UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA
DIVISION DE ESTUDIOS DE POSGRADO

# DISEÑO DE UN SISTEMA DE MONITOREO Y CONTROL DISTRIBUIDO EN TIEMPO REAL.

TESIS QUE PARA OBTENER EL TITULO DE:
MAESTRO EN INGENIERIA (ELECTRICA)

PRESENTA:
ALVAREZ CASTILLO, JESUS



1997

ESTE TRABAJO PRETENDE SER UN EJEMPLO DE CONSTANCIA, DEDICACION, ORGANIZACION Y TRABAJO EN EQUIPO.

MI MAS PROFUNDO AGRADECIMIENTO A LAS PERSONAS QUE APOYARON ESPIRITUAL Y MATERIALMENTE LA CONCLUSION DE ESTA TESIS.

## RESUMEN

CONTROLADORES LÓGICOS PROGRAMABLES (PLCs) y una ESTACION de CONTROL y MONITOREO (MCS), conformando una arquitectura especifíca, se usan cuando es necesario monitorear y controlar diferentes procesos. En dicho arreglo hay interconectividad entre PLCs, y estos con la ESTACIÓN DE MONITOREO y CONTROl, representando los procesos a controlar mediante un modelo mátematico, algoritmo difuso o alguna otra estrategia. Esta tesis presenta un SISTEMA DE MONITOREO Y CONTROL DISTRIBUIDO EN TIEMPO REAL (SMCDTR), construido con controladores locales, construyendo éstos mediante arquitecturas paralelo con microcontroladores de diferentes familias, utilizando algoritmos difusos para controlar los procesos y usando como estaciÓn de Monitoreo y Control, una Computadora Personal (PC), con un Sistema Operativo en Tiempo Real. Todo ello con la finalidad de conectar diferentes controladores, capaces de sustituir a los (PLCs), y/Ó a la ESTACION DE CONTROL Y MONITOREO (MCS). El SISTEMA DE MONITOREO Y CONTROL DISTRIBUIDO EN TIEMPO REAL (SMCDTR), incrementa la rapidez, disminuye costo, prescinde del modelo matemático de los procesos a controlar, es adaptable a casi cualquier tipo de proceso y estandariza la comunicaciÓn entre los controladores locales, constituidos por diferentes microcontroladores.



## ABSTRACT

PROGRAMMABLE LOGIC CONTROLLERS (PLCs), and a MONITORING and CONTROLLING STATION (MCS) are used when is necessary to monitor and control several processes, establishing a specific architecture. With this arrengement there is interconnectivity between PLCs, and of these with the MONITORING and CONTROLLING STATION (MCS), performing the processes to control through a mathematical model, fuzzy algorithm or other strategy. This thesis shows a REAL TIME DISTRIBUITED CONTROL and MONITORING SYSTEM (RTDCMS), builded with local controllers, made with paralell architectures with microcontrollers from different families, using fuzzy algorithms to control the processes and using as MONITORING and CONTROLLING STATION (MCS), a PERSONAL COMPUTER (PC), with a Operative System in Real Time, with objetive of connecting different controllers, capable to substitute the (PLCs) and/or the MONITORING and CONTROLLING STATION (MCS). The REAL TIME DISTRIBUITED CONTROL and MONITORING SYSTEM (RTDCMS), increases speed and decreases the cost, regardless of the mathematical model of the processes to control, is adaptable almost to any type of process and standardizes the communication between local controllers, builded with different microcontrollers.

# DISEÑO DE UN SISTEMA DE MONITOREO Y CONTROL DISTRIBUIDO EN TIEMPO REAL

ÍNDICE

# 1.- INTRODUCCIÓN

Ante la necesidad de monitorear procesos que se llevan a cabo en diferentes lugares, es adecuado tener un puesto central donde se puedan monitorear y controlar los mismos, y dependiendo de los requerimientos de la aplicación se puede tener un controlador para cada uno de estos procesos. Precisamente es la finalidad de un control distribuido en tiempo real, objetivo de esta tesis.

Mediante el análisis de conceptos como tiempo real, sistemas distribuidos, paralelismo y lógica difusa tendremos un marco referencial del monitoreo y control de procesos. Ya que el objetivo de esta tesis es implementar un control distribuido en tiempo real estructurado mediante controladores difusos constituidos mediante arquitecturas paralelo. Cada uno de ellos, teniendo la posibilidad de conectarse mediante un bus hacia un comando central, el cual puede monitorear, registrar informaciones, envió de señales hacia puntos distantes, y controlar diferentes procesos en tiempo real, ya que este concepto es sumamente importante para algunas aplicaciones especificas. En el presente trabajo se manejan diferentes áreas del conocimiento, tales como la electrónica, el control y el cálculo distribuido, en las cuales se manejan dispositivos semiconductores, sistemas digitales, interfaces etc..Se puede decir que dichos procesos aunque son diferentes en su naturaleza, pero que en un determinado momento, pueden tener comunicación entre si, haciendo versátil el uso de este tipo de implementación. Como proceso principal consideramos un sistema de seguridad automático de locales, el cual involucra control de velocidad, posición, control de equipo de video, sensores ácusticos, ópticos, de temperatura y de humedad entre otros. El sistema deberá ser transportable a otras áreas de aplicación como el control y monitoreo de calderas, abriendo y cerrando válvulas para mantener temperatura y presión en un rango determinado. Otro de los procesos puede ser el monitoreo y registro de información de pacientes en hospitales etc. La idea fundamental de dicho sistema es que sea dentro de los límites permisibles adaptable a cualquier tipo de proceso, de tal forma que pueda usarse en la industria, hospitales, centros comerciales, oficinas, es decir que tenga el mayor grado de aplicabilidad posible. El presente trabajo estará constituido por 4 controladores difusos, los cuales estarán formados por arquitecuras pipeline y paralelo constituidos por microprocesadores y microcontroladores de diferentes familias con la finalidad de homogenizar su acoplamiento entre ellos para hacer más fácil su grado de adaptabilidad a sistemas ya elaborados.

La tesis contiene 7 capítulos, donde el capítulo 2 trata sobre las generalidades de los sistemas en tiempo real y los sistemas distribuidos dando un marco referencial que permita hacer algunas elecciones para implementar el tipo de sistema usado en este trabajo. El capítulo 3 tratará de la programación de los sistemas de tiempo real permitiendo visualizar la estructura de programación utilizada en los programas utilizados en el sistema. El capítulo 4 tratará sobre los tipos de

comunicación utilizados en este tipo de sistemas, dando un panorama para la sincronización y comunicación entre el puesto central y los controladores externos del sistema a realizar, así como los tipos de arquitecturas posibles a usar. El capítulo 5 trata sobre las estructuras de los algoritmos de control utilizados en los controladores externos. El capítulo 6 trata de la aplicación de todos los conceptos anteriores al diseño del sistema de este trabajo. El capítulo 7 trata sobre las pruebas y resultados del sistema, finalizando con las conclusiones y las perspectivas para este tipo de sistemas.

De acuerdo a esto el desarrollo tendrá características que permitan tener un grado de aplicabilidad a diferentes áreas donde se aplique un control distribuido, como lo son procesos industriales en una procesadora de alimentos, ó bien en una central termoélectrica, controladores de vuelo interactuando con computadores de vuelo, así como en sistemas de seguridad industrial y hospitales etc, etc. Ya que suscontroladores reúnen características de compatibilidad con sistemas ya establecidos, basados en microprocesadores ó bien moicrocontroladores de familias similares, gracias a la estandarización de sus comunicaciones.

Por otra parte, los controladores al ser compatibles con equipos existentes, se tiene la posibilidad de escalamiento de bajo costo sin tener que recurrir a equipos de características específicas , además de que dichos controladores tienen implementados algoritmos difusos con los cuales no necesitamos el modelo matemático del comportamiento del sistema dinámico a controlar ya que solo basta conocer las diferentes posibilidades de comportamiento de esto, lo cual permite ahorrar tiempo de cálculo y aprovechar los tiempos de interproceso e intraproceso al máximo posible.

Las perspectivas que se tienen con la utilización de este tipo de controladores es alentadora, ya que se implementan con microcontroladores que tienen soporte de software y herramientas de desarrollo, de tal forma que vienen siendo compatibles entre familias hasta cierto punto.

# 2.- GENERALIDADES DE LOS SISTEMAS DE TIEMPO REAL Y DISTRIBUIDOS

## 2.1.- GENERALIDADES DE LOS SISTEMAS DE TIEMPO REAL

### 2.1.1.- Introducción

Este capítulo esta encaminado en conocer las características tanto de los sistemas de tiempo real como de los sistemas distribuidos, lo cual permitirá formar un marco referencia para establecer los lineamientos que tendremos que tomar en cuenta en la elaboración del proyecto. De esta forma cumpliendo dicho proyecto con los párametros establecidos que tiene un sistema de esta naturaleza.

### 2.1.2.- Concepto y Características de Tiempo Real

Un sistema en tiempo real esta definido, como aquel que realiza funciones y responde a eventos asíncronos en una cantidad de tiempo predecible. Otro punto de vista sobre tiempo real se refiere a que cualquier información procesada responderá a una entrada externa en una cantidad de tiempo finita y especifica . Consecuentemente el tiempo real depende no solo de los resultados de cálculo, si no también del tiempo al cual los resultados son producidos. Dentro de las características generales podemos resumir que los sistemas de tiempo real [1]:

-Pueden reconocer y responder a eventos discretos con intervalos de tiempo predecibles.

-Son capaces de procesar y almacenar grandes cantidades de datos.

La respuesta del sistema dependerá, evidentemente de la aplicación en cuestión. Dependiendo de dicha aplicación, en términos generales puede variar desde 1 seg. hasta 100 nseg. Ejemplos de algunas aplicaciones dentro de dicho rango son; sistemas de voz y audio, robótica, control de procesos, y telemetría [1]:

## 2.2.- CLASIFICACIÓN DE SISTEMAS DE TIEMPO REAL

La clasificación de los Sistemas de Tiempo Real (fig. 2.1) está sujeta a las propiedades que poseen, como por ejemplo [1]:

a) Tiempo de Respuesta: En lo que respecta al tiempo de respuesta, se entiende como el tiempo requerido por realizar determinado proceso, y a su vez se subdividen en:

a.1) HRT (Hard Real Time): Es el tiempo especifico en que el sistema requiere para producir resultados.

a.2) SRT (Soft Real Time): Es el tiempo donde el nivel de urgencia de tareas es
especificado en forma de prioridades, la principal diferencia entre HRT y SRT radica en la forma de planear sus tareas.

b) Grupo (Interacción de hardware y software): Los sistemas desde este punto de vista están divididos en:

b.1) Los Sistemas Propietarios dependen del sistema operativo propio, de la arquitectura de hardware y del conjunto de instrucciones.

b.2) Los Sistemas Abiertos se basan en arquitecturas de hardware, desarrollo de software, sistemas operativos, además de comunicación e interfaces de los buses.

c) Arquitectura: Desde este punto de vista los sistemas están divididos en :

c.1) Sistemas Centralizados: Es aquel donde el procesador es localizado en un simple nodo del sistema y el tiempo de comunicación de interproceso es insignificante comparado con la ejecución del tiempo del procesador.

c.2) Sistemas Distribuidos: Dicho sistema esta constituido por diferentes procesadores colocados en diferentes puntos del sistema.

d) Comunicación: Desde este punto de vista se puede mencionar que los sistemas están divididos en:

d.1) Sistema de múltiple acceso: En dicho sistema se asumen diferentes computadoras, desde diferentes fuentes. Cada computadora tiene su propio sistema operativo, estructura de comando y sus aplicaciones, ya que de otro modo se dificultaría tener una terminal común para lograr una comunicación hacia el exterior del sistema, ó comunicación entre las computadoras.

d.2) Redes empaquetadas de switcheo: Este tipo de red se utiliza para manejar datos locales y para expandir la capacidad de la red utilizando las fuentes existentes. Para este caso se utiliza una computadora como procesador de comunicaciones, de tal forma que manejará una alta velocidad en las comunicaciones.

d.3) Procesamiento de transacción en línea: Se utiliza en transacciones programadas. Tipicamente mainframe y computadoras tolerantes a fallas.

## 2.3.- APLICACIONES DE SISTEMAS EN TIEMPO REAL

Primeramente mencionaremos algunos de los principales parámetros que estructuran un sistema en tiempