrategia de resolución cada vez).

Uso personal. Se adaptan y aprenden del usuario y no insisten en una determinada solución si el usuario decide otra (la reglamentación por parte del usuario es una característica de los agentes inteligentes). Los agentes pueden tomar decisiones en situaciones que no son cómodas para el usuario, pero nunca han de forzar a tener un cierto comportamiento.

Asistentes. Son entidades computacionales que nos asisten en el uso de las aplicaciones existentes. Los asistentes nos exponen de una manera fácil que es lo que se ha de hacer y pueden entender palabras escritas, habladas o acciones graficas e interpretarlas. Interpretar quiere decir que el asistente puede hacer acciones complejas u ordenes cortas. Un requerimiento importante para los asistentes es que han de ser muy flexibles en la manera en que reciben las

instrucciones. Los asistentes son muchas veces más flexibles que los menús y las macros por que el usuario nada mas dice lo que quiere hacer.

Diseños de Interfaces

El diseño de la interfase puede ser realizado desde tres puntos de vista: el cualitativo, el cuantitativo y desde el enfoque tecnológico exclusivamente. Los diseños de interfaces como estudios centrados en la Interacción Humano Máquina, pueden ser clasificados desde lo cualitativo como desde lo cuantitativo. El estudio de la interfase de modo cualitativo esta centrado en los requerimientos del usuario, mientras que el modo cuantitativo está centrado en los fines o tareas a realizar y los propios del sistema.

Análisis cualitativo de la interfase

Tres cuestiones son de capital importancia para que se produzca la interfase de manera funcional:

1º La accesibilidad perceptual y motora del usuario, sea éste persona sana o con alguna discapacidad.

2º La necesidad de un nivel cognitivo por parte del usuario acorde a la tarea.

3º El nivel de affordance (Norman, 1986,1990, 1998) que implica cuán intuible es el sistema, para que pueda ser manejado acorde al nivel cognitivo del usuario.

Por otra parte, ningún usuario quiere concentrarse en las interfaces sino en las tareas que desea desarrollar, pues la interfase es la puerta de entrada a la funcionalidad del sistema subyacente. Sheiderman (1987,1993, 2004) plantea como hacer que las interfaces sean cada vez mas entretenidas, dando una serie de reglas para el diseño centrado en el usuario, tales como:

1. Esforzarse para dar mayor consistencia a la interfase.
2. Proveer siempre de una utilidad de alcance universal.
3. Que permita una oferta de regeneración informática.
4. Diálogos para cerrar los programas adecuadamente.
5. Prevención de errores.

6. Permitir si es necesario la inversión de las funciones (dar flexibilidad).
7. Dar apoyo al mando interior para la resolución de problemas que pueda tener el usuario.
8. Reducir la carga de la memoria de trabajo

El sentido relacional entre el usuario y la maquina tiende a una mejora en el mas amplio sentido del termino usabilidad. La nueva manera de entender la interacción con la interfase es como un sistema cada vez más inteligente que pretende no sobrecargar la actividad del usuario tanto para la memoria de corto como la de largo plazo. Desde este modo la manipulación, concepto definido por Norman, tiene un sentido relacional entre las representaciones continuas de objetos, los diseños gráficos, iconos y todo elemento del hardware que permita realizar una operación. Se deja de utilizar sintaxis o lenguaje específicamente según Norman (1990,1998). Este autor plantea que el carácter abstracto de la computadora plantea un desafío especial al diseñador. La computadora funciona electrónicamente, de forma invisible, sin ningún indicio de los actos que está realizando. Y recibe sus instrucciones mediante un lenguaje abstracto, que especifica la corriente interna de control y desplazamiento de la información, pero no esta especialmente adaptado a las necesidades del usuario. Existen programadores especializados que trabajan en eso, lenguajes para decirle al sistema que realice esas operaciones.

Las tareas complejas y los programas deben de contar con toda una serie de conocimientos y talentos. El diseño de un programa exige una combinación de la tarea y de conocimiento de las necesidades y capacidades de los usuarios. Además una serie de conceptos son imprescindibles como factores que se implican en los conceptos de la psicología cognitiva. Las creaciones de objetos físicos que funcionen por medio de una interface eficiente deben tener en cuenta los aspectos formales de dicha teoría, es importante además, tener en cuenta los materiales que se emplean, el peso, la textura, la conductividad, es decir las cualidades físicas pueden apoyar el sentido que se quiere dar al objeto, como si

fuesen las pistas (hacer la tecnología mas amigable, confiable, intuitiva, segura y eficiente para el usuario).

Las características del funcionamiento de las cosas se denominan prestaciones. Son las propiedades percibidas y efectivas del objeto. Las prestaciones que se presenten con pistas claras del funcionamiento de las cosas son las que proveerán el éxito al usuario. Los usuarios al enfrentarse a las maquinas realizan interpretaciones inherentes a las mismas, concepto que se le conoce como modelo mental.

Los modelos mentales de un dispositivo es lo que Norman califica como la imagen del sistema. Por esto la imagen del sistema es el resultado de la estructura física que se ha establecido, que comprende la documentación, las instrucciones y etiquetas. Así el diseñador espera que el modelo del usuario sea idéntico al modelo del diseño. Pero el diseñador no habla directamente con el usuario, todas las comunicaciones se realizan al final por medio de la imagen del sistema y la conducta que adopte el usuario frente al mismo. El modelo del diseño es el modelo es el modelo conceptual del diseñador. El modelo del usuario es el modelo mental elaborado mediante la interacción con el sistema. Ahora bien, debemos introducir un nuevo concepto para comprender cual es la relación de lo anterior con nuestro estudio formal de las interfaces. El principio de la topografía, es un término técnico que significa la relación con dos cosas: la máquina y el usuario, indica que un dispositivo es fácil de utilizar cuando existe cierta visión de conjunto de la usabilidad posible, de modo que los controles físicos y las imágenes se correlacionan en la topografía. Otro principio importante es el de retroalimentación que implica el envió de información al usuario a partir de una tarea a realizar por parte de la computadora, información que implica un acto de resolución de un problema de modo eficaz; es decir con un resultado logrado. Con la retroalimentación se nutre al usuario, cerrando así el ciclo que se establece con el sistema que se esta utilizando. De este modo podemos inferir como ha influenciado la psicología cognitiva, dado que previo a la década de los noventa,

dicha ciencia planteaba en la interacción las nociones de resolución de problemas, con metas y objetivos a cumplir.

Los obstáculos y los conflictos referidos a la interacción eran entendidos en relación a la carga de la memoria de trabajo y las memorias implementadas en los diferentes procesos, como asimismo los tiempos de reacción o los niveles de atención involucrados en los diseños visuales y las lecturas de las imágenes frente a la falta de sintaxis complicadas. Norman y Sheiderman enfatizan la acción que lleva a cabo la interfase en la interacción con el usuario, por esto los procesos interactivos van más allá de los conceptos de ambiente y de las herramientas para llevar a cabo diferentes acciones, ya que se implican las nuevas tecnologías. Así se observa que el estudio de las interfaces es un área de conocimientos en los que convergen gran cantidad de actividades investigativas tales como el diseño visual, el diseño industrial y el factor humano (medicina y psicología), la ingeniería de sistemas informáticos, la ingeniería del software, los analistas de sistemas, la pedagogía, y la sociedad entre otros.

De allí que los tradicionales o clásicos conocimientos de la tecnología del pasado siglo XX, no alcanzan para dar con un entendimiento completo del complejo mundo de la ingeniería de los sistemas informáticos. Por ello, se hace mas necesario encontrar enfoques generalistas o interdisciplinarios, así como aquellos de índole holisticos o sistemáticos para comprender esta tecnología, ya sea como interface relacionada con el hardware o con el software (Von Bertanlaffy, 1992).

Shannon propuso una visión de la tecnología mucho mas radical estableciendo que "la evolución de la computadora es hacia la desaparición de la diferenciación y el establecimiento de la homogeneidad" (Von Bertanlaffy, 1976). Es decir, que la interface sea lo mas transparente o invisible posible, donde la interacción logre una totalidad.

Laurel (1993) toma como centro del diseño y de la interacción por medio de la interface, exclusivamente al usuario y analiza de manera cualitativa el concepto de interface utilizando los postulados de Aristóteles sobre causalidad.

Define una serie de conceptos tales como:

1. Funcionalidad: la funcionalidad se define como lo que cada persona puede realizar con una interface, más que con la capacidad de la funcionalidad de la misma.
2. Programa: se le define como un conjunto de instrucciones que definen las acciones potenciales de una interacción.
3. Aplicación: se le describe como los programas diseñados para otorgar una funcionalidad particular dirigido a los usuarios finales, los cuales no son accesibles a estos usuarios, porque forma parte de la estructura interna de un sistema operativo.
4. Representación: éstas pueden representar un determinado aspecto de las cosas reales o virtuales.
5. Agente: es el término que le da a los sujetos que inician y realizan una acción.

Cuando integra los conceptos de Aristóteles con relación a la Interface, establece que:

1. La causa formal de una actividad entre el ser humano y la computadora es una representación de la acción llevada a cabo por el sujeto, por un agente informático, o por los dos en conjunto.

2. La causa material de una interacción persona computadora o máquina es la representación, a través de la cual, se activan o manifiestan los sentidos de las personas. En una interface, serán los gráficos, el sonido, la música, el texto y los efectos táctiles o cinestésicos.

3. La causa eficiente de una interacción persona computadora es la destreza y las herramientas proporcionadas por el fabricante o los realizadores de la aplicación. Así, la interface se convierte en una especie

de drama "colaborativo" donde los fragmentos de información de cada programa o aplicación, prediseñados por los fabricantes, interactúan con los usuarios.

4. La causa final de una interacción persona maquina es lo que la persona hace o realiza, una vez que ha finalizado la tarea. La causa final involucra la funcionalidad y la experiencia del usuario, interactuar con la aplicación, Laurel (1993; Pág. 47).

Los elementos cualitativos de un sistema referidos a la interacción por medio de una interface se estructuran en orden jerárquico, siendo cada uno de ellos, los siguientes:

1. Accion, es el tiempo de ejecución de una tarea.
2. Carácter, son los patrones del comportamiento de los agentes.
3. Razonamiento, procesos internos de cognición, emoción y razonamiento en los sujetos o en los sistemas los procesamientos.
4. Lenguaje, disposición de palabras y lenguajes en forma de signos verbales, visuales, auditivos y otros fenómenos no verbales.
5. Patrones, es la composición y ritmos de las formas a nivel sensorial y kinestésico del agente o usuario.
6. Representaciones, es la dimensión experimentación de la acción del usuario, sobre el sistema. De este modo los puntos del 1 al 6 se integran unos a otros para la comprensión de la interface.

Por otra parte, International Standard (1999) Norma ISO 13407, referida al diseño de la interface centrado en el usuario, da prioridad a los siguientes puntos:

1º Identificar la necesidad del diseño centrado en el usuario.

2º Entender y especificar el contexto de uso.

3º Especificar los requisitos del usuario y los del sistema a interactuar.

4º Proveer las posibles soluciones.

5º Contrastar el diseño con los requisitos.

6º Evaluar el sistema diseñado con la satisfacción de los requisitos del usuario y los del sistema.

Análisis cuantitativo de la interface

Como lo explicita Constantine (2003) el diseño de una interface centrada en el uso, difiere del diseño centrado en el usuario, primeramente porque el foco de atención ya no es directamente el usuario, sino el uso que éste hace del sistema, o sea en las tareas que éste debe realizar.

El aspecto mas importante del diseño centrado en el uso radica en que el diseño renuncia al modelo cualitativo, para así dar con un procedimiento de diseño de interface en favor de las soluciones finales que se derivan directamente de la definición de procedimientos y de las tareas. Todo lo cual refleja como se debe proceder para la realización de aquellas por parte de los usuarios. Cada uno de estos procedimientos se deriva del desarrollo de aplicaciones informáticas, software o dispositivos de entrada y salida de datos.

Según Raskin (2000), esta forma de evaluar las interfaces, pertenece al modelo clásico de metas, objetivos y operaciones, como así también métodos y selección de reglas (Goals, Objets, Methods, and Selection of rules; GOMS), el cual fue propuesto por Card, Morán y Newell (1983). El modelo GOMS permite predecir la duración de una tarea específica. Se realiza un análisis de la tarea como una sucesión de acciones para el logro de una meta, de modo que dicho modelo se enfoca en los procesos cognoscitivos involucrados en el logro de una tarea. Por lo tanto, se diferencian dos niveles: el nivel psíquico o de programación de la tarea a realizar por el usuario, es decir el nivel cognoscitivo y el nivel físico involucrado en la acción realizada por el usuario.

Este modelo determina las siguientes características: en primer lugar, el logro de metas por parte del usuario. Las metas están en la memoria a largo plazo de aquél, de modo que esto permite evaluar los errores que comete y reinician la acción. Las metas se estructuran en forma jerárquica. En segundo lugar, se encuentran en forma intermedia, los objetivos y operaciones que el usuario debe

realizar, que son las acciones básicas que pueden dividirse en las que se producen en la mente misma del usuario o sobre el sistema. También debemos mencionar los métodos, que son las diversas formas de realizar las tareas por los medios descriptos como hardware o software, en los que se debe operar en base a reglas. Sintetizando GOMS es una teoría de las habilidades cognoscitivas implicadas en las tareas de interacción persona computadora, que está basada en el procesamiento de la información y los diferentes tipos de memoria o fases utilizados en este procedimiento.

El GOMS se utiliza para construir y predecir la estructura particular de cada tarea, con el objeto de identificar y prever errores. De modo que este método incluye tres principios generales:

1. Eliminar las operaciones innecesarias para mejorar las habilidades cognoscitivas, es decir no sobrecargar ni la memoria de largo plazo ni la memoria de trabajo.
2. Las tareas tienen un sistema de operaciones específicas que involucran las habilidades cognoscitivas.
3. El desempeño de la tarea puede mejorarse proporcionando métodos de error-recuperación.

Según Raskin (2000), el modelo GOMS permite entender la forma en que las personas interactúan con las maquinas, definir sus métodos de trabajo, seleccionar los procedimientos, calcular tiempos y velocidades de la ejecución de tareas. Ejemplos de la duración de tiempo requerido por el usuario, entre otros:

Preparación mental (M) igual a 1.35 s, tiempo que toma el usuario para preparar mentalmente el próximo paso.

Respuesta (R) es el tiempo que debe esperar el usuario para que la computadora responda la entrada de información.

De esta manera se puede observar como se utiliza la cuantificación para realizar la interacción con la interface a nivel cuantitativo.

Análisis del diseño de la interface dirigido por la tecnología

Cuando la tecnología dirige el diseño de la interface, esta a cargo de los desarrolladores que solo están enfocados en los problemas estructurales y funcionales de un sistema. Solo contribuyen a la formación de los mismos grupos de técnicos, cuyo enfoque principal es la arquitectura interna y tecnológica del sistema. La calidad solo es medida en base a los defectos del producto y rendimiento, es decir a la calidad del sistema en si, y su implementación esta dada previa a cualquier validación de usabilidad humana ya que solo interesa la validación funcional, y a la solución de requisitos funcionales (Gulliksen & Göransson, 2003).

III. METODOLOGIA

Panorama General

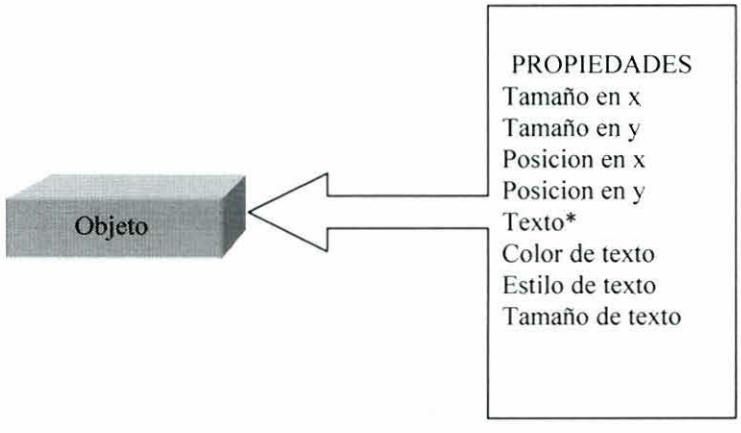
El trabajo fue desarrollado en 3 lugares principalmente, en un principio de estuvo trabajando en la planta baja del CEDIT en colaboración con Genaro Soto Zarazúa y Luís Morales haciendo pruebas de la respuesta de la primer versión del programa con una tarjeta Spartan 3. Las pruebas realizadas fueron especialmente orientadas a la compatibilidad del programa con el hardware desarrollado por Soto – Zarazúa, también se realizo una prueba de destrucción dejando la etapa de potencia funcionando por 48 horas continuas.

Después de ello se laboro la segunda versión del programa en mi domicilio, donde se checo la compatibilidad del sistema con las alternativas de comunicación, en esta etapa las opciones de comunicación en cuestión fueron el RS-232 y el puerto paralelo.

La etapa final se desarrollo en los invernaderos de la UAQ Amazcala, El Márquez, Querétaro; donde también se desarrollo la tercera y ultima versión del software ya corrigiendo las deficiencias de las dos versiones anteriores del software, ya integrando la comunicación del software con la etapa de potencia y el correcto funcionamiento del sistema en conjunto. Así bien ya estando listo el sistema se realizaron las pruebas finales de desempeño, y por consiguiente la integración del sistema completo.

Primera versión

En un principio el programa se hizo en Turbo C++ tomando como base un mini ambiente grafico en el cual se pretendía simular ventanas, botones cajas de texto, y etiquetas. Cada uno de estos componentes esta constituido por un constructor que inicializa los parámetros, parámetros como tipo de letra, tamaño de letra, estilo, tamaño del componente, color del texto y del componente; este ambiente grafico se inicio como una opción para presentar las practicas de programación avanzada que mas tarde termino siendo la base de lo que fuera la primera versión para el control de horarios.



Cuadro 1. Arquitectura general de las clases

A partir de esta idea fue como se empezó a construir este pequeño programa, la primer parte como su nombre lo indica fue el constructor, que como su nombre lo sugiere inicializa las propiedades. Otra característica importante de este programa es la inclusión del puntero, para una interacción más amigable en vez de un manejo por comandos (Terminal). Con el puntero incluso en esta versión el programa ofrecía muchas expectativas sobre todo para la interacción con los botones y cajas de texto. También (aunque primitiva) se creo una base de datos donde se guardan los horarios para cada día de la semana permitiendo al usuario especificar la cantidad de programaciones por día (domingo - sábado).

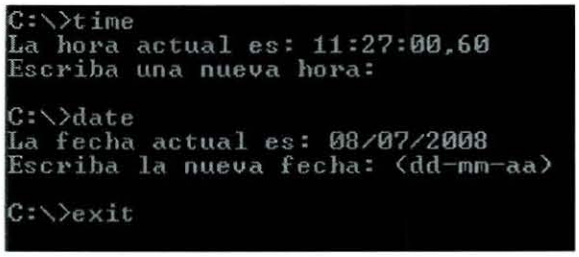
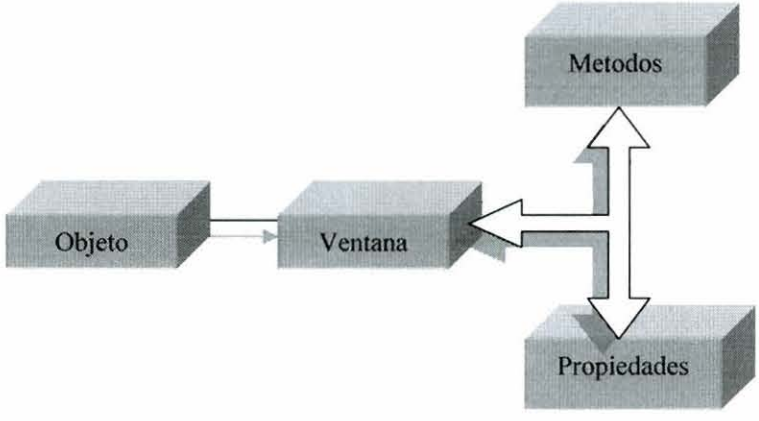


Figura 2. Terminal, comandada mediante comandos.

Las cajas de texto así como los demás componentes tenían la opción de ser relacionados con una ventana, o bien para dejadlos libres (fuera de una ventana).

En una primera etapa se pensó de manera muy general las necesidades básicas de cada uno de los componentes. En primer instancia la ventana, esta debía tener un titulo, un color de fondo para este titulo, el color de la ventana, el color del texto del titulo, tipo de letra del titulo, tamaño del texto, posición en x de la ventana, posición en y, tamaño en x, tamaño en y, y finalmente una bandera que indicase si la ventana esta cerrada, minimizada o si no esta activa, por ello en el transcurso de esta breve explicación se le llamara la bandera de estado; además de estas propiedades debía tener métodos (funciones) con las cuales se pudieran cambiar o adquirir estos valores de forma dinámica.

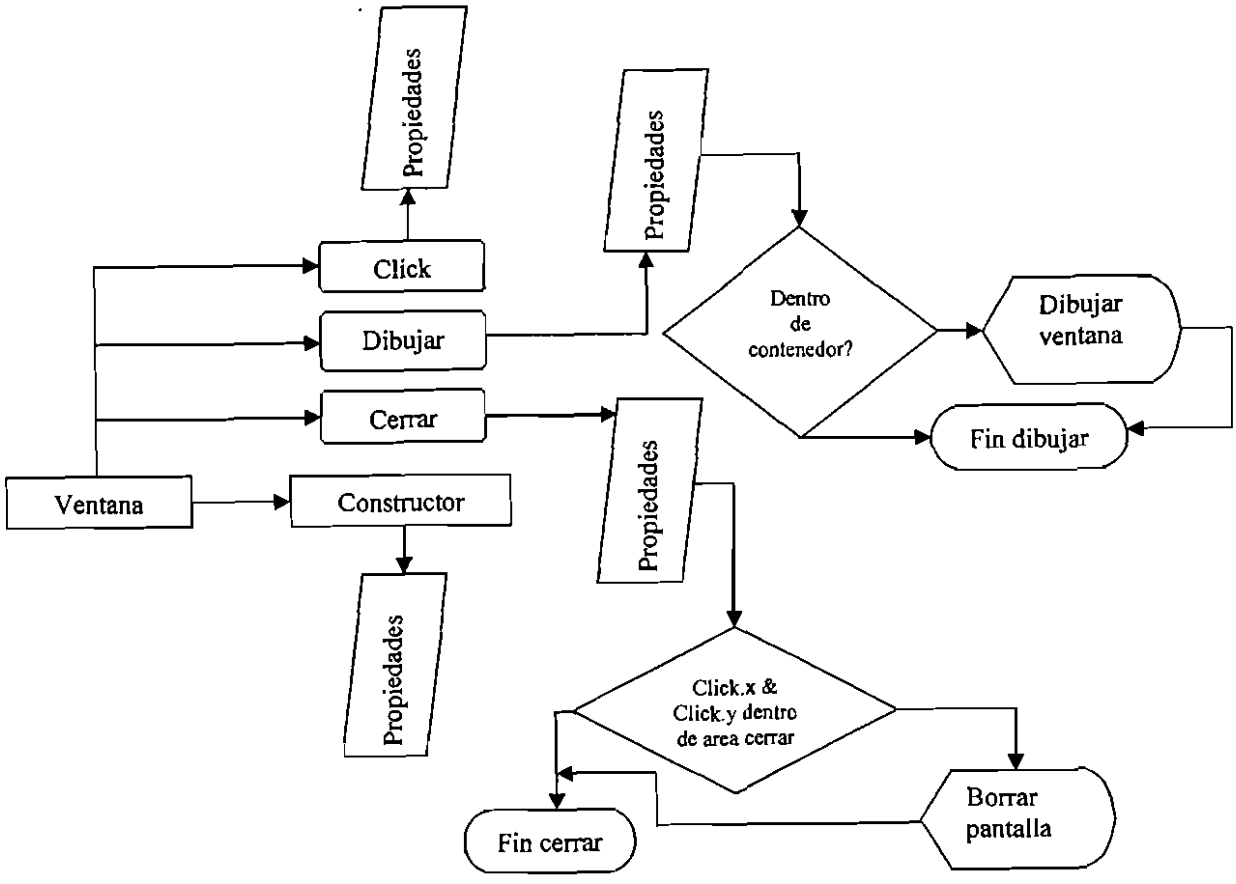


Cuadro 2. Organización de la clase

Al dibujar una ventana, hablando en un contexto muy básico, esta se puede centrar en el espacio total de la pantalla o bien especificarle una posición absoluta

en el plano xy; si la opción optada es el darle una posición absoluta es necesario conocer a partir de donde se ha especificado a dibujar la ventana, y para facilitar el diseño visual sobre este mini lenguaje grafico se a convenido no mostrar una ventana que no se muestre completamente sobre el área de la pantalla, esto con la simple justificación de que la función mover ventana no esta contemplada dentro del esquema básico y eso de dibujar una ventana "a medias" no implica nada constructivo. Cuando el método dibujar es implementado el susodicho necesita conocer la posición a partir de la cual esta ventana es dibujada al mismo tiempo de validar si esta se dibuja completamente sobre la pantalla evitando ser truncada (a si sea 1 píxel) es por ello que también se describió un método que regresa o modifica cada una de estas propiedades.

Una vez validada la ventana se hacen los trazos correspondientes respetando los colores asignados a la ventana (no habiendo muchos colores, ¡16 colores!) y listo se tiene una ventana. Ya creada dicha ventana empieza una de las etapas más importantes del programa la interacción con el usuario. Esta ventana tiene también un método llamado clic el cual al ser llamado indica a través de una bandera si el mouse esta sobre el objeto, o en su defecto esta sobre el objeto y se esta presionando el botón izquierdo del Mouse. Al conocer este estado a partir de clic se crea un nuevo método que aunque primitivo eficaz cierra la ventana infiriendo sobre las propiedades si clic ocurrió sobre la cruz que fue exprofesa a cerrar la ventana pintando un rectángulo negro sobre lo que fuese la ventana al tiempo de establecer la bandera de estado a cerrada así como el repintar las ventanas aun activas.



