20

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CENTRO DE DISEÑO Y MANUFACTURA

"CONTROL DIFUSO PARA LAVADO DE ROPA"

TESIS PROFESIONAL

PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

AREA ELECTRICA ELECTRONICA

PRESENTAN:

SERAFIN CASTAÑEDA CEDEÑO

FERNANDO VINICIO MIRANDA CORDERO

DIRECTOR: DR. SAUL D. SANTILLAN GUTIERREZ

MEXICO, D. F. MAYO 2002

PAGINACIÓN DISCONTINUA

Dedico este trabajo:

A ti señor, porque ahora sé que has andado el camino junto conmigo con paciencia, porque estoy buscándote y me permites conocerte cada vez más.

A mis padres Elvia y Octavio por todos los cuidados que me dedicaron, porque me dieron la oportunidad de ser libre y por su fe en convertirme en un hombre de bien.

A Yanet por acompañarme en la vida y llenármela de amor y de alegría, por ayudar a encontrarme a mi mismo, por compartir todas esas experiencias que nos han hecho crecer, y por tu apoyo para construir nuestro propio camino.

A mis abuelos Aurelia y Santos, Enriqueta y Lorenzo por transmitir y procurarnos con cariño una vida mejor para todos nosotros.

A mis hermanos Octavio, Diana, Minerva y Patricia por expresarme su cariño y cuidado en diferentes formas, que me han permitido ser una persona mas sensible, segura y feliz

A Javier (Java) por esa gran amistad que el tiempo a través de los años, en las buenas y en las malas, ha formado y consolidado.

A mis amigos Luis, Mónica y Enrique, Sandra y Miguel, Jorge y Olga por dejarme ser parte de ellos y ellos parte de mí compartiéndome sus experiencias y sus alegrías, pero sobre todo por su amistad incondicional.

A todos los miembros de mi familia con cariño, admiración y respeto.

A todas aquellas personas que me han brindado su ayuda, su confianza, su amistad y su cariño ...

Agradecimientos

Agradezco a Dr. Saúl Gutiérrez Santillán por su colaboración y la confianza que depositó en mi.

A la Mtra. Guadalupe Gonzáles de Ortiz por sus consejos y su apoyo que me sirvieron para seguir adelante.

A mi compañero Serafín Castañeda Cedeño por toda la ayuda para la realización de este gran trabajo.

Al CDM por darnos la oportunidad de desarrollarnos profesionalmente en sus instalaciones.

A la compañía VED por la confianza y visión para el desarrollo de nuevas tecnologías en sus productos.

Fernando Vinicio Miranda Cordero

Este trabajo lo dedico principalmente a:

MIS PADRES por haberme inculcado buenas costumbres y haberme llevado por el mejor camino.

ELSA por el apoyo y cariño durante todo este tiempo.

PABLO JOAQUIN por ser la luz que ilumina mi camino.

A todos ellos GRACIAS.

Serafín Castañeda Cedeño.

Índice

INTRODUCCIÓN.

El rápido avance de la tecnología ha llevado al hombre a generar máquinas que lo ayudan en la industria, en la oficina y en el hogar. Las lavadoras en los últimos años han entrado en casi todos los hogares del mundo, cada vez existen lavadoras más pequeñas, autónomas, versátiles, ergonómicas, silenciosas y que su objetivo principal debería de ser el ahorro de energía y recursos naturales, además de hacerle más cómodo el trabajo al usuario o al ama de casa.

En la actualidad existen grandes proyectos de investigación que están enfocados al cuidado del ecosistema, al ahorro de energía y en especial al ahorro de agua. En todos estos aspectos el proceso de lavado influye nocivamente, y lo peor de todo es que es una tarea que en el transcurso de un solo día se realiza infinidad de veces en todo el mundo.

Muchas compañías están invirtiendo grandes cantidades de dinero en el desarrollo de nuevas máquinas de lavado para poder ganarse un lugar en el gran mercado de enseres domésticos, sin dejar a un lado el aspecto económico y los gustos de los usuarios.

El lavado de ropa es un problema muy complejo, en el cual influyen muchas variables que muchas veces no guardan una relación entre ellas; en contraparte existen otras que mantienen una relación muy estrecha y si cualquiera de ellas cambia las otras podrían modificarse en una proporción muy diferente.

Las técnicas de lavado son variadas pero lo más importante es encontrar una forma de lavar bien la ropa de forma que está no se dañe, en menor tiempo y utilizando la menor cantidad de agua y detergente. La búsqueda de técnicas de lavado no es fácil y muchas empresas han desarrollado algunas formas nuevas de lavado pero todas están basadas en el mismo principio ejercer una acción mecánica sobre las prendas de modo que estás se muevan y mediante la fricción con el agua puedan desprender la suciedad.

Aunque cada persona es diferente, desarrolla hábitos para el lavado de ropa en los cuales utiliza casi siempre el mismo detergente, lava por lo general la misma cantidad de ropa y en algunos casos tienen bien definidos los días y horario de lavado.

El ahorro de agua, energía y detergente ha llevado a construir lavadoras con programas y formas de lavado más sofisticados pero la mayoría de estas lavadoras no se encuentran en el mercado

nacional y las pocas que existen tienen precio alto. Además de ser lavadoras que fueron diseñadas para cubrir necesidades y costumbres de lavado diferentes a las que se tienen en México.
En cada país la gente tiene diferente forma de lavar su ropa, las características del agua son diferentes, el tipo de ropa, la suciedad, además de la composición de los detergentes, el diseñar una lavadora que pueda adaptarse fácilmente a las necesidades y costumbres de lavado sería ideal y tendría grandes ventajas sobre las que hay en el mercado.

Son varias y amplias las áreas que se deben estudiar si se desea un proceso de lavado que se acerque más a lo ideal y adecuado. En especial, el presente trabajo pone especial interés en el ahorro del agua, el ahorro de energía y como consecuencia el cuidado del ambiente.

En este trabajo se pretende desarrollar un sistema el cual sea capaz de poder obtener las variables que más peso tienen en el proceso de lavado y poder obtener una relación clara de ellas. Se hace uso de la teoría de control mediante lógica difusa para poder desarrollar un control apto para tomar decisiones para agregar más detergente, adecuar los tiempos de lavado si es necesario, cambiar el agua de la tina si esta se encuentra muy sucia, etc.

Se instrumentó mediante una tarjeta de control un sistema para poder adquirir y almacenar los datos obtenidos por los sensores y poderlos visualizar en la pantalla de una PC, además de poder tener control sobre los actuadores de la lavadora (motor, bomba, válvulas de admisión, etc.).

Una de las principales tareas de este trabajo es el desarrollo de un software gráfico para poder programar algoritmos de lógica difusa de manera sencilla y transparente, de manera que se puedan probar diferentes ciclos de lavado con diferentes variables y así poder obtener el ciclo más óptimo y adecuado para el lavado de la ropa. Este programa tiene una interfaz serial con la tarjeta de control de manera que la adquisición y manipulación de los datos se realiza en tiempo real.

La forma en que está organizado el trabajo es de la siguiente manera: En el capítulo I se estudian cuales son las necesidades de los usuarios, cuales son las lavadoras que existen actualmente en el mercado, principales características de estas máquinas, se estudian las variables que influyen en el proceso de lavado y cuales son más fácil de controlar, también se plantea el problema y se explican cuales serán los alcances y objetivos del desarrollo de este proyecto.

En el capítulo II se dan algunos puntos que se tomaron en cuenta para la selección de los sensores y actuadores de acuerdo a la variable que se quiera medir o controlar, así como características de estos.

En el capítulo III se explican algunas técnicas de control tradicional que son utilizadas comúnmente y se habla brevemente de las características de la lógica difusa así como de sus ventajas y desventajas.

En el capítulo IV hablaremos del programa de adquisición y manipulación de datos que se utilizó en el proyecto, características de este así como la conexión con los sensores y actuadores.

El diseño del banco de pruebas se explica en el capítulo V poniendo gran énfasis en el software llamado Control Universal Difuso (FUC) de sus siglas en ingles (Fuzzy Universal Control).

El algoritmo de control que se utilizó, así como la descripción de las entradas, salidas y reglas de control difusas, etc.; se presenta en el capítulo VI.

Las pruebas realizadas se presentan en el VII y por último las conclusiones.

El motivo del trabajo es el deseo de hacer una contribución para la solución de uno de los más viejos problemas de la humanidad, el proceso de lavado, por eso que hemos dedicado todos nuestros esfuerzos, pues se pretende con los resultados obtenidos establecer las bases que permitan revolucionar el complicado y hasta ahora inalterable proceso de lavado.

CAPITULO I.

I.1 Estudio del Mercado Actual.

I.1.1 Necesidades de los usuarios.

Cuando una persona adquiere una lavadora, lo único que espera de ella es que limpie su ropa, sin poner mucha atención en su desempeño, no se preguntan por la cantidad de agua necesita, la energía que consume o la cantidad de detergente que consume.

El usuario principalmente pone más atención en la ergonomía de la lavadora y en el precio, buscan aparatos con más durabilidad, fácil de operar y que no necesite de mantenimiento en un largo periodo de tiempo.

Los fabricantes por su parte han invertido muchos recursos en encontrar productos de gran durabilidad, de bajo costo, por lo que queda en gran parte satisfechas las necesidades de los usuarios, pero actualmente es necesario y de gran importancia investigar y desarrollar productos que sean más eficientes en cuanto al ahorro de energía y recursos naturales así como productos que dependan menos del usuario, proporcionándole así mayor comodidad y confort.

Lo anterior implica desarrollar una metodología apropiada para optimizar el uso de agua, detergente y el gasto de energía, para reducir los efectos secundarios que afectan la ecología y minimizar los costos económicos.

Actualmente se les están dedicando más tiempo y recursos, al diseño y desarrollo de enseres domésticos, tal es el caso de la constante aparición de lavadoras y centros de lavado de ropa, así como innumerables electrodomésticos, en los cuales se implementan los últimos avances tecnológicos.

Los avances científicos han fomentado nuevas áreas de desarrollo, tal es el caso de la domótica.

De acuerdo con la empresa *Antwerp Building Consultants* [QUO,2000],"la domótica es el lado amigable del mundo electrónico y permitirá, mediante una computadora central, controlar todos los aparatos a través de una interfaz". Por ejemplo, hoy la lavadora tiene un operador que la programa; en el futuro, la computadora central se encargará de hacerlo por si solo, lo cual abaratará el costo de los aparatos.

I.1.2 Lavadoras comerciales en el mercado.

Desde la década de los 50's abundantes mecanismos del sistema de lavado han sido desarrollados, la gran mayoría de éstos pretendían incorporar sistemas automáticos de control para máquinas que pudieran ser utilizadas fácilmente en los hogares. En la actualidad no sólo se ha logrado hacer de las lavadoras automáticas algo común en los hogares, sino que cada vez los usuarios necesitan menor tiempo y energía para obtener ropa limpia.

I.1.2.1 Clasificación

Existen 2 tipos básicos de lavadoras automáticas [DETERGENTES,1998]: máquinas con agitador y máquinas de tambor. En los Estados Unidos y en el Sur de América y Asia, la de tipo agitador o las máquinas de tipo de impulsión es común hallarlas, con un molde interno para la trasmitir calor, por lo tanto, se puede conectar a una fuente externa de agua caliente si se desea. En contraste, este tipo de máquinas ha perdido virtualmente el significado en Europa, donde fue reemplazada totalmente por la lavadora de tipo tambor.

Máquinas con Agitador. Las lavadoras de Estados Unidos de tipo agitador tienen una tina metálica para lavar la ropa, la cual es usualmente esmaltada. El interior de la tina está hueco y cuenta con un cesto perforado de un tamaño adecuado para almacenar de 5 a 10 kilos de ropa para lavar.

La tina de lavado es llenada con 40, 50, o 60 litros de agua, dependiendo del ciclo de lavado y de la cantidad de carga. El fabricante recomienda introducir el detergente primero, seguido de la ropa a lavar, distribuida lo más uniformemente posible. Sólo hasta entonces se selecciona la cantidad de agua permitida para iniciar el lavado. Las lavadoras de este tipo normalmente tienen 2 conexiones de agua, una para agua caliente y otra para agua fría. La temperatura del agua (caliente, templada, o fría) es seleccionada a conveniencia por las opciones del control, y se lleva a cabo automáticamente mezclando el agua de las 2 entradas. El blanqueador normalmente no está incluido en detergentes para uso en los Estados Unidos y Japón. En su lugar, la solución de hipoclorito de sodio es agregada manualmente antes de terminar el ciclo de lavado, algunas lavadoras son equipadas con un recipiente para este propósito.

Después de que el agua fue introducida, el agitador empieza a funcionar. El brazo agitador tiene un movimiento rítmico por 10 o 15 minutos (*Figura I.1*). Cuando el ciclo de lavado acaba, el agua utilizada se vacía. Las lavadoras americanas que se programan en el ciclo normal empiezan con una serie de giros rápidos y rocían agua para enjuagar, el agua fría es rociada sobre la ropa para quitar el detergente. Posteriormente se empieza a llenar la tina con agua para un breve enjuague (2 minutos aproximadamente), después, una vez más saca el agua y al final da unos giros rápidos para secar la ropa.

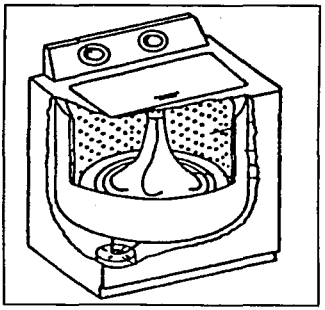


Figura I.1 Lavadora comercial con agitador.

Máquinas de Impulsor. Las lavadoras Japonesas y Europeas de tipo impulsión usan una canastilla montada en un disco rotatorio colocado en la base de la tina o a un costado (Figura I.2). Estas máquinas son invariablemente más pequeñas que su contraparte en los Estados Unidos, tienen un límite de capacidad de 1 a 1.5 Kilos de ropa.

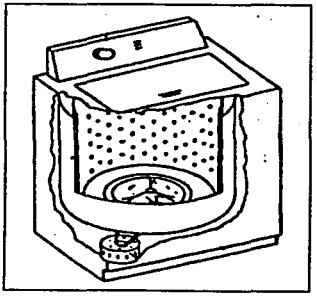


Figura I.2 Lavadora comercial con impulsor.

Máquinas de tambor. En las lavadoras automáticas de tipo tambor, el lavado se realiza en forma horizontal, en un cilindro perforado, el cual gira sobre su eje y en dirección alternativa (Figura I.3). Para lavar algodón normal, solo la tercera parte del cilindro es llenado con agua lo cual significa que, en contraste con las máquinas de agitador, la ropa a lavar no es sumergida totalmente. Las prendas del lavado son repetidamente levantadas por paletas localizadas sobre el borde del cilindro, cada tiempo de caída sirve para frotar y limpiar la carga. El agua se calienta internamente por medio de bobinas eléctricas localizadas en la parte baja de la tina.

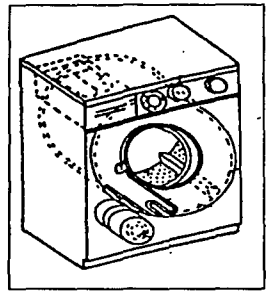


Figura I.3 Lavadora comercial tipo tambor.

Algunas máquinas están equipadas con un abastecedor de agua caliente, esta la obtiene en el caso que se tenga una llave que suministre agua caliente. En este caso la temperatura inicial de la tina es de 30 a 40 °C debido a la introducción del agua fría, de ese modo se elimina de manera más fácil la mancha existente en las prendas. En consecuencia el calor puede ser suministrado si lo requiere el ciclo automático seleccionado. La ropa se introduce por la puerta del lado frontal del cilindro. Las máquinas vistas hasta ahora se les conoce como de carga hacia arriba y de carga frontal respectivamente.

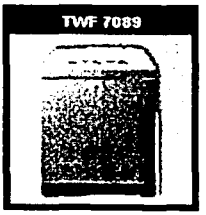
El detergente se introduce en un dosificador integrado. En el de carga frontal, es usualmente un dispositivo localizado en el frente de la máquina. El recipiente para las máquinas de carga hacia arriba generalmente consiste de una serie de compartimentos localizados debajo de la puerta donde se abre la lavadora (arriba). Tales dosificadores permiten secuencias automáticas de lavado que pasan de una a otra (por ejemplo un prelavado y un lavado principal).

Las lavadoras de tambor semiautomáticas son máquinas que sólo realizan el lavado y el enjuague, requiriendo la compra por separado de un secador.

Las lavadoras de tambor completamente automáticas utilizan el mismo tambor para acoplar sucesivamente 3 operaciones: lavado, enjuague y secado. Algunos problemas de balance que ocurrían durante el ciclo de secado requerían que las máquinas construidas antes de 1975 estuvieran firmemente fijas al piso, aunque ahora las maquinas más comunes son las que no requieran estar fijas. Los nuevos modelos están equipados con dispositivos mecánicos especiales que compensan el desbalanceo de la tina.

Principalmente estos tipos de lavadoras son las que más abundan en el mercado nacional, aunque actualmente se han estado introduciendo lavadoras con control electrónico las cuales ya no utilizan alguno de los principios de lavado mencionados anteriormente, ahora se han diseñado lavadoras que utilizan burbujas de aire para desprender la suciedad de las prendas ó chorros de agua. Las figuras siguientes muestran algunos tipos de estas lavadoras Figura I.4:

Marca Singapore Radio and Industy

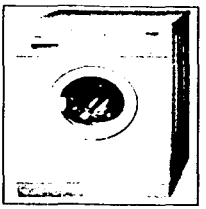


- Lavadora automática de burbujas de aire.
- Capacidad para auto programarse 7 veces.
- Efecto de lavado a mano equivalente a 780 veces
- Medidor de Nivel de Agua
- Utilización de Lógica Difusa

Marca BOSCH

- Utilización de Lógica Difusa
- 16 Programas de Lavado
- Ahorro de energía

Marca Matsushita



- Determina el grado de suciedad
- Utiliza Lógica Difusa
- Control electrónico

Figura I.4 Algunas Lavadoras Comerciales Modernas

Sin embargo, aunque estás lavadoras tienen controles electrónicos, y algunas ahorran energía, ninguna se preocupa por el ahorro de recursos naturales, como el ahorro de agua, la disminución de la cantidad de detergente en cada lavado, la reducción de los tiempos de lavado o la adecuación de los parámetros de lavado para las nuevas condiciones del proceso.

También algunas lavadora tienen incorporada rutinas de control implementadas con Lógica Difusa (Fuzzy Logic), pero estas rutinas son para :

1. El balanceo de la carga que finalmente sirve para la reducción de vibraciones.
2. La detección del grado de suciedad de la ropa para la asignación de un ciclo de lavado.
3. La medición de la cantidad de la ropa.

Ninguna de estas lavadoras tiene la capacidad de modificar sus parámetros de lavado conforme este vaya avanzando, es decir poder controlar los parámetros que más influyen en el lavado y que en gran parte son los más importantes para la adecuada limpieza de la ropa.

I.1.2.2 Parámetros Operacionales

Son 4 los factores que regulan la operación de lavado: química de lavado, entradas mecánicas, temperatura y tiempo de lavado [MINASSIAN,1999].

Los efectos de cada factor sobre los varios funcionamientos de lavado, dependen de las técnicas de lavado utilizadas.

En los días cuando el lavado se hacia en una tina abierta, el requerimiento de agua era más grande, el papel de las acciones mecánicas estaba muy limitado, y el tiempo era un factor muy importante. Actualmente las lavadoras ejercen una mayor acción mecánica: así, el tiempo puede ser disminuido, pero se corre el riesgo de dañar la ropa.

Relación del agua. La relación del agua es proporcional a: la ropa a lavar (seca y en kilogramos), y a la cantidad de agua (en litros). La cantidad de agua requerida en el proceso de lavado es calculada de 2 porciones: La que es absorbida por la ropa y aquella que permanece en exceso.

Varios procesos de lavado tienen diferentes requerimientos de agua:

- La tina de lavado que generalmente necesita por cada 10 litros de agua 1 Kg. de ropa, tiene una relación del agua de 1:10.
- La lavadora de tipo agitador normalmente usa 15 litros de agua por kilogramo de ropa, su relación de agua es de 1:15.
- Una lavadora de impulsión requiere de una relación de agua de 1:20.
- La lavadora automática de tipo tambor se caracteriza por tener baja relación de agua 1:5 con algodón, pero los rangos suben de 1:20 a 1:30 para ropa fina y delicada.

Nivel de agua y Acción Mecánica. La lavadora de tipo tambor está programada para girar en ambos sentidos, horario y antihorario, esto es, primero en una dirección y después en sentido contrario. El tiempo de esta acción mecánica puede ser alterada por la modificación del ritmo de alternancia o del nivel del agua (o ambas). Un nivel de agua bajo (una relación de agua de 1:5) y una acción de reversa relativamente rápida (por ejemplo un ritmo de reversa de 12s de rotación / 4s en pausa) causa un efecto grande de agitación mecánica en la ropa. En contraste, un gran nivel de agua (una relación de 1:20 o 1:30) y una reversa lenta (un ritmo de reversa de 4s de rotación / 12s en pausa) causa una menor acción mecánica, de ahí el término "ciclo suave".

Secado y Residuos de Humedad. Los residuos de humedad son definidos como la cantidad de residuos de agua en la ropa después del secado. Los giros del ciclo de secado con un tambor vertical permiten relativamente rangos altos de rotación: arriba de 2800 revoluciones / minuto. Los niveles de residuos de humedad corresponden de un 40 a un 50% de la capacidad de agua que puede absorber la prenda. En las lavadoras automáticas de tipo tambor los giros de secado alcanzan rangos de 600 a 1200 revoluciones / minuto y los niveles de residuos de humedad están entre 70 y 90%.

Temperatura del agua. En el norte de Europa comúnmente el rango de temperatura de agua utilizada está entre 30 y 80°C, en Estados Unidos la temperatura del agua es de 55°C, en Japón de 25°C y en otros casos excepcionales de 40°C. Las altas temperaturas del agua usadas en Europa están basadas en largas tradiciones y sobre un dicho que dice "sólo con agua caliente la ropa se limpia mejor". Esta actitud esta cambiando como resultado del incremento de la popularidad de la conservación de energía, y la introducción de detergentes efectivos, diseñados para bajas temperaturas.

Tiempo de Lavado. El tiempo de lavado también difiere entre Europa, Estados Unidos y Japón entre otros. Muchas de las diferencias se atribuyen a las diferentes técnicas de lavado: bajas temperaturas del agua en Estados Unidos y Japón comparado con Europa, donde comúnmente el agua se caliente a los 80°C. Cerca de una hora se requiere para obtener los 80°C de temperatura del agua en las máquinas de tipo tambor, sin contar el tiempo que ocupa el tiempo de prelavado. El ciclo de lavado que incluye el prelavado ocasiona 15 minutos más en el proceso. El tiempo requerido para el enjuague y el secado también se suman a la cuenta, por lo que el ciclo total de lavado puede consumir alrededor de 2 horas.

I.1.2.3 Ciclos Y Hábitos de Lavado.

Los ciclos de lavados son las secuencias de operaciones que realizan las lavadoras para poder obtener un lavado de ropa, estas secuencia de operaciones son básicamente tres: lavado, enjuague y centrifugado, aunque algunas lavadoras cuentan con otros ciclos complementarios como es el prelavado que es una agitación pequeña en intervalos de tiempo grande que sirve para poder remover la mugre de manera más rápida en el lavado, además del centrifugado intermedio que ayuda a que las prendas desprender el detergente del lavado, para después pasar al enjuague. Debido a los hábitos de lavado y las diferentes lavadoras consideradas en diferentes partes del mundo, en los siguientes párrafos se discuten los ciclos de lavado en las 3 lugares: en Japón, en Estados Unidos incluyendo México y en Europa.

Hábitos de Lavado en Japón.

Aproximadamente un 75% de las lavadoras encontradas en Japón son de tipo impulsión, el otro 25% están equipadas con tipo agitador. Cerca de las dos terceras partes son variantes de tina gemela, la cual tiene un separador para el ciclo de lavado y otra para el ciclo de secado, con impulsión o con agitador localizados en la parte inferior. Una característica de todas las lavadoras japonesas es el control de temperatura.

Las variables más importantes en las lavadoras japonesas son el nivel de agua y hasta cierto punto las acciones mecánicas. El tiempo de lavado se ajusta al grado de suciedad de la ropa (25-35 minutos). Otras lavadoras de tina gemela ofrecen una selección de los 3 procedimientos básicos de lavado lavado, enjuague y centrifugado.

Programa	Aplicación
Programa Automático	
Suave	Para ropa ligeramente sucia (15 min.)
Normal	Para ropa normalmente sucia (25 min.)
Pesado	Para ropa extremadamente sucia (29 min.)
Programa opcional	
Solo Lavado	Para reuso de agua y detergente si la cantidad de ropa es suficiente para separarla en mas de una sola carga de lavado (9 min.)
Desaguar, Enjuagar, centrifugar	Para enjuagar ropa que anteriormente ha sido lavada (21 min.)
Enjuagar, centrifugar	Para ser utilizado sólo cuando el centrifugado se desea (6 min.)
Selector de Lavado	
Suave	Para ropa fina y delicada
Regular	Para sintéticos normales, mezcla de fibras
Selector de centrifugado	
Ciclo completo	La maquina automáticamente realiza todas las operaciones normales de lavado: lavado, secado, centrifugar.
Sin centrifugar, Secado rápido	

Tabla I.1 Selección de programa de una típica lavadora automática japonesa.

Hábitos de lavado y Lavadoras americanas.

Las lavadoras americanas son parecidas a las japonesas; estas están diseñadas para 2 conexiones de agua, la fría y la caliente. La temperatura del agua manejada es: caliente, tibia y fría. La temperatura del enjuague puede ser seleccionada, usualmente hay 2 opciones, tibia o fría. El nivel del agua se ajusta de acuerdo a la cantidad y tipo de ropa a lavar. Las entradas mecánicas son también ajustables dentro de ciertos límites para introducir cambios en la velocidad del agitador y el tiempo de lavado. Los siguientes ciclos son típicos (Ver Tabla I.2).

Ciclo de lavado	Tipo de tela	Suciedad	T. min. lavado	Temperatura del agua	
				Lavado	Enjuague
Pesado regular	gruesa	alta	10-14	tibia o caliente	Fría
		moderada	8-14	tibia	Fría
		ligera	4-14	fría o tibia	fría
Delicado	sintéticos	ligera	4-8	tibia	fría
	delicados	moderada	4-6	tibia	fría
Planchado permanente	regular	alta	8-10	tibia o caliente	fría
		ligera a moderada	4-10	tibia	fría
Remojo	color	pesado	22	fría o tibia	ninguna
		manchada	22	fría o tibia	ninguna
Prelavado	color	pesado	6	fría o tibia	ninguna
		manchada	6	fría o tibia	ninguna

Tabla I.2.Ciclos típicos de lavado en los Estados Unidos.

Hábitos de lavado en Europa.

Las lavadoras de tipo tambor en las cuales el agua puede ser calentada de 80 a 85 °C dominan en los mercados y en las casas del Noreste de Europa. Para obtener más demandas de lavadoras en las fábricas actuales, las lavadoras tienen una gran variedad de ciclos de lavado.

Se tiene la selección del ciclo apropiado dependiendo de la ropa, selección para decidir si se hace el prelavado o no, entre otras. La construcción de lavadoras para el mercado mediterráneo comúnmente ofrece un compartimiento en el cual permite agregar cloro y otro para el enjuague.

En las lavadoras de la Europa moderna, el sistema de control está basado en la temperatura del agua, de tal modo aseguran que la temperatura deseada es alcanzada en cada etapa de la operación. Así el total del tiempo utilizado para tener agua caliente es variable, y por tanto también el tiempo del ciclo completo.

El desarrollo de los nuevos equipos está caracterizado por la presencia de circuitos electrónicos integrados, algunos son tan complejos que algunos utilizan las microcomputadoras. Estos circuitos realizan funciones muy complejas en la lavadora y en el proceso de lavado.

Las prendas de vestir vendidas en Europa contienen en sus etiquetas el uso de uno de tres niveles de temperatura: 40 °C, 60 °C, y 85 °C.

Los ciclos básicos de una lavadora Europea son: el prelavado, el lavado, el centrifugado intermedio el enjuague y el centrifugado final.

I.1.2.4 Consumo de Energía

La automatización del proceso de lavado tiene mucha relación con el descanso de la persona que va a lavar. Al mismo tiempo, el incremento del costo de energía tuvo mucho que ver en el uso de las lavadoras de tipo tambor en todo el mundo. Debido a la reducción del tiempo de lavado cuando se usa agua caliente.

El factor energía limita la operación de estas lavadoras de tambor, pues teniendo ciclos de lavado con temperaturas de 85°C el consumo de energía es grande y es mayor que usar una temperatura de 60°C.

Algunas lavadoras tienen incorporadas un ciclo de conservación de energía. Lo más usual es que los ciclos estén diseñados para funcionar con 60°C.

I.1.2.5 El mercado de las Lavadoras.

El porcentaje de los hogares equipados con lavadoras varía considerablemente en el noreste de Europa. Los porcentajes estimados son los siguientes:

Austria	81%
Bélgica	86%
República Federal de Alemania	90%
Francia	82%
Gran Bretaña	88%
Italia	87%
Suiza	50%

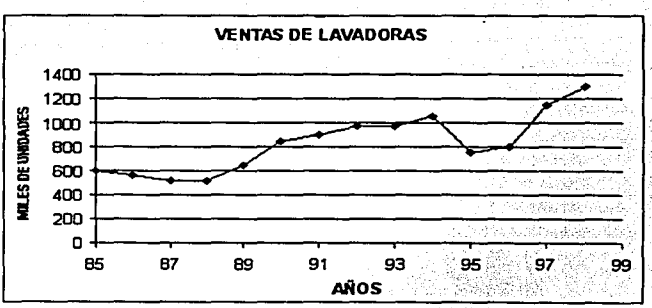
En 1983, 1.6 *10^6 lavadoras fueron producidas en la República Federal de Alemania. De las cuales, 679 000 fueron exportadas, pero 488 000 fueron importadas. El total de las lavadoras de carga frontal fueron un 80% y mientras que el 20% restante es en lavadoras de carga hacia arriba.

En todo el mundo, los Estados Unidos representa una gran porción del mercado de lavadoras. La penetración de las lavadoras en el hogar de los Estados Unidos es muy baja comparada con los países de Europa y Japón. Sólo el 74% de los hogares estadounidenses cuentan con lavadoras, esto es hasta 1984. En 1990 los estados Unidos fueron responsables de la manufactura de 5.05 * 10^6 unidades.

Virtualmente cada hogar cuenta con una lavadora en Japón. Los japoneses manufacturaron 4.72 * 10^6 unidades en 1982. De los cuales, 922 000 fueron lavadoras totalmente automáticas, y el 3.8 * 10^6 fueron del tipo simple. Los japoneses exportaron en 1982 1.3 * 10^5 unidades a U.S.A, y Europa, más las que fueron destinadas a los países de Asia.

Según la Asociación Nacional de Fabricantes de Aparatos Domésticos, A.C. en 1996 en México exportaron lavadoras de ropa con valor de 57.2 millones de dólares y se importaron 20.6 millones de dólares en lavadoras [LUNA,2000], cabe señalar que las lavadoras son los aparatos electrodomésticos que más se importan en México.

En la gráfica I.1 se muestran las ventas de lavadoras en México de 1985 a 1998, con lo que se ve una tendencia de incremento en los últimos 2 años. [ANFAD,1996].



Gráfica I.1 Ventas de lavadores en los últimos años en México.

I.2 Estudio del Problema de Lavado.

El lavado de ropa es un procedimiento complejo el cual involucra numerosos factores físicos y químicos. El proceso de lavado puede ser definido como la acción de remover residuos e impurezas que se encuentran en las prendas.

Existen cuatro factores principales que intervienen en el proceso de lavado: factor químico (detergente), factor mecánico (provisto al tallar la ropa), factor térmico (temperatura del agua) y tiempo de lavado. Una combinación correcta de estos factores da como resultado una mejor limpieza de la ropa, sin dañarla, el ahorro de agua, logrando así menores tiempos de lavado lo que implica un ahorro de energía y de recursos naturales [Minassian, 1999].

Para determinar si una prenda ha quedado completamente limpia, o para determinar que tan eficiente ha sido el ciclo de lavado se realiza una serie de pruebas de laboratorio, sin embargo el usuario no puede realizarlas en casa.

El agua toma un papel muy importante en el lavado de la ropa, ya que sirve como solvente para el detergente y como medio de transporte para disolver y dispersar la mugre. Una cantidad correcta del nivel del agua ayuda a poder obtener mejores resultados en el lavado. De tal forma que una cantidad de agua menor impide la circulación libre de esta a través de la ropa y una cantidad mayor hace que la ropa no tenga suficiente fricción y no se limpie correctamente.

El detergente tiene gran influencia en el lavado de la ropa, estos son compuestos orgánicos sintéticos los que tienen por objeto desprender la mugre de las telas sin dañarlas. Los detergentes pueden ser clasificados como de líquidos o granulados, de alta espuma o espuma controlada. Esta acción puede llevarse a cabo por procesos químicos (destrucción de los enlaces covalentes) o debilitando las uniones físicas para remover la mugre y disolverla en el baño.

La absorción del detergente en las superficies sólidas permite la humectación de la superficie por el agua, y hace que las partículas de suciedad se desprendan, formando pequeñas gotas. La agitación del baño provoca que dichas gotas y la suciedad adherida se eliminen, quedando la superficie limpia.

Si el detergente hace espuma, la suciedad puede desprenderse más fácilmente cuando el sistema se agita, debido a que se divide en películas finas en la superficie de las burbujas, sin embargo existen muchos detergentes no espumosos que presentan ventajas en cuanto a la protección de los equipos y del medio ambiente.

La cantidad de detergente que es necesario agregar al baño depende de los siguientes factores:

- Dureza del agua: contenido de minerales
- Grado de suciedad de la ropa
- Tamaño de la carga: cantidad de ropa
- Tipo de detergente: con fosfatos, sin fosfatos, líquido, jabones
- Temperatura de lavado.

La cantidad de agua, tiempo de lavado, frecuencia de agitación y temperatura del agua, son algunos de los parámetros que hay que controlar para poder tener un lavado adecuado, estos parámetros dependen de las siguientes variables físicas :

Naturaleza de la mugre. Existen diferentes tipos de mugre, dependiendo de su naturaleza :

- Materiales Solubles al agua : Sales inorgánicas, azúcar, transpiración.
- Pigmentos : Oxidos metálicos, carbonatos, silicatos, carbón.
- Grasas : Grasa animal, grasa vegetal, sebo, aceite, cera.
- Proteínas : Sangre, huevo, leche.
- Residuos naturales: Frutas, vegetales, vino, café, té.

Naturaleza del substrato. La tela se puede agrupar en diferentes tipos, dependiendo del origen de la fibra, composición y tipo de tejido:

- Textiles Básicos: Algodón, Lino, Seda y combinaciones de estas.
- Sintéticos : Rayón, nylon.

La composición del baño. El baño se conoce como la mezcla de agua, detergente y otros productos de limpieza.

Las condiciones físicas y mecánicas del lavado. La temperatura, tipo y tiempo que se aplica la acción mecánica.

La cantidad de mugre, ropa y baño. El aplicar el detergente necesario para que se disuelva completamente y para que la ropa se limpie es muy importante por lo que estos factores influyen en la cantidad que hay que aplicar de este.

II.2.1 Características a considerar durante el lavado de Textiles

- La ropa de algodón blanco y de color, se lava comúnmente con un detergente duro, un ciclo de lavado normal, a una temperatura de 80 ° C, con una frecuencia de agitación baja.
- La ropa de algodón que no incluye colores es lavada usualmente de 40 - 60 ° C. La lana y seda se lavan en condiciones especiales.
- Para obtener un mejor lavado de la lana y seda, estas deberán someterse a una temperatura baja (máximo 30 ° C), usando una frecuencia de agitación alta, y usar un detergente blando.
- Las fibras sintéticas basadas en celulosa, como el rayón y la lana artificial, requieren una temperatura de lavado de 60 ° C, pero si tiene colores, la temperatura tendrá que ser reducida a 40 ° C, se recomienda que la agitación sea alta.
- Las prendas de poliéster blanco son lavadas de 40 - 60 ° C , y las de color a una temperatura de 40 ° C. Las fibras sintéticas inorgánicas, por ejemplo la fibra de vidrio , requieren una mayor frecuencia de agitación y una temperatura de 30°C.

FIBRAS	BLANCO		COLOR		COLOR Y BLANCO	
	Temp.° C	Agitación	Temp.° C	Agitación	Temp.° C	Agitación
Fibras Naturales						
Algodón	85	baja	40, 60 ó 80	Baja	85 ó 60	baja ó alta
Lino	85	baja	40, 60, ó 80	Baja	85 ó 60	baja ó alta
Lana	fría, < 30	alta	fría, < 30	Alta	fría, < 30	alta
Seda	fría, < 30	alta	fría, < 30	Alta	fría, < 30	alta
Fibras Químicas	(celulosas)					
Rayón	60	alta	60 ó 40	Alta	60 ó 40	alta
Acetato	40	alta	40	Alta	40	alta
Fibras Químicas	(sintéticas)					
Poliéster	30 - 60	alta	30 ó 40	Alta	30 ó 40	alta
Poliuretano	60	alta	40 - 60	Alta	40 - 60	alta
Polyvinilico	30	alta	30	Alta	30	alta

Agitación. Baja : frecuencia 1: 5 y entrada mecánica normal; alta : frecuencia 1:20 a 1:30 y decremento

Tabla I.3 Características para el lavado de Textiles.

La cantidad de agua requerida en una máquina de lavado, para obtener una adecuada operación depende de la cantidad de carga y del tipo de ropa que se tenga. Por ejemplo si se pone una gran cantidad de ropa, por consiguiente se debe de tener el suficiente nivel de agua para que todas las prendas queden completamente sumergidas. También influye la frecuencia de agitación, ya que si se tiene una agitación alta, generalmente se tiene un ciclo de lavado para ropas fuertes y la cantidad de agua requerida es menor. Por el contrario si la frecuencia de agitación es menor, se trata de prendas más delicadas, y hay que suministrar un nivel de agua mayor.

Por lo regular, la cantidad de ropa en una lavadora es determinada visualmente por el usuario, el cual selecciona el nivel de carga que cree el correcto para el lavado de la ropa. Actualmente se han desarrollado sistemas que determinan automáticamente la cantidad de ropa y por lo tanto el nivel de agua necesario para obtener un lavado adecuado.

Otro parámetro que hay que tener en cuenta, es el tiempo de lavado. Este depende principalmente del grado y tipo de suciedad. El tiempo de lavado es la duración de la acción mecánica en las prendas, y es importante, ya que si no es suficiente las prendas no quedarán limpias, mientras que si es mayor al necesario, se esta desperdiciando energía.

Como podemos observar existen relaciones entre cada una de las variables del proceso de lavado, algunas de estas variables son definidas por el usuario de manera intuitiva pero muchas veces no es realmente lo óptimo para obtener un lavado correcto.

En la Figura I.5 se presenta un diagrama en el cual se muestra la relación que guardan cada uno de las variables del proceso de lavado con los parámetros del proceso.

Existen variables como la cantidad de detergente que se le agrega a la ropa, la cual depende de la Turbiedad, Temperatura del Agua, Grado de suciedad y Cantidad de Ropa; una vez agregado el detergente podemos estar midiendo este mediante las variaciones entre la Conductividad y Turbiedad del Agua. Estas mediciones nos ayudaran a determinar si es necesario agregar más detergente, o incrementar o disminuir el tiempo de lavado. Otros parámetros por ejemplo, como la Cantidad de Agua y la Frecuencia de Agitación se determinan desde el inicio del proceso, y no se modifican durante este avanzando.

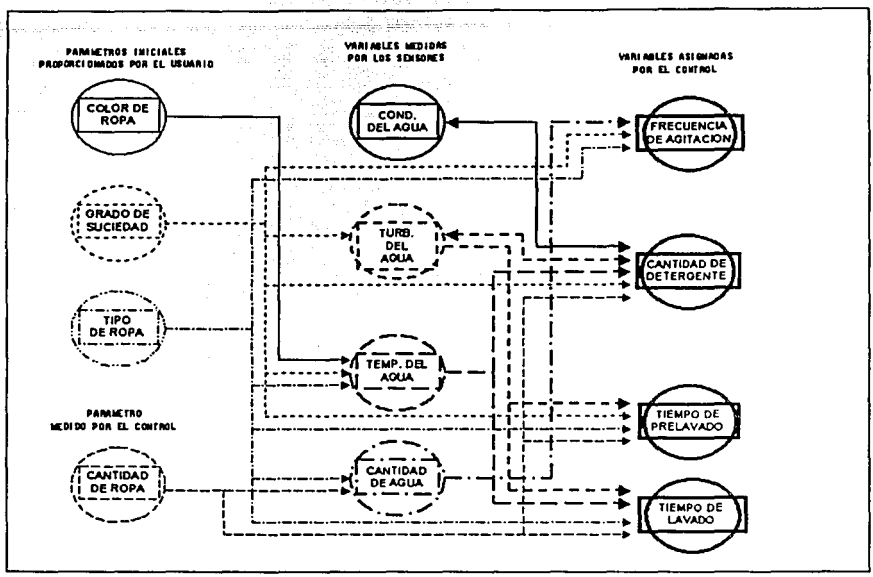


Figura I.5 Relación entre las variable y los parámetros del Proceso de Lavado.

La relación de cada una de estás variables en algunos casos es muy sencilla, por ejemplo la cantidad de agua que se asigna a la lavadora esta relacionada con el tipo y cantidad de ropa. La relación entre ellas es de manera lineal, esto es, si aumentamos la cantidad de ropa, tenemos que aumentar lógicamente la cantidad de agua, y si el tipo de ropa cambia de una que absorbe mayor cantidad de agua a una que no lo hace, tenemos que tener más agua en la tina para que no se afecten las condiciones de lavado. Pero por ejemplo, la cantidad de detergente es una variable que depende de la cantidad y suciedad de la ropa, y este también se relaciona con la turbiedad y conductividad del agua. Como se puede observar en estos dos ejemplos las relaciones pueden ser muy complejas.

En la Tabla I.4 se resumen los pesos o la fuerza con la que estas relaciones están ligadas, y como afecta la variación de las variables con los parámetros de lavado.

VARIABLE	RANGO / DIVISION	COLOR DE ROPA	TIPO DE ROPA	GRADO DE SUCIEDAD	CANTIDAD DE ROPA	CONDUCTIVIDAD DEL AGUA	TURBIEDAD DEL AGUA	TEMPERATURA DEL AGUA	CANTIDAD DE AGUA	FRECUENCIA DE AGITACIÓN	CANTIDAD DE DETERGENTE	TIEMPO DE PRELAVADO	TIEMPO DE LAVADO
COLOR DE ROPA	*BLANCO COLOR*							ALTA BAJA					
TIPO DE ROPA	*ALGODÓN DELICADOS SINTÉTICOS*							ALTA MEDIA BAJA	POCA REGULAR MUCHA	RAPIDA MEDIA BAJA		GRANDE MEDIO BAJO	GRANDE MEDIO BAJO
GRADO DE SUCIEDAD	*MUY SUCIA SUCIA POCO SUCIA*						MUY TURBIA TURBIA POCO TURBIA	ALTA MEDIA BAJA		RAPIDA MEDIA BAJA	MUCHO MEDIO BAJO	GRANDE MEDIO BAJO	GRANDE MEDIO BAJO
CANTIDAD DE ROPA	*0-7 Kgs POCA CARGA MEDIA CARGA CARGA COMPLETA*								POCA REGULAR MUCHA		MUCHO MEDIO BAJO	GRANDE MEDIO BAJO	GRANDE MEDIO BAJO
CONDUCTIVIDAD DEL AGUA	*0-14 Msiemens ALTA MEDIA BAJA*										MUCHO MEDIO BAJO		
TURBIEDAD DEL AGUA	*0-3500 NTU MUY TURBIA TURBIA POCO TURBIA*			MUY TURBIA TURBIA POCO TURBIA							MUCHO MEDIO BAJO	GRANDE MEDIO BAJO	GRANDE MEDIO BAJO
TEMPERATURA DEL AGUA	*0-60 °C ALTA MEDIA BAJA*	ALTA BAJA	ALTA MEDIA BAJA	ALTA MEDIA BAJA							MUCHO MEDIO BAJO		
CANTIDAD DE AGUA	*20-40 cms*		POCA REGULAR MUCHA		POCA REGULAR MUCHA					RAPIDA MEDIA BAJA			
FRECUENCIA DE AGITACIÓN	*21-77 osc/mn*		RAPIDA MEDIA BAJA	RAPIDA MEDIA BAJA					RAPIDA MEDIA BAJA				
CANTIDAD DE DETERGENTE	*45-121 grs*			MUCHO MEDIO BAJO	MUCHO MEDIO BAJO	MUCHO MEDIO BAJO	MUCHO MEDIO BAJO	MUCHO MEDIO BAJO					
TIEMPO DE PRELAVADO	*20-100 mn*		GRANDE MEDIO BAJO	GRANDE MEDIO BAJO	GRANDE MEDIO BAJO		GRANDE MEDIO BAJO						
TIEMPO DE LAVADO	*3-13.5 mn*		GRANDE MEDIO BAJO	GRANDE MEDIO BAJO	GRANDE MEDIO BAJO		GRANDE MEDIO BAJO						

Figura I.4 .Relaciones de peso en cada una de las relaciones del proceso de Lavado.

Una vez iniciado el proceso de lavado, se leen los parámetros proporcionados por el usuario: *Color de Ropa, Grado de Suciedad y Tipo de Ropa*, se mide la *Cantidad de Ropa* y se asignan condiciones iniciales de lavado que llamaremos primarias: *Cantidad de Agua*, y *Frecuencia de Agitación*, que son las que no se modifican durante el proceso, además de las variables secundarias como *Temperatura del Agua*,. *Cantidad de Detergente* y se asignan tiempos máximos de *Prelavado y Lavado*, cuando el proceso comienza, solamente se leerán a través de los sensores las variables llamadas terciarias: *Conductividad, Turbiedad y Temperatura*, y se podrán ir modificando las variables Secundarias dependiendo de los valores de cada una de estas .

1.3 Presentación del Problema

A pesar de todos los avances de la tecnología, y el desarrollo de nuevas máquinas el lavado de ropa no ha sufrido cambios importantes, el principio sigue siendo el mismo, la fricción generada por la acción mecánica de un agitador, sobre la ropa sumergida en agua con detergente, produce un desprendimiento de la suciedad adherida y su disolución en el baño.

¿Por qué considerar el proceso del lavado como un problema? ¿Por que preocuparse por un proceso de lavado que casi no ha sufrido grandes cambios desde su origen, y que aún hoy en día es tan popular?

Una de las respuestas es que el proceso de lavado no es un proceso óptimo. Con esta respuesta tal vez surja otra pregunta: ¿existen los procesos óptimos? Obviamente no, pero los procesos de hoy son bastante deficientes en cuanto al ahorro de agua y de energía se refieren. Además, si no existieran más y mejores formas de llevar a cabo la tarea de lavado de la ropa, seguramente esto dejaría de ser un problema.

Si bien es cierto que existen varios adelantos en distintas disciplinas que involucran al proceso de lavado y también es cierto que es un problema de interés mundial. ¿Porque no se ha hecho algo para mejorarlo?

Es evidente que se necesita urgentemente un cambio en el proceso. Sin embargo, para poder llevar a cabo un cambio de esta magnitud deben de cumplirse tres aspectos: querer, poder y saber.

El rechazo a lo nuevo y a lo desconocido provoca que la gente prefiera dejarse llevar por sus hábitos y sus costumbres, evitando así cualquier intento de cambio. En otros casos el rechazo se debe a la idea que una acción mecánica tiene mejores resultados, debido a que se parece más a la forma "natural" y tiene más "sentido".

Como consecuencia de esto, las compañías se ven obligadas a abandonar un proyecto que no tiene mercado. En otras ocasiones este abandono puede deberse a los altos costos que implica un nuevo tipo de lavado.

Los avances en la electrónica, sensores, actuadores, la computación y el desarrollo de la Inteligencia Artificial, así como también en la química de detergentes, de blanqueadores y suavizantes, nos acercan a la posibilidad de realizar parte de ese cambio.

Si tomamos en cuenta las preferencias de la gente y sacamos ventaja de los beneficios que ofrecen las nuevas tecnologías, en el diseño de un nuevo proceso de lavado, sólo nos resta reconocer si disponemos del conocimiento necesario y suficiente para poder realizar un cambio integral.

Es debido en parte, desafortunadamente, a la carencia de información de cómo se relacionan las diferentes variables involucradas en el proceso, lo que finalmente ha detenido la evolución del proceso de lavado.

Esto se debe a que los parámetros en que se mide la eficiencia del ciclo de lavado, tienen un origen muy difícil de implementar en un modelo matemático, esto es, que los parámetros están más asociados con el sentido común de la gente que con un estricto pensamiento matemático. Más aun, la experiencia acumulada, registrada y estudiada es bastante difícil de integrar en un proceso tan complejo.

Por ello, se han creado formas alternas o indirectas para medir la eficiencia de los ciclos actuales de lavado.

Por otro lado, debido a que la efectividad del proceso es juzgada por el sentido común, y en esto hay diversas opiniones, se ha tenido que llegar a una normalización o generalización, forzando a que las apreciaciones personales desaparezcan o se unifiquen, con el fin de llevar un control de calidad internacional.

Los ingenieros están tratando de desarrollar lavadoras con las siguientes tendencias: ahorro de consumo de recursos, mayor número de programas o adecuación de los programas de lavado de acuerdo a las condiciones de la carga, operación más silenciosa y con menos vibraciones, mayor cuidado de la ropa, materiales más económicos, resistentes al desgaste y a la corrosión.

Se ha detectado que el camino más rentable es realizar un **control más inteligente**: poder determinar automáticamente la cantidad de ropa, grado de suciedad, suministro de detergente necesario para el lavado, cantidad de agua suficiente y adecuar el proceso de lavado debido a las nuevas condiciones que se estén presentando en la lavadora, esto es por ejemplo, si el agua se encuentra muy sucia entonces agregar más agua manteniendo la temperatura asignada, o si hace falta detergente intercambiar el agua para que no se sature, además de ir adecuando el tiempo de lavado con esto queremos decir que si la ropa no esta soltando más mugre entonces no es necesario seguir agitando por lo que el conveniente para el proceso de lavado.

En las lavadoras actuales el usuario selecciona la cantidad, de ropa, tipo y grado de suciedad que cree conveniente y el control electrónico mediante ecuaciones paramétricas matemáticas es capaz de calcular el nivel de agua, cantidad de detergente, temperatura del agua, tiempo y frecuencia de agitación, etc. [Minassian, 1998]

Los problemas que observamos en todas las lavadoras, es que los ciclos de lavado son fijos, esto es, el tiempo de lavado, la cantidad de detergente, la temperatura, la cantidad de agua etc., son valores que se determinan cuando se pone a trabajar la lavadora pero que no se ajustan mientras el proceso de lavado esta avanzando. Estos muchas veces influyen a que la ropa no quede completamente limpia o que estemos desperdiciando insumos.

Las relaciones que existen entre cada una de las variables que influyen en el proceso de lavado son difíciles de determinar, por lo que el desarrollo de una máquina que permita realizar pruebas cambiando las variables que influyen en el lavado, desarrollar nuevas técnicas y modos de agitación, probar partes mecánicas, ayudará a los ingenieros a poder identificar las variables más importantes que hay que controlar, saber como se comporta la lavadora a diferentes condiciones y desarrollar partes mecánicas más adecuadas para las nuevas lavadoras.

1.4 Objetivos Generales.

Este sistema es la primera parte de un proyecto que se está realizando en el CDM, en el cual se pretende construir un sistema en el que se puedan aplicar los conocimientos obtenidos y como resultado obtener los siguiente:

a) Obtener una base de conocimiento mínima que permita establecer las relaciones entre algunas de las variables involucradas en el proceso.

El sistema que se pretende desarrollar tiene que ser capaz de adquirir, analizar y almacenar grandes cantidades de datos provenientes de los sensores y controlar todos los actuadores de forma precisa y confiable.

Obtener parte de la base del conocimiento significa identificar las relaciones que guardan cada una de las variables que se van a medir (temperatura, turbiedad, conductividad), con las variables a controlar (temperatura, nivel de agua, cantidad de detergente, tiempo y frecuencia de agitación, etc.).

El poder obtener está base del conocimiento ayudará en un futuro a desarrollar nuevos ciclos de lavado (cambiar la frecuencia de agitación y modo de agitación, cambiar nivel del agua, girar la tina, inyectar burbujas de aire, inyectar agua, etc.) y no solamente los comúnmente utilizados (movimiento por impulsor o por agitador), poder monitorear las partes mecánicas (motor, tina, aspas, etc.), y medir variables eléctricas (consumo de energía, potencia) que ayuden al diseño y

construcción de lavadoras más robustas, de bajo costo y que se adapten a las necesidades que los usuarios tienen hoy en día.

b) Poder optimizar los insumos.

Este conocimiento obtenido servirá para poder implementar un algoritmo de control mediante lógica difusa y probar su funcionamiento hasta poder obtener un ciclo de lavado óptimo y más adecuado dependiendo de las características de la carga que se tenga.

c) Crear un sistema de control que permita experimentar de manera fácil con actuales y nuevos procesos de lavado.

Lo que se propone en este trabajo es implementar lo aquí desarrollado en una lavadora comercial, de eje vertical de 7 kgs. de capacidad, incorporando los sensores y actuadores necesarios de forma modular, esto es que sea fácil poder incorporárselos a otras lavadores del mismo modelo.

El poder desarrollar un sistema en el cual se puedan programar de manera fácil algoritmos de control difuso, es otra de las tareas que nos proponemos realizar en este trabajo.

Se pretende también probar el funcionamiento de este sistema así como el del control, y si es posible el validar los resultados comparando con el control de una lavadora comercial, o con pruebas realizadas por los expertos en el diseño de lavadoras.

Las variables que se van ha medir son las siguientes:

- Cantidad de Agua
- Temperatura del Agua
- Conductividad del Agua
- Turbiedad del Agua
- Cantidad de ropa

Lo importante de cada una de estas mediciones es que podamos encontrar las relaciones más adecuadas e importantes que existen entre cada una de ellas y como afectan al proceso de lavado, es decir que tanta información se puede obtener de la turbiedad, la conductividad y la temperatura del agua y poderla relacionar con el grado de suciedad de la ropa, cantidad de detergente y los tiempos de lavado.

CAPITULO II.
CARACTERÍSTICAS DE LOS SENSORES Y ACTUADORES.

II.1 Introducción.

Se denomina transductor a todo dispositivo que convierte un señal de una forma física en una señal correspondiente pero de otra forma física distinta. Por lo tanto podemos decir que se trata de un convertidor de energía [BALCELLS,1998].

Existen 6 tipos de señales: mecánicas, térmicas, magnéticas, eléctricas, ópticas y moleculares (químicas).

En la práctica, no obstante, se consideran transductores aquellos que ofrecen una señal de salida eléctrica. Ello se debe a que la mayoría de las mediciones son conectados a sistemas electrónicos para su procesamiento.

Existen varias ventajas por las que se utilizan sistemas de medición electrónica:

1. Debido a la estructura electrónica de la materia, cualquier variación de un parámetro no eléctrico de un material viene acompañada por la variación de un parámetro eléctrico. Eligiendo el material adecuado, esto permite realizar transductores con salida eléctrica para cualquier magnitud física no eléctrica.
2. Dado que en el proceso de medida no conviene extraer energía del sistema donde se mide, lo mejor es amplificar la señal de salida del transductor.
3. Además de la amplificación, hay una gran variedad de recursos, en forma de circuitos integrados, para acondicionar o modificar las señales eléctricas. Incluso hay transductores que incorporan físicamente en un mismo encapsulado parte de estos recursos.
4. Existen también numerosos recursos para presentar o registrar información si se hace electrónicamente, pudiéndose manejar no sólo datos numéricos, sino también textos, gráficos y diagramas.
5. La transmisión de señales eléctricas es más versátil que las de señales mecánicas, hidráulicas o neumáticas, y si bien no hay que olvidar que éstas pueden ser la presencia de radiaciones ionizantes o atmósferas explosivas, en muchos casos estos sistemas han sido sustituidos por otros eléctricas.

Un sensor es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida que esta función de la variable medida [CREUS,1998].

Sensor y transductor se emplean a veces como sinónimos, pero sensor sugiere un significado más extenso: la ampliación de los sentidos para adquirir un conocimiento de cantidades físicas que, por su naturaleza o tamaño, no pueden ser percibidas directamente por los sentidos. Transductor, en cambio, sugiere que la señal de entrada y la de salida no deben ser homogéneas.

La tendencia actual, particularmente en robótica, es emplear el término sensor para designar el transductor de salida. Los primeros pretenden la obtención de información, mientras que los segundos buscan la conversión de energía.

En ese trabajo utilizamos el término sensor para referirnos a los traductores de entrada. No se trata de los accionamientos o transductores de salida.

Para la selección de los sensores, se tomaron en cuenta las siguientes características:

1. *Campo de medida*: Rango de los valores de la magnitud de entrada comprendido entre el máximo y el mínimo detectable por el sensor, con una tolerancia de error aceptable.
2. *Resolución*: Capacidad del sensor para discernir entre valores de la variable de entrada.
3. Precisión: La máxima desviación entre la salida real obtenida de un sensor en determinadas condiciones de entorno y el valor teórico de dicha salida, en idénticas condiciones, según el modelo ideal especificado como patrón.
4. *Repetibilidad*: Característica que indica la máxima desviación entre los valores de salida obtenidos al medir varias veces un mismo valor de entrada.
5. *Linealidad*: Un transductor es lineal, si existe una constante de proporcionalidad que relaciona los incrementos de señal de salida con los correspondientes incrementos de la señal de entrada.
6. *Sensibilidad*: Característica que indica la mayor o menor variación de la salida por unidad de la magnitud de la entrada
7. *Ruido*: Se trata que el transductor sea lo menos susceptible perturbaciones de la señal de entrada.
8. *Velocidad de respuesta*: Mide la capacidad de un transductor para que la señal de salida siga sin retraso las variaciones de la señal de entrada.
9. *Tipo de salida* : Corriente, voltaje ya sea en C.D. o C.A., salida digital, etc.
10. *Alimentación*: Niveles de voltaje y de corriente requeridos para que el sensor trabaje en forma adecuada, si es necesario.

La instrumentación de los sensores y el control de los actuadores se hará en una lavadora con agitador de 7 Kg. de ropa de capacidad.

II.1.1 Descripción de la lavadora

Una división por sistemas más adecuada de este tipo de lavadoras es la siguiente [MINASSIAN,1999] :

Sistema envolvente: comprende las partes que dan cuerpo a la lavadora, y que sirven como soporte del resto de los sistemas. Las partes que lo componen son:

- Páneles laterales de lámina de acero
- Cubierta superior también de lámina de acero
- Panel de control (copete) de lámina de aluminio
- Patas con tornillo para nivelación
- Cable de conexión para el suministro de energía eléctrica
- Mangueras de entrada de agua caliente y fría
- Conexión para la manguera de desagüe
- Manguera de desagüe de la tina a la bomba
- Orificio de entrada para dosificar el blanqueador
- Perilla del control principal
- Perilla del control del nivel del agua.

Cabe mencionar que para poder realizar las pruebas necesarias fue necesario hacerle modificaciones como son: Eliminación de las perillas de control principal y nivel de agua, ya que estos parámetros serán definidos y controlados por el algoritmo de control difuso.

Sistema de función primaria: es el agregado propiamente de producir la acción mecánica necesaria para el lavado. Esta compuesto por:

- Motor eléctrico de CA: motor de inducción de ¼ Hp, monofásico
- Capacitor de arranque del motor : 45 μF, 220/250 Volts
- Solenoide de resorte
- Transmisión:
    - Poleas y banda
    - Engranaje planetario

  - Freno de banda y tambor
  - Flecha de agitador
  - Flecha de la tina de centrifugado
- Agitador: de plástico inyectado, con aspas rectas rígidas
- Tina de centrifugado (interior, móvil): lámina de acero esmaltada y perforada
- Tina de lavado (exterior, fija): plástico con orificios para la entrada de blanqueador, del sensor de presión y salida para el desagüe.

Sistema de funciones secundarias: las funciones secundarias son aquellas no indispensables para llevar a cabo el proceso de lavado, pero que contribuyen a simplificar la operación al usuario. La lavadora que se esta utilizando, cuenta con los siguiente elementos auxiliares:

- Anillo de plástico relleno de agua para el balance de la tina
- Suspensión a base de resortes / amortiguadores
- Filtro quitapelusa
- Depósito de blanqueador
- Válvulas solenoide para alimentación de agua caliente y fría
- Bomba de desagüe.
- Características: bomba centrífuga, álabes rectos, con impulsor de plástico y motor independiente de 127 Volts, 105 W.
- Tina de centrifugado.

Sistema de control: es el alma de la lavadora, encargado de coordinar la secuencia de operaciones tanto del sistema primario como de los sistemas de funciones secundarias, el sistema de control original de la lavadora está compuesto por:

- Temporizador electromecánico (Figura II.1):
  - Perilla
  - Mecanismos de trinquete (no retorno)
  - 8 levas de eje común
  - 8 interruptores, que actúan como seguidores de levas
  - Puertos de conexión de los interruptores hacia los actuadores
  - Motor de CA de 3 Watts con reductor de velocidad (tren de engranes), acoplado al juego de levas

- Conexión con alimentación de electricidad, válvulas de admisión de agua, circuito oscilador, bomba de desagüe, Presostato, interruptor de seguridad.

- Interruptor de seguridad de la tapa
- Circuito electrónico oscilador (es el encargado de dar los pulsos eléctricos al motor para que realice la función de agitación)
- Presostato: dispositivo que mide el nivel del agua en la tina y el cual tiene tres posiciones, abierto, medio nivel y nivel completo.

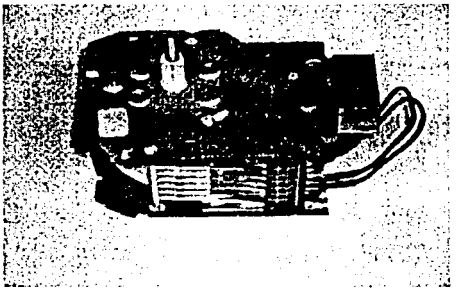


Fig. II.1 Temporizador electromecánico

El sistema de control será remplazado por un tarjeta electrónica en base al microcontrolador COP8 el cual servirá como interfaz para poder encender y apagar los actuadores y poder leer el valor de los sensores que se pondrán la lavadora proporcionada para el desarrollo del banco de pruebas el cual será explicado más adelante en este trabajo.

En la figura II.2 se muestran las partes principales de una lavadora con agitador y en donde se localizan cada una de las partes que en los párrafos anteriores se describieron.

Los acondicionadores de señal, adaptadores o amplificadores, en sentido amplio, son los elementos del sistema de medida que ofrecen, a partir de la señal de salida de un sensor electrónico, una señal apta para ser presentada o registrada o que simplemente permita un procesamiento posterior mediante un equipo o instrumento estándar. Consiste en circuitos electrónicos que ofrecen entre otras funciones: amplificación, filtrado, adaptación de impedancias y modulación o demodulación.

Si se considera por ejemplo, el caso en que unas de las etapas de tratamiento de la señal de medida es digital, si la salida del sensor es analógica, que es lo más frecuente, hará falta un

convertidor A/D. Éstos tienen una impedancia de entrada limitada, exigen que la señal aplicada sea continua o de frecuencia de variación lenta, y que su amplitud esté entre unos límites determinados, que no suelen exceder de 5 Volts. Todas estas exigencias obligan a colocar un acondicionador de señal entre el sensor y el convertidor A/D ya que muchas veces ofrece señales apenas de unos milivolts.

Los sensores se clasifican en analógicos o digitales. En los analógicos la señal de salida varía, a nivel macroscópico, de forma continua. La información está en la amplitud. En los sensores digitales, la salida varia en forma de saltos o pasos discretos. No requieren conversión A/D y la transmisión de su salida es más fácil, en algunos casos se utiliza una comunicación serial.

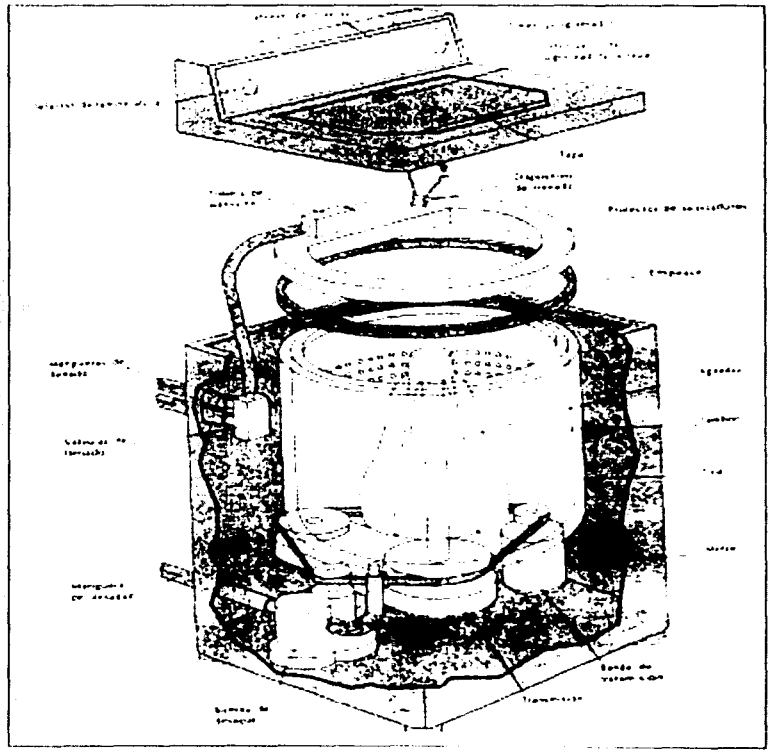


Fig. II 2 Partes principales de una lavadora con agitador.

II.2 Selección de los sensores.

De acuerdo a las variables que se van a medir que se presentaron en el tema I.3 de este trabajo, se realizó la tabla II.1 para poder seleccionar el sensor más adecuado.

Aquí se muestran cada una de las variables a medir, rango y unidades de medida, se presentan también el principio por el cual se puede hacer la medición de dicha señal.

	Variable	Rango	Unidades	1	2	3	4	5	6
Agua	Nivel	40	cms	Por presión	Por Flotador	Ultrasónicos	Lasér	Radiación	
	Temperatura	0-60	Grados C	Termostatos	Termopares	Termoresisten-cias	Pirómetros de radiación	Electrónicos	Infrarrojos
	Conductividad	0-14	mSiemens	Puente de Wheatstone	Potenciómetricos	Electrodos			
	Turbiedad	0-3500	NTU	Infrarrojos	Ópticos				
	PH	1-10	Ppm	Electrodos					
	Presión	0-4	Kpa	Trans. Resistivos	Trans.magnéticos	Piezoeléctricos	Capacitivos		
Motor Principal	Voltaje	0-150	V	Volmetros					
	Corriente	0-10	A	Ampermetros					
	Potencia	1/16-1	HP	Wattmetro					
Tina	Temperatura	0-200	Grados C	Termostatos	Termopares	Termoresisten-cias	Pirómetros de radiación	Electrónicos	Infrarrojos
	Torque ó Par	0-15	N m	Interferómetros lasér	Torquimetro	Dinametro			
	Velocidad	0-1200	Rpm	Dinamo-tacometria	Generadores de impulsos				
	Posición	0-10	cm	Encoders	Potenciómetros	Sincros	Resolvers	Lasér	
Aspas	Deflexiones	0-10	mm	Galgas	LVDT	Piezolectricos	Sensores Hall	Strain Gauges	
	Angulo de desplazamien to	0-360	Grados	Encoders	Potenciometros				

II.1 Tabla de Variables a medir y posibles elementos de medición.

II.2.1 Sensor de Presión

Uno de los principales problemas que se tienen en las lavadoras comerciales en el mercado mexicano es que cuentan con un presostato ver Figura II.3 el cual mide el nivel de agua que se tiene en la tina, estos presostatos solamente cuentan con tres posiciones:

- Abierto: es la posición que tiene el Presostato cuando se detecte que la tina de la lavadora no contiene agua.

- Medio Nivel: se tiene esta posición cuando se tiene la mitad de la tina con agua, que es aproximadamente de 30-35 lts de agua.

- Nivel Completo: Cuando se alcanza el nivel máximo de agua que podemos suministrar a la tina sin que esta se derrame y son de 60-65 lts. de agua.

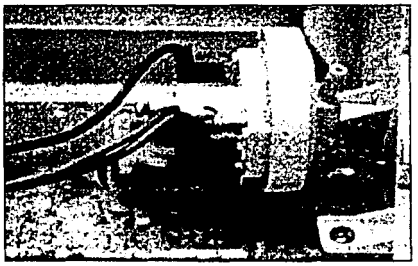


Fig II 3 Presostato utilizado en las lavadoras comerciales

Como podemos observar no se tiene precisión al medir la cantidad de agua que se le está suministrando a la lavadora, y el tener tres niveles de agua reduce los ciclos de lavado que se pueden tener, en este banco de pruebas se decidió reemplazar este dispositivo por un sensor con mayor resolución y precisión cuando se este midiendo el nivel de agua.

Se pensaron en varias posibilidades para medir el nivel del agua, como podría ser mediante la altura del nivel del agua en la tina, medir el flujo que pasa a través de las válvulas de admisión, medir el peso que nos genera la cantidad de agua que se tiene en la tina, pero muchos de estos dispositivos son dificiles de colocar y de gran costo. Se decidió medir la presión ejercida por la columna agua en el fondo de la tina y poder utilizar la manguera colocada en la lavadora y en la cual estaba inicialmente el Presostato.

Una de las formas de obtener una indicación analógica de nivel de liquidos consiste en medir la presión sobre el fondo del depósito que los contiene. Las diferencias de presiones entre el fondo y la superficie, es directamente proporcional al nivel respecto a dicho fondo y al peso especifico del líquido.

Otro punto importante para la selección del sensor es el rango de medida, para lo cual consideramos el nivel máximo de agua que puede tenerse en la tina para una carga completa de ropa y la cual es de 40 cm, por lo tanto la presión hidrostática que se tiene la podemos calcular como:

$$P = \rho gh = (1000\ Kg/m^3)(9.81\ m/s^2)(0.4m) = 3924\ [Pa] = 3.924\ [Kpa]$$

Donde :

P: Presión hidrostática

ρ: Densidad del agua

g: Aceleración gravitacional

h: Altura de la columna de agua.

Además el sensor cumple con las siguientes caracteristicas:

- Rango de medida 4 KPa
- Resolución alta
- Salida de voltaje analógica lineal
- Bajo consumo de potencia
- Sencillo de colocar
- Pequeñas dimensiones
- Bajo costo

Se decidió utilizar el sensor MPX de Motorola (Figura II.4), el cual es un sensor de presión piezorresistivo provisto de una precisa salida de voltaje lineal, directamente proporcional a la presión aplicada. Este sensor alberga un simple silicon monolitico dado por un strain gauge y una red de resistencias integrada en cada chip. El sensor esta ajustado mediante láser para una medición precisa, calibración de offset y compensación de temperatura [MOTOROLA,1998].

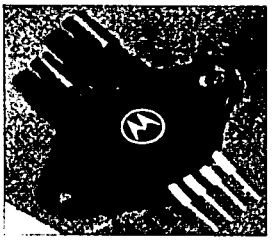


Fig. II.4 Sensor de presión MPX.

Caracteristicas:

- Rango de presión 0 a 10 kPa (0 a 1.45 PSI)
- Compensado respecto a variaciones de temperatura de 0 °C a 85 °C
- Salida de 25 mV a escala completa 10 kPa
- Linealidad de ±1.0% (max)
- Fuente de poder radio métrica
- Presión absoluta o diferencial.

La Figura II.5 muestra un diagrama de bloques del circuito interno del sensor de presión.

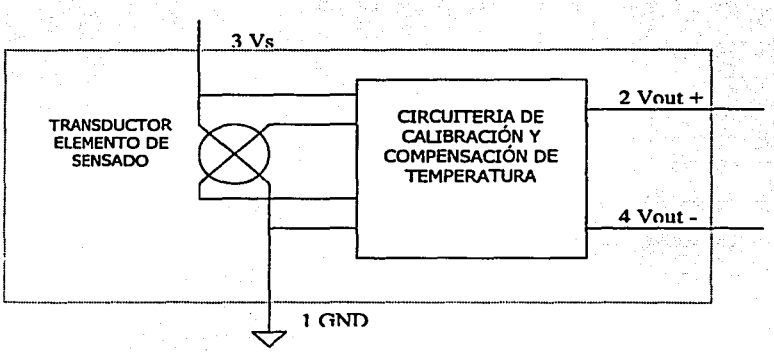


Fig. II.5 Esquema del sensor de presión compensado.

II.2.2 Sensor de Conductividad, Turbiedad y Temperatura

La Temperatura, Turbiedad y Conductividad del agua son buenos indicadores de la calidad o caracteristicas del agua que se tiene en la tina, la temperatura es importante controlarla, ya que cambian completamente las condiciones de lavado si no se tiene una temperatura adecuada,

La Turbiedad indica que tan "sucia" se tiene el agua de lavado, que cantidad de detergente se tiene disuelto y si la ropa esta soltando suciedad, también me dice si hay que cambiar el agua de lavado por agua limpia.

La conductividad puede variar considerablemente en base a la cantidad y tipo de detergente usado en el proceso de lavado.

Estas tres variables son importantes e influyen considerablemente en el proceso de lavado además de ser las tres variables que se estarán midiendo durante la rutina de lavado.

El sensor APMS Wash Process Sensor (WPS) de Honeywell Figura II.6 es un dispositivo en base a micropocesador y tiene la función de medir [HONEYWELL]:

- Turbiedad
- Conductividad
- Temperatura

La información del sensor puede ser usada en un esquema de control para monitorear y controlar un proceso y minimizar el consumo de energía, agua, materiales y tiempo.

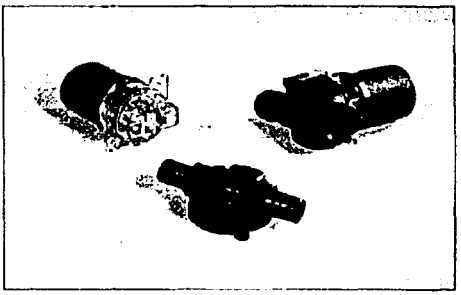


Fig. II.6 Sensor AMPS.

Cada una de las funciones sensadas son acondicionadas por el microprocesador interno. Todos los datos son trasmitidos a la PC via comunicación serial RS-232 de 5 VDC. El sensor opera en modo de esclavo, esperando a que la PC requiera información. La Figura II.7 muestra el diagrama de bloques funcional del sensor APMS-10G.

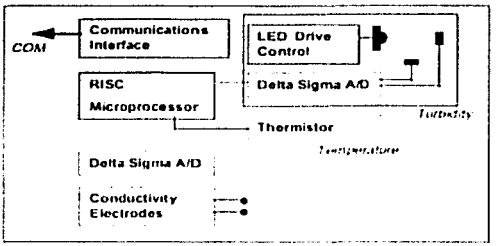


Fig. II.7 Diagrama de bloques funcional.

II.2.2.1 TURBIEDAD

La turbiedad se relaciona con la cantidad de sólidos suspendidos en el agua o en líquidos similares. El sensor no mide directamente los sólidos suspendidos, pero se hace a través del efecto absorber / esparcer que tienen estas sobre una luz. La cantidad de luz esparcida por una partícula depende del tamaño de la partícula, forma, composición e índice de refractividad. La turbiedad es medida usando un led infrarrojo de 935 nm y dos detectores Figura II.8.

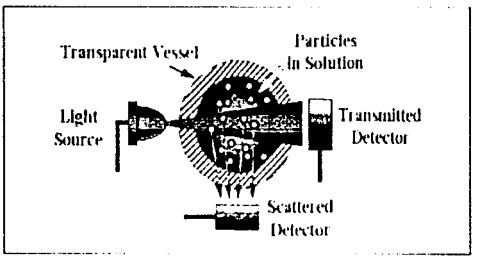
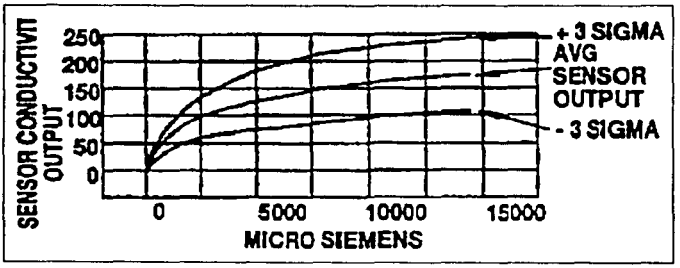


Fig. II.8 Principio de medición de Turbiedad

El sensor provee dos señales: la del detector de esparcimiento y la del detector del transmisor. El rango de la salida del sensor es 0-65535 (sin unidades). Para estas salidas, la salida llamada radio (esparcimiento divido por el transmisor) que puede ser calculado en la PC. El radio asegura una salida monótica y un amplio rango de sensitividad. El rango de la lectura de turbiedades 0-4000 NTUs (Nephelometric Turbidity Units). El sensor usa un control de ganancia para manejar el led, reduciendo asi el imparto de los efectos de una fuente de luz semejante. La salida del sensor de turbiedad es constante con respecto a variaciones de temperatura.

II.2.2.2 CONDUCTIVIDAD

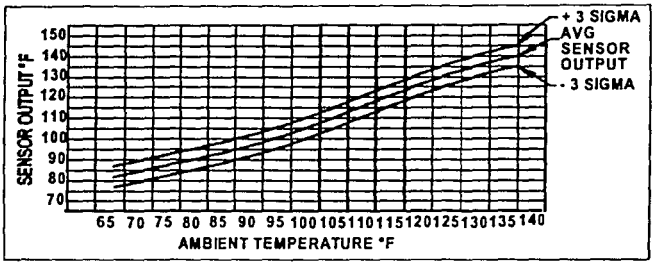
La conductividad mide la conductividad eléctrica de un fluido, y es usada para determinar la presencia de detergente en una solución de agua. La conductividad es medida usando dos eléctrodos de plata-niquel 303 estas puntas de prueba emiten una señal de AC (Corriente Alterna), de baja amplitud. El rango de medición es de 0.00001 a 15 mSiemens, el cual corresponde a una salida del sensor de 0-255 (sin unidades). En la Gráfica II.1 se muestra salida del sensor con respecto a la medición de conductividad.



Grafica II.1 Gráfica de salida del sensor con respecto a la conductividad medida.

II.2.2.3 TEMPERATURA

El sensor mide temperatura mediante un termistor, que es un dispositivo, el cual convierte la señal de temperatura en una señal de voltaje, este voltaje es leído por el microprocesador interno del AMPS y enviado a la PC mediante la interfaz de comunicación. En la Tabla II.2 se muestran las características de la medición de temperatura, y en la Gráfica II.2 se muestra la salida del sensor con respecto a la temperatura medida.



Grafica II.2 Gráfica de salida del sensor con respecto a la Temperatura medida.

	Mínimo	*Máximo*	*Unidades*
Rango	68	140	° F
	20	60	° C
Precisión	+4	+4	° F de 68 a 148° F
			° C de 20 a 60° C
Tiempo de Respuesta del Circuito		0.03	Segundos
Tiempo de Estabilización	3.00	5.00	Minutos

Tabla. II.2 Características del sensor de Temperatura.

II.3 Características de los actuadores.

II.3.1 Motor Principal

El motor principal (Figura II.9) de la lavadora se emplea en las rutinas de lavado, enjuague y centrifugado. En las dos primeras se utiliza en el modo de agitación, y en la última en el modo de rotación continua. El motor que se utiliza es un motor de inducción, jaula de ardilla, de ¼ Hp de potencia nominal, con dos embobinados que le permiten girar en sentido horario y antihorario y un capacitor de arranque de 45 μF conectado en paralelo entre ambos embobinados. El motor se encuentra acoplado al eje del agitador por medio de un juego de banda y poleas, con una reducción de velocidad de 5.1 a 1. La polea del agitador también se encuentra acoplada a la tina metálica interna de la lavadora, y por medio de un freno de banda (accionado por el solenoide de la transmisión) y un engranaje planetario, se permite el giro de la tina durante la etapa del centrifugado únicamente. El control del motor en las dos modalidades de operación se explica a continuación:

a) Modo de Agitación: Durante las etapas de lavado y enjuague, el motor ejecuta un movimiento oscilatorio rotacional continuo, este movimiento se logra enviando señales de voltaje a cada uno de los embobinados alternadamente.

b) Modo de Agitación Continua:
Para llevar acabo el centrifugado de la ropa por medio del movimiento giratorio continuo de la tina de la lavadora, se energiza uno de los embobinados del motor, de modo que el giro es constante y por un determinado lapso de tiempo.

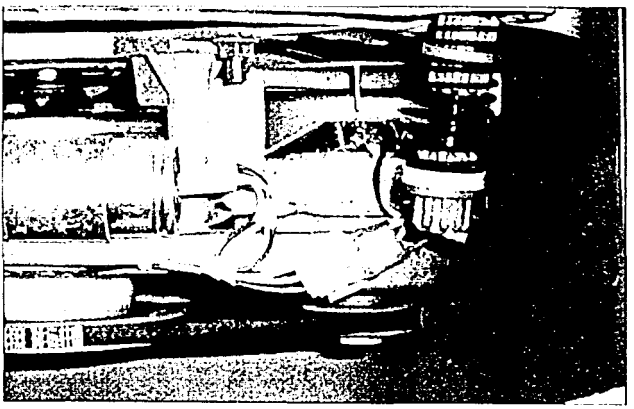


Fig. II.9 Motor Principal

II.3.2 Solenoide de la Transmisión

Como se mencionó en los párrafos anteriores, para ejecutar el movimiento del agitador y el de la tina al momento de centrifugar se emplea el mismo motor, acoplado por medio de una banda y poleas al eje común de ambos elementos. Sin embargo, antes de llegar a la tina, el eje está acoplado a un engranaje planetario. En este elemento se restringe el movimiento de la tina de la siguiente forma: la corona (o engranaje anular externo) del engranaje planetario es detenida por la parte exterior por medio de un freno de banda sujeto por un resorte durante la etapa de agitación. Cuando se tiene la etapa de centrifugado, el solenoide de la transmisión es energizado de modo que su vástago es desplazado hacia adentro. Como este elemento está conectado al freno de banda y al resorte que detiene a la corona, cuando se conecta el solenoide el freno libera a dicho engrane y el movimiento del motor se transmite a la tina a través del engranaje planetario. Cuando

la rutina de centrifugado termina, el solenoide es apagado y el resorte regresa a su posición inicial, provocando que el freno sujete nuevamente a la corona del tren planetario.

II.3.3 Bomba de Desagüe

La bomba empleada es la encargada de el drenado de la tina hacia el sistema de desagüe. La bomba se conecta a la energía eléctrica y permanece en operación con gasto constante hasta que la tina se vacía, posteriormente es desconectada. La bomba empleada tiene las siguientes características (Tabla II.3):

Modelo:	Hanning E-W DMP 358-158
Potencia eléctrica Nominal:	105 W
Impulsor: • Material: • Diámetro: • Alabes:	 Polipropileno 4.8 cm 5 alabes, rectos
Diámetro de la succión:	38mm (1 ½ ")
Diámetro de la descarga:	28 mm (1.1")

Tabla II.3 Caracteristicas de la bomba de desague

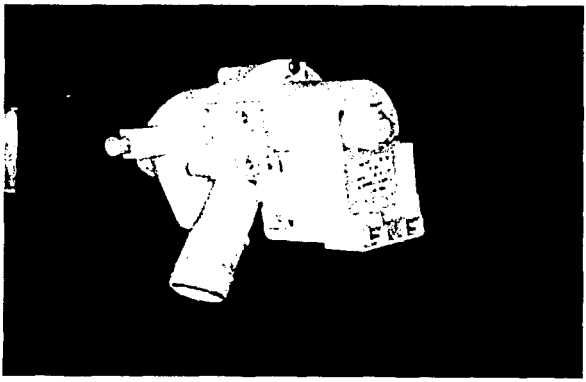


Figura II 10 Bomba de desague

II.3.4 Válvulas de Admisión de Agua

Las válvulas empleadas para llenar la tina de agua en las etapas de lavado y enjuague son del tipo de diafragma actuadas por solenoide. La lavadora cuenta con un elemento que contiene las dos válvulas, con entradas individuales para conectarse por medio de mangueras de hule a las tomas de agua caliente y fria, y una sola salida conectada al difusor de entrada de la tina de la lavadora. Las caracteristicas generales de las válvulas de admisión de Agua se muestran en la Tabla II.4:

Modelo:	EATON S-55
Potencia eléctrica nominal:	28 W
Material del cuerpo:	Polipropileno
Rango de Presión Admisible del agua de entrada:	0 a 48.2 Kpa (0 7 psi)
Temperatura máxima permisible del agua:	71 °C (160 °F)
Diámetro de la entrada:	17 mm (11/16")
Diámetro de la salida:	15 mm (5/8")
Coeficiente aproximado de pérdidas:	0.79

Tabla II 4. Caracteristicas de las válvulas de admisión de agua

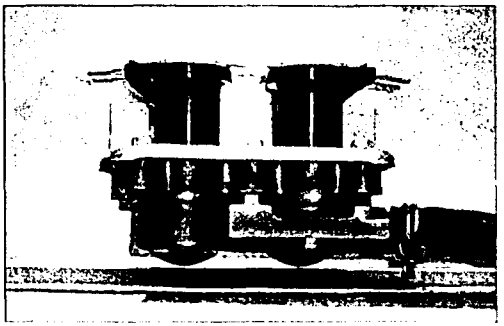


Figura II 11 Válvulas de admisión de agua

CAPITULO III.
ALTERNATIVAS DE CONTROL.

III.1 Introducción a los controladores.

El control automático ha jugado un papel vital en el avance de la ingeniería y de la ciencia. Además de su extrema importancia en vehículos espaciales, en guiado de proyectiles y sistemas de pilotaje de aviones, etc., el control automático se ha convertido en parte importante e integral de los procesos; maquinado, manejo y armado de piezas mecánicas en las industrias de fabricación, entre muchos otros, aunque en este trabajo solamente estaremos hablando de control de procesos, ya que consideraremos el lavado de ropa como un *proceso* el cual lo podemos definir como: una operación o desarrollo natural, progresivamente continúa, caracterizada por una serie de cambios graduales que llevan de una a otra de un modo relativamente fijo y que tienden a un determinado resultado o final; o una operación artificial o voluntaria, progresivamente continua que consiste en una serie de acciones controladas o movimientos dirigidos sistemáticamente hacia determinado resultado o fin [OGATA, 1993].

En los inicios de la era industrial, el control de los procesos se llevó a cabo mediante tanteos basados en la intuición y en la experiencia acumulada. Más tarde, el mercado exigió mayor calidad en las piezas o procesos que se llevarían a cabo, lo que condujo al desarrollo de teorías para explicar el funcionamiento del proceso, de las que derivaron estudios analíticos que a su vez permitieron realizar el control de la mayor parte de las variables de interés en los procesos.

Los avances en la teoría y práctica del control automático brindan medios de lograr el funcionamiento óptimo de sistemas dinámicos, mejorar la calidad y abaratar los costos de producción, expandir el ritmo de producción, liberar de la complejidad de muchas rutinas, de las tareas manuales repetitivas. Tal el es caso del problemas de lavado, el cual es una rutina repetitiva que lleva una secuencia de pasos, pero que es muy difícil de poder obtener un modelo matemático que represente su comportamiento, por eso hay que pensar en un control diferente al tradicional.

Antes de avanzar un poco más acerca de la teoría de control haremos unas definiciones que nos ayudaran a comprender un poco más acerca de que estamos hablando:

- **Sistema :** Un sistema es una combinación de componentes que actúan conjuntamente y cumplen determinado objetivo. Un sistema no está limitado a los objetivos físicos. El concepto de sistema puede ser aplicado a fenómenos abstractos y dinámicos, como lo es la economía.

Por tanto hay que interpretar el término <<sistema>> como referido a sistemas físicos, biológicos, económicos, etc. [BALCELLS,1998]

- **Sistemas de control realimentado:** Es aquel que tiende a mantener una relación preestablecida entre la salida y la entrada de referencia, comparando ambas y utilizando ambas como parámetro de control.

- **Sistemas de regulación automática:** Es un sistema de control realimentado en el que la entrada de referencia o la salida deseada son o bien constantes o varían lentamente en el tiempo, y donde la tarea fundamental consiste en mantener la salida en el valor deseado a pesar de las perturbaciones presentes.

- **Sistema de control de procesos:** Un sistema de regulación automático en el que la salida es una variable como temperatura, presión, flujo, nivel de líquido o pH, se llama sistema de control de procesos.

III.2 Controladores tradicionales.

Cuando se diseña un controlador hay que darse que cuenta que se pueden realizar de dos formas: un control de lazo cerrado o un control de lazo abierto. Un sistema de control de lazo cerrador es aquel en el que la señal de salida tiene efecto directo sobre la acción de control. Esto es, los sistemas de control de lazo cerrado son sistemas de control realimentado Fig. III.1

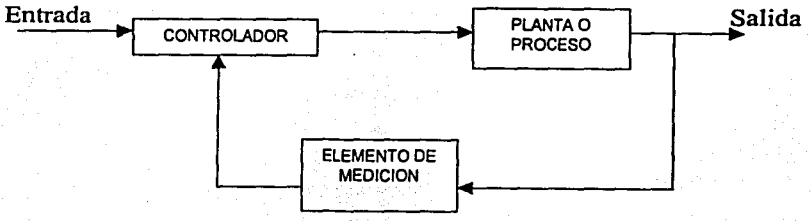


Fig. III.1 Configuración típica de un control de lazo cerrado.

La señal de error actuante, que es la diferencia entre la señal de entrada y la realimentación (que puede ser la señal de salida o una función de la señal de salida y sus derivadas), entra al detector o control de manera de reducir el error y llevar la salida del sistema al valor deseado. En otras palabras, el término <<lazo cerrado>> implica el uso de realimentación para reducir el error del sistema.

Los sistemas de control de lazo abierto son sistemas de control en los que la salida no tiene efecto sobre la acción de control. Es decir, en un sistema de control de lazo abierto la salida ni se mide ni se realimenta para su comparación con la entrada. La Fig. III.2 muestra la relación entrada salida de tal sistema.

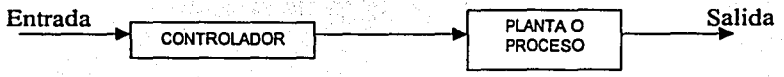


Fig. III.2 Configuración típica de un control de lazo abierto.

Un ejemplo práctico es la máquina de lavado que conocemos comúnmente. El remojo, el lavado y el enjuague en la máquina de lavado se cumplen sobre una base de tiempos. La máquina no mide la señal de salida, es decir, la limpieza de la ropa, el controlador es el timer electromecánico o el circuito electrónico, pero no se tiene una retoalimentación de las salidas del proceso.

En un sistema de control de lazo abierto cualquiera, no se compara la salida con la entrada de referencia. Por tanto, para cada entrada de referencia corresponde una condición de operación fijada. Así, la exactitud del sistema depende de la calibración. (Los sistemas de control de lazo abierto deben de ser cuidadosamente calibrados y para que sean útiles deben mantener esa calibración). En presencia de perturbaciones un sistema de control de lazo abierto no cumple su función asignada. En la práctica solo se puede usar el control de lazo abierto si la relación entre la entrada y la salida es conocida y si no hay perturbaciones ni externas no internas.

Una ventaja del sistema de control de lazo cerrado es que el uso de la realimentación hace al sistema en su respuesta, relativamente insensible a perturbaciones externas y a variaciones internas de parámetros del sistema. De este modo es posible utilizar componentes relativamente inexactos y económicos y lograr la exactitud de control requerida en determinada planta; mientras esto sería imposible en el caso del control de lazo abierto.

Desde el punto de vista de estabilidad, en el sistema de control de lazo abierto es más fácil lograr, ya que en un sistema de lazo cerrado constituye un problema de importancia, por la tendencia a corregir errores, que puede producir oscilaciones de amplitud constante o variables que afecta en gran parte a la planta o al proceso que sé este controlando.

Hay que recalcar que para sistemas en los que las entradas son conocidas previamente y en los que no hay perturbaciones, es preferible usar el control de lazo abierto. Los sistemas de control de lazo cerrado solamente tienen ventajas si se presentan perturbaciones no previsibles y/o variaciones imprevisibles de componentes del sistema.

Para obtener un mejor resultado es deseable medir y controlar directamente la variable que indica el estado del sistema. En el caso de sistemas de control de procesos se puede desear medir y controlar directamente la calidad del producto. Sin embargo, esto puede presentar un problema difícil, por ejemplo se pueden controlar variables (como presión, turbiedad y temperatura) directamente relacionadas con la calidad del proceso.

Como puede haber otras variables que afectan la relación entre la calidad y las variables medidas, el control indirecto de un sistema suele no ser tan eficaz como el control directo. Aunque pueda ser difícil, siempre hay que tratar de controlar la variable primaria o las que más afecten al proceso directamente.

Otro tipo de control son los llamados controles adaptables o adaptativos. La característica dinámicas de la mayoría de los sistemas de control no son constantes por diversas razones, como el deterioro de los componentes al transcurrir el tiempo o las modificaciones en los parámetros del sistema. Aunque en un sistema de control realimentado se atenúan los efectos de pequeños cambios en las características dinámicas, si las modificaciones en los parámetros del sistema y en el medio son significativas, un sistema para ser satisfactorio ha de tener la capacidad de adaptación. La adaptación implica la capacidad de ajustarse o modificarse por si mismo de acuerdo con modificaciones imprevisibles del medio o estructura.

En un sistema de control adaptable, las características dinámicas deben estar identificadas en todo momento de manera que los parámetros de control o detección puedan ajustarse para mantener el funcionamiento óptimo.

Muchos sistemas de control que aparentemente son de lazo abierto pueden ser convertidos a sistemas de lazo cerrado si se considera un detector o control humano que compara la entrada y la salida y realiza las acciones correctivas basadas en la diferencia resultante, o error.

Si intentamos analizar sistemas de control de lazo cerrado con operación humana nos encontramos con el difícil problema de escribir ecuaciones que describan el comportamiento del control que realiza un ser humano. A medida que éste va adquiriendo experiencia, se convierte en un mejor elemento de control.

Los controles en las lavadoras actuales, algunas ya incluyen algoritmos de control en donde tratan de describir el comportamiento que un controlador humano realiza cuando lava su ropa, pero ningún control pone atención en la salida final del proceso: la ropa limpia y sobre todo realizarlo en el menor tiempo y con el mayor ahorro de insumos naturales y energéticos.

El nacimiento y desarrollo de nuevos tipos de controladores donde no es necesario conocer un modelo exacto del proceso que se quiera controlar nos permite pensar en nuevas áreas de aplicación y el desarrollo de sistemas en donde la experiencia del operador humano es la base del control. Tal es el caso de control mediante Lógica Difusa que permite tratar información imprecisa, en términos de conjuntos borrosos o difusos.

De esta manera, los sistemas de control basados en lógica difusa combinan unas variables de entrada (definidas en términos de conjuntos difusos) por medio de grupos de reglas que producen uno o varios valores de salida.

Los sistemas difusos permiten modelar cualquier proceso no lineal, y aprender de los datos haciendo uso de determinados algoritmos de aprendizaje (a veces tomados de otros campos, como las redes neuronales o los algoritmos genéticos). No obstante a diferencia de las redes neuronales, los sistemas basados en lógica difusa permiten utilizar fácilmente el conocimiento de los expertos en un tema, directamente o bien como punto de partida para una optimización automática, al formalizar el conocimiento a veces ambiguo de un experto.

III.3 Control con Lógica Difusa.

La lógica difusa en los últimos años se ha desarrollado como una herramienta de gran utilidad para controlar sistemas y procesos industriales muy complejos, también se han estado aplicando en productos domésticos como son: sistemas de aire acondicionado, controles de temperatura en hornos, lavavajillas, lavadoras y secadoras de ropa, etc. Además de sistemas médicos de diagnóstico de enfermedades y sistemas expertos.

En la década de los años veinte J. Lukaslewics desarrollo los principios de la lógica multivariada, ahora conocida como lógica difusa, cuyos enunciados pueden tener valores de verdad comprendidos entre 0 (falso) y 1 (cierto) de la lógica binaria clásica.

- En 1965, Lofti Zadeth que es conocido como el padre de la lógica difusa aplico la lógica multivariada a la teoría de conjuntos, estableciendo la posibilidad de que los elementos pudieran tener diferentes grados de pertenencia a un conjunto (por ejemplo: un vaso con 90% de líquido tendría un grado de pertenencia a un conjunto llamado VASOS LLENOS de 0.9, en un rango de 0 a 1), Zadeth introdujo el termino fuzzy (difuso) y desarrollo un álgebra completa de lo conjunto fuzzy.
- En 1975, se desarrollo la primera aplicación de un control difuso para un motor de vapor, este control fue diseñado por E. H. Mandami.
- En 1980, Se desarrolla uno de los más importantes proyectos en una cementera de Copenaghe.
- En 1988, los japoneses aplicaron la lógica difusa para controlar el sistema velocidad y de frenado en el tren de Sendai.
- En 1992, Japón vende más de 2 millones de dólares en productos con lógica difusa. [MOTOROLA, 1999]

Actualmente se desarrollan gran cantidad de productos aplicando la lógica difusa ya que es un método en el cual se tienen grandes ventajas sobre otros sistemas de control.

Existen muchas formas de cómo se puede interpretar el Control Difuso, pero principalmente podemos verlo como un sistema de control difuso es un sistema experto en tiempo real, donde se implementa la parte del operador humano ó procesos de ingeniería donde por si mismo no es tan

fácil poder expresar el proceso mediante ecuaciones diferenciales ó encontrar mediante parámetros PID y se representa mediante reglas de acción/situación.

Los controles difusos difieren de los sistemas expertos tradicionales en diferentes aspectos. Una principal característica de los sistemas de control difuso es la existencia de dos niveles: primero, existen reglas si-entonces (if-then) simbólicas y cualitativas, variables difusas y valores como :

SI (IF) el nivel es bajo ***y (AND)*** la temperatura es alta, ***entonces (THEN)*** agrega agua fría .

Estos valores difusos como "bajo" , "alta" y operadores como "y" son representados dentro de objetos numéricos elementales y algoritmos: tablas de funciones, interpolación, comparadores, etc. La existencia de estos niveles de compilación es la base para rápidas implementaciones en tiempo real, como pueden ser para control difusos embebidos dentro de un ambiente numérico esencial de control convencional.

En la inteligencia artificial, el campo de física cualitativa sigue una aproximación similar, usando variables cualitativas y que son a veces llamadas "ecuaciones diferenciales cualitativas". La principal diferencia con esta técnica es el uso de intervalos "crisp" – para representar valores cualitativos.

El control difuso puede ser visto como un camino heurístico y modular para definir sistemas de control no lineal basado en tablas. Un camino para combinar controles difusos y PID es usar un sistema lineal PID alrededor de un punto de inicio, en donde se trabaja, y deslinealiza el sistema en otras áreas que describen el ambiente deseado o la estrategia de control con reglas difusas.

Un teorema desarrollado por Kosko [TERANO,1987], dice que cualquier función no lineal puede ser aproximada exactamente con un conjunto finito de variables, valores y reglas difusas. Este teorema describe el poderío de la representación de un control difuso en principio, pero no es la respuesta de cuantas reglas se necesitan y como poder encontrarlas, las cuales son esenciales en problemas y soluciones del mundo real. En muchos casos son sistemas simples y relativamente pequeños los que se necesitan controlar, pero también existen sistemas muy complejos que se tienen que controlar.

En los últimos años el control difuso ha tenido gran interés en aplicaciones de control para procesos industriales, existen un gran número de razones por lo que se ha vuelto tan "popular " el control difuso. Primero en la vida real, la mayoría de los procesos a controlar son completamente no lineales: su dinámica cambia en cada punto de operación y existen muchas no linealidades en el proceso.

El principal beneficio de la Lógica Difusa es que se puede describir el comportamiento de un sistema, mediante simples relaciones "si-entonces" ó "if-then". En muchas aplicaciones esto hace que se tenga una simple solución y se tenga un menor tiempo de diseño. Además se pueden utilizar todos los conocimientos de ingeniería para optimizar directamente el proceso o sistema.

Mientras esta característica de la lógica difusa en gran parte ayuda a los diseñadores de control, podría traer una gran limitación. En muchas aplicaciones, el conocimiento que describe el ambiente del sistema deseado esta contenida en conjuntos de datos. Aquí el diseñador tiene que derivar las reglas "if-then" de estos conjuntos manualmente por lo tanto si se tiene un gran número de conjuntos de datos representa un gran esfuerzo el poder encontrar las reglas.

Así podemos decir que la Lógica Difusa tiene las siguientes características:

- Usa una representación de conocimiento explícito.
- Realiza verificación y optimización de manera fácil y eficiente.
- No se puede entrenar, esto es que sea capaz de obtener nuevos conocimiento.

III.3.1 Los beneficios del control difuso.

Considerando las aplicaciones existentes del control difuso, desde sistemas basados en microcontrolador para el hogar hasta aplicaciones en control de procesos de gran escala, las ventajas del uso de la lógica difusa la podemos definir en las siguientes categorías [CHIN-TENG LEE, 1996]:

1. **Implementación del conocimiento de un experto para un amplio grado de automatización:** En muchos casos de control de procesos industriales, por ejemplo en la industria química, el grado de automatización es bajo, muchas veces se tiene a una persona supervisando el proceso. El conocimiento del operador esta basado en la experiencia de este y el cual no puede ser expresado en ecuaciones diferenciales. En este caso, el control difuso ofrece un método para representar e implementar el conocimiento de los expertos.

2. **Control no lineal robusto:** El control difuso ofrece alternativas para implementar soluciones sencillas, pero robustas para cubrir un amplio rango de sistemas con parámetros desconocidos o disturbios que se puedan presentar durante el proceso.

3. **Reducción del tiempo de desarrollo**: Ya que un control mediante lógica difusa se compone principalmente de reglas lingüísticas, las cuales representan el conocimiento humano o la experiencia de un operador el tiempo de desarrollo de un sistema se puede reducir notablemente en comparación con el desarrollo de un control en el cual es necesario el modelo matemático de la planta o del proceso.

4. **El mercado**: En Japón, la palabra "fuzzy" es una de las más populares, los productos del hogar etiquetados con "control difuso", son asociados con modernidad, alta calidad y producto amigable. Los Japoneses han reportado ventas de controles difusos en productos del hogar en niveles de billones de dólares, lo que hace pensar que son productos de alta confiabilidad.

III.3.2 Conceptos básicos de la lógica Difusa.

A continuación definiremos algunos de los conceptos básicos de la lógica difusa y como están estructurados los sistemas de control en donde se aplica lógica difusa.

La lógica difusa es un método de decisión basado en reglas haciendo uso del conocimiento de expertos para el control de sistemas y procesos.

La lógica difusa puede ser usada para controlar procesos que una persona comúnmente controla manualmente con los conocimientos obtenidos con la experiencia. Las reglas de control lingüísticas que un humano experto puede describir de una manera intuitiva y general pueden ser trasladadas a una base de reglas para un controlador con lógica difusa.

Un término lingüístico puede ser definido cuantitativamente por un tipo de conjunto difuso, conocido como una función de membresía. La función de membresía especificamente define grados de membresía en una propiedad o variable física como puede ser por ejemplo temperatura o presión. Con funciones de membresía definidas para el controlador, se puede formular una base de reglas de tipo IF – THEN, las cuales son de tipo condicional. Dicho esto, una base de reglas y su correspondiente función de membresía son empleadas para analizar las entradas y determinar las salidas controladas por el proceso de inferencia difusa.

Si el control de un proceso puede ser descrito cualitativamente por un experto, la lógica difusa puede ser usada para definir un controlador que emule las reglas heurísticas del experto. Además,

la lógica difusa puede ser usada para controlar un proceso que un humano controla manualmente con su conocimiento ganado de la experiencia.

Uno de los conceptos básicos en la lógica difusa es la descripción matemática de incertidumbres lingüísticas usando conjuntos difusos. Las personas están forzadas muchas veces a hacer decisiones basadas en información subjetivas o imprecisa. Siempre y cuando la información no contenga elementos cuantitativos precisos, la gente puede usar conjuntos difusos para manejar de manera satisfactoria situaciones complejas.

No se necesitan tener reglas bien definidas para hacer decisiones. Muy a menudo, uno se puede aproximar con reglas que cubran solo unos cuantos casos y aplicar estas en una situación dada. Esta aproximación es posible gracias a la flexibilidad de las reglas.

¿ Qué es un set difuso (fuzzy set) ?

Es el elemento más básico de los sistemas difusos. En matemáticas clásicas nosotros estamos familiarizados con lo que llamamos elementos crisp. Consideremos un conjunto X de todos los números entre 0 y 10 el cual llamaremos universo del discurso, ahora definiremos un subconjunto A de X de todos los números reales en el rango de 5 y 8.

$$A = [5,8]$$

Ahora mostramos el conjunto A dado por esta función característica: por ejemplo esta función asigna un numero 1 o 0 a cada elemento de X, dependiendo si el elemento se encuentra o no en el subconjunto A, el resultado se muestra Figura II.3:

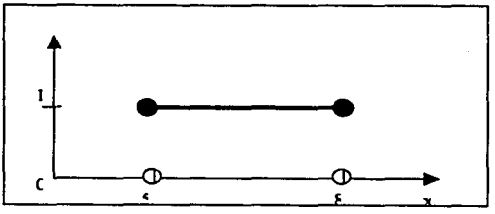


Figura III.3 Fuzzy Set.

Podemos interpretar a los elementos que fueron asignados con un número 1 como *Los elementos pertenecen al conjunto A* y los elementos que fueron asignados con el número 0 como *Los elementos que no pertenecen al conjunto A*. Estos conceptos son suficientes para muchas áreas y aplicaciones. Pero podemos fácilmente encontrar situaciones donde se necesite mayor flexibilidad, por ejemplo si queremos describir un conjunto de personas jóvenes. Más formalmente podemos denotar:

B={*conjunto de personas jóvenes*}

Así podríamos definir que la edad de una persona joven se encuentra entre 0 y 20 años y podemos definir el conjunto B como un intervalo crisp:

B = [0,20]

Ahora, la pregunta es: ¿Una persona un día antes de cumplir 20 años, se considera joven y una persona un día después de cumplir 20 no se considera una persona joven?

Un mejor camino para construir el conjunto B podría ser ampliar un poco la separación entre joven y no joven, y no solo utilizar los valores crisp, ahora en lugar de tener las decisiones:

Si, el / ella esta en el conjunto de personas jóvenes ó
No, el / ella está en el conjunto de personas no jóvenes.

Una frase o enunciado más flexible puede ser:

Esta bien, el / ella está un poco más relacionada con el conjunto de personas jóvenes ó
No, el / ella está un poco más relacionada con el conjunto de personas no jóvenes.

La lógica difusa es un método de formalizar la capacidad humana de razonar en forma imprecisa (razonamiento aproximado).Por lo tanto, en lógica difusa se trabaja con conjuntos que se definen por funciones de pertenencia que se denotan como μ c(x) e indican el grados de pertenencia (entre 0 y 1) de elemento con valor x al conjunto c.

III.3.3 MAQUINA DE INFERENCIA DIFUSA

Los motivos por los que se comienza a utilizar la lógica difusa en los controladores, se refieren sobre todo, a su simplicidad, ya que no requieren de modelos matemáticos complejos (no es preciso conocer la expresión algebraica exacta que gobierna el funcionamiento del sistema) , permitiendo en cambio diseñar mediante la descripción del funcionamiento con lenguaje natural y facilitando también las tareas de prueba y mantenimiento del sistema. Otras características de los sistemas difusos son su mayor suavidad en el control que en el caso de sistemas convencionales y su posible combinación con tecnologías clásica establecidas y con otras más modernas, como las redes neuronales.

La máquina de inferencia difusa Figura III.4, la podemos describir como los pasos que se necesitan para poder realizar un sistema difuso, esto es si tenemos una entrada nítida (valor numérico), se puede representar este valor con una variable lingüística de un peso dado, después se realiza la evaluación de las reglas que son las que gobiernan el sistema difuso, por último se vuelve a pasar de una variable lingüística a un valor nítido o numérico.

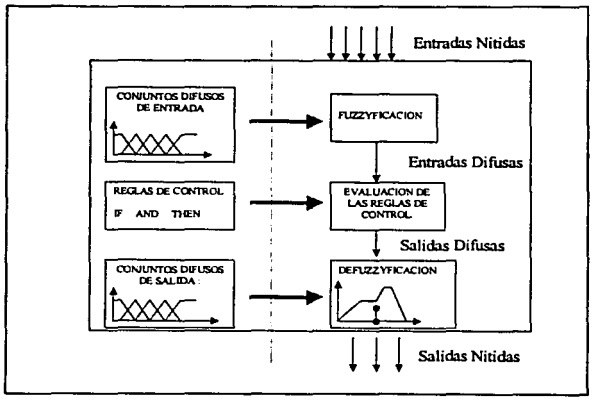


Figura III.4 Máquina de Inferecia Difusa

En este sistema la máquina de inferencia difusa esta residente en la PC, la cual se encargará de realizar todo el algoritmo y mediante la tarjeta de control electrónica se leerán las variables de entrada provenientes de los sensores y se mandarán las señales de control para los actuadores.

Las ventajas que tenemos en utilizar la PC como la encargada de realizar el control difuso son:

- Capacidad de modificar el algoritmo de control difuso.
- Capacidad de memoria.
- Velocidad de procesamiento.
- Despliegue visual de las gráficas de comportamiento de las variables de entrada y salida.

En la figura III.5 se muestra un diagrama de cómo es el flujo de información entre el sistema de control difuso y el proceso que se quiere controlar, se tiene un nivel lingüístico el cual es el representado por todas las variables lingüísticas y las reglas de control, además de un nivel técnico el cual es el representado por los valores de los sensores y de los actuadores.

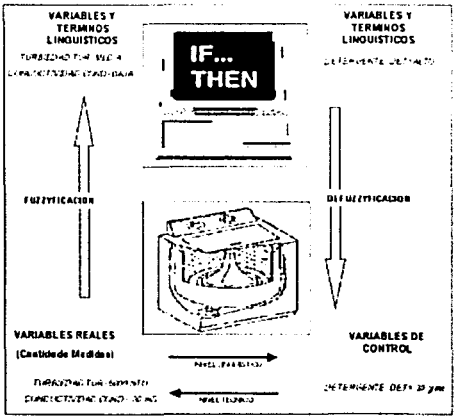
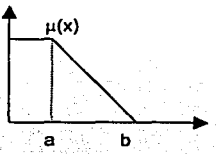


Figura III.5. Sistema de control difuso y flujo de información.

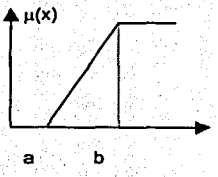
III.3.4 CONJUNTOS DIFUSOS DE ENTRADA Y SALIDA:

Existe gran cantidad de formas con las que podemos representar los conjuntos difusos de entradas y de salidas como pueden ser:

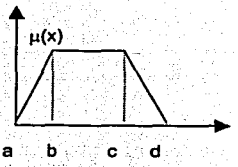
- **Forma Z**



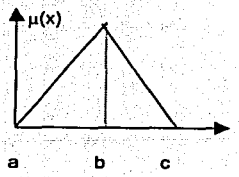
- **Forma S**

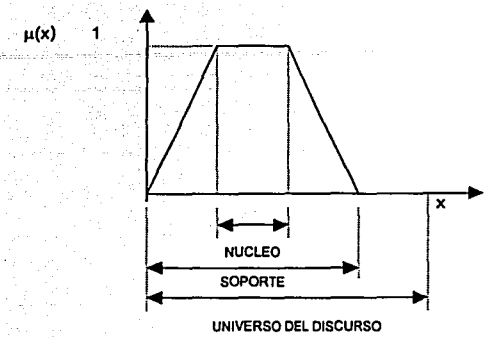


- **Forma Π o trapezoidal**



- **Forma Triangular**





La definición y forma de estos conjuntos de entrada dependen de el problema que se tenga aunque se sugiere lo siguiente:

- Los conjuntos deben de ser nones y preferentemente mayores que tres.
- Los cruces de los traslapes se deben de presentar en $\mu(x) \leq 0.5$.
- No debe de existir traslapes entre tres conjuntos.

III.3.5 DEFINICION DE LAS REGLAS DE CONTROL :

Para gobernar el comportamiento del sistema, el diseñador debe establecer una serie de reglas de la forma SI, ENTONCES para indicar la acción a realizar en función del conjunto al que pertenece la entrada al sistema, la forma general de la regla es :

R: Si (x es A) entonces (y será C)

También se pueden utilizar las notaciones:

R: A -> C *R: (A;C)*

La primera parte de la regla (SI) se denomina ***antecedente***, y contiene una o varias condiciones referidas a si cada una de las entradas del sistema pertenece a tal o cual conjunto borroso. La segunda parte (ENTONCES), denominada ***consecuente***, contiene los nombres de los conjuntos borrosos a los que deben pertenecer las salidas del sistema si cumple el antecedente correspondiente.

Una vez establecidos los conjuntos, se pueden crear las reglas de control que determinen la salida de un control en función de las entradas obtenidas por sensores.

III.3.6 FUZZIFICACION DE LAS ENTRADAS:

La fuzzificación de una entrada es el proceso por el cual se calcula su grado de pertenencia a uno o varios de los conjuntos difusos en que se divide el rango de valores posibles para dicha entrada. Por ejemplo si se trata de un sistema de control de la velocidad de giro de un ventilador cuya entrada es la temperatura ambiente, el rango (universo del discurso 5 a 40 ° C) de posibles temperaturas podría ser dividido en tres conjuntos difusos : FRIA, que incluiría las temperaturas, por ejemplo en el rango de 5 – 13 ° C, el conjunto FRESCA, con valores desde 9 hasta 21 ° C; el conjunto de temperatura AGRADABLE, con los valores 17 a 29 ° C; CALIDA, con los valores 25 a 37 ° C y el conjunto de temperatura CALIENTE con los valores entre 33 y 40 ° C.

Estos conjuntos pueden considerarse difusos si se supone que los valores de temperatura que contiene no pertenecen en el mismo grado al conjunto. En este caso, es evidente que una temperatura de 20° C es menos FRESCA que una de 15° C, con lo que la primera pertenecerá en menor grado que la segunda al conjunto de temperaturas FRESCAS. De hecho, la primera (20° C) también puede considerarse como AGRADABLE, ya que pertenece a este conjunto aunque en menor medida que otra.

Como se indicó anteriormente cuando se trabaja con conjuntos difusos, hay que establecer funciones de pertenencia de los elementos a los diferentes conjuntos, lo cual permite determinar, a partir del valor de un elemento, su grado de pertenencia al conjunto, siendo éste un valor real normalizado entre 0 (no pertenece en absoluto) y 1 (pertenece al 100 %). Esta función se denota como $\mu(x)$ siendo x el valor del elemento.

Las funciones de pertenencia deben definirse a partir de la experiencia o la intuición o simplemente utilizando el sentido común, y suelen tener como ya se dijo forma triangular , trapezoidal o gaussiana, a diferencia de las funciones escalón que se utilizan cuando se trabaja en lógica binaria (crisp logic) con conjuntos no difusos (crisp set).

III.3.7 EVALUACION DE LAS REGLAS DE CONTROL.

Su evaluación consiste en determinar qué regla (o reglas) se activará ante determinado valor de entrada. Para averiguarlo, se parte del grado de pertenencia de dicho valor a cada uno de los diferentes conjuntos difusos del dominio de entrada.

A cada regla Ri: (Ai;Ci) se le asocia un valor determinado peso que, en principio, coincide con el grado de pertenencia de la entrada (x) al conjunto indicado en el antecedente ($\mu Ai(x)$). Este peso será el que permitirá establecer el grado de pertenencia de la salida (y) del sistema al conjunto indicado en el consecuente de la regla ($\mu Ci(y)$).

Una vez ponderadas las reglas que van a gobernar el funcionamiento del sistema, se procede a defuzzificación de las salidas.

III.3.8 DEFUZZIFICACION DE LAS SALIDAS.

La defuzzificación de las salidas consiste en obtener un valor numérico para cada una de las salidas del sistema a partir de los conjuntos difusos a los que pertenece [MOTOROLA, 1999].

Existen varias técnicas de defuzzificación como son :

Método del centroide o centro de gravedad. Es la más utilizada. Consiste en crear para la salida del sistema una función de pertenencia a un nuevo conjunto obtenido como unión de aquellos a los que pertenece parcialmente el valor de salida. Esta nueva función puede calcularse mediante la suma de las funciones de pertenencia de estos conjuntos, pero multiplicada aritméticamente por el grado de pertenencia de la salida al subconjunto, que ya fue calculado en la fase anterior de la evaluación de las reglas de control.

Si por ejemplo se han activado dos reglas (Ri y Rj), obteniéndose que la salida (y) pertenece a los conjuntos Ci y Cj en los grados dados por los pesos de las correspondientes reglas, entonces la función de pertenencia del nuevo conjunto C, unión de los anteriores, sería:

$$\mu_C(y) = Peso\,\mathrm{Re}glaR_i * \mu_{ci}(y) + Peso\,\mathrm{Re}glaR_j * \mu_{cj}(y)$$

Otra posibilidad, sería el considerar las funciones originales, simplemente limitadas superiormente por el valor de los pesos. En este caso, se estaría utilizando el producto lógico difuso, en lugar del producto aritmético :

$$\mu_c(y) = \min(Peso\,\mathrm{Re}glaR_i * \mu_{ci}(y)) + \min(Peso\,\mathrm{Re}glaR_j * \mu_{cj}(y))$$

Una vez obtenida la función de pertenencia global, se calcula el valor exacto de la salida como el centroide (centro de gravedad) de esta función mediante la expresión general:

$$y = \frac{\sum_{l=1}^{M} y^l(\mu_B(y^l))}{\sum_{l=1}^{M}(\mu_B(y^l))}$$

donde y^l representa el centro del conjunto difuso, y $\mu_B(y)$ está definido como:

$$\mu_B(y) = \sup_{X \in U} \left[\mu_{F_1^l x \ldots x F_n^l \to G^l}(x, y) * \mu_A(x)\right]$$

La Figura III.6 muestra un diagrama del sistema de control en donde el flujo de control va de derecha a izquierda, cada una de las entradas es fuzzificada, esto es, la variable de entrada que es un valor numérico es representada por una variable lingüística, después se pasa a la evaluación de las reglas que en realidad es el corazón del control, cada una de las reglas representa el comportamiento del sistema que queremos controlar, esta evaluación de las reglas generan el grado de membresía de cada uno de las funciones de membresía de las salidas, entonces las salidas son defuzzificadas, esto es, la variable lingüística es convertida a un valor numérico y el cual es la señal de control para la planta.

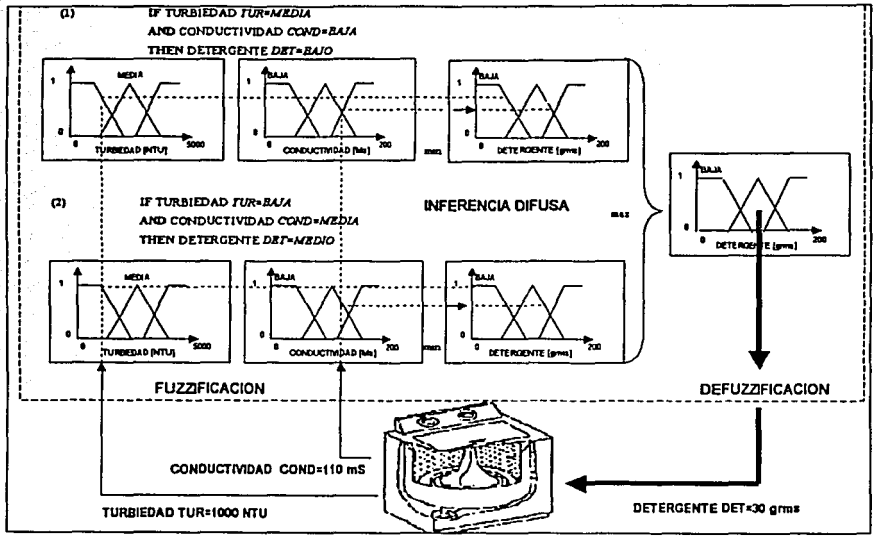


Figura III.6 Sistema de control difuso.

CAPITULO IV.

TARJETA ELECTRÓNICA DE CONTROL.

IV.1 Introducción.

Para poder manipular la lavadora es necesario contar con un dispositivo el cual sirva como interfaz entre la PC y la el banco de pruebas, esto es la PC la cual contiene el software de control difuso, una vez leídos los valores de los sensores y evaluado el control difuso, envía comandos de encendido y apagado de los actuadores y la tarjeta de control tiene que ser capaz de modificar el estado de estos.

En este capitulo se presenta el diseño de la tarjeta electrónica la cual cuenta con las siguientes características:

1. Entradas analógicas .
2. Etapa de amplificación para los sensores.
3. Conversión Analógica / Digital para las señales analógicas.
4. Entradas / Salidas digitales.
5. Etapa de potencia para los actuadores.
6. Comunicación con la PC.

La tarjeta electrónica fue diseñada pensando en las características anteriormente mencionadas y además debe de ser de un pequeño tamaño, contar con alimentación propia, utilizar componentes existentes en el mercado y que sea de fácil conexión.

En la Figura IV.1 se muestra un diagrama de bloques de la tarjeta electrónica desarrollada y en la Figura IV.2 una figura de la tarjeta real. El circuito principal de esta tarjeta es un microcontrolador el cual se programa mediante lenguaje ensamblador para realizar una tarea específica, el programa residente en la memoria cuenta con un protocolo de comunicación entre la PC y la tarjeta, además entre otras tares se encarga de :

- Enviar las señales de encendido y apagado de los actuadores
- Controlar que el convertidor analógico / digital realice la conversión
- Almacenar el estado de los actuadores
- Leer el valor de los sensores

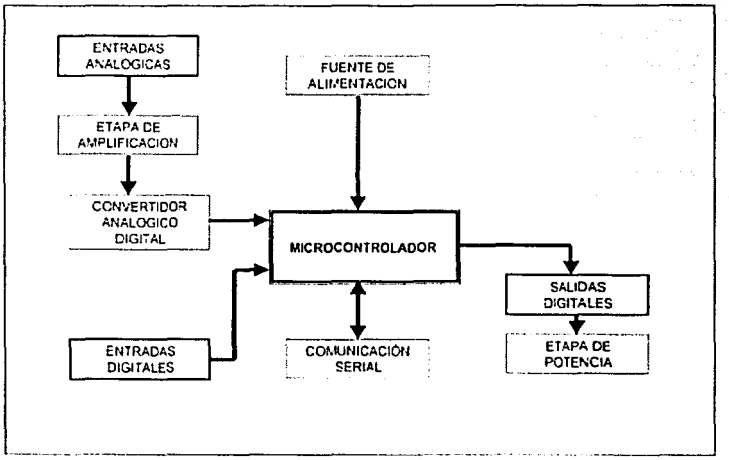


Figura IV.1 Diagrama de bloques de la tarjeta electrónica

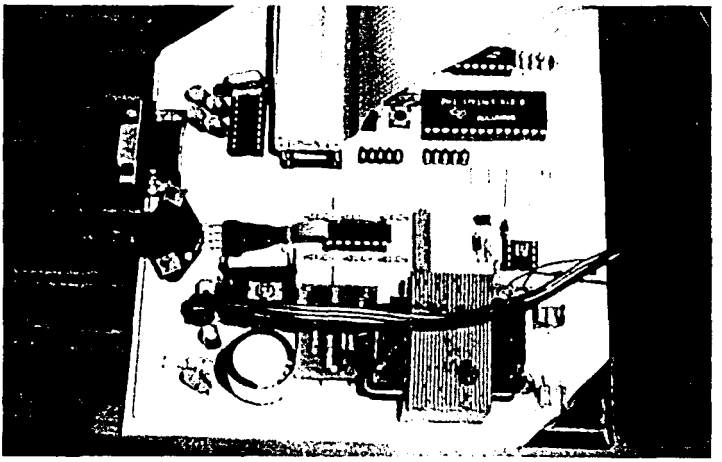


Figura IV.2 Tarjeta electrónica

IV.2 Descripción de los bloques de la tarjeta electrónica de control.

IV.2.1 Entradas analógicas. En todo proceso necesitamos medir variable física mediante sensores, los sensores convierten la señal física en una señal eléctrica que en la mayoría de los casos es una señal de analógica de voltaje.

Las entradas analógicas por tanto no sirven para poder conectar la salida de los sensores, esta tarjeta cuenta con 8 entradas analógicas en las cuales se pueden conectar señales de voltaje de 0 a 5 Volts.

IV.2.2 Etapa de amplificación para los sensores: Algunos sensores dan una señal de voltaje muy pequeña (mV), otros la entregan de manera diferencial, esto es la señal que proporciona no esta referida a una tierra, por lo que es necesario contar con un circuito que amplifique esa señal y que la entregue en forma no diferencial, cada circuito de amplificación tiene que ser diseñado de acuerdo al sensor que se vaya a conectar.

En el presente trabajo se utiliza un sensor de presión, el cual se describió en la sección II.2. Como se mencionó el sensor entrega una señal de voltaje de 25 mV para una presión de 10 Kpa, en la lavadora el valor máximo de presión es de 3.924 Kpa, por lo que el sensor entregaría un máximo 9.81 mV.

El amplificador de voltaje Figura IV.3, que se diseño tiene una ganancia de Av = 500, para amplificar la señal de plena escala del sensor de 9.81 mV a 4.9 V para poder mandarla como una señal de entrada analógica.

Se utilizo una configuración de un amplificador de instrumentación, para poder amplificar la señal de voltaje diferencial, la ecuación que gobierna el voltaje de salida Vo, esta dada por [FAULKENBERRY, 1990]:

$$Vo = 1 + (2R1 / R2)(V2 - V1)$$

Si se tiene:

$$(V2 - V1) = 9.81\ mV$$

$$Vo = 4.9\ V$$

$$R1 = 100\ K\Omega$$

Entonces podemos despejar R2, el cual esta dado por:

$$R2 = 2R1(V2 - V1) / (Vo - 1)$$

$$R2 = 503.7\ \Omega$$

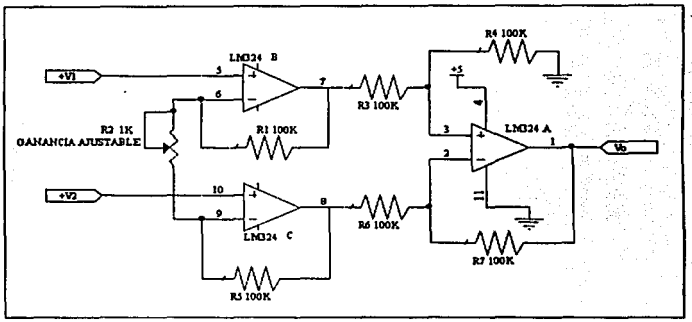


Figura IV.3. Diagrama eléctrico del amplificador para el sensor de presión.

IV.2.3 Conversión analógica / digital para las señales analógicas: Ya que el microcontrolador no cuenta con un convertidor analógico / digital (ADC : Analog to Digital Converter) propio para la lectura de las señales analógicas, fue necesario incorporar uno, se selecciono un convertidor de la serie ADC0800 de National Semiconductor el cual tiene las siguientes características [NATIONAL SEMICONDUCTORS, 1995]:

- Fácil interfaz a todos los microcontroladores.
- Opera con una sola fuente de 5 Volts.
- No requiere ajuste en cero y escala completa.
- 8 canales multiplexados con dirección lógica.
- Rango de voltaje de entrada de 0 a 5 Volts
- Salidas de Voltaje TTL (Transistor-Transistor Logic).
- Resolución 8 Bits.
- Bajo consumo de potencia 15 mW.
- Tiempo de conversión 100 μs.
- Mínima dependencia contra variaciones de temperatura.
- Excelente precisión y repetitibilidad.

Si se tiene:

$$(V2 - V1) = 9.81 \text{ mV}$$

$$Vo = 4.9 \text{ V}$$

$$R1 = 100 \text{ K}\Omega$$

Entonces podemos despejar R2, el cual esta dado por:

$$R2 = 2R1(V2 - V1) / (Vo - 1)$$

$$R2 = 503.7\ \Omega$$

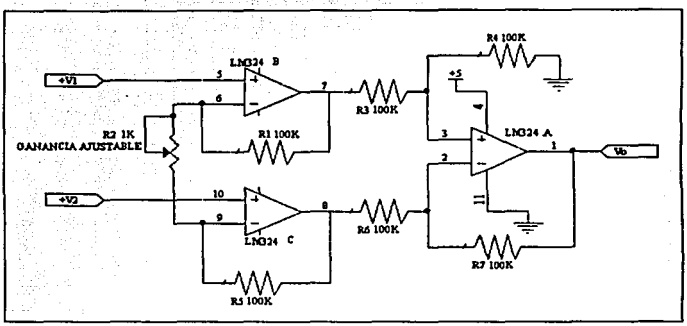


Figura IV.3. Diagrama eléctrico del amplificador para el sensor de presión.

IV.2.3 Conversión analógica / digital para las señales analógicas: Ya que el microcontrolador no cuenta con un convertidor analógico / digital (ADC : Analog to Digital Converter) propio para la lectura de las señales analógicas, fue necesario incorporar uno, se selecciono un convertidor de la serie ADC0800 de National Semiconductor el cual tiene las siguientes características [NATIONAL SEMICONDUCTORS, 1995]:

- Fácil interfaz a todos los microcontroladores.
- Opera con una sola fuente de 5 Volts.
- No requiere ajuste en cero y escala completa.
- 8 canales multiplexados con dirección lógica.
- Rango de voltaje de entrada de 0 a 5 Volts
- Salidas de Voltaje TTL (Transistor-Transistor Logic).
- Resolución 8 Bits.
- Bajo consumo de potencia 15 mW.
- Tiempo de conversión 100 μs.
- Mínima dependencia contra variaciones de temperatura.
- Excelente precisión y repetitibilidad.

Las características de este ADC lo hacen ideal para aplicaciones de control de procesos. En nuestro trabajo lo utilizamos para convertir las señales de las entradas analógicas, solamente se esta utilizando un solo canal del ADC los siete restantes están libres para lo conexión de cualquier otra entrada. La Figura IV.4 muestra el diagrama de bloques del ADC, como se puede ver, lleva un circuito de reloj el cual es necesario para su funcionamiento, en esta entrada (CLK) se conecto la salida de un circuito oscilador LM555 a una frecuencia de 500 KHz.

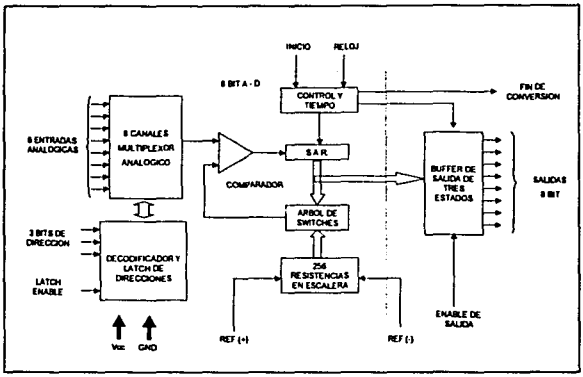


Figura IV.4 Diagrama de bloques del ADC0809.

IV.2.4 <u>Entradas Salidas / Digitales:</u> Las entradas digitales nos sirven para poder leer el estado de algún interruptor se utilizan dos niveles de voltaje 0 V que representa un 0 Lógico y 5 V que representa un 1 lógico. Tal es el caso del interruptor de la tapa que cuando se encuentra abierta se abre el switch mecánico y manda un 1 lógico , viceversa se tiene un 0 lógico cuando la tapa se llega a cerrar. Se pueden conectar otro tipo de entradas como son botones, sensores digitales, etc. En esta tarjeta se tienen 8 entradas digitales donde solamente en la entrada llamada DIG0 se conecto el interruptor de la tapa.

En las salidas digitales se conectaron los actuadores que son manipulados desde el programa de control. Se tienen 8 salidas digitales y a cada una de ellas se le asigno un actuador de la siguiente manera:

Salida	Nombre	Actuador
1	ACT0	Válvula VC
2	ACT1	Válvula VH
3	ACT2	Solenoide
4	ACT 3	Bomba desagüe
5	ACT4	
6	ACT5	
7	ACT 6	Motor 1
8	ACT 7	Motor 2

Tabla IV.1. Conexión de los actuadores.

IV.2.5 Etapa de potencia para los actuadores. Todos los actuadores encienden con un voltaje nominal 127 VAC, para poder encenderlos a partir de una señal digital (5V), es necesario contar con elementos que sirvan como interruptores, en este caso se utilizaron relevadores y TRIAC's.

Los relevadores son interruptores electromecánicos que son accionados al aplicar un voltaje de CD en un par de sus terminales. Esto provoca que circule una corriente a través de una bobina solenoide, la cual, por el efecto magnético cierra unos platinos que sirven como interruptor al paso de la corriente, los relevadores se utilizaron para el encendido y apagado de las válvulas de admisión, solenoide y bomba de desagüe.

Los TRIAC's son dispositivos electrónicos formados por varias capas de material semiconductor similares a los transistores que permiten el paso de corriente alterna a través de sus terminales (ánodo 1 y ánodo 2) cuando recibe una señal de corriente directa en su terminal conocida como compuerta.

Como medida de protección a los circuitos electrónicos, o para aislar la etapa de baja potencia de la de alta, se incluyen optoacopladores (también conocidos como optoasiladores) que permiten controlar la señal de disparo de los TRIAC's sin que exista conexión eléctrica entre los circuitos de carga y los circuitos de integrados de control. Los optoacopladores son dispositivos compuestos por un LED infrarrojo y un fototransistor.

En la Figura IV.5 se presenta el esquema eléctrico de la etapa de potencia. Las señales ACTX son las señales de encendido y apagado de los actuadores, por simplicidad en el esquema no se muestran todas las etapas de potencia con TRIAC y relevadores, ya que son idénticamente

iguales, solamente se indica que etapa de potencia lleva cada uno de los actuadores. La etapa de potencia cuenta también con un CI 74LS373 que es un LACTH y que solamente nos sirve para disminuir la corriente que el microcontrolador tiene que suministrar para encender ya sea el transistor y posteriormente el relevador o para encender el LED de optoacoplador.

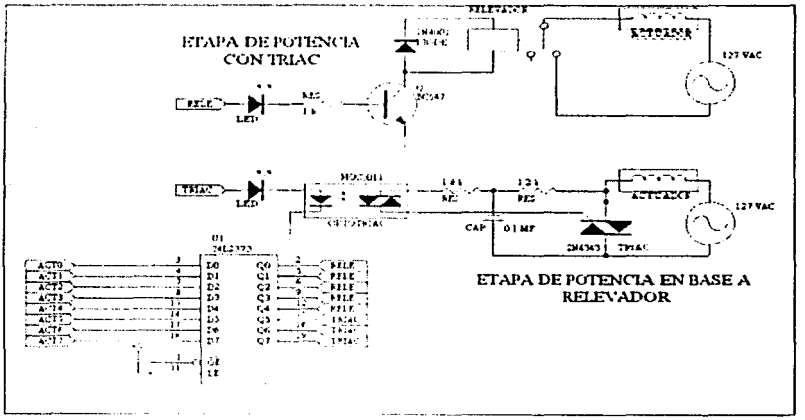


Figura IV.5 Esquema eléctrico de la etapa de potencia.

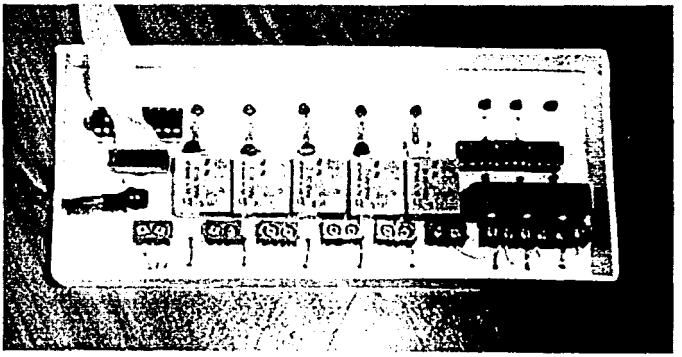


Figura IV.6 Etapa de potencia.

IV.2.6 Fuente de alimentación. Todos los actuadores son conectados a la alimentación de la energía eléctrica (127 V, corriente alterna) , pero la tarjeta electrónica, el sensor de presión y la etapa de potencia necesitan ser alimentados con corriente directa. La tarjeta electrónica y la etapa de potencia necesitan de 5 Volts para su funcionamiento y el sensor de presión de 12 Volts. Por tanto es necesario tener una fuente de alimentación (Figura IV.5) que convierta el voltaje de corriente alterna 127 VAC a voltajes de corriente directa de 5 y 12 Volts.

Una fuente de voltaje regulado está compuesta por los siguientes elementos [BOYLESTAD,1994]:

Transformador. El voltaje de corriente alterna (127 V) debe conectarse en primera instancia a un transformador de voltaje para reducir el voltaje a un nivel cercano al voltaje de corriente directa que se requiera.

Rectificador. Por medio de diodos se puede hacer un arreglo que transforme la señal de corriente alterna en una señal rectificada de corriente directa variable (en este caso se utilizo un rectificador de onda completa).

Filtro. La señal rectificada pasa a través de un filtro o capacitor, para generar una señal de corriente continua.

Regulador. Generalmente se emplean Circuitos Integrados (CI) para realizar la regulación del voltaje de corriente directa. La tarea del regulador de voltaje es mantener una salida de voltaje fija a la salida, aun cuando se tengan pequeñas variaciones en el voltaje de entrada.

El circuito de la fuente de alimentación se muestra en la figura IV.7, el transformador reduce el voltaje de entrada de 127 VAC a 18 VAC, después el puente de diodos realiza la rectificación de onda completa para la señal de 18 VA C esta señal se hace pasa por el filtro en el cual el valor del capacitor con un valor de 6000 μF, el cual fue calculado por:

$$C = 2.4\ Icd / Vr$$

Donde:

C : Valor del capacitor en [μF]

Icd : Corriente consumida por la carga [mA]

Vr : Voltaje de rizo deseado [V]

El consumo estimado de las etapa de potencia y de componentes electrónicos es de 500 mA, y el voltaje de rizo deseado de 200 mV.

Para regular la señal de voltaje se utilizaron dos reguladores 7805 y 7812 para las señales de 5 y 12 volts respectivamente.

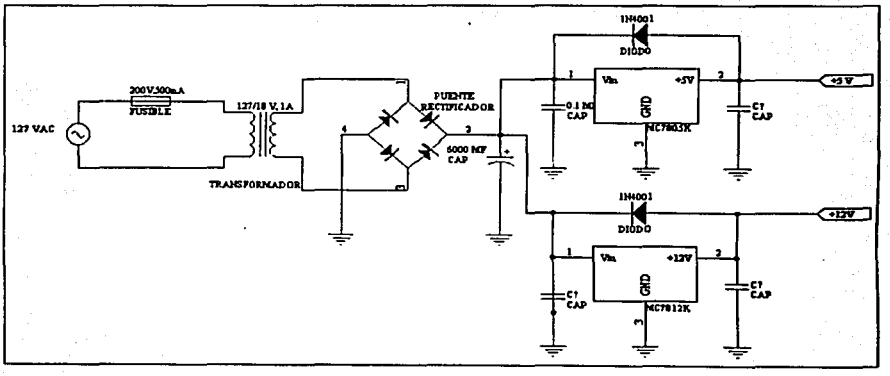


Figura IV.7 Esquema eléctrico de la fuente de alimentación.

IV.2.7 Comunicación serial. Para que exista una comunicación entre la PC y tarjeta de control, se utilizó una comunicación serial asíncrona con las siguientes características:

- Velocidad de comunicación: 9600 bps
- Bits de datos: 8
- Bits de paridad: Ninguno
- Bits de paro: 1
- Flujo de información: XON/XOFF
- Bytes de información: 6

La información que se envía a la PC o se recibe a la tarjeta de control, nos proporciona información acerca del proceso, para esto se definieron algunas instrucciones, como lo es escribir un dato en la memoria de la tarjeta de control, leer el estado de un actuador o pedir el valor de lectura de un sensor.

IV.2.8 Instrucciones para la comunicación entre la PC y la tarjeta de control.

La comunicación entre la PC y el Microcontrolador es mediante una trama de 6 bytes que se van a enviar y 6 bytes que se van ha recibir, cada comando de comunicación tendrá el siguiente formato:

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHK

- **ENCABEZADO1**: Indica si la información va de la PC al Microcontrolador (# CC) o viceversa (#AA)
- **OBJETO**: Indica el tipo de variable que se desea leer o escribir, puede ser una variable difusa, un sensor ó un actuador y están definidos como:
  3 : Variable de entrada
  4: Variable de salida
  5: Dirección de memoria.
- **ENCABEZADO2** : Indica la instrucción a realizar lectura, escritura, mensajes de éxito, error o notificación de algún evento:
  10: Lectura
  11: Escritura
  12: Éxito
  19:Error

- **ID1** : Identificador del sensor/actuador o localidad de memoria que solamente es de 00 a FF ya que es donde se encuentra la RAM y registros de control del Microcontrolador
- **DATO** : dato a enviar, cualquier valor entre 00 y FF.
- **CHK** : Checksum byte para detectar errores en la información el cual se calcula de la siguiente manera CHK = ENCABEZADO1+OBJETO+ENCABEZADO2+ID+DATO+1, el CHK se calcula cuando se recibe la trama de información e indica si la información que llego es correcta, en caso contrario se manda un código de error.

Cuando el valor de un byte en una trama de información no es esencial se utiliza don't care (no importa) el cual se representará como " * " y su valor en hexadecimal estará dado por #FF.

1 Solicitud del valor de un Sensor o estado de un Actuador:

La PC pide el valor de un sensor o el estado de un actuador, el cual esta identificado con ID.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC	# OBJ	# 10	#sensor/#actuador	*	CC+OBJ+10+ID+DATO+1

El microcontrolador envía de regreso el valor del sensor o el estado del actuador

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# AA	# OBJ	# 11	#sensor/#actuador	#dato	AA+OBJ+11+ID+DATO+1

# sensor	Descripción	# Actuador	Descripción
20	Sensor Presión	30	VC: Válvula de agua fría.
21	Sensor Cant. De Ropa	31	VH: Válvula de agua caliente.
22	Interruptor Tapa	32	Solenoide
43	AMPS: Turbiedad	33	Bomba Desagüe
44	AMPS: Conductividad	38	Tiempo de encendido MOT1
56	AMPS: Temperatura	39	Tiempo de encendido MOT2

2 Lectura del dato de una localidad de Memoria:

La PC pide el dato de una localidad de memoria del microcontrolador, la dirección de la localidad de memoria esta definida por ID, y es de 8 bits, el dato que contiene ésta es de 8 bits.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC	# OBJ	# 10	# memoria	*	CC+OBJ+10+ID+FF+1

El microcontrolador, le envía a la PC el valor de la memoria solicitada.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# AA	# OBJ	# 11	# memoria	# dato	CC+OBJ+11+ID+DATO+1

3 Escritura de una Localidad de Memoria:

La PC pide modificar una localidad de memoria del microcontrolador la cual puede ser un registro de propósito general, un puerto de E/S, un registro de control, etc.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC	# OBJ	#11	# memoria	# dato	CC+OBJ+11+ID+DATO+1

El microcontolador le envía a la PC la trama siguiente, La cual indica que la localidad de memoria fue modificada correctamente.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC	# OBJ	#18	# memoria	# dato	CC+OBJ+30+18+DATO+1

4 Errores en la comunicación o en el proceso :

La PC o el tarjeta de control envía un código de error indicando que existe un problema.

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC o #AA	# OBJ	#19	# error	*	ENCABEZADO1+OBJ+19+ ID+FF+1

# error	Descripción	# error	Descripción
01	Detener comunicación	10	Detener comunicación
02	CHK no coincide	20	CHK no coincide
03	Apagar Actuadores	30	Int. externa

5 Envío de mensajes de Notificación:

Estos mensajes de notificación, no tienen efecto sobre los actuadores, ni es la petición de algún dato, sirven para que la PC y el Microcontrolador puedan realizar otras tareas o para indicar que algún proceso ha terminado

ENCABEZADO1	OBJETO	ENCABEZADO2	ID	DATO	CHECKSUM CHK
# CC o #AA	# OBJ	#19	# mensaje	*	ENCABEZADO1+OBJ+19+ ID+FF+1

# mensaje	Descripción	# mensaje	Descripción
01	Inicio de proceso	10	MC ok
02	Fin de Proceso	20	
03	Pausa	30	

IV.2.9 Microcontrolador. El microcontrolador que se utilizo fue un COP8 de la compañía National Semiconductors, el cual es un microcontrolador de 8 bits, el cual es de gran velocidad, y bajo costo para aplicaciones de control en sistemas embebidos, los dispositivos de la familia COP8 son fabricados con la tecnología M²CMOS (Propia de National Semiconductors), para bajo consumo de corriente y amplios rangos de voltaje de alimentación. Muchas de las instrucciones son solo de un byte y tienen tiempo de ejecución de un ciclo de reloj (aproximadamente 1μs). Tiene múltiples modos direccionamiento y un gran conjunto de instrucciones que hacen que los programas realizados se reduzcan en tamaño. Otra característica del COP8 es que cuenta con entradas y salidas reconfigurables, timers multi-modo de propósito general, y una interfaz serial denominada MICROWIRE/PLUS que provee gran flexibilidad para la solución de una gran amplia gama de aplicaciones.

Cada microcontrolador de la familia COP8 ofrece las siguientes características [NATIONAL SEMINCONDUCTOR, 1996/1997]:

- Núcleo con un procesador de 8 bits.
- Tecnología CMOS (Metal Oxido Semiconductor de Simetría Complementaria) para bajo consumo de potencia.
- Modo HALT (parado) o de bajo consumo de potencia.

- Arquitectura de Memoria Mapeada. Toda la RAM (Memoria de Solo Lectura), puertos de entrada / salida y registros son mapeados dentro de un espacio de memoria de datos.

- Memoria de datos y de programa en un solo circuito.
- Timer versátil de 16 bits, el cual puede operar en modos PWM (Modulación por Ancho de Pulso), contador de eventos externos o registro de entrada de captura.
- Interrupciones mascarables.
- Dos registros de 8 bits como apuntadores.
- Un apuntador de programa de 8 bits.
- Tres tipos de señal de reloj: Cristal, circuito RC o reloj interno.

Un diagrama de bloques de la arquitectura de los microcontroladores de la familia COP8 se muestra en la Figura IV.8. Todos los dispositivos de la familia COP8 contienen los elementos mostrados en la figura. Estos elementos incluyen Unidad Aritmética Lógica (ALU), Memoria de Datos, Memoria de Programa, Timer, Puertos de entrada y salida y lógica de interrupciones.

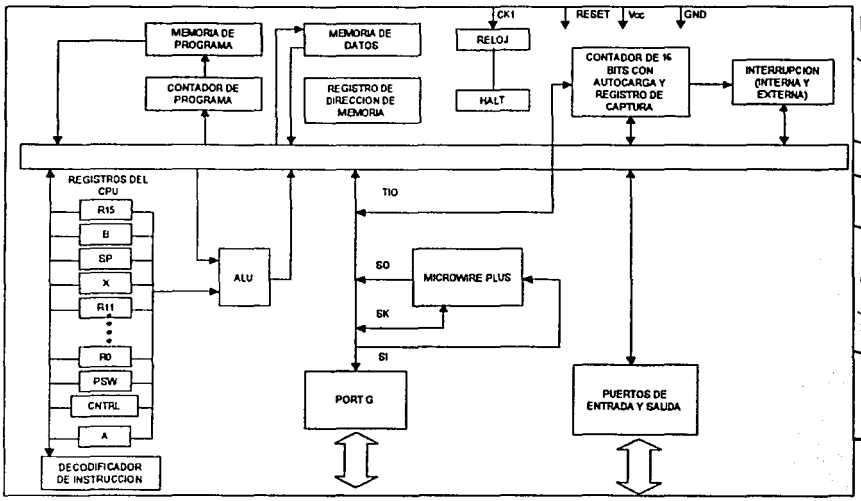


Figura IV.8. Diagrama de bloques del Microcontrolador de la Familia COP8.

El microcontrolador maneja el convertidor analógico digital, envía los valores de los sensores en forma serial cuando se los pide la PC y enciende y apaga los actuadores cuando la PC lo indica. El programa del microcontrolador fue desarrollado en lenguaje ensamblador, aquí se definieron alguna rutinas para la recepción y transmisión de los datos, validación de los datos recibidos, esto es se calcula el checksum enviado por la PC y se indica a ésta si la instrucción recibida es correcta, se configuro el timer para generar la base de tiempo que sirve para encender y apagar el motor principal en modo de agitación.

Se cuenta además con una rutina de interrupción externa, la cual es conectada al interruptor de la tapa, cuando el interruptor de la tapa cambia de estado, esto es de abierta a cerrada o viceversa, se genera una interrupción en el programa del microcontrolador y si se encuentra el sistema en la etapa de centrifugado, éste se detiene e indica mediante un mensaje que la tapa fue abierta, con el fin de dar protección al usuario, ya que es peligroso introducir las manos cuando la lavadora esta centrifugando a gran velocidad.

En la figura IV.9 se muestra un diagrama de flujo de las rutinas básicas del microcontrolador, y a continuación se explican cada una de ellas:

Inicialización del Microcontrolador: Aquí se configuran los puertos de entrada y salida. A los puertos de entrada se encuentra conectado el convertidor analógico digital y el interruptor de la tapa; en los puertos de salida se conectan los actuadores.

Configuración del Timer: Se modifican los registros de control del Timer para que se genere una interrupción en tiempo real de 50 milisegundos, esta interrupción sirve como base de tiempo para dar los pulsos para el motor en modo de agitación.

Configuración de la UART (Unidad Asíncrona de Recepción y Transmisión). En esta rutina se configura la UART para que tenga una transmisión de 9600 bps (Bauds por segundo), 8 bits de datos, sin paridad y un bit de stop, además de configuran la interrupciones que indican cuando llego un dato y cuando el buffer de transmisión está vació y listo para enviar otro dato.

Decodificar información llegada. Aquí se decodifica y ejecuta la instrucción que se esta pidiendo por parte de la PC, como puede ser que se lea el valor de un sensor, se modifique el estado de un actuador, se encienda o apague el motor (agitación), se lea o escriba alguna localidad de memoria del microcontrolador. Cuando alguna de las instrucciones no es decodificada correctamente o el checksum no coincide, entonces se manda un mensaje de error, en el cual se indica el error posible.

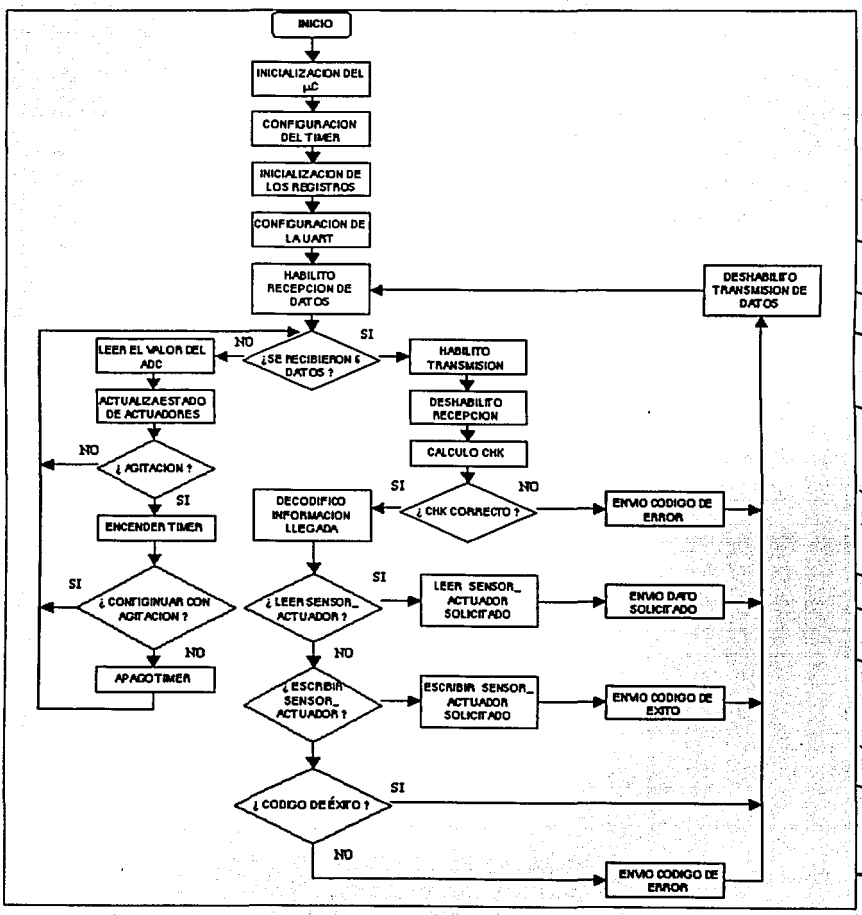


Figura IV.9. Diagrama de flujo del programa del microcontrolador.

CAPITULO V.
Desarrollo del Software de Lógica Difusa.

V.1 Generalidades del Programa "CONTROL UNIVERSAL DIFUSO" (FUC).

"Control Universal Difuso" (FUC), es un programa que permite el diseño, implementación, monitoreo y control de sistemas difusos para un amplio rango de aplicaciones, el programa fue desarrollado para que trabaje conjuntamente con la tarjeta de control.

El programa FUC cuenta con las siguientes características:

- Creación y edición de sistemas difusos a través de objetos.
- Creación y edición de direcciones de memoria a través de objetos.
- Comunicación serial con cualquier dispositivo externo a través de un protocolo de comunicación propio.
- Depuración del sistema difuso de manera manual y vía comunicación serial en donde es posible solicitar información de manera manual y automática a ciertos intervalos.
- Monitoreo y control en tiempo real del proceso.
- Generación de gráficas de comportamiento de las variables difusas en tiempo real.
- Generación de una bitácora que muestra el desarrollo completo del proceso de evaluación difuso y permite valorar la eficiencia del sistema.
- Almacenamiento de la información recabada y procesada para la alimentación de la base de conocimiento.

A través una interfaz gráfica es posible crear de manera rápida y fácil un sistema de control en lógica difusa . En la Figura V.1 se muestra la pantalla principal del sistema y en la cual se puede observar que se encuentra dividida en tres áreas principales las cuales se denominan como:

- El área de diagrama de árbol.
- El área de edición de sistemas.
- El área de control.

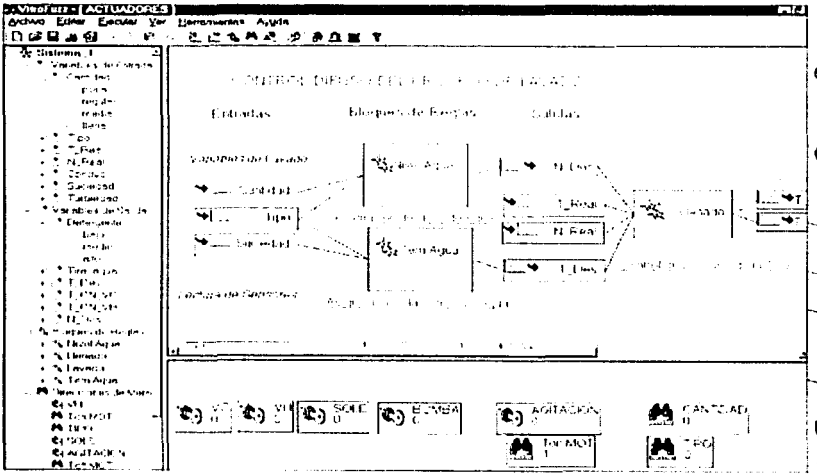


Figura V.1. Pantalla principal del Programa FUC

- *El área de edición de sistemas.* Ubicada en la parte central, es el lugar en donde se colocan y se editan los objetos que componen el sistema difuso. Aquí se muestran de manera gráfica las relaciones entre las variables de entrada, los bloques de reglas y las variables de salida.

- *El área de control.* Ubicada en la parte inferior derecha, es el lugar donde se crean y editan las direcciones de memoria que se desean controlar.

- *El área de diagrama de árbol.* Localizado en la parte izquierda de la ventana, muestra todos los objetos que han sido creados en el sistema actual, aquí los objetos son clasificados por su tipo, sin embargo también es posible observar las relaciones existentes entre las variables difusas de entrada y salida y los bloques de reglas. En este diagrama además es posible observar otros elementos como son los fuzzy sets correspondientes a cada variable

V.2 Descripción de los objetos .

V.2.1 Variables Difusas.

Variables de Entrada 

Son aquellas variables cuyo propósito es informar al sistema de las condiciones en que se encuentra un determinado parámetro, por ello este tipo de variable esta generalmente asociado externamente, de manera directa o indirecta, a un sensor. Sin embargo, es posible y en ocasiones preferible que la variable no adquiera su valor a través de un actuador, sino que el usuario indique en cualquier momento su valor de manera manual.

Variables de Salida 

Son aquellas variables cuyo propósito es llevar a cabo el control de dicho sistema, esto es, cambiar o mantener en un estado a un parámetro de control; por ello este tipo de variables generalmente esta asociado externamente, de manera directa o indirecta, a un actuador. Además, el valor de este tipo de variable, puede ser direccionado como una variable de entrada hacia otro bloque de reglas.

Bloque de Reglas 

Representa las relaciones existentes entre las variables que entran al bloque y las variables que salen de éste. Las relaciones están formadas por un conjunto de condiciones difusas que se evalúan partiendo del estado o valor actual de las variables de entrada asociadas y que generan a través de un proceso difuso un estado o valor para sus respectivas variables de salida.

Leyendas 

Son textos o etiquetas que se colocan en el área de edición de sistemas para señalizar o informar de alguna particularidad del sistema.

V.2.2 Direcciones de Memoria.

Memoria General 

Este tipo de objetos representan a una localidad de memoria localizada en el microcontrolador. El objetivo de este tipo de objetos es el de monitorear y alterar en cualquier momento el valor asociado a dicha localidad. Aunque teóricamente se puede acceder a cualquier dirección de microcontrolador, sólo pueden ser direccionadas 255 localidades a la vez y éstas deben ser definidas previamente. Existe, sin embargo, otro tipo de objeto de dirección de memoria que no se corresponde con ninguna dirección de memoria física del microcontrolador, sino que su propósito es el de avisar o generar algún evento. Los objetos dirección de memoria dividen en:

Interruptor 

Es un tipo especial de dirección de memoria de alcance limitado. Su función es la de informar al usuario de manera automática cuando un interruptor externo conectado al microcontrolador es accionado, y de ser requerido, llevar a cabo alguna función especifica, como puede ser la suspensión temporal o total del proceso de evaluación del sistema difuso.

Mensaje 

Es un tipo especial de dirección de memoria de alcance limitado. Su función es la de informar al usuario y/o al microcontrolador que un evento ha sucedido y es necesario llevar a cabo la tarea asociada al evento. Este tipo de objetos es utilizado, por ejemplo, cuando se desea pasar de la evaluación de un sistema anterior a uno nuevo; esto es, cuando la evaluación de un sistema ha llegado a su fin y se requiere establecer las condiciones iniciales para la evaluación del siguiente sistema.

Actuador 

Es un tipo especial de dirección de memoria de alcance limitado. Su función es informar al usuario del estado que guarda un actuador y de ser requerido cambiar el estado de éste en el momento que se desee.

***Nota:** Las funciones que desempeñan los objetos de tipo Interruptor, Actuador y Mensaje pueden ser llevadas a cabo por un objeto Dirección de Memoria, sin embargo, al restringir su funcionalidad se evita un uso inapropiado y además permiten una fácil identificación del objeto o evento que se desea controlar.*

V.3 Sistemas difusos en FUC.

Un sistema difuso básico (Figura V.2), consta de un objeto variable de entrada y un objeto variable de salida relacionados a través de un objeto bloque de reglas.

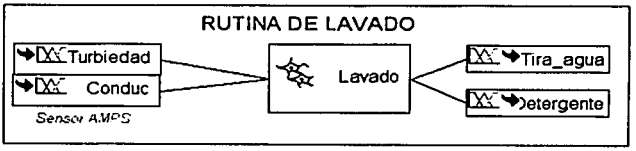


Figura V.2. Sistema Difuso Básico.

Para crear un sistema como este y que realmente funcione se deben establecer una serie de propiedades correspondientes a cada objeto y algunas otras propiedades que definan la relación entre ellos.

Comencemos por las propiedades de las variables de entrada. Sin embargo, para de tener una idea más clara de cómo se construye y funciona una variable difusa, un bloque de reglas y un sistema de control difuso, se incluye listados en código C++ con una breve explicación. El sistema FUC esta completamente realizado en código C++ y compilado con el compilador de Visual C++ 6.0. Los listados y estructuras mencionados se encuentran en el Apéndice 4.

V.3.1 Definición de Variables difusas

Como se ha mencionado en capítulos anteriores, las variables difusas nacen de la necesidad de representar a un fenómeno, propiedad, etc. por medio de conceptos que describan, a juicio propio, su comportamiento. Dichos conceptos cuyo objetivo es definir el estado en que se encuentra el fenómeno, reciben el nombre de conjuntos difusos o fuzzy sets. Los fuzzy sets, entonces conforman la parte fundamental de la representación de cualquier variable difusa.

La representación lógica de un fuzzy set esta determinada por una estructura de C llamada **FDB (Fuzzy Data Base)**. La estructura **FDB** define las propiedades como son el nombre del fuzzy set, un identificador, su tipo de generación (triangular, curva PI, singleton), su dominio, el valor de corte alfa, un vector que contiene los valores de verdad que representan al fuzzy set a lo largo de su dominio y un puntero que apunte a la siguiente estructura **FDB** que defina el próximo fuzzy set de la variable fuzzy. Sin embargo, un fuzzy set es esencialmente una tabla que contiene una serie de valores de membresia. Este grupo de valores es interpretado como la superficie del conjunto difuso.

Las variables fuzzy en FUC son entidades que relacionan a un grupo de estructuras **FDB** y almacenan las propiedades comunes de éstas.

El sistema FUC permite la creación de variables difusas y la modificación de sus propiedades a través de un diálogo llamado "Asistente de Variables" el cual se muestra en la Figura V.3.

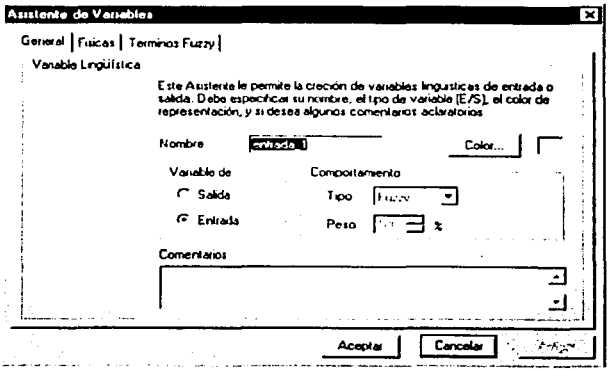


Figura V.3. Asistente de Variables en FUC

En la pestaña **General** se definen propiedades tales como el identificador de la variable difusa, su modo de operación (de entrada o de salida), su tipo de comportamiento y algunos comentarios relacionados con ésta.

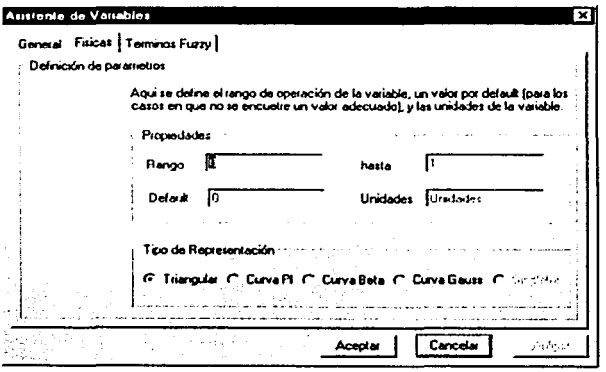


Figura V.4. Pantalla de Propiedades físicas de las variables

En la pestaña **Físicas** (Figura V.4), se define el rango de la variable, esto corresponde a todos los valores posibles que la variable puede alcanzar, el valor de default es el valor que toma la variable al comienzo de la evaluación del sistema difuso. Aqui también se define el tipo de representación general que tendrán todos y cada uno de los fuzzy sets que conformen a la variable.

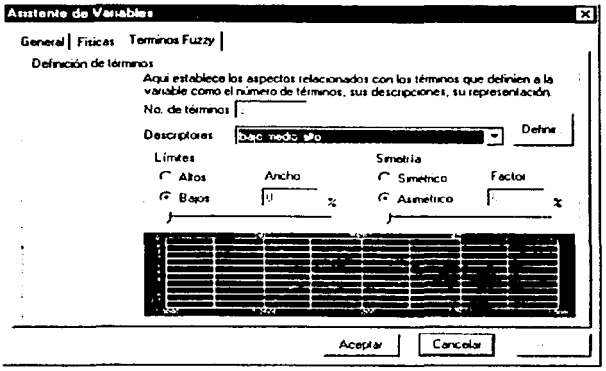


Figura V.5. Pantalla de Propiedades de los términos Fuzzy

En la pestaña **Terminos Fuzzy** (Figura V.5), se puede establecer el número de términos que representarán a la variable, a través de un combo de selección, es posible establecer un grupo de los descriptores más comunes para los términos. La sección **límites** se refiere al comportamiento de la variable en sus extremos, esto solamente afecta a los términos que delimitan a la variable. En la parte más baja del diálogo se muestra la distribución y forma de los fuzzy sets a lo largo del dominio de la variable. El botón **Definir** permite establecer de manera más precisa las propiedades de los fuzzy sets a traves de un cuadro de diálogo como el que se muestra en la Figura V.6.

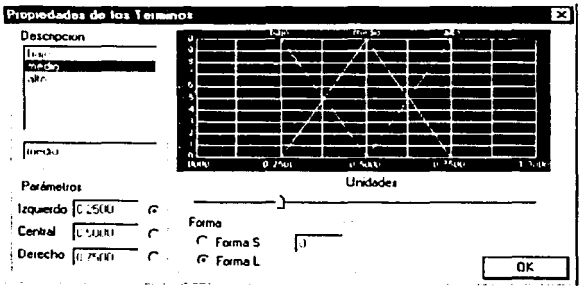


Figura V.6 Pantalla para definir los conjuntos difusos

En la parte superior se encuentra una lista con los descriptores de los fuzzy sets. Estos descriptores pueden ser modificados mediante el cuadro de edición que se encuentra debajo. La colocación de cada fuzzy set dentro del rango de la variable puede ser alterada mediante la sección **Parámetros** o mediante la barra de deslizamiento, finalmente, la forma del fuzzy set puede modificarse mediante los botones de selección ubicados en la sección **Forma**. Todos los cambios realizados pueden ser observados en la pantalla.

V.3.2 Propiedades de las variables de Salida

Las propiedades antes mencionadas son comunes tanto para las variables de entrada como para las variables de salida. Sin embargo, cuando la variable difusa opera como variable de salida existen algunas otras propiedades por definir. Dichas propiedades pueden establecerse mediante la pestaña **Métodos** (Figura V.7).

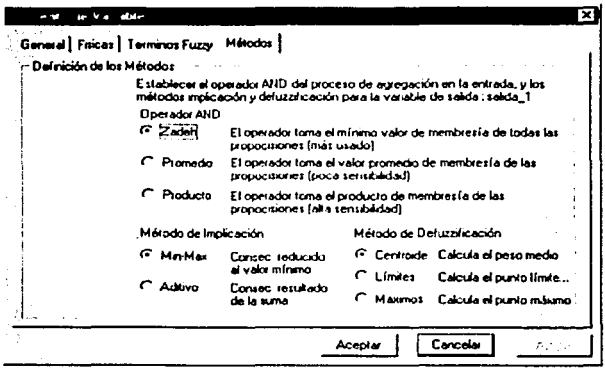


Figura V.7. Pantalla de Métodos de evaluación difuso

Estas propiedades se refieren a los métodos utilizados durante el proceso de evaluación difuso para ésta variable en particular. Por omisión la variable de salida toma los valores que se muestran en la figura anterior. A continuación se explican brevemente las características de los operadores y métodos mostrados en el diálogo.

Operador AND: esta propiedad se refiere al tipo de operador que se utilizara en el proceso de agregación de los valores de membresía del consecuente. Los operadores disponibles presentan la Tabla V.1, la cual indica el método utilizado, la ecuación correspondiente y como es el valor de la salida o comportamiento para cada una de las entradas, como se puede observar la salida depende del método utilizado, lo que implica que se tiene que seleccionar el método más adecuado dependiendo del comportamiento del sistema. Por ejemplo para una entrada A =0.75 y una entrada B =1, se tiene que para un operador AND tipo Zadeh se tiene 0.75, para una AND Promedio se tiene 0.875 y para una AND Producto, 0.750. En la practica el más utilizado es el operador de Zadeh y es la que nosotros utilizamos en el desarrollo del control.

AND	Ecuación	Comportamiento

Zadeh	$Min(\mu_A[x], \mu_B[x])$	**A/B**	**0.000**	**0.250**	**0.500**	**0.750**	**1.000**
		0.00	0.000	0.000	0.000	0.000	0.000
		0.25	0.000	0.250	0.250	0.250	0.250
		0.50	0.000	0.250	0.500	0.500	0.500
		0.75	0.000	0.250	0.500	0.750	0.750
		1.00	0.000	0.250	0.500	0.750	1.000
Promedio	$(\mu_A[x] + \mu_B[x])/2$	**A/B**	**0.000**	**0.250**	**0.500**	**0.750**	**1.000**
		0.00	0.000	0.125	0.250	0.375	0.500
		0.25	0.125	0.250	0.375	0.500	0.625
		0.50	0.250	0.375	0.500	0.625	0.750
		0.75	0.375	0.500	0.625	0.750	0.875
		1.00	0.500	0.625	0.750	0.875	1.000
Producto	$(\mu_A[x] * \mu_B[x]$	**A/B**	**0.000**	**0.250**	**0.500**	**0.750**	**1.000**
		0.00	0.000	0.000	0.000	0.000	0.000
		0.25	0.000	0.062	0.125	0.187	0.250
		0.50	0.000	0.125	0.250	0.375	0.500
		0.75	0.000	0.187	0.375	0.562	0.750
		1.00	0.000	0.250	0.500	0.750	1.000

Tabla V.1 Operador AND.

Método de implicación: se refiere a las operaciones que se evaluarán para que todas las proposiciones contribuyan a crear un espacio de salida. Cada proposición condicional cuyo valor de verdad es evaluado y está por encima del valor del umbral alfa contribuye a la forma de representación de la variable fuzzy de solución.

Los métodos de implicación se llevan a cabo en dos pasos. El primero es la actualización de la región fuzzy del consecuente. La segunda es la actualización de la región fuzzy de solución.

En el método **min-max** la región fuzzy del consecuente es restringida al valor mínimo de membresía del antecedente.[EARL COX 1994]

$$\mu_{cfs}[X_i]\min(\mu_{pl}, \mu_{cfs}[X_i])$$

La ecuación anterior indica que el conjunto fuzzy del consecuente (*cfs*) es modificado antes de ser utilizado. Esta modificación pone a cada elemento de la función de verdad al valor mínimo (de la función o del predicado (*pl*)).

$$\mu_{sfs}[X_i]\max(\mu_{sfs}, \mu_{cfs}[X_i])$$

La ecuación anterior indica que el espacio fuzzy de solución (*sfs*) es actualizado tomando el valor máximo de cada función de verdad. Cuando todas las proposiciones han sido evaluadas, la salida tiene un conjunto fuzzy que refleja la contribución de cada proposición. [EARL COX 1994]

Por ejemplo, las siguientes sentencias muestran el comportamiento de la apertura de las válvulas, con respecto a la temperatura y nivel que se tiene en la tina cuando esta se esta llenando:

if presión(p) is LOW and temperatura(t) is COOL
then abrir valvula is POSITIVEMODERATE;
if presión(t) is OK and temperatura(t) is COOL
then abrir válvula is ZERO;

Una figura representativa de estas reglas es la que se muestra en la Figura V.8:

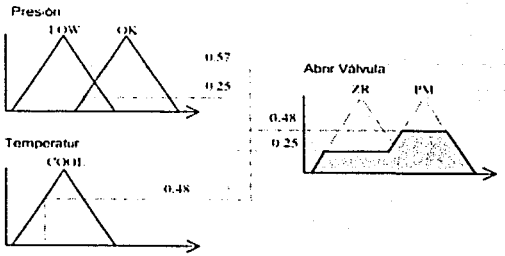


Figura V.8 Representación de la reglas.

En el método de implicación **aditivo** las operaciones se llevan a cabo mediante una ligera variante. La región fuzzy del consecuente todavía es reducida por el valor de verdad mínimo del predicado.

$$\mu_{cfs}[X_i] \min(\mu_{pt}, \mu_{cfs}[X_i])$$

pero la región de salida es actualizada por una regla diferente,

$$\mu_{sfs}[X_i] \min(1, \mu_{sfs} + \mu_{cfs}[X_i])$$

Esto es esencialmente la suma limitada aplicada a la región fuzzy de salida. La suma está limitada al intervalo [0, 1] de manera que el resultado de la suma no puede exceder el máximo valor de verdad de una región fuzzy.

El método fuzzy **aditivo** resuelve muchos problemas en la decisión de modelos donde deseamos que todas las reglas contribuyan en algo a la solución final del modelo. Utilizando el método de implicación **min-max**, sólo las reglas que tienen un valor de verdad alto en el conjunto fuzzy de salida harán alguna contribución (a la salida). El método fuzzy aditivo supera este problema en un gran número de casos. [EARL COX 1994]

Previo a la aplicación del método de implicación seleccionado se aplica un método de correlación. Existen dos métodos principales de correlación: **correlación mínima** y **correlación producto**. Sin embargo el sistema FUC aplica sólo el método de **correlación mínima**.

El método de **correlación mínima** es el método más común y se llama así, debido a que el fuzzy set de la variable de salida es minimizado por el corte o truncamiento en el punto de máximo valor de verdad del predicado.

La Figura V.9 muestra el resultado obtenido al aplicar la siguiente regla:

if presión[p] is LOW and temperatura[t] is COOL
then abrir válvula is POSITIVEMODERATE;

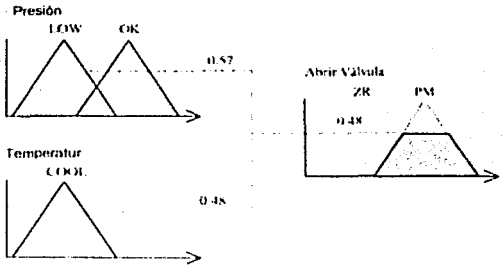


Figura V.9. Representación de la regla

Como se observa el valor mínimo [0.48] es el valor que se selecciona. El mecanismo de correlación mínima usualmente crea un altiplano debido a que la parte alta de la región fuzzy es cortada por el valor de verdad del predicado. Esto provoca una cierta pérdida de información. Si el conjunto fuzzy truncado es multimodal o irregular, la topología de la superficie por encima del valor de verdad del predicado se descarta. A pesar de esto es preferido sobre el método de correlación producto, debido a que es intuitivo, requiere menos complejidad y de una aritmética más rápida, y frecuentemente genera una superficie de salida que es más fácil de defuzzificar utilizando técnicas convencionales. [EARL COX 1994]

La función **FzyCondProposition()** lleva a cabo los procesos de correlación e implicación.

Método de Defuzzificación: se refiere al método utilizado para encontrar el valor que represente a la región de solución difusa. Los métodos de defuzzificación existentes en el sistema FUC son el **método del centroide**, el **método de límites** y el **método de máximos compuestos**. A continuación se mencionan las características de cada uno de ellos.

La técnica del **centroide o centro de gravedad** encuentra el punto de *balance* de la región fuzzy calculando el peso medio de la región. La figura V.10 muestra como el método del centroide encuentra un punto representativo del centro de gravedad del conjunto fuzzy.

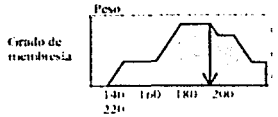


Figura V 10 Método Centroide de de Defussificación

La técnica de defuzzificación del **centroide** es la más ampliamente utilizada debido a que cuenta con varias propiedades deseables:

- Los valores defuzzificados tienden a un movimiento suave alrededor de la región fuzzy de salida, esto es, los cambios en la topología del conjunto fuzzy de la estructura del modelo al siguiente usualmente resultan en cambios suaves o pequeños del valor esperado.
- Es relativamente fácil su cálculo.
- Puede ser aplicado tanto a conjuntos fuzzy como conjuntos singleton de salida. [EARL COX 1994]

Explotando el concepto de centro de masa asociado con la solución de espacios difusos, podemos generar valores que reflejen la región con la mayor cantidad de *votos*. Así, el valor esperado es tomado de una región donde el número de conjuntos fuzzy de soporte tienen la más alta densidad. Esto significa que el valor esperado es seleccionado de una región que es soportada por un gran número de reglas. El primer momento de esta región o área es utilizada para calcular el valor esperado.

Existen tres tecnicas de **máximo compuesto** estrechamente relacionadas: *el máximo promedio, el centro de máximos*, y el simple *máximo compuesto*. Una máxima descomposición, como muestra la figura, encuentra el punto del dominio con el máximo valor de membresía. Si este punto es ambiguo (esto es, que se encuentre a lo largo de una altiplanicie), entonces este método emplea un promedio de valores o encuentra el centro de la altiplanicie.

A diferencia de la tecnica del **centroide**, la técnica de **descomposición máxima** tiene algunos atributos que son generalmente aplicables a una reducida clase de problemas, debido a que:

- El valor esperado es sensitivo a la regla que domina el conjunto fuzzy
- El valor esperado tiende a saltar de una estructura a la siguiente cuando la forma de la región fuzzy cambia.

Sin embargo, las técnicas de defuzzificación máxima son importantes para una amplia clase de modelos que calculan la máxima extensión de alguna propiedad fuzzy. El cálculo del riesgo es un ejemplo de tales modelos. Cuando las siguientes reglas de riesgo son evaluadas:

If age is young then risk is HIGH;
if distance.to.work is far then risk is MODERATE;
If accidents are above acceptable then risk is EXCESSIVE;
If DWI.convictions are above near zero then risk is UNACCEPTABLE;

el modelo debería ser sensitivo a la proposición que genera el valor máximo de verdad. Las decisiones sobre riesgo suelen ser discontinuas. [EARL COX 1994]

La función **FzyDefuzzyfi()** se encarga de aplicar el método de defuzzificación seleccionado para encontrar el valor de la región de solución.

Nota 1: *Cabe mencionar que a menos que se tengan razones para pensar que un sistema requiere de un método de defuzzificación más avanzado o especializado, se debe estar limitado al uso de la técnica del centroide y máximo compuesto para la defuzzificación. O al menos, estas técnicas deberían de ser el punto de partida.*

Nota 2: *Para llevar a cabo el proceso de evaluación difuso de principio a fin se hace uso de dos tipos de estructuras: la estructura FSV y la estructura VDB. Por cada variable de salida se crean unas estructura de este tipo. Su función es la de manejar la información necesaria para realizar el proceso de evaluación.*

V.3.3 Definición de un bloques de Reglas

Una vez definidas las propiedades de las variables involucradas en el proceso es necesario definir las propiedades del bloque de reglas. Como se sabe de capítulos anteriores, las variables difusas se relacionan entre sí a través de una o más reglas de condición. Las reglas o proposiciones en este sistema son manejadas por un objeto de bloque de reglas. FUC permite la creación y modificación de las propiedades de un objeto bloque a través de un diálogo llamado "Asistente de Bloque de Reglas" que se muestra en la Figura V.11.

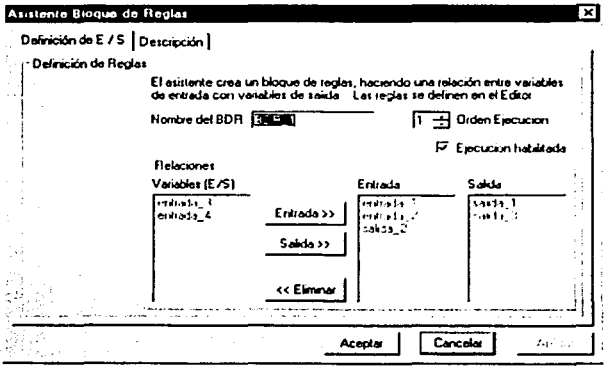


Figura V.11. Asistente para el bloque de reglas

En la pestaña **Definición de E / S** se puede establecer el nombre del bloque. La sección **Relaciones** presenta del lado izquierdo una lista con los nombres de todas las variables de entrada y salida del sistema actual que no se encuentran vinculadas todavía con el bloque de reglas. En la parte central se encuentran los botones que permiten vincular a las variables seleccionadas al bloque como variables de entrada o salida. En este punto cabe mencionar que las variables difusas de entrada sólo pueden vincularse a la entrada del bloque, mientras que las variables difusas de salida pueden vincularse a la entrada o a la salida del bloque como se muestra en la Figura V.12.

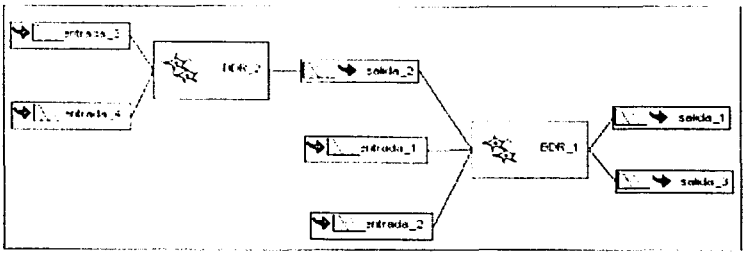


Figura V.12. Conexión entre las diferentes bloques de reglas

Con esto es posible establecer que la evaluación de un proceso difuso dependa de la evaluación previa de otro proceso. Para lograr esto, es necesario establecer el orden de ejecución de cada bloque de reglas del sistema a través del cuadro de edición **Orden Ejecución**. Para el ejemplo anterior el bloque *BDR_1* debería contar con un índice de ejecución mayor al índice de ejecución del bloque *BDR_2*. El botón **Eliminar** desvincula del bloque a las variables de Entrada o de Salida que se encuentren seleccionadas.

En la pestaña **Descripción** se puede establecer un comentario relacionado al proceso que evaluará el bloque.

Con lo anterior sólo se han definido las variables involucradas en el proceso que se desea controlar, falta sin embargo establecer las reglas que rijan al proceso difuso.

Un modelo fuzzy puede manejar proposiciones fuzzy condicionales e incondicionales. Sin embargo el sistema FUC maneja sólo proposiciones condicionales. Las proposiciones condicionales fuzzy tienen la forma general,

if w is Z then x is Y

donde *w* y *x* son valores escalares del modelo y *Z* e *Y* son variables lingüísticas. La proposición que sigue al término *if* es el antecedente o predicado y es cualquier proposición fuzzy arbitraria. La proposición a continuación del término *then* es el consecuente y es también cualquier proposición arbitraria. La declaración *x is Y* es condicional sobre el grado de verdad del antecedente. La proposición fundamental puede ser extendida con conectores fuzzy

if (w is Z) • (y is W) • ...(u is S) then x is Y

donde • es algunos de los conectores AND o OR. En el caso de múltiples antecedentes, el estado de *x* dentro de *Y* está determinado por el grado de membresía compuesto del antecedente completo. Esto es la base para el razonamiento fuzzy. En modelos fuzzy reales *x* es una región temporal (indicada por la letra *X*) que contiene los resultados de cada proposición en el espacio de *Y*. [EARL COX 1994]

Cuando un modelo se evalúa, se ejecuta como una serie de reglas o proposiciones fuzzy de manera que simula un procesamiento en paralelo (todas las reglas contribuyen al valor final de la variable de solución). Esto significa que todos los valores de las variables de solución no están disponibles hasta que todas las reglas han sido ejecutadas.

La definición de las reglas fuzzy en el sistema FUC se hacen a través del "Editor de Reglas". Por ejemplo para un sistema como el que se muestra en la figura V.13

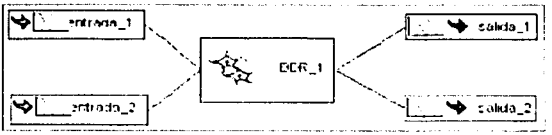


Figura V.13 Ejemplo para la generación del Bloque de Reglas

el editor de reglas podria mostrarse así

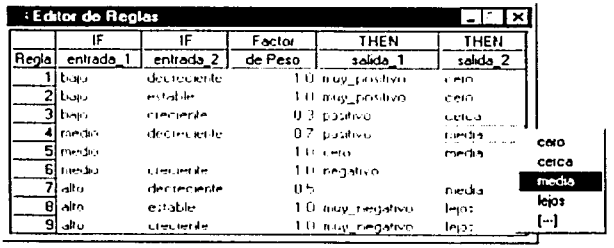


Figura V.14 Generación automática de las reglas

Como se puede apreciar en la parte superior se despliegan las etiquetas "IF" y "THEN" y junto con los nombres de las variables que aparecen debajo de estos, definen las variables del antecedente y el consecuente respectivamente de cada proposición fuzzy.

Cada uno de los renglones numerados representa una regla fuzzy a evaluar. El contenido de las celdas corresponde al nombre del conjunto fuzzy que se evalua para esa regla en particular. Por medio de un menú contextual es posible seleccionar el fuzzy set que se desea de la variable. Es importante hacer notar que la celda puede estar vacia, esto se logra seleccionando del menú el simbolo **[...]**, y significa que para la evaluación de esa regla en particular no se considera ningún estado de la variable. Del ejemplo anterior significaría que en la regla número 5 el valor de la variable **entrada_2** no es considerado; el disparo de la regla depende únicamente del grado de membresía del fuzzy set **medio** que pertenece a la variable **entrada_1.**

En la parte central del Editor, aparece la etiqueta "Factor de peso" que corresponde a un factor de proporcionalidad que se aplicará a la regla al momento de obtener el grado de membresía total del antecedente. Esto tiene la función de ajustar la medida en que una regla en particular afecta al modelo. El valor del factor de peso fluctúa entre 0.1 y 1.0.

Una regla o proposición fuzzy en el modelo no se mantiene como una estructura, sino que consiste de código de aplicación que realiza tareas como:

- aplicar algún tipo de compensador a un conjunto fuzzy
- determinar el grado de membresía de un escalar en un conjunto fuzzy
- realizar operaciones AND u OR fuzzy
- aplicar las funciones que manejan proposiciones condicionales e incondicionales.

Para evaluar una regla condicional el proceso es el siguiente: Primero es necesario encontrar en el conjunto fuzzy del predicado, el grado de membresía de un escalar. La función ***FzyGetMembership()*** encuentra el grado de membresía para un valor *dado* dentro de un conjunto fuzzy.

Utilizando este valor de verdad se llama a la función ***FzyCondProposition()***. Esta función se utiliza para evaluar proposiciones condicionales dado un cierto grado de membresía, finalmente se actualiza el fuzzy set de trabajo para la correspondiente variable de salida.

V.4 Ejecución del Sistema.

V.4.1 Ejecución del sistema en modo de Depuración

En este momento ya se puede probar nuestro sistema de control para observar su comportamiento. Sin embargo, cuando se esta diseñando un sistema no es de ningún modo recomendable probarlo y depurarlo directamente sobre su implementación física, antes necesita pasar por una etapa de pruebas para ajustarlo o sintonizarlo. FUC cuenta con dos modos de ejecución para los sistemas. El modo de *depuración* permite probar el sistema de manera manual; esto es que el usuario proporciona los valores de entrada para las variables fuzzy que actúan como entradas del sistema. En el modo de *enlace serial* los valores de las variables de entrada son tomadas directamente de un puerto de comunicación serial que previamente se les ha asignado.

El modo de ejecución de tipo depuración se realiza a través de un diálogo como el que se muestra en la Figura V.15.

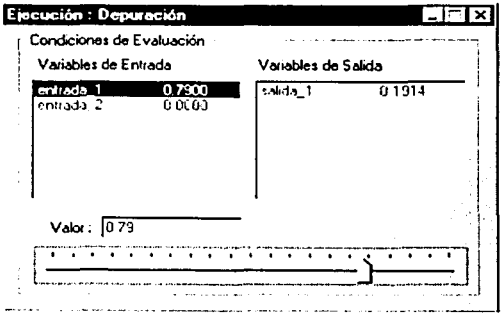


Figura V.15. Ventana de depuración del sistema difuso.

Este cuenta con dos listas, una para las variables fuzzy de entrada y otra para las variables fuzzy de salida. Las variables que se muestran en cada lista corresponden a las variables que se encuentran vinculadas por lo menos a un bloque de reglas que está habilitado para ejecución.

Al iniciarse la depuración del sistema, las variables de entrada toman su valor de defualt, por su parte las variables de salida toman el valor resultado de la evaluación difusa.

Para iniciar la depuración es necesario seleccionar una variable de entrada. El cuadro de edición que se encuentra debajo (etiquetado con **Valor**) refleja entonces el valor actual de esa variable, al mismo tiempo, el control de desplazamiento (localizado en la parte mas baja) refleja la posición relativa del valor dentro de todo el rango o dominio de la variable. Por medio de estos dos controles es posible cambiar el valor de la variable seleccionada. En cada cambio de valor de alguna variable de entrada se realiza la evaluación del sistema y los resultados son reflejados en los valores de las variables de salida.

Una correcta interpretación de los resultados nos permite realizar los cambios pertinentes para sintonizar nuestro sistema. Para lograr esto, FUC proporciona gráficas de comportamiento de cada variable como las que se muestran en la Figura V.16 en ellas se puede apreciar de una manera más evidente los estados por los que pasa una variable de entrada y la forma en que el sistema difuso responde a éstos a través de una variable de salida.

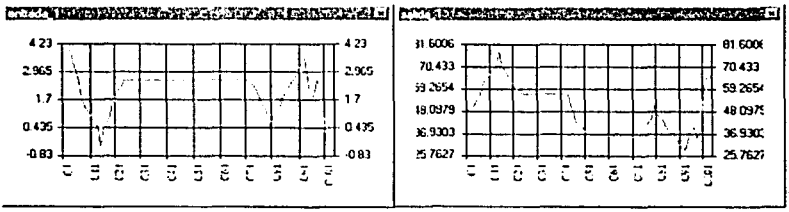


Figura V.16 Gráficas de Comportamiento de las variables de entrada y salida

Cuando se ejecuta un modelo difuso, ya sea en el modo de Depuración o en el modo de Enlace Serial, el sistema FUC genera dos archivos. Estos archivos se ubican por default en el directorio del programa, sin embargo, es posible establecerlos en alguna otra ruta.

Un archivo contiene toda la información de la prueba como es fecha y hora de inicio de la prueba, algunos comentarios referentes a las condiciones en que se realizó dicha prueba, nombres de las variables fuzzy y los bloques de reglas que se evalúan y un listado que tiene como encabezado el nombre de cada variable y todas las lecturas en forma de registros con los valores y el tiempo en que fue obtenida la información. El segundo archivo registra el comportamiento exclusivamente difuso, que corresponde a las áreas de solución de las cuales se obtiene el valor esperado después de aplicárseles el método de defuzzificación seleccionado. Estos archivos constituyen la base de conocimiento para nuestro sistema.

V.4.2 Ejecución del sistema en modo serial.

Superada la prueba de depuración es posible ejecutar de manera más segura el sistema en modo de enlace serial. Para lograr esto es necesario definir algunas otras propiedades de las variables y de los puertos de comunicación serial para que la comunicación pueda establecerse.

A través del diálogo **Opciones** es posible establecer dichas propiedades. En la pestaña **Generales**, en la parte superior se tiene un cuadro de selección, que en caso de estar habilitado, crea un archivo que contendrá todos los datos recabados (de las variables difusas) durante la ejecución del sistema en modo de enlace serial. En el cuadro de edición que sigue a continuación se establece la ubicación donde se creará el archivo. Los comentarios que se introduzcan serán añadidos a los archivos anteriormente mencionado como encabezado, esto es con el objeto de identificar las condiciones en que se realizó la prueba. Aquí también es posible cambiar, si se desea, el valor del corte alfa de todo el sistema. Por último es posible, si también así se desea, crear un archivo que contenga toda la actividad que se presente en los puertos de comunicación serial.

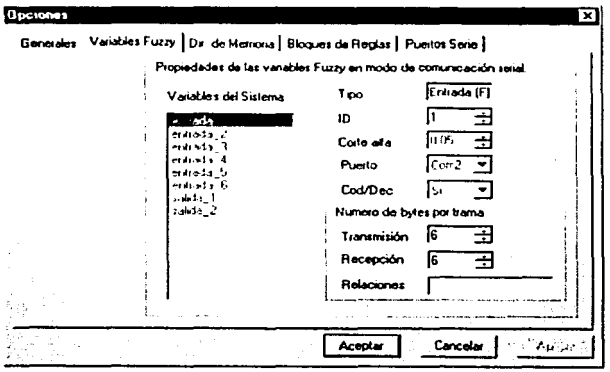


Figura V.17. Variables Fuzzy.

La pestaña **Variables Fuzzy** (Figura V.17), permite establecer las propiedades de comunicación para todas las variables fuzzy existentes en el sistema. La lista del lado izquierdo permite seleccionar una variable para poder establecer sus propiedades. Las propiedades de comunicación se muestran en la Tabla V.2.

ID	Define al identificador único de la variable que es un número comprendido entre 0 – 255. Este ID es como reconoce el dispositivo externo a la variable.
Puerto	Determina el puerto de comunicacion serial por el cual será procesada la información referente a la variable.
Cod/Dec	Se refiere a que si la información referente al valor asociado a la variable debe ser codificado y decodificado cuando se envíe o reciba la trama.
Transmisión y Recepción	Establece la cantidad de bytes en la trama de comunicación de la PC al dispositivo externo y del dispositivo externo a la PC respectivamente. Mas adelante se definirán las estructuras de cada una de las tramas.
Relaciones	Determina una función matemática que se aplicará a los bytes que correspondan con el valor de la variable.

Tabla V.2 Propiedades de la comunicación

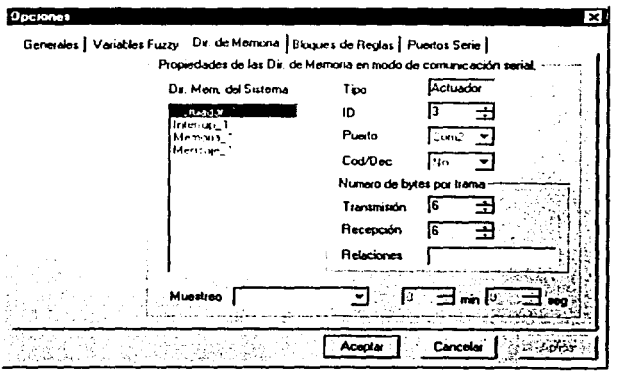


Figura V.18 Propiedades de la dirección de memoria.

La pestaña **Dir. de Memoria** (Figura V.18), permite establecer las propiedades de comunicación de los objetos direcciones de memoria del sistema. Como se comentó al principio del capitulo estos objetos son utilizados para tener un control sobre un actuador, un sensor, o un interruptor. Las propiedades de comunicación de este tipo de objetos son similares a las propiedades de las variables fuzzy. Adicionalmente en este diálogo se puede establecer una propiedad común que es el **Muestreo**, esto se refiere al intervalo de tiempo en el cual se solicitará información referente al estado de todas las direcciones de memoria (excepto para las dir. de Memoria tipo Mensaje).

Nota: *El ID de un objeto de dirección de memoria pueden coincidir con el ID de un objeto variable fuzzy, ya que en la trama de comunicación se identifica el tipo de objeto que se esta procesando.*

En la pestaña **Bloques de Reglas** se establecen dos propiedades de los objetos Bloques de Reglas que aunque no son realmente propiedades de comunicación, juegan un papel importante. La propiedad **Orden de ejecución** determina el orden en que será evaluado el bloque y por tanto el orden en que se solicita y procesa la información. Como ya se ha mencionado, esto es sumamente importante cuando la evaluación de un proceso difuso dependa de la evaluación previa de otro proceso.

La Propiedad **Ejecución habilitada** determina si el bloque será procesado o evaluado y por tanto, si la información de las variables que se encuentran vinculadas (únicamente) a éste será transmitida por algún puerto de comunicación serial.

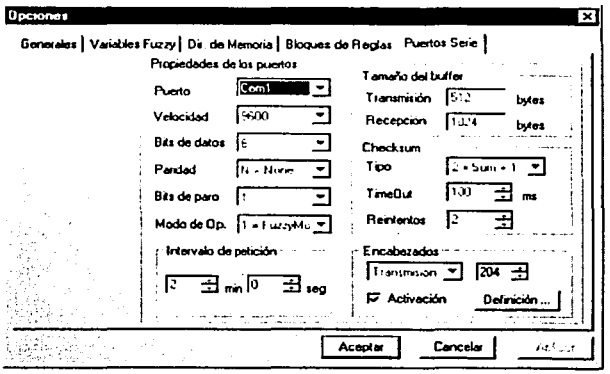


Figura V.19 Configuración del puerto serial de comunicación.

En la pestaña **Puertos Serie** se definen las propiedades de los cuatro puertos de comunicación serial que es posible manejar. Las propiedades como son Velocidad, Bits de datos, Paridad, Bit de paro y Tamaño de los buffers son de uso común.

La propiedad Modo de Operación se refiere al tipo de trama que se manejara a través del puerto. La trama *normal* maneja por lo general cuatro bytes: un encabezado de transmisión / recepción, un identificador de dispositivo, un dato y el checksum que ya se explicaron en el Capitulo anterior.

La propiedad **Timeout** se refiere al tiempo máximo de espera en respuesta antes de generarse un mensaje de error por time out (3). La propiedad **Reintentos** se refiere al número máximo de reintentos que hace automáticamente el puerto si la información recibida esta dañada o no se recibe respuesta del dispositivo externo.

La sección **Intervalo de petición** establece el lapso entre un requerimiento de información y otro cuando se ejecuta el sistema con enlace serial en modo automático.

La sección **Encabezados** cuenta con un combo de selección, el cual muestra los valores asociados a los encabezados de transmisión y recepción respectivamente. Estos encabezados pueden ser modificados en este mismo diálogo. Sin embargo, el botón Definicion muestra un nuevo diálogo en el cual se pueden editar todos y cada uno de los encabezados definidos por el sistema (Figura V.20).

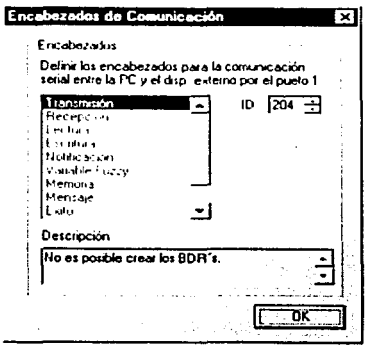


Figura V.20 Encabezados de comunicación.

El cuadro de opción **Activación** determina si todos los encabezados (excepto los de transmisión, recepción y tipo de objeto que siempre van) son agregados en las tramas de comunicación.

Nota: *Pudiera parecer una lista interminable de propiedades ha definir para poder ejecutar el sistema en modo de enlace serial, sin embargo, FUC por default establece las propiedades más frecuentemente utilizadas.*

Para llevar a cabo la ejecución del sistema en modo de enlace serial, se selecciona la opción **Enlace Serial** del menú **Ejecución**. Al seleccionar esta opción se abre una ventana como la que se muestra en la Figura V.21.

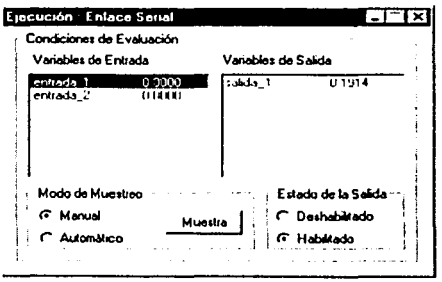


Figura V.21. Pantalla de Ejecución del enlace serial.

Este diálogo, al igual que en la prueba de depuración, muestra dos listas: la lista izquierda muestra todas las variables de entrada y la lista derecha muestra todas las variables de salida. Todas las variables mostradas en el diálogo corresponden sólo a variables que se encuentran vinculadas los bloques de reglas que están habilitados para ejecución.

La sección *Modo de Muestreo* determina el tiempo en que será solicitada información via comunicación serial de todas y cada una de las variables que contengan un puerto de comunicación válido. El modo por omisión es Manual. Este modo cuenta con el botón **Muestra**, el cual, cada vez que es presionado solicita información, quedando temporalmente desactivado, hasta que la información que corresponde a las variables de entrada es obtenida. En el modo **Automático** la información de las variables es solicitada a intervalos determinados. La definición del intervalo se lleva a cabo por medio de una barra de deslizamiento y una caja de edición. Los intervalos de muestreo permitidos van de 2 a 60 segundos.

Una vez recibidos los valores de todas las variables de entrada, inmediatamente se evalúa el sistema difuso y el valor de las variables de salida es actualizado. El nuevo valor o estado de las variables de salida podrá ser enviado a su correspondiente puerto de comunicación dependiendo de si el estado de la salida esta habilitado. Esto se establece en la sección *Estado de la salida* con las opciones **Deshabilitado y Habilitado.**

Este modo de ejecución presenta en la parte inferior derecha una ventana denominada **Monitor de Puertos**, mostrada en la Figura V.22, el **Monitor** es el encargado de supervisar toda la actividad relacionada con los puertos de comunicación serial. Éste realiza las siguientes tareas:

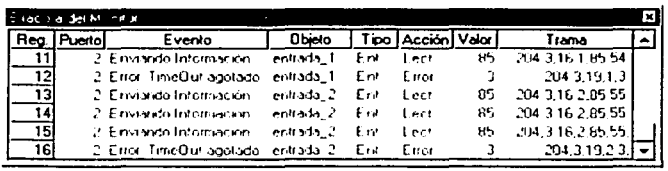
- Inicializa la actividad del puerto
- Presenta información relevante del estado del puerto como es:
  - Estado: indica si actualmente se puede enviar información por ese puerto
  - Transmisión: despliega la última trama de información enviada por el puerto.
  - Recepción: despliega la última trama de información recibida por el puerto.
  - Ultimo evento: indica el último evento ocurrido en el puerto.
- Solicita información de los objetos direcciones de memoria a intervalos determinados o a solicitud expresa.
- Informa y lleva a cabo las tareas correspondientes cuando surgen errores en la comunicación.
- Informa y responde a las acciones y eventos asociados a los objetos direcciones de memoria.
- Crea la Bitácora
- Inspecciona a intervalos el estado del puerto para determinar si no se ha perdido la comunicación cuando transcurren largos tiempos de inactividad.

Monitor de Puertos

Puerto	Estado	Transmisión	Recepción	Ultimo evento
Com1	Cerrado	...	...	
Com2	Abierto	204.3.16.2.	204.3.19.2.3.	Error TimeOut agot
Com3	Cerrado	...	...	...
Com4	Cerrado	...	...	...

Figura V.22 Monitor de Puertos

El monitor cuenta con una bitácora (Figura V.23), la cual es una ventana que registra toda la actividad en los puertos de comunicación y muestra la secuencia en que es procesada la información. Esta bitácora es muy utilizada para detectar posibles errores en la comunicación.



Reg.	Puerto	Evento	Objeto	Tipo	Acción	Valor	Trama
11	2	Enviando Información	entrada_1	Ent	Lect	85	204 3,16 1,85 54
12	2	Error TimeOut agotado	entrada_1	Ent	Error	3	204 3,19,1,3
13	2	Enviando Información	entrada_2	Ent	Lect	85	204 3,16 2,85 55
14	2	Enviando Información	entrada_2	Ent	Lect	85	204 3 16 2,85 55
15	2	Enviando Información	entrada_2	Ent	Lect	85	204,3 16,2 85 55
16	2	Error TimeOut agotado	entrada_2	Ent	Error	3	204,3,19,2 3,

Figura V.23 Bitácora del Monitor de Puertos

Después de concluida una prueba de este tipo, viene una etapa de análisis de resultados en donde toda la información generada por el sistema debe ser clasificada y examinada a detalle, para determinar la autenticidad del modelo.

En el Apéndice 1, se brindan algunas recomendaciones y sugerencias para el modelado de sistemas difusos, así como también, una serie de métodos que son de gran utilidad para determinar la validez de los resultados o detectar posibles fallas en el sistema.

CAPITULO VI.

ALGORITMO DE CONTROL DIFUSO.

VI.1 Introducción.

En este capitulo se presenta los algoritmos difusos introducidos para el sistema de control, cada uno de los algoritmos fue programado en el Software FUC, además se presentan las interconexiones entre cada uno de los bloques de manera que el sistema completo es un sistema retroalimentado, el programa implementado es solo para la rutina de lavado. Se deben indicar los parámetros de lavado: Cantidad de Ropa, Tipo de Ropa y Grado de suciedad.

Con los parámetros iniciales, se calcula la cantidad de agua necesaria, la temperatura de lavado, la cantidad de detergente y el tiempo de lavado.

El primer bloque de control es para llenar la tina, aquí las variables del control son la Diferencia de Nivel de Agua que es igual al Nivel de Agua Deseado menos el Nivel de Agua Real (Medida por el sensor de presión) y la Diferencia de Temperatura que es igual a la Temperatura de Lavado Deseado menos la Temperatura de Agua Real (Medida por el sensor de Temperatura).

Una vez llenada la tina, se agrega el detergente indicado, esta Cantidad de Detergente es la calculada con los parámetros iniciales pero puede cambiar dependiendo de como este avanzando el proceso de lavado, esto es, es posible agregar más detergente si todo el que fue agregado ya no tiene efecto sobre el lavado de la ropa, para medir la cantidad de detergente disuelto en el agua nos ayudamos de la lectura del sensor de conductividad.

Al iniciar el proceso de lavado, esto es, cuando se comienza a agitar, se miden las lecturas de sensor de conductividad y de Turbiedad, este último nos ayuda a decidir si el agua de lavado se encuentra muy sucia (Turbia) y sigue incrementado, entonces hay que desalojar una cantidad de agua, pero si tiramos agua, entonces tenemos que agregar más agua limpia y por lo tanto más detergente, por lo que el sistema de llenado se vuelve a activar de manera que no cambien las condiciones de lavado.

El tiempo de lavado calculado con los parámetros iniciales nos sirve solamente como parámetro de referencia, si la rutina de lavado alcanza este valor de máximo, solamente se indica al usuario que si quiere parar el proceso, si no es así, la programa de lavado continua y se sigue con la agitación hasta que las condiciones de conductividad y turbiedad son las adecuadas para detener el proceso. Estas condiciones están dadas por las variaciones de turbiedad, cuando ya no existen cambios en la turbiedad, esto es se mantiene estable por un tiempo, y la conductividad se encuentra en un valor óptimo, entonces el control por sí solo determina que hay que detener el proceso de lavado.

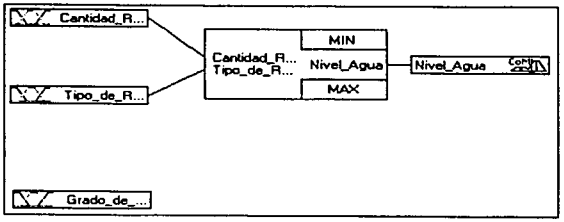
VI.2 Asignación del nivel de agua.

En este control difuso se asigna el nivel de agua que se utilizará en el lavado las variables de las que depende la cantidad de agua son: Cantidad de Ropa y Tipo de Ropa, esta cantidad de agua es muy importante ya que tenemos que tener el agua suficiente para que la ropa se lave correctamente y que no se maltrate, además no podemos poner una cantidad mayor de agua que la necesaria en primera porque se tiene un desperdicio, en segunda porque tendremos que aplicar más fuerza mecánica para poder mover el agua y la ropa, por último si tenemos una cantidad de agua mayor se tiene que agregar más detergente para que se mantengan las condiciones de lavado adecuadas.

VI.2.1 Estructura del Sistema.

La estructura del sistema identifica el flujo de la inferencia lógica difusa desde las variables de entrada hasta las variables de salida. La Fuzzificación en la interfaz de las entradas convierte las entradas analógicas en valores difusos. La inferencia difusa toma lugar en el bloque de reglas el cual contiene las reglas lingüísticas de control. Las salidas de este bloque de reglas son variables lingüísticas. La defuzzificación en la interfaz de la salida pasa estas variables lingüísticas en variables analógicas nuevamente.

La siguiente Figura VI.1 muestra la estructura de este sistema difuso incluyendo la interfaz de entrada, bloque de reglas e interfaz de salida. Las líneas de interconexión simbolizan el flujo de datos.



FiguraVI.1.: Estructura del Sistema de Lógica Difusa.

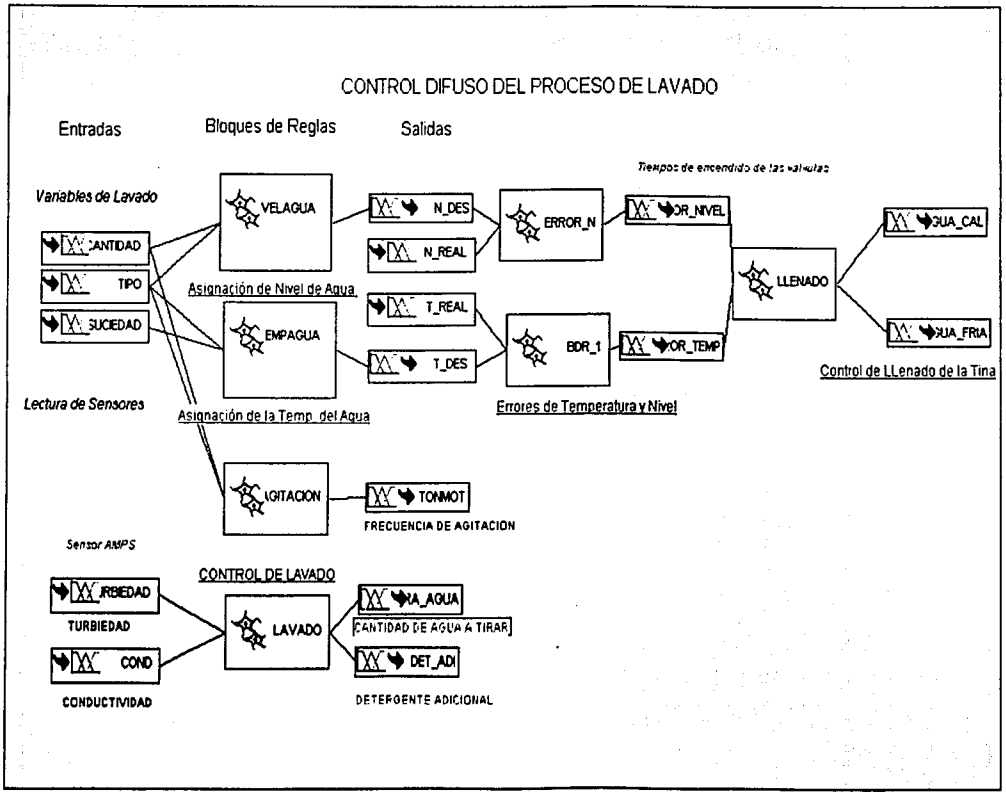


Fig. VI.2 Diagrama de bloques del sistema de control difuso para lavado, programado en el software FUC.

En la Figura VI.2 se presenta un sistema de bloques del control de lavado, en el se tienen las entradas que tienen que proporcionar el usuario, como es: Tipo de Ropa, Cantidad de Ropa, Grado de Suciedad, el primer bloque asigna el nivel de agua y Temperatura de lavado, posteriormente se calculan el error de nivel y error de Temperatura, para después generar los tiempos de apertura de las válvulas de admisión. Por otra parte se calcula la frecuencia de agitación que depende de la Cantidad y Tipo de Ropa. El último bloque realiza el control de Detergente y Cantidad de Agua que hay que desalojar para mantener una Turbiedad correcta, estos parámetros dependen de la Conductividad y la Turbiedad del agua. En las siguientes secciones se presenta una descripción de cada uno de los bloques utilizados.

VI.2.2 Variables Lingüísticas

En esta sección definimos todas las variables lingüísticas y todas las funciones de membresía. Las variables lingüísticas son usadas para trasladar valores reales en valores lingüísticos. Los posibles valores de una variable lingüística no son números, pero son llamados "términos lingüísticos".

Las variables lingüísticas tienen que ser definidas para todas las entradas salidas y variables intermedias. Las funciones de membresía son definidas usando solo pocos puntos de definición.

La Tabla VI.1 lista todas las variables del sistema y sus respectivos términos.

Nombre de la Variable	Nombre de los Términos
Cantidad_Ropa	Poca, Regular, Mucha, Llena
Grado_de_Sucieda	Poco_Sucia, Regular_Sucia, Muy_Sucia, Extra_Sucia
Tipo_de_Ropa	Sint_Blancos, Sint_Color, Del_Blancos, Del_Color, Alg_Blancos, Alg_Color
Nivel_Agua	Muy_Pequeno, Pequeno, Medio, Grande, Muy_Grande

Tabla VI.1. Variables Lingüísticas.

Las propiedades de todas las variables base son listadas en las Tabla VI.2 que se muestra a continuación.

Nombre de la Variable	Min	Max	Default	Unidad
Cantidad_Ropa	0	7	0.5	Kg
Grado_de_Sucieda	0	1	0.5	Sin Unidades
Tipo_de_Ropa	0	5	0	Sin Unidades
Nivel_Agua	20	40	20	Cms

Tabla VI.2 Variables Base

Nota: El valor por default para una variable es utilizado sólo al comienzo de la ejecución del modelo.

VI.2.2.1 Variable de entrada "Cantidad_Ropa"

La variable de entrada Cantidad de Ropa esta dividida en cuatro funciones de membresía como lo muestra la Figura VI.3.

En la Tabla VI.3 se definen los puntos de cada una de las funciones de membresía dadas para la variable.

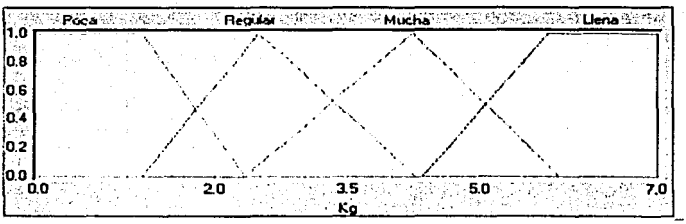


Figura VI.3. Función de Membresía de "Cantidad_Ropa"

Nombre del Término	Forma	Definición de puntos (x, y)		
Poca	Lineal	(0, 1) (7, 0)	(1.16662, 1)	(2.33325, 0)
Regular	Lineal	(0, 0) (4.281, 0)	(1.16662, 0) (7, 0)	(2.48762, 1)
Mucha	Lineal	(0, 0) (5.90088, 0)	(2.33325, 0) (7, 0)	(4.22312, 1)
Llena	Lineal	(0, 0) (7, 1)	(4.33887, 0) (7, 0)	(5.78512, 1)

Tabla VI.3 Definición de puntos de la Función de Membresía "Cantidad_Ropa"

VI.2.2.2 Variable de entrada "Grado_de_Suciedad"

Este parámetro, es asignado inicialmente por el usuario, pero después el sensor de turbiedad y conductividad nos dará información acerca del valor para esta variable.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Poco_Sucia	Lineal	(0, 1) (1, 0)	(0.18182, 1)	(0.39256, 0)
Regular_Sucia	Lineal	(0, 0) (0.62396, 0)	(0.16666, 0) (1, 0)	(0.39256, 1)
Muy_Sucia	Lineal	(0, 0) (0.83884, 0)	(0.3967, 0) (1, 0)	(0.59504, 1)
Extra_Sucia	Lineal	(0, 0) (1, 1)	(0.61984, 0) (1, 0)	(0.79752, 1)

Tabla VI.4 Definición de puntos de la Función de Membresía "Grado_de_Sucieda"

VI.2.2.3 Variable de entrada "Tipo_de_Ropa"

Esta variable es asignada por el usuario y puede ser:

1 Algodón Blanco
2 Algodón Color
3 Delicado Blanco
4 Delicado Color
5 Sintético Blanco
6 Sintético Color

Esta no es una variable analógica sino que dependiendo del tipo de ropa seleccionado, se asigna un valor constante a dicha variable como se muestra en la Figura VI.3 este valor es constante durante todo el proceso de lavado y depende del tipo de ropa que se tenga.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Sint_Blancos	Lineal	(0, 1) (5, 0)	(0.124, 1)	(0.124, 0)
Sint_Color	Lineal	(0, 0) (1.095, 0)	(0.9504, 0) (5, 0)	(1, 1)
Del_Blancos	Lineal	(0, 0) (2.0455, 0)	(1.9628, 0) (5, 0)	(1.9835, 1)
Del_Color	Lineal	(0, 0) (3.0372, 0)	(2.9339, 0) (5, 0)	(3, 1)
Alg_Blancos	Lineal	(0, 0) (4.0496, 0)	(3.9463, 0) (5, 0)	(4, 1)
Alg_Color	Linear	(0, 0) (5, 1)	(4.876, 0)	(4.876, 1)

Tabla VI.5 Definición de puntos de la Función de Membresía "Tipo_de_Ropa"

VI.2.2.4 Variable de Salida "Nivel_Agua"

Cantidad de Agua que se asigna para el proceso de lavado, este parámetro depende de la cantidad de ropa y del tipo de ropa.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Muy_Pequeno	Lineal	(20, 0) (40, 0)	(23.3335, 1)	(26.6665, 0)
Pequeno	Lineal	(20, 0) (29.9995, 0)	(23.3335, 0) (40, 0)	(26.6665, 1)
Medio	Lineal	(20, 0) (33.3325, 0)	(26.6665, 0) (40, 0)	(29.9995, 1)
Grande	Lineal	(20, 0) (36.666, 0)	(29.9995, 0) (40, 0)	(33.3325, 1)
Muy_Grande	Lineal	(20, 0) (40, 0)	(33.3325, 0)	(36.666, 1)

Tabla VI.6. Definición de puntos de la Función de Membresía "Nivel_Agua"

VI.2.2.3 Bloque de Reglas

El bloque de reglas contiene la estrategia de control de un sistema de lógica difusa. Cada bloque encierra todas las reglas en un mismo contexto. Un contexto esta definido por las mismas variables de entrada y salida de las reglas.

Las parte "si" (if) de las reglas describen la situación, para el cual las reglas están diseñadas. La parte "entonces" (then) describe la respuesta del sistema difuso en esta situación. El grado de soporte (DoS) es usado como peso de cada regla de acuerdo a su importancia.

El procesamiento de las reglas comienza calculando la parte "si" (if). Los métodos utilizados en el proceso de evaluación se encuentran ligados a cada una de las variables de salida. Los métodos disponibles son MIN-MAX, ADITIVO.

VI.2.3.1 Bloque de Reglas "Control de Nivel de Agua"

Este control sirve para asignar el Nivel de Agua que se utilizara durante el proceso de lavado, esta cantidad de Agua no cambia y pasa después al control de temperatura y de llenado. Aquí se determina la Cantidad de Agua con la que se llenará la Tina.

La Cantidad de Agua depende del Tipo de Ropa y de la Cantidad de Ropa y la tabla del bloque de reglas se muestra en la Tabla VI.7. y la gráfica de control en la Figura VI.4.

SI		ENTONCES	
Cantidad_Ropa	**Tipo_de_Ropa**	**DoS**	**Nivel_Agua**
Poca	Sint_Blancos	1.00	Muy_Pequeno
Poca	Sint_Color	0.50	Muy_Pequeno
Poca	Sint_Color	0.50	Pequeno
Poca	Del_Blancos	1.00	Pequeno
Poca	Del_Color	0.50	Pequeno
Poca	Del_Color	0.50	Medio
Poca	Alg_Blancos	1.00	Medio
Poca	Alg_Color	0.50	Medio
Poca	Alg_Color	0.50	Grande
Regular	Sint_Blancos	1.00	Pequeno
Regular	Sint_Color	0.50	Pequeno
Regular	Sint_Color	0.50	Medio
Regular	Del_Blancos	1.00	Medio

Regular	Del_Color	0.50	Medio
Regular	Del_Color	0.50	Grande
Regular	Alg_Blancos	1.00	Grande
Regular	Alg_Color	0.50	Grande
Regular	Alg_Color	0.50	Muy_Grande
Mucha	Sint_Blancos	1.00	Medio
Mucha	Sint_Color	0.50	Medio
Mucha	Sint_Color	0.50	Grande
Mucha	Del_Blancos	1.00	Grande
Mucha	Del_Color	0.50	Grande
Mucha	Del_Color	0.50	Muy_Grande
Mucha	Alg_Blancos	1.00	Muy_Grande
Mucha	Alg_Color	1.00	Muy_Grande
Llena	Sint_Blancos	1.00	Grande
Llena	Sint_Color	0.50	Muy_Grande
Llena	Del_Blancos	1.00	Muy_Grande
Llena	Del_Color	1.00	Muy_Grande
Llena	Alg_Blancos	1.00	Muy_Grande
Llena	Alg_Color	1.00	Muy_Grande

Tabla VI 7 Reglas del bloque "Control de Nivel de Agua"

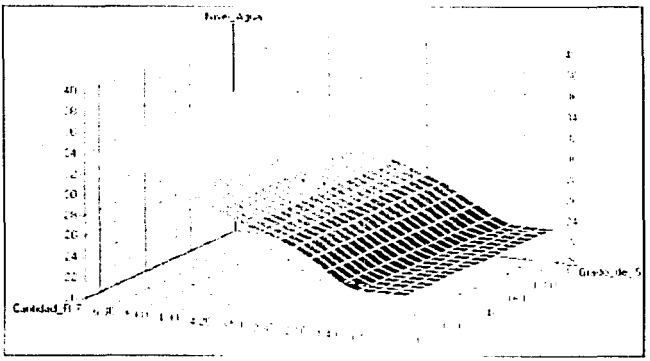


Figura VI.4 Curva de control de Nivel de Agua

VI.3 Asignación de la Temperatura de Lavado.

En la asignación de temperatura (Figura VI.5), se obtiene el valor más adecuado del agua para realizar el proceso de lavado, este valor depende del tipo de ropa y del grado de suciedad.

Debemos de tener en cuenta que la temperatura del agua puede dañar algunas prendas y que cierto tipo de suciedad solamente se desprende a una temperatura específica. Las prendas de color deben de lavarse a una temperatura menor que las de color para que no pierda su color y que una prenda delicada soporta menos temperatura que una de algodón.

V.3.1 Estructura del sistema

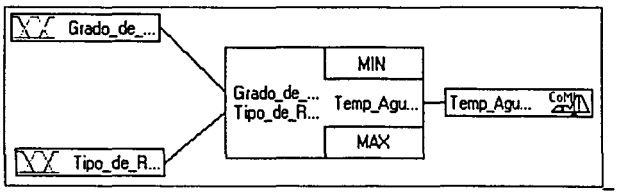


Figura VI.5. Estructura del sistema de Lógica Difusa

VI.3.2 Variables Lingüísticas

Nombre de la Variable	Nombre de los términos
Grado_de_Sucieda	Poco_Sucia, Regular_Sucia, Muy_Sucia, Extra_Sucia
Tipo_de_Ropa	Sint_Blancos, Sint_Color, Del_Blancos, Del_Color, Alg_Blancos, Alg_Color
Temp_Agua_Des	Muy_Fria, Fria, Tibia, Caliente, Muy_Caliente

Tabla VI.8 Variables Lingüísticas

Las propiedades de todas las variables base están listadas en la Tabla. VI.9.

Nombre de las Variables	Min	Max	Default	Unidades
Grado_de_Suciedad	0	1	0.5	Sin Unidades
Tipo_de_Ropa	0	5	0	Sin Unidades
Temp_Agua_Des	20	60	20	Grados Centigrados

Tabla VI.9 Propiedades de las Variables Base.

VI.3.2.1 Variable de Salida "Temp_Agua_Des".

Temperatura del Agua Deseada que se asigna para el proceso de lavado, este parámetro depende del grado de suciedad y del tipo de ropa.

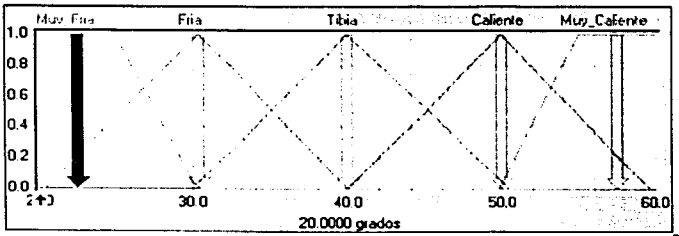


Figura VI.6 Función de Membresía de "Temp_Agua_Des"

Nombre del Término	Forma	Definicion de puntos		
Muy_Fria	Lineal	(20, 1) (60, 0)	(25, 1)	(30.248, 0)
Fria	Lineal	(20, 0) (39.999, 0)	(20.331, 0) (60, 0)	(30.413, 1)
Tibia	Lineal	(20, 0) (50.413, 0)	(29.917, 0) (60, 0)	(39.999, 1)
Caliente	Lineal	(20, 0) (59.669, 0)	(39.999, 0) (60, 0)	(49.917, 1)
Muy_Caliente	Lineal	(20, 0) (60, 1)	(49.752, 0)	(55, 1)

Tabla VI.10 Definicion de Puntos de la Funcion de Membresia "Temp_Agua_Des"

VI.3.3 Bloque de Reglas

Bloque de Reglas "Asig_Temp"

Este control sirve para asignar la Temperatura del Agua que se utilizara durante el proceso de lavado, el control mantendrá este valor de temperatura de manera que no cambien las condiciones de lavado, esta variable también pasa al control de temperatura y de llenado.

SI		ENTONCES	
Grado_de_Sucieda	**Tipo_de_Ropa**	**DoS**	**Temp_Agua_Des**
Poco_Sucia	Sint_Blancos	0.60	Caliente
Poco_Sucia	Sint_Blancos	0.40	Muy_Caliente
Poco_Sucia	Sint_Color	0.60	Muy_Fria
Poco_Sucia	Sint_Color	0.40	Fria
Poco_Sucia	Del_Blancos	1.00	Fria
Poco_Sucia	Del_Color	1.00	Muy_Fria
Poco_Sucia	Alg_Blancos	0.60	Muy_Fria
Poco_Sucia	Alg_Blancos	0.40	Fria
Poco_Sucia	Alg_Color	1.00	Muy_Fria
Regular_Sucia	Alg_Color	1.00	Muy_Fria
Muy_Sucia	Alg_Color	1.00	Muy_Fria
Extra_Sucia	Alg_Color	1.00	Muy_Fria
Extra_Sucia	Del_Color	1.00	Muy_Fria
Muy_Sucia	Del_Color	1.00	Muy_Fria
Regular_Sucia	Del_Color	1.00	Muy_Fria
Regular_Sucia	Sint_Blancos	0.60	Caliente
Regular_Sucia	Sint_Blancos	0.40	Muy_Caliente
Regular_Sucia	Sint_Color	0.20	Muy_Fria
Regular_Sucia	Sint_Color	0.80	Fria
Regular_Sucia	Del_Blancos	0.50	Fria
Regular_Sucia	Del_Blancos	0.50	Tibia
Regular_Sucia	Alg_Blancos	0.20	Muy_Fria
Regular_Sucia	Alg_Blancos	0.80	Fria
Muy_Sucia	Sint_Blancos	0.20	Caliente
Muy_Sucia	Sint_Blancos	0.20	Muy_Caliente
Muy_Sucia	Sint_Color	1.00	Fria

Muy_Sucia	Del_Blancos	0.70	Tibia
Muy_Sucia	Del_Blancos	0.30	Fria
Muy_Sucia	Alg_Blancos	1.00	Fria
Extra_Sucia	Sint_Blancos	1.00	Muy_Caliente
Extra_Sucia	Sint_Color	1.00	Fria
Extra_Sucia	Del_Blancos	0.90	Tibia
Extra_Sucia	Del_Blancos	0.10	Fria
Extra_Sucia	Alg_Blancos	1.00	Fria

Tabla VI.11 Reglas del control difuso "Asig_Temp"

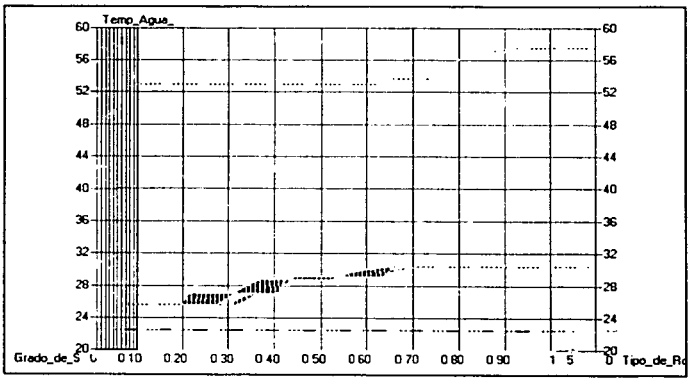


Figura VI.7 Curva de Control "Asignación de Temperatura"

VI.4 Control de llenado de la tina.

Este control se encarga del llenado de la tina para el proceso de lavado, aquí lo que se mide es la diferencia de Nivel de Agua y la Diferencia de Temperatura, el control se encarga de encender y apagar las válvulas de admisión de manera que cuando se tenga el nivel de Agua deseado también se tenga la temperatura que se asigno en el control anterior. Cabe señalar que es importante mantener la temperatura asignada en el nivel correcto ya que si esta es errónea cambian todas las condiciones de lavado.

VI.4.2 Estructura del Sistema

El sistema tiene dos entradas: Diferencia de Nivel de Agua y Diferencia de la Temperatura del Agua, las cuales vienen directamente de la tarjeta de control y pasan al programa de lógica difusa para el control de estas.

Las salidas son los tiempo de encendido de las válvulas de admisión: Válvula de Agua Fría y Válvula de agua Caliente.

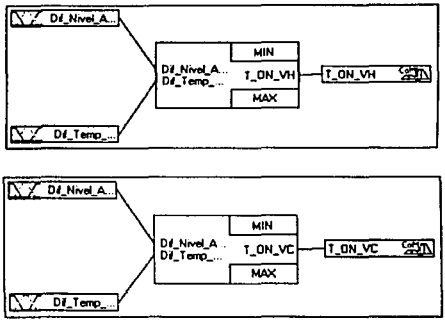


Figura VI.8 Estructura del sistema de Control de Llenado de la Tina a) Control para el tiempo de encendido de la válvula de agua caliente, b) Control para el tiempo de encendido de la válvula de agua fría.

Variable Lingüísticas

La siguiente tabla muestra el nombre de las variables lingüísticas y los términos que se utilizaron para este control, también se muestra una tabla con el nombre de las Variables Base así como el método de fuzzificación y defuzzificación.

Nombre de la Variable	Nombre de los Términos
Dif_Nivel_Agua	Muy_Pequeno, Pequeno, Medio, Grande, Muy_Grande
Dif_Temp_Agua	Negativa_Grande, Negativa_Media, Zero, Positiva_Media, Positiva_Grande
T_ON_VH	Muy_Corto, Corto, Regular, Largo, MuyLargo
T_ON_VC	Muy_Corto, Corto, Regular, Largo, MuyLargo

Tabla VI.12 Variables Lingüísticas

Nombre de la Variable	Min	Max	Default	Unidades
Dif_Nivel_Agua	20	40	20	Cms
Dif_Temp_Agua	-40	40	0.5	Grados Centígrados
T_ON_VH	0	5	0.5	Segundos
T_ON_VC	0	5	0.5	Segundos

Tabla VI.13 Propiedades de las Variables Lingüísticas.

VI.4.2.1 Variable de Entrada "Dif_Nivel_Agua"

Esta variable se obtiene de la siguiente relación:

Dif_Nivel_Agua = Nivel de agua Deseado - Nivel de agua real

En realidad es el error de cantidad de agua, el nivel de agua deseado se asigna por el control difuso y el nivel de agua real es medido por el sensor de presión a través del convertidor A/D de la tarjeta de control.

Aquí lo que tenemos es un control en lazo cerrado donde tenemos una señal de referencia en este caso el nivel de agua deseado y una señal de error que es la diferencia de Nivel de Agua.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Muy_Pequeno	Lineal	(0, 1) (60, 0)	(0.992, 1)	(5, 0)
Pequeño	Lineal	(0, 0) (15, 0)	(0.992, 0) (60, 0)	(9.669, 1)
Medio	Lineal	(0, 0) (35, 0)	(5.95, 0) (60, 0)	(20.331, 1)
Grande	Lineal	(0, 0) (53.802, 0)	(16.364, 0) (60, 0)	(35.207, 1)
Muy_Grande	Lineal	(0, 0) (60, 1)	(39.998, 0)	(49.998, 1)

Tabla VI.14 Definición de Puntos de la Función de Membresía "Dif_Nivel_Agua"

VI.4.2.2 Variable de Entrada "Dif_Temp_Agua"

La diferencia de temperatura, esta definida como :

Dif_Temp_Agua = Temperatura Deseada - Temperatura Real.

La Temperatura Deseada es asignada por el control difuso y la Temperatura Real es medida por el sensor de Temperatura AMPS de Honeywell a través del Puerto Serial de Comunicación de la PC.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Negativa_Grande	Lineal	(-40, 1) (40, 0)	(-26.6675, 1)	(-13.335, 0)
Negativa_Media	Lineal	(-40, 0) (- 0.0025, 0)	(-26.6675, 0) (40, 0)	(-13.335, 1)
Zero	Lineal	(-40, 0) (10, 0)	(-10, 0) (40, 0)	(-0.0025, 1)
Positiva_Media	Lineal	(-40, 0) (26.665, 0)	(-0.0025, 0) (40, 0)	(13.33, 1)
Positiva_Grande	Lineal	(-40, 0) (40, 1)	(13.33, 0)	(26.665, 1)

Tabla VI.15 Definición de Puntos de la Función de Membresía "Dif_Temp_Agua"

VI.4.2.3 Variable de salida "T_ON_VH"

Tiempo que permanece encendida la válvula de admisión de Agua Caliente para el llenado de la tina, su tiempo va de 0 a 5 segundos, dependiendo de la temperatura y del nivel de agua que se tenga en la tina.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Muy_Corto	Lineal	(0, 1) (5, 0)	(0.1033, 1)	(0.5165, 0)
Corto	Lineal	(0, 0) (1, 0)	(0.0413, 0) (5, 0)	(0.4752, 1)
Regular	Lineal	(0, 0) (2.4793, 0)	(0.5785, 0) (5, 0)	(1.5496, 1)
Largo	Lineal	(0, 0) (4.1665, 0)	(1.5, 0) (5, 0)	(2.7479, 1)
MuyLargo	Lineal	(0, 0) (5, 1)	(3.0579, 0)	(4.1665, 1)

Tabla VI.16 Definición de Puntos de la "T_ON_VH"

VI.4.2.4 Variable de salida "T_ON_VC"

Tiempo que permanece encendida la válvula de admisión de Agua Fría para el llenado de la tina, su tiempo va de 0 a 5 segundos, dependiendo de la temperatura y del nivel de agua que se tenga en la tina.

Nombre del Término	Forma	Definición de Puntos (x, y)		
Muy_Corto	Lineal	(0, 1) (5, 0)	(0.1033, 1)	(0.5165, 0)
Corto	Lineal	(0, 0) (1, 0)	(0.0413, 0) (5, 0)	(0.4752, 1)
Regular	Lineal	(0, 0) (2.4793, 0)	(0.5785, 0) (5, 0)	(1.5496, 1)
Largo	Lineal	(0, 0) (4.1665, 0)	(1.5, 0) (5, 0)	(2.7479, 1)
MuyLargo	Lineal	(0, 0) (5, 1)	(3.0579, 0)	(4.1665, 1)

Tabla VI.17 Definición de Puntos de la Función de Membresía "T_ON_VC"

VI.4.3 Bloque de Reglas

VI.4.3.1 Bloque de Reglas "Válvula de Agua Caliente"

Este bloque contiene todas las reglas y las relaciones entre las entradas y la salida para el control de Tiempo de encendido de la Válvula de admisión de agua Caliente.

SI		ENTONCES	
Dif_Nivel_Agua	Dif_Temp_Agua	DoS	T_ON_VH
Muy_Pequeno	Negativa_Grande	0.50	Muy_Corto
Muy_Pequeno	Negativa_Media	0.50	Muy_Corto
Muy_Pequeno	Zero	0.50	Corto
Muy_Pequeno	Positiva_Media	0.50	Regular
Muy_Pequeno	Positiva_Grande	0.50	Largo
Pequeño	Negativa_Grande	0.50	Muy_Corto
Pequeño	Negativa_Media	0.50	Corto
Pequeño	Zero	0.50	Regular
Pequeño	Positiva_Media	0.50	Largo
Pequeño	Positiva_Grande	0.50	MuyLargo
Medio	Negativa_Grande	0.50	Corto
Medio	Negativa_Media	0.50	Regular

Medio	Zero	0.50	Largo
Medio	Positiva_Media	0.50	MuyLargo
Medio	Positiva_Grande	0.50	MuyLargo
Grande	Negativa_Grande	0.50	Regular
Grande	Negativa_Media	0.50	Largo
Grande	Zero	0.50	MuyLargo
Grande	Positiva_Media	0.50	MuyLargo
Grande	Positiva_Grande	0.50	MuyLargo
Muy_Grande	Negativa_Grande	0.50	Regular
Muy_Grande	Negativa_Media	0.50	Largo
Muy_Grande	Zero	0.50	MuyLargo
Muy_Grande	Positiva_Media	0.50	MuyLargo
Muy_Grande	Positiva_Grande	0.50	MuyLargo

Tabla VI.18 Reglas del Bloque de control "T_ON_VH"

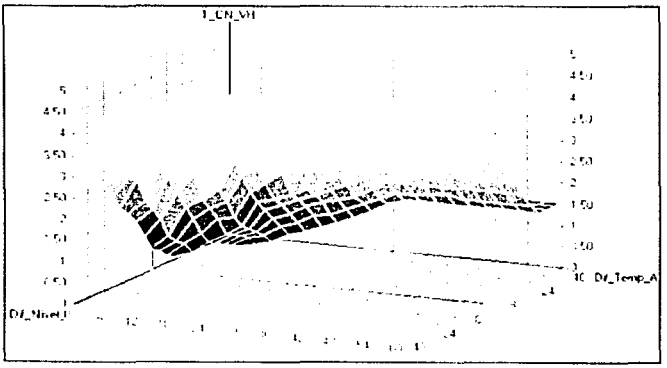


Figura VI.9 Curva del control "T_ON_VH"

VI.4.3.2 Bloque de Reglas "Válvula de Agua Fria"

Este bloque contiene todas las reglas y las relaciones entre las entradas y la salida para el control de Tiempo de encendido de la Válvula de admisión de agua Fria.

SI		ENTONCES	
Dif_Nivel_Agua	**Dif_Temp_Agua**	**DoS**	**T_ON_VC**
Muy_Pequeño	Negativa_Grande	0.50	Largo
Muy_Pequeño	Negativa_Media	0.50	Regular
Muy_Pequeño	Zero	0.50	Corto
Muy_Pequeño	Positiva_Media	0.50	Muy_Corto
Muy_Pequeño	Positiva_Grande	0.50	Muy_Corto
Pequeño	Negativa_Grande	0.50	Regular
Pequeño	Negativa_Media	0.50	Largo
Pequeño	Zero	0.50	Regular
Pequeño	Positiva_Media	0.50	Corto
Pequeño	Positiva_Grande	0.50	Muy_Corto
Medio	Negativa_Grande	0.50	MuyLargo
Medio	Negativa_Media	0.50	MuyLargo
Medio	Zero	0.50	Largo
Medio	Positiva_Media	0.50	Regular
Medio	Positiva_Grande	0.50	Corto
Grande	Negativa_Grande	0.50	MuyLargo
Grande	Negativa_Media	0.50	MuyLargo
Grande	Zero	0.50	MuyLargo
Grande	Positiva_Media	0.50	Largo
Grande	Positiva_Grande	0.50	Regular
Muy_Grande	Negativa_Grande	0.50	MuyLargo
Muy_Grande	Negativa_Media	0.50	MuyLargo
Muy_Grande	Zero	0.50	MuyLargo
Muy_Grande	Positiva_Media	0.50	Largo
Muy_Grande	Positiva_Grande	0.50	Regular

Tabla VI.19 Reglas del bloque Control de control "T_ON_VC"

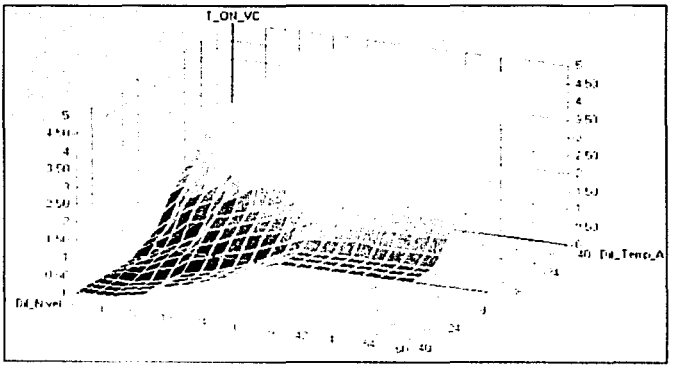


Figura VI.10 Curva del control "T_ON_VC"

VI.5 Frecuencia de agitación.

En esta etapa se asigna la frecuencia de agitación para el proceso de lavado, esta frecuencia de agitación, depende de la Cantidad de Ropa y del Tipo de Ropa que se vaya a lavar, le denominamos frecuencia de agitación, a las oscilaciones de vaivén que genera el motor para proporcionar la fuerza mecánica en el lavado, es importante que se tenga una relación con el tipo de ropa, ya que si aplicamos una mayor agitación podríamos provocar que las prendas se dañen. La relación que guarda la frecuencia de agitación es lineal con respecto a la cantidad de ropa.

VI.5.1 Estructura del Sistema

El sistema tiene como entradas Cantidad de Ropa y Tipo de Ropa, la salida es la frecuencia de agitación que es un parámetro que se envía a la tarjeta de control, para realizar la agitación en la lavadora.

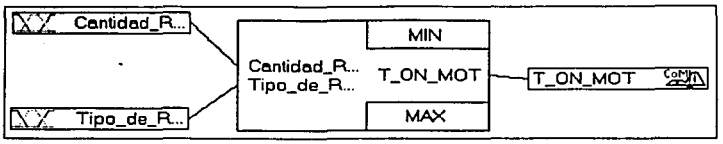


Figura VI.11 Estructura del Sistema de Lógica Difusa.

VI.5.2 Variable Lingüísticas

La siguiente tabla muestra el nombre de las variables lingüísticas y los términos que se utilizaron para este control, también se muestra una tabla con el nombre de las Variables Base así como el método de fuzzificación y defuzzificación.

Nombre de la Variable	Nombre de los Términos
Cantidad_Ropa	Poca, Regular, Media, Llena
Tipo_de_Ropa	Sint_Blancos, Sint_Color, Del_Blancos, Del_Color, Alg_Blancos, Alg_Color
T_ON_MOT	Muy_Bajo, Bajo, Medio, Alto, Muy_Alto

Tabla VI.20. Variables Lingüisticas

Nombre de la Variable	Min	Max	Default	Unidad
Cantidad_Ropa	0	7	0.5	Kg
Tipo_de_Ropa	0	5	0	Units
T_ON_MOT	0	10	5	OCS_MIN

Tabla VI.21 Propiedades de las Variables Liguisticas.

VI.5.2.1 Variable de salida "T_ON_MOT"

Frecuencia de Agitación del motor, este dato se envía a la tarjeta de control, para que realice el encendido y apagado alternadamente de cada uno de los dos embobinados del motor, para realizar el vaivén.

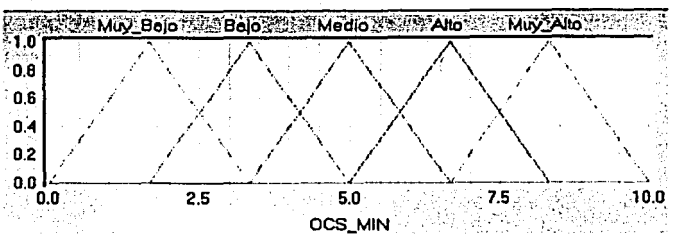


Figura VI.12 Funciones de Membresía

VI.5.3 Bloque de Reglas

IF		THEN	
Cantidad_Ropa	**Tipo_de_Ropa**	**DoS**	**T_ON_MOT**
Poca	Sint_Blancos	1.00	Bajo
Regular	Sint_Blancos	1.00	Bajo
Media	Sint_Blancos	1.00	Medio
Llena	Sint_Blancos	1.00	Medio
Poca	Sint_Color	1.00	Bajo
Regular	Sint_Color	1.00	Bajo
Media	Sint_Color	1.00	Medio
Llena	Sint_Color	1.00	Medio
Poca	Del_Blancos	1.00	Muy_Bajo
Regular	Del_Blancos	1.00	Muy_Bajo
Media	Del_Blancos	1.00	Bajo
Llena	Del_Blancos	1.00	Bajo
Poca	Del_Color	1.00	Muy_Bajo
Regular	Del_Color	1.00	Muy_Bajo
Media	Del_Color	1.00	Bajo
Llena	Del_Color	1.00	Bajo
Poca	Alg_Blancos	1.00	Alto
Regular	Alg_Blancos	1.00	Alto
Media	Alg_Blancos	1.00	Muy_Alto
Llena	Alg_Blancos	1.00	Muy_Alto
Poca	Alg_Color	1.00	Alto
Regular	Alg_Color	1.00	Alto
Media	Alg_Color	1.00	Muy_Alto
Llena	Alg_Color	1.00	Muy_Alto

Tabla VI 22 Reglas del Control Difuso "Frecuencia de Agitación"

VI.6 Control para tirar agua.

Como se explicó al inicio de este capitulo, el sistema tiene que ser capaz de determinar cuando el agua que se encuentra en la tina se encuentra muy sucia, para esto nos apoyamos del sensor de Turbiedad; que es un parámetro que nos da información a cerca de la suciedad del agua, con este parámetro podemos determinar cuanta agua debe ser desalojada de la tina, para que posteriormente el control de nivel del agua descrito anteriormente se encargue de llenar la tina nuevamente a su nivel adecuado.

VI.6.1 Estructura del Sistema

El sistema tiene como entradas la Turbiedad del Agua y la Conductividad del agua, la salida es la cantidad de agua que hay que tirar la cual pasa a la tarjeta de control, el cual se encarga de tirar el agua y agregar la misma que se tiro, pero con agua limpia.

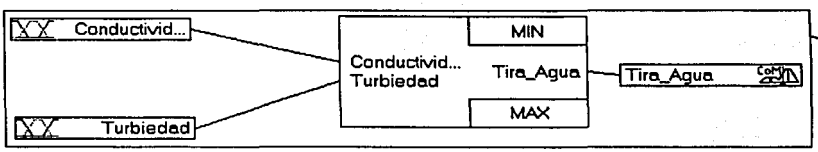


Figura VI.13 Estructura del Sistema de Lógica Difusa.

VI.6.2 Variable Lingüísticas

La siguiente tabla muestra el nombre de las variables lingüísticas y los términos que se utilizaron para este control, también se muestra una tabla con el nombre de las Variables Base así como el método de fuzzificación y defuzzificación.

Nombre de la Variable	Nombre de los Términos
Conductividad	Baja, Media, Grande
Turbiedad	Chica, Media, Grande
Tira_Agua	Poca, Regular, Mucha

Tabla VI.23. Variables Lingüísticas

Nombre de la Variable	Min	Max	Default	Unit
Conductividad	0	255	0.5	mS
Turbiedad	0	255	0.5	NTU
Tira_Agua	0	20	0.5	Litros

Tabla VI 24 Propiedades de las Variables Linguisticas

VI.6.2.1 Variable de salida "TiraAgua "

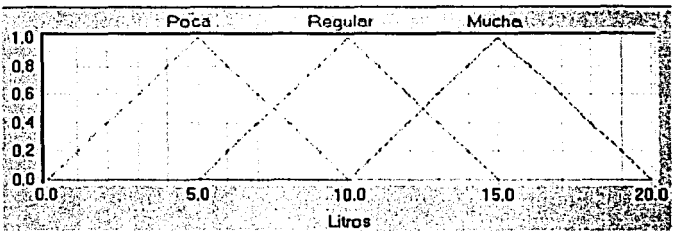


Figura VI 14 Funciones de Membresia de la Salida.

VI.6.3 Bloque de Reglas

IF		THEN	
Conductividad	Turbiedad	DoS	Tira_Agua
Baja	Chica	1.00	Poca
Baja	Media	1.00	Regular
Baja	Grande	1.00	Mucha
Media	Chica	1.00	Poca
Media	Media	1.00	Regular
Media	Grande	1.00	Mucha
Grande	Chica	1.00	Poca
Grande	Media	1.00	Regular
Grande	Grande	1.00	Mucha

Tabla VI 25 Reglas del Control Difuso "Tirar Agua".

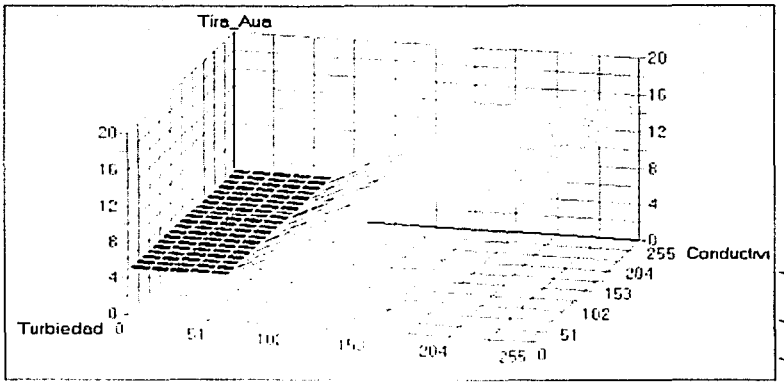


Figura VI.15 Curva de Control .

VI.7 Control para agregar detergente.

Cuando inicia la agitación, el detergente comienza a hacer efecto y éste con la suciedad comienza a perder sus propiedades de limpieza, además cuando se elimina agua sucia y se adiciona agua limpia es necesario agregar más detergente para poder mantener las condiciones de lavado óptimas. Para llevar a cabo el ajuste se utiliza este control; en nuestro caso solamente se indica en la pantalla la cantidad de detergente que hay que agregar, ya que no se cuenta con un dosificador de detergente.

VI.7.1 Estructura del Sistema

El sistema tiene como entradas la Turbiedad del Agua y la Conductividad del agua, la salida es la cantidad de detergente adicional a agregar.

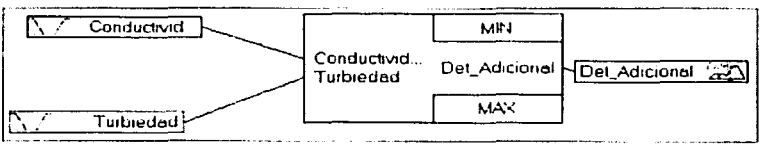


Figura VI.16 Estructura del Sistema para control de Detergente Adicional

VI.7.2 Variable Lingüisticas

La siguiente tabla muestra el nombre de las variables lingüísticas y los términos que se utilizaron para este control, también se muestra una tabla con el nombre de las Variables Base así como el método de fuzzificación y defuzzificación.

Nombre de la Variable	Nombre de Términos
Conductividad	Baja, Media, Grande
Turbiedad	Chica, Media, Grande
Det_Adicional	Bajo, Medio, Alto

Tabla VI.26 Variables Lingüísticas.

Nombre de la Variable	Min	Max	Default	Unidad
Conductividad	0	255	0.5	mS
Turbiedad	0	255	0.5	NTU
Det_Adicional	0	100	0.5	Gramos

Tabla VI.27 Propiedades de las Variables Lingüísticas.

VI.7.3 Bloque de Reglas

IF		THEN	
Conductividad	Turbiedad	DoS	Det_Adicional
Baja	Chica	1.00	Medio
Baja	Media	1.00	Bajo
Baja	Grande	1.00	Bajo
Media	Chica	1.00	Bajo
Media	Media	1.00	Medio
Media	Grande	1.00	Bajo
Grande	Chica	1.00	Bajo
Grande	Media	1.00	Bajo
Grande	Grande	1.00	Bajo

Tabla VI.28 Reglas del Control Difuso „Detergente Adicional".

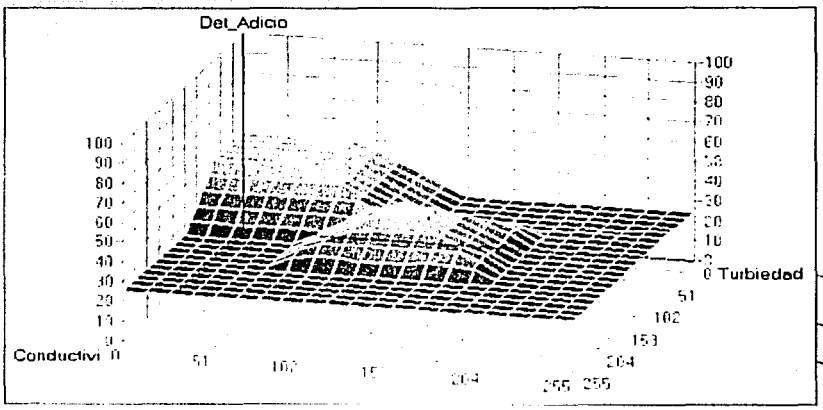


Figura VI 17 Curva de Control

Capitulo VII.

PRUEBAS REALIZADAS.

Las pruebas realizadas, son con condiciones de carga específica, se selecciono un tipo de ropa, se midió su peso y se decidió el grado de suciedad que contenía, estos son los parámetros principales que se deben de dar al control para que pueda trabajar:

Parámetros proporcionados por el usuario

- Cantidad de ropa. 0-7 Kg.
- Tipo de Ropa:
  - AB. Algodón Blancos
  - AC. Algodón Color
  - SB. Sintéticos Blancos
  - SC. Sintéticos Color
  - DB. Delicados Blancos.
  - DC. Delicados Color

Se seleccionaron los siguientes parámetros en el sistema de control en cada una de las siguientes opciones:

- Cantidad de Ropa:
  - Poca.
  - Regular.
  - Mucha.
  - Llena.
- Tipo de Ropa.
- Grado de Suciedad:
  - Poco Sucia
  - Regular Sucia
  - Muy Sucia
  - Extra Sucia

Una vez iniciado el proceso de lavado, el control asigna las siguientes variables:

- Cantidad de agua: 10- 60 litros
- Cantidad de Detergente: 10-250 gramos
- Temperatura de agua: 20-80 °C
- Tiempo de apertura de Válvula de Agua Caliente: 0-60 s
- Tiempo de apertura de Válvula de Agua Fría: 0-60 s
- Frecuencia de agitación 20-80 Osc/nmin

En la Figura VII.1 se muestra una foto del sistema real donde fueron hechas las pruebas, estas pruebas fueron hechas bajo condiciones reales de lavado.

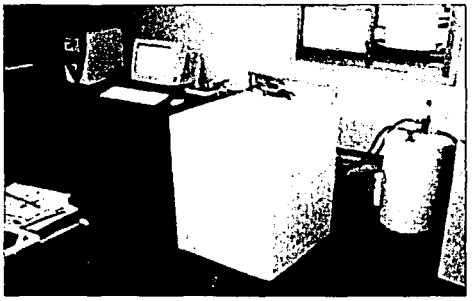


Figura VII.1 Sistema donde fueron realizadas las pruebas y probados los algoritmos de control

Los principales componentes de nuestro sistema son os siguientes:

- Computadora donde se encuentra cargado el programa de lógica difusa FUC y de comunicación serial con la tarjeta de control.
- Lavadora comercial de 7 Kgs. De eje vertical, con agitador, bomba de desagüe, válvulas de Admisión, solenoide, etc.
- Tarjeta de control, Figura VII.2 encargada de encender y apagar los actuadores, amplificación de las señales de los sensores, etc.
- Calentador eléctrico de agua, de 40 litros de capacidad, esto para poder elevar la temperatura del agua a 85 °C aprox. Y poder tener las condiciones para un proceso más real.
- Eliminador de burbujas Figura. VII.3, en el cual va montado el sensor de Turbiedad, Conductividad y temperatura, este sistema nos ayuda a eliminar las burbujas de aire que se tiene cuando se toma la lectura del sensor de Turbiedad, de tal manera que no nos arroje una lectura errónea.

Figura VII 2 Tarjeta de control

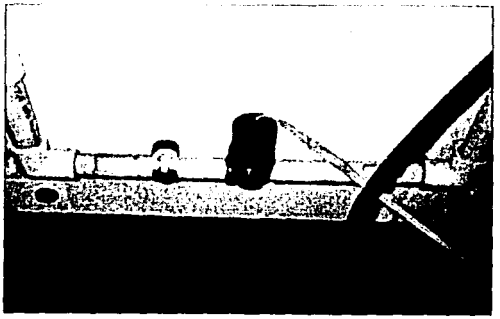


Figura VII 3 Sistema de eliminación de burbujas

Las pruebas realizadas se realizan bajo las siguientes condiciones.

- Ropa de uso común, esta se midió en una báscula para determinar la cantidad que se va ha lavar.
- El tipo de ropa se selecciono por sentido común y se clasifico dentro de una de las categorías anteriormente señaladas.

- El grado de suciedad es determinado por el usuario a simple vista.
- El detergente utilizado es ARIEL baja espuma, que es un detergente que se recomienda para lavadoras comerciales automáticas.
- Cabe señalar que no se utiliza cloro ni suavizantes durante la realización de las pruebas.

Proceso:

1. Se llena la tina abriendo intermitentemente cada una de las válvulas (Agua Fría y Agua Caliente) un tiempo determinado por el control, este tiempo depende del nivel de agua que se tiene y además de temperatura que se encuentra dentro de la tina, si la temperatura real se va acercando a la temperatura deseada asignada por el control, el tiempo de apertura se va modificando hasta que se alcanza la temperatura y el nivel de agua deseado.

2. Una vez llenada la tina, se inicia con la agitación la frecuencia con la que se realiza el vaivén del motor esta íntimamente ligada con el Tipo y Cantidad de ropa, cuando se tiene una ropa más delicada que otra, la agitación se tiene que realizar de manera más suave de tal forma que no se tenga algún daño en las prendas que se están lavando. Está frecuencia de agitación asignado por el control es un valor máximo al cual se va a agitar la ropa, este valor puede disminuir también con la turbiedad del agua el cual es un parámetro que me indica si la ropa esta soltando suciedad, cuando tengo una turbiedad menor a la que se tuvo cuando se inicio el proceso de lavado, la frecuencia de agitación disminuye de tal manera que se tenga un ahorro de energía. Pero cuando la suciedad sigue en niveles altos, la agitación se encuentra en el valor máximo asignado por el control de tal manera que estamos aplicando mayor fuerza mecánica a la prenda para que esta pueda desprender la suciedad que lleva impregnada.

3. Durante el proceso de agitación, se esta monitoreando la Turbiedad, Conductividad y Temperatura del agua, la conductividad es un parámetro que me indica cuanto detergente disuelto tengo en el agua, entonces, si la conductividad es muy baja quiere decir que tengo que agregar más detergente, para que las condiciones de lavado no se modifiquen, la cantidad de detergente es un factor muy importante en el proceso de lavado, si tengo mucho detergente puedo saturar el agua y el detergente no realizará la misma tarea que si tengo el necesario para lavar la ropa.
   Por otro lado, si la turbiedad es muy alta quiere decir que las prendas han soltado suciedad, en valore muy altos, se tira una cantidad de agua y se vuelve a agregar más agua limpia por las válvulas de admisión, cuando la turbiedad del agua no cambia (esto es

se mantiene constante durante un tiempo determinado por el control) se decide que el proceso de lavado ha terminado y se procede a vaciar la tina, en ese momento el proceso de lavado ha terminado.

El sistema de control solamente esta enfocado al proceso de lavado, no se contemplaron las funciones adicionales que la lavadora realiza, como lo es un prelavado, un enjuague y un centrifugado ya que en el lavado es donde podemos sacar más provecho de las variables y poder ahorrar insumos.

A continuación se muestran dos las condiciones de dos pruebas realizadas:

PRUEBA 1.

Características de la carga:

- Cantidad de ropa: 5 Kgs
- Tipo de Ropa: Sintéticos Blancos

Se seleccionaron los siguientes parámetros en el sistema de control:

- Cantidad de Ropa: Mucha
- Tipo de Ropa: 5 SB
- Grado e Suciedad: Muy Sucia

Una vez iniciado el proceso de lavado, el control asigna las siguientes variables:

- Cantidad de agua: 41.01 litros
- Cantidad de Detergente: 50 gramos
- Temperatura de agua: 32 °C
- Tiempo de apertura de Válvula de Agua Caliente: 28.12 s
- Tiempo de apertura de Válvula de Agua Fría: 28.82 s
- Frecuencia de agitación 50 Osc/nmin

PRUEBA 2.

Características de la carga:

- Cantidad de ropa: 8 Kgs
- Tipo de Ropa: Algodón Colores

Se seleccionaron los siguientes parámetros en el sistema de control:

- Cantidad de Ropa: Llena
- Tipo de Ropa: 2 AC
- Grado e Suciedad: Extra Sucia

Una vez iniciado el proceso de lavado, el control asigna las siguientes variables:

- Cantidad de agua: 57 litros
- Cantidad de Detergente: 90 gramos
- Temperatura de agua: 35 °C
- Tiempo de apertura de Válvula de Agua Caliente: 35.2 s
- Tiempo de apertura de Válvula de Agua Fría: 15.8 s
- Frecuencia de agitación 55 Osc/nmin

En el Apéndice 2 se presentan los datos y gráficas obtenidas de las pruebas realizadas al sistema. Se presentan los datos antes y después de aplicar al sistema el algoritmo de control difuso.

CONCLUSIONES

Tomando como marco de referencia los objetivos del proyecto, podemos obtener algunas conclusiones las cuales nos ayudan para poder evaluar el trabajo realizado. Recapitulando, la finalidad del proyecto se centra en tres factores:

a) Obtener una base de conocimiento mínima que permita establecer las relaciones entre algunas de las variables involucradas en el proceso.

- El proceso de lavado es un sistema complejo para el cual no existe un modelo matemático que lo reproduzca, sin embargo hoy día se cuenta con otros medios para crear modelos que nos permitan conocer más de las relaciones que guardan los factores involucrados en el proceso.

- El resultado de la ejecución, procesamiento y análisis sistemático de las condiciones de operación, nos brindó la oportunidad de interpretar como se relacionan algunas de las variables involucradas, e hizo posible la creación y ajuste de reglas de control para el proceso, tomando en cuenta nuestras variables seleccionadas.

- Analizando las graficas de conducta anteriores y posteriores a la implantación del nuevo sistema de control, podemos decir que se cuenta con el conocimiento básico para poder alterar y mejorar el proceso.

b) Poder optimizar los insumos.

- El poder determinar si el ahorro de insumos en este trabajo aún es limitado debido a que nos encontramos en una etapa de simulación del sistema de control convencional mediante un sistema de control difuso, donde inicialmente, para observar las ventajas reales, es necesario partir de un punto de comparación; esto se logra no alterando de forma significativa las condiciones actuales.

- El control mediante lógica difusa representa una alternativa importante en la creación de sistemas para los cuales se cuenta con un amplio conocimiento adquirido de la experiencia.

c) Crear un sistema de control que permita experimentar de manera fácil con actuales y nuevos procesos de lavado.

- Las gráficas obtenidas antes y después de utilizar las rutinas de control, nos indican que los algoritmos utilizados mantienen la turbiedad, temperatura y conductividad dentro de rangos estables, sin grandes picos o variaciones. Esto revela que el comportamiento del sistema con un control difuso es robusto, es decir que las variaciones en las condiciones de entrada no afectan al control.

- El nivel en que se logre expresar la semántica del fenómeno en el diseño de un modelo difuso, se reflejará en la certidumbre del comportamiento del sistema.

- Los resultados generados por un modelo difuso, deben ser validados y verificados por medio de un comportamiento anteriormente previsto, contra los casos conocidos o contra el juicio razonable de los expertos o, a falta de estos, contra el sentido común.

- El conocimiento de los ingenieros y el análisis del sistema, sin embargo, todavía deben determinar si la salida de un modelo difuso es consistente con las relaciones entre las reglas y las variables, y si el valor esperado de las variables de solución es o no valido dentro del contexto de la lógica interna del modelo.

- Podemos obtener un sistema en hardware como en software para poder desarrollar, simular e implementar sistemas difusos, en este caso lo utilizamos para el control de lavado en una lavadora comercial, además de que el sistema se puede aplicar a otros sistemas que se comportan de manera similar.

- Para obtener una base de conocimiento amplia y general se requiere añadir al modelo difuso diferentes variables de control, y probar y verificar el comportamiento para distintas condiciones de operación. Sin embargo, estamos seguros que nuestro sistema ofrece la posibilidad de llevar a cabo esta tarea de una manera relativamente fácil.

A pesar de que los resultados obtenidos satisfacen nuestras expectativas, hace falta someter el sistema a pruebas mas rigurosas y a la opinión técnica de los expertos en el área, así como la validación basándose en las normas de lavado existentes.

Propuestas de trabajo a futuro

Algunos de los trabajos que proponemos y que pueden ser de gran interés, no solo para el área de la electrónica sino para otras ingenierías como lo es la mecánica, computación, el diseño industrial, etc.

- El desarrollo de un dosificador de detergente, el cual ayude a dosificar en pequeñas cantidades ya sea el detergente líquido o en polvo.
- Estudio de nuevos materiales para la tina, el chasis, el panel.
- El diseño de un panel digital, con display para presentación de información, botones, etc.
- Estudio de los sensores de turbiedad, conductividad y temperatura para poder seleccionar el más adecuado en cuanto a costo, instalación, mantenimiento, etc.
- Estudio del microcontrolador a ser utilizado para las rutinas de lógica difusa, de manera que todo quede incorporado en una sola tarjeta de control, este microcontrolador debe tener gran capacidad de memoria, periféricos incorporados como lo es un convertidor analógico digital, timers de programación, comunicación serial, velocidad de procesamiento, etc.

Referencias.

[Axelon, 1998] **Serial Port Complete**
Jan Axelon

Ed. Lakeview Research
USA 1998

[Balcells,1998]. **Autómatas Programables**
Joseph Balcells, José Luis Romero
Alfa Omega, México 1998

[Chin-Teng, 1996] **Neural Fuzzy Systems**
Chin-Teng Lin, C.S. George Lee
Prentice Hall U.S.A. 1996

[Cox, 1994] **The Fuzzy Systems Handbook**
Earl Cox
AP Professional 1994

[Creus, 1998] **Instrumentación Industrial**
Antonio Creus
Alfa Omega, México 1998.

[Faulkenberry, 1990] **Introducción a los amplificadores operacionales con aplicaciones ACI lineales**
Luces M. Faulkenberry
Limusa, México 1990.

[Joyanes,1998] **Programación orientada a objetos**
Luis Joyanes Aguilar
McGrawHill 1998

[Kaufman,1996] **Active X Programming**
Sanders Kaufman Jr., Jeff Perkins, Dina Fleet
Sams.net Publishing 1996.

[Minnasian, 1998] **Desarrollo de Sistemas Innovadores para Lavadoras de Ropa**
Minassian Rafael
Tesis de Licenciatura UNAM, FI Octubre 1998

[National, 1995] **National Operational Amplifiers Data Book**
National Semiconductor 1995

[National, 1995] **National Data Acquisition Data Book**
National Semiconductor 1995

[National, 1997] **COP8 Microcontroller Data Book**
National Semiconductor 1997

[National, 1998] **COP8 Family user's manual feature**
National Semiconductor 1998

[Ogata, 1993] **Ingeniería de control moderna**
Katsuhiko Ogata
Prentice Hall Hispanoamericana, México 1993.

[Terano,1987] **Fuzzy Systems Theory and Applications**
Toshiro Terano, Kiyoji Asai. Michio Sugeno
Academic Press, Inc. U.S.A. 1987

[Toth, 1999] **Visual C++ 6 Unleashed**
Víctor Toth
Sams Publishing 1999

Paginas de Internet:

http://www.national.com

http://www.whirpool.com

http://www.honeywell.com

http://www.mot-ic.com

APÉNDICE DE LA TESIS
" CONTROL DIFUSO PARA LAVADO DE ROPA "

APENDICES

A1. Consejos y Sugerencias

A1.1 Descomposición semántica de una variable

La adecuada descomposición de una variable en un conjunto completo de términos difusos es un aspecto muy importante en la creación de modelos robustos y resistentes. Solo las regiones fuzzy nombradas pueden ser direccionadas o referidas desde nuestro conjunto de reglas. Por ello, todas las regiones con algún significado semántico importante para nuestra variable deben estar representadas por un conjunto fuzzy.

A1.2 Convenciones en el nombramiento de los conjunto fuzzy

El nombramiento de los fuzzy sets componentes de un modelo tiene importantes implicaciones en la estabilidad, mantenimiento y validación del modelo. En ingeniería de control, las variables comúnmente representan cambios de estado: cambio en la velocidad, cambio en el error, cambio en temperatura, etc. Los fuzzy sets asociados con una variable, llamados el *conjunto de términos*, son generalmente rotulados para indicar los cambios positivos o negativos. La figura A1.1 muestra un ejemplo.

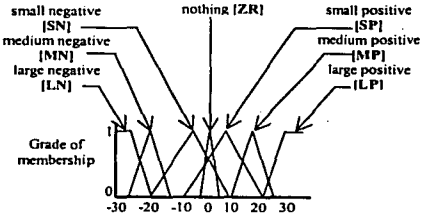


Figura A1.1Definición de los conjuntos fuzzy

A los conjuntos fuzzy se le otorga etiquetas que describen la acción en lugar de la semántica asociada con la acción misma. Por ejemplo el término Large Negative refleja de manera más precisa la magnitud de un valor negativo que el término *Big Loss*.

Cuanto más sea posible, el nombre de un conjunto fuzzy debería indicar lo que significa en lugar de lo que es en realidad.

Así mismo, deberían evitarse las abreviaturas o los nombres secretos tales como *BL, SL, BE*. Debido a que los modelos fuzzy están basados en una aproximación lingüística para representar el modelo, los nombres de los conjuntos fuzzy deberían reflejar tanto como sea posible, el significado

natural del término. Desde el punto de vista de mantenimiento y validación esta es una propiedad importante de los modelos fuzzy. Por ejemplo reglas como:

If profits are a BIGLOSS then the expansion.program is very REDUCED;
if profits are at Break.Even then the expansion.program is slightly ADVANCED;

ofrecen de manera clara el significado de sus acciones. Esto es preferible a las reglas escritas con abreviaturas de los nombres de los conjuntos fuzzy:

If profits are BL then expansion.program is VR;
if profits are BE then expansion.program is SA;

lo cual hace más difícil entender el significado de la regla

A1.3 El significado y el grado de traslape en los conjuntos fuzzy

Para convertir una serie de regiones fuzzy individuales en una superficie continua y suave, cada conjunto fuzzy debe en cierto grado traslaparse con sus conjuntos vecinos. No existe un algoritmo que determine el grado mínimo o máximo de traslape, pero este patrón de interferencia debe reflejar la semántica del control asociado o variable de solución. Este traslape no es un artificio del razonamiento difuso, sino que refleja la naturaleza real de la razón de ser de los conjuntos fuzzy en el dominio de la variable. Un traslape es la consecuencia natural de la imprecisión y la ambigüedad asociada con la segmentación y clasificación de un espacio continuo.

La experiencia determina que el traslape entre el punto medio del borde y las regiones fuzzy vecinas se encuentra entre el 25% y 50% de la base del conjunto fuzzy. La figura A1.2 ilustra el traslape convencional a partir del punto medio para conjuntos fuzzy trapezoidales, triangulares y de forma de campana Figura A1.2:

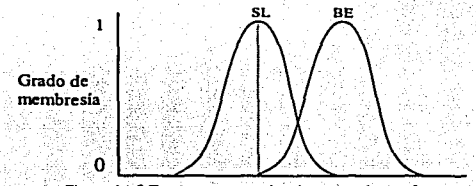


Figura A1.2 Traslape convencional para conjuntos fuzzy

El punto máximo de membresía para un conjunto fuzzy representa el punto de mínimo de membresía para cada uno de los conjuntos vecinos. Esto tiene algo de sentido. Después de que una región fuzzy alcanza su punto máximo de membresía [1.0] su valor comienza a decaer al

incrementares el valor del dominio (esto significa que no es más un perfecto representante del conjunto).

Existen algunos casos en que un traslape excesivo es necesario, usualmente, en problemas de control. En el modelado de sistemas de información, sin embargo, deberíamos examinar con sumo cuidado el significado de un traslape excesivo. Para evitar en alguna medida la ambigüedad provocada por el traslape, se han desarrollado algunos métodos prácticos para establecer el traslape entre conjuntos vecinos.

La siguiente ecuación establece que la suma de todos los puntos involucrados en un traslape de conjuntos debe ser igual o menor que uno [1.0]. Esta es la regla de *suma a uno [o menos]*.

$$\sum_{i=0}^{n} \mu_i[x] \leq 1 \qquad (1)$$

En la mayoría de los casos *n* debería ser igual a 2 y en general el valor máximo es de 3. Esto significa que en una región en particular del dominio de la variable no deberían existir más de dos interpretaciones simultaneas, o más de tres para casos especiales. Usualmente, la tercera región representa una definición del espacio comprendido entre otras dos. La figura A1.3 muestra este caso.

Esta excepción viola la regla de *suma a uno [o menos]*, pero como se ha dicho esto es en ocasiones necesario.

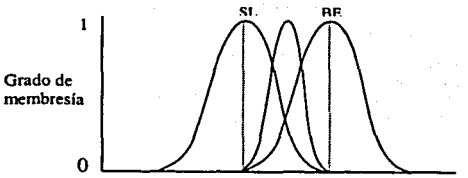


Figura A1.3 Traslape Triple.

En un modelo fuzzy robusto (robusto en el sentido de ingeniería en que el modelo puede tolerar cambios en su estado de operación) se desea que múltiples reglas se disparen cuando un valor del dominio existe en varias regiones fuzzy vecinas. Esto corresponde al grado en el cual el sistema se encuentra en transición de un estado al siguiente. Cuando las reglas se ejecutan debido a un subconjunto de membresía, el modelo no se encuentra en un verdadero estado de transición sino que esta reflejando un fortalecimiento de los valores de verdad debido al traslape de los conjuntos involucrados.

A1.4 Perspectivas en ingeniería de control sobre traslapo y composición

Los sistemas fuzzy en ingeniería de control tienen una orientación y un propósito diferente que en los modelos de información. Un sistema de control usualmente opera en tiempo real con el objetivo de encontrar los valores de desempeño óptimo (o cerca del óptimo) para un dispositivo. El estado actual del dispositivo es sensado y retroalimentado a través de ingeniería fuzzy como un conjunto de diferencia de mediciones. Las reglas fuzzy aseguran que el estado de operación del dispositivo permanece dentro de un rango aceptable. La naturaleza del conjunto de términos asociados con variables de sistemas de control difiere algo de aquellos normalmente utilizados en modelos de información. Para ver estas diferencias, consideremos un sistema fuzzy que controla una válvula de estrangulación de gasolina de un motor. El consumo del inyector de gasolina esta determinado por la lectura de dos sensores: la temperatura actual y la presión. La Figura A1.4 esquematiza la planta de la turbina:

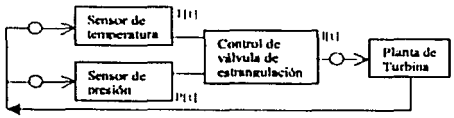


Figura A1.4 Ejemplo de una planta.

El inyector que controla la válvula de estrangulación de gasolina del motor debería permanecer en la posición normal o posición cero mientras que la temperatura y la presión se encuentren en los rangos de operación normal. Cuando la temperatura o la presión se incrementen o disminuyan, la válvula de estrangulación se abre o se cierra para incrementar o disminuir la cantidad de gasolina que recibe la turbina. La Figura A1.5 muestra el conjunto de términos que conforman a la variable de solución THROTTLE.ACTION:

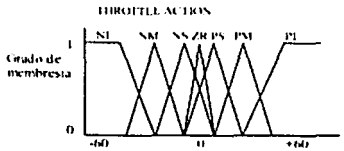


Figura A1.5 Conjuntos de Términos fuzzy

De la figura se pueden hacer varias observaciones acerca de la descomposición del conjunto de términos para un sistema de ingeniería de control. De manera general un control fuzzy o variable de solución tiene las siguientes propiedades:

- En la variable los conjuntos fuzzy están ordenados en regiones de diferentes áreas (tamaños). El área cubierta por los conjuntos fuzzy disminuye en la región del dominio donde se desea que la variable opere. Esto significa que un valor en una de las regiones externas producirá que la acción de una regla traiga al sistema de vuelta a la región de comportamiento aceptable. El efecto aquí es reducir el número de reglas necesarias para controlar la planta debido que una sola regla puede manejar todos los estados del dispositivo en esa región límite.

- El grado de traslape en las regiones de rendimiento deseado frecuentemente excede la regla de *suma a uno (o menos)*. Esto brinda tanto sintonización y control obligado por las reglas, así como también una unión de la región reduciendo las regiones vecinas. Sin embargo, esto viola nuestras reglas sobre el grado de traslape, el incremento de traslape de los conjuntos fuzzy en este punto asegura que múltiples reglas se ejecutarán cuando el estado del problema se mueva hacia la derecha o hacia la izquierda de la región de operación deseada. En algunos dispositivos mecánicos de la clase de balanceo de péndulo, un alto grado de traslape, asegura que, cuando el sistema se encuentra en el estado óptimo, cualquier pequeño cambio en este estado es inmediatamente detectado y manipulado.

- El número de conjuntos fuzzy dentro de la variable es casi siempre un número impar entre 3 y 11, siendo 5 o 7 los de mayor ocurrencia.

- Si una variable, tal como la temperatura, es descompuesta en sus rangos de operación con la región central como el estado de comportamiento deseado, entonces las regiones a la derecha e izquierda se deben etiquetar y debe utilizarse en el conjunto de reglas para indicar que el estado requiere de ciertas acciones. Como consecuencia, esta idea de descomposición se repite. Si una de las regiones a la derecha o a la izquierda se descompone en un número fuzzy (ya sea un conjunto triangular o de forma de campana), entonces las regiones a su derecha e izquierda deben de etiquetarse también. Este proceso invariablemente conduce a un número impar de conjuntos fuzzy.

Los problemas de control son frecuentemente definidos en términos de conjuntos fuzzy triangulares y trapezoidales debido que la representación de estas formas puede ser manejada con un mínimo de almacenaje y la determinación del grado de membresía se realiza en una pequeña cantidad de ciclos.

A1.5 Modelos no deseados

El resultado de un modelo no deseado es un valor esperado de cero con un índice de compatibilidad también de cero. En la Tabla A1.1, no se tomó ninguna acción especial cuando el modelo regresó un resultado no deseado. En los modelos de producción (especialmente donde un valor de cero pudiera ser un valor esperado) algunos procesos excepcionales deben de iniciarse.

Orders	BackOrder	CIX
60	0.00	0.000
80	0.00	0.000
100	212.11	0.290
130	212.11	0.125
150	212.11	0.190
190	212.11	0.315
200	212.11	0.348

Tabla A1.1. Tabla de modelo no deseado.

Cuando la regla falla al ejecutarse la región de salida fuzzy se deja en su condición vacía o nula. La función de defuzzificación detecta esta condición e indica que la salida no puede ser procesada. Así mismo, ninguno de los otros procesos de defuzzificación trabajará con un conjunto fuzzy vacío.

A1.6 Medición del índice de compatibilidad

¿ Como podemos medir la robustez de un modelo fuzzy ? Los conjunto fuzzy de solución son creados por medio de un proceso de agregación. Esto consecuentemente indica el grado de verdad asociado con la regla que generó el resultado de salida, lo cual indica que tan bien las reglas responden a los datos del modelo. La relación entre estos factores es expresada en el **índice de Compatibilidad**.

Los modelo fuzzy que utilizan la idea de un índice de compatibilidad, cuentan con una forma intrínseca de medir su compatibilidad con los datos del modelo. La compatibilidad estadística mide que tan bien un modelo se desempeña sobre un amplio rango de datos (y es una medida real de la compatibilidad del sistema), mientras que la compatibilidad unitaria mide la fuerza de la recomendación de una sola ejecución del modelo.

La idea fundamental de **índice de compatibilidad** es simple: si la altura de una región de salida fuzzy esta cercana a [0] o a [1] entonces el modelo asume las propiedades de un espacio boleano. Una altura muy alta o muy baja indica que el dato se encuentra en los extremos del conjunto fuzzy, provocando que el valor de verdad del predicado sea permanentemente cercano a uno o a cero. La Figura A1.6 muestra que existen regiones de incompatibilidad en los conjuntos fuzzy de solución.

A1.7 Indice de compatibilidad unitario

Existe un índice de compatibilidad para cada variable de solución en el sistema. La compatibilidad unitaria mide la altura de la región de la variable de solución fuzzy. La lectura de este índice es importante debido a que las técnicas de defuzzificación (centroide y máximo compuesto) son generalmente insensibles a la fuerza de la región de salida. La Figura A1.6 muestra el método de defuzzificación del centroide aplicado a la variable de salida *RISK* para un conjunto cuya altura es de [0.80], [0.50] y [0.2]. El centro de gravedad produce un valor esperado de 492 del dominio.
En ciertas circunstancias es posible que para diferentes condiciones, la región de salida no varíe en su forma, sólo en su altura, provocando que el centro de gravedad sea esencialmente el mismo y por consiguiente que el método de defuzzificación arroje valores iguales o muy similares. Los métodos de defuzzificación son esencialmente sensibles al ancho de la región fuzzy pero no así, para la altura.

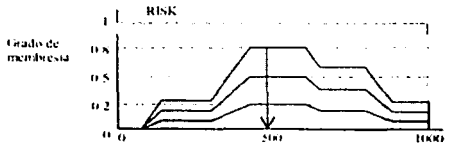


Figura A1.6 Factor de Riesgo

La importancia del índice de compatibilidad en la fuerza de una recomendación de un modelo se puede ver claramente en la salida generada por una regla, de un modelo de inventario:

if orders are high then backorderAmt must be large

La Tabla A1.2 muestra el índice de compatibilidad (CIX) para distintos valores de *Order*.

La única regla produce aproximadamente la misma forma de la región de salida a diferentes alturas, dependiendo del valor de verdad del predicado de la regla: *orders are high*. El método de defuzzificación del centroide encuentra el mismo valor esperado para cada cantidad *backorder* sin importar la altura del conjunto de solución.

Order	BackOrder	CIX
60	0.00	0.000
80	0.00	0.000
92	212.11	0.003
95	212.11	0.012
97	212.11	0.019
100	212.11	0.029
130	212.11	0.125
150	212.11	0.190
190	212.11	0.315
200	212.11	0.348
240	212.11	0.473
280	212.11	0.602
300	212.11	0.664

Tabla A1.2. Tabla de modelo no deseado.

Nota: no se debería asumir que un valor bajo en el índice de compatibilidad sobre una ejecución del modelo en particular es una causa de preocupación. Los modelos fuzzy, como cualquier sistema de decisión, responde con datos que se encuentran a través de todo el espectro del rango de operación. Uno mismo debe decidir dentro del contexto del modelo si un valor bajo afecta la confianza sobre los resultados del modelo.

Una aproximación que ha tenido éxito en modelos como determinación de riesgo, asignación de recursos, inventario, y detección anómala, utiliza el índice de compatibilidad como factor de escala del valor esperado. La Tabla A1.3 muestra los resultados, de aplicar el factor de escala a *BackOrder*, en la última columna:

Order	BackOrder	CIX	CIX*BOAmt
60	0.00	0.000	0.00
80	0.00	0.000	0.00
92	212.11	0.003	0.68
95	212.11	0.012	2.73
97	212.11	0.019	4.10
100	212.11	0.029	6.15
130	212.11	0.125	26.66
150	212.11	0.190	40.34
190	212.11	0.315	67.00
200	212.11	0.348	73.83
240	212.11	0.473	100.50
280	212.11	0.602	127.84
300	212.11	0.664	140.83

Tabla A1.3. Tabla de modelo no deseado con índice de compatibilidad.

Esto, de forma intuitiva, parece tener buenos resultados.

A2. Gráficas y Datos.

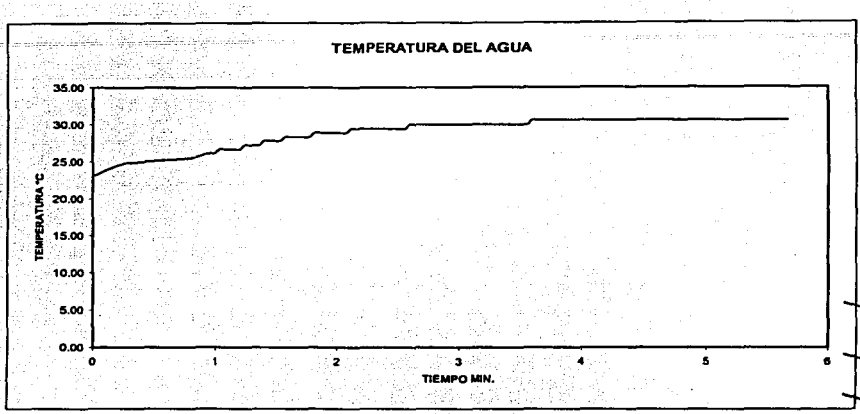
A2.1 Datos de la Prueba 1.

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
0	23.00	50	0.168	79
0.2	24.50	51	0.168	79
0.4	25.00	51.5	0.168	79
0.8	25.50	52	0.168	79
0.9	26.00	52.5	0.168	79
0.955	26.11	53	0.168	79
0.996	26.11	52	0.173	79
1.036	26.67	52	0.175	80
1.078	26.67	53	0.215	80
1.119	26.67	52	0.214	80
1.159	26.67	52	0.246	80
1.2	26.67	52	0.258	80
1.241	27.22	53	0.252	81
1.283	27.22	52	0.286	81
1.324	27.22	57	0.349	81
1.365	27.22	59	0.337	81
1.405	27.78	67	0.429	82
1.447	27.78	67	0.466	82
1.488	27.78	72	0.539	82
1.529	27.78	75	0.458	82
1.57	28.33	79	0.514	83
1.611	28.33	73	0.618	83
1.652	28.33	79	0.563	83
1.693	28.33	78	0.628	83
1.734	28.33	82	0.612	83
1.775	28.33	83	0.731	83
1.816	28.89	81	0.827	84
1.858	28.89	83	0.781	84
1.899	28.89	83	0.693	84
1.94	28.89	81	0.844	84
1.981	28.89	84	0.747	84
2.022	28.89	84	0.541	84
2.064	28.89	85	0.807	84
2.105	29.44	83	0.855	85
2.146	29.44	83	0.763	85
2.187	29.44	82	1.089	85
2.228	29.44	83	0.922	85
2.27	29.44	72	0.855	85
2.311	29.44	71	0.98	85

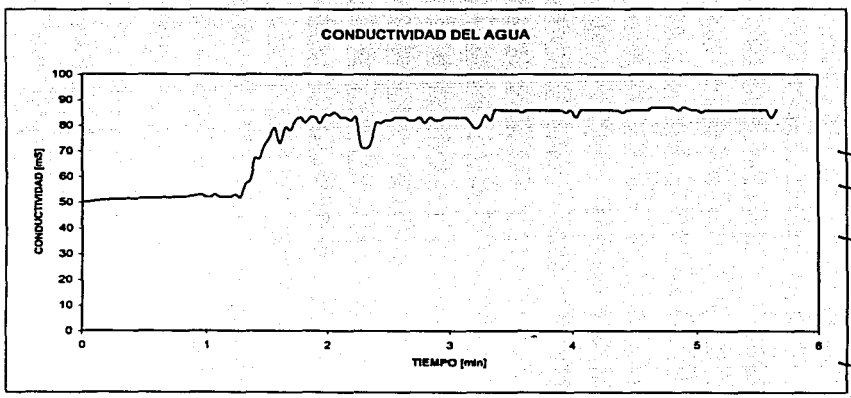
TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB. *25000[NTU]	TEMP. [°F]
2.352	29.44	73	1.007	85
2.393	29.44	81	1.149	85
2.434	29.44	81	0.95	85
2.475	29.44	82	0.817	85
2.517	29.44	82	0.701	85
2.558	29.44	83	0.856	85
2.598	30.00	83	0.89	86
2.639	30.00	83	0.754	86
2.681	30.00	82	0.921	86
2.722	30.00	82	0.967	86
2.763	30.00	83	0.786	86
2.804	30.00	81	0.994	86
2.844	30.00	83	1.17	86
2.886	30.00	82	1.136	86
2.927	30.00	82	1.311	86
2.968	30.00	83	1.251	86
3.009	30.00	83	1.292	86
3.05	30.00	83	0.923	86
3.091	30.00	83	1.15	86
3.132	30.00	83	1.021	86
3.173	30.00	81	1.236	86
3.214	30.00	79	1.228	86
3.255	30.00	80	1.132	86
3.337	30.00	82	1.307	86
3.378	30.00	86	1.531	86
3.419	30.00	86	1.488	86
3.46	30.00	86	1.549	86
3.502	30.00	86	1.322	86
3.543	30.00	86	1.036	86
3.583	30.56	85	1.779	87
3.624	30.56	86	1.541	87
3.666	30.56	86	1.201	87
3.707	30.56	86	1.888	87
3.748	30.56	86	1.341	87
3.789	30.56	86	1.184	87
3.829	30.56	86	1.805	87
3.871	30.56	86	1.399	87
3.912	30.56	86	1.048	87
3.953	30.56	85	1.771	87
3.994	30.56	86	1.453	87
4.035	30.56	83	1.271	87
4.076	30.56	86	1.573	87
4.117	30.56	86	1.683	87
4.158	30.56	86	1.224	87
4.199	30.56	86	1.898	87
4.24	30.56	86	1.798	87
4.281	30.56	86	1.598	87

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB. *25000[NTU]	TEMP. [°F]
4.323	30.56	86	1.622	87
4.364	30.56	86	1.714	87
4.405	30.56	85	1.683	87
4.446	30.56	86	1.823	87
4.488	30.56	86	1.894	87
4.529	30.56	86	1.946	87
4.57	30.56	86	1.952	87
4.611	30.56	86	1.996	87
4.652	30.56	87	2.151	87
4.694	30.56	87	1.308	87
4.735	30.56	87	1.731	87
4.776	30.56	87	1.976	87
4.817	30.56	87	1.555	87
4.858	30.56	86	1.987	87
4.9	30.56	87	1.932	87
4.964	30.56	86	1.71	87
5.005	30.56	86	1.666	87
5.045	30.56	85	2.007	87
5.086	30.56	86	1.925	87
5.127	30.56	86	2.408	87
5.169	30.56	86	1.849	87
5.21	30.56	86	1.833	87
5.251	30.56	86	1.852	87
5.291	30.56	86	1.676	87
5.333	30.56	86	1.858	87
5.374	30.56	86	1.989	87
5.415	30.56	86	1.667	87
5.456	30.56	86	2.212	87
5.497	30.56	86	2.24	87
5.538	30.56	86	1.874	87
5.58	30.56	86	2.395	87
5.621	30.56	83	2.336	87
5.661	30.56	86	2.134	87

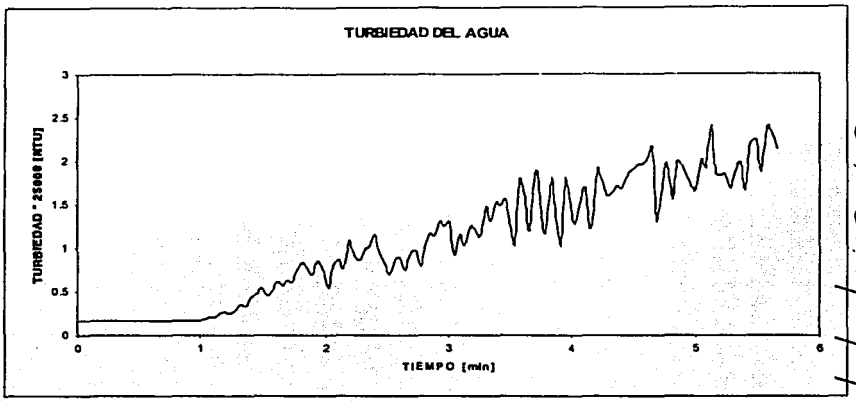
Tabla A2.1 Datos Obtenidos de la Prueba 1 sin el Control.



Grafica A2.1 Temperatura del agua durante el proceso de lavado de la Prueba 1 sin el Control.



Grafica A2.2 Conductividad del agua durante el proceso de lavado de la Prueba 1 sin el Control.



Grafica A2.3 Turbiedad del agua durante el proceso de lavado de la Prueba 1 sin el Control.

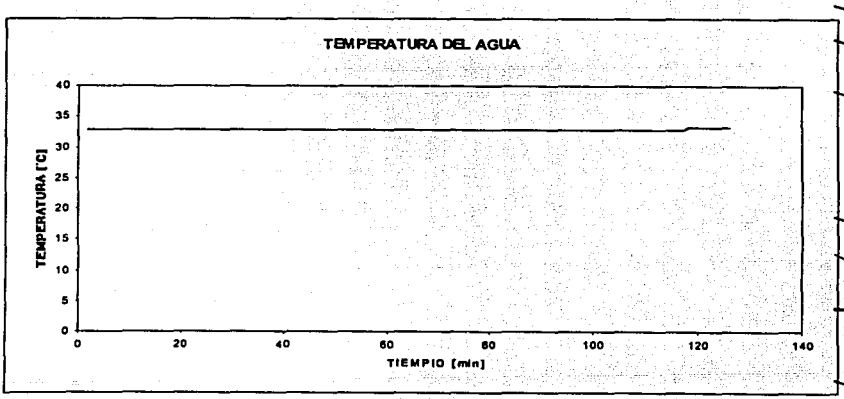
TIEMPO[min]	TEMP. [°C]	COND.[mS]	TURB.*25000 [NTU]	TEMP. [°F]
0	32.78	57	0.08	73
0.041	32.78	57	0.081	73
0.083	32.78	56	0.081	73
0.124	32.78	56	0.088	73
0.165	32.78	56	0.101	73
0.206	32.78	56	0.107	73
0.247	32.78	56	0.108	73
0.288	32.78	56	0.13	73
0.33	32.78	56	0.119	73
0.371	32.78	56	0.121	73
0.412	32.78	56	0.117	73
0.453	32.78	56	0.123	73
0.494	32.78	56	0.126	73
0.536	32.78	56	0.119	73
0.577	32.78	56	0.126	73
0.618	32.78	56	0.12	73
0.659	32.78	55	0.145	73
0.7	32.78	56	0.123	73
0.742	32.78	56	0.133	73
0.783	32.78	56	0.126	73
0.824	32.78	56	0.115	73

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB. *25000[NTU]	TEMP. [°F]
0.865	32.78	56	0.125	73
0.906	32.78	56	0.119	73
0.948	32.78	56	0.125	73
0.989	32.78	56	0.149	73
1.03	32.78	56	0.124	73
1.071	32.78	56	0.142	73
1.112	32.78	54	0.125	73
1.154	32.78	56	0.125	73
1.195	32.78	56	0.133	73
1.236	32.78	56	0.12	73
1.277	32.78	56	0.11	73
1.318	32.78	56	0.134	73
1.36	32.78	56	0.117	73
1.401	32.78	56	0.116	73
1.442	32.78	56	0.134	73
1.483	32.78	56	0.134	73
1.524	32.78	56	0.129	73
1.566	32.78	56	0.119	73
1.607	32.78	56	0.124	73
1.648	32.78	56	0.131	73
1.689	32.78	56	0.118	73
1.73	32.78	56	0.123	73
1.771	32.78	56	0.13	73
1.813	32.78	56	0.145	73
1.854	32.78	56	0.146	73
1.895	32.78	56	0.12	73
1.936	32.78	56	0.125	73
1.977	32.78	56	0.129	73
2.019	32.78	56	0.112	73
2.06	32.78	56	0.121	73
2.101	32.78	56	0.121	73
2.142	32.78	56	0.122	73
2.183	32.78	56	0.129	73
2.225	32.78	56	0.139	73
2.266	32.78	56	0.133	73
2.307	32.78	56	0.12	73
2.348	32.78	56	0.123	73
2.389	32.78	56	0.121	73
2.431	32.78	56	0.137	73
2.472	32.78	56	0.13	73
2.513	32.78	56	0.125	73
2.554	32.78	56	0.128	73
2.595	32.78	56	0.134	73
2.637	32.78	56	0.126	73

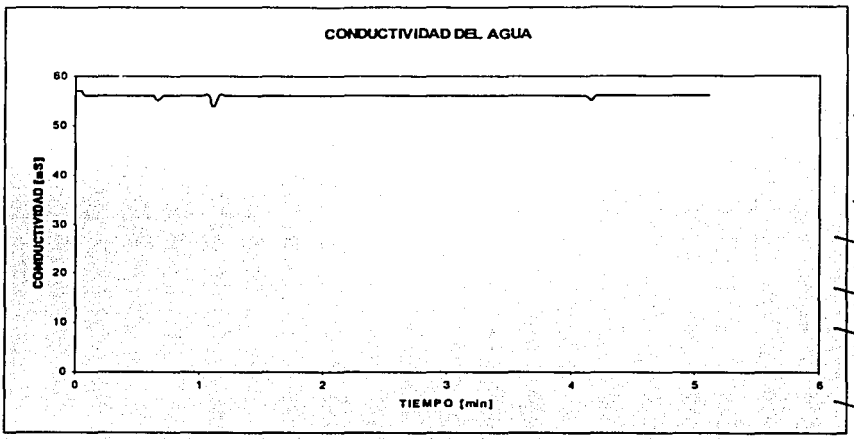
TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB. *25000[NTU]	TEMP. [°F]
2.678	32.78	56	0.122	73
2.719	32.78	56	0.127	73
2.76	32.78	56	0.125	73
2.801	32.78	56	0.13	73
2.843	32.78	56	0.118	73
2.884	32.78	56	0.122	73
2.925	32.78	56	0.132	73
2.966	32.78	56	0.123	73
3.007	32.78	56	0.127	73
3.049	32.78	56	0.143	73
3.09	32.78	56	0.132	73
3.131	32.78	56	0.153	73
3.172	32.78	56	0.13	73
3.213	32.78	56	0.122	73
3.254	32.78	56	0.126	73
3.296	32.78	56	0.139	73
3.337	32.78	56	0.137	73
3.378	32.78	56	0.13	73
3.419	32.78	56	0.145	73
3.46	32.78	56	0.146	73
3.502	32.78	56	0.142	73
3.543	32.78	56	0.132	73
3.584	32.78	56	0.124	73
3.625	32.78	56	0.126	73
3.666	32.78	56	0.148	73
3.708	32.78	56	0.151	73
3.749	32.78	56	0.167	73
3.79	32.78	56	0.145	73
3.831	32.78	56	0.167	73
3.872	32.78	56	0.163	73
3.914	32.78	56	0.179	73
3.955	32.78	56	0.163	73
3.996	32.78	56	0.127	73
4.037	32.78	56	0.133	73
4.078	32.78	56	0.132	73
4.119	32.78	56	0.148	73
4.161	32.78	55	0.159	73
4.202	32.78	56	0.145	73
4.243	32.78	56	0.165	73
4.284	32.78	56	0.155	73
4.326	32.78	56	0.162	73
4.367	32.78	56	0.153	73
4.408	32.78	56	0.131	73
4.449	32.78	56	0.125	73

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB. *25000[NTU]	TEMP. [°F]
4.49	32.78	56	0.184	73
4.531	32.78	56	0.153	73
4.573	32.78	56	0.148	73
4.614	32.78	56	0.154	73
4.655	32.78	56	0.172	73
4.696	32.78	56	0.163	73
4.737	32.78	56	0.16	73
4.778	33.33	56	0.155	74
4.82	33.33	56	0.142	74
4.861	33.33	56	0.124	74
4.902	33.33	56	0.126	74
4.943	33.33	56	0.128	74
4.985	33.33	56	0.118	74
5.026	33.33	56	0.113	74
5.067	33.33	56	0.105	74
5.108	33.33	56	0.1	74

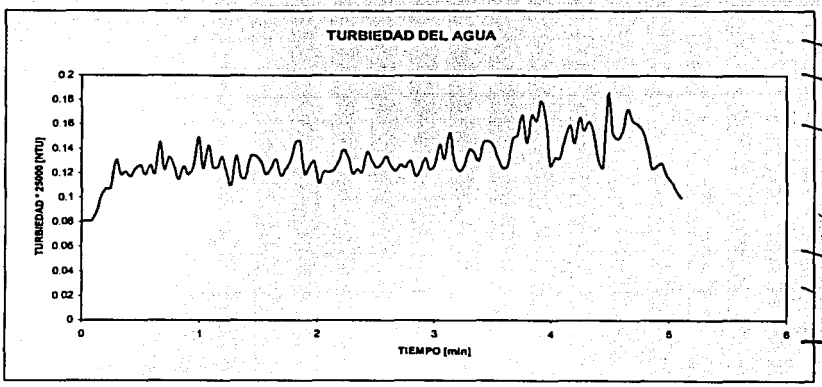
Tabla A2.2 Datos Obtenidos de la Prueba 1 con el Control.



GraficaA2.4 Temperatura del agua durante el proceso de lavado de la Prueba 1 con el Control.



GraficaA2.5 Conductividad del agua durante el proceso de lavado de la Prueba 1con el Control.



Grafica A2.6 Turbiedad del agua durante el proceso de lavado de la Prueba 1con el Control.

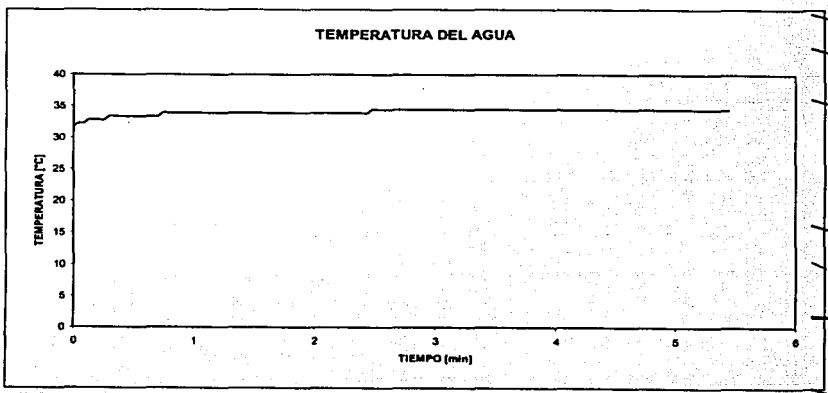
A2.2 Datos de la Prueba 2.

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
0.02	25.00	42	0.14708	77
0.06	25.00	43	0.18296	77
0.09	25.00	44	0.20194	77
0.12	25.00	44	0.23585	77
0.18	25.00	44	0.26991	77
0.24	25.00	45	0.30675	77
0.27	25.00	45	0.29894	77
0.30	25.00	45	0.33153	77
0.33	25.00	45	0.30153	77
0.36	25.00	45	0.30291	77
0.39	25.00	45	0.31304	77
0.45	25.00	45	0.3182	77
0.48	25.00	45	0.31063	77
0.51	25.00	45	0.30121	77
0.54	25.00	45	0.25574	77
0.57	25.00	45	0.27407	77
1.00	25.00	45	0.29648	77
1.03	25.00	45	0.33554	77
1.06	25.00	45	0.29516	77
1.09	25.00	45	0.29589	77
1.12	25.00	45	0.31504	77
1.15	25.00	45	0.28108	77
1.18	25.00	45	0.29465	77
1.21	25.00	45	0.35124	77
1.24	25.00	45	0.32587	77
1.27	25.00	45	0.31867	77
1.30	25.00	45	0.3307	77
1.33	25.00	45	0.36226	77
1.36	25.00	45	0.30085	77
1.39	25.00	45	0.31837	77
1.42	25.00	45	0.35284	77
1.45	25.00	45	0.37562	77
1.48	25.00	45	0.37532	77
1.51	25.00	45	0.36497	77
1.54	25.00	45	0.3494	77
1.57	25.00	45	0.33196	77
2.00	25.00	45	0.32344	77
2.04	25.00	45	0.36254	77
2.07	25.00	45	0.34223	77
2.11	25.00	45	0.33483	77

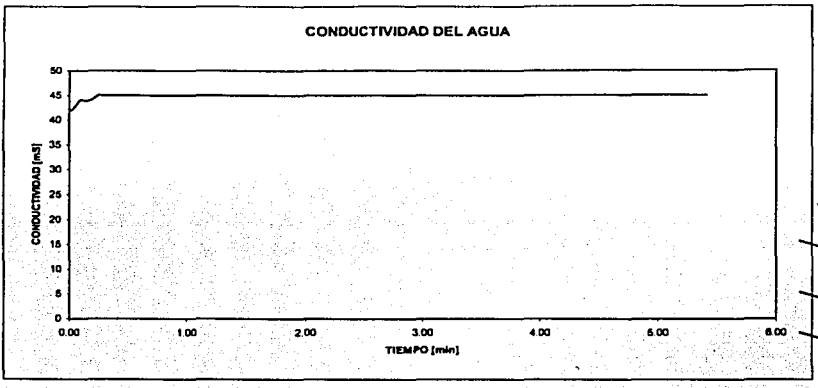
TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
2.14	25.00	45	0.34169	77
2.17	25.00	45	0.3846	77
2.20	25.00	45	0.4035	77
2.23	25.00	45	0.42123	77
2.26	25.00	45	0.41186	77
2.29	25.00	45	0.334	77
2.32	25.00	45	0.29342	77
2.35	25.00	45	0.34885	77
2.38	25.00	45	0.3585	77
2.41	25.00	45	0.35474	77
2.44	25.00	45	0.40334	77
2.47	25.00	45	0.37807	77
2.50	25.00	45	0.36875	77
2.53	25.00	45	0.31176	77
2.56	25.00	45	0.34811	77
2.59	25.00	45	0.39742	77
3.02	25.00	45	0.41213	77
3.08	25.00	45	0.34665	77
3.11	25.00	45	0.32267	77
3.15	25.00	45	0.3456	77
3.18	25.00	45	0.35996	77
3.21	25.00	45	0.36593	77
3.24	25.00	45	0.34405	77
3.27	25.00	45	0.36268	77
3.30	25.00	45	0.35466	77
3.33	25.00	45	0.38208	77
3.29	25.56	45	0.33437	78
3.42	25.56	45	0.33964	78
3.45	25.56	45	0.35688	78
3.48	25.56	45	0.41	78
3.51	25.56	45	0.38679	78
3.54	25.56	45	0.36156	78
3.57	25.56	45	0.36817	78
4.00	25.56	45	0.33845	78
4.03	25.56	45	0.32081	78
4.09	25.56	45	0.32334	78
4.13	25.56	45	0.32224	78
4.16	25.56	45	0.33435	78
4.19	25.56	45	0.35862	78
4.22	25.56	45	0.34104	78
4.25	25.56	45	0.32938	78
4.28	25.56	45	0.33076	78
4.31	25.56	45	0.3393	78
4.34	25.56	45	0.32246	78

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
4.37	25.56	45	0.35687	78
4.40	25.56	45	0.38142	78
4.43	25.56	45	0.37764	78
4.46	25.56	45	0.35012	78
4.49	25.56	45	0.35949	78
4.52	25.56	45	0.3591	78
4.55	25.56	45	0.37218	78
4.58	25.56	45	0.35304	78
5.02	25.56	45	0.36792	78
5.05	25.56	45	0.37329	78
5.08	25.56	45	0.41213	78
5.11	25.56	45	0.34766	78
5.14	25.56	45	0.33224	78
5.17	25.56	45	0.35022	78
5.20	25.56	45	0.3567	78
5.23	25.56	45	0.37536	78
5.26	25.56	45	0.38727	78
5.29	25.56	45	0.37852	78
5.32	25.56	45	0.35351	78
5.35	25.56	45	0.35937	78
5.38	25.56	45	0.36185	78
5.41	25.56	45	0.38329	78

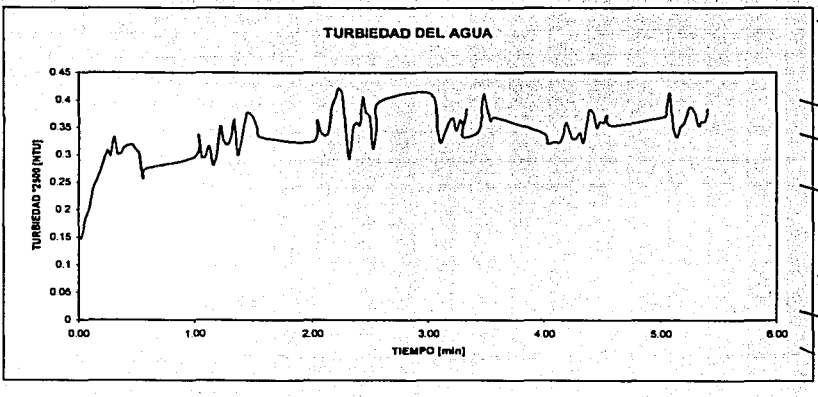
TablaA2.3 Datos Obtenidos de la Prueba 2 sin el Control.



Grafica A2.7 Temperatura del agua durante el proceso de lavado de la Prueba 2 sin el Control.



GraficaA2.8 Conductividad del agua durante el proceso de lavado de la Prueba 2 sin el Control.



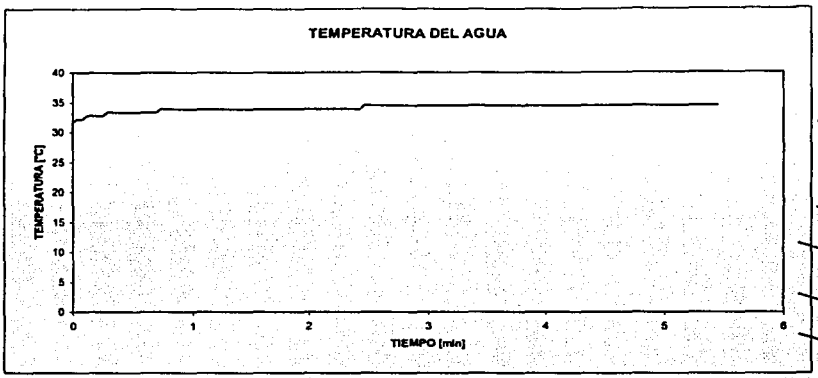
Grafica A2.9 Turbiedad del agua durante el proceso de lavado de la Prueba 2 sin el Control.

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
0	31.66	69	0.27	89
0.041	32.22	67	0.268	90
0.083	32.22	69	0.262	90
0.124	32.77	71	0.311	91
0.165	32.77	76	0.407	91
0.206	32.77	75	0.495	91
0.247	32.77	74	0.665	91
0.288	33.33	74	0.494	92
0.33	33.33	77	0.585	92
0.371	33.33	78	0.599	92
0.412	33.33	77	0.633	92
0.453	33.33	79	0.649	92
0.494	33.33	79	0.737	92
0.536	33.33	78	0.622	92
0.577	33.33	78	0.651	92
0.618	33.33	80	0.899	92
0.659	33.33	79	0.765	92
0.7	33.33	79	0.638	92
0.742	33.88	80	0.75	93
0.783	33.88	80	0.658	93
0.824	33.88	80	0.618	93
0.865	33.88	81	0.838	93
0.906	33.88	80	0.806	93
0.948	33.88	81	0.773	93
0.989	33.88	81	1.151	93
1.03	33.88	81	0.764	93
1.071	33.88	81	1.062	93
1.112	33.88	81	0.982	93
1.154	33.88	81	0.711	93
1.195	33.88	81	0.993	93
1.236	33.88	81	0.859	93
1.277	33.88	80	0.795	93
1.318	33.88	81	0.95	93
1.36	33.88	81	0.833	93
1.401	33.88	80	0.71	93
1.442	33.88	81	0.689	93
1.483	33.88	81	0.678	93
1.524	33.88	81	0.639	93
1.565	33.88	81	0.742	93
1.607	33.88	81	0.764	93
1.648	33.88	81	0.729	93
1.689	33.88	81	0.819	93
1.73	33.88	81	0.853	93

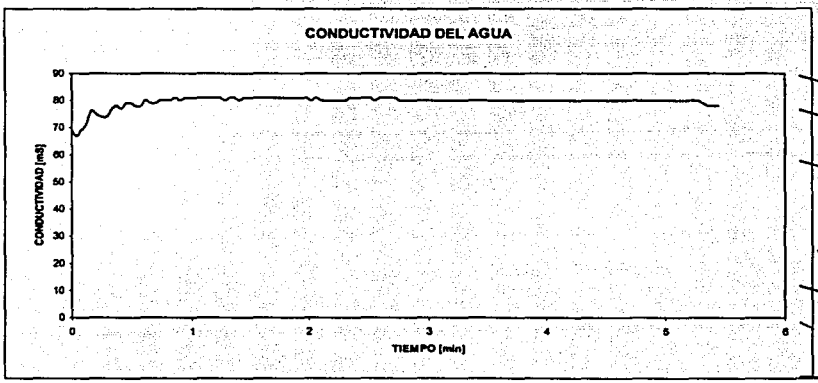
TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
1.771	33.88	81	0.875	93
1.813	33.88	81	0.749	93
1.854	33.88	81	0.693	93
1.895	33.88	81	0.807	93
1.936	33.88	81	0.821	93
1.977	33.88	81	0.731	93
2.019	33.88	80	0.787	93
2.06	33.88	81	0.752	93
2.101	33.88	80	0.697	93
2.142	33.88	80	0.767	93
2.183	33.88	80	0.736	93
2.225	33.88	80	0.802	93
2.266	33.88	80	0.846	93
2.307	33.88	80	0.955	93
2.348	33.88	81	0.853	93
2.389	33.88	81	0.844	93
2.431	33.88	81	0.884	93
2.472	34.44	81	0.825	94
2.513	34.44	81	0.732	94
2.554	34.44	80	0.954	94
2.595	34.44	81	0.893	94
2.637	34.44	81	0.901	94
2.678	34.44	81	0.799	94
2.719	34.44	81	0.812	94
2.76	34.44	80	0.971	94
2.801	34.44	80	0.807	94
2.843	34.44	80	0.856	94
2.884	34.44	80	0.832	94
2.925	34.44	80	0.818	94
2.966	34.44	80	0.962	94
3.007	34.44	80	1.027	94
3.048	34.44	80	0.961	94
3.09	34.44	80	0.73	94
3.131	34.44	80	0.956	94
3.172	34.44	80	1.095	94
3.213	34.44	80	0.78	94
3.254	34.44	80	1.058	94
3.296	34.44	80	0.99	94
3.337	34.44	80	0.81	94
3.378	34.44	80	1.043	94
3.419	34.44	80	1.128	94
3.46	34.44	80	1.002	94
3.502	34.44	80	0.964	94
3.543	34.44	80	1.388	94

TIEMPO [min]	TEMP. [°C]	COND. [mS]	TURB.*25000 [NTU]	TEMP. [°F]
3.584	34.44	80	0.959	94
3.625	34.44	80	1.087	94
3.666	34.44	80	1.239	94
3.708	34.44	80	0.923	94
3.749	34.44	80	1.119	94
3.79	34.44	80	1.043	94
3.831	34.44	80	0.772	94
3.872	34.44	80	0.854	94
3.914	34.44	80	0.904	94
3.955	34.44	80	0.874	94
3.996	34.44	80	0.805	94
4.037	34.44	80	0.913	94
4.078	34.44	80	1.093	94
4.119	34.44	80	0.902	94
4.161	34.44	80	0.974	94
4.202	34.44	80	1.064	94
4.243	34.44	80	0.776	94
4.284	34.44	80	1.026	94
4.325	34.44	80	1.192	94
4.367	34.44	80	1.025	94
4.408	34.44	80	0.997	94
4.449	34.44	80	1.232	94
4.49	34.44	80	0.959	94
4.531	34.44	80	0.877	94
4.573	34.44	80	1.018	94
4.614	34.44	80	0.944	94
4.655	34.44	80	0.902	94
4.696	34.44	80	1.003	94
4.737	34.44	80	1.154	94
4.778	34.44	80	0.905	94
4.82	34.44	80	0.609	94
4.861	34.44	80	0.59	94
4.902	34.44	80	0.594	94
4.943	34.44	80	0.563	94
4.985	34.44	80	0.502	94
5.026	34.44	80	0.495	94
5.067	34.44	80	0.529	94
5.108	34.44	80	0.503	94
5.149	34.44	80	0.5	94
5.191	34.44	80	0.485	94
5.232	34.44	80	0.465	94
5.273	34.44	80	3.8	94

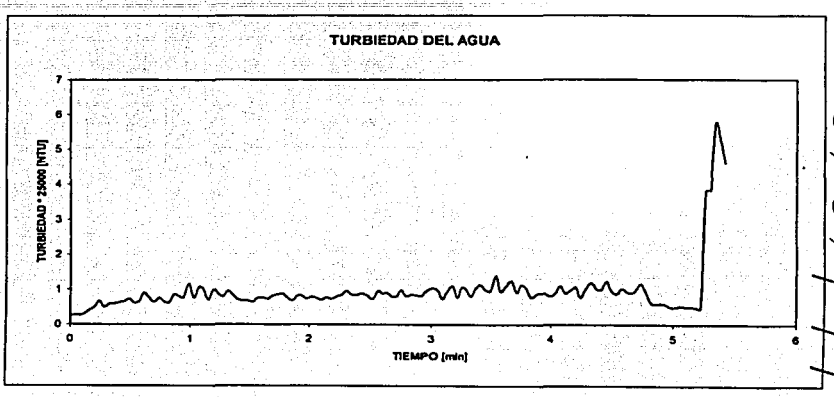
Tabla A2.3 Datos Obtenidos de la Prueba 2 sin el Control.



Grafica A2.10 Temperatura del agua durante el proceso de lavado de la Prueba 2 con el Control.



Grafica A2.11 Conductividad del agua durante el proceso de lavado de la Prueba 2 con el Control.



Grafica A2.12 Turbiedad del agua durante el proceso de lavado de la Prueba 2 con el Control.

A3. Listado del COP8

```
; Programa de comunicación serial para la tarjeta de control
; Se utilizará el microcontrolador COP888
; la transmisión será a 9600 bauds


; Definición de constantes

; Formato de transmisión de la tarjeta de control a la PC
; el programa de comunicación esta corriendo y se llama control0.asm
; a control1 se  le agrago el timer para una interrupción de 1 ms inicialmente
; el programa fue modificado el dia 28 de Agosto por SCC
; En este programa (control2) se modifican los comandos de comunicación entre la PC y el MC
; en el nuevo formato se utilizan 5 bytes para el envio y recepción de los datos


encabezado1_r       = 001   ;Dirección donde se guardan los bytes recibidos
obj_r           = 002
encabezado2_r  = 003
ID_r            = 004
dato_r          = 005
chk_r           = 006


encabezado1_e       = 007   ;Dirección donde se guardan los bytes enviados
obj_e           = 008
encabezado2_e  = 009
ID_e            = 00a
dato_e          = 00b
chk_e           = 00c


AGITACION     = 00d   ;Registro que lleva ON/OFF de la agitacion


flags = 01b ;Registro de Banderas
; 0 = Se recibieron 6 datos
; 1 = Se mandaron 6 datos
; 2 = Bit que lleva el encendido y apagado de los devanados del motor
; 3 = Bit en espera de comando de exito
; 4 = bit de pausa

datos_r = 01c  ;Contador que lleva los datos recibidos
datos_e = 01d  ;contador que lleva los datos enviados
bit = 01e                ; bit de localización de los actuadores

ADC0 = 010     ;registro donde se guarda el valor de la lectura del convertidos analógico digital
ADC1 = 011
ADC2 = 012
ADC3 = 013
ADC4 = 014
ADC5 = 015
```

```
ADC6 = 016
ADC7 = 017

DIG0 = 028      ; registro donde se guarda una entrada digital
DIG1 = 029
DIG2 = 02A
DIG3 = 02B
DIG4 = 02C
DIG5 = 02D
DIG6 = 02E
DIG7 = 02F


ACT0 = 030      ; Registros que llevan el estado de los actuadores
ACT1 = 031
ACT2 = 032
ACT3 = 033
ACT4 = 034
ACT5 = 035
ACT6 = 036
ACT7 = 037


T_ON_MOT = 038      ; registro que lleva el Tiempo de Encendido del Motor
T_OFF_MOT = 039 ; registro que lleva el tiempo de apagado del Motor
T_ON_MOT1= 03a
T_OFF_MOT1 = 03b

temp = 03c      ; registro temporal de uso comun

ACTUADORES = 03d   ; registro que lleva el estado de los actuadores


lobaud  = 0e8       ;valor del contador parte baja para int de 50mS
hibaud  = 003       ; c350 valor del contador parte alta para int de 50mS


;*****************************************************************************************
; Inicialización del microcontrolador
;*****************************************************************************************
        .title serial.asm
        .incld  COP888EG.inc
        .sect  main,rom,abs=0

start:  ld sp,#060      ;inicializo Stack Pointer


;*****************************************************************************************
;Inicializacion del timer
;*************************************************************
;

;se generara una interrupcion cada 50mS
;el valor del contador se encuentra en los registros hibaud y lobaud
```

```
    ld b,#0ea     ;apuntamos a tmrlo
                  rc            ;reset carry
                  ld [b+],#lobaud ;cargamos tmrlo
                  ld [b+],#hibaud ;cargamos tmrhi
                  ld [b+],#lobaud ;cargamos taulo
                  ld [b+],#hibaud ;cargamos tauhi
                  ld [b],#080   ;timer con autoload
                  sbit 4,PSW    ; timer ON


;*****************************************************************
; Inicialización de registros y puertos del microcontrolador
;*****************************************************************

                  ld PORTD,#00          ; apago todos los actuadores
                  ld PORTCC,#003                ; PC0: output
                  rbit 2,PORTGC         ; PG2 : input
                  ld flags,#00          ; inicializo  registro de banderas
                  ld datos_r,#001 ; inicializo registro de datos recibidos
                  ld datos_e,#001       ; inicializo registro de datos enviados
                  ld encabezado1_e,#0aa ; encabezado para enviar datos

;*****************************************************************
; Inicialización de registros del ADC y de los registro digitales
;*****************************************************************

                  ld ADC0,#00
                  ld ADC1,#00
                  ld ADC2,#00
                  ld ADC3,#00
                  ld ADC4,#00
                  ld ADC5,#00
                  ld ADC6,#00
                  ld ADC7,#00

                  ld DIG0,#00
                  ld DIG1,#00
                  ld DIG2,#00
                  ld DIG3,#00
                  ld DIG4,#00
                  ld DIG5,#00
                  ld DIG6,#00
                  ld DIG7,#00

                  ld ACT0,#00
                  ld ACT1,#00
                  ld ACT2,#00
                  ld ACT3,#00
                  ld ACT4,#00
                  ld ACT5,#00
                  ld ACT6,#00
                  ld ACT7,#00


    ld encabezado1_r,#00
    ld obj_r,#00
```

```
        ld encabezado2_r,#00
        ld ID_r,#00
        ld dato_r,#00
        ld chk_r,#00


            ld encabezado1_e,#00        ;Dirección donde se guardan los bytes enviados
        ld obj_e,#00
        ld encabezado2_e,#00
        ld ID_e,#00
        ld dato_e,#00
        ld chk_e,#00

                ld T_OFF_MOT1,#00
                ld T_ON_MOT1,#00
                ld bit,#001

                jsr inituart            ; inicializa UART
                sbit 1,ENUI             ; Receive interrup enable
                rbit 0,ENUI             ; Transmite interrup desable
            rbit 0,flags

;*****************************************************************
; Programa principal
;*****************************************************************

main:   ifbit 0,flags       ; Se han recibido 6 datos?
                jmp main1               ; si
                jsr lee_adc     ; subrutina para leer el ADC
                ifbit 4,flags   ; la pausa esta activada
                jmp main
                jsr actuadores  ; actualizo estado de los actuadores
                jmp main                ; no,espera

main1:  rbit 1,ENUI     ; desable receive interrup
                rbit 0,flags
                ld a,encabezado1_r ;calculo checksum
                add a,obj_r
                add a,encabezado2_r
                add a,ID_r
                add a,dato_r
                add a,#0x01     ; le sumo un uno
                ifeq a,chk_r    ; el checksum está correcto?
                jmp main2
                jmp main3
main2:  ld a,encabezado1_r
                ifeq a,#0cc
                jmp main4
                jmp main3
main4:  jsr deco                ; decodifica información que llego
                jmp main
main3:  jsr error_chk   ; error de chk
                jmp main

;*****************************************************************
; Subrutina para restaurar acumuladores despues de una interrupcion
```

```
;*****************************************************************

                .=00ff
                push A
                ld A,B
                push A
                VIS             ;vector interrup select
REST:  pop A
                x A,B
                pop A
                ld B,#PSW
                sbit GIE,[B]    ;enable general interrups
                reti            ;retorno de interrupción
;*****************************************************************
; Vectores de interrupción
;*****************************************************************

                .=01ec
                .addrw xmitint  ;Transmitter interrup

                .=01ee
                .addrw rcvint   ;Receiver interrup

                .=01f6
                .addrw TIMER1A ;Timer Overflow Interrup

                .=01fa
                .addrw intropa  ;External interrup falling edge
;*****************************************************************
; Subrutina de Incialización de la UART
;*****************************************************************

inituart: ld PORTLC,#065        ;configuro Puerto L
                ld a,#004
                x a,BAUD                ;configuro la comunicación a 9600
                ld a,#0c8
                x a,PSR
                ld ENUR,#00     ;limpio errores
                ld ENU,#00      ;8 bits de datos sin paridad
                ld ENUI,#020    ;Configuro TDx
                ld B,#PSW
                sbit 1,[B]      ; external interrup enable (tapa)
                sbit 4,PSW      ; timer ON

        sbit GIE,[B]    ;enable global interrup
                ret


;*****************************************************************
; Subrutina para decodificar la información llegada
;*****************************************************************

deco:   ld a,obj_r
```

```
                ifeq a,#005
                nop
                ld a,encabezado2_r
                ifeq a,#010
                jmp deco1
                jmp deco2
deco1:  jsr act_sen_read
                ret
deco2:  ifeq a,#011
                jmp deco3
                jmp deco4
deco3:  jsr act_sen_write
                ret
deco4:  ifeq a,#012
                jmp deco5
                jmp deco6
deco5:  jsr exito_PC                    ; La PC me indica que el comando fue un exito
                ret
deco6:  ifeq a,#013
                jmp deco7
                jmp deco8
deco7:  jsr error_PC                    ;ocurrio un error en la PC Byte 5 codigo de erro
                ret
deco8:  ifeq a,#014
                jmp deco9
                jmp deco10
deco9:  jsr notifica_PC         ; Mensaje de notificación de la PC al MC
                ret
deco10:         jsr error_Transmision   ; no llego la informacion correcta
                ret

;*****************************************************************************
; 1 Subrutina para leer el valor de una variable (sensor o actuador)
;*****************************************************************************

act_sen_read:  ld a,#011                        ; envio exito de operación
                        x a,encabezado2_e
                        ld a,ID_r                       ; apunto en la dirección solicitada
                        x a,b
                        ld a,[B]                ; cargo el dato que me solicitaron
                        x a,dato_e
                        jsr envia                       ; mando dato solicitado
                        ret
;*****************************************************************************
; 2 Subrutina para escribir en una variable (sensor o actuador)
;*****************************************************************************

act_sen_write:  ld a,#012                       ; envio accion de escritura
                        x a,encabezado2_e
                        ld a,ID_r                       ; apunto en la dirección solicitada
                        x a,b
                        ld a,dato_r
                        x a,[B]
                        ld a,dato_r
                        x a,dato_e              ; confirmo dato escrito
                        jsr envia                       ; mando dato solicitado
```

```
                ret
;*****************************************************************
; 3 Subrutina de exito de la PC al MC
;*****************************************************************
exito_PC:       ret
;*****************************************************************
; 4 Subrutina de errores enviados por la PC
;*****************************************************************
error_PC:       ld encabezado2_e,#013
                ld a,dato_r
                ifeq a,#000
                jsr envia               ; la PC me pide reenvio de instrucción
                ret
;*****************************************************************
; 5 Notifica MC Mensaje de notificación de la PC al MC
;*****************************************************************
notifica_PC:    ld encabezado2_e,#012           ; envio exito de operación
                jsr envia
                ld a,dato_r
                ifeq a,#000
                jmp noti1
                jmp noti2
noti1:          jsr     nada
noti2:          ifeq a,#001
                jmp noti3
                jmp noti4
noti3:          jsr     mostrar
noti4:          ifeq a,#002
                jmp noti5
                jmp noti6
noti5:          jsr     pausa
noti6:          ifeq a,#003
                jmp noti7
                jmp noti8
noti7:          jsr     stop
noti8:          ifeq a,#004
                jsr break
                jsr error_Transmision
                ret
;*****************************************************************
; Mensajes de notificación
nada:   ret
mostrar:        ret
pausa:  ld a,#000
                x a,PORTD               ; apago actuadores
```

```
                ld a,flags
                xor a,#010              ; cambio estado de pausa
                x a,flags
                ret

stop:   ld a,#000
                x a,PORTD
                jmp start                       ;inicio de programa
                ret


break:  ld a,#000
                x a,PORTD
                rbit 0,ENUI             ; Transmite interrup desable
                ret


;*****************************************************************
;Subrutina de errores enviados por el MC
;*****************************************************************

error_chk:      ld encabezado2_e,#013
                        ld dato_e,#00
                        jsr envia               ; peticion de reenvio de informacion
                        ret

;*****************************************************************
;Subrutina de errores enviados por el MC
;*****************************************************************

error_Transmision:      ld encabezado2_e,#013
                        ld dato_e,#00
                        jsr envia               ; peticion de reenvio de informacion
                        ret

;*****************************************************************
; 6 Subrutina de para enviar comando de exito a la PC
;*****************************************************************

exito:          ret


;*****************************************************************
;Subrutina de errores enviados por el MC
;*****************************************************************

error_MC:               ; falta por definir los posibles errores
                        ret

;*****************************************************************
; Subrutina para enviar los datos
```

```
;*****************************************************************

envia:  ld datos_e,#001
        ld encabezado1_e,#0aa          ; encabezado de comunicación
        ld a,obj_r
        x a,obj_e
        ld a,ID_r
        x a,ID_e                                    ; encabezado de identificación
        ld a,encabezado1_e      ; calculo el CHK a enviar
        add a,obj_e
        add a,encabezado2_e
        add a,ID_e
        add a,dato_e
        add a,#001      ;le sumo un uno
        x a,chk_e
        rbit 1,flags
        sbit 0,ENUI     ; transmit interrup enable
envia2: ifbit 1,flags       ; se enviaron los 6 datos?
        jmp envia1
        jmp envia2
envia1: rbit 1,flags        ;apago bandera de datos enviados
        rbit 0,ENUI     ; transmit interrup desable
      sbit 1,ENUI  ; receive interrup enable
        ret

;*****************************************************************
; Subrutina de interrupción de Recepción de la UART
;*****************************************************************

rcvint:  ld B,#PSW
        rbit GIE,[B]    ;desable interrups
        ifeq RBUF,#0cc
        jmp rcv1
        jmp rcv2
rcv1:   ld datos_r,#001
rcv2:   ld A,datos_r
        x A,B
        ld A,RBUF       ;leo el dato recibido
      x A,[B+]          ;lo guardo
;       jmp REST
        ld A,datos_r
        inc A
        x A,datos_r
        ld A,datos_r
        ifeq A,#007 ;llegaron 6 datos
        jmp rcvint1     ; si
        jmp REST                ; no,Restauro acumuladores

rcvint1: ld datos_r,#001
        sbit 0,flags    ; se recibieron 6 datos
        jmp REST

;*****************************************************************
; Subrutina de interrupción de Transmisión de la UART
;*****************************************************************
```

```
xmitint: ifeq datos_e,#001
                jmp x1
                jmp x2
x1:             ld a,encabezado1_e
                x a,TBUF
                jmp x10
x2:             ifeq datos_e,#002
                jmp x33
                jmp x44
x33:            ld a,obj_e
        ifeq obj_e,#0cc
        ld a,obj_r
                x a,TBUF
                jmp x10
x44:            ifeq datos_e,#003
                jmp x3
                jmp x4
x3:             ld a,encabezado2_e
                x a,TBUF
                jmp x10
x4:             ifeq datos_e,#004
                jmp x5
                jmp x6
x5:             ld a,ID_e
                x a,TBUF
                jmp x10
x6:             ifeq datos_e,#005
                jmp x7
                jmp x8
x7:             ld a,dato_e
                x a,TBUF
                jmp x10
x8:             ifeq datos_e,#006
                jmp x9
                jmp x10
x9:             ld a,chk_e
                x a,TBUF
x10:            ld a,datos_e
                inc a
                x a,datos_e
                ld a,datos_e
                ifeq a,#007             ; se enviaron los 6 datos
                jmp xmit1
                jmp xmit2
xmit1:  ld datos_e,#001
                sbit 1,flags    ;se enviaron 6 datos
xmit2:  jmp REST                ;Restauro acumuladores

;*****************************************************************
; Subrutina de interrupción externa (medición de cantidad de ropa pin PG0)
;*****************************************************************

intropa: rbit 1,PSW     ; external interrup deshable
                nop
                nop
                ld A,DIG1
```

```
                inc A           ; incremento registro que lleva
                x A,DIG1                ; el número de rev. para medir la cantidad de ropa
                rbit 3,PSW      ; Reset External Interrup Pending
                sbit 1,PSW      ; external interrup enable (tapa)
                jmp REST                ; restauro los acumuladores

;*****************************************************************
; Subrutina de interrupción del Timer Overflow
;*****************************************************************

TIMER1A:        rbit  5,PSW     ;RESETEA LA BANDERA T1A PENDING DEL TIMER
                ifbit 2,flags           ; Tiempo de apagado de bobinas ?
                jmp tim3                        ; si
                ifbit 3,flags           ; motor 1 o motor2
                jmp mot1
                jmp mot2


mot1:   sbit 6,PORTD                            ; enciendo bobina1
                rbit 7,PORTD
                jmp tim4

mot2:   sbit 7,PORTD                            ; enciendo bobina2
                rbit 6,PORTD


tim4:   ld A,T_ON_MOT1                          ; no
                inc A
                x A,T_ON_MOT1
                ld A,T_ON_MOT1
                ifeq A,T_ON_MOT
                jmp tim1
                jmp tim2
tim1:   ld A,flags
                xor A,#008
                x  A,flags      ;CAMBIA EL CONTENIDO DE D CON EL CONTENIDO DE A
                sbit 2,flags                    ; transcurrio tiempo de encendido
                ld T_ON_MOT1,#000

tim3:   rbit 6,PORTD
                rbit 7,PORTD
                ld A,T_OFF_MOT1
                inc A
                x A,T_OFF_MOT1
                ld A,T_OFF_MOT1
                ifeq A,T_OFF_MOT
                jmp tim5
                jmp tim2
tim5:   rbit 2,flags                    ; transcurrio tiempo de apagado
                ld T_OFF_MOT1,#000
tim2:   jmp REST                ;SALTA A RESTAURA

;*****************************************************************
; Subrutina para leer el ADC y el interruptor de la tapa
;*****************************************************************
```

```
lee_adc:        ifbit 2,PORTGP          ; lee interruptor de la tapa
                jmp lee1
                jmp lee2
lee1:       ld DIG2,#0ff
                jmp lee3
lee2:       ld DIG2,#000
lee3:       ld PORTCD,#000      ; pongo la Dirección del canal A/D a leer canal 1
                nop
                nop
                sbit 3,PORTCD ;activo el pin START
                nop
                nop
                rbit 3,PORTCD  ; START OFF
                nop
                nop
                sbit 3,PORTCD ;START ON
                nop
                nop
                nop
                rbit 3,PORTCD ;START OFF
                nop
                nop
                nop
                nop
                nop
                nop
                ld a,PORTI      ; leo el valor del ADC
                subc a,#050
                x a,ADC0                ; lo guardo en le registro de nivel
                ret

;*************************************************************
; Subrutina para actualizar el estado de los actuadores
;*************************************************************

actuadores:  ifbit 0,AGITACION              ;RAM para encender el motor
                jmp ac1
                jmp ac2
ac1:            sbit 4,CNTRL            ; timer run
                jmp ac3
ac2:            rbit 4,CNTRL            ;timer stop
                rbit 6,PORTD            ;apago los embobinados del motor
                rbit 7,PORTD

ac3:            ld a,PORTD
                x a,ACTUADORES              ;la memoria ACTUADORES lleva el estado
de estos


                ld a,#030               ;apunto al inicio de los actuadores
                x a,b           ; intercambio
act3:       ld a,[B+]           ; saco el dato del actuador
                ifeq a,#001
                jp act1
                jp act2
```

```
act1:       ld a,PORTD
            or a,bit
            x a,PORTD           ; enciendo actuador
act4:       ld a,bit
            rc
            rlc a               ; posicion del siguiente actuador
            x a,bit
            ld a,bit
            ifeq a,#020         ; ultimo actuador
            jmp act5
            jmp act3
act5:       ld bit,#001
            ld a,PORTD
            x a,ACTUADORES
            ret
            ret

act2:       ld a,#0ff
            sc
            subc a,bit          ; algoritmo para apagar actuador
            and a,PORTD
            x a,PORTD
            jmp act4

            .end start
```

A4. Listado del Programa FUC

```
// BDR.h: interface for the CBDR class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_BDR_H__E4D0FA06_EA8F_11D3_9F48_188A01C10100__INCLUDED_)
#define AFX_BDR_H__E4D0FA06_EA8F_11D3_9F48_188A01C10100__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "FuzzyVar.h"
#include "FSV.hpp"

typedef CTypedPtrMap<CMapStringToOb, CString, CFuzzyVar*> CMapKeyToFuzzyVar;
typedef CTypedPtrArray<CPtrArray, FDB*> CPtrArrToFDB;
typedef CTypedPtrArray<CPtrArray, VDB*> CPtrArrToVDB;
typedef CTypedPtrArray<CPtrArray, FSV*> CPtrArrToFSV;
typedef CArray<float, float>                CArrFloat;


class CMapKeytoVar : public CMapKeyToFuzzyVar
{
public:
        void Copy(CMapKeytoVar &source);
        void Copy(CStringArray &source);
        CMapKeytoVar();
        virtual ~CMapKeytoVar();
};

class CMapVarToKey : public CMapStringToString
{
public:
        void Copy(CMapVarToKey &source);
        CMapVarToKey();
        virtual ~CMapVarToKey();
};

class CBDR : public CBase
{
public:
        void DeleteExtern();
        void Serialize(CArchive &ar);
        void InitialUpdate(CMapKeyToFuzzyVar& MapVAR);
        BOOL CreateExtern(PDB *pPDB, CMapKeyToFuzzyVar& MapVAR);
        void Copy(CBDR &source, BOOL flag);
        virtual void OnDraw(CDC& dc);
        void RunBDR(PDB* p_XPolicy, CMapKeyToFuzzyVar& MapVAR);
        UINT DetectTypeUpdate(CBDR &BDRcmp);
        CBDR(CBDR &source);
        CBDR::CBDR();
        virtual ~CBDR();

        UINT            m_uRunIndex;
        BOOL            m_bRun;
        BOOL            m_bAND;
        int                     m_nRules;
        CString         m_sComent;
        CMapVarToKey m_MapFAM;
        CArrFloat       m_afWeight;
        CPtrArray       m_paFAM;
        CMapKeytoVar m_MapVarInp;
        CMapKeytoVar m_MapVarOut;
        CPtrArrToFSV m_aFSV;
protected:
        DECLARE_SERIAL(CBDR)
};
```

```
#endif // !defined(AFX_BDR_H__E4D0FA06_EA8F_11D3_9F48_188A01C10100__INCLUDED_)

// BDR.cpp: implementation of the CBDR class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "VitroFuzz.h"
#include "Globales.h"
#include "XSYSctl.h"
#include "XFZYctl.h"
#include "BDR.h"
#include "MDB.hpp"
#include "fuzzy.hpp"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

extern CXSYSctl XSYSctl;
extern CXFZYctl XFZYctl;
extern int status;

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CMapKeytoVar::CMapKeytoVar()
{

}

CMapKeytoVar::~CMapKeytoVar()
{

}

void CMapKeytoVar::Copy(CMapKeytoVar &source)
{
        RemoveAll();
        POSITION pos = source.GetStartPosition();
        CString Key;
        CFuzzyVar* pVar;

        while (pos)
        {
                source.GetNextAssoc(pos, Key, pVar);
                SetAt(Key, pVar);
        }
}

void CMapKeytoVar::Copy(CStringArray &source)
{
        CFuzzyVar* pVar = NULL;
        RemoveAll();
        int count = source.GetSize();

        for (int j=0; j<count; j++)
                SetAt(source.GetAt(j), pVar);
}

CMapVarToKey::CMapVarToKey()
{

}

CMapVarToKey::~CMapVarToKey()
{

}

void CMapVarToKey::Copy(CMapVarToKey &source)
```

```
{
        RemoveAll();
        POSITION pos = source.GetStartPosition();
        CString Var, Key;


        while (pos)
        {
                source.GetNextAssoc(pos, Var, Key);
                SetAt(Var, Key);
        }
}

IMPLEMENT_SERIAL(CBDR, CBase, 0)
CBDR::CBDR(CBDR &source)
{
        Copy(source, TRUE);
}

CBDR::CBDR()
{
        m_sComent       = "";
        m_nRules        = 0;
        m_MapFAM.RemoveAll();
        m_paFAM.RemoveAll();
        m_MapVarInp.RemoveAll();
        m_MapVarOut.RemoveAll();
        m_afWeight.RemoveAll();
        m_bRun  = TRUE;
        m_bAND = FALSE;
        m_uRunIndex = 0;
        m_sizeRect.cx =  280;
        m_sizeRect.cy = -200;
        m_etype             = t_BDR;
}

void CBDR::Copy(CBDR &source, BOOL flag)
{
        m_sName         = source.m_sName;
        m_sComent       = source.m_sComent;
        m_MapVarInp.Copy(source.m_MapVarInp);
        m_MapVarOut.Copy(source.m_MapVarOut);

        if (!flag) return;

        m_nRules = source.m_nRules;
        m_MapFAM.Copy(source.m_MapFAM);
        m_afWeight.Copy(source.m_afWeight);
        m_paFAM.RemoveAll();
        m_bRun  = source.m_bRun;
        m_bAND = source.m_bAND;
        m_uRunIndex = source.m_uRunIndex;
        m_pointTopLeft= source.m_pointTopLeft;
        m_sizeRect.cx = source.m_sizeRect.cx;
        m_sizeRect.cy = source.m_sizeRect.cy;
        m_pBmp          = source.m_pBmp;
        m_color         = source.m_color;
        m_etype         = t_BDR;
        m_bGetFocus     = FALSE;

        return;
}

CBDR::~CBDR()
{

}

void CBDR::DeleteExtern()
{
        CPtrArrToFDB* p_aFDB;
        int     i, cont;

        cont = m_paFAM.GetSize();
        for (i=0; i< cont; i++)
```

```
        {
                p_aFDB = (CPtrArrToFDB*)m_paFAM.GetAt(i);
                p_aFDB->RemoveAll();
                delete p_aFDB;
        }
        m_paFAM.RemoveAll();
        m_aFSV.RemoveAll();
}

void CBDR::RunBDR(FDB* p_XPolicy, CMapKeyToFuzzyVar& MapVAR)
{
        CPtrArrToFDB    a_pAnt;
        CPtrArrToFDB*   p_Rule;
        CFuzzyVar*          pVar;
        FDB* p_FDB;
        HDB* p_HDB;
        FDB* p_wkFDB=NULL;
        BOOL    update;
        POSITION pos;
        CString  varname;
        double inpval;
        double outval;
        float membership[20];
        float mshiptotal[20];
        float grade[20];
        int    ninpvar, noutvar, idx;
        register int i, j, k;

        ninpvar = m_MapVarInp.GetCount();
        noutvar = m_MapVarOut.GetCount();

        a_pAnt.SetSize(ninpvar, 0);


        for (i=0; i < m_nRules; i++)
        {
                update = TRUE;
                pos = m_MapVarInp.GetStartPosition();

                for (j=0; j < ninpvar; j++)
                {
                        p_Rule = (CPtrArrToFDB*)m_paFAM.GetAt(j);

                        p_FDB = p_Rule->GetAt(i);
                        m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                        if (a_pAnt.GetAt(j) != p_FDB)
                                a_pAnt.SetAt(j, p_FDB), update = TRUE;
                        else
                                continue;

                        if (p_FDB == NULL) continue;

                        if (*p_FDB->FDBhdgapp != '\0')
                        {
                                p_HDB = XSYSctl.MdlFindHDB(p_FDB->FDBhdgapp, p_XPolicy, &status);

                                p_wkFDB = FzyCreateSet("NULL", EMPTYSET,
                                                p_FDB->FDBdomain, p_FDB->FDBparms, 0, &status);
                                FzyInitFDB(p_wkFDB);
                                FzyApplyHedge(p_FDB, p_HDB, p_wkFDB, &status);
                                p_FDB = p_wkFDB;
                        }

                        inpval = pVar->m_dActual;
                        membership[j] = FzyGetMembership(p_FDB, inpval,     &idx, &status);

                        if (p_wkFDB != NULL)
                        {
                                delete p_wkFDB;
                                p_wkFDB = NULL;
                        }
                }

                if (update)
                        for (j=0; j < noutvar; j++)
```

```
                {
                        mshiptotal[j] = 0;
                        for (k=0; k < ninpvar; k++)
                        {
                                if ((a_pAnt.GetAt(k)) == NULL) continue;
                                if (membership[k] == 0 &&
                                        pVar->m_nMethod.AND != MEANAND)
                                {
                                        mshiptotal[j] = 0;
                                        break;
                                }

                                if (mshiptotal[j])
                                        mshiptotal[j] = FzyCompAND(m_aFSV[j]->FzySVclassAnd,
                                                                        0, mshiptotal[j], membership[k],
&status);
                                else
                                        mshiptotal[j] = membership[k];
                        }
                }

        for (j=0; j < noutvar; j++)
                if (mshiptotal[j] > 0)
                {
                        p_Rule = (CPtrArrToFDB*)m_paFAM.GetAt(ninpvar + j);
                        p_FDB = p_Rule->GetAt(i);
                        if (p_FDB == NULL) continue;
                        mshiptotal[j] *= m_afWeight.GetAt(i);
                        FzyCondProposition(p_FDB, m_aFSV[j],
                                m_aFSV[j]->FzySVcorrMethod,  mshiptotal[j], &status);
                }
nextrule: ;
        }

        pos = m_MapVarOut.GetStartPosition();
        for (i=0; i < noutvar; i++)
        {
                outval = FzyDefuzzify(m_aFSV[i]->FzySVfdbptr,
                                m_aFSV[i]->FzySVdefuzzMethod, &grade[i], &status);
                m_MapVarOut.GetNextAssoc(pos, varname, pVar);
                pVar->m_dActual = outval;
        }
}

void CBDR::OnDraw(CDC &dc)
{
        CSize   sizebmp(44, 19);
        CSize   lencad;
        CDC             auxDC;
        CPoint  point;
        CPen    penblue;
        CPen*   oldpen;

        auxDC.CreateCompatibleDC(&dc);
        dc.DPtoLP(&sizebmp);
        auxDC.SelectObject(m_pBmp);
        lencad = dc.GetTextExtent(m_sName);
        point = m_pointTopLeft;
        if (m_bGetFocus)
        {
                penblue.CreatePen(PS_SOLID, 1, RGB(0, 0, 255));
                oldpen = dc.SelectObject(&penblue);
        }
        dc.Rectangle(CRect(point, m_sizeRect));
        if (m_bGetFocus) dc.SelectObject(oldpen);
        dc.TextOut((point.x + 260) - lencad.cx,
                                point.y - 50, m_sName);
        dc.StretchBlt(point.x + 5, point.y - 50,
                        sizebmp.cx, -sizebmp.cy, &auxDC, 0, 0,
                        44, 19, SRCCOPY);
        return;
}

UINT CBDR::DetectTypeUpdate(CBDR &BDRcmp)
{
```

```
        CString  var1, var2;
        POSITION pos1, pos2;
        CFuzzyVar* pVar;
        int             cont;
        UINT     updates=0;

        if (m_sComent != BDRcmp.m_sComent)
                m_sComent = BDRcmp.m_sComent;

        if (m_uRunIndex != BDRcmp.m_uRunIndex)
                m_uRunIndex = BDRcmp.m_uRunIndex;

        if (m_bRun != BDRcmp.m_bRun)
                m_bRun = BDRcmp.m_bRun;

        if (m_sName != BDRcmp.m_sName)
                updates |= UPDATENAME;

        if ((cont = m_MapVarInp.GetCount()) != BDRcmp.m_MapVarInp.GetCount())
        {
                updates |= UPDATEINPS;
                return updates;
        }
        if ((cont = m_MapVarOut.GetCount()) != BDRcmp.m_MapVarOut.GetCount())
        {
                updates |= UPDATEOUTS;
                return updates;
        }

        pos1 = m_MapVarInp.GetStartPosition();
        pos2 = BDRcmp.m_MapVarInp.GetStartPosition();

        while (pos1)
        {
                m_MapVarInp.GetNextAssoc(pos1, var1, pVar);
                BDRcmp.m_MapVarInp.GetNextAssoc(pos2, var2, pVar);
                if (var1 != var2)
                        updates |= UPDATEINPS;
        }

        pos1 = m_MapVarOut.GetStartPosition();
        pos2 = BDRcmp.m_MapVarOut.GetStartPosition();

        while (pos1)
        {
                m_MapVarOut.GetNextAssoc(pos1, var1, pVar);
                BDRcmp.m_MapVarOut.GetNextAssoc(pos2, var2, pVar);
                if (var1 != var2)
                updates |= UPDATEOUTS;
        }

        return updates;
}

BOOL CBDR::CreateExtern(PDB *pPDB, CMapKeyToFuzzyVar& MapVAR)
{
        FDB                     *p_FDB = NULL;
        FSV                     *p_FSV = NULL;
        VDB                     *p_VDB = NULL;
        CPtrArrToFDB *p_aFDB= NULL;
        CFuzzyVar*      pVar;
        POSITION        pos;
        CString         varname,
                                termname,
                                vector;
        char            charname[35];
        int                     nSets,
                                index,
                                nMaxSets,
                                inpcnt,
                                outcnt;
        register int i, j;

        m_paFAM.RemoveAll();
        m_aFSV.RemoveAll();
```

```
        m_bAND = FALSE;

        inpcnt = m_MapVarInp.GetCount();
        outcnt = m_MapVarOut.GetCount();


        for (i=0; i < inpcnt+outcnt; i++)
        {
                if (i < inpcnt)
                {
                        if (i == 0)
                                pos = m_MapVarInp.GetStartPosition();
                        m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                }
                else
                {
                        if (i == inpcnt)
                                pos = m_MapVarOut.GetStartPosition();
                        m_MapVarOut.GetNextAssoc(pos, varname, pVar);
                        if (pVar->m_nMethod.AND == MEANAND)
                                m_bAND = TRUE;
                        strcpy(charname, varname);
                        p_FSV = XPZYctl.FzyFindFZYctl(charname, &p_VDB);
                        m_aFSV.Add(p_FSV);
                }
                ASSERT_KINDOF(CFuzzyVar, pVar);
                if (pVar == NULL)
                {
                        if (i < inpcnt)
                                m_MapVarInp.RemoveKey(varname);
                        else
                                m_MapVarOut.RemoveKey(varname);
                        goto nextvar;
                }

                m_MapFAM.Lookup(varname, vector);
                nSets = atoi(GetNextTerm(vector));
                nMaxSets = pVar->m_nSets;
                if ((p_aFDB = new CPtrArrToFDB) == NULL)
                        return FALSE;
                p_aFDB->SetSize(m_nRules, 1);
                for (j=0; j < m_nRules; j++)
                {
                        index = atoi(GetNextTerm(vector)) - 1;
                        p_FDB = NULL;
                        if (index >= 0)
                        {
                                if (index > nMaxSets) index = nMaxSets;
                                termname = pVar->m_saNameSets[index];
                                termname = varname + '_' + termname;
                                strcpy(charname, termname);
                                p_FDB = XSYSctl.MdlFindFDB(charname, pPDB, &status);
                        }
                        p_aFDB->SetAt(j, p_FDB);
                }
                m_paFAM.Add(p_aFDB);

nextvar: ;
        }

        return TRUE;
}


void CBDR::InitialUpdate(CMapKeyToFuzzyVar& MapVAR)
{
        int cntfuzzysets = 1,
                maxfdbout       = 0,
                cntfdb,
                cont,
                indice,
            i=0, j;
        CUIntArray      setidx,
                        change,
                        skip;
```

```
        CString         varname;
        CString         vector;
        CString         newterm;
        CFuzzyVar*      pVar;
        POSITION        pos;

        pos = m_MapVarInp.GetStartPosition();
        while (pos)
        {
                m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                cntfuzzysets *= pVar->m_nSets;
        }

        pos = m_MapVarOut.GetStartPosition();
        while (pos)
        {
                m_MapVarOut.GetNextAssoc(pos, varname, pVar);
                maxfdbout = max(maxfdbout, pVar->m_nSets);
        }
        m_nRules = cntfuzzysets;
        m_afWeight.RemoveAll();
        for (j=0; j < m_nRules; j++)
                m_afWeight.Add(1.0);
        m_MapFAM.RemoveAll();
        cntfdb = cntfuzzysets;
        pos = m_MapVarInp.GetStartPosition();
        while (pos)
        {
                m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                cntfdb = cntfdb / pVar->m_nSets;
                skip.Add(cntfdb);
                setidx.Add(1);
                change.Add(0);
        }
        pos = m_MapVarInp.GetStartPosition();
        while (pos)
        {
                m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                vector.Format("%u,", pVar->m_nSets);
                for (j=0; j < m_nRules; j++)
                {
                        newterm.Format("%u,", setidx.GetAt(i));
                        vector += newterm;
                        cont = change.GetAt(i);
                        cont++;
                        if (cont == skip.GetAt(i))
                        {
                                indice = setidx.GetAt(i);
                                indice = (pVar->m_nSets < ++indice) ? 1 : indice;
                                setidx.SetAt(i, indice);
                                cont = 0;
                        }
                        change.SetAt(i, cont);
                }
                i++;
                m_MapFAM.SetAt(varname, vector);
        }
        pos = m_MapVarOut.GetStartPosition();
        while (pos)
        {
                m_MapVarOut.GetNextAssoc(pos, varname, pVar);
                vector.Format("%u,", pVar->m_nSets);
                for (j=0; j < m_nRules; j++)
                        vector += "1,";
                m_MapFAM.SetAt(varname, vector);
        }
        return;
}

void CBDR::Serialize(CArchive &ar)
{
        CBase::Serialize(ar);

        if (ar.IsStoring())
        {
```

```
                ar << m_nRules;
                ar << m_bRun;
                ar << m_sComent;
                ar << m_uRunIndex;
        }
        else
        {
                ar >> m_nRules;
                ar >> m_bRun;
                ar >> m_sComent;
                ar >> m_uRunIndex;
        }
        m_MapVarInp.Serialize(ar);
        m_MapVarOut.Serialize(ar);
        m_afWeight.Serialize(ar);
        m_MapFAM.Serialize(ar);
}
// DMemory.h: interface for the CDMemory class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_DMEMORY_H__39880641_A7FC_11D4_9F49_907804C10100__INCLUDED_)
#define AFX_DMEMORY_H__39880641_A7FC_11D4_9F49_907804C10100__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "BaseCom.h"

class CDMemory : public CBaseCom
{
public:
        virtual int Notify(stComPort *pstPort, CString &info);
        virtual int Write(stComPort* pstPort, int newval);
        virtual int Read(stComPort *pstPort, CString &info);
        void Serialize(CArchive &ar);
        virtual void OnDraw(CDC &dc);
        void Copy(CDMemory &source);
        CDMemory();
        virtual ~CDMemory();

        UINT    m_uMode;
        int     m_nEvent;
        double m_dValActual;
        double m_dValDefault;
        double m_dValFired;

protected:
        BYTE    TransfDataOutput();
        double AssignDataInput(int newval);

        DECLARE_SERIAL(CDMemory)
};

#endif // !defined(AFX_DMEMORY_H__39880641_A7FC_11D4_9F49_907804C10100__INCLUDED_)


// DMemory.cpp: implementation of the CDMemory class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include <math.h>
#include "vitrofuzz.h"
#include "DMemory.h"
#include "mtypes.hpp"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
```

```
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

IMPLEMENT_SERIAL(CDMemory, CBaseCom, 0)

CDMemory::CDMemory()
{
        m_bCodific          = FALSE;

        m_uMode             = NORMALTYPEMEM;
        m_nEvent            = NONE_ACTION;
        m_dValActual   = 0;
        m_dValDefault  = 0;
        m_dValFired    = 0;
        m_sizeRect.cx  = 260;
        m_sizeRect.cy  = -100;
        m_etype             = t_Memory;
}

CDMemory::~CDMemory()
{

}

void CDMemory::Copy(CDMemory &source)
{
        m_sComent               = "";
        m_byID                  = source.m_byID;
        m_uPortComm             = source.m_uPortComm;
        m_bCodific              = source.m_bCodific;
        m_uBytesRecive   = source.m_uBytesRecive;
        m_uBytesSend    = source.m_uBytesSend;
        m_uMode                 = source.m_uMode;
        m_nEvent                = source.m_nEvent;
        m_dValActual    = source.m_dValActual;
        m_dValDefault   = source.m_dValDefault;
        m_dValFired             = source.m_dValFired;
}

void CDMemory::OnDraw(CDC &dc)
{
        CSize   sizebmp(33, 36);
        CSize   lencad;
        CDC             auxDC;
        CPoint  point;
        CPen    penblue;
        CPen*   oldpen;
        CString valor;

        auxDC.CreateCompatibleDC(&dc);
        dc.DPtoLP(&sizebmp);
        auxDC.SelectObject(m_pBmp);
        lencad = dc.GetTextExtent(m_sName);
        m_sizeRect.cx = 100 + ((lencad.cx <= 250) ? lencad.cx : 250);
        point = m_pointTopLeft;
        if (m_bGetFocus)
        {
                penblue.CreatePen(PS_SOLID, 1, RGB(0, 0, 255));
                oldpen = dc.SelectObject(&penblue);
        }
        dc.Rectangle(CRect(point, m_sizeRect));
        if (m_bGetFocus) dc.SelectObject(oldpen);
        dc.TextOut(point.x + 95, point.y - 3, m_sName);
        if (m_bCodific)
        {
                double  rango = m_dDominio[1] - m_dDominio[0];
                CString formato;
                if (fabs(rango) <= 1.0)  formato = "%.4f";
                if (fabs(rango) > 1.0 && fabs(rango <= 100)) formato = "%.2f";
                if (fabs(rango) > 100)   formato = "%.0f";

                valor.Format(formato, m_dValActual);
        }
        else
                valor.Format("%d",  (int)m_dValActual);
```

```
        dc.TextOut(point.x + 95, point.y - 37, valor);
        dc.StretchBlt(point.x + 5, point.y - 3,
                      sizebmp.cx, -sizebmp.cy, &auxDC, 0, 0,
                      33, 36, SRCCOPY);
        return;
}

void CDMemory::Serialize(CArchive &ar)
{
        CBaseCom::Serialize(ar);

        if (ar.IsStoring())
        {
                ar << m_uMode;
                ar << m_nEvent;
                ar << m_dValActual;
                ar << m_dValDefault;
                ar << m_dValFired;
        }
        else
        {
                ar >> m_uMode;
                ar >> m_nEvent;
                ar >> m_dValActual;
                ar >> m_dValDefault;
                ar >> m_dValFired;
        }
}

int CDMemory::Read(stComPort *pstPort, CString &info)
{
        UINT action = NONE_ACTION;

        if (info == "message") action = WM_SENDNOTIFY;
        info.Format("%d,%d,%d,%d,%d", m_uBytesRecive, m_etype, pstPort->HeadRead, m_byID, DONTCARE);
        return action;
}

int CDMemory::Write(stComPort *pstPort, int newval)
{
        int action = NONE_ACTION;

        if (newval > -1 && newval < 256)
        {
                m_dValActual = (m_bCodific) ? AssignDataInput(newval) : (double)newval;

                if (m_dValActual == m_dValFired)
                        action = WM_FIRE_EVENT;
        }
        return action;
}

int CDMemory::Notify(stComPort *pstPort, CString &info)
{
        BYTE value;
        value = (m_bCodific) ? TransfDataOutput() : (BYTE)m_dValActual;
        info.Format("%d,%d,%d,%d", m_etype, pstPort->HeadNotify, m_byID, value);
        return NONE_ACTION;
}

double CDMemory::AssignDataInput(int newval)
{
        double dMin, dMax, dSlope;

        dMin   = m_dDominio[0];
        dMax   = m_dDominio[1];
        dSlope = (dMax - dMin) / 255;
        return (dSlope * newval + dMin);

}

BYTE CDMemory::TransfDataOutput()
{
        double dMin, dMax, dSlope;
```

```
        dMin    = m_dDominio[0];
        dMax    = m_dDominio[1];
        dSlope = (dMax - dMin) / 255;

        if (dSlope)
                return (BYTE)((m_dValActual - dMin) / dSlope);
        else
                return (BYTE)(m_dValActual - dMin);
}
// FuzzyVar.h: interface for the CFuzzyVar class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_FUZZYVAR_H__E4D0FA05_EA8F_11D3_9F48_188A01C10100__INCLUDED_)
#define AFX_FUZZYVAR_H__E4D0FA05_EA8F_11D3_9F48_188A01C10100__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "BaseCom.h"
#include "PDB.hpp"

typedef struct  {
        int Defuzz;
        int     Implic;
        int     AND;
} stMethods;

struct stFuzzyTerm {
        double m_dParam[3];
        int         m_nGeometry;
        void    Copy(stFuzzyTerm *source);
};

#if _MSC_VER > 1020
template <> void AFXAPI SerializeElements<stFuzzyTerm*>(CArchive &ar, stFuzzyTerm **ppElements, int nCount);
#else
void SerializeElements(CArchive &ar, stFuzzyTerm **ppElements, int nCount);
#endif

typedef CTypedPtrArray<CPtrArray, stFuzzyTerm*> CPtrArrToFZT;

class CFuzzyVar : public CBaseCom
{
public:
        void DeleteFuzzySets();
        double TransformXY(double value, BOOL flag=FALSE);
        void DefineFuzzySets(CFuzzyVar* source=NULL);
        virtual int Read(stComPort* pstPort, CString &info);
        virtual int Write(stComPort *pstPort, int newval);
        BYTE TransfDataOutput();
        double AssignDataInput(int newval);
        UINT TypeofUse();
        void Serialize(CArchive &ar);
        void DeleteExtern(PDB *pPDB);
        BOOL CreateExtern(PDB* pPDB);
        void DetachBDR(CString bdrname);
        void AttachBDR(CString bdrname, CString mode);
        UINT DetectTypeUpdate(CFuzzyVar &Varcomp);
        void SetTermNames();
        CFuzzyVar();
        virtual void OnDraw(CDC& dc);
        void Copy(CFuzzyVar* source, UINT updates = 55);
        CFuzzyVar(CFuzzyVar* source);
        virtual ~CFuzzyVar();
        CMapStringToString m_MapAttach;
        CStringArray m_saNameSets;
        stMethods m_nMethod;
        CString m_sUnits;
        CString m_sEquation;
        int             m_nSets;
        int             m_nOffset;
        int             m_nGeometry;
        double  m_dActual;
```

```
        double m_dDefval;
        float  m_fAlfacut;
        BOOL   m_bType;
        BOOL   m_bLimits;
        BOOL   m_bSymmetry;
        UINT   m_uBehave;
        UINT   m_uBoolWeight;
        CPtrArrToPZT m_astPzyTerms;
protected:
        void PzyPlotVar(char *Varid,PDB *PDBptr[],int PzySetCnt,int Medium,int *statusPtr);
        DECLARE_SERIAL(CPuzzyVar)
};

#endif // !defined(AFX_PUZZYVAR_H__E4D0FA05_EA8F_11D3_9F48_188A01C10100__INCLUDED_)




// PuzzyVar.cpp: implementation of the CPuzzyVar class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "VitroFuzz.h"
#include "Globales.h"
#include "PuzzyVar.h"
#include "XSYSctl.h"
#include "XPZYctl.h"
#include "Puzzy.hpp"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

extern CXSYSctl XSYSctl;
extern CXPZYctl XPZYctl;
extern CString Terminos[][7];
extern int status;

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

void stPuzzyTerm::Copy(stPuzzyTerm *source)
{
        for (int j=0; j < 3; j++)
                m_dParam[j] = source->m_dParam[j];

        m_nGeometry = source->m_nGeometry;
}

IMPLEMENT_SERIAL(CPuzzyVar, CBaseCom, 0)
CPuzzyVar::CPuzzyVar()
{
        m_bCodific      = TRUE;

        m_sUnits        = "Unidades";
        m_nGeometry     = TRIANGLE;
        m_dActual       = 0.0;
        m_dDefval       = 0.0;
        m_fAlfacut      = (float)0.05;
        m_bLimits       = TRUE;
        m_bSymmetry     = TRUE;
        m_bType         = FALSE;
        m_nOffset       = 0;
        m_nSets         = 0;
        m_uBehave       = TYPEFUZZY;
        m_uBoolWeight = 100;
        m_saNameSets.SetSize(0,1);
```

```
        double m_dDefval;
        float  m_fAlfacut;
        BOOL   m_bType;
        BOOL   m_bLimits;
        BOOL   m_bSymmetry;
        UINT   m_uBehave;
        UINT   m_uBoolWeight;
        CPtrArrToFZT m_astFzyTerms;
protected:
        void FzyPlotVar(char *Varid,FDB *FDBptr[],int FzySetCnt,int Medium,int *statusPtr);
        DECLARE_SERIAL(CFuzzyVar)
};

#endif // !defined(AFX_FUZZYVAR_H__E4D0FA05_EA8F_11D3_9F48_188A01C10100__INCLUDED_)
```

```
// FuzzyVar.cpp: implementation of the CFuzzyVar class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "VitroFuzz.h"
#include "Globales.h"
#include "FuzzyVar.h"
#include "XSYSctl.h"
#include "XFZYctl.h"
#include "Fuzzy.hpp"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

extern CXSYSctl XSYSctl;
extern CXFZYctl XFZYctl;
extern CString Terminos[][7];
extern int status;

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

void stFuzzyTerm::Copy(stFuzzyTerm *source)
{
        for (int j=0; j < 3; j++)
                m_dParam[j] = source->m_dParam[j];

        m_nGeometry = source->m_nGeometry;
}

IMPLEMENT_SERIAL(CFuzzyVar, CBaseCom, 0)
CFuzzyVar::CFuzzyVar()
{
        m_bCodific      = TRUE;

        m_sUnits        = "Unidades";
        m_nGeometry     = TRIANGLE;
        m_dActual       = 0.0;
        m_dDefval       = 0.0;
        m_fAlfacut      = (float)0.05;
        m_bLimits       = TRUE;
        m_bSymmetry     = TRUE;
        m_bType         = FALSE;
        m_nOffset       = 0;
        m_nSets         = 0;
        m_uBehave       = TYPEFUZZY;
        m_uBoolWeight = 100;
        m_saNameSets.SetSize(0,1);
```

```
        }
        dc.Rectangle(CRect(point, m_sizeRect));
        if (m_bGetFocus) dc.SelectObject(oldpen);
        dc.TextOut((point.x + 250) - lencad.cx,
                                point.y - 10, m_sName);
        dc.StretchBlt(point.x + 5, point.y - 3,
                    sizebmp.cx, -sizebmp.cy, &auxDC, 0, 0,
                    44, 19, SRCCOPY);
        return;
}

void CFuzzyVar::SetTermNames()
{
        CString term, lista;

        lista = Terminos[m_nSets-2][m_nOffset];
        m_saNameSets.RemoveAll();
        m_saNameSets.SetSize(m_nSets,1);
        for (int j=0; j < m_nSets; j++)
        {
                term = GetNextTerm(lista);
                m_saNameSets.SetAt(j, term);
        }
        return;
}


UINT CFuzzyVar::DetectTypeUpdate(CFuzzyVar &Varcomp)
{
        UINT updates = 0;
        if (m_nOffset != Varcomp.m_nOffset) updates |= UPDATEDESC;
        if (m_nSets     != Varcomp.m_nSets)  updates |= UPDATESETS;
        if (m_sName   != Varcomp.m_sName)    updates |= UPDATENAME;
        if (m_bType     != Varcomp.m_bType)
        {
                if (!m_MapAttach.IsEmpty())
                {
                        AfxMessageBox(IDS_ERR_CHGVAR_ISLINK,
                                                MB_OK | MB_ICONSTOP, 0);
                        updates ^= 0x8;
                }
                else
                        updates |= UPDATETYPE;
        }
        Copy(&Varcomp, updates);
        return updates;
}

void CFuzzyVar::AttachBDR(CString bdrname, CString mode)
{
        m_MapAttach.SetAt(bdrname, mode);
}

void CFuzzyVar::DetachBDR(CString bdrname)
{
        CString mode;
        if (m_MapAttach.Lookup(bdrname, mode))
                m_MapAttach.RemoveKey(bdrname);
}

BOOL CFuzzyVar::CreateExtern(PDB *pPDB)
{
        double param1, param2, param3;
        CString desc;
        FDB *p_FDBTerm    = NULL;
        FDB *p_aFDBFzyVar[20];
        VDB *p_VDBSolution = NULL;
        FDB *p_FDB;
        FSV *p_FSV;
        stFuzzyTerm *p_stFzy;
        char msg[50];

        if (!m_bType)
        {
                p_VDBSolution = VarCreateScalar(m_sName, REAL,
```

```
                m_dDominio, "0", &status);
        p_VDBSolution->VDBdefuzzmethod = m_nMethod.Defuzz;
        p_VDBSolution->VDBfzyimplmethod = m_nMethod.Implic;
        p_VDBSolution->VDBfzyclassAnd = m_nMethod.AND;
        XSYSctl.MdlLinkVDB(p_VDBSolution, pPDB, &status);
        if (!(XFZYctl.FzyAddFZYctl(p_VDBSolution, &p_FDB, &p_FSV, &status)))
        {
                status = 1;
                MtsSendError(12, "RunBDR", "Fallo en ejecución");
                return FALSE;
        }
    }
    for (int j=0; j < m_nSets; j++)
    {
        p_FDBTerm = new FDB;
        if (p_FDBTerm == NULL) return FALSE;
        FzyInitFDB(p_FDBTerm);
        desc = m_sName + '_' + m_saNameSets.GetAt(j);
        strcpy(p_FDBTerm->FDBid, desc);
        p_stFzy = m_astFzyTerms.GetAt(j);

        param1 = TransformXY(p_stFzy->m_dParam[0], TRUE);
        param2 = TransformXY(p_stFzy->m_dParam[1], TRUE);
        param3 = TransformXY(p_stFzy->m_dParam[2], TRUE);

        switch (p_stFzy->m_nGeometry)
        {
        case LEFTSHOULDER:
                p_FDBTerm->FDBdomain[0] = m_dDominio[0];
                p_FDBTerm->FDBdomain[1] = (m_bType) ? param2 : m_dDominio[1];
                FzyShoulderedCurve(p_FDBTerm, LEFTSHOULDER, param1, param2, &status);
                break;
        case RITESHOULDER:
                p_FDBTerm->FDBdomain[0] = (m_bType) ? param1 : m_dDominio[0];
                p_FDBTerm->FDBdomain[1] = m_dDominio[1];
                FzyShoulderedCurve(p_FDBTerm, RITESHOULDER, param2, param1, &status);
                break;
        case PI:
                break;
        case BETA:
                break;
        case GAUSS:
                break;
        case TRIANGLE:
                p_FDBTerm->FDBdomain[0] = (m_bType) ? param1 : m_dDominio[0];
                p_FDBTerm->FDBdomain[1] = (m_bType) ? param3 : m_dDominio[1];
                FzyTriangleCurve(p_FDBTerm, param1, param2, param3, &status);
                break;
        case DECLINE:
                break;
        case GROWTH:
                break;
        }

        XSYSctl.MdlLinkFDB(p_FDBTerm, pPDB, &status);

        if (m_bType == TIPOVAROUT)
                p_VDBSolution->VDBfuzzysets[j] = p_FDBTerm;
        p_aFDBFzyVar[j] = p_FDBTerm;
    }

    strcpy(msg, "Variable: ");
    strcat(msg, m_sName);
    FzyPlotVar(msg, p_aFDBFzyVar, m_nSets, SYSMODFILE, &status);

    return TRUE;
}

void CFuzzyVar::DeleteExtern(PDB *pPDB)
{
    int i;
    char name[50];
    for (i=0; i < m_nSets; i++)
    {
        strcpy(name, m_sName + "_" + m_saNameSets[i]);
```

```
            XSYSctl.MdlRemoveFDB(name, pPDB, &status);
        }

        if (m_bType == TIPOVAROUT)
        {
            strcpy(name, m_sName);
            XSYSctl.MdlRemoveVDB(name, pPDB, &status);
        }
}
void CFuzzyVar::Serialize(CArchive &ar)
{
        stFuzzyTerm *pstFzyTerm;
        WORD nCount;
        int  j, i;

        CBaseCom::Serialize(ar);

        if (ar.IsStoring())
        {
            ar << m_bLimits;
            ar << m_bSymmetry;
            ar << m_bType;
            ar << m_dActual;
            ar << m_fAlfacut;
            ar << m_dDefval;
            ar << m_nGeometry;
            ar << m_nMethod.AND;
            ar << m_nMethod.Defuzz;
            ar << m_nMethod.Implic;
            ar << m_nOffset;
            ar << m_nSets;
            ar << m_sUnits;
            ar << m_uBehave;
            ar << m_uBoolWeight;

            ar << m_sEquation;

            nCount = (WORD)m_astFzyTerms.GetSize();
            ar << nCount;
            for (j=0; j < nCount; j++)
            {
                pstFzyTerm = m_astFzyTerms.GetAt(j);
                for (i=0; i < 3; i++)
                    ar << pstFzyTerm->m_dParam[i];

                ar << pstFzyTerm->m_nGeometry;
            }
        }
        else
        {
            ar >> m_bLimits;
            ar >> m_bSymmetry;
            ar >> m_bType;
            ar >> m_dActual;
            ar >> m_fAlfacut;
            ar >> m_dDefval;
            ar >> m_nGeometry;
            ar >> m_nMethod.AND;
            ar >> m_nMethod.Defuzz;
            ar >> m_nMethod.Implic;
            ar >> m_nOffset;
            ar >> m_nSets;
            ar >> m_sUnits;
            ar >> m_uBehave;
            ar >> m_uBoolWeight;

            ar >> m_sEquation;

            ar >> nCount;
            for (j=0; j < nCount; j++)
            {
                pstFzyTerm = new stFuzzyTerm;
                for (i=0; i < 3; i++)
                    ar >> pstFzyTerm->m_dParam[i];
```

```
                    ar >> pstFzyTerm->m_nGeometry;
                    m_astFzyTerms.Add(pstFzyTerm);
                }
        }
                m_saNameSets.Serialize(ar);
                m_MapAttach.Serialize(ar);
}

UINT CFuzzyVar::TypeofUse()
{
        POSITION pos;
        CString  namebdr, Key;
        UINT     byte=0;

        pos = m_MapAttach.GetStartPosition();
        if (pos == NULL)
                return MODENONE;

        while (pos)
        {
                m_MapAttach.GetNextAssoc(pos, namebdr, Key);
                byte |= (Key == "MODEIN") ? MODEIN : MODEOUT;
        }

        return byte;
}

double CFuzzyVar::AssignDataInput(int newval)
{
        double dMin, dMax, dSlope;

        dMin   = m_dDominio[0];
        dMax   = m_dDominio[1];
        dSlope = (dMax - dMin) / 255;
        return (dSlope * newval + dMin);
}

BYTE CFuzzyVar::TransfDataOutput()
{
        double dMin, dMax, dSlope;

        dMin   = m_dDominio[0];
        dMax   = m_dDominio[1];
        dSlope = (dMax - dMin) / 255;
        if (dSlope)
                return (BYTE)((m_dActual - dMin) / dSlope);
        else
                return (BYTE)(m_dActual - dMin);
}

int CFuzzyVar::Write(stComPort *pstPort, int newval)
{
        m_dActual = (m_bCodific) ? AssignDataInput(newval) : (double)newval;
        return NONE_ACTION;
}

int CFuzzyVar::Read(stComPort *pstPort, CString &info)
{
        if (m_etype == t_VarInp)
        {
                if (pstPort->HeadsEnable)
                        info.Format("%d,%d,%d,%d,%d", m_uBytesRecive, m_etype, pstPort->HeadRead, m_byID,
DONTCARE);
                else
                        info.Format("%d,%d", m_uBytesRecive, m_byID);
                return NONE_ACTION;
        }
        else
        {
                BYTE val = (m_bCodific) ? TransfDataOutput() : (UINT)m_dActual;
                info.Format("%d,%d,%d,%d,%d", m_uBytesRecive, m_etype, pstPort->HeadWrite, m_byID, val);
                return WM_SENDMESSAGE;
        }
}
```

```
void CFuzzyVar::DefineFuzzySets(CFuzzyVar *source)
{
	stFuzzyTerm *p_stFzyTerm;
	int		j, cont=0;
	BOOL	flag=FALSE;
	double liminf, limsup, ancho, limant;
	DeleteFuzzySets();
	if (source != NULL)
		cont = source->m_astFzyTerms.GetSize();
	if (cont >= m_nSets) flag = TRUE;
	if (!flag)
	{
		liminf	= limant = m_dDominio[0];
		limsup	= m_dDominio[1];
		ancho	= (limsup - liminf) / (m_nSets + 1);
	}
	for (j=0; j < m_nSets; j++)
	{
		p_stFzyTerm = new stFuzzyTerm;
		if (p_stFzyTerm == NULL)
		{
			AfxMessageBox("No es posible crear los terminos");
			return;
		}
		if (flag)
			p_stFzyTerm->Copy(source->m_astFzyTerms[j]);
		else
		{
			switch (m_nGeometry)
			{
			case TRIANGLE:
				if (j == 0 && m_bLimits == HEDGETYPEUP)
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[1] = TransformXY(liminf+2*ancho);
					p_stFzyTerm->m_dParam[2] = 0;
					p_stFzyTerm->m_nGeometry = LEFTSHOULDER;
				}
				else if (j == m_nSets-1 && m_bLimits == HEDGETYPEUP)
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf);
					p_stFzyTerm->m_dParam[1] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[2] = 0;
					p_stFzyTerm->m_nGeometry = RITESHOULDER;
				}
				else
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf);
					p_stFzyTerm->m_dParam[1] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[2] = TransformXY(liminf+2*ancho);
					p_stFzyTerm->m_nGeometry = TRIANGLE;
				}
				break;
			case PI:
			case BETA:
			case GAUSS:
				if (j == 0 && m_bLimits == HEDGETYPEUP)
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf);
					p_stFzyTerm->m_dParam[1] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[2] = TransformXY(liminf+2*ancho);
					p_stFzyTerm->m_nGeometry = DECLINE;
				}
				else if (j == m_nSets-1 && m_bLimits == HEDGETYPEUP)
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf);
					p_stFzyTerm->m_dParam[1] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[2] = TransformXY(liminf+2*ancho);
					p_stFzyTerm->m_nGeometry = GROWTH;
				}
				else if (m_nGeometry == PI)
				{
					p_stFzyTerm->m_dParam[0] = TransformXY(liminf+ancho);
					p_stFzyTerm->m_dParam[1] = TransformXY(ancho);
```

```
                                        p_stFzyTerm->m_dParam[2] = 0;
                                        p_stFzyTerm->m_nGeometry = PI;
                                }
                                else if (m_nGeometry == BETA)
                                {
                                        p_stFzyTerm->m_dParam[0] = TransformXY(liminf+ancho);
                                        p_stFzyTerm->m_dParam[1] = TransformXY(ancho);
                                        p_stFzyTerm->m_dParam[2] = 0;
                                        p_stFzyTerm->m_nGeometry = BETA;
                                }
                                else
                                {
                                        p_stFzyTerm->m_dParam[0] = TransformXY(liminf+ancho);
                                        p_stFzyTerm->m_dParam[1] = 0.10;
                                        p_stFzyTerm->m_dParam[2] = 0;
                                        p_stFzyTerm->m_nGeometry = GAUSS;
                                }
                                break;
                        case MEMSERIES:
                                break;
                        }

                        liminf = liminf + ancho;
                }

                m_astFzyTerms.Add(p_stFzyTerm);
        }
}

double CFuzzyVar::TransformXY(double value, BOOL flag)
{
        double liminf, limsup, pendiente, transform;

        liminf    = m_dDominio[0];
        limsup    = m_dDominio[1];
        pendiente = (limsup - liminf) * 0.01;

        if (flag)
                transform = pendiente * value + liminf;
        else
                transform = (value - liminf) / pendiente;

        return transform;
}


void CFuzzyVar::DeleteFuzzySets()
{
        stFuzzyTerm *p_stFzyTerm;

        int cont = m_astFzyTerms.GetSize();
        for (int j=0; j < cont; j++)
        {
                p_stFzyTerm = m_astFzyTerms.GetAt(j);
                delete p_stFzyTerm;
        }
        m_astFzyTerms.RemoveAll();
}

void CFuzzyVar::FzyPlotVar(char *Varid, FDB *FDBptr[], int FzySetCnt, int Medium, int *statusPtr)
{
        #define PLOTROWS   26
        #define PLOTCOLS   70
        #define BIGNUMBER  999999999

        char        WkArea[PLOTROWS][PLOTCOLS+1];
    int         lencad,i,j,k,PltHgt,Sidx,N,HorzPos,VertPos,Lochp;
    int         ScalingIdx=1;
    int         ScaleCtl =4;
    double      x,Domain[2],Range,Scalar;
    FILE        *outfp;

    const char  Symbol[]=".*+:-!@&#$=";
    const int   SymCnt=11;
    const int   PlotHeight[]    = {0,11,21,26,51};
```

```
    const float ScalingFactor[] = {0,10.0,20.0,25.0,50.0};

        CString         term;

    *statusPtr=0;
    if(FzySetCnt==0) return;
    if(FzySetCnt>SymCnt) FzySetCnt=SymCnt;
    outfp=MtsGetSystemFile(Medium);

        lencad = m_sName.GetLength()+1;

    for(i=0;i<26;i++)
     {
       for(j=0;j<70;j++) WkArea[i][j]=' ';
      WkArea[i][j]='\0';
     }
   Domain[0]=BIGNUMBER;
   Domain[1]=0;
   for(i=0;i<FzySetCnt;i++)
    {
     Domain[0]=min(FDBptr[i]->FDBdomain[0],Domain[0]);
     Domain[1]=max(FDBptr[i]->FDBdomain[1],Domain[1]);
    }
   Range=Domain[1]-Domain[0];
   for(i=0;i<FzySetCnt;i++)
    {
     for(k=0;k<VECMAX;k+=ScaleCtl)
        {
         Sidx=ScalingFactor[ScalingIdx];
         VertPos=(int)(Sidx*(FDBptr[i]->FDBvector[k]+NUDGE));
         Scalar=FzyGetScalar(FDBptr[i],k,statusPtr);
         Lochp=(int)(((Scalar-Domain[0])/Range)*VECMAX);
         HorzPos= Lochp / ScaleCtl;
         if((VertPos+1>=0)&&(VertPos+1<PLOTROWS))
              WkArea[VertPos+1][HorzPos]=Symbol[i];
        }
     }
     fputc('\f',outfp);
     fputc('\n',outfp);
     fprintf(outfp,"%s\n",Varid);
     fprintf(outfp,"%s%10.2f%s%10.2f\n", "Dominio [UofD]: ",
          Domain[0]," a ",Domain[1]);
     fputc('\n',outfp);
     fputc('\n',outfp);
     PltHgt=PlotHeight[ScalingIdx];
     for(N=PltHgt,i=0;i<PltHgt;i++,--N)
       {
        j=PltHgt-i;
        x=(j-1)/ScalingFactor[ScalingIdx];
        fprintf(outfp,"%10.2f%s\n",x,&WkArea[N][0]);
       }
     fprintf(outfp,"%s%s",
      "          0---|---|---|---|---|---|---|---|---|---|---|",
      "---|---|---|---0\n");
     x=(Domain[1]-Domain[0])/8.0;
     fputs("      ",outfp);
     for(i=0;i<9;i++) fprintf(outfp,"%8.2f",Domain[0]+(x*i));
     fputc('\n', outfp);
     for(i=0;i<FzySetCnt;i++)
      {
          term = FDBptr[i]->FDBid;
          term = term.Mid(lencad);
       fprintf(outfp,"%s%c%s%s\n",
        "      ",Symbol[i],          "    FuzzySet:    ",(LPCTSTR)term);
       fprintf(outfp,"%s%s\n","          Descripcion: ",FDBptr[i]->FDBdesc);
       fprintf(outfp,
         "%s%10.2f%s%10.2f\n","          Dominio:     ",
           FDBptr[i]->FDBdomain[0]," to ",FDBptr[i]->FDBdomain[1]);
       if(FDBptr[i]->FDBalfacut>0)
         fprintf(outfp,
          "%s%10.2f\n",          "          CorteAlfa:   ",FDBptr[i]->FDBalfacut);
      }
     fputc('\n',outfp);
        fflush(outfp);
}
```

```
// Policy.h: interface for the CPolicy class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_POLICY_H__E4D0FA04_EA8F_11D3_9F48_188A01C10100__INCLUDED_)
#define AFX_POLICY_H__E4D0FA04_EA8F_11D3_9F48_188A01C10100__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "BDR.h"
#include "Text.h"

typedef CTypedPtrMap<CMapStringToOb, CString, CBDR*> CMapKeyToBDR;
typedef CTypedPtrArray<CObArray, CText*>                    CArrayText;
typedef CTypedPtrArray<CObArray, CBDR*>                     CArrayBDR;
typedef CTypedPtrMap<CMapStringToOb, CString, CBaseCom*>  CMapKeyToBaseCom;
typedef CTypedPtrMap<CMapStringToOb, CString, stComPort*> CMapKeyToComm;

class CPolicy : public CObject
{
public:
        UINT CheckID(CFuzzyVar *source, BYTE ID, UINT numPort, UINT numBit);
        BYTE CreateID(UINT numPort);
        BOOL CheckRunIndex(UINT index);
        UINT GetLimRunIndex(BOOL bLimit);
        UINT CreateRunIndex();
        void Serialize(CArchive &ar);
        void DeleteExtern();
        BOOL CreateExtern();
        BOOL DeleteBDR(CString bdrname);
        BOOL DeleteFuzzyVar(CString varname);
        void SetAlfacut(float alfa);
        void SetActivePolicy();
        BOOL InitPolicy();
        void RunMapBDR();
        void SortStringArray(CStringArray &vars);
        void ClosePolicy();
        CString CreateBDRname();
        CBDR* CreateBDR(CBDR &source);
        CFuzzyVar* FindFuzzyVar(CString keyword);
        CFuzzyVar* CreateFuzzyVar(CFuzzyVar* source);
        CPolicy(CString name);
        CPolicy();
        virtual ~CPolicy();
        CString                 m_sName;
        CString                 m_sComent;
        CString                 m_sNameBase;
        CString                 m_sPathBase;
        CString                 m_sComentBase;
        CArrayText              m_aText;
        PDB*                    m_pPDB;
        CMapKeyToFuzzyVar m_MapVAR;
        CMapKeyToBDR      m_MapBDR;
        CArrayBDR               m_aRunBDR;
        BOOL                    m_RunBDRs;
        BOOL                    m_bCreateBase;
        float                   m_fAlfacut;
        stComPort               m_astPortCom[4];
        CMapKeyToBaseCom  m_MapObjectRun;
protected:
        void InitialPortSettings();
        DECLARE_SERIAL(CPolicy)
};

#endif // !defined(AFX_POLICY_H__E4D0FA04_EA8F_11D3_9F48_188A01C10100__INCLUDED_)
```

```
// Policy.cpp: implementation of the CPolicy class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "VitroFuzz.h"
#include "Policy.h"
#include "XSYSctl.h"
#include "XFZYctl.h"
#include "Fuzzy.hpp"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

extern CXSYSctl XSYSctl;
extern CXFZYctl XFZYctl;
extern int status;

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

IMPLEMENT_SERIAL(CPolicy, CObject, 0)

CPolicy::CPolicy(CString name)
{
        m_sName         = name;
        m_sComent       = "";
        m_sNameBase = "";
        m_sPathBase     = "";
        m_sComentBase = "";
        m_RunBDRs     = FALSE;
        m_bCreateBase = FALSE;
        m_pPDB          = NULL;
        m_fAlfacut      = (float)0.05;
        m_MapVAR.RemoveAll();
        m_MapBDR.RemoveAll();
        m_aText.RemoveAll();
        InitialPortSettings();
}

CPolicy::CPolicy()
{
        m_sName         = "";
        m_sComent       = "";
        m_sNameBase = "";
        m_sPathBase     = "";
        m_sComentBase = "";
        m_RunBDRs     = FALSE;
        m_bCreateBase = FALSE;
        m_pPDB          = NULL;
        m_fAlfacut      = (float)0.05;
        m_MapVAR.RemoveAll();
        m_MapBDR.RemoveAll();
        m_aText.RemoveAll();
        InitialPortSettings();
}

CPolicy::~CPolicy()
{
}

CFuzzyVar* CPolicy::FindFuzzyVar(CString keyword)
{
        CFuzzyVar* p_FuzzyVar = NULL;
        m_MapVAR.Lookup(keyword, p_FuzzyVar);
        return p_FuzzyVar;
}
```

```
CFuzzyVar* CPolicy::CreateFuzzyVar(CFuzzyVar* source)
{
	CFuzzyVar* p_newvar = new CFuzzyVar(source);
	if (p_newvar == NULL) return NULL;
	p_newvar->m_sName = source->m_sName;
	p_newvar->m_byID = CreateID(PORTCOMM2);
	m_MapVAR.SetAt(p_newvar->m_sName, p_newvar);
	return p_newvar;
}

CBDR* CPolicy::CreateBDR(CBDR &source)
{
	CBDR	*p_BDR=NULL;
	if (source.m_sName.IsEmpty())
	{
		source.m_sName = CreateBDRname();
		source.m_uRunIndex = CreateRunIndex();
	}

	if ((p_BDR = new CBDR(source)) == NULL) return NULL;
	CString varname;
	CFuzzyVar*	pVar;
	POSITION pos;
	pos = source.m_MapVarInp.GetStartPosition();
	while (pos)
	{
		source.m_MapVarInp.GetNextAssoc(pos, varname, pVar);
		m_MapVAR.Lookup(varname, pVar);
		p_BDR->m_MapVarInp.SetAt(varname, pVar);
		pVar->AttachBDR(p_BDR->m_sName, "MODEIN");
	}

	pos = source.m_MapVarOut.GetStartPosition();
	while (pos)
	{
		source.m_MapVarOut.GetNextAssoc(pos, varname, pVar);
		m_MapVAR.Lookup(varname, pVar);
		p_BDR->m_MapVarOut.SetAt(varname, pVar);
		pVar->AttachBDR(p_BDR->m_sName, "MODEOUT");
	}
	p_BDR->InitialUpdate(m_MapVAR);
	m_MapBDR.SetAt(p_BDR->m_sName, p_BDR);

	return p_BDR;
}

CString CPolicy::CreateBDRname()
{
	CBDR	*p_BDR;
	CString	Name;
	char buff[5];

	Name = "BDR_";
	for (int i=1;;i++)
		if (!(m_MapBDR.Lookup(Name + _itoa(i, buff, 10), p_BDR)))
			return (Name + _itoa(i, buff, 10));
}

void CPolicy::ClosePolicy()
{
	CFuzzyVar* p_VAR;
	CBDR*	p_BDR;
	CText*	pText;
	CString	Key;
	POSITION	pos;

	pos = m_MapBDR.GetStartPosition();
	while (pos != NULL)
	{
		m_MapBDR.GetNextAssoc(pos, Key, p_BDR);
		delete p_BDR;
	}
	m_MapBDR.RemoveAll();

	pos = m_MapVAR.GetStartPosition();
```

```
        while (pos != NULL)
        {
                m_MapVAR.GetNextAssoc(pos, Key, p_VAR);
                delete p_VAR;
        }
        m_MapVAR.RemoveAll();

        for (int i=0; i<m_aText.GetSize(); i++)
        {
                pText = m_aText.GetAt(i);
                delete pText;
        }
        m_aText.RemoveAll();

        return;
}

void CPolicy::SortStringArray(CStringArray &vars)
{
        CString temp;
        int cont, i, j;

        cont = vars.GetSize()-1;

        for (i=0; i<cont; i++)
        {
                if (m_MapVAR[vars.GetAt(i)]->m_nSets >
                        m_MapVAR[vars.GetAt(i+1)]->m_nSets)
                {
                        temp = vars.GetAt(i);
                        vars.SetAt(i, vars.GetAt(i+1));
                        vars.SetAt(i+1,temp);
                        if (i>0)
                                for (j=i; j>0; j--)
                                {
                                        if (m_MapVAR[vars.GetAt(j)]->m_nSets <
                                                m_MapVAR[vars.GetAt(j-1)]->m_nSets)
                                        {
                                                temp = vars.GetAt(j);
                                                vars.SetAt(j, vars.GetAt(j-1));
                                                vars.SetAt(j-1, temp);
                                        }
                                }
                }
        }
}

void CPolicy::RunMapBDR()
{
        CBDR* pBDR;
        int       count;
        count = m_aRunBDR.GetSize();
        for (int i=0; i < count; i++)
        {
                pBDR = m_aRunBDR.GetAt(i);

                pBDR->RunBDR(m_pPDB, m_MapVAR);
        }
        XFZYctl.FzyResetFZYctl(&status);
}

BOOL CPolicy::InitPolicy()
{
        m_pPDB = XSYSctl.MdlCreatePolicy(m_sName, MODELADD, &status);
        if (m_pPDB == NULL) return NULL;
        int Hdgcnt;
        XSYSctl.MdlInsertHedges(m_pPDB, &Hdgcnt, &status);
        return (TRUE);
}

void CPolicy::SetActivePolicy()
{
        PDB* pXPolicy;
        char name[20];
```

```
        strcpy(name, m_sName);

        pXPolicy = XSYSctl.MdlFindPDB(name, &status);
        XSYSctl.MdlSetPolicy(pXPolicy);
}

void CPolicy::SetAlfacut(float alfa)
{
        m_fAlfacut = alfa;
        XSYSctl.MdlSetAlfacut(alfa);
}

BOOL CPolicy::DeleteFuzzyVar(CString varname)
{
        CString bdrname, mode;
        CFuzzyVar* pVar;
        CBDR* pBDR;
        POSITION pos;
        if (!m_MapVAR.Lookup(varname, pVar))
        {
                AfxMessageBox(IDS_ERR_VARNOEXIST_INPOLICY,
                                                MB_OK | MB_ICONSTOP, 0);
                return FALSE;
        }
        pos = pVar->m_MapAttach.GetStartPosition();
        while (pos)
        {
                pVar->m_MapAttach.GetNextAssoc(pos, bdrname, mode);
                m_MapBDR.Lookup(bdrname, pBDR);
                if (pBDR != NULL)
                {
                        if (mode == "MODEIN")
                                pBDR->m_MapVarInp.RemoveKey(varname);
                        else
                                pBDR->m_MapVarOut.RemoveKey(varname);

                        pBDR->m_MapFAM.RemoveKey(varname);
                }
        }
        m_MapVAR.RemoveKey(varname);
        delete pVar;
        return TRUE;
}

BOOL CPolicy::DeleteBDR(CString bdrname)
{
        CString         varname;
        CFuzzyVar*      pVar;
        CBDR*           pBDR;
        POSITION        pos;

        if (!m_MapBDR.Lookup(bdrname, pBDR))
        {
                AfxMessageBox(IDS_ERR_BDRNOEXIST_INPOLICY,
                                                MB_OK | MB_ICONSTOP, 0);
                return FALSE;
        }
        m_MapBDR.RemoveKey(bdrname);
        pos = pBDR->m_MapVarInp.GetStartPosition();
        while (pos)
        {
                pBDR->m_MapVarInp.GetNextAssoc(pos, varname, pVar);
                pVar->DetachBDR(pBDR->m_sName);
        }

        pos = pBDR->m_MapVarOut.GetStartPosition();
        while (pos)
        {
                pBDR->m_MapVarOut.GetNextAssoc(pos, varname, pVar);
                pVar->DetachBDR(pBDR->m_sName);
        }
        delete pBDR;
        return TRUE;
}
```

```
BOOL CPolicy::CreateExtern()
{
	CFuzzyVar* pVar;
	CBaseCom*  pBaseCom;
	CBDR*      pBDR;
	POSITION   pos;
	CString    key;
	UINT       index;
	int                count, j;

	CWaitCursor wait;
	XFZYctl.FzyInitFZYctl(&status);
	if (!InitPolicy())
		return FALSE;
	pos = m_MapVAR.GetStartPosition();
	while (pos != NULL)
	{
		m_MapVAR.GetNextAssoc(pos, key, pVar);
		if (!pVar->CreateExtern(m_pPDB))
		{
			AfxMessageBox(IDS_ERR_VAREXTERN_NOCREATE,
							MB_OK | MB_ICONSTOP, 0);
			return FALSE;
		}
	}
	pos = m_MapBDR.GetStartPosition();
	while (pos != NULL)
	{
		m_MapBDR.GetNextAssoc(pos, key, pBDR);
		if (pBDR->m_bRun)
		{
			if (!pBDR->CreateExtern(m_pPDB, m_MapVAR))
			{
				AfxMessageBox(IDS_ERR_BDREXTERN_NOCREATE,
								MB_OK | MB_ICONSTOP, 0);
				return FALSE;
			}
		}
	}

	index = GetLimRunIndex(FALSE);
	count = m_MapBDR.GetCount();
	m_aRunBDR.RemoveAll();
	m_MapObjectRun.RemoveAll();

	for (;count;)
	{
		pos = m_MapBDR.GetStartPosition();
		while (pos)
		{
			m_MapBDR.GetNextAssoc(pos, key, pBDR);
			if (index == pBDR->m_uRunIndex)
			{
				if (pBDR->m_bRun)
					m_aRunBDR.Add(pBDR);
				count--;
			}
		}
		index++;
	}
	for (j=0; j<4; j++)
		m_astPortCom[j].Create = FALSE;
	count = m_aRunBDR.GetSize();

	for (j=0; j < count; j++)
	{
		pBDR = m_aRunBDR.GetAt(j);

		for (int k=0; k < 2; k++)
		{
			if (!k)
				pos = pBDR->m_MapVarInp.GetStartPosition();
			else
				pos = pBDR->m_MapVarOut.GetStartPosition();
```

```
                        while (pos)
                        {
                                if (!k)
                                        pBDR->m_MapVarInp.GetNextAssoc(pos, key, pVar);
                                else
                                        pBDR->m_MapVarOut.GetNextAssoc(pos, key, pVar);
                                m_astPortCom[pVar->m_uPortComm - 1].Create = TRUE;
                                key.Format("%d,%d", pVar->m_etype, pVar->m_byID);
                                if (!m_MapObjectRun.Lookup(key, pBaseCom))
                                        m_MapObjectRun.SetAt(key, (CBaseCom*)pVar);
                        }
                }
        }

        return TRUE;
}

void CPolicy::DeleteExtern()
{
        CFuzzyVar* pVar;
        CBDR*      pBDR;
        POSITION   pos;
        CString    key;
        long       memory;

        XFZYctl.FzyCloseFZYctl(&status);
        m_MapObjectRun.RemoveAll();
        m_aRunBDR.RemoveAll();
        pos = m_MapBDR.GetStartPosition();
        while (pos != NULL)
        {
                m_MapBDR.GetNextAssoc(pos, key, pBDR);
                pBDR->DeleteExtern();
        }
        pos = m_MapVAR.GetStartPosition();
        while (pos != NULL)
        {
                m_MapVAR.GetNextAssoc(pos, key, pVar);
                pVar->DeleteExtern(m_pPDB);
        }
        XSYSctl.MdlClosePolicy(m_pPDB, &memory, &status);
        delete m_pPDB;
        m_pPDB = NULL;

        return;
}

void CPolicy::Serialize(CArchive &ar)
{
        int j;

    CObject::Serialize(ar);

        if (ar.IsStoring())
        {
                ar << m_sName;
                ar << m_sComent;
                ar << m_RunBDRs;
                ar << m_fAlfacut;
                ar << m_sNameBase;
                ar << m_sComentBase;
                ar << m_sPathBase;
                for (j=0; j < 4; j++)
                {
                        ar << m_astPortCom[j].Settings;
                        ar << m_astPortCom[j].WorkMode;
                        ar << m_astPortCom[j].InBuff;
                        ar << m_astPortCom[j].OutBuff;
                        ar << m_astPortCom[j].Checksum;
                        ar << m_astPortCom[j].TimeOut;
                        ar << m_astPortCom[j].Repeater;
                        ar << m_astPortCom[j].HeadsEnable;
                        ar << m_astPortCom[j].HeadTransm;
                        ar << m_astPortCom[j].HeadRecept;
                        ar << m_astPortCom[j].HeadRead;
```

```
                    ar << m_astPortCom[j].HeadWrite;
                    ar << m_astPortCom[j].HeadNotify;
                    ar << m_astPortCom[j].HeadVariable;
                    ar << m_astPortCom[j].HeadMemory;
                    ar << m_astPortCom[j].HeadMessage;
                    ar << m_astPortCom[j].HeadOk;
                    ar << m_astPortCom[j].HeadErr;
                    ar << m_astPortCom[j].HeadErrTrans;
                    ar << m_astPortCom[j].TimeAutoMin;
                    ar << m_astPortCom[j].TimeAutoSeg;
                    ar << m_astPortCom[j].Create;
                    ar << m_astPortCom[j].PortID;
                    ar << m_astPortCom[j].InputLen;
                    ar << m_astPortCom[j].NullDiscard;
            }
        }
        else
        {
            ar >> m_sName;
            ar >> m_sComent;
            ar >> m_RunBDRs;
            ar >> m_fAlfacut;
            ar >> m_sNameBase;
            ar >> m_sComentBase;
            ar >> m_sPathBase;
            for (j=0; j < 4; j++)
            {
                    ar >> m_astPortCom[j].Settings;
                    ar >> m_astPortCom[j].WorkMode;
                    ar >> m_astPortCom[j].InBuff;
                    ar >> m_astPortCom[j].OutBuff;
                    ar >> m_astPortCom[j].Checksum;
                    ar >> m_astPortCom[j].TimeOut;
                    ar >> m_astPortCom[j].Repeater;
                    ar >> m_astPortCom[j].HeadsEnable;
                    ar >> m_astPortCom[j].HeadTransm;
                    ar >> m_astPortCom[j].HeadRecept;
                    ar >> m_astPortCom[j].HeadRead;
                    ar >> m_astPortCom[j].HeadWrite;
                    ar >> m_astPortCom[j].HeadNotify;
                    ar >> m_astPortCom[j].HeadVariable;
                    ar >> m_astPortCom[j].HeadMemory;
                    ar >> m_astPortCom[j].HeadMessage;
                    ar >> m_astPortCom[j].HeadOk;
                    ar >> m_astPortCom[j].HeadErr;
                    ar >> m_astPortCom[j].HeadErrTrans;
                    ar >> m_astPortCom[j].TimeAutoMin;
                    ar >> m_astPortCom[j].TimeAutoSeg;
                    ar >> m_astPortCom[j].Create;
                    ar >> m_astPortCom[j].PortID;
                    ar >> m_astPortCom[j].InputLen;
                    ar >> m_astPortCom[j].NullDiscard;
            }
        }

        m_MapVAR.Serialize(ar);
        m_MapBDR.Serialize(ar);
        m_aText.Serialize(ar);
}

UINT CPolicy::CreateRunIndex()
{
        CBDR* pBDR;
        CString bdrname;
        POSITION pos;
        UINT index = 0;
        BOOL flag=TRUE;

        while (flag)
        {
                flag = FALSE;
                pos = m_MapBDR.GetStartPosition();
                index++;
                while (pos)
                {
```

```
                        m_MapBDR.GetNextAssoc(pos, bdrname, pBDR);

                        if (index == pBDR->m_uRunIndex)
                                flag = TRUE;
                }
        }
        return index;
}

UINT CPolicy::GetLimRunIndex(BOOL bLimit)
{
        UINT index;
        POSITION pos;
        CBDR* pBDR;
        CString bdrname;

        index = (bLimit) ? 0 : 1000;

        pos = m_MapBDR.GetStartPosition();
        while (pos)
        {
                m_MapBDR.GetNextAssoc(pos, bdrname, pBDR);

                if (bLimit)
                {
                        if (pBDR->m_uRunIndex > index)
                                index = pBDR->m_uRunIndex;
                }
                else
                        if (pBDR->m_uRunIndex < index)
                                index = pBDR->m_uRunIndex;
        }
        return index;
}

BOOL CPolicy::CheckRunIndex(UINT index)
{
        POSITION pos;
        CBDR* pBDR;
        CString bdrname;
        BOOL flag = TRUE;

        pos = m_MapBDR.GetStartPosition();
        while (pos)
        {
                m_MapBDR.GetNextAssoc(pos, bdrname, pBDR);
                if (index == pBDR->m_uRunIndex)
                        flag = FALSE;
        }
        return flag;
}

void CPolicy::InitialPortSettings()
{
        int j;

        for (j=0; j < 4; j++)
        {
                m_astPortCom[j].Settings = "9600,n,8,1";
                m_astPortCom[j].WorkMode = 1;
                m_astPortCom[j].InBuff = 1024;
                m_astPortCom[j].OutBuff        = 512;
                m_astPortCom[j].Checksum = 2;
                m_astPortCom[j].TimeOut        = 100;
                m_astPortCom[j].Repeater = 2;
                m_astPortCom[j].HeadsEnable = TRUE;
                m_astPortCom[j].HeadTransm          = 0xcc;
                m_astPortCom[j].HeadRecept          = 0xaa;
                m_astPortCom[j].HeadRead            = 0x10;
                m_astPortCom[j].HeadWrite           = 0x11;
                m_astPortCom[j].HeadNotify          = 0x14;
                m_astPortCom[j].HeadVariable  = 0x04;
                m_astPortCom[j].HeadMemory          = 0x05;
                m_astPortCom[j].HeadMessage         = 0x06;
                m_astPortCom[j].HeadOk              = 0x12;
```

```
                m_astPortCom[j].HeadErr                 = 0x13;
                m_astPortCom[j].HeadErrTrans  = 0x01;
                m_astPortCom[j].TimeAutoMin             = 2;
                m_astPortCom[j].TimeAutoSeg             = 0;
                m_astPortCom[j].Create                  = FALSE;
                m_astPortCom[j].PortID                  = j;
                m_astPortCom[j].InputLen                = 0;
                m_astPortCom[j].NullDiscard             = FALSE;
        }
}

BYTE CPolicy::CreateID(UINT numPort)
{
        CFuzzyVar *pVar;
        CString varname;
        POSITION pos;
        BYTE ID = 0;
        BOOL flag=TRUE;

        while (flag)
        {
                flag = FALSE;
                pos = m_MapVAR.GetStartPosition();
                ID++;
                while (pos)
                {
                        m_MapVAR.GetNextAssoc(pos, varname, pVar);

                        if (ID == pVar->m_byID &&
                                pVar->m_uPortComm == numPort)
                                        flag = TRUE;
                }
        }
        return ID;
}

UINT CPolicy::CheckID(CFuzzyVar *source, BYTE ID, UINT numPort, UINT numBit)
{
        POSITION pos;
        CFuzzyVar *pVar;
        CString varname;
        UINT flag = 0;

        pos = m_MapVAR.GetStartPosition();
        while (pos)
        {
                m_MapVAR.GetNextAssoc(pos, varname, pVar);

                if (source->m_uBehave == TYPEBOOLEAN)
                {
                        if (source != pVar &&
                                ID == pVar->m_byID &&
                                numPort == pVar->m_uPortComm &&
                                numBit  == pVar->m_uBytesSend)
                                        flag = 1;
                }
                else
                {
                        if (source != pVar &&
                                ID == pVar->m_byID &&
                                numPort == pVar->m_uPortComm)
                                        flag = 2;
                }
        }
        return flag;
}
```

```
// VitroFuzzDoc.h : interface of the CVitroFuzzDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_VITROFUZZDOC_H__A84E23CD_E9FC_11D3_9F48_188A01C10100__INCLUDED_)
#define AFX_VITROFUZZDOC_H__A84E23CD_E9FC_11D3_9F48_188A01C10100__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Policy.h"
#include "DMemory.h"

typedef CTypedPtrMap<CMapStringToOb, CString, CPolicy*>      CMapKeyToPolicy;
typedef CTypedPtrMap<CMapStringToOb, CString, CDMemory*> CMapKeyToDMemory;

class CVitroFuzzView;
class CEditorView;
class CControlView;

class CVitroFuzzDoc : public CDocument
{
protected:
        CVitroFuzzDoc();
        DECLARE_DYNCREATE(CVitroFuzzDoc)

// Attributes
public:

// Operations
public:
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CVitroFuzzDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        virtual void DeleteContents();
        virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
        //}}AFX_VIRTUAL

// Implementation
public:
        CString CreateName(etypeobj type, int mode);
        BYTE CreateID(etypeobj type, UINT numPort);
        BOOL CheckID(CBaseCom *source, BYTE ID, UINT numPort);
        BOOL UpdateLinksVar(CFuzzyVar* pVar, UINT updates, CString newname="");
        BOOL UpdateLinksBDR(CBDR* pBDR, UINT updates, CBDR &source);
        CPolicy* FindPolicy(CString keyword);
        CPolicy* ChangePolicyName(CString newname, CString oldname="", BOOL update=TRUE);
        void UpdateContentsAllViews(CString polyname = "");
        static CVitroFuzzDoc* GetDocumentActive();
        BOOL ActiveSpreadSheet(CBDR* p_BDR);
        CString CreateFzyVarname(int type, CPolicy* pPolicy=NULL);
        CString CreatePolicyname();
        virtual ~CVitroFuzzDoc();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif
        CPolicy *m_pPolicy;
        CBase   *m_pBase;
        UINT    m_uMODE;
        CVitroFuzzView* m_pTreeView;
        CEditorView*    m_pEditView;
        CControlView*   m_pControlView;
        CMapKeyToDMemory  m_MapDMemory;

protected:
        CMapKeyToPolicy                 m_MapPolicy;
        CString                         m_sPassword;
```

```
protected:
        void CloseSystem();
        CPolicy* CreatePolicy(BOOL active, CString Policyname = "");
        void SetActivePolicy(CString name);

        // Generated message map functions
        //{{AFX_MSG(CVitroFuzzDoc)
        afx_msg void OnArchivoWizardmodel();
        afx_msg void OnWindowOpenallEditrules();
        afx_msg void OnUpdateDebugInteractive(CCmdUI* pCmdUI);
        afx_msg void OnOpenSpreadSheet();
        afx_msg void OnDebugInteractive();
        afx_msg void OnUpdateDebugLinkserial(CCmdUI* pCmdUI);
        afx_msg void OnDebugLinkserial();
        afx_msg void OnDebugFilerecord();
        afx_msg void OnUpdateDebugFilerecord(CCmdUI* pCmdUI);
        afx_msg void OnFileListmystems();
        afx_msg void OnUpdateFileListmystems(CCmdUI* pCmdUI);
        afx_msg void OnToolsOptions();
        //}}AFX_MSG
        afx_msg void OnEditarVariable(UINT nID);
        afx_msg void OnEditarBDR(UINT nID);
        afx_msg void OnEditarText(UINT nID);
        afx_msg void OnEditMemory(UINT nID);
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_VITROFUZZDOC_H__A84E23CD_E9FC_11D3_9F48_188A01C10100__INCLUDED_)



// VitroFuzzDoc.cpp : implementation of the CVitroFuzzDoc class
//

#include "stdafx.h"
#include "VitroFuzz.h"
#include "VitroFuzzDoc.h"
#include "SheetWizard.h"
#include "SheetFuzzyVar.h"
#include "SheetBDR.h"
#include "SheetComm.h"
#include "TextEditor.h"
#include "SelectSystem.h"
#include "XSYSctl.h"
#include "XFZYctl.h"
#include "Globales.h"
#include "Fuzzy.hpp"
#include "PropMemoryDlg.h"
#include "VitroFuzzView.h"
#include "EditorView.h"
#include "ControlView.h"
#include "FlexGridDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern CXSYSctl XSYSctl;
extern CXFZYctl XFZYctl;
extern int status;

/////////////////////////////////////////////////////////////////////////////
// CVitroFuzzDoc
```

```
IMPLEMENT_DYNCREATE(CVitroFuzzDoc, CDocument)

BEGIN_MESSAGE_MAP(CVitroFuzzDoc, CDocument)
        //{{AFX_MSG_MAP(CVitroFuzzDoc)
        ON_COMMAND(ID_ARCHIVO_WIZARDMODEL, OnArchivoWizardmodel)
        ON_COMMAND(ID_WINDOW_OPENALL_EDITRULES, OnWindowOpenallEditrules)
        ON_UPDATE_COMMAND_UI(ID_DEBUG_INTERACTIVE, OnUpdateDebugInteractive)
        ON_COMMAND(ID_BDR_EDITORBDR, OnOpenSpreadSheet)
        ON_COMMAND(ID_DEBUG_INTERACTIVE, OnDebugInteractive)
        ON_UPDATE_COMMAND_UI(ID_DEBUG_LINKSERIAL, OnUpdateDebugLinkserial)
        ON_COMMAND(ID_DEBUG_LINKSERIAL, OnDebugLinkserial)
        ON_COMMAND(ID_DEBUG_FILERECORD, OnDebugFilerecord)
        ON_UPDATE_COMMAND_UI(ID_DEBUG_FILERECORD, OnUpdateDebugFilerecord)
        ON_COMMAND(ID_FILE_LISTSYSTEMS, OnFileListsystems)
        ON_UPDATE_COMMAND_UI(ID_FILE_LISTSYSTEMS, OnUpdateFileListsystems)
        ON_COMMAND(ID_TOOLS_OPTIONS, OnToolsOptions)
        //}}AFX_MSG_MAP
        ON_COMMAND_EX(ID_EDITAR_NEWVARIABLE, OnEditarVariable)
        ON_COMMAND_EX(ID_EDITAR_VARIABLE,     OnEditarVariable)
        ON_COMMAND_EX(ID_EDITAR_NEWBDR,              OnEditarBDR)
        ON_COMMAND_EX(ID_EDITAR_BDR,          OnEditarBDR)
        ON_COMMAND_EX(ID_EDITAR_NUEVOTEXTO, OnEditarText)
        ON_COMMAND_EX(ID_EDITAR_TEXT,         OnEditarText)
        ON_COMMAND_EX(ID_MEMORY_EDIT,         OnEditMemory)
        ON_COMMAND_EX(ID_MESSAGE_EDIT,              OnEditMemory)
        ON_COMMAND_EX(ID_INTERRUP_EDIT,             OnEditMemory)
        ON_COMMAND_EX(ID_ADD_DMEMORY,         OnEditMemory)
        ON_COMMAND_EX(ID_ADD_MESSAGE,         OnEditMemory)
        ON_COMMAND_EX(ID_ADD_INTERRUP,              OnEditMemory)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CVitroFuzzDoc construction/destruction

CVitroFuzzDoc::CVitroFuzzDoc()
{
        m_pTreeView = NULL;
        m_pEditView = NULL;
        m_pControlView = NULL;
        m_MapDMemory.RemoveAll();
}

CVitroFuzzDoc::~CVitroFuzzDoc()
{
        XFZYctl.FzyCloseFZYctl(&status);
}

BOOL CVitroFuzzDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        if (CreatePolicy(TRUE) == NULL)
        {
                AfxMessageBox(IDS_ERR_INITSYSTEM, MB_ICONSTOP);
                return FALSE;
        }
        m_uMODE = EDITION_MODE;
        m_pPolicy->SetAlfacut((float)0.05);
        XSYSctl.MdlConnecttoFMS(&status);
        XSYSctl.m_sReportPath = "./";
        XSYSctl.m_sReportFile = "Reporte.dat";
        XSYSctl.m_sSystemFile = "Sistema.dat";
        return TRUE;
}

////////////////////////////////////////////////////////////////////////////
// CVitroFuzzDoc serialization

void CVitroFuzzDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                ar << m_sPassword;
        }
        else
```

```
        {
                ar >> m_sPassword;
        }
        m_MapDMemory.Serialize(ar);
        m_MapPolicy.Serialize(ar);
}

/////////////////////////////////////////////////////////////////////////////
// CVitroFuzzDoc diagnostics

#ifdef _DEBUG
void CVitroFuzzDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CVitroFuzzDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CVitroFuzzDoc commands

CString CVitroFuzzDoc::CreateFzyVarname(int type, CPolicy* pPolicy)
{
        CFuzzyVar *p_FuzVar;
        CString   Name;
        char buff[5];

        if (pPolicy == NULL)
                pPolicy = m_pPolicy;
        Name = (type) ? "entrada_" : "salida_";
        for (int i=1;;i++)
                if (!(pPolicy->m_MapVAR.Lookup(Name + _itoa(i, buff, 10), p_FuzVar)))
                        return (Name + _itoa(i, buff, 10));
}

CPolicy* CVitroFuzzDoc::CreatePolicy(BOOL active, CString Policyname)
{
        CPolicy* pDPolicy = NULL;

        if (Policyname == "") Policyname = CreatePolicyname();
        if (!(pDPolicy = new CPolicy(Policyname))) return NULL;
        m_MapPolicy.SetAt(pDPolicy->m_sName, pDPolicy);
        if (active) SetActivePolicy(pDPolicy->m_sName);
        return pDPolicy;
}

CString CVitroFuzzDoc::CreatePolicyname()
{
        CPolicy   *p_Policy;
        CString   Name;
        char buff[5];

        Name = "Sistema_";
        for (int i=1;;i++)
                if (!(m_MapPolicy.Lookup(Name + _itoa(i, buff, 10), p_Policy)))
                        return (Name + _itoa(i, buff, 10));
}

void CVitroFuzzDoc::CloseSystem()
{
        CPolicy* p_DPolicy=NULL;
        CString  Key;
        POSITION pos;
        if (m_MapPolicy.IsEmpty()) return;
        pos = m_MapPolicy.GetStartPosition();
        while (pos != NULL)
        {
                m_MapPolicy.GetNextAssoc(pos, Key, p_DPolicy);
                p_DPolicy->ClosePolicy();
                delete p_DPolicy;
        }
```

```
        return;
}
void CVitroFuzzDoc::DeleteContents()
{
        CloseSystem();
        m_MapPolicy.RemoveAll();
        m_pPolicy       = NULL;
        m_pBase         = NULL;
        m_sPassword     = "";
        CDMemory *p_Mem;
        CString Key;
        POSITION pos;

        pos = m_MapDMemory.GetStartPosition();
        while (pos != NULL)
        {
                m_MapDMemory.GetNextAssoc(pos, Key, p_Mem);
                delete p_Mem;
        }
        m_MapDMemory.RemoveAll();

        if (m_pTreeView != NULL) m_pTreeView->ResetContents();
        if (m_pEditView != NULL) m_pEditView->ResetContents();
        if (m_pControlView != NULL) m_pControlView->ResetContents();
        CDocument::DeleteContents();
}

BOOL CVitroFuzzDoc::ActiveSpreadSheet(CBDR *p_BDR)
{
        CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
        CView *pView = (CView *) pFrame->GetActiveView();
        CFlexGridDlg dlg(pView);
        m_uMODE = EDITRULES_MODE;
        dlg.DoModal();
        m_uMODE = EDITION_MODE;
        return TRUE;
}

void CVitroFuzzDoc::OnArchivoWizardmodel()
{
        CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
        CView *pView = (CView *) pFrame->GetActiveView();
        CSheetWizard Wizard("Asistente de modelado", pView, 0);
        Wizard.SetWizardMode();
        CFuzzyVar datos[20];
        Wizard.m_pVar = &datos[0];

        if (Wizard.DoModal() == ID_WIZFINISH)
        {
                CBDR            BDRtmp;
                int Varcnt,     Inpcnt;
                CString         polyname;
                CFuzzyVar*      p_datos;

                polyname=  Wizard.m_PageGlobal.m_sPolicyName;
                Varcnt  =  Wizard.m_PageDefVars.m_cont;
                Inpcnt  =  Wizard.m_PageDefVars.m_nInput;
                p_datos = &datos[0];

                if (Wizard.m_PageGlobal.m_nCreate)
                {
                        if (Wizard.m_PageGlobal.m_bNewArchive)
                        {
                                OnNewDocument();
                                ChangePolicyName(polyname);
                        }
                        else
                        {
                                if (CreatePolicy(TRUE, polyname) == NULL)
                                {
                                        AfxMessageBox(IDS_ERR_INITSYSTEM, MB_ICONSTOP);
                                        return ;
                                }
                        }
```

```
                m_pPolicy->SetAlfacut(Wizard.m_PageGlobal.m_fAlfacut);
            }
            m_pPolicy->m_sComent = Wizard.m_PageComent.m_sComent;
            for (int i=0; i<=Varcnt; i++, p_datos++)
            {
                p_datos->m_sComent      = "";
                p_datos->m_dDefval      = (double)0.0;
                p_datos->m_bLimits      = HEDGETYPEUP;
                p_datos->m_bSymmetry = SIMETRICTYPE;

                CFuzzyVar* pVar = m_pPolicy->CreateFuzzyVar(p_datos);
                if (pVar == NULL)
                {
                    AfxMessageBox(IDS_ERR_VAR_NOCREATE,
                                  MB_OK | MB_ICONSTOP, 0);
                    return;
                }

                if (pVar->m_bType == TIPOVARINP)
                {
                    pVar->m_etype = t_VarInp;
                    BDRtmp.m_MapVarInp.SetAt(pVar->m_sName, pVar);
                }
                else
                {
                    pVar->m_etype = t_VarOut;
                    BDRtmp.m_MapVarOut.SetAt(pVar->m_sName, pVar);
                }
            }

            CBDR* pBDR = m_pPolicy->CreateBDR(BDRtmp);

            if (pBDR == NULL)
            {
                AfxMessageBox(IDS_ERR_BDR_NOCREATE,
                              MB_OK | MB_ICONSTOP, 0);
                return;
            }
            UpdateContentsAllViews(m_pPolicy->m_sName);
        }
}

void CVitroFuzzDoc::OnEditarVariable(UINT nID)
{
        CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
        CView *pView = (CView *)pFrame->GetActiveView();

        CSheetFuzzyVar Wizard("Asistente de Variables", pView, 0);
        CFuzzyVar fuzzyvar;
        CFuzzyVar* pVar;
        CPoint oldpoint(0,0);
        UINT    updates;
        if (nID == ID_EDITAR_NEWVARIABLE)
        {
                Wizard.SetWizardMode();
                Wizard.m_mode = 0;
                fuzzyvar.m_nSets = 3;
                fuzzyvar.SetTermNames();
                fuzzyvar.DefineFuzzySets();
        }
        else
        {
                pVar = (CFuzzyVar*)m_pBase;
                fuzzyvar.Copy(pVar);
                fuzzyvar.m_sName = pVar->m_sName;
                Wizard.m_mode = 1;
        }

        Wizard.m_pVar = &fuzzyvar;
        int ans = Wizard.DoModal();
        if (ans == IDCANCEL)
                return;

        if (nID == ID_EDITAR_VARIABLE)
        {
```

```
            updates = pVar->DetectTypeUpdate(fuzzyvar);
            if (!updates) return;
            UpdateAllViews(NULL, DELETEITEM, m_pBase);
            if (updates >= UPDATENAME)
                  UpdateLinksVar(pVar, updates, fuzzyvar.m_sName);
      }
      else
      {
            pVar = m_pPolicy->CreateFuzzyVar(&fuzzyvar);
            if (pVar == NULL)
            {
                  AfxMessageBox(IDS_ERR_VAR_NOCREATE,
                                          MB_OK | MB_ICONSTOP, 0);
                  return;
            }
      }
      CBase* pBase = pVar;
      UpdateAllViews(NULL, INSERTITEM, pBase);
      return;
}

void CVitroFuzzDoc::OnWindowOpenallEditrules()
{
}

void CVitroFuzzDoc::SetActivePolicy(CString name)
{
      m_MapPolicy.Lookup(name, m_pPolicy);
}

void CVitroFuzzDoc::OnEditarBDR(UINT nID)
{
      CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
      CView *pView = (CView *) pFrame->GetActiveView();

      CSheetBDR Wizard("Asistente Bloque de Reglas", pView, 0);
      Wizard.m_PageComent.m_ntypeWizard = WIZBDR;
      CBDR*    pBDR;
      CFuzzyVar* pVar;
      CString   keyword;
      CMapKeyToFuzzyVar& map = m_pPolicy->m_MapVAR;
      POSITION pos = map.GetStartPosition();

      if (nID == ID_EDITAR_NEWBDR)
      {
            while (pos != NULL)
            {
                  map.GetNextAssoc(pos, keyword, pVar);
                  Wizard.m_PageDefBDR.m_IOvars.Add(keyword);
            }

            Wizard.SetWizardMode();
            Wizard.m_PageDefBDR.m_sBDRName   = m_pPolicy->CreateBDRname();
            Wizard.m_PageDefBDR.m_nRunindex  = m_pPolicy->CreateRunIndex();
            Wizard.m_PageDefBDR.m_bRunEnable = TRUE;
      }
      else
      {
            pBDR = (CBDR*)m_pBase;
            while (pos != NULL)
            {
                  map.GetNextAssoc(pos, keyword, pVar);
                  if (pBDR->m_MapVarInp.Lookup(keyword, pVar))
                  {
                        Wizard.m_PageDefBDR.m_Inpvars.Add(keyword);
                        goto nextitem;
                  }
                  if (pBDR->m_MapVarOut.Lookup(keyword, pVar))
                  {
                        Wizard.m_PageDefBDR.m_Outvars.Add(keyword);
                        goto nextitem;
                  }
                  Wizard.m_PageDefBDR.m_IOvars.Add(keyword);
            nextitem: ;
            }
```

```
            Wizard.m_PageDefBDR.m_sBDRName        = pBDR->m_sName;
            Wizard.m_PageComent.m_sDesc           = pBDR->m_sName;
            Wizard.m_PageComent.m_sComent = pBDR->m_sComent;
            Wizard.m_PageDefBDR.m_nRunindex       = pBDR->m_uRunIndex;
            Wizard.m_PageDefBDR.m_bRunEnable= pBDR->m_bRun;
      }

      int ans = Wizard.DoModal();
      if (ans == IDCANCEL)
            return;
      CBDR    BDRtmp;
      CString VarName;
      UINT    updates;
      BDRtmp.m_sName  =       Wizard.m_PageDefBDR.m_sBDRName;
      BDRtmp.m_sComent =      Wizard.m_PageComent.m_sComent;
      BDRtmp.m_uRunIndex = Wizard.m_PageDefBDR.m_nRunindex;
      BDRtmp.m_bRun =           Wizard.m_PageDefBDR.m_bRunEnable;
      BDRtmp.m_MapVarInp.Copy(Wizard.m_PageDefBDR.m_Inpvars);
      BDRtmp.m_MapVarOut.Copy(Wizard.m_PageDefBDR.m_Outvars);
      if (nID == ID_EDITAR_BDR)
      {
            updates = pBDR->DetectTypeUpdate(BDRtmp);
            if (!updates) return;
            UpdateAllViews(NULL, DELETEITEM, m_pBase);
            UpdateLinksBDR(pBDR, updates, BDRtmp);
      }
      else
      {
            pBDR = m_pPolicy->CreateBDR(BDRtmp);
            if (pBDR == NULL)
            {
                  AfxMessageBox(IDS_ERR_BDR_NOCREATE,
                                    MB_OK | MB_ICONSTOP, 0);
                  return;
            }
      }
      CBase* pBase = pBDR;
      UpdateAllViews(NULL, INSERTITEM, pBase);
}

void CVitroFuzzDoc::OnEditarText(UINT nID)
{
      CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
      CView *pView = (CView *) pFrame->GetActiveView();

      CTextEditor dlg(pView);

      CText* pText;

      if (nID == ID_EDITAR_NUEVOTEXTO)
      {
            pText = new CText;
            if (pText == NULL)
            {
                  AfxMessageBox(IDS_ERR_TEXT_NOCREATE,
                                    MB_OK | MB_ICONSTOP, 0);
                  return;
            }
      }
      else
            pText = (CText*)m_pBase;

    dlg.m_sTexto          = pText->m_sName;
    dlg.m_colorTextItem = pText->m_color;
    dlg.m_logfontItem   = pText->m_logfont;

    if (dlg.DoModal() == IDOK)
    {
            pText->m_color    = dlg.m_colorTextItem;
            pText->m_logfont = dlg.m_logfontItem;
            pText->m_sName  = dlg.m_sTexto;

            if (nID == ID_EDITAR_NUEVOTEXTO)
            {
```

```
                        m_pPolicy->m_aText.Add(pText);
                        UpdateAllViews(NULL, INSERTITEM, pText);
                }
                else
                        UpdateAllViews(NULL, UPDATEITEM, pText);
        }
        else
                if (nID == ID_EDITAR_NUEVOTEXTO)
                        delete pText;
}


void CVitroFuzzDoc::OnOpenSpreadSheet()
{
        if (m_pBase->IsKindOf(RUNTIME_CLASS(CBDR)))
                ActiveSpreadSheet((CBDR*)m_pBase);
}

void CVitroFuzzDoc::OnUpdateDebugInteractive(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_pPolicy->m_MapBDR.IsEmpty());
}

void CVitroFuzzDoc::OnDebugInteractive()
{
        m_pControlView->PostMessage(WM_CREATE_EVL, (WPARAM)m_pPolicy,
                                                        DEBUG_INTERACTIVE_MODE);
}

void CVitroFuzzDoc::OnDebugLinkserial()
{
        m_pControlView->PostMessage(WM_CREATE_EVL, (WPARAM)m_pPolicy,
                                                        DEBUG_SERIALCOM_MODE);
}

void CVitroFuzzDoc::OnUpdateDebugLinkserial(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_pPolicy->m_MapBDR.IsEmpty());
}

void CVitroFuzzDoc::OnDebugFilerecord()
{
        m_pControlView->PostMessage(WM_CREATE_EVL, (WPARAM)m_pPolicy,
                                                        DEBUG_FILEINPUT_MODE);
}

void CVitroFuzzDoc::OnUpdateDebugFilerecord(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_pPolicy->m_MapBDR.IsEmpty());
}

CVitroFuzzDoc* CVitroFuzzDoc::GetDocumentActive()
{
        CMDIChildWnd* pChild =
                ((CMDIFrameWnd*)(AfxGetApp()->m_pMainWnd))->MDIGetActive();

        if (!pChild)
                return NULL;
        CDocument* pDoc = pChild->GetActiveDocument();
        if (!pDoc)
                return NULL;
        if (! pDoc->IsKindOf(RUNTIME_CLASS(CVitroFuzzDoc)))
                return NULL;
        return (CVitroFuzzDoc*)pDoc;
}

void CVitroFuzzDoc::UpdateContentsAllViews(CString polyname)
{
        CBase*     pBase;
        CString    Key;
        POSITION   pos;

        if (m_MapPolicy.IsEmpty())
        {
                return;
```

```
        }

        m_pPolicy = NULL;

        if (polyname != "")
                m_MapPolicy.Lookup(polyname, m_pPolicy);

        if (m_pPolicy == NULL)
        {
                pos = m_MapPolicy.GetStartPosition();
                m_MapPolicy.GetNextAssoc(pos, Key, m_pPolicy);
        }
        m_pTreeView->ResetContents();
        m_pEditView->ResetContents();
        m_pControlView->ResetContents();
        m_pTreeView->ChangePolicyName();
        pos = m_pPolicy->m_MapVAR.GetStartPosition();
        while (pos != NULL)
        {
                m_pPolicy->m_MapVAR.GetNextAssoc(pos, Key, (CFuzzyVar*&)pBase);
                UpdateAllViews(NULL, INSERTITEM, pBase);
        }

        pos = m_pPolicy->m_MapBDR.GetStartPosition();
        while (pos != NULL)
        {
                m_pPolicy->m_MapBDR.GetNextAssoc(pos, Key, (CBDR*&)pBase);
                UpdateAllViews(NULL, INSERTITEM, pBase);
        }

        for (int i=0; i < m_pPolicy->m_aText.GetSize(); i++)
        {
                pBase = (CBase*)m_pPolicy->m_aText.GetAt(i);
                UpdateAllViews(NULL, INSERTITEM, pBase);
        }

        pos = m_MapDMemory.GetStartPosition();
        while (pos != NULL)
        {
                m_MapDMemory.GetNextAssoc(pos, Key, (CDMemory*&)pBase);
                UpdateAllViews(NULL, INSERTITEM, pBase);
        }
}

CPolicy* CVitroFuzzDoc::ChangePolicyName(CString newname, CString oldname, BOOL update)
{
        if (m_pTreeView == NULL || newname.IsEmpty())
                return NULL;

        CPolicy* pPolicy=NULL;

        if (oldname.IsEmpty())
                oldname = m_pPolicy->m_sName;
        m_MapPolicy.Lookup(oldname, pPolicy);
        m_MapPolicy.RemoveKey(oldname);
        pPolicy->m_sName = newname;
        m_MapPolicy.SetAt(pPolicy->m_sName, pPolicy);
        if (update)
                m_pTreeView->ChangePolicyName();
        return pPolicy;
}

void CVitroFuzzDoc::OnFileListsystems()
{
        CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
        CView *pView = (CView *) pFrame->GetActiveView();
        CString oldname = m_pPolicy->m_sName;
        CString polyname;
        CSelectSystem dlg(pView);
        dlg.m_pMapPolicy = &m_MapPolicy;
        if (dlg.DoModal() == IDOK || dlg.m_bModify)
        {
                polyname = dlg.m_sPolyname;
                if (polyname != oldname)
                        UpdateContentsAllViews(polyname);
```

```
        }
}

void CVitroFuzzDoc::OnUpdateFileListsystems(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_MapPolicy.IsEmpty());
}

CPolicy* CVitroFuzzDoc::FindPolicy(CString keyword)
{
        CPolicy* pPolicy = NULL;
        m_MapPolicy.Lookup(keyword, pPolicy);
        return pPolicy;
}
BOOL CVitroFuzzDoc::UpdateLinksVar(CFuzzyVar* pVar, UINT updates, CString newname)
{
        CBDR*       pBDR;
        CString     bdrname,
                        vector,
                        newvector,
                        newterm,
                        mode;
        int             index,
                        nMaxSets,
                        j;
        POSITION    pos;

        pos = pVar->m_MapAttach.GetStartPosition();
        while (pos)
        {
                pVar->m_MapAttach.GetNextAssoc(pos, bdrname, mode);
                m_pPolicy->m_MapBDR.Lookup(bdrname, pBDR);
                if (updates & UPDATENAME)
                {
                        if (mode == "MODEIN")
                                pBDR->m_MapVarInp.RemoveKey(pVar->m_sName);
                        else
                                pBDR->m_MapVarOut.RemoveKey(pVar->m_sName);
                        if (mode == "MODEIN")
                                pBDR->m_MapVarInp.SetAt(newname, pVar);
                        else
                                pBDR->m_MapVarOut.SetAt(newname, pVar);
                        m_pTreeView->ChangeItemName(pBDR->m_hItem, pVar->m_sName,
        newname, TRUE);
                }
                pBDR->m_MapFAM.Lookup(pVar->m_sName, vector);
                if (updates & UPDATESETS)
                {
                        newvector = "";
                        nMaxSets  = pVar->m_nSets;
                        index = atoi(GetNextTerm(vector));
                        for (j=0; j < pBDR->m_nRules; j++)
                        {
                                index = atoi(GetNextTerm(vector)) - 1;
                                if (index > nMaxSets) index = nMaxSets;
                                index++;
                                newterm.Format("%u,", index);
                                newvector += newterm;
                        }
                        vector.Format("%u,", nMaxSets);
                        vector += newvector;
                }
                pBDR->m_MapFAM.RemoveKey(pVar->m_sName);
                pBDR->m_MapFAM.SetAt(newname, vector);
        }
        if (updates & UPDATENAME)
        {
                m_pPolicy->m_MapVAR.RemoveKey(pVar->m_sName);
                m_pPolicy->m_MapVAR.SetAt(newname, pVar);
                pVar->m_sName = newname;
        }

        return TRUE;
}
```

```
BOOL CVitroFuzzDoc::UpdateLinksBDR(CBDR* pBDR, UINT updates, CBDR &source)
{
	POSITION pos;
	CString varname, mode, none, vector, sets;
	CMapStringToString map_vars;
	CFuzzyVar *pVar1, *pVar2;
	BOOL oldbdr, newbdr, oldmaptype, newmaptype;
	int j;

	if (updates & UPDATEINPS || updates & UPDATEOUTS)
	{
		map_vars.RemoveAll();
		pos = pBDR->m_MapVarInp.GetStartPosition();
		while (pos)
		{
			pBDR->m_MapVarInp.GetNextAssoc(pos, varname, pVar2);
			map_vars.SetAt(varname, none);
		}
		pos = pBDR->m_MapVarOut.GetStartPosition();
		while (pos)
		{
			pBDR->m_MapVarOut.GetNextAssoc(pos, varname, pVar2);
			map_vars.SetAt(varname, none);
		}
		pos = source.m_MapVarInp.GetStartPosition();
		while (pos)
		{
			source.m_MapVarInp.GetNextAssoc(pos, varname, pVar2);
			map_vars.SetAt(varname, none);
		}
		pos = source.m_MapVarOut.GetStartPosition();
		while (pos)
		{
			source.m_MapVarOut.GetNextAssoc(pos, varname, pVar2);
			map_vars.SetAt(varname, none);
		}
		pos = map_vars.GetStartPosition();
		while (pos)
		{
			map_vars.GetNextAssoc(pos, varname, none);
			if ((newbdr = source.m_MapVarInp.Lookup(varname, pVar2)))
				newmaptype = FALSE;
			else if ((newbdr = source.m_MapVarOut.Lookup(varname, pVar2)))
				newmaptype = TRUE;
			if ((oldbdr = pBDR->m_MapVarInp.Lookup(varname, pVar1)))
				oldmaptype = FALSE;
			else if ((oldbdr = pBDR->m_MapVarOut.Lookup(varname, pVar1)))
				oldmaptype = TRUE;
			if (newbdr)
			{
				if (oldbdr)
				{
					if (newmaptype != oldmaptype)
					{
						if (oldmaptype)
						{
							pBDR->m_MapVarOut.RemoveKey(varname);
							pBDR->m_MapVarInp.SetAt(varname, pVar1);
							pVar1->AttachBDR(pBDR->m_sName, "MODEIN");
						}
						else
						{
							pBDR->m_MapVarInp.RemoveKey(varname);
							pBDR->m_MapVarOut.SetAt(varname, pVar1);
							pVar1->AttachBDR(pBDR->m_sName, "MODEOUT");
						}

					}
				}
				else
				{
					m_pPolicy->m_MapVAR.Lookup(varname, pVar1);
					if (newmaptype)
					{
```

```
                                        pBDR->m_MapVarOut.SetAt(varname, pVar1);
                                        pVar1->AttachBDR(pBDR->m_sName, "MODEOUT");
                                }
                                else
                                {
                                        pBDR->m_MapVarInp.SetAt(varname, pVar1);
                                        pVar1->AttachBDR(pBDR->m_sName, "MODEIN");
                                }
                                vector = "";
                                for (j=0; j < pBDR->m_nRules; j++)
                                        vector += "1,";
                                sets.Format("%u,", pVar1->m_nSets);
                                vector = sets + vector;
                                pBDR->m_MapFAM.SetAt(varname, vector);
                        }
                }
                else
                {
                        if (oldmaptype)
                                pBDR->m_MapVarOut.RemoveKey(varname);
                        else
                                pBDR->m_MapVarInp.RemoveKey(varname);
                        pBDR->m_MapFAM.RemoveKey(varname);
                        pVar1->DetachBDR(pBDR->m_sName);
                }
        }
    }
    if (updates & UPDATENAME)
    {
        for (j=0; j < 2; j++)
        {
                if (!j)
                        pos = pBDR->m_MapVarInp.GetStartPosition();
                else
                        pos = pBDR->m_MapVarOut.GetStartPosition();
                while (pos)
                {
                        if (!j)
                                pBDR->m_MapVarInp.GetNextAssoc(pos, varname, pVar1);
                        else
                                pBDR->m_MapVarOut.GetNextAssoc(pos, varname, pVar1);
                        pVar1->m_MapAttach.Lookup(pBDR->m_sName, mode);
                        pVar1->DetachBDR(pBDR->m_sName);
                        pVar1->AttachBDR(source.m_sName, mode);
                }
        }
        m_pPolicy->m_MapBDR.RemoveKey(pBDR->m_sName);
        m_pPolicy->m_MapBDR.SetAt(source.m_sName, pBDR);
        pBDR->m_sName = source.m_sName;
    }

    return TRUE;
}
void CVitroFuzzDoc::OnToolsOptions()
{
    CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
    CView *pView = (CView *) pFrame->GetActiveView();
    CSheetComm Wizard("Opciones", pView, 0);
    if (Wizard.DoModal() != IDOK) return;
}

BOOL CVitroFuzzDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
    if (!CDocument::OnOpenDocument(lpszPathName))
        return FALSE;
    CString key;
    POSITION pos = m_MapPolicy.GetStartPosition();
    if (pos)
    {
        m_MapPolicy.GetNextAssoc(pos, key, m_pPolicy);
        UpdateContentsAllViews();
    }
    return TRUE;
}
```

```
void CVitroFuzzDoc::OnEditMemory(UINT nID)
{
	CMDIFrameWnd *pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
	CView *pView = (CView *) pFrame->GetActiveView();

	CPropMemoryDlg dlg(pView);
	CDMemory* p_Mem;
	UINT action;
	UINT mode;

	if (nID == ID_ADD_DMEMORY ||
		nID == ID_ADD_MESSAGE ||
		nID == ID_ADD_INTERRUP)
	{
		p_Mem = new CDMemory;
		if (p_Mem == NULL)
		{
			AfxMessageBox(IDS_ERR_MEMORY_NOCREATE,
					MB_OK | MB_ICONSTOP, 0);
			return;
		}
		switch (nID)
		{
		case ID_ADD_DMEMORY:  mode = NORMALTYPEMEM;   break;
		case ID_ADD_MESSAGE:  mode = MESSAGETYPEMEM;  break;
		case ID_ADD_INTERRUP: mode = INTERRUPTYPEMEM; break;
		}
		p_Mem->m_uMode = mode;
		p_Mem->m_sName = CreateName(t_Memory, mode);
		p_Mem->m_byID  = CreateID(t_Memory, p_Mem->m_uPortComm);
	}
	else
		p_Mem = (CDMemory *)m_pBase;

	dlg.m_pMem = p_Mem;

	if (dlg.DoModal() == IDOK)
	{
		if (nID == ID_ADD_DMEMORY ||
			nID == ID_ADD_MESSAGE ||
			nID == ID_ADD_INTERRUP)
				action = INSERTITEM;
		else
		{
			action = UPDATEITEM;
			if (dlg.oldName != p_Mem->m_sName)
				m_MapDMemory.RemoveKey(dlg.oldName);
		}
		m_MapDMemory.SetAt(p_Mem->m_sName, p_Mem);
		UpdateAllViews(NULL, action, p_Mem);
	}
	else
		if (nID == ID_ADD_DMEMORY ||
			nID == ID_ADD_MESSAGE ||
			nID == ID_ADD_INTERRUP)
				delete p_Mem;
}

CString CVitroFuzzDoc::CreateName(etypeobj type, int mode)
{
	CDMemory *p_Mem;
	CString  Name;
	char buff[5];
	if (type == t_Memory)
	{
		switch (mode)
		{
		case NORMALTYPEMEM:   Name = "Memoria_";  break;
		case INTERRUPTYPEMEM: Name = "Interrup_"; break;
		case MESSAGETYPEMEM:  Name = "Mensaje_";  break;
		}
		for (int i=1;;i++)
			if (!(m_MapDMemory.Lookup(Name + _itoa(i, buff, 10), p_Mem)))
				return (Name + _itoa(i, buff, 10));
	}
```

```
        return Name;
}

BYTE CVitroFuzzDoc::CreateID(etypeobj type, UINT numPort)
{
        CDMemory *p_Mem;
        CString  name;
        POSITION pos;
        BYTE     ID = 0;
        BOOL     flag=TRUE;

        while (flag)
        {
                flag = FALSE;
                ID++;
                if (type == t_Memory)
                {
                        pos = m_MapDMemory.GetStartPosition();

                        while (pos)
                        {
                                m_MapDMemory.GetNextAssoc(pos, name, p_Mem);
                                if (ID == p_Mem->m_byID &&
                                        p_Mem->m_uPortComm == numPort)
                                                flag = TRUE;
                        }
                }
        }
        return ID;
}

BOOL CVitroFuzzDoc::CheckID(CBaseCom *source, BYTE ID, UINT numPort)
{
        POSITION pos;
        CDMemory *p_Mem;
        CString  name;
        BOOL flag = TRUE;

        if (source->m_etype == t_Memory)
        {
                pos = m_MapDMemory.GetStartPosition();
                while (pos)
                {
                        m_MapDMemory.GetNextAssoc(pos, name, p_Mem);

                        if (source != p_Mem &&
                                ID == p_Mem->m_byID &&
                                numPort == p_Mem->m_uPortComm)
                                        flag = FALSE;
                }
        }
        return flag;
}
```