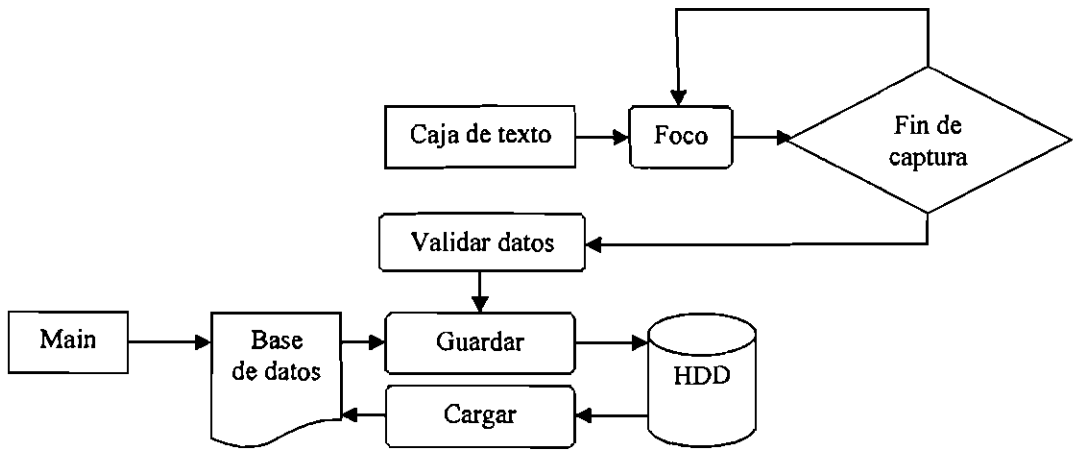
Cuadro 3. Diagrama de flujo de la clase ventana, una de las clases mas representativas y elaboradas.

De manera similar los otros controles como etiqueta, caja de texto y botón heredan las propiedades posición en x, en y, tamaño en x, en y, color de texto, texto, tipo de letra, tamaño de letra, color del botón o color del fondo del texto para etiqueta y caja de texto. Lo único distinto entre la ventana y estos otros objetos es una variante en el método dibujar que bien puede hacer las validaciones para saber si el objeto en cuestión con las propiedades establecidas en dado momento esta dentro de los límites gráficos visibles con respecto a la pantalla completa o bien con respecto a una ventana que le contenga. Lo que en resumidas cuentas se puede entender de una manera más simple para el lector como si el control se dibuja con respecto a una ventana o con respecto a la pantalla.

En cuanto al botón únicamente el evento clic se monitorea para que al ser llamado a través del cursor, se ejecute un comando o una serie de estos. En

cuanto a la etiqueta constituye un elemento inerte por decirlo de alguna manera o con un termino mas propio a estos menesteres sin interacción con el usuario; esto debido a la falta de reacción al dar clic o no sobre este.

La caja de texto podría considerarse el elemento mas elaborado (después del objeto ventana) dado que esta implica la interacción con la rudimentaria base de datos, validación de la entrada del usuario, un indicador de foco, cambio de tamaño dinámico y borrado para corrección en línea.



Cuadro 4. Diagrama de flujo de los métodos para la base de datos.

Primeramente la base de datos consta de un arreglo de una estructura que con una función guardar se escribe esta base de datos en un archivo y por consiguiente una función cargar lee el archivo y carga los valores cuando el programa hace uso de ellos. En cuanto se tiene una manera de guardar/cargar los datos la caja de texto muestra su potencial, este es capturar en primera instancia para que locación de la base de datos serán los datos capturados, ya conocido el destino de los datos se captura hora, minuto y segundo del inicio de la programación, en seguida de ello se captura hora, minuto y segundo del fin de la programación.

La validación de la entrada del usuario respecta al acto de estipular si los datos ingresados son correctos para el fin que estos son capturados. El ejemplo más lucido de ello es la hora, minuto y segundo de inicio o final que como es bien conocido nuestro sistema horario consta de valores establecidos y sumamente

estrictos para representar un momento en el día con respecto al giro de la tierra. La hora bien puede variar entre 0 – 23, y los minutos y segundos entre 0 – 59. Lo que se comprueba a grosso modo es que no existan valores como 67 horas, 81 minutos, 99 segundos; esto agregado a la misma no existencia de "a9" horas, "qm" minutos, "i3" segundos.

El indicador de foco pudiese parecer cosa estética o insulsa como algo relevante, pero contemplando que existen 2 cajas de texto por cada programación y el limite para el numero de programaciones es de 12 (es decir una latente posibilidad de tener 24 cajas de texto) al tener al menos 2 cajas de texto activas para el usuario resulta sumamente importante en que caja de texto esta escribiendo es por esto que la caja de texto en lugar de tener un borde negro este es azul cuando esta es la que tiene el foco de escritura.

Aunque mas que una ventaja para el usuario quizás represente un grado de flexibilidad para la programación del programa (valga la redundancia) es común que en ocasiones el programador establece un tamaño de la caja de texto y este al entrar mas de los caracteres esperados muestre solo algunos de ellos; en este caso el control caja de texto establece un tamaño inicial y conforme se agreguen/eliminen caracteres esta caja de texto se ajusta al tamaño justo, no mas, no menos; un principio de la filosofía "slim" solo lo necesario.

El borrado dinámico, aunque pudiese considerarse ya incluso en las primeras versiones no fue un hecho. Y lo único que hace es borrar el último carácter digitado.

Al contemplar las necesidades del programa para que este ejecutase múltiples tareas (o al menos 2) el lenguaje dio de si limitando esta necesidad lo que en resumidas cuentas obligo al proyecto a ser migrado a algún otro lenguaje de programación visual orientado a objetos; pese a la basta cantidad de lenguajes de esta índole la selección del mas adepto no fue muy profunda y solo con base el que estuviese disponible en el campus se torno nuestra atención a Borland Builder C++ 6.0. Que ofrece una IDE básica así como bases de datos que bien pudieren ayudar en el desarrollo, en cuanto a las limitaciones para hilos y procesos múltiples no existe ningún problema.

Segunda versión

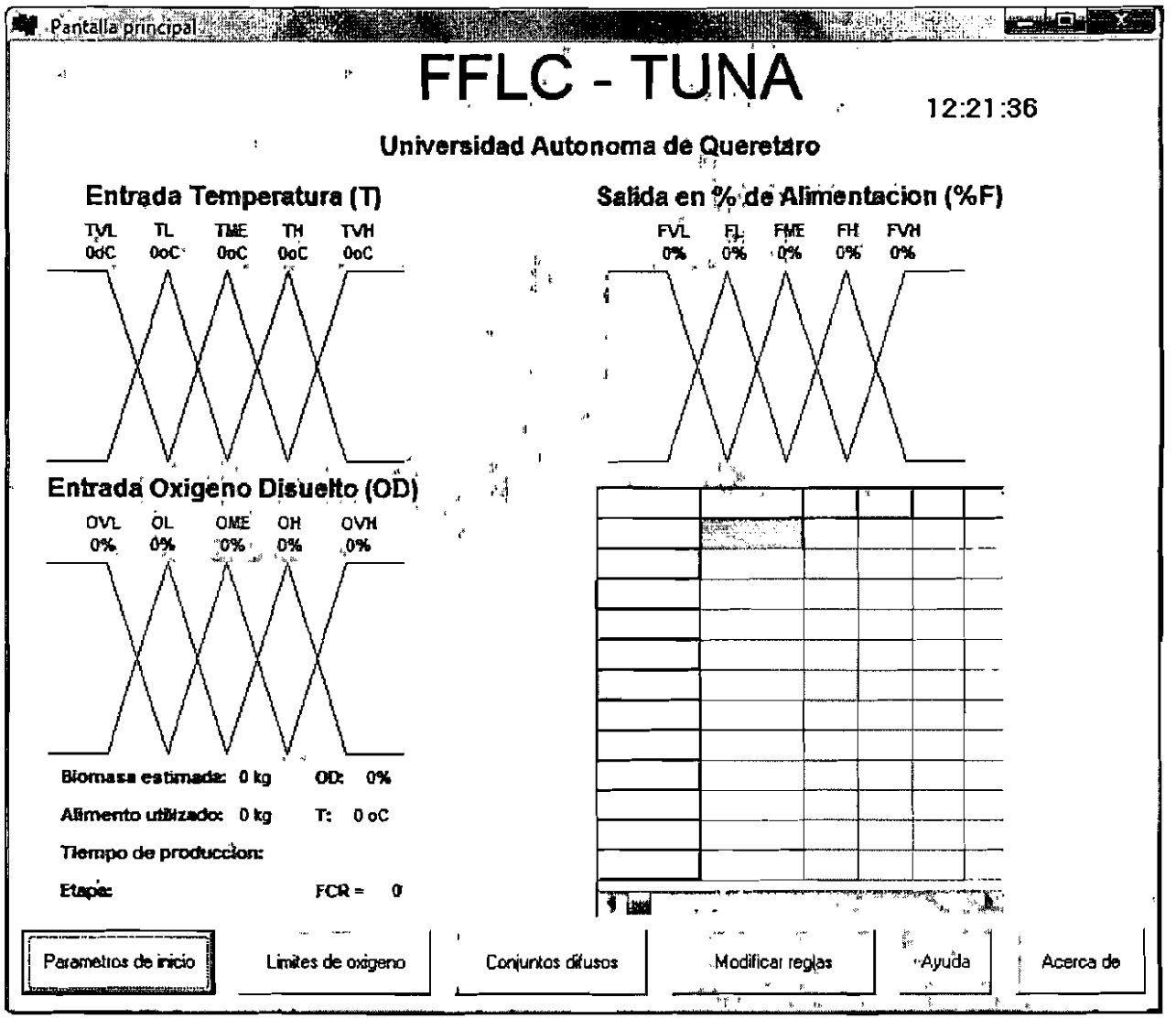


Figura 3. Pantalla principal en la segunda versión.

En un primer acto se vuelve a diseñar las pantallas básicas del programa, la pantalla de parámetros iniciales, conjuntos difusos y reglas. Ya migrados a este lenguaje la construcción de un botón u otro control cualquiera se resume a 2 pasos seleccionar que tipo de control y dibujarlo sobre el formulario. En cuanto a la pantalla de parámetros iniciales se ha hecho una vinculación entre este y el formulario principal, esta función se ve reflejada en el caso de modificar valores en la ventana de parámetros iniciales y al aceptar cambios estos son actualizados en

la pantalla principal. Incluso los controles utilizados en este formulario tienen una forma peculiar para funcionar, en la ventana de parámetros iniciales se muestran solo etiquetas, y al dar doble clic sobre algún parámetro esta etiqueta es sustituida por una caja de texto para entrar el nuevo valor del parámetro en cuestión.

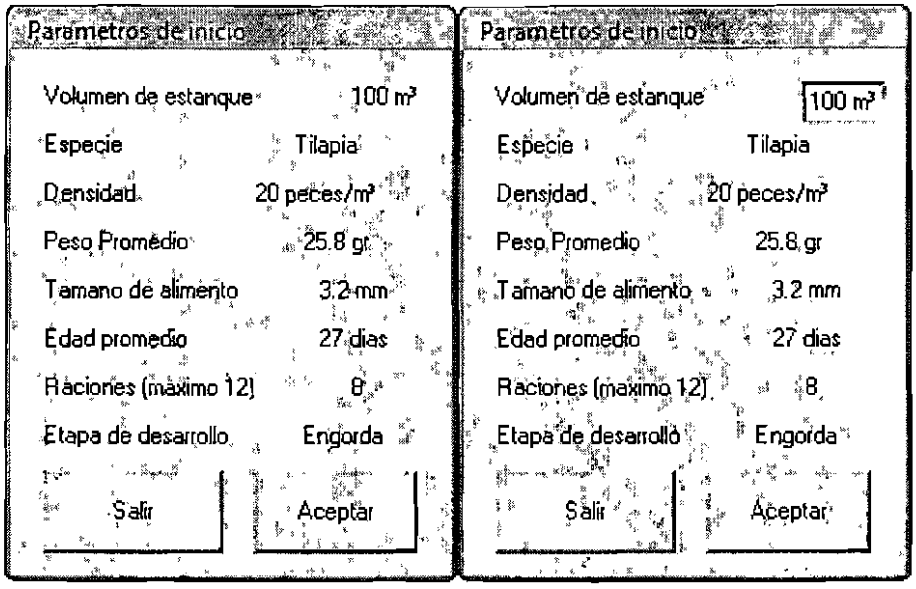


Figura 4. Ventana de parámetros inícíales (derecha), interacción a través de cajas de texto al hacer click en dato a modificar.

Partiendo del supuesto de que todo puede tener ventajas y desventajas, la desventaja de esta ventana de parámetros iniciales es el hecho de que el programa no tiene control sobre lo que el usuario introduce en cada uno de los parámetros, prácticamente puede ser cualquier texto. Por ejemplo en la casilla de etapa el usuario bien puede introducir por equivocación la densidad del estanque, sin que esto represente una condición de error. Al dar clic en el botón aceptar el programa toma los valores de cada uno de los parámetros y los confina en un búfer de escritura para ser escritos en el archivo de configuración; en caso contrario los cambios no son reportados a este búfer y por lo tanto no son grabados.

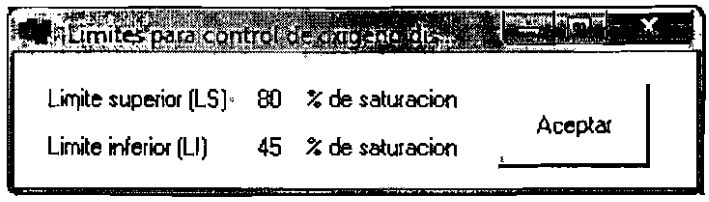
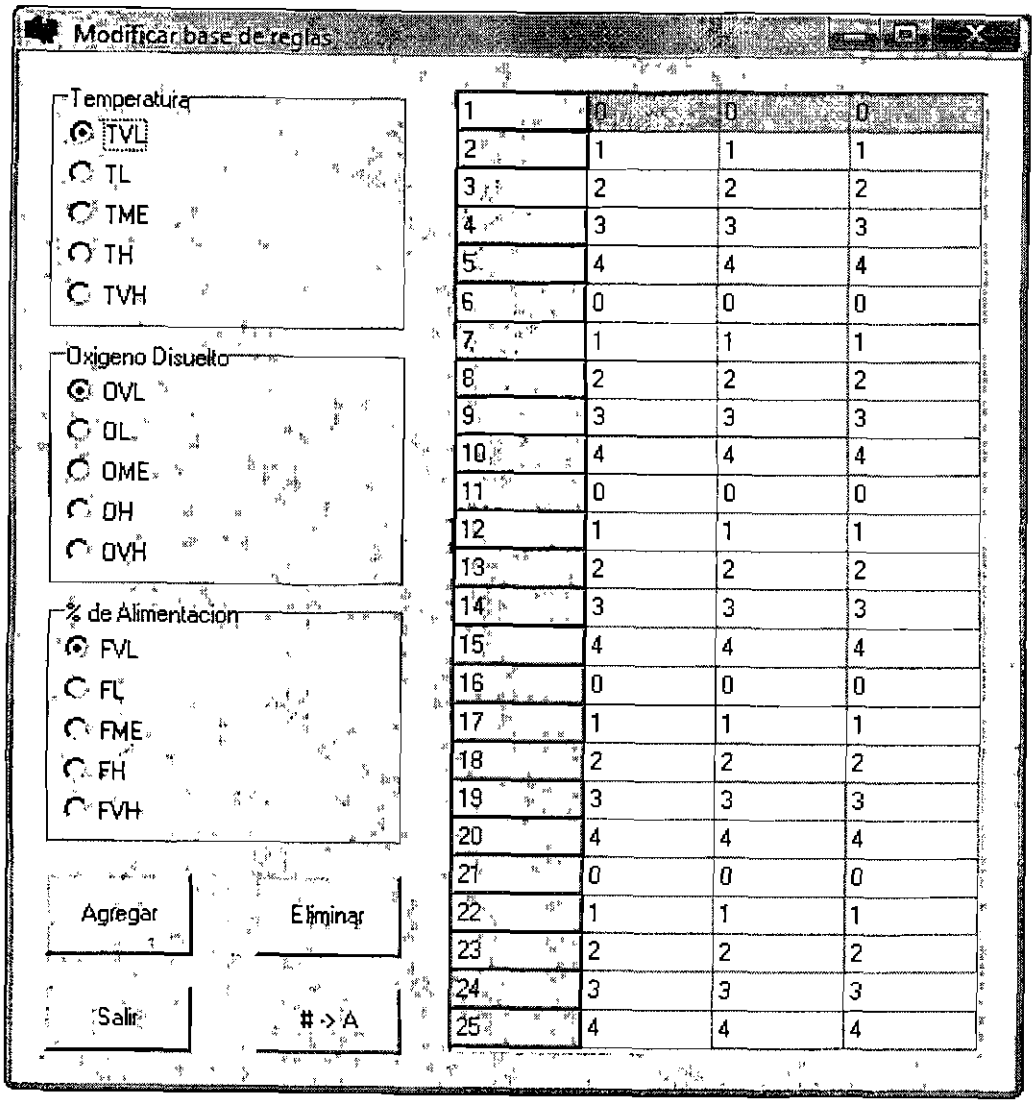


Figura 5. Ventana de límites de oxigeno disuelto.

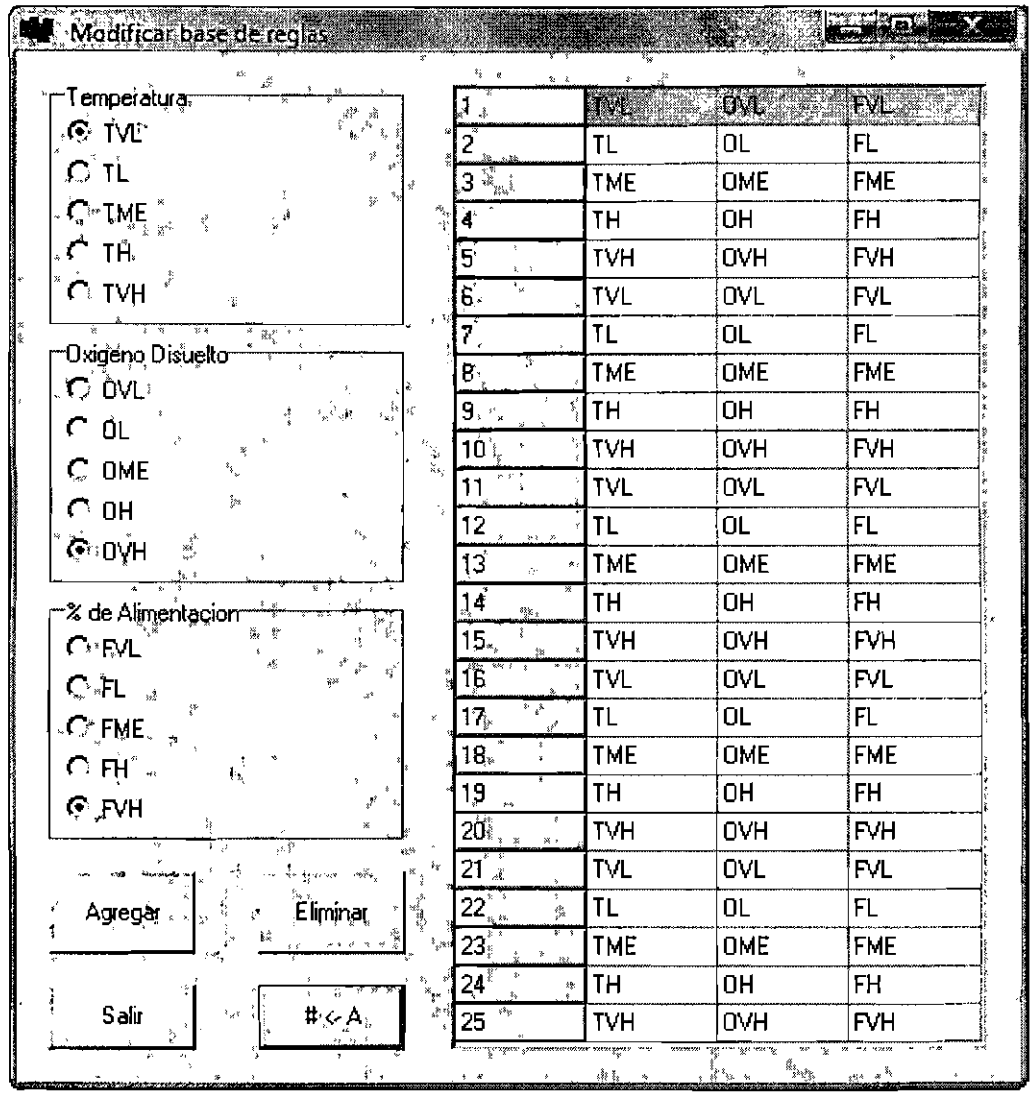
Para el caso de los limites superior e inferior de oxigeno disuelto, lo único que se hace es inscribir una condición para evitar el funcionamiento erróneo del sistema, esta condición consta únicamente de prohibir que el limite superior sea menor al inferior; de igual forma se guardan los cambios vía el búfer común de escritura.

En cuanto a la pantalla de reglas, con ayuda de una tabla se capturan estos valores y puede ser editada directamente en 2 modos, modo literales introduciendo VLX, LX, MEX, HX y VHX donde X puede ser oxigeno disuelto, alimentación o temperatura. Es importante remarcar que a pesar de que se indica el modo actual (literales o valores) una vez más el programa no tiene ningún control o filtro sobre lo que el usuario entra en cada campo. Cuando la edición se hace sobre valores el 0(cero) corresponde muy bajo, 1 a bajo, 2 medio, 3 alto, y 4 muy alto; al cambiar de modo (literales o valores) una función se encarga de traducir de literales a valores y viceversa. El botón eliminar, elimina una regla a la vez mientras que reiniciar borra todas las reglas, de igual forma que las pantallas anteriores cancelar descarta cambios al no reportarlos al búfer de escritura y aceptar los escribe en el búfer.



1	0	0	0
2	1	1	1
3	2	2	2
4	3	3	3
5	4	4	4
6	0	0	0
7	1	1	1
8	2	2	2
9	3	3	3
10	4	4	4
11	0	0	0
12	1	1	1
13	2	2	2
14	3	3	3
15	4	4	4
16	0	0	0
17	1	1	1
18	2	2	2
19	3	3	3
20	4	4	4
21	0	0	0
22	1	1	1
23	2	2	2
24	3	3	3
25	4	4	4

Figura 6. Ventana de reglas mostrando números representativos de las variables.



1	TVL	OVL	FVL
2	TL	OL	FL
3	TME	OME	FME
4	TH	OH	FH
5	TVH	OVH	FVH
6	TVL	OVL	FVL
7	TL	OL	FL
8	TME	OME	FME
9	TH	OH	FH
10	TVH	OVH	FVH
11	TVL	OVL	FVL
12	TL	OL	FL
13	TME	OME	FME
14	TH	OH	FH
15	TVH	OVH	FVH
16	TVL	OVL	FVL
17	TL	OL	FL
18	TME	OME	FME
19	TH	OH	FH
20	TVH	OVH	FVH
21	TVL	OVL	FVL
22	TL	OL	FL
23	TME	OME	FME
24	TH	OH	FH
25	TVH	OVH	FVH

Figura 7. Ventana de reglas mostrando palabras representativas en vez de números.

Para la pantalla de conjuntos difusos se tienen los parámetros de un alimentador solamente, esto es igual para las pantallas anteriormente descritas es por un solo alimentador, esta pantalla no tiene ninguna función especial simplemente se capturan los datos y estos pueden ser guardados o no. De nuevo el programa no tiene control para limitar si los parámetros que se han capturado tienen o no el formato que les corresponde.

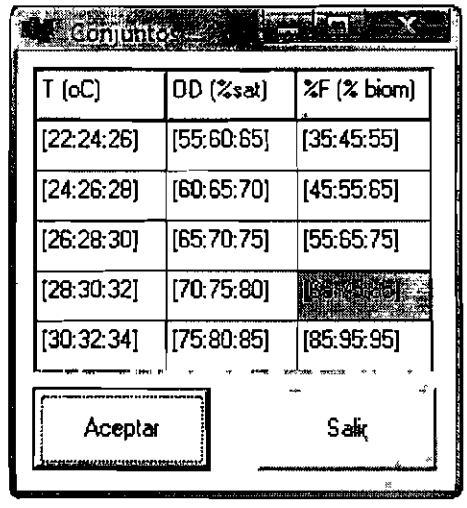


Figura 8. Ventana de conjuntos difusos para controlador

Volviendo a la pantalla principal se tienen algunos de los parámetros iniciales, los botones para acceder a cada una de las pantallas ya descritas y una parte importante las graficas en directo de los valores leídos de temperatura y oxigeno disuelto; así como la grafica de la respuesta del sistema.

Después de haber ya hecho un primer bosquejo se procede a considerar métodos de entrada/salida para la respuesta por parte del control. Este es el punto donde esta nueva opción se ve limitada. La comunicación con el puerto paralelo, para la cual no existe modo de escribir datos en el puerto. Debido a que la tarjeta que se diseño para este sistema esta pensada para interactuar con el puerto paralelo de la computadora es necesario establecer comunicación entre el puerto paralelo y esta mentada tarjeta, al hacer una búsqueda exhaustiva acerca de las opciones para llevar esta comunicación a cabo se concluyo que no era posible. Es por ello que la solución emprendió una búsqueda para soluciones alternas.

Tercera versión

Al cabo de 3 días de búsqueda y análisis de soluciones alternas se encontró un lenguaje de programación orientada a objetos que es de libre distribución y también la licencia para vender los programas desarrollados en este lenguaje no tiene costo alguno. Este es Microsoft Visual C# 2008 Express Edition aunado a la ventaja de no tener ningún costo, existe una librería para el manejo del puerto paralelo desarrollada por un externo que tiene completa compatibilidad con este lenguaje.

En el caso de que el programa necesitase ser actualizado y tener mas opción es de comunicación USB, FireWire, Wireless u otro, el programa se comunica en directo con el sistema operativo mas conocido en el mundo entero Microsoft Windows XP, incluso soportado en Microsoft Windows Vista en cualquiera de sus variantes.

Esta vez un desarrollo poco menos ambicioso pero con la misma meta (rediseño de la interfaz grafica con el usuario, desarrollado por Soto-Zarazúa Control Fuzzy) se opto por seccionar el curso en 2 etapas; Control automático por horarios, y el Control Fuzzy.

En la primera fase del programa se ha hecho un completo rediseño a la forma de programar los horarios así como el formato en el que estos son guardados.

Control por tiempos

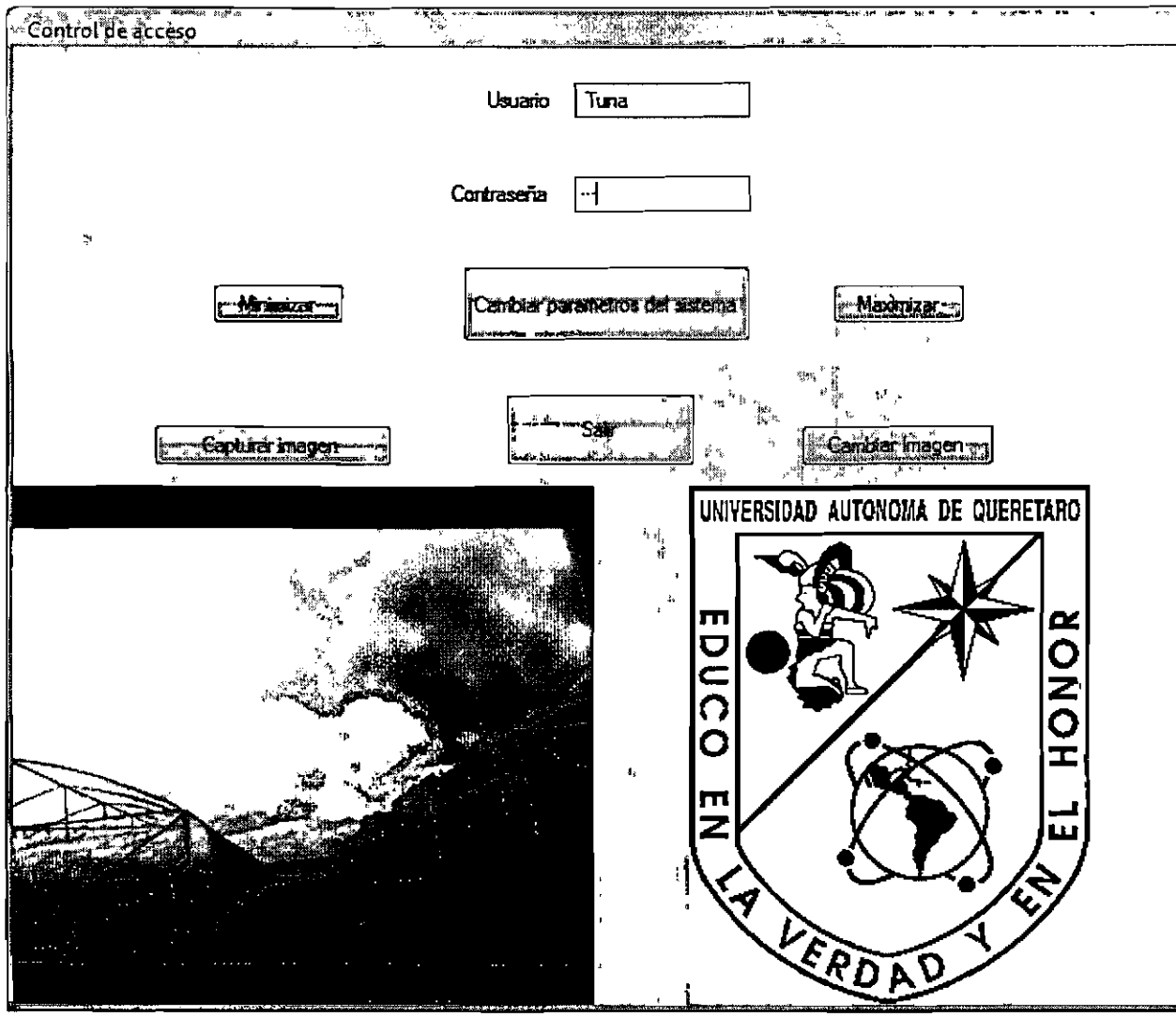


Figura 9. Pantalla principal tercer versión con captura de video en línea.

La pantalla principal es ahora un formulario con 6 botones, video, imagen, 2 cajas de texto y 2 etiquetas. Esto es en si una pantalla de acceso a la programación automática por horarios. Las etiquetas aunque no meramente inertes como en un principio su uso se extralimita a indicar el contenido esperado en las cajas de texto correspondientes nombre de usuario y contraseña; los botones son Cambiar parámetros del sistema, salir, minimizar, maximizar, cambiar imagen y capturar imagen. Y la dinámica del sistema es, minimizar y maximizar desempeñan la conocida y poco trivial tarea de ocultar la pantalla principal a la

barra de tareas y extender la pantalla principal para cubrir toda el área visible del escritorio, respectivamente. Cambiar imagen permite al usuario seleccionar una imagen para ser desplegada en el recuadro correspondiente, capturar imagen toma la imagen actual del video en vivo y la guarda con el nombre especificado por el usuario. En tanto al botón salir y cambiar parámetros del sistema respecta solamente son accesibles si la contraseña y nombre de usuario son validas de lo contrario se limitan a ser presionados sin ninguna acción aparente.

Una vez entrada la contraseña correcta el sistema puede configurarse y es cuando al dar clic en cambiar parámetros se inicia este proceso de configuración.

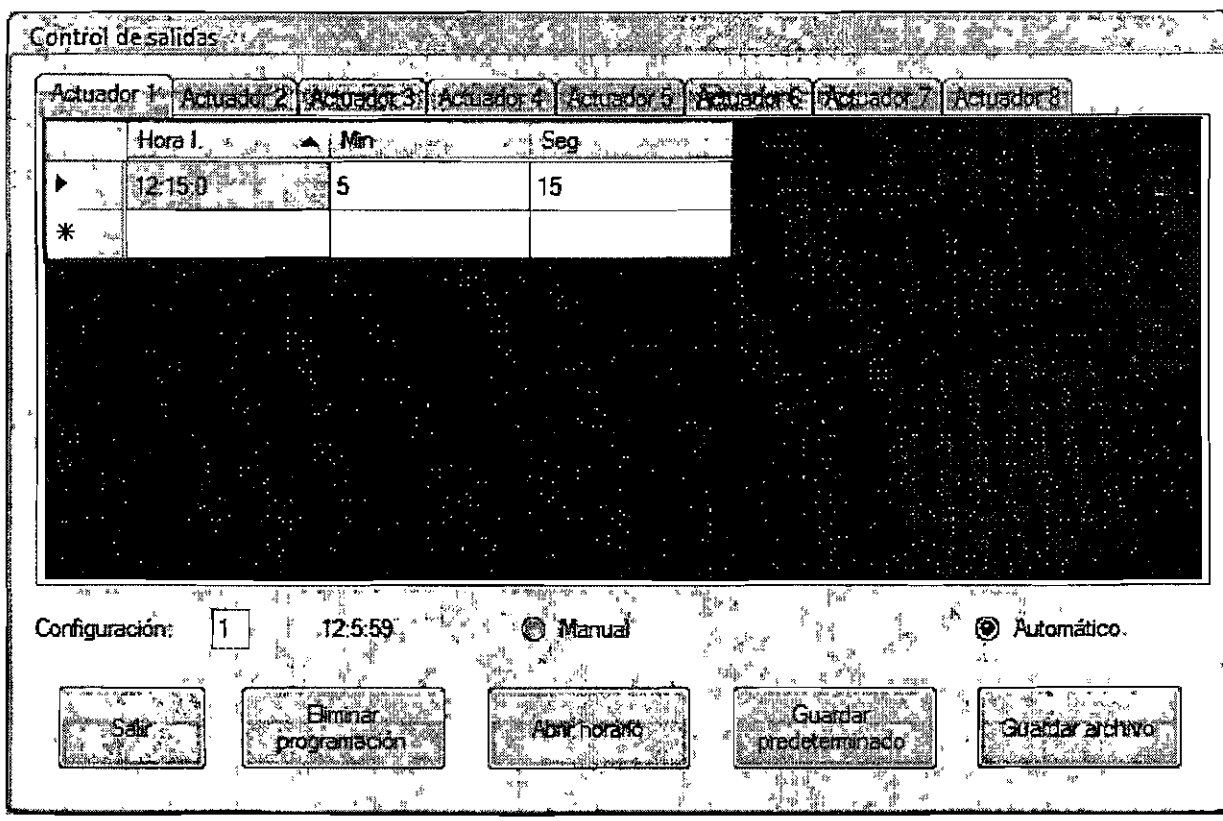


Figura 10. Ventana de configuración de horario

El programa en general se compone de 5 botones, 2 botones radiales, 2 etiquetas, 1 caja de texto, 1 menú emergente, 9 temporizadores, 1 cuadro de dialogo para guardar archivos, 1 cuadro de dialogo para abrir archivos, 8 pestañas y 8 tablas. Las pestañas obedecen a la configuración de un dispositivo; al

seleccionar cada una de ellas es cargado el contenido desde un archivo con el nombre "horarioX.xml" donde "X" es un numero entre 1 – 8 que indica el numero del dispositivo, junto con este, es también leído el esquema de cada tabla nomenclado como "schemX" donde X es una letra entre a – h que corresponden a los dispositivos 1 – 8 respectivamente; este esquema indica al control tabla el tipo de datos de cada campo así como su longitud en caso de ser necesario (por ejemplo en cadenas de caracteres, lo cual no se aplica en este caso).

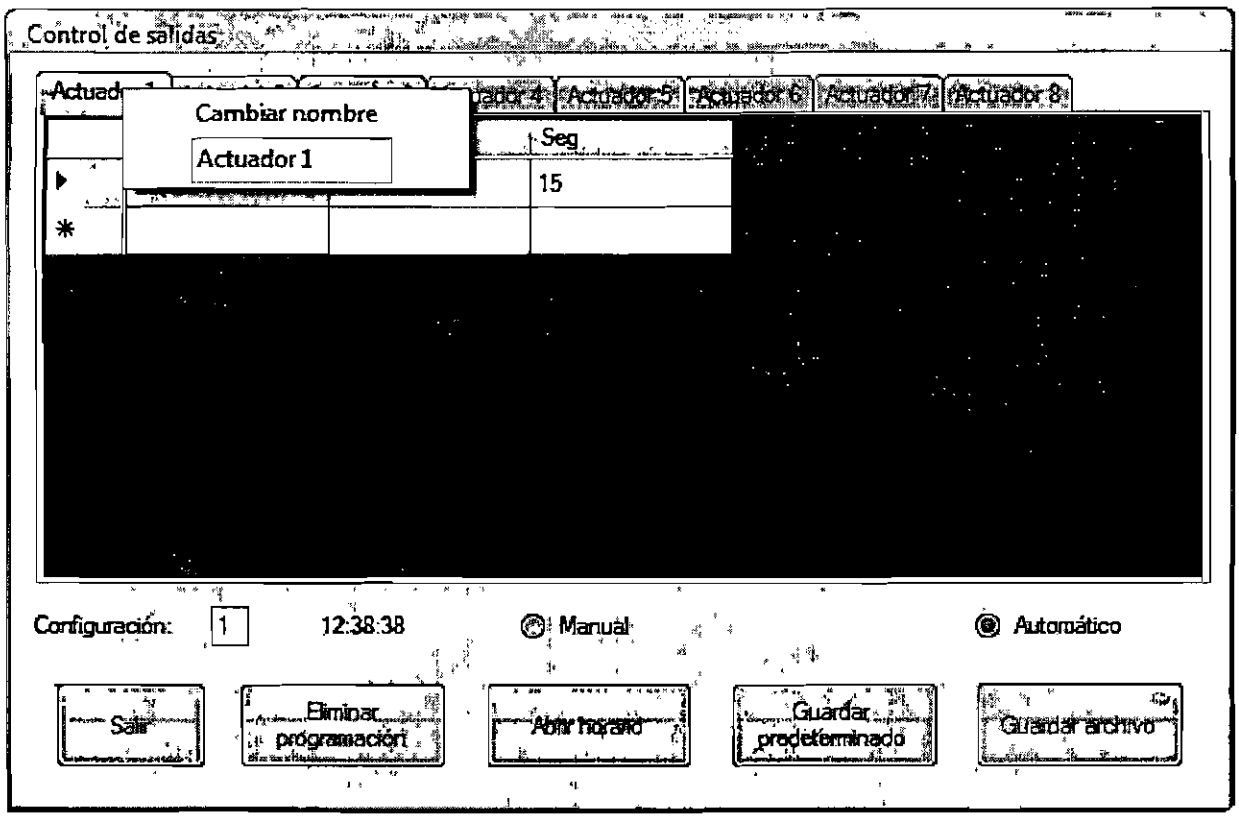


Figura 11. Cambio de nombre del dispositivo a través de menú emergente

Cada pestaña esta vinculada al menú emergente para cambiar el nombre asignado a cada dispositivo, tal menú emergente esta compuesto de un botón y una caja de texto. Al dar clic derecho sobre la pestaña es capturado el nombre actual de la pestaña para ser modificado o simplemente sustituido por el que el usuario considere mas adecuado, al dar clic izquierdo en el botón cambiar nombre los cambios son guardados en el archivo de configuración "etil.xml". Cada tabla

tiene la capacidad para programar 50 activaciones por dispositivo. Los primeros 8 temporizadores controlan el tiempo de encendido de cada dispositivo mientras que el noveno es utilizado para hacer un barrido a través de todas las tablas en busca de una activación programada. Cuando una activación coincide con la hora actual al temporizador correspondiente le es cargado el valor de los campos Min y Seg para lo cual son primero los minutos convertidos a segundos, el resultado es sumado al campo Seg y finalmente a milisegundos; el valor resultante se carga directamente al temporizador correspondiente. Despues de haber cargado el valor al temporizador el programa se desentiende del temporizador, el temporizador por su cuenta mantiene la salida en alto y cuando termina el lapso programado apaga la salida. En cuanto al orden de precedencia para las programaciones la hora de inicio que primero sea encontrada es la que lleva en control y en el caso de los minutos y segundos el último valor encontrado tiene la precedencia; es muy importante que esto sea considerado por el personal que ha de programar la tabla de horarios.

A pesar de que a la tabla se le pueden agregar mucho más de 50 programaciones es recomendable no exceder este límite para preservar la estabilidad y buen funcionamiento del sistema. Al hacer caso omiso de esta advertencia no se garantizan óptimos resultados.

Una de las etiquetas sirve para indicar el contenido de la caja de texto en este caso "Configuración:", o bien, la otra sirve para mostrar la hora actual del sistema HH:MM:SS (HH = Hora, MM = Minuto, SS = Segundo).

El botón salir guarda los cambios en los horarios y regresa a la pantalla de acceso. El botón guardar predeterminado guarda la configuración actual como la configuración que el sistema carga cada vez que el programa es inicializado. El botón guardar archivo guarda las configuraciones con un nombre distinto y no las establece como predeterminadas, si surge la necesidad de hacer un respaldo o tener configuraciones alternas esta es la manera de hacerlo; en una primera

instancia se preguntara por el nombre con el cual se guardan los esquemas de las tablas seguido de esto el programa ha de preguntar el nombre general de la tabla añadiendo un numero del 1 – 8, respectivamente, al nombre de los archivos. El botón abrir horario es la contraparte del ulterior, siendo que este carga la programación de los horarios desde una ruta que ha de ser especificada por el usuario. Eliminar programación como su nombre lo sugiere elimina la prógramación seleccionada, al ser presionado repetidamente se eliminan las celdas que hubieren estado abajo de la antes eliminada. La tabla aparentemente nunca estará vacía, quedando siempre el renglón de inserción. Este renglón es parte del control tabla y es meramente virtual ante el control de salidas.

Para desactivar una programación basta con dejar en blanco la casilla de hora de inicio. Para cada pestaña y tabla se sigue la misma dinámica. En cuanto a los botones radiales "Manual" y "Automático" respecta, el botón automático siempre es la opción predeterminada aun cuando se regresa a la pantalla de acceso. Todo lo anteriormente descrito corresponde al modo "Automático", cuando se hace clic sobre "Manual" se desactiva todo exceptuando el botón salir y los mismos botones radiales "Manual" y "Automático"; en cambio se activan 8 cajas de opción múltiple para controlar (valga la redundancia) manualmente las salidas, donde ☐ indica apagado mientras que ☑ indica prendido. Si el usuario opta por regresar a la pantalla de acceso el programa regresa el sistema a modo automático por cuestiones de seguridad.

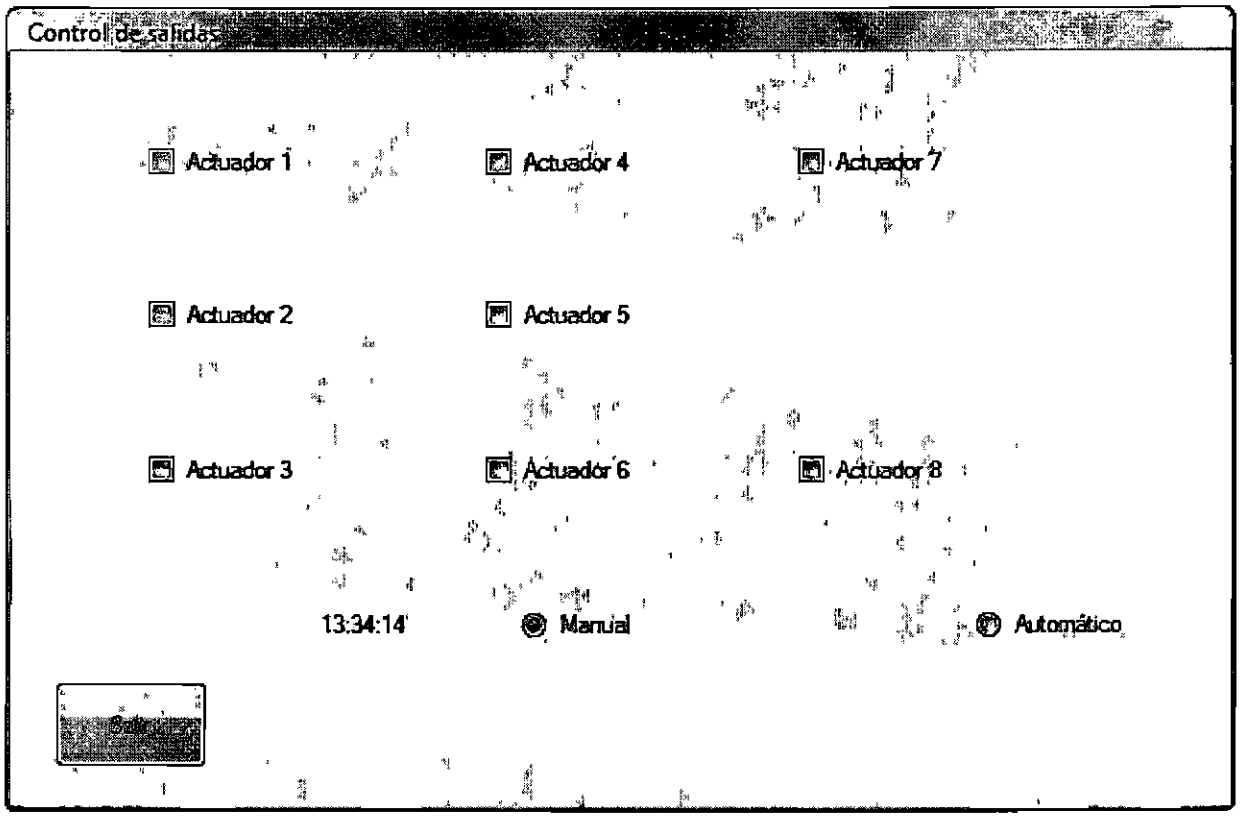


Figura 12. Transformación de ventana a operación manual

Como una mejora al final se implemento la ayuda instantánea para proveer al usuario de información breve y concisa que le ayude en la manipulación del sistema; este sistema es por si solo entonces auto documentado. Los globos de ayuda emergen al dejar el puntero mas de 2 segundos sobre el control, en esta primera versión su inclusión no denota bastamente su potencial, pero en la fase del control difuso el usuario se enfrentara con parámetros menos usuales que los que se pudieren encontrar en una tabla de horarios, es ahí cuando la inclusión de la ayuda instantánea cobra verdadero sentido.

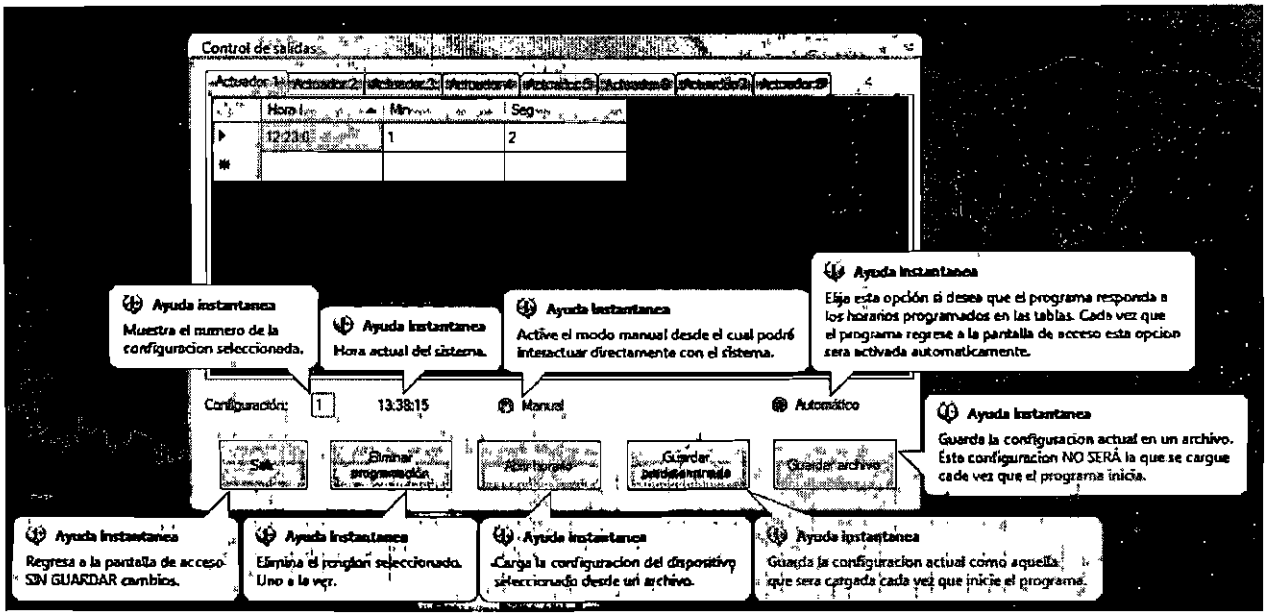


Figura 13. Integración de mejoras en accesibilidad e interfaceado

Una descripción más adentrada en el funcionamiento específico del programa hace primordial la descripción de la inicialización del puerto paralelo con la librería de distribución independiente como se menciono en un principio.

```
public class PortAccess
{
    [DllImport("C:\\inpout32.dll", EntryPoint = "Out32")]
    public static extern void Output(int adress, int value);

    [DllImport("C:\\inpout32.dll", EntryPoint = "Inp32")]
    public static extern byte Input(int adress);
}
```

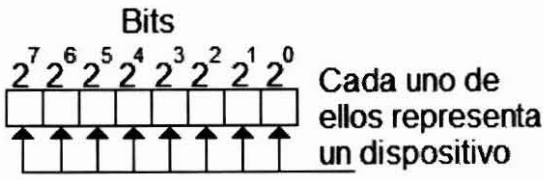
La clase descrita arriba es lo que se utiliza para externar los datos a traves del puerto paralelo, como se muestra es la clase PortAccess con 2 métodos; PortAccess.Output (dirección, valor) y PortAccess.Input (dirección), la primera no regresa ningún dato (void) y tiene como parámetros la dirección del puerto y el valor a escribir (entre 0 – 255), en el caso de Input se especifica la dirección y esta regresa el valor leído como una variable byte.

Ya cubierta la parte de comunicación, la etapa de potencia trabaja en lógica negada esto es, si se pretende apagar el dispositivo en cuestión un '1' es

esperado y viceversa si lo que se desea es activarlo se necesita un '0'. Para esto se crearon 2 funciones dentro de la clase principal:

```
public void pp_of(byte pin)
public void pp_on(bool timer, byte pin)
```

La funcion pp_of pone en alto el bit especificado en la lista de parámetros (en este caso 'pin'); por tanto pp_of(5) pone la posición 2^5 a 1, mientras pp_on(5) lo pone a 0.



Cuadro 5. Significancia de bits en un octeto

El puerto paralelo por su parte esta constituido, como ya se ha explicado, de una parte de datos que consta de 8 bits ($D^7 - D^0$) que están directamente relacionados con ($2^7 - 2^0$).

A la par de esta versión se diseño una versión que incorpora un teclado e pantalla para los sistemas con pantalla táctil (Touch-Screen), esta versión tiene la misma funcionalidad que el programa descrito ulteriormente con la única diferencia de que en la computadora que este se instale no es necesario considerar el uso de un teclado y ratón, estando el ratón ahora comandado a través de la pantalla táctil y el teclado emulado vía software así mismo con interacción vía pantalla. Para el ratón es necesario calibrar la pantalla por el usuario final a través del asistente proporcionado por el fabricante.

El teclado a pesar de ser emulado su objetivo es desempeñar las mismas funciones que un teclado normal. En el caso de soporte de caracteres especiales y soporte internacional se ha limitado un poco; así estando disponible únicamente las letras del alfabeto español (a-z) incluyendo por supuesto la letra ñ, los

números del 0 al 9, y como símbolos únicamente "." y ":". Este teclado tiene el objetivo de hacer la captura de valores requeridos por el sistema de control por tiempos, quedando excluso cualquier otra función o uso del mismo.

Entonces el teclado ha de activarse únicamente cada que el usuario ha de inquirir el uso de él; el programa ha de ser quien decida cuando es necesaria la interacción del usuario vía teclado es entonces cuando la condición se valida y se despliega el teclado virtual; en cuanto el usuario no requiera mas el uso del teclado este ha de ocultarse automáticamente.

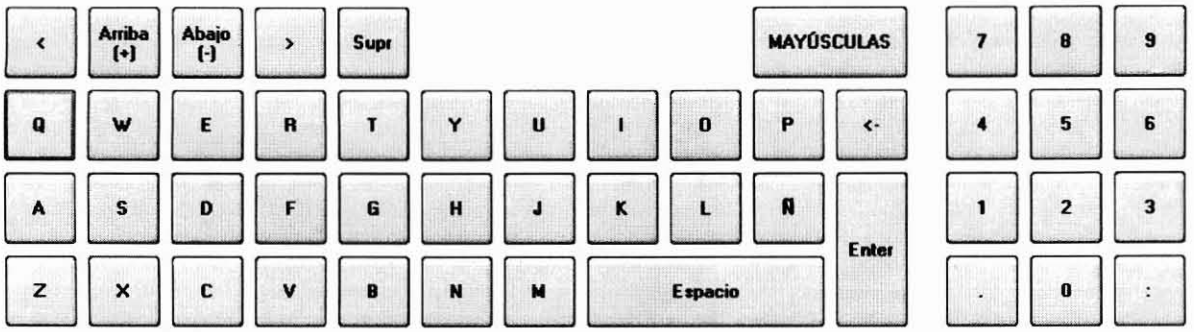


Figura 14. Teclado virtual para monitor táctil.

El teclado se muestra en la imagen de arriba, se procuro darle la misma organización que un teclado estándar en español esto para evitarle al usuario la memorización de nuevas plantillas de teclados de texto. Para digital alguna letra o numero vasta con presionar el botón correspondiente y el programa se encarga de imprimir el valor correspondiente. De igual manera que en un teclado físico, cuando la tecla de mayúsculas se presiona, esta queda engarzada hasta que se vuelve a presionar, mientras esta está activa las letras que se digiten son escritas en mayúsculas, y viceversa.

Es preciso indicar que el teclado se activara cada que el usuario requiera escribir o cambiar un valor en el programa, esto incluye cajas de usuario y contraseña, tabla de tiempos, botones, y cajas de valor numérico variable (up/down box).

Control Fuzzy

Para el caso del control fuzzy en su integración final con su antecesor control automático por tiempos queda de la siguiente manera para la pantalla principal:

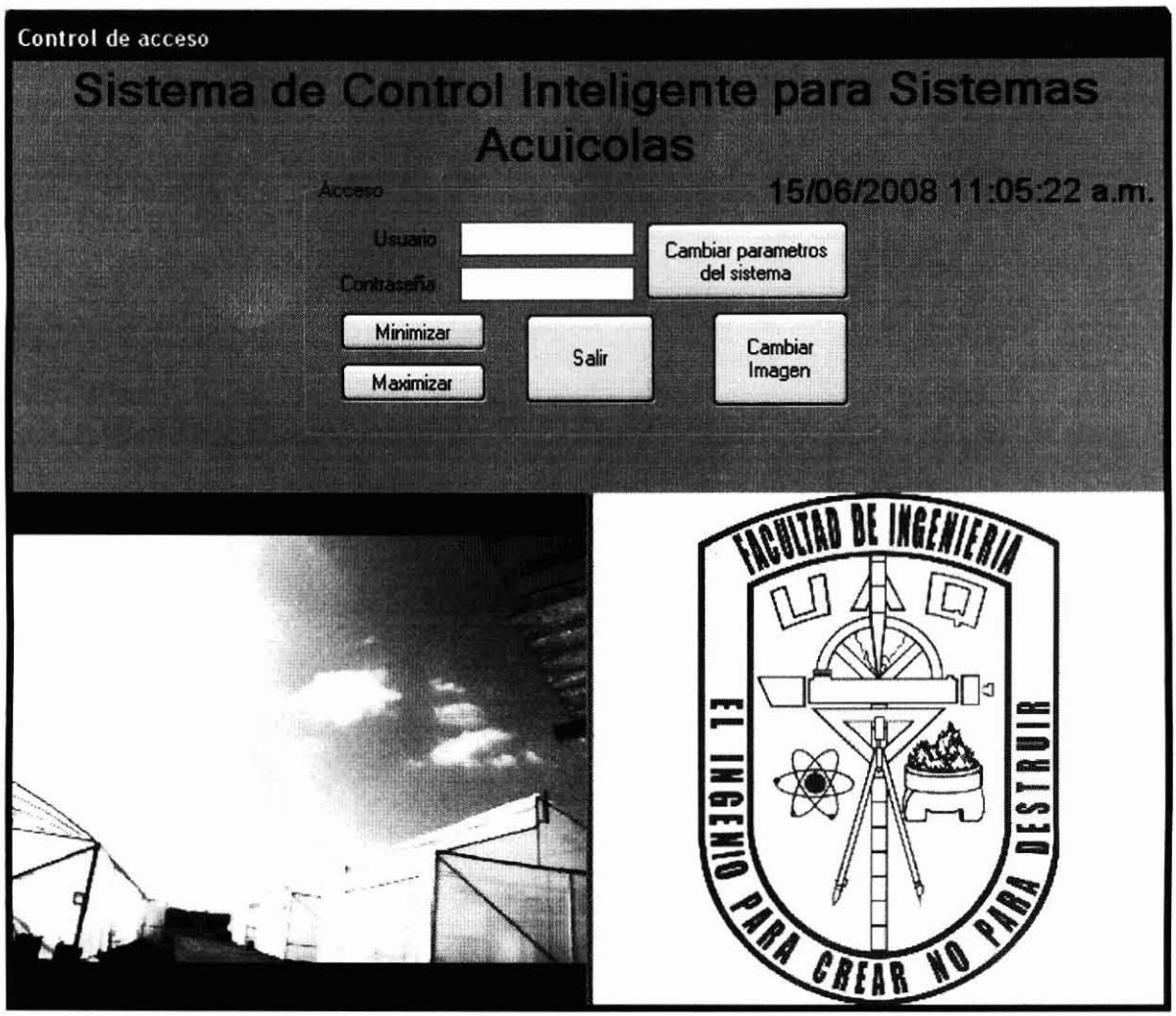


Figura 15. Pantalla principal de tercera versión con retroalimentación de usuarios implementada.

En donde se ha hecho uno ligero rediseño a la manera en que se organizan los controles para una mejor accesibilidad esto debido a que el diseño esta centrado en el usuario por lo tanto se han recibido y aplicado las sugerencias emitidas por los mismos.

En tanto al control manual es elementalmente el mismo panorama, el funcionamiento es el mismo al descrito para el sistema de control automático por tiempos, es por ello que al menos para estas pantallas se omite la descripción.

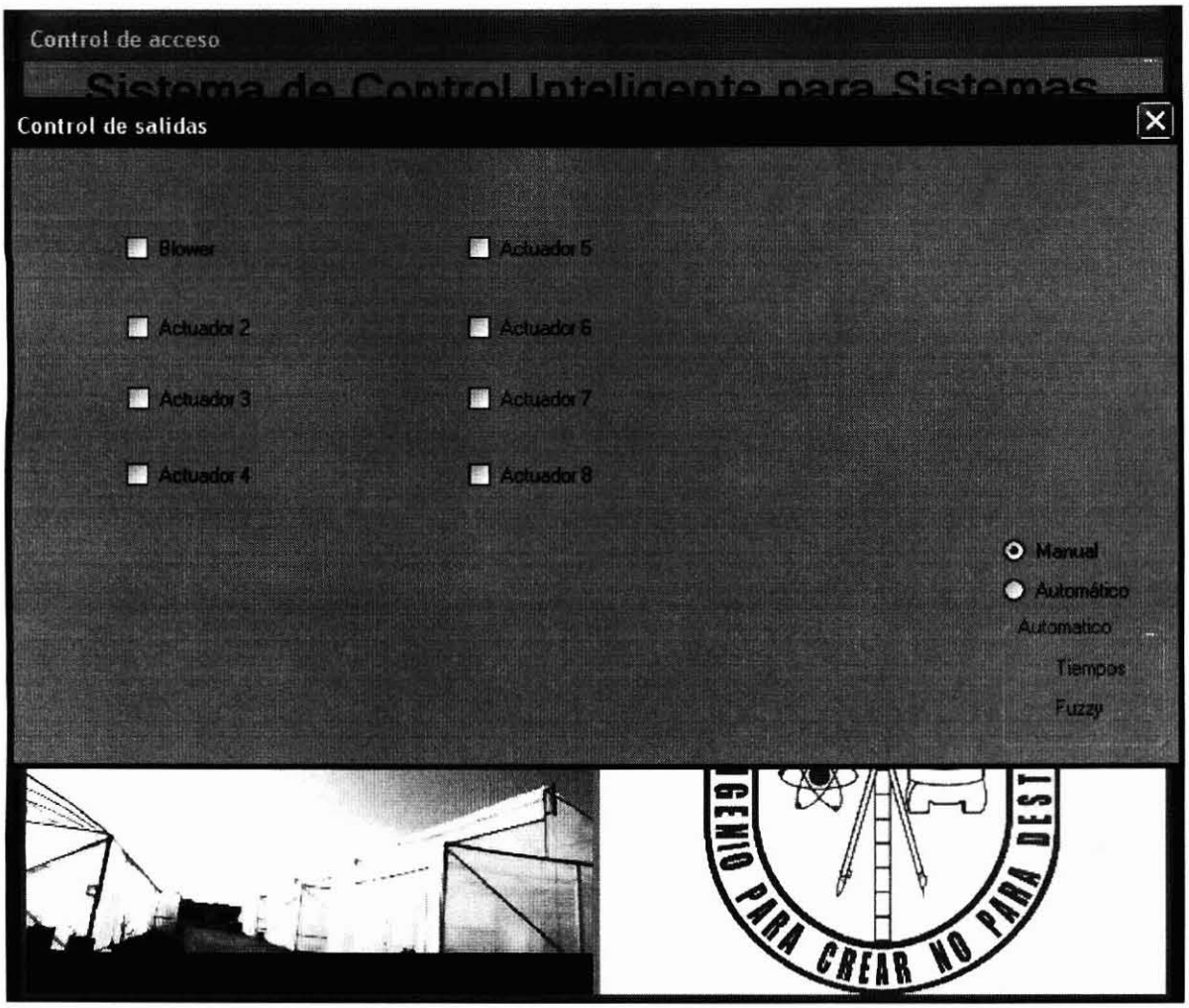


Figura 16. Transformación modo manual en versión Fuzzy.

Es entonces para la pantalla principal en donde se presentan nuevos controles que adaptan el contenido al nuevo esquema, estos son un bloque de selección por tiempos o Fuzzy, un bloque de calculo automático de parámetros para monitorización de la salida, y los botones graficar en línea, conjuntos difusos, reglas, parámetros iníciales y calcular; cuya dinámica de funcionamiento se describe a continuación de manera independiente.

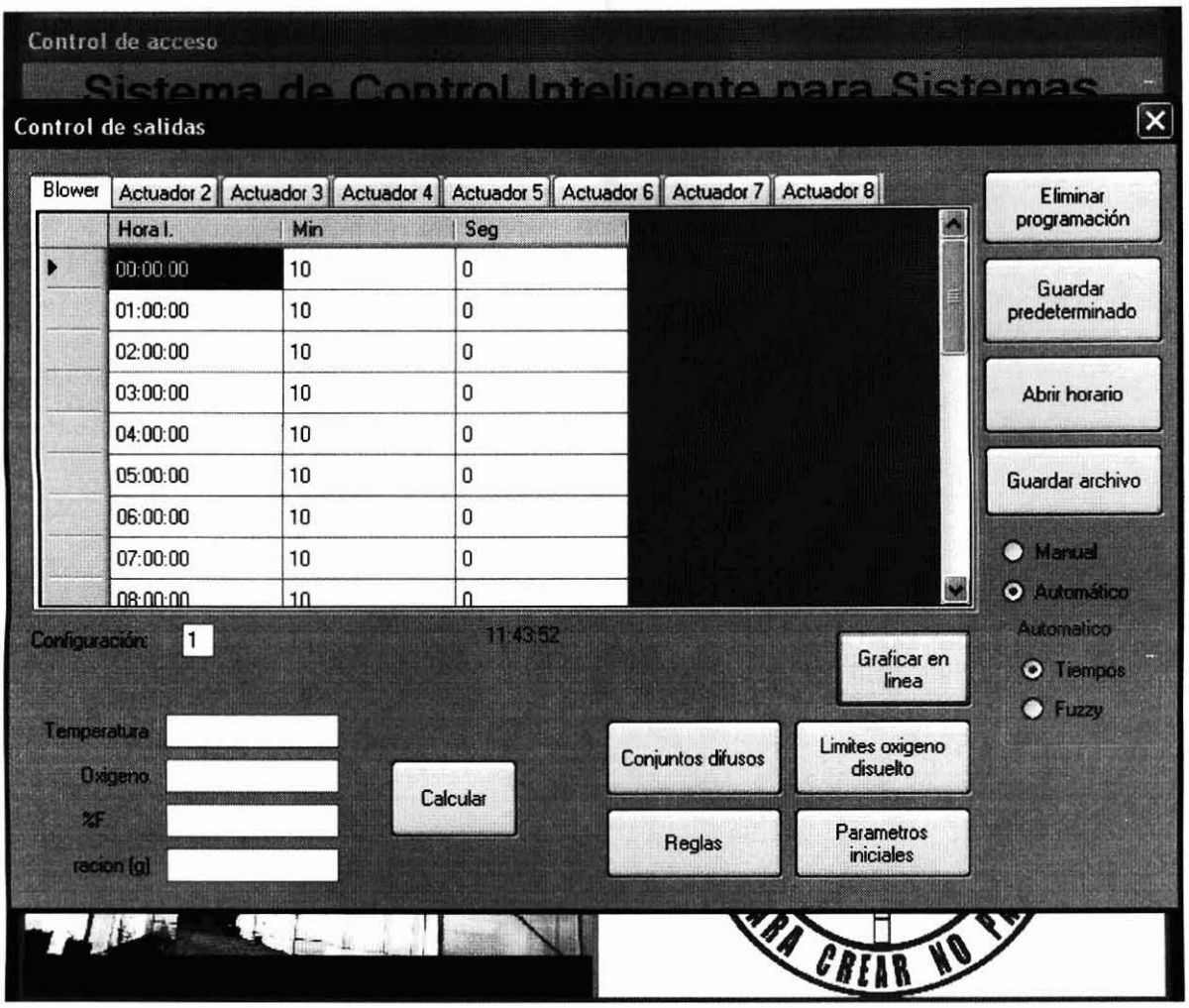


Figura 17. Ventana de configuración de horarios y funciones avanzadas.

El botón conjuntos difusos como su nombre lo indica permite al usuario iniciar la ventana donde estos se han de agregar, eliminar o bien modificar; estos conjuntos son los que delimitan la dinámica del controlador difuso, para esto en el formulario se hace notar la presencia de una tabla donde se despliega de manera ordenada ascendente los conjuntos actuales cargados desde un archivo de configuración guardado en el disco duro. Es también evidente la presencia de 2 controles UpDown que le proporcionan al usuario la posibilidad de explorar los conjuntos difusos de los distintos actuadores así como la de aumentar o disminuir el numero de estos en incrementos de 2 empezando por 1 (1,3,5,...).

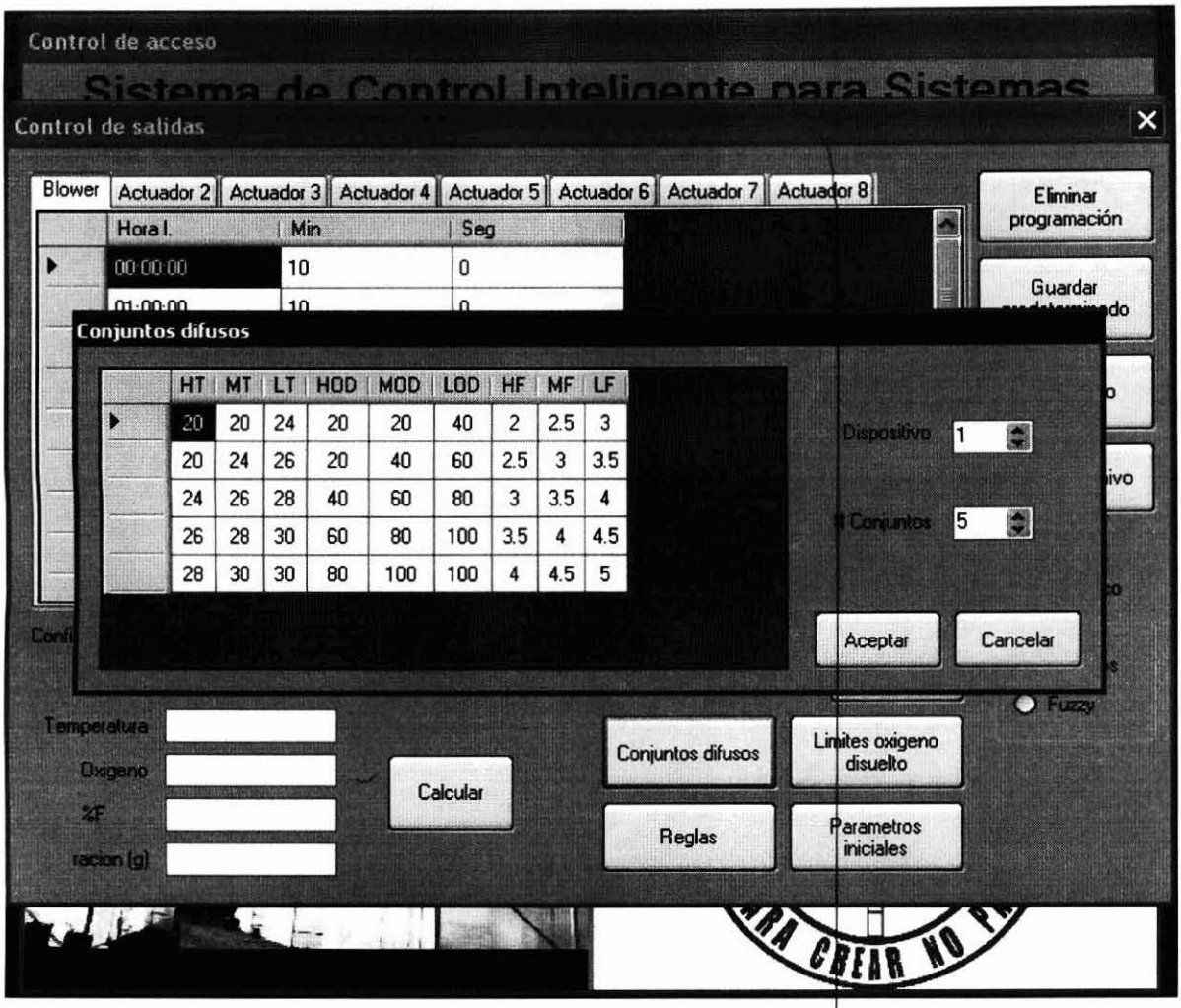


Figura 18. Ventana de conjuntos difusos, 1 conjunto por cada actuador (mejora de la versión 2 a la 3).

La tabla sigue un formato estricto donde los campos designados a la temperatura no puede tener un valor superior a 255 o inferior a 0 (que para casos prácticos es un rango bastante amplio) esta limitante implicada por el tipo de dato "byte", las primeras 3 columnas respectan a la temperatura para las cuales H indica alto, M medio y L bajo, respectivamente. Al oxigeno disuelto le corresponden las siguientes 3 columnas con los mismos significados para H, M y L, en donde los valores tienen el mismo rango con la limitante adicional de que por representar un porcentaje se recomienda establecerlo entre 0 y 100, en caso omiso de esta indicación el programa asignara un valor de 0 si el capturado es menor que 0 y 100 para cualquier numero mayor o igual a 101;, quedando

entonces las ultimas 3 para la biomasa donde cabe hacer la aclaración de que el valor es ahora un numero decimal positivo sin restricción alguna.

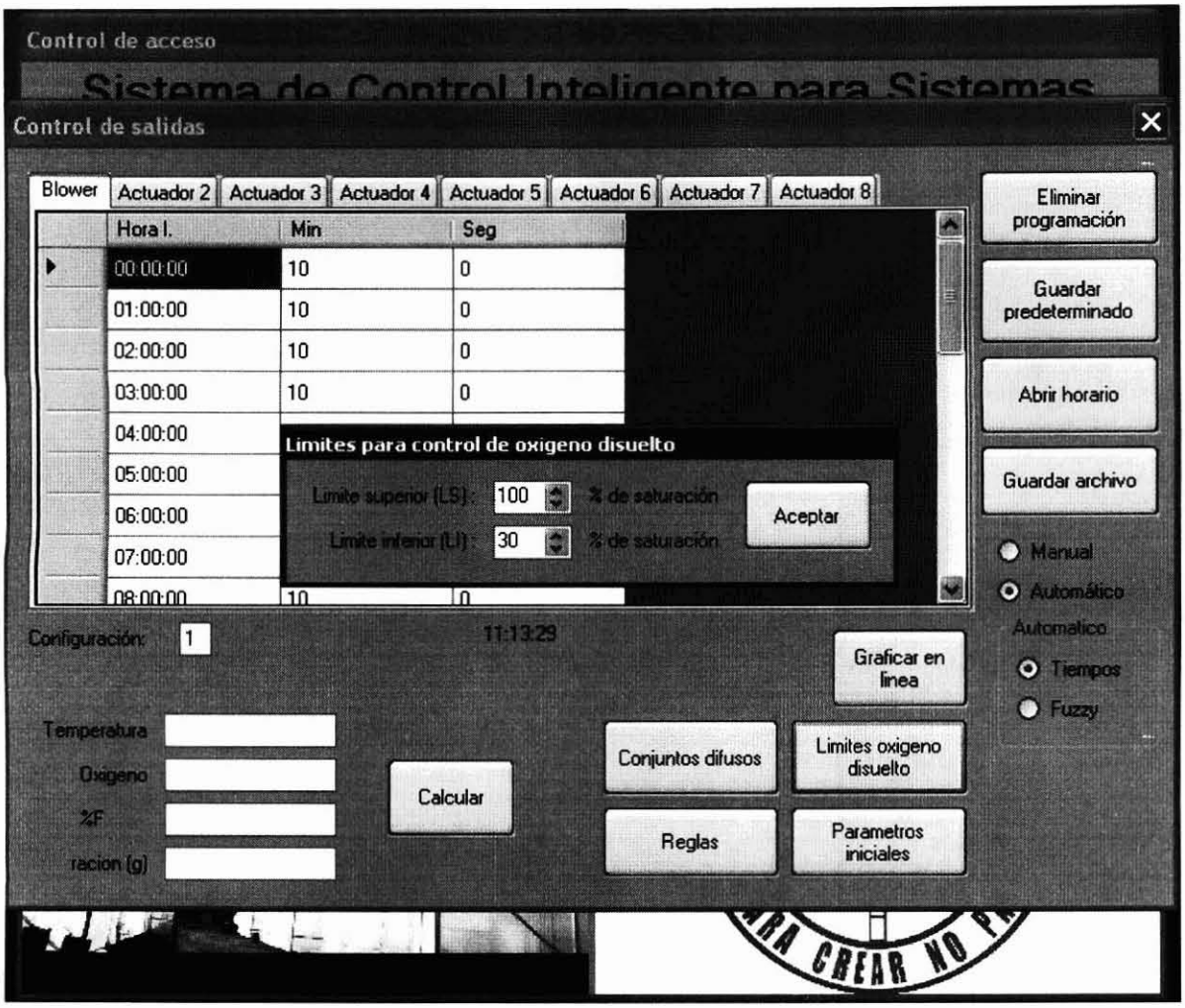


Figura 19. Dialogo para cambio de límites en oxigeno disuelto.

Los limites de oxigeno disuelto permiten al control difuso establecer los valores máximo y mínimo para controlar la oxigenación del agua que en cuanto a funcionamiento establece que si el nivel de oxigeno disuelto esta por debajo del limite inferior el control aplicara una acción correctiva (que en bien podría ser encender el aireador o activar la recirculación), en caso de que el limite superior sea alcanzado se podría de igual manera aplicar una acción correctiva o bien cancelar la acción correctiva aplicada en el caso del limite inferior. Como última observación se notifica la evicción de la condición en donde el límite inferior sea mayor al superior y viceversa.

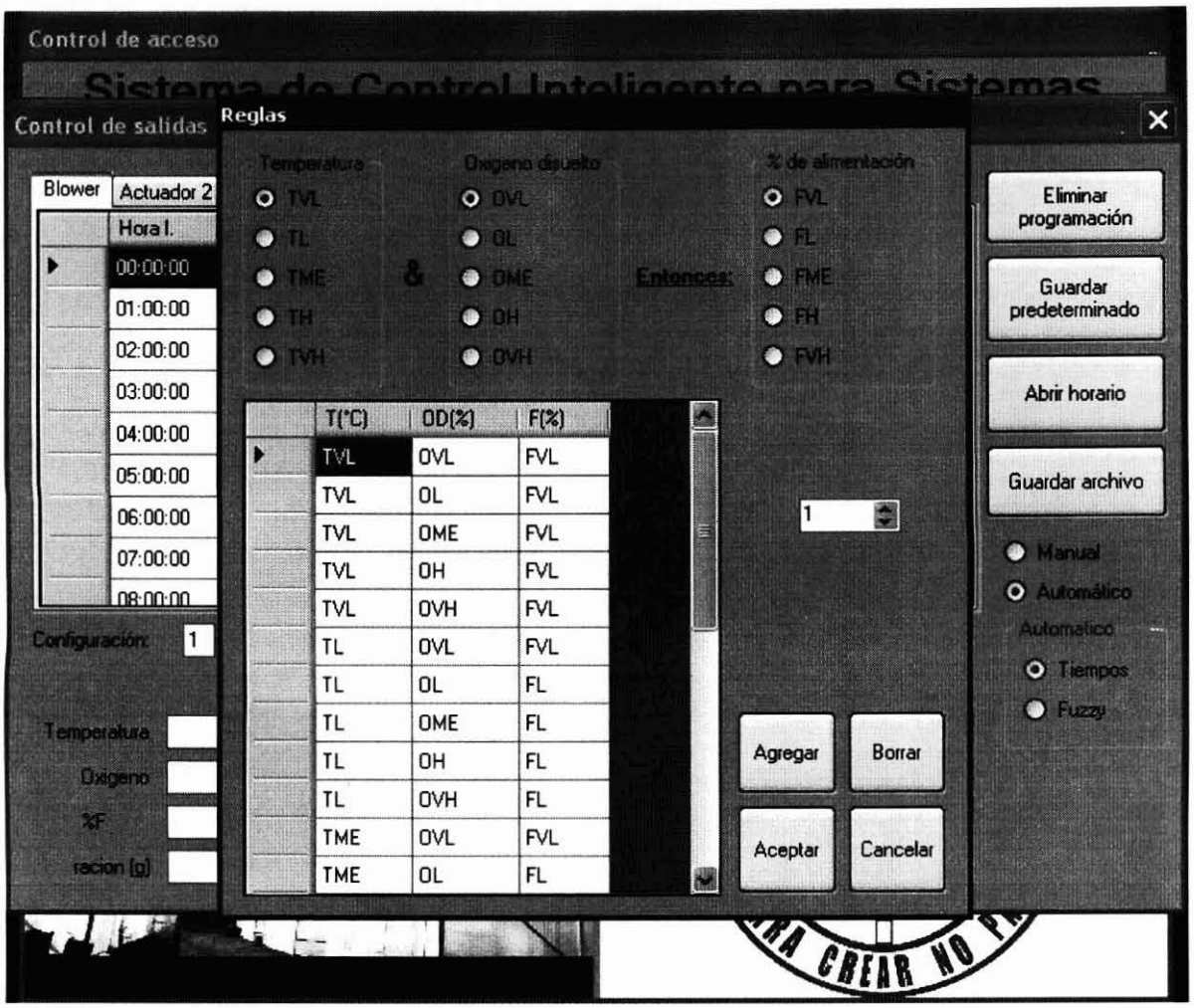


Figura 20. Ventana de conjuntos reglas, 1 por cada dispositivo (mejora de versión 2 a 3).

La ventana de reglas es entonces donde se ha de desplegar la lista de condiciones proporcionadas por el experto piscicultor donde con base en su conocimiento se generan condiciones si... y si... entonces..., estas condiciones son el núcleo de inteligencia proporcionada al control; un ejemplo de ello es "Si la temperatura es alta y el oxigeno disuelto es bajo entonces el porcentaje de alimentación es medio", para capturar cada una de las reglas es preciso hacer uso de los campos de opción múltiple (3 ilustrados arriba) cuyo significado para VL, L, ME, H y VH es muy alto, alto, medio, bajo y muy bajo, respectivamente. El control UpDown es utilizado para alternar entre las distintas configuraciones asignadas a cada actuador el botón borrar elimina la regla seleccionada, si se acciona el botón

aceptar se guardan los cambios y cierra la ventana, para el botón cerrar los cambios son pasados por alto y se conserva la configuración actual.

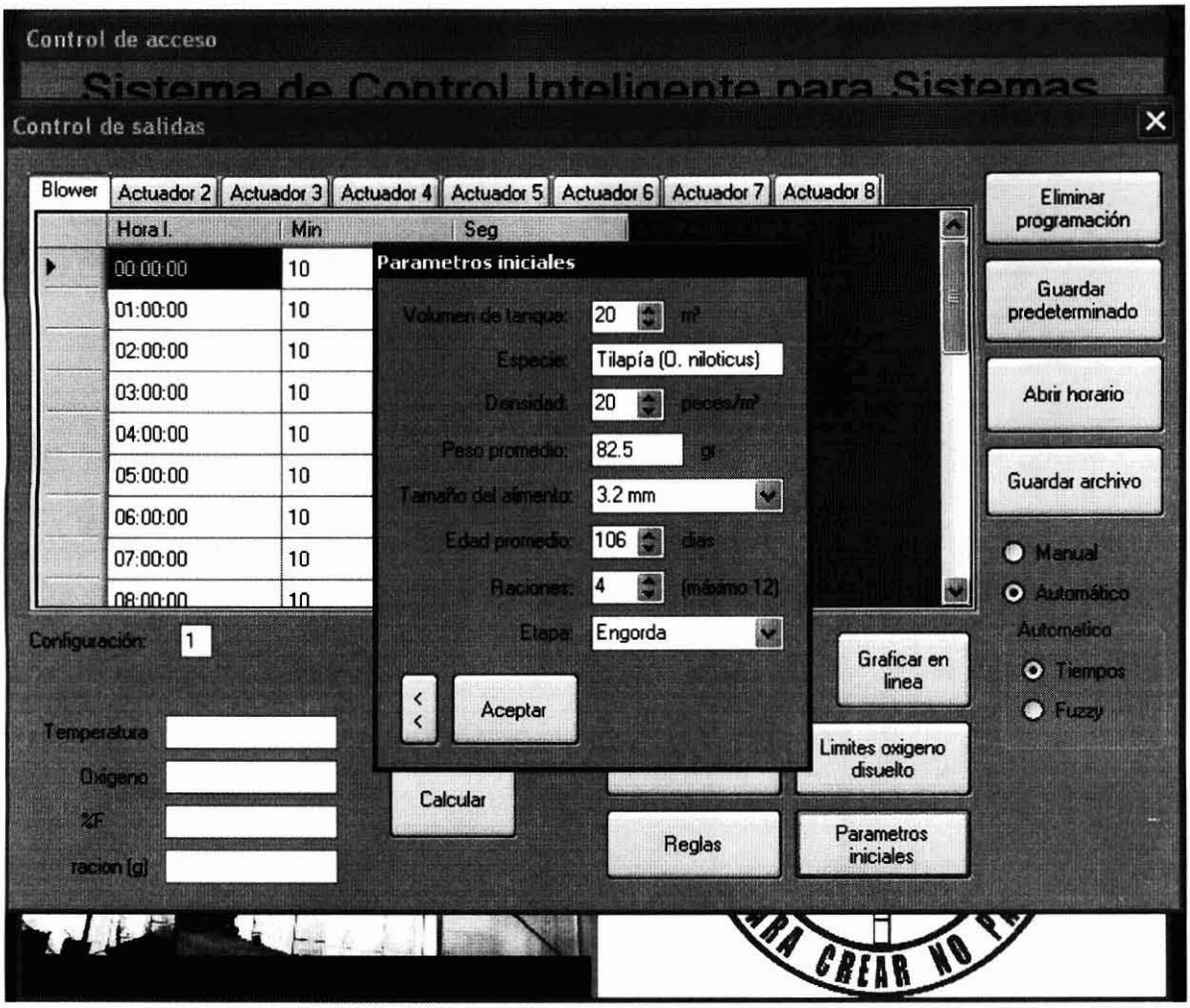


Figura 21. Dialogo de parámetros iníciales con interfaz mejorada de acuerdo a sugerencias recibidas por usuarios.

La ventana de parámetros iníciales almacena información relevante sobre el sistema, entre ello indica parámetros que permiten llevar un record del progreso en el crecimiento asi como la densidad del estanque. Esta información es estática, a excepción de la edad promedio cuyo valor se aumenta en 1 por día cumplido. Los campos tamaño del alimento y etapa están vinculados, es decir, para cada elemento de la lista tamaño del alimento existe uno correspondiente para etapa, y

además al seleccionar un elemento de una de las dos listas la otra lista adopta el valor correspondiente.

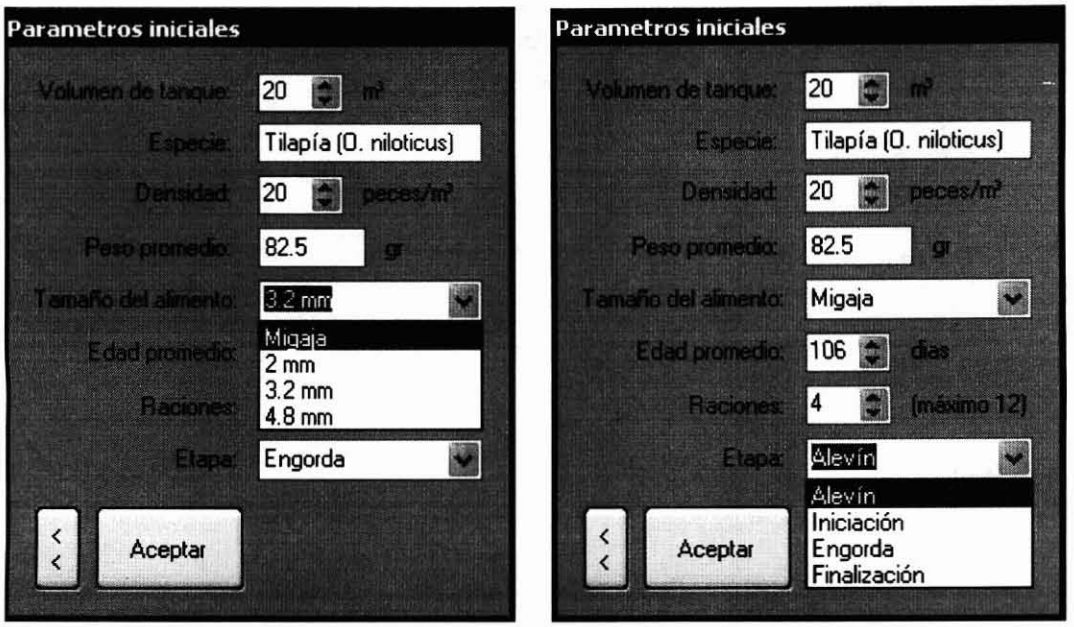


Figura 22. Ventanas de parámetros iníciales con interfaz mejorada por retroalimentación por parte de usuarios.

El modo graficar en línea despliega la interfaz de monitoreo y consulta de historiales; el monitoreo es la puesta en marcha de graficar la temperatura u oxigeno disuelto según escoja el usuario esta acción es comandada por el botón iniciar que llama los modulos XBee (comunicación inalámbrica por radio frecuencia) e incorpora una lectura del parámetro seleccionado (temperatura / oxigeno disuelto) para luego de 5 minutos calcular la media de los valores leidos y graficar el valor correspondiente en el plano escalado de la ventana, este plano escalado ha de adaptarse en su escala máxima al máximo valor leído.

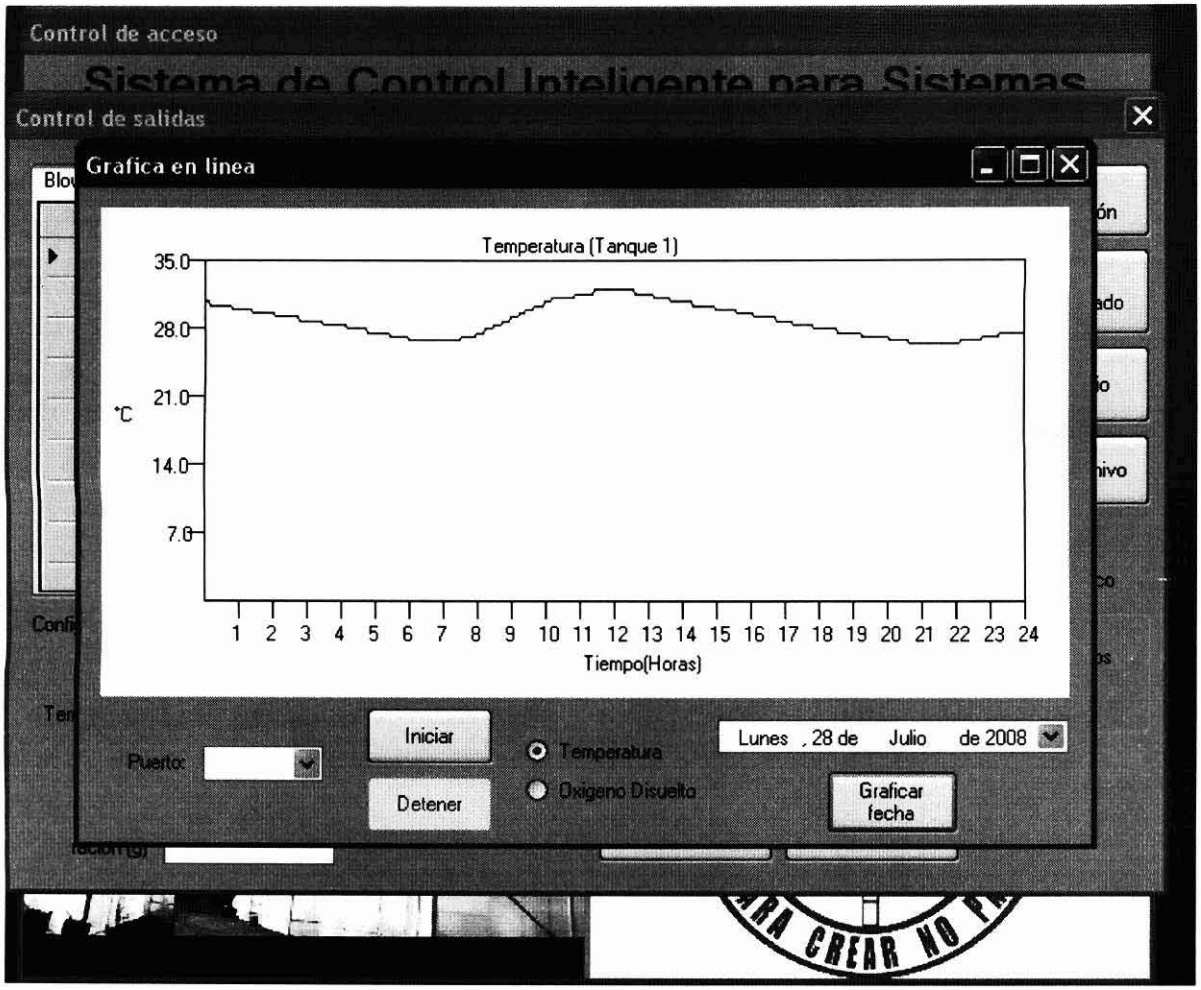


Figura 23. Grafica de temperatura en historial.

Etapas complementarias

Etapa electrónica

Es un hecho que el proyecto esta orientado al desarrollo de un programa para controlar dispositivos por tiempo, no obstante es necesario hacer la interacción directa con el sistema físico, para esto se utilizó una tarjeta de interfaz desarrollada por Soto-Zarazúa de la cual se presenta una serie de fotografías a continuación:

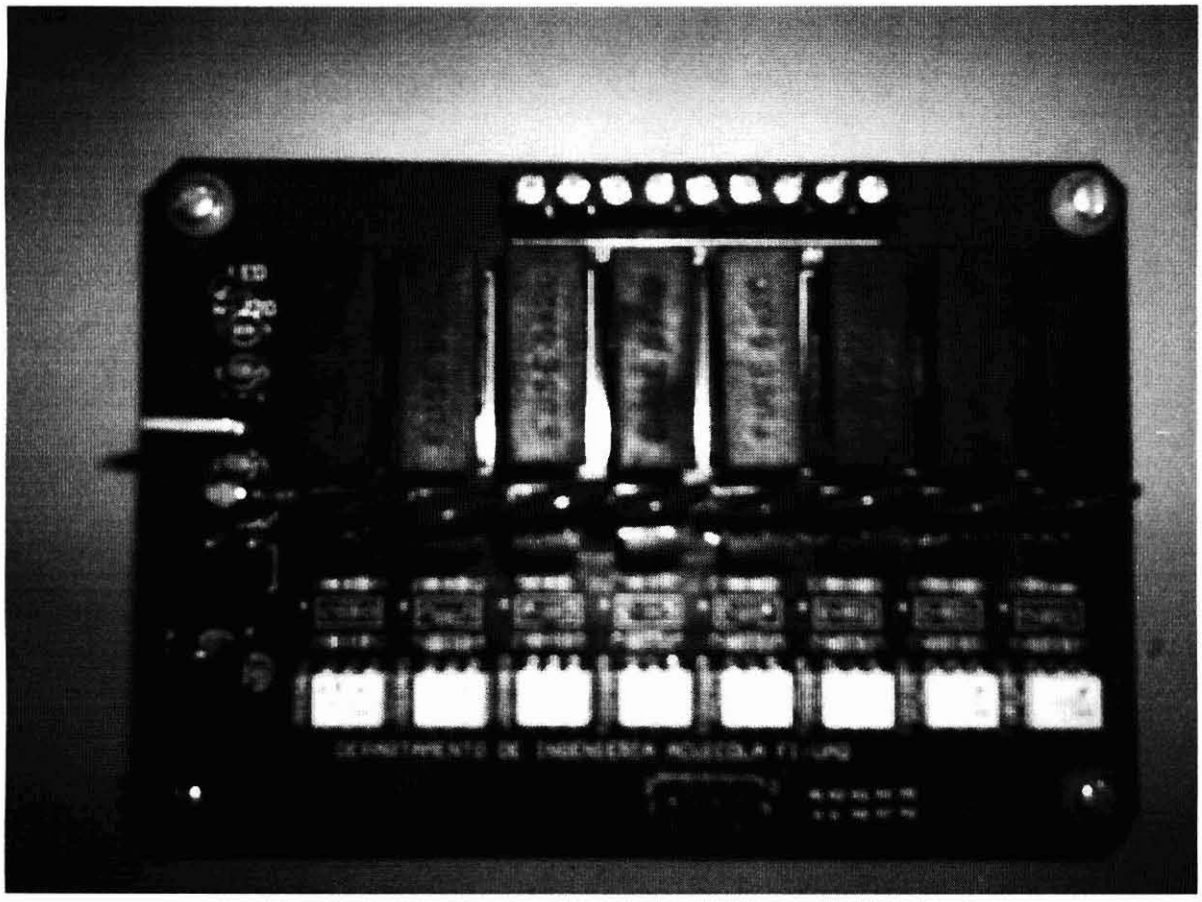


Figura 24. Tarjeta de interfaz desarrollada por Soto-Zarazúa.

Esta tarjeta permite controlar por medio de señales digitales de entrada con voltaje en nivel TTL, 8 interruptores con elementos de estado solidó para corriente alterna a 120 v. A pesar de no ser objeto de estudio de este proyecto, a grandes

rasgos la tarjeta consta de una fase de optó acoplado, donde las señales digitales con niveles TTL se aíslan totalmente de la etapa de corriente alterna, así evitando retornos de corriente elevados causados principalmente por cargas inductivas. Seguido de ello la etapa de potencia consta de Triacs cuya función es básicamente conmutar una Terminal de las salidas de AC con el común conectado en la misma tira de terminales de tornillo. A la salida tiene también un fusible de protección. La etapa de regulación únicamente consta de un puente de diodos que permite así alimentación con alterna o directa, después se convierte a 5v con un regulador y se alimentan los componentes correspondientes.

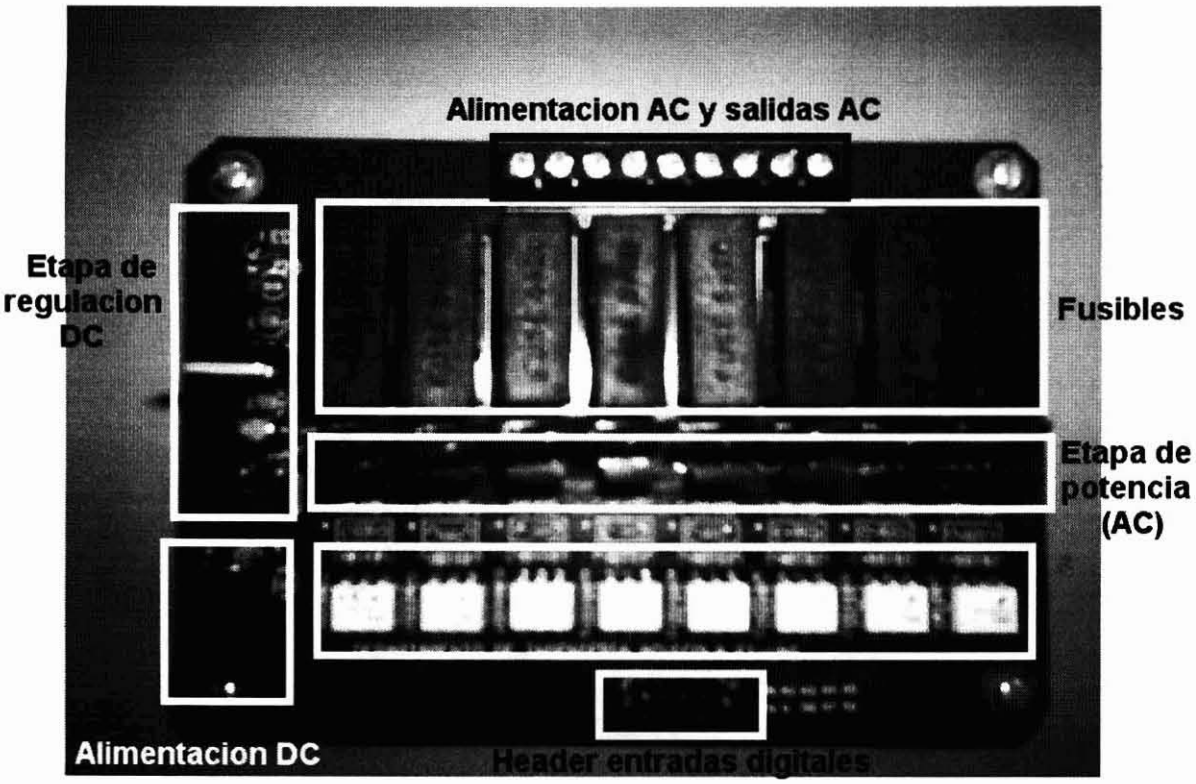


Figura 25. Distintas etapas, entradas y salidas que constituyen la tarjeta.

Únicamente como referencia a esta tarjeta se incluye una fotografía del PCB de la tarjeta; el diseño del PCB también es parte del trabajo desarrollado por Soto-Zarazúa.

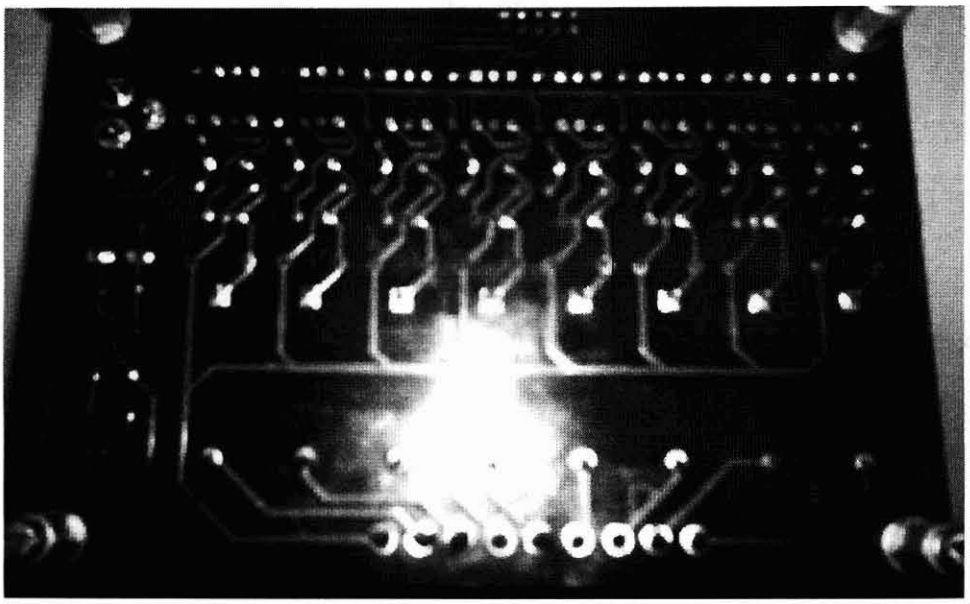


Figura 26. Fotografía de PCB parte trasera.

Etapa computacional

En el transcurso de la elaboración de este programa se utilizaron diversos elementos de prueba que serán descritos a continuación. El mas importante de todos es por supuesto la computadora donde se desarrollo, diseño e implemento el software en cuestión, por los mismos requerimientos para desarrollar el sistema se utilizo una maquina con capacidades medianas en procesamiento esta fue una computadora con un procesador Pentium 4 a 3 GhZ , con 512 MB en RAM, disco duro de 40 GB y un monitor UVGA a 1280x1024, el lenguaje de desarrollo utilizado como ya se menciono es el Microsoft Visual Studio Express 2008, se utilizo también una cámara LifeCam VX-6000 3.0 Mpix para la interfaz video en vivo, una memoria flash 8 GB y un monitor Táctil VGA (1024x768).

La cámara utilizada por supuesto comprada, tiene una resolución de 3 Píxeles, ajuste de foco manual, micrófono integrado (no utilizado para este proyecto), y conexión USB, se incluye una fotografía a continuación:

Figura 27. Microsoft LiveCam VX-6000.

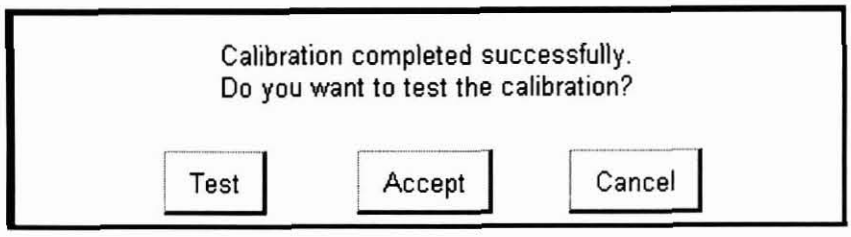
La pantalla utilizada en este sistema es un monitor de cristal liquido con membrana táctil (Touch-Screen), resolución súper VGA (1024x768), bocinas integradas. La siguiente fotografía corresponde al monitor:

Figura 28. Monitor SVGA Micro Touch BM EnTec, Touch-Screen.

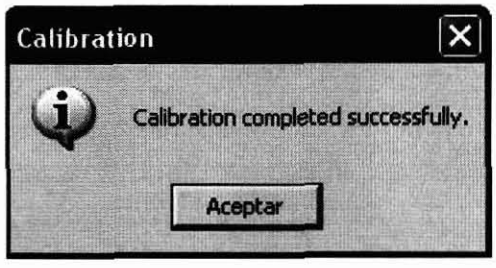
Para el correcto funcionamiento del sensor táctil es necesario realizar una calibración al menos cada año y así evitar el funcionamiento erróneo del sensor. Las pantallas de calibración son las mostradas a continuación, el tamaño de las imágenes ha sido modificado para su correcta adaptación en este texto:

Touch each target for several seconds and then liftoff.

Pantalla donde se pide al usuario señale distintos puntos de la pantalla



Al finalizar la calibración se pueden aceptar los cambios, probar los resultados u omitir los cambios.



Final de la calibración.

Etapa de potencia

La tarjeta de interfaz ya descrita forma parte de un sistema completo el cual fue albergado en un tablero que se ilustra en la siguiente imagen, en este tablero es donde se tienen contenidos el relevador, el contactor, la protección térmica, los interruptores principales del sistema y la tarjeta de interfaz. La función que este tablero desempeña es el control por tiempos de un aireador (Blower) para la crianza de peces, este tablero se acoadyuda de la tarjeta de interfaz, el programa por tiempos, y el monitor táctil para controlar la aireación automatizada preprogramada de los estanques de crianza.

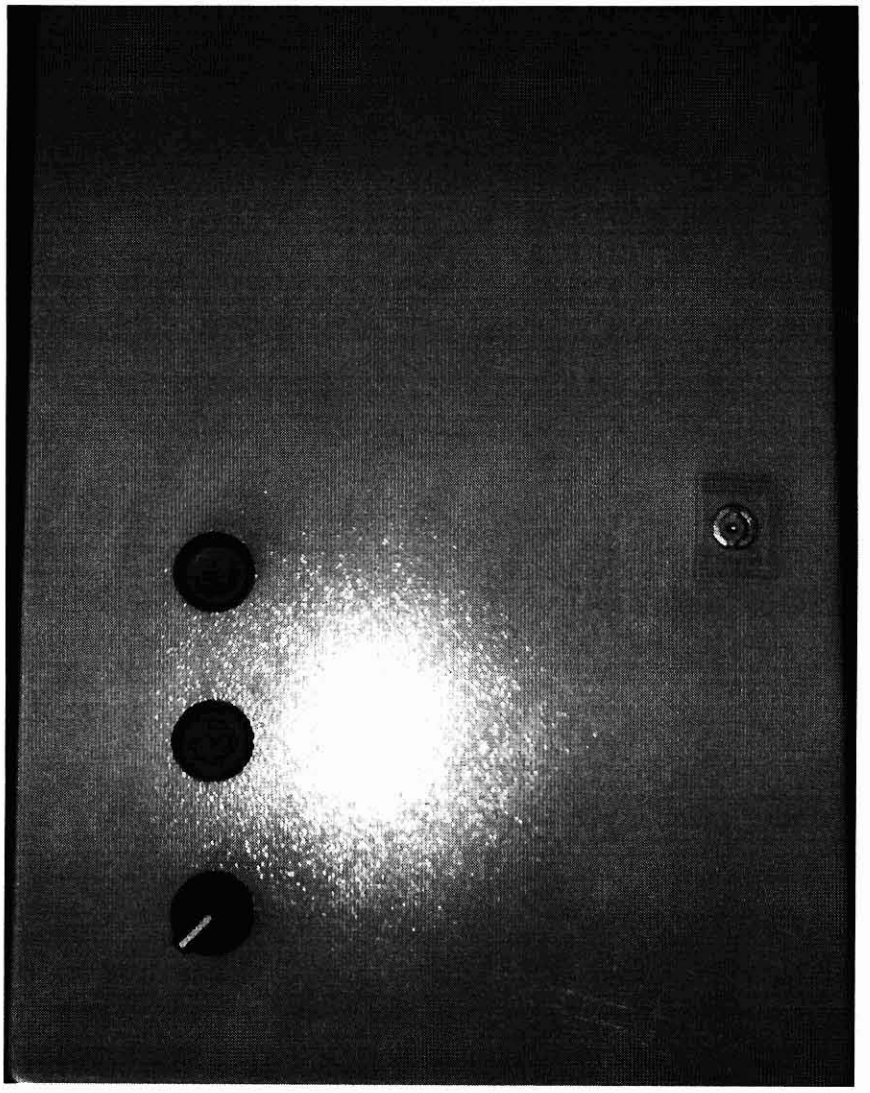


Figura 29. Tablero de potencia, incluso selector de modo apagado, manual y automático; con botones de arranque y paro en modo manual.

El aireador utilizado en el sistema cuenta con las siguientes características:

Compresor de anillo

Marca: Fuji Electric

Modelo VFC500A-7W

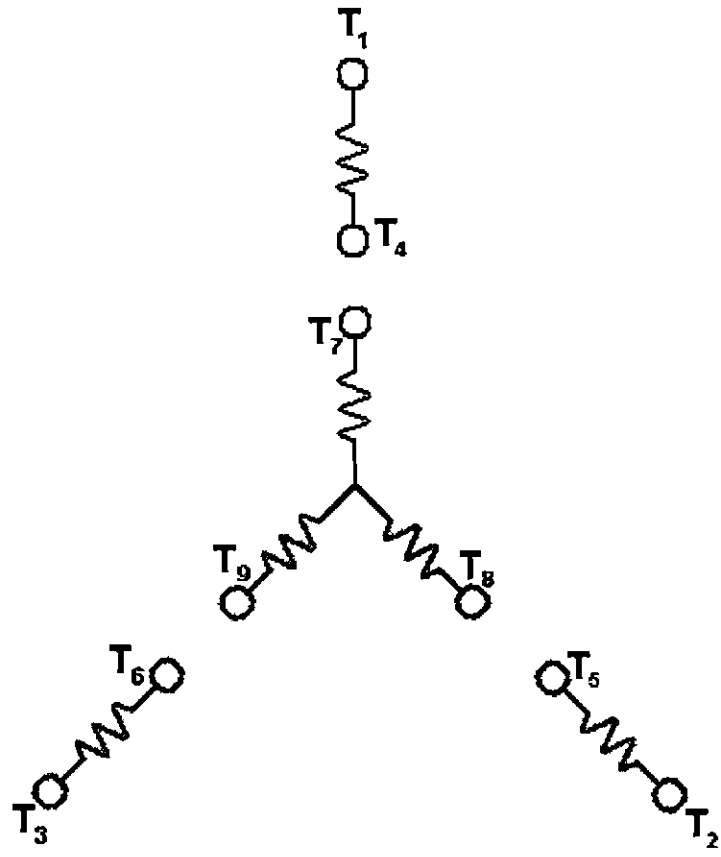
3 Fases

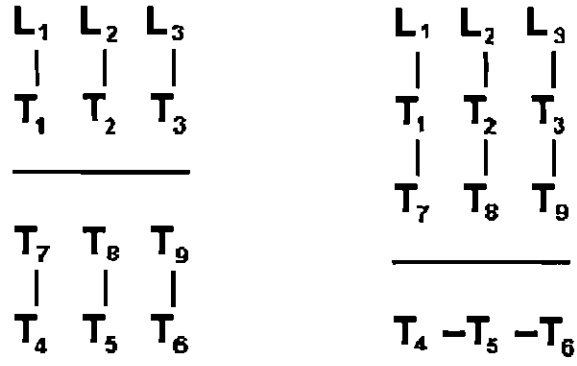
2 Polos

Datos de placa:

Voltaje	200-230	460
HP de salida	2.5	2.5
Máxima corriente	6.9-6.2	3.1
Máxima H2O	80	80
Máximo CFM	154	154

Diagramas de conexión





Alto Voltaje Bajo Voltaje

Internamente el tablero controla el aireador por medio de un contactor, este contactor tiene una bobina que al ser energizada conecta el motor a la alimentación trifásica de 220V el diagrama a bloques queda por tanto de la siguiente manera:

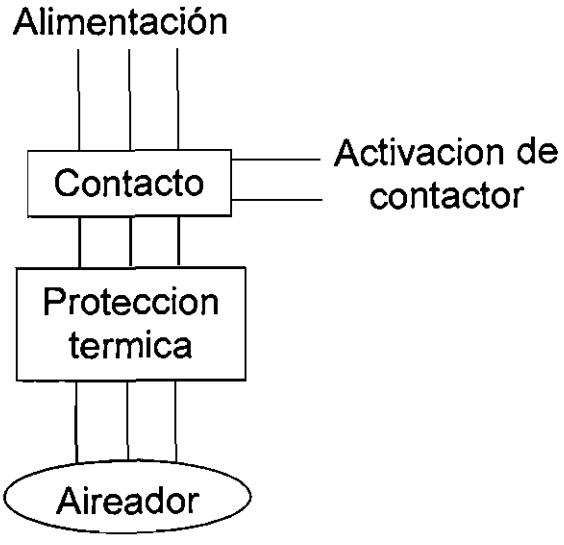
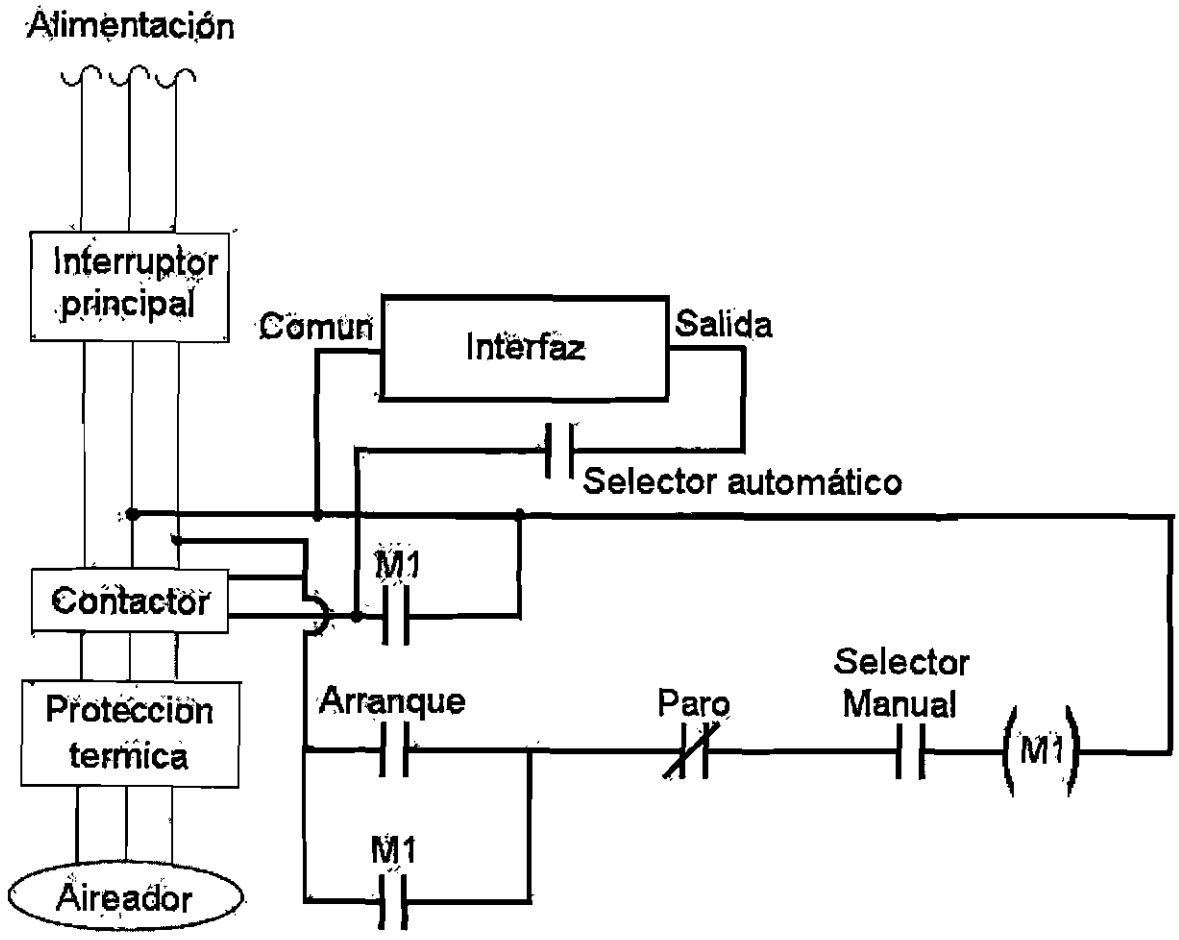


Figura 30. Diagrama a bloques de arrancador para aireador.

Esta activación es controlada entonces por el selector para ser conectada a la salida de la tarjeta de interfaz en modo automático o bien al botón de arranque para operación manual. Cuando se ha de optar por modo manual la activación es controlada por un relevador donde uno de los cables de activación se conecta directamente a 110V y la otra se conecta al otro hilo de alterna a través del relevador.

Control etapa de potencia



El diagrama arriba mostrado es lo que internamente constituye el diagrama de control para el tablero de potencia. Hágase notar el uso de la tarjeta de interfaz como parte del la interacción del sistema de control comandado por la computadora.

Básicamente el diagrama esta conectado para tener 3 modos de operación, Apagado, Manual, y Automático; así pues teniendo el selector 3 posiciones para cada uno de los modos de operación. En modo apagado el tablero de control aunque alimentado no tiene ninguna operación ante cualquier entrada, en el modo manual el sistema responde única y exclusivamente a los botones arranque y paro, botones verde y rojo respectivamente, en el modo automático existen 3 submodos

operacionales todos ellos comandados desde el computador interfceado con el sistema de potencia.

Los 3 modos en los que este puede operar son Automático por tiempos, Control automático Fuzzy (monitoreado oxigeno disuelto y temperatura para establecer la salida correspondiente), y modo manual comandado desde la computadora; lo que realmente se resume a un modo de prueba para la corroboración del correcto conexionado en el interfaceado.

En la operación manual vía tablero de potencia, al presionar el botón de arranque se activa la bobina M1 así quedando enclavada por el arreglo utilizado, esto siempre y cuando el modo manual sea preseleccionado con el seleccionador, al presionar el botón de paro se desenergiza la bobina y por tanto se interrumpe el enclavamiento. Para los modos comandados desde el computador se hace presente el uso de la tarjeta de interfaz una vez más únicamente cuando el selector de modo automático se presente activo. En estos modos como ya fue enunciado se hace uso de la tarjeta de interfaz con la cual desde el computador se enciende la salida por tiempo preprogramado, por tiempo de acuerdo a la respuesta del control Fuzzy principalmente influenciado por la temperatura y el oxigeno disuelto y el modo manual comandado desde el computador.

Para el testeo del programa se utilizo una tarjeta de potencia desarrollada por Soto-Zarazúa. La tarjeta de potencia fue probada con focos de corriente alterna 20W, motores de inducción 4IK25GK de Oriental Motors 25W / 100V 60Hz 1550 rpm. Se utilizo también cable calibre 14, y 18 AWG para la interconexión de la tarjeta; la alimentación utilizada fue 120V corriente alterna, así como un regulador de voltaje Nokia 9.0V corriente directa a 265 mA tipo FW1179.

Descripción de pruebas

Las pruebas realizadas fueron las siguientes se enumeran a continuación más adelante entonces dando una explicación detallada para explicar en que consiste la prueba:

- Pruebas de hardware
  - Prueba de funcionalidad
  - Prueba de destrucción.
  - Prueba de máxima carga.
  - Prueba compatibilidad.
- Pruebas de software
  - Prueba de repetibilidad.
  - Prueba de respuesta.
  - Prueba de compatibilidad.
  - Prueba de entradas no validas.
  - Prueba de confiabilidad.

Pruebas de hardware

Con respecto a las pruebas de hardware las pruebas fueron aplicadas a todos los dispositivos utilizados en el desarrollo de este proyecto. La intención de estas pruebas es conocer los límites reales del sistema y sus capacidades; así como corregir posibles errores de diseño.

La prueba de funcionalidad consta básicamente de corroborar de manera precisa que el sistema responda de manera coherente a las entradas actuales del sistema, para ello se necesita simular físicamente las entradas en el mismo y comprobar la respuesta correcta en el sistema. Para el caso específico del tablero de potencia en este sistema se puede describir el proceso seguido:

1. Conectar correctamente alimentación del sistema, siempre verificando el sentido de giro en dispositivos conectados a 3 hilos.
2. Simular físicamente condiciones eléctricas en las entradas de la bobina para el contactor.
3. De manera audible notar la reacción del contactor.
4. Interactuar presionando botones de inicio y paro manual para verificar correcta operación.
5. Cambiar la posición del selector y verificar que en modo automático única y exclusivamente responda a la excitación a través de la entrada digital; para el modo apagado, el sistema deberá de permanecer inmutable ante cualquier entrada y en el modo manual deberá atender a la comanda a través de los botones arranque y paro.

El enumerado para el procedimiento correspondiente la prueba de destrucción para el tablero de potencia se describe a continuación haciendo presente el uso de la tarjeta interfaz entre la computadora y el tablero:

1. Provocar condiciones adversas o de falla para corroborar el hecho de que la protección en las líneas de alimentación comete su propósito.
2. Someter el sistema a programaciones intermitentes, y repetitivas para evaluar el desempeño de los componentes bajo estas condiciones tomando nota de efectos negativos como sobrecalentamiento y retardo de respuesta.

Es importante remarcar que esta prueba en particular es aplicada tanto en el tablero de potencia como en la tarjeta de interfaz por separado y a su vez en la interacción de ambas. Lo que en resumidas cuentas supone 3 meses ininterrumpidos de trabajo constante de los sistemas. Entre las 3 etapas se utilizan 2 tableros de potencia y 2 tarjetas de interfaz. En la primera fase se testea un tablero por separado, en la siguiente una tarjeta por separado y en la tercera y ultima la interacción de ambas de igual manera durante un mes.

En la prueba de carga máxima se somete el sistema a condiciones límite para los componentes, llegando así al límite de corriente y voltaje para así entonces verificar la fidelidad de las características de los contactores, relevadores, SCRs y la misma suministración eléctrica. En esta prueba normalmente el sistema se deja trabajando de manera continua durante al menos 1 mes sin apagar para comprobar posibles fatigas en los componentes enumerados ulteriormente.

En tanto a la prueba de compatibilidad, consta de probar el sistema con distintas características de entradas para así determinar su capacidad para interactuar con sistemas distintos a los utilizados en el campo de pruebas. El sistema se conecta a distintos tipos y marcas de computadoras para corroborar su

completo y correcto funcionamiento. Una prueba mas completa constaría en alimentar el sistema con distintas características de suministro eléctrico como las que se pueden encontrar en este país (México).

Pruebas de software

Las pruebas de software verifican la correcta estructuración del código para evitar de esta manera que el sistema colapse, trabaje lento o no desempeñe las tareas esperadas. Estas pruebas de igual manera llevan al límite todo el sistema para garantizar que el sistema no fallará bajo condiciones y parámetros establecidos. En primera instancia los requerimientos mínimos para correr el programa son 256MB de memoria en RAM y un procesador de arriba de 70Mhz esto esta determinado por las librerías de .Net Framework 3.5 que son un conjunto de librerías utilizadas por el programa. Para conocer los parámetros y corroborar las condiciones se desempeñan las pruebas descritas a continuación:

- El programa se ejecutara por 1 mes continuamente así comprobando que no exista alguna condición adversa que habría de colapsar.
- Analizar el consumo de memoria por parte del programa.
- Analizar el consumo de procesador.
- Verificar la razón justificada del consumo tanto en procesador como en memoria RAM.

La prueba de repetibilidad esta asociada a la capacidad del programa para ejecutar los accionamientos día tras día con la misma duración y hora de inicio, esto hasta cierto punto es determinado por el programa si este es muy denso es probable que el programa tarde en tomar cuenta de los cambios en la hora actual debido a la carga del sistema. Con esta prueba lo que se pretende comprobar es que el sistema ha de responder en todo momento con exactitud y precisión tanto al momento de arrancar así como de respetar el tiempo programado.

En la prueba de respuesta lo que se verifica es la velocidad con la que el programa reacciona que hasta cierto punto tiene que ver un poco con la prueba anterior (la prueba de repetibilidad), pero en esta prueba solamente se evalúan los

aspectos correspondientes al tiempo que al programa le toma ejecutar lo que sea comandado.

Hoy en día es muy común encontrar una gran diversidad de computadoras de infinidad de marcas y modelos, muy aparte de la marca o modelo de las computadoras, estas varían entre distintas capacidades y configuraciones de hardware así como del software que en estas pueda estar alojado. Pensando en que los sistemas operativos que comúnmente se utiliza a nivel nacional son Microsoft Windows Xp/Vista *[en cualquiera de sus versiones]* el programa se desarrollo para que trabaje sobre estos sistemas operativos y a pesar de ello es necesario instalar las librerías necesarias (OCX LiveCam, .Net Framework 3.5, Drivers de LiveCam, controlador puerto paralelo) para su correcto funcionamiento. Una vez cubiertos estos requisitos se ha de verificar la configuración del hardware, principalmente los modos en los que el puerto paralelo puede trabajar SPP, EPP, ECP, o EPP+ECP. En resumen lo descrito es en lo que consiste una prueba de compatibilidad.

En la prueba de entradas no validas lo que se ha de verificar es que cada campo para la captura de datos por el usuario admita solamente los valores correctos, intentando así introducir valores incorrectos en los distintos campos que componen el programa.

El sistema por si se considera autónomo o que no necesita de la interacción de un usuario diariamente para su correcto funcionamiento, partiendo de este supuesto el programa deberá de brindar un funcionamiento correcto y continuo, para así no depender de verificar que funcione correctamente a cada instante. La prueba de confiabilidad consta de esto, de comprobar que el sistema es confiable y ha de desempeñar correctamente las tareas asignadas.

RESULTADOS

Resultados control por tiempos

Lo correspondiente al control por tiempos para las pruebas enunciadas en el capitulo anterior se han separado de acuerdo a la etapa correspondiente. Las etapas son entonces enunciadas a continuación enumerando para cada una de ellas los resultados correspondientes.

Resultados para pruebas de hardware

Resultados en la prueba de funcionalidad

La funcionalidad del sistema ha sido comprobada interactuando con la tarjeta de interfaz, así encendiendo cada una de las salidas en el modo manual y corroborando su correcto funcionamiento a la salida conectando el motor antes descrito (motor de inducción 4IK25GK de Oriental Motors 25W / 100V 60Hz 1550 rpm). Teniendo por tanto resultados exitosos logrando encender el motor en configuraciones por separado y también en coexistencia con motores similares conectados a la misma tarjeta.

Resultados prueba de destrucción

La prueba de destrucción fue desempeñada únicamente sobre la tarjeta de interfaz se desempeño por un lapso de 4 meses, sin problema alguno, de trabajo ininterrumpido, teniendo una temperatura en el regulador de voltaje de 31°C y de entre 25 y 28 °C para los demás componentes electrónicos; comparada la temperatura leída al final de la prueba con el rango operacional de temperatura proporcionada por el fabricante de entre 0 – 125 °C (para el regulador de voltaje) y para el resto de circuitos integrados y elementos pasivos de entre 10 – 55 °C el

resultado se consuma como exitoso al no tener sobre calentamiento o perdida de funcionamiento.

Prueba de máxima carga

Esta prueba fue realizada una sola vez utilizando una tarjeta distinta a la de la prueba de destrucción, en la cual se conectaron 8 motores de inducción 4IK25GK cada uno en una de las salidas de la tarjeta de interfaz teniendo como resultados una temperatura de 68 °C en el regulador de voltaje y de 34 °C en el resto de los componentes lo que muestra que el sistema es capaz de responder de manera adecuada a pesar de tener carga en todas sus salidas y una vez mas mostrándose dentro de los parámetros operacionales para temperatura.

Prueba compatibilidad

La tarjeta mostro restringida compatibilidad, siendo solo compatible con los voltajes manejados en el estándar SPP, con lo que se limita su uso en tecnologías equivalentes o similares para el puerto paralelo;

Resultados en pruebas de software

Resultados de repetibilidad.

El sistema fehacientemente cumple con las pruebas de repetibilidad propuestas con antelación siendo que a precisión de segundos activa y desactiva las salidas hacia la tarjeta electrónica de interfaz. Esto comprobado por corridas del programa guardando para ello un registro del momento en que se activo o desactivo cada dispositivo.

Resultados en respuesta.

El tiempo de ejecución para una computadora con los requisitos minimos es de 2 minutos, en una de mediana escala es de 20 segundos, y para computadores de capacidades elevadas oscila entre 2 y 20 segundos, esto siempre variable dependiendo de el numero de tareas y subprocesos en ejecución asi como la versión del programa, si esta incluye cámara web el tiempo incrementa a razón de la velocidad de respuesta del dispositivo en cuestión. La respuesta para interactuar con los dispositivos externos como es la tarjeta de interfaz es despreciable basándose en la correcta activación/desactivación con relación a la tabla de tiempos.

Resultados en compatibilidad.

En cuanto a la compatibilidad vía software el programa es compatible únicamente con los sistemas operativos enunciados a continuación: Windows 98, Windows ME, Windows XP, y Windows Vista. En particular para Windows 98/ME se requiere un parche de actualización para el Internet Explorer, y en el caso de XP se requiere el Service Pack 2 así como las correspondientes librerías del Net Framework 3.5 para todos los sistemas operativos. Aunado a ello se requiere un mínimo de 64 MB de memoria RAM y un procesador de más de 500MhZ.

Resultados para entradas no validas.

Ante entradas que no correspondiesen con las esperadas el control de excepciones hizo su primer y más significante aporte; siendo que al inicializar la captura de variables detecto correctamente las entradas de valores no validos en campos como la tabla de tiempos, limitando así la entrada de este campo a un numero entero sin signo de 8 bits (0 a 255). En el campo de hora de inicio no se

conto con la misma funcionalidad debido a que es de tipo texto quedando abierta la entrada a cualquier carácter digitable a través del teclado ya sea por tecla directa o combinación de las mismas, un caso particular registrado también en el control UpDown fue el que la entrada se combino para poderla hacer con la flechas arriba/abajo o bien con el valor del campo digitado por medio de los números 0 al 9, en el cual el control UpDown proporciona como una propiedad del mismo los limites superior e inferior.

Resultados prueba de confiabilidad.

Como ya se ha mencionado, el software y hardware ha sido sometido a trabajo continuo e ininterrumpido durante un periodo de pruebas de 3 meses, sumado a esto el sistema desde su creación a estado controlando 9 estanques con peces, el sistema por si ha registrado buenos resultados siendo que a mas de 7 meses de su puesta en marcha no ha requerido ningún ajuste o interaccion obligada por parte del usuario y/o programador. Con el sistema de registro de sucesos fue incluso posible seguir un historial de las acciones desempeñadas por parte del usuario y los registros de en que momento el programa activó la salida especificando para este caso si la activación fue ´por parte de una programación o por parte del usuario en modo manual.

Respuesta control Fuzzy

En general el algoritmo matematico que modela este sistema no ha sido modificado, además de que este análisis no forma parte de este trabajo. A continuación se muestran las graficas de los datos evaluados asi como las graficas de su respuesta ante estas entradas.

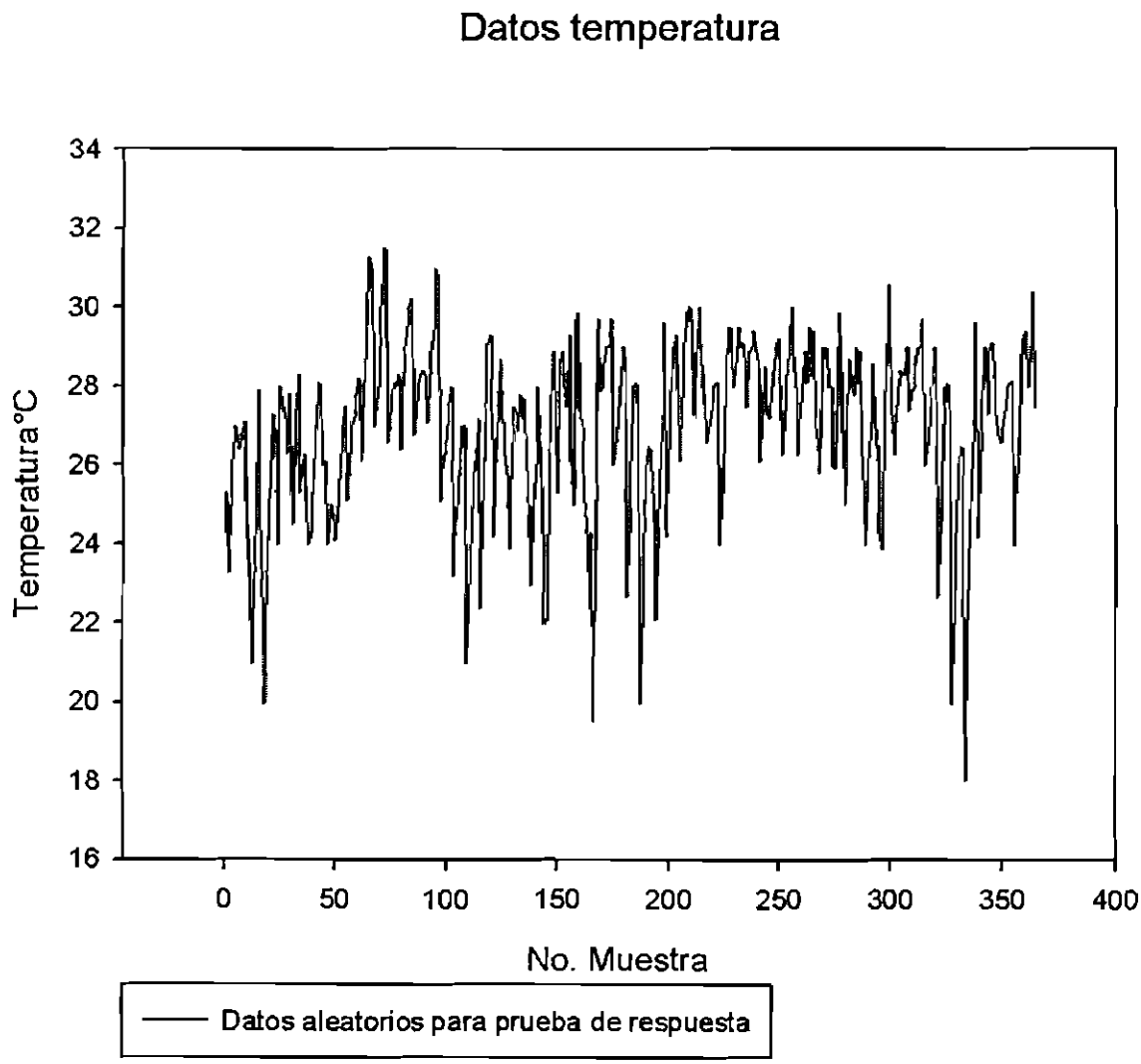


Figura 31.

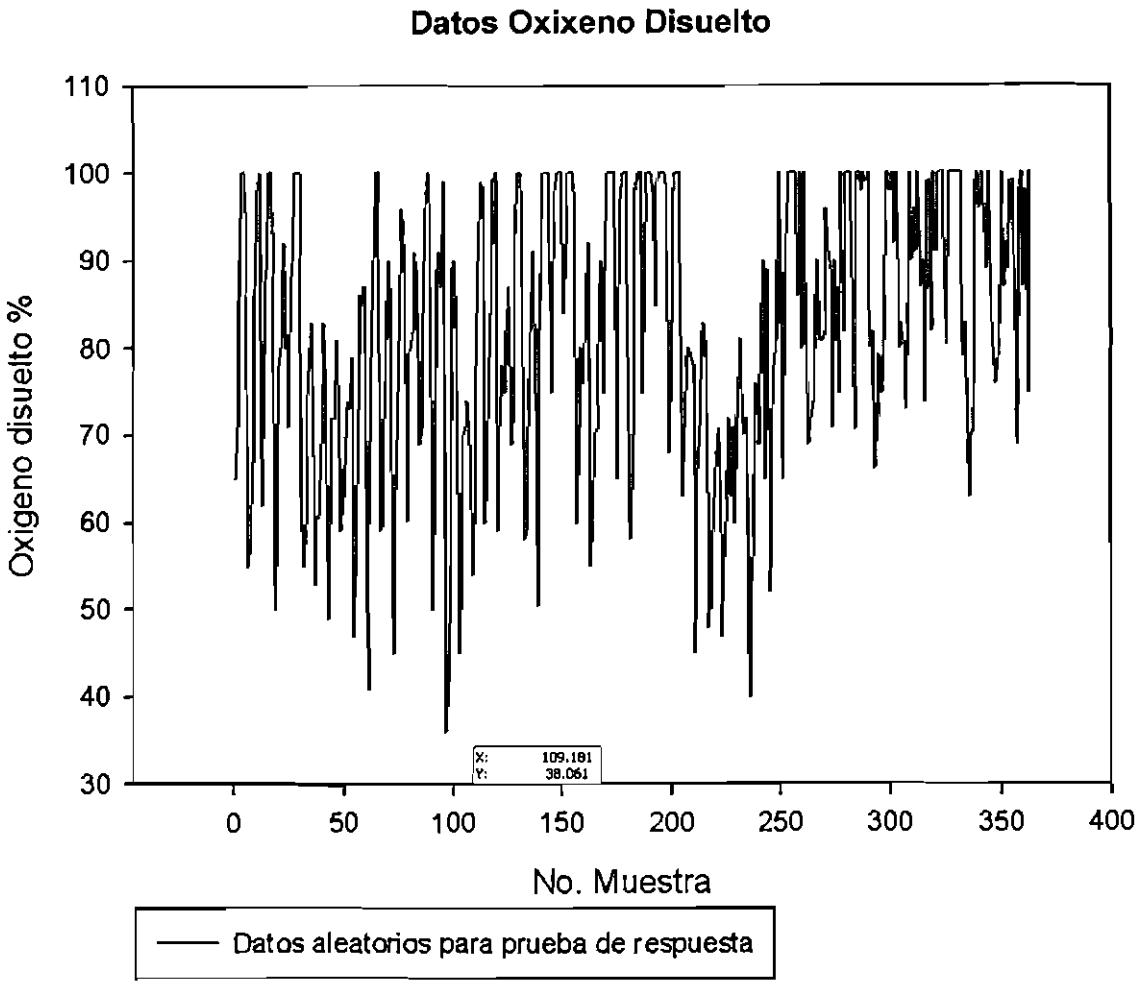

Datos Oxixeno Disuelto
Oxigeno disuelto %
110
100
90
80
70
60
50
40
30
0
50
100
150
200
250
300
350
400
X: 109.181
Y: 38.061
No. Muestra
Datos aleatorios para prueba de respuesta

Figura 32.

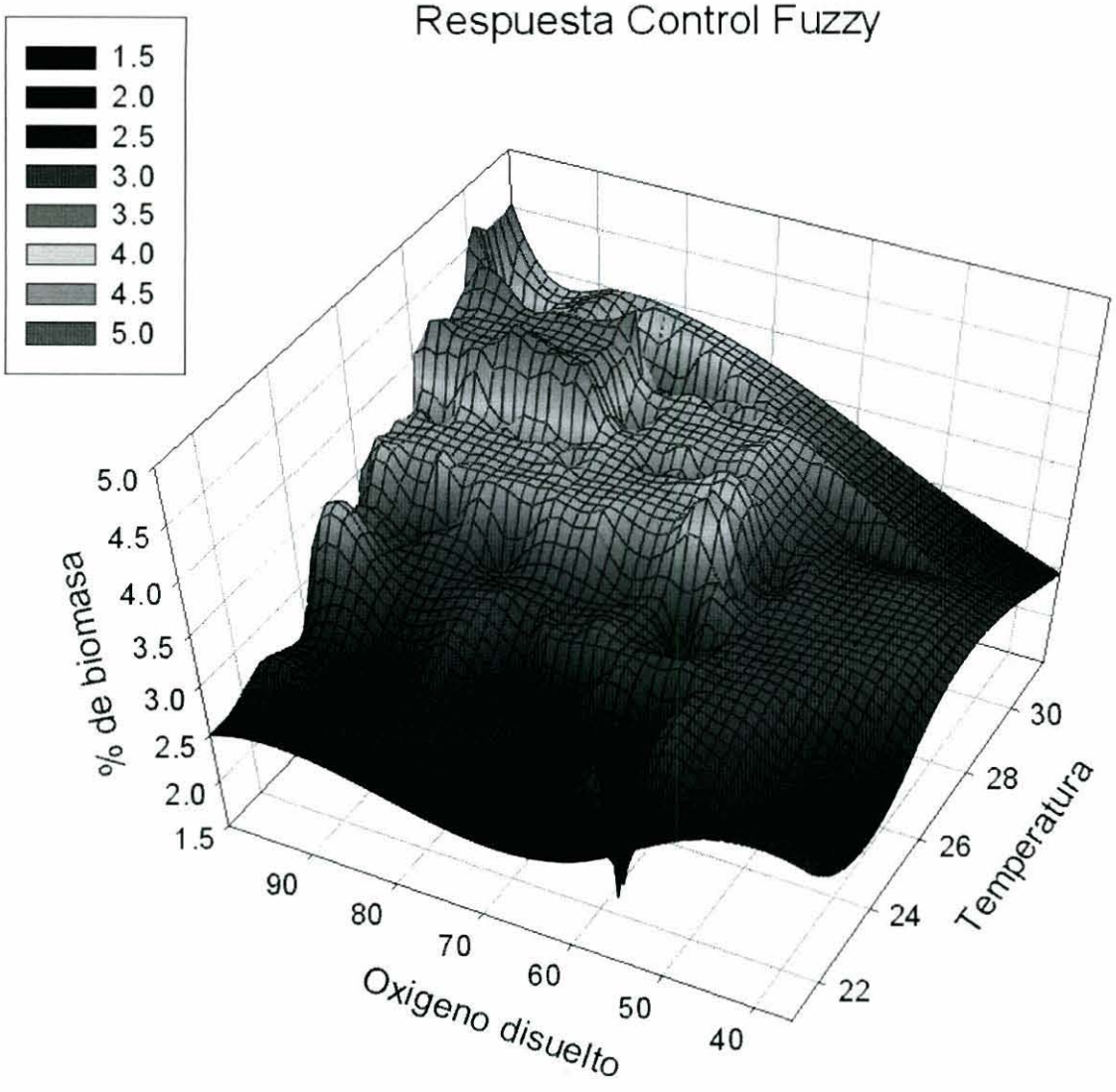

Respuesta Control Fuzzy
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
% de biomasa
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Oxigeno disuelto
90
80
70
60
50
40
Temperatura
30
28
26
24
22

Figura 33.

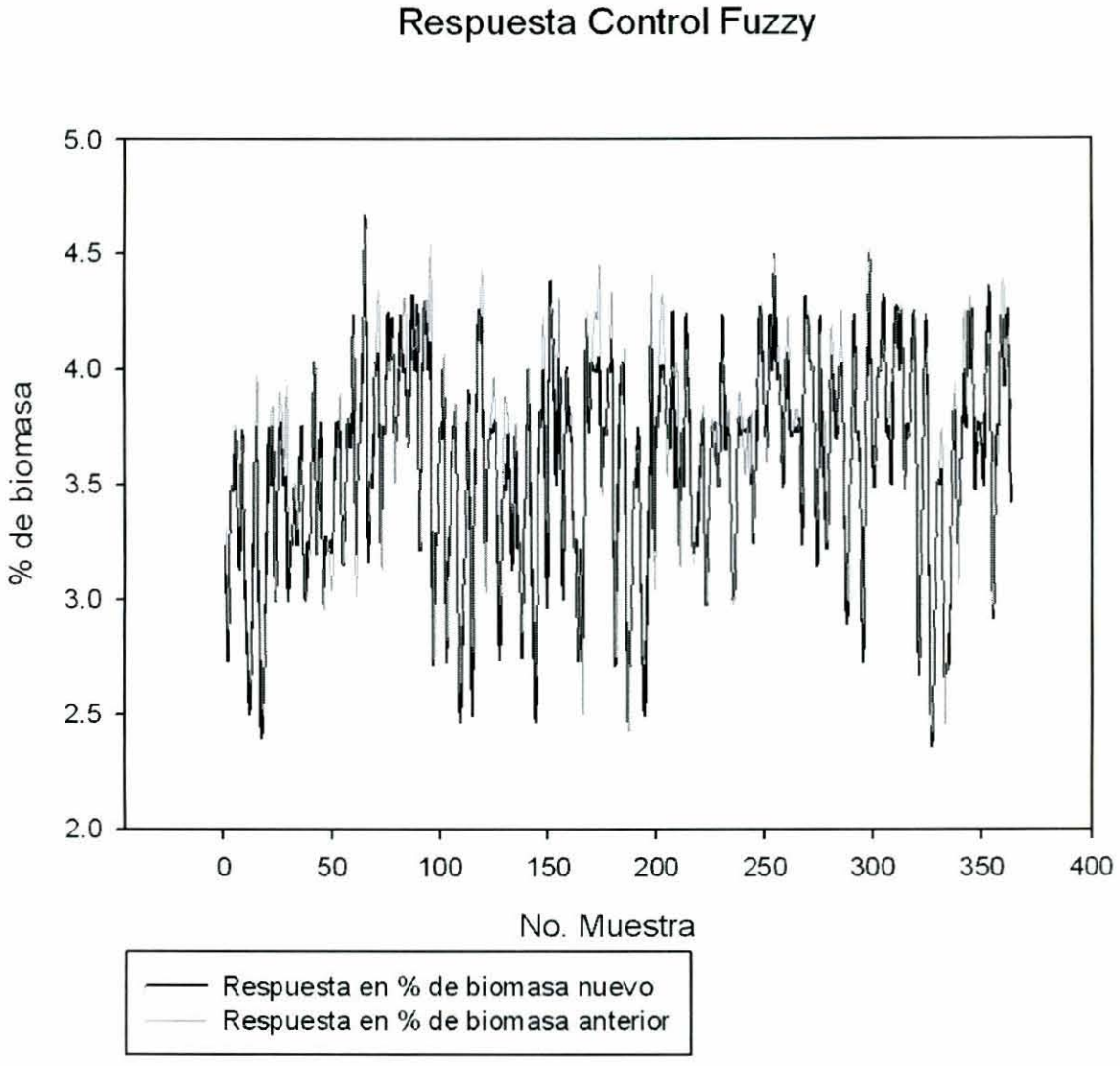

Respuesta Control Fuzzy
5.0
4.5
4.0
3.5
3.0
2.5
2.0
% de biomasa
0
50
100
150
200
250
300
350
400
No. Muestra
Respuesta en % de biomasa nuevo
Respuesta en % de biomasa anterior

Figura 34.

Conclusiones de pruebas en Hardware y Software

En general se ha observado una alta compatibilidad con los distintos sistemas operativos Windows, lo que ha favorecido a la fácil incorporación del software en casi cualquier computadora. Además el sistema tiene una respuesta de trabajo muy buena esto como resultado de la no saturación de la memoria creando objetos dinámicamente (los objetos son creados cuando se utilizan, en caso contrario no se carga nada a la memoria); una consecuencia favorable de ello es la posibilidad de que el sistema permanezca funcionando por meses (tiempo máximo de prueba 9 meses con resultados exitosos). El hecho de ser un sistema de control por horarios nos permite utilizarlo para cualquier sistema donde el elemento final de control sea un actuador eléctrico. Y por ultimo el modulo de video en vivo así como el empleo de tecnologías de comunicación inalámbricas (desarrollo no correspondiente a este trabajo, pero si implementado en él) permiten al sistema estar vigente en cuanto a medios tecnológicos. Desafortunadamente no todo es tan perfecto quedando el numero de salidas fijo a 8 y 8 entradas analógicas por modulo; en el caso de necesitar mas salidas o entradas es necesario instalar otro modulo y darlo de alta en el dispositivo de comunicación inalámbrica. Aunque lo anterior no representa una limitante tajante, se considera como una capacidad del hardware. En tanto a la etapa de potencia es también bastante flexible quedando la opción de manejar dispositivos monofásicos, bifásicos, trifásicos, etc., siempre tomando en cuenta los correspondientes acoplamientos eléctricos. Es por todo ello que el sistema en general es por si un complejo completo para el control de dispositivos por tiempos en general (en su versión control por tiempos), y un sistema mas especializado con inteligencia programable aplicable en la acuicultura para cualquier especie (pudiendo controlar/monitorear cualquier otra variable como pH, salinidad, dureza, TAN, etc.) en su versión Fuzzy.

Conclusiones finales

Desde una perspectiva global del proyecto es evidente que el sistema a parte de tener una funcionalidad dirigida hacia el control de sistemas desde un punto de vista ingenieril, se hizo notar que esto implico cubrir áreas de diseño en electrónica, electricidad, panoramas generales sobre la interacción del humano y las maquinas, diseño de software, y ámbitos generales sobre normas en usabilidad; con todas estas disciplinas interactuando entre si se construyo un sistema que por si solo es capaz de estar dentro de estándares industriales. Partiendo de tal hecho es evidente que el proyecto queda consumado como un desarrollo completo, a pesar de no ser un requisito, el sistema a tenido buena aceptación por piscicultores del sector productivo y su desarrollo no se extralimita a la conclusión de este trabajo de investigación sino al continuidad del proyecto ahora planteando una vista prospectiva hacia el desarrollo de un sistema independiente de una computadora esta vez ahora remplazando el computador por un sistema mínimo desarrollado para eliminar el uso del PC y así entonces incrementar los aspectos de compatibilidad cuyo ultimo requisito será el contar con una alimentación de energía alterna. Como parte de una observación final, se considera que la interacción de distintas disciplinas otorga un entendimiento mas completo del problema y por consiguiente una mejor solución. La reducción de las entradas disponibles para el usuario (entiéndase por entrada teclado, mouse) le permitieron entonces un sistema fácil de manejar así como un entendimiento y manejo mas completo del mismo; aunado a ello el espacio utilizado por el sistema se redujo en mas de un 60%. Al hacer uso de controles estándar en el ambiente de sistema operativo mas utilizado a nivel mundial el usuario identifico intuitivamente el funcionamiento del sistema y los objetos que tenían una funcionalidad poco común fueron explicadas a través de la "ayuda instantánea" parte del proyecto.

LITERATURA CONSULTADA

Booth, P. (1989). An introduction to Human-Computer Interaction- Hillsdale, New Jersey: Lawrence Earlbaum Associates.

Card, SK; Morán,TP., & Newell, A.(1983). The psychology of human computer interaction. Hillsdale, New Jersey: Lawrence Earlbaum Associates.

Cobo, C. (2006). De la interfaz a la información. Tomado de la URL http://e-rgonomic.blogspot.com/2006/05/de-la-interfaz-la-informacin.html

Cobo, C. (2007a). Aprendizaje Invisible: M-learning + Personal Learning Environment. Tomado de la URL http://e-rgonomic.blogspot.com/2007/05/aprendizaje-invisible-m-learning.html

Cobo, C. (2007b). Movilidad y Conectividad para el Futuro. URL http://e-rgonomic.blogspot.com/search?q=m-learning

Cobo, C. (2008). Uso Moderado de las TIC para un Mejor Depempeño. URL http://e-rgonomic.blogspot.com/search/label/aprendizaje%20colaborativo

Constantine, L. L., & Windl, H. (2003). Usage-Centered Design: Scalability and Integration with Software Engineering. Human-Computer Interaction: theory and Practice.vol 1). Lawrence Erlbaum Associates.

Gulliksen, J., & Göransson, B. (2003). Usability Design: Integrating User-Centred Systems Design in the Software Development Process. Proceedings of INTERACT 2003, Zurich (Suiza)

International Standard (1999). ISO 13407. Human-centred design processes for interactive systems.

Johnson-Laird P. N. (1988).The computer and the mind: an introduction to cognitive Science. Galsgow: William Collins Sonsnad Co.

Johnson, P. (1992). Human-Computer Interaction: Psychology, Task Analysis and Software Engineering. McGraw Hill.

Laurel, B. (1990).The art of human-computer interface design. Massachusettes: Addison- Wesley.

Laurel, B. (1993). Computers as Theatre. Massachussettes: Addison Wesley.

Lewis, C., & Reiman, J. (1993). Task-Centered user interface design: A Practical Introduction. University of Colorado.

Mcluhan, M. (1964). Understanding media: the extensions of man. New York: McGraw-Hill.

McLuhan, M., & Fiore,Q. (1967). The Medium is the Massage. New York: Bantam.

Norman, D..A., & Shallice, T.(1986). Attention to action: willed and automatic control of behavior. In R.J. Davidson, Schwarts, G.E., & Shapiro, D. (Eds.), *Conciousness and self-regulation. Advances in resarch and theory, 4,* 1-18. New York: Plenum Press. Tomado de Baddeley, A.(1999).

Norman, D.A.(1986). Cognitive Engineering en User centered system design: new perspectives on human–computer interaction. New Jersey: Lawrence Erlbaum Hillsdale.

Norman, D.A.(1990). The design of evereday things. Nueva York: Doubleday.

Norman, D. A., (1998).The Invisible computer. Cambridge, Massachusetts, London, England: The MIT Press. 302.

Preece, J. (1994). Human–Computer Interaction. Massachussetts: Addison–Wesley.

Raskin, J.(2000). The human interface: new direction for design interactive system. Massachussetts: Addison Wesley.

Sheinderman, B. (1987).Designing the User Interface: Strategies for Effective Human Computer Interaction.Massachusetts: Addison-Wesley Publishing.

Shneiderman, B.(1993). Designing the User Interface: Strategies for Effective Human- Computer Interaction.Massachusetts: Addison-Wesley Publishing.

Shneiderman, B.,y Plaisant, C. (2004). Designing the user interface: Strategies for effective Human-Computer Interaction (4th ed.). Boston, MA: Addison Wesley.

Von Bertalanffy, L. (1976). Teoría general de los sistemas. México: F.C.E.

Von Bertalanffy, L (1992). Perspectivas en la Teoría general de los sistemas. España: Alianza.

APENDICE

A continuación se lista parte del código del programa, solo lo mas relevante.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace TimeChartMon
{
    public partial class TimingControl : Form
    {
        public Form keyboard = new KeyBoard();
        public struct tiempo
        {
            public int hora;
            public int minuto;
            public int segundo;
        };

        public void pp_of(byte pin)
        {
            byte puerto = 0;

            puerto =  PortAccess.Input(888);
            switch (pin)
            {
                case 1:
                    {
                        PortAccess.Output(888, puerto | 0x01);
                        break;
                    }
                case 2:
                    {
                        PortAccess.Output(888, puerto | 0x02);
                        break;
                    }

                case 3:
                    {
                        PortAccess.Output(888, puerto | 0x04);
                        break;
                    }
                case 4:
                    {
                        PortAccess.Output(888, puerto | 0x08);
                        break;
                    }
                case 5:
                    {
                        PortAccess.Output(888, puerto | 0x10);
```

```
                break;
            }
        case 6:
            {
                PortAccess.Output(888, puerto | 0x20);
                break;
            }
        case 7:
            {
                PortAccess.Output(888, puerto | 0x40);
                break;
            }
        case 8:
            {
                PortAccess.Output(888, puerto | 0x80);
                break;
            }
    }

}

public void pp_on(bool timer, byte pin)
{
    byte puerto = 0;

    puerto = PortAccess.Input(888);
    switch (pin)
    {
        case 1:
            {
                PortAccess.Output(888, puerto & 0xFE);
                timer1.Enabled = timer;
                break;
            }
        case 2:
            {
                PortAccess.Output(888, puerto & 0xFD);
                timer2.Enabled = timer;
                break;
            }
        case 3:
            {
                PortAccess.Output(888, puerto & 0xFB);
                timer3.Enabled = timer;
                break;
            }
        case 4:
            {
                PortAccess.Output(888, puerto & 0xF7);
                timer4.Enabled = timer;
                break;
            }
        case 5:
            {
                PortAccess.Output(888, puerto & 0xEF);
                timer5.Enabled = timer;
                break;
            }
        case 6:
            {
                PortAccess.Output(888, puerto & 0xDF);
```

```
                timer6.Enabled = timer;
                break;
            }
        case 7:
            {
                PortAccess.Output(888, puerto & 0xBF);
                timer7.Enabled = timer;
                break;
            }
        case 8:
            {
                PortAccess.Output(888, puerto & 0x7F);
                timer8.Enabled = timer;
                break;
            }
    }
}

public TimingControl()
{
    DataTable etiq = new DataTable("Etiquetas");
    InitializeComponent();

    etiq.ReadXmlSchema("C:\\scetil.xml");
    etiq.ReadXml("C:\\etil.xml");
    tabPage1.Text = etiq.Rows[0].ItemArray.GetValue(0).ToString();
    tabPage2.Text = etiq.Rows[1].ItemArray.GetValue(0).ToString();
    tabPage3.Text = etiq.Rows[2].ItemArray.GetValue(0).ToString();
    tabPage4.Text = etiq.Rows[3].ItemArray.GetValue(0).ToString();
    tabPage5.Text = etiq.Rows[4].ItemArray.GetValue(0).ToString();
    tabPage6.Text = etiq.Rows[5].ItemArray.GetValue(0).ToString();
    tabPage7.Text = etiq.Rows[6].ItemArray.GetValue(0).ToString();
    tabPage8.Text = etiq.Rows[7].ItemArray.GetValue(0).ToString();

    checkBox1.Text = tabPage1.Text;
    checkBox2.Text = tabPage2.Text;
    checkBox3.Text = tabPage3.Text;
    checkBox4.Text = tabPage4.Text;
    checkBox5.Text = tabPage5.Text;
    checkBox6.Text = tabPage6.Text;
    checkBox7.Text = tabPage7.Text;
    checkBox8.Text = tabPage8.Text;
    etiq.Dispose();
}

private void timer1_Tick(object sender, EventArgs e)
{
    pp_of(1);
    timer1.Enabled = false;
}

private void timer2_Tick(object sender, EventArgs e)
{
    pp_of(2);
    timer2.Enabled = false;
}

private void timer3_Tick(object sender, EventArgs e)
{
    pp_of(3);
    timer3.Enabled = false;
```

```
}

private void timer4_Tick(object sender, EventArgs e)
{
    pp_of(4);
    timer4.Enabled = false;
}

private void timer5_Tick(object sender, EventArgs e)
{
    pp_of(5);
    timer5.Enabled = false;
}

private void timer6_Tick(object sender, EventArgs e)
{
    pp_of(6);
    timer6.Enabled = false;
}

private void timer7_Tick(object sender, EventArgs e)
{
    pp_of(7);
    timer7.Enabled = false;
}

private void timer8_Tick(object sender, EventArgs e)
{
    pp_of(8);
    timer8.Enabled = false;
}

private void Form1_Load(object sender, EventArgs e)
{
    DataTable da = new DataTable("dispositivo");
    DataTable db = new DataTable("dispositivo");
    DataTable dc = new DataTable("dispositivo");
    DataTable dd = new DataTable("dispositivo");
    DataTable de = new DataTable("dispositivo");
    DataTable df = new DataTable("dispositivo");
    DataTable dg = new DataTable("dispositivo");
    DataTable dh = new DataTable("dispositivo");

    /*pp_of(1);
    pp_of(2);
    pp_of(3);
    pp_of(4);
    pp_of(5);
    pp_of(6);
    pp_of(7);
    pp_of(8);*/
    da.ReadXmlSchema(@"C:\schema.xml");
    da.ReadXml(@"C:\horario1.xml");
    db.ReadXmlSchema(@"C:\schema.xml");
    db.ReadXml(@"C:\horario2.xml");
    dc.ReadXmlSchema(@"C:\schema.xml");
    dc.ReadXml(@"C:\horario3.xml");
    dd.ReadXmlSchema(@"C:\schema.xml");
    dd.ReadXml(@"C:\horario4.xml");
    de.ReadXmlSchema(@"C:\schema.xml");
    de.ReadXml(@"C:\horario5.xml");
```

```
            df.ReadXmlSchema(@"C:\schema.xml");
            df.ReadXml(@"C:\horario6.xml");
            dg.ReadXmlSchema(@"C:\schema.xml");
            dg.ReadXml(@"C:\horario7.xml");
            dh.ReadXmlSchema(@"C:\schema.xml");
            dh.ReadXml(@"C:\horario8.xml");
            da.AcceptChanges();
            db.AcceptChanges();
            dc.AcceptChanges();
            dd.AcceptChanges();
            de.AcceptChanges();
            df.AcceptChanges();
            dg.AcceptChanges();
            dh.AcceptChanges();

            this.ac_1.DataSource = da.DefaultView;
            this.ac_2.DataSource = db.DefaultView;
            this.ac_3.DataSource = dc.DefaultView;
            this.ac_4.DataSource = dd.DefaultView;
            this.ac_5.DataSource = de.DefaultView;
            this.ac_6.DataSource = df.DefaultView;
            this.ac_7.DataSource = dg.DefaultView;
            this.ac_8.DataSource = dh.DefaultView;

            timer9.Enabled = true;
            keyboard.Tag = this.Name;
        }

        private void timer9_Tick(object sender, EventArgs e)
        {
            string hora = "";

            if (System.DateTime.Now.Hour < 10) hora = "0";
            hora = hora + Convert.ToString(System.DateTime.Now.Hour) +
":";
            if (System.DateTime.Now.Minute < 10) hora = hora + "0";
            hora = hora + System.DateTime.Now.Minute.ToString() + ":";
            if (System.DateTime.Now.Second < 10) hora = hora + "0";
            hora = hora + System.DateTime.Now.Second.ToString();
            time.Text = hora;
            ac_sa();
        }

        public void ac_sa()
        {
            byte x;

            for (x = 0; x < ac_1.RowCount; x++)
            {
                if (Convert.ToString(ac_1.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_1.Rows[x].Cells[0].Value))
                    {
                        timer1.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 1);
                    }
                }
            }
```

```
            for (x = 0; x < ac_2.RowCount; x++)
            {
                if (Convert.ToString(ac_2.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_2.Rows[x].Cells[0].Value))
                    {
                        timer2.Interval =
((Convert.ToInt32(ac_2.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_2.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 2);
                    }
                }
            }
            for (x = 0; x < ac_3.RowCount; x++)
            {
                if (Convert.ToString(ac_3.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_3.Rows[x].Cells[0].Value))
                    {
                        timer3.Interval =
((Convert.ToInt32(ac_3.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_3.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 3);
                    }
                }
            }
            for (x = 0; x < ac_4.RowCount; x++)
            {
                if (Convert.ToString(ac_4.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_4.Rows[x].Cells[0].Value))
                    {
                        timer4.Interval =
((Convert.ToInt32(ac_4.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_4.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 4);
                    }
                }
            }
            for (x = 0; x < ac_5.RowCount; x++)
            {
                if (Convert.ToString(ac_5.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_5.Rows[x].Cells[0].Value))
                    {
                        timer5.Interval =
((Convert.ToInt32(ac_5.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_5.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 5);
                    }
                }
            }
            for (x = 0; x < ac_6.RowCount; x++)
            {
                if (Convert.ToString(ac_6.Rows[x].Cells[0].Value) != "")
                {
```

```
                    if (time.Text ==
Convert.ToString(ac_6.Rows[x].Cells[0].Value))
                    {
                        timer6.Interval =
((Convert.ToInt32(ac_6.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_6.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 6);
                    }
                }
            }
            for (x = 0; x < ac_7.RowCount; x++)
            {
                if (Convert.ToString(ac_7.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_7.Rows[x].Cells[0].Value))
                    {
                        timer7.Interval =
((Convert.ToInt32(ac_7.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_7.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 7);
                    }
                }
            }
            for (x = 0; x < ac_8.RowCount; x++)
            {
                if (Convert.ToString(ac_8.Rows[x].Cells[0].Value) != "")
                {
                    if (time.Text ==
Convert.ToString(ac_8.Rows[x].Cells[0].Value))
                    {
                        timer8.Interval =
((Convert.ToInt32(ac_8.Rows[x].Cells[1].Value) * 60) +
Convert.ToInt32(ac_8.Rows[x].Cells[2].Value)) * 1000;
                        pp_on(true, 8);
                    }
                }
            }
            /*for (x = 0; x < ac_1.RowCount; x++)
            {
                if (Convert.ToString(ac_1.Rows[x].Cells[0].Value) != "")
                {
                    if (textBox73.Text ==
Convert.ToString(ac_1.Rows[x].Cells[1].Value))
                    {
                        switch
(Convert.ToByte(ac_1.Rows[x].Cells[0].Value))
                        {
                            case 1:
                                {
                                    timer1.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 2:
                                {
                                    timer2.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
```

```
                                }
                            case 3:
                                {
                                    timer3.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 4:
                                {
                                    timer4.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 5:
                                {
                                    timer5.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 6:
                                {
                                    timer6.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 7:
                                {
                                    timer7.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                            case 8:
                                {
                                    timer8.Interval =
((Convert.ToInt32(ac_1.Rows[x].Cells[2].Value) * 60) +
Convert.ToInt32(ac_1.Rows[x].Cells[3].Value)) * 1000;
                                    break;
                                }
                        }

pp_on(Convert.ToByte(ac_1.Rows[x].Cells[0].Value));
                    }
                }
            }*/
            //dataGridView1.Rows[0].Cells[1].Value = "12:30:00";
            //dataGridView1.Rows[0].Cells[2].Value = "5";
            //dataGridView1.Rows[0].Cells[3].Value = "0";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            //System.IO.File.Open("C:\\holl.crg",
System.IO.FileMode.OpenOrCreate);
            //System.IO.File.WriteAllText("C:\\holl.crg",
dataGridView1.Rows[0].Cells[0].Value);
            DataTable da = ((DataView)this.ac_1.DataSource).Table;
```

```
            da.WriteXml(@"C:\horario1.xml");
            da.WriteXmlSchema(@"C:\schema.xml");
            DataTable db = ((DataView)this.ac_2.DataSource).Table;
            db.WriteXml(@"C:\horario2.xml");
            DataTable dc = ((DataView)this.ac_3.DataSource).Table;
            dc.WriteXml(@"C:\horario3.xml");
            DataTable dd = ((DataView)this.ac_4.DataSource).Table;
            dd.WriteXml(@"C:\horario4.xml");
            DataTable de = ((DataView)this.ac_5.DataSource).Table;
            de.WriteXml(@"C:\horario5.xml");
            DataTable df = ((DataView)this.ac_6.DataSource).Table;
            df.WriteXml(@"C:\horario6.xml");
            DataTable dg = ((DataView)this.ac_7.DataSource).Table;
            dg.WriteXml(@"C:\horario7.xml");
            DataTable dh = ((DataView)this.ac_8.DataSource).Table;
            dh.WriteXml(@"C:\horario8.xml");
        }

        private void saveFileDialog1_FileOk(object sender,
CancelEventArgs e)
        {

        }

        private void button4_Click(object sender, EventArgs e)
        {
            DataTable da = ((DataView)this.ac_1.DataSource).Table;
            DataTable db = ((DataView)this.ac_2.DataSource).Table;
            DataTable dc = ((DataView)this.ac_3.DataSource).Table;
            DataTable dd = ((DataView)this.ac_4.DataSource).Table;
            DataTable de = ((DataView)this.ac_5.DataSource).Table;
            DataTable df = ((DataView)this.ac_6.DataSource).Table;
            DataTable dg = ((DataView)this.ac_7.DataSource).Table;
            DataTable dh = ((DataView)this.ac_8.DataSource).Table;

            saveFileDialog1.Title = "Guardar esquema de tabla...";
            saveFileDialog1.FileName = "";
            saveFileDialog1.ShowDialog();
            if (saveFileDialog1.FileName != "")
da.WriteXmlSchema(saveFileDialog1.FileName);

            saveFileDialog1.Title = "Guardar tabla...";
            saveFileDialog1.FileName = "";
            saveFileDialog1.ShowDialog();
            if (saveFileDialog1.FileName != "")
            {

da.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 1" + ".xml");

db.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 2" + ".xml");

dc.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 3" + ".xml");

dd.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 4" + ".xml");

de.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 5" + ".xml");
```

```
df.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 6" + ".xml");

dg.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 7" + ".xml");

dh.WriteXml(saveFileDialog1.FileName.Remove(saveFileDialog1.FileName.Leng
th - 4) + " 8" + ".xml");
                /*db.WriteXml(saveFileDialog1.FileName + "2");
                dc.WriteXml(saveFileDialog1.FileName + "3");
                dd.WriteXml(saveFileDialog1.FileName + "4");
                de.WriteXml(saveFileDialog1.FileName + "5");
                df.WriteXml(saveFileDialog1.FileName + "6");
                dg.WriteXml(saveFileDialog1.FileName + "7");
                dh.WriteXml(saveFileDialog1.FileName + "8");*/
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if ((tabulador.SelectedIndex == 0) && (ac_1.RowCount > 1))
ac_1.Rows.Remove(ac_1.CurrentRow);
            if ((tabulador.SelectedIndex == 1) && (ac_2.RowCount > 1))
ac_2.Rows.Remove(ac_2.CurrentRow);
            if ((tabulador.SelectedIndex == 2) && (ac_3.RowCount > 1))
ac_3.Rows.Remove(ac_3.CurrentRow);
            if ((tabulador.SelectedIndex == 3) && (ac_4.RowCount > 1))
ac_4.Rows.Remove(ac_4.CurrentRow);
            if ((tabulador.SelectedIndex == 4) && (ac_5.RowCount > 1))
ac_5.Rows.Remove(ac_5.CurrentRow);
            if ((tabulador.SelectedIndex == 5) && (ac_6.RowCount > 1))
ac_6.Rows.Remove(ac_6.CurrentRow);
            if ((tabulador.SelectedIndex == 6) && (ac_7.RowCount > 1))
ac_7.Rows.Remove(ac_7.CurrentRow);
            if ((tabulador.SelectedIndex == 7) && (ac_8.RowCount > 1))
ac_8.Rows.Remove(ac_8.CurrentRow);
        }

        private void button5_Click(object sender, EventArgs e)//con
esquema normal
        {
            DataTable dt = new DataTable("dispositivo");
            bool a = false, b = false;

            openFileDialog1.Title = "Abrir esquema...";
            openFileDialog1.FileName = "";
            openFileDialog1.ShowDialog();
            if (openFileDialog1.FileName != "")
            {
                dt.ReadXmlSchema(openFileDialog1.FileName);
                a = true;
            }
            openFileDialog1.Title = "Abrir tabla...";
            openFileDialog1.FileName = "";
            openFileDialog1.ShowDialog();
            if (openFileDialog1.FileName != "")
            {
                dt.ReadXml(openFileDialog1.FileName);
                b = true;
            }
```

```
            dt.AcceptChanges();
            if (a && b)
            {
                if (tabulador.SelectedIndex == 0) this.ac_1.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 1) this.ac_2.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 2) this.ac_3.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 3) this.ac_4.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 4) this.ac_5.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 5) this.ac_6.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 6) this.ac_7.DataSource =
dt.DefaultView;
                if (tabulador.SelectedIndex == 7) this.ac_8.DataSource =
dt.DefaultView;
                //this.ac_2.DataSource = dt.DefaultView;
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            DataTable dt = new DataTable("tmptable");
            int j = 1, h, m, s;

            dt.ReadXmlSchema(@"C:\schema.xml");
            for (h = 10; h < 12; h++)
            {
                for (m = 0; m < 60; m = m + 2)
                {
                    s = 0;
                    dt.Rows.Add(j.ToString(), h.ToString() + ":" +
m.ToString() + ":" + s.ToString(), "1", "0");
                    s = 30;
                    dt.Rows.Add(Convert.ToString(j + 1), h.ToString() +
":" + m.ToString() + ":" + s.ToString(), "1", "0");
                    s = 0;
                    dt.Rows.Add(Convert.ToString(j + 2), h.ToString() +
":" + Convert.ToString(m + 1) + ":" + s.ToString(), "1", "0");
                    s = 30;
                    dt.Rows.Add(Convert.ToString(j + 3), h.ToString() +
":" + Convert.ToString(m + 1) + ":" + s.ToString(), "1", "0");
                }
            }
            dt.AcceptChanges();
            this.ac_1.DataSource = dt.DefaultView;
        }

        private void copiarToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            DataTable dt = ((DataView)this.ac_1.DataSource).Table;
            int i;

            //dt.ReadXmlSchema(@"C:\schema.xml");
            for (i = 0; i < ac_1.SelectedCells.Count; i = i + 4)
            {
```

```
                dt.Rows.Add(ac_1.SelectedCells[i].Value,
Convert.ToByte(ac_1.SelectedCells[i + 1].Value),
Convert.ToByte(ac_1.SelectedCells[i + 2].Value));
            }
            dt.AcceptChanges();
            this.ac_1.DataSource = dt.DefaultView;
        }

        private void automatico_CheckedChanged(object sender, EventArgs e)
        {
            if (manual.Checked == false)
            {
                timer9.Enabled = true;
                ar_gu.Visible = true;
                gu_pr.Visible = true;
                ab_hr.Visible = true;
                el_pr.Visible = true;
                txt_yi.Visible = true;
                fx_yi.Visible = true;
                tabulador.Visible = true;

                this.Height = 460;

                nud_hin.Visible = true;
                nud_hpe.Visible = true;
                nud_mdu.Visible = true;
                nud_mpe.Visible = true;
                nud_npr.Visible = true;
                nud_sdu.Visible = true;
                nud_sin.Visible = true;
                nud_spe.Visible = true;

                lbl_dur.Visible = true;
                lbl_hhm.Visible = true;
                lbl_hmm.Visible = true;
                lbl_ini.Visible = true;
                lbl_mms.Visible = true;
                lbl_per.Visible = true;
                lbl_pro.Visible = true;

                checkBox1.Checked = false;
                checkBox2.Checked = false;
                checkBox3.Checked = false;
                checkBox4.Checked = false;
                checkBox5.Checked = false;
                checkBox6.Checked = false;
                checkBox7.Checked = false;
                checkBox8.Checked = false;
                checkBox1.Visible = false;
                checkBox2.Visible = false;
                checkBox3.Visible = false;
                checkBox4.Visible = false;
                checkBox5.Visible = false;
                checkBox6.Visible = false;
                checkBox7.Visible = false;
                checkBox8.Visible = false;
                pp_of(1);
                pp_of(2);
                pp_of(3);
                pp_of(4);
                pp_of(5);
```

```
            pp_of(6);
            pp_of(7);
            pp_of(8);
        }
    }

    private void manual_CheckedChanged(object sender, EventArgs e)
    {
        if (manual.Checked == true)
        {
            timer9.Enabled = false;
            ar_gu.Visible = false;
            gu_pr.Visible = false;
            ab_hr.Visible = false;
            el_pr.Visible = false;
            txt_yi.Visible = false;
            fx_yi.Visible = false;

            nud_hin.Visible = false;
            nud_hpe.Visible = false;
            nud_mdu.Visible = false;
            nud_mpe.Visible = false;
            nud_npr.Visible = false;
            nud_sdu.Visible = false;
            nud_sin.Visible = false;
            nud_spe.Visible = false;

            lbl_dur.Visible = false;
            lbl_hhm.Visible = false;
            lbl_hmm.Visible = false;
            lbl_ini.Visible = false;
            lbl_mms.Visible = false;
            lbl_per.Visible = false;
            lbl_pro.Visible = false;

            this.Height = 400;

            tabulador.Visible = false;
            checkBox1.Visible = true;
            checkBox2.Visible = true;
            checkBox3.Visible = true;
            checkBox4.Visible = true;
            checkBox5.Visible = true;
            checkBox6.Visible = true;
            checkBox7.Visible = true;
            checkBox8.Visible = true;
        }
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        if (checkBox1.Checked == true)
        {
            pp_on(false, 1);
        }
        else
        {
            pp_of(1);
        }
    }
```

```
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2.Checked == true)
    {
        pp_on(false, 2);
    }
    else
    {
        pp_of(2);
    }
}

private void checkBox3_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3.Checked == true)
    {
        pp_on(false, 3);
    }
    else
    {
        pp_of(3);
    }
}

private void checkBox4_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox4.Checked == true)
    {
        pp_on(false, 4);
    }
    else
    {
        pp_of(4);
    }
}

private void checkBox5_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox5.Checked == true)
    {
        pp_on(false, 5);
    }
    else
    {
        pp_of(5);
    }
}

private void checkBox6_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox6.Checked == true)
    {
        pp_on(false, 6);
    }
    else
    {
        pp_of(6);
    }
}

private void checkBox7_CheckedChanged(object sender, EventArgs e)
```

```
        {
            if (checkBox7.Checked == true)
            {
                pp_on(false, 7);
            }
            else
            {
                pp_of(7);
            }
        }

        private void checkBox8_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox8.Checked == true)
            {
                pp_on(false, 8);
            }
            else
            {
                pp_of(8);
            }
        }

        private void tabulador_DoubleClick(object sender, EventArgs e)
        {

        }

        private void ac_2_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void ac_3_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void cambiarNombreToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            /*cn_mb.Visible = true;
            cn_mb.Left = tabulador.Left + (64 * tabulador.SelectedIndex);
            cn_mb.Focus();*/
            if (tabulador.SelectedIndex == 0)
            {
                tabPage1.Text = toolStripTextBox1.Text;
                checkBox1.Text = tabPage1.Text;
            }
            else if (tabulador.SelectedIndex == 1)
            {
                tabPage2.Text = toolStripTextBox1.Text;
                checkBox2.Text = tabPage2.Text;
            }
            else if (tabulador.SelectedIndex == 2)
            {
                tabPage3.Text = toolStripTextBox1.Text;
                checkBox3.Text = tabPage3.Text;
            }
```

```
            else if (tabulador.SelectedIndex == 3)
            {
                tabPage4.Text = toolStripTextBox1.Text;
                checkBox4.Text = tabPage4.Text;
            }
            else if (tabulador.SelectedIndex == 4)
            {
                tabPage5.Text = toolStripTextBox1.Text;
                checkBox5.Text = tabPage5.Text;
            }
            else if (tabulador.SelectedIndex == 5)
            {
                tabPage6.Text = toolStripTextBox1.Text;
                checkBox6.Text = tabPage6.Text;
            }
            else if (tabulador.SelectedIndex == 6)
            {
                tabPage7.Text = toolStripTextBox1.Text;
                checkBox7.Text = tabPage7.Text;
            }
            else if (tabulador.SelectedIndex == 7)
            {
                tabPage8.Text = toolStripTextBox1.Text;
                checkBox8.Text = tabPage8.Text;
            }
        }

        private void ac_1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void ac_1_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            if (tabulador.SelectedIndex == 0) txt_yi.Text =
Convert.ToString(ac_1.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 1) txt_yi.Text =
Convert.ToString(ac_2.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 2) txt_yi.Text =
Convert.ToString(ac_3.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 3) txt_yi.Text =
Convert.ToString(ac_4.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 4) txt_yi.Text =
Convert.ToString(ac_5.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 5) txt_yi.Text =
Convert.ToString(ac_6.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 6) txt_yi.Text =
Convert.ToString(ac_7.CurrentCell.RowIndex + 1);
            if (tabulador.SelectedIndex == 7) txt_yi.Text =
Convert.ToString(ac_8.CurrentCell.RowIndex + 1);
        }


        private void ac_2_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }
```

```
        private void ac_3_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ac_4_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ac_5_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ac_6_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ac_7_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ac_8_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            ac_1_CellEnter(sender, e);
        }

        private void ch_im_Click(object sender, EventArgs e)
        {

        }

        private void pa_bo_Click(object sender, EventArgs e)
        {

        }

        private void pa_bo_DoubleClick(object sender, EventArgs e)
        {

        }

        private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
            DataTable etiq = new DataTable("Etiquetas");
            etiq.Columns.Add("Etiqueta", typeof(String));
            etiq.Rows.Add(tabPage1.Text);
            etiq.Rows.Add(tabPage2.Text);
            etiq.Rows.Add(tabPage3.Text);
            etiq.Rows.Add(tabPage4.Text);
            etiq.Rows.Add(tabPage5.Text);
```

```
    etiq.Rows.Add(tabPage6.Text);
    etiq.Rows.Add(tabPage7.Text);
    etiq.Rows.Add(tabPage8.Text);
    etiq.AcceptChanges();
    etiq.WriteXmlSchema("C:\\scetil.xml");
    etiq.WriteXml("C:\\etil.xml");
}

private void checkBox5_MouseClick(object sender, MouseEventArgs e)
{

}

private void cn_mb_TextChanged(object sender, EventArgs e)
{

}

private void toolStripTextBox1_Click(object sender, EventArgs e)
{

}

private void checkBox5_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 4;
}

private void checkBox6_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 5;
}

private void checkBox7_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 6;
}

private void checkBox8_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 7;
}

private void checkBox1_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 0;
}

private void checkBox2_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 1;
}

private void checkBox3_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 2;
}

private void checkBox4_MouseHover(object sender, EventArgs e)
{
    tabulador.SelectedIndex = 3;
```

```
        }

        private void contextMenuStrip1_Opening(object sender,
CancelEventArgs e)
        {
            if (tabulador.SelectedIndex == 0)
            {
                toolStripTextBox1.Text = tabPage1.Text;
            }
            else if (tabulador.SelectedIndex == 1)
            {
                toolStripTextBox1.Text = tabPage2.Text;
            }
            else if (tabulador.SelectedIndex == 2)
            {
                toolStripTextBox1.Text = tabPage3.Text;
            }
            else if (tabulador.SelectedIndex == 3)
            {
                toolStripTextBox1.Text = tabPage4.Text;
            }
            else if (tabulador.SelectedIndex == 4)
            {
                toolStripTextBox1.Text = tabPage5.Text;
            }
            else if (tabulador.SelectedIndex == 5)
            {
                toolStripTextBox1.Text = tabPage6.Text;
            }
            else if (tabulador.SelectedIndex == 6)
            {
                toolStripTextBox1.Text = tabPage7.Text;
            }
            else if (tabulador.SelectedIndex == 7)
            {
                toolStripTextBox1.Text = tabPage8.Text;
            }
            //toolStripTextBox1.Text = "";
        }

        private void tabulador_Selecting(object sender,
TabControlCancelEventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.automatico.Checked = true;
            this.Visible = false;
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            DataTable gentbl = new DataTable("dispositivo");
            int n, hi, mi, si, hp, mp, sp, m, s, a;
            tiempo opa, opb, resultado;

            n = Convert.ToInt16(nud_npr.Value);
            hi = Convert.ToInt16(nud_hin.Value);
            mi = Convert.ToInt16(nud_min.Value);
```

```
            si = Convert.ToInt16(nud_sin.Value);
            hp = Convert.ToInt16(nud_hpe.Value);
            mp = Convert.ToInt16(nud_mpe.Value);
            sp = Convert.ToInt16(nud_spe.Value);
            m = Convert.ToInt16(nud_mdu.Value);
            s = Convert.ToInt16(nud_sdu.Value);

            gentbl.Columns.Add("Hora I.", typeof(String));
            gentbl.Columns.Add("Min", typeof(byte));
            gentbl.Columns.Add("Seg", typeof(byte));
            opa.hora = hi;
            opa.minuto = mi;
            opa.segundo = si;
            opb.hora = hp;
            opb.minuto = mp;
            opb.segundo = sp;
            resultado = sumar_hr(opa, opb);
            gentbl.Rows.Add(Convert.ToString(opa.hora) + ":" +
Convert.ToString(opa.minuto) + ":" + Convert.ToString(opa.segundo), m, s);
            for (a = 1; a < n; a++)
            {
                string hora = "";

                if (System.DateTime.Now.Hour < 10) hora = "0";
                hora = hora + Convert.ToString(resultado.hora) + ":";
                if (resultado.minuto < 10) hora = hora + "0";
                hora = hora + resultado.minuto.ToString() + ":";
                if (resultado.segundo < 10) hora = hora + "0";
                hora = hora + resultado.segundo.ToString();
                gentbl.Rows.Add(hora, m, s);
                resultado = sumar_hr(resultado, opb);
            }
            //gentbl.Rows.Add(Convert.ToString(resultado.hora) + ":" +
Convert.ToString(resultado.minuto) + ":" +
Convert.ToString(resultado.segundo), m, s);
            gentbl.AcceptChanges();
            if (tabulador.SelectedIndex == 0)
            {
                this.ac_1.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 1)
            {
                this.ac_2.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 2)
            {
                this.ac_3.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 3)
            {
                this.ac_4.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 4)
            {
                this.ac_5.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 5)
            {
                this.ac_6.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 6)
```

```
            {
                this.ac_7.DataSource = gentbl.DefaultView;
            }
            else if (tabulador.SelectedIndex == 7)
            {
                this.ac_8.DataSource = gentbl.DefaultView;
            }
        }

        public tiempo sumar_hr( tiempo uno, tiempo dos)
        {
            tiempo suma;
            suma.hora = 0;
            suma.minuto = 0;
            suma.segundo = 0;

            suma.segundo = uno.segundo + dos.segundo;
            if (suma.segundo >= 60)
            {
                suma.segundo = suma.segundo - 60;
                suma.minuto = 1;
            }
            suma.minuto = suma.minuto + uno.minuto + dos.minuto;
            if (suma.minuto >= 60)
            {
                suma.minuto = suma.minuto - 60;
                suma.hora = 1;
            }
            suma.hora = suma.hora + uno.hora + dos.hora;
            if (suma.hora >= 24)
            {
                suma.hora = suma.hora - 24;
            }
            return suma;
        }

        private void txt_yi_TextChanged(object sender, EventArgs e)
        {

        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            //f.ShowDialog(this);
        }

        private void ac_1_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            //Application.OpenForms[this.Tag.ToString()].Focus();
            if (keyboard.Visible == false) keyboard.Show();
            ac_1.Focus();
        }

        private void ac_2_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_2.Focus();
        }
```

```
        private void ac_3_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_3.Focus();
        }

        private void ac_4_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_4.Focus();
        }

        private void ac_5_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_5.Focus();
        }

        private void ac_6_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_6.Focus();
        }

        private void ac_7_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_7.Focus();
        }

        private void ac_8_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            ac_8.Focus();
        }

        private void ac_1_Leave(object sender, EventArgs e)
        {
            if (keyboard.Visible == true) keyboard.Hide();
        }

        private void ac_2_Leave(object sender, EventArgs e)
        {
            if (keyboard.Visible == true) keyboard.Hide();
        }

        private void ac_3_Leave(object sender, EventArgs e)
        {
            if (keyboard.Visible == true) keyboard.Hide();
        }

        private void ac_4_Leave(object sender, EventArgs e)
        {
            if (keyboard.Visible == true) keyboard.Hide();
        }
```

```
private void ac_5_Leave(object sender, EventArgs e)
{
    if (keyboard.Visible == true) keyboard.Hide();
}

private void ac_6_Leave(object sender, EventArgs e)
{
    if (keyboard.Visible == true) keyboard.Hide();
}

private void ac_7_Leave(object sender, EventArgs e)
{
    if (keyboard.Visible == true) keyboard.Hide();
}

private void ac_8_Leave(object sender, EventArgs e)
{
    if (keyboard.Visible == true) keyboard.Hide();
}

private void nud_npr_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_npr.Focus();
}

private void nud_hin_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_hin.Focus();
}

private void hud_min_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_min.Focus();
}

private void nud_sin_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_sin.Focus();
}

private void nud_hpe_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_hpe.Focus();
}

private void nud_mpe_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_mpe.Focus();
}

private void nud_spe_Click(object sender, EventArgs e)
{
    if (keyboard.Visible == false) keyboard.Show();
    nud_spe.Focus();
```

```
        }

        private void nud_mdu_Click(object sender, EventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            nud_mdu.Focus();
        }

        private void nud_sdu_Click(object sender, EventArgs e)
        {
            if (keyboard.Visible == false) keyboard.Show();
            nud_sdu.Focus();
        }

    }
}
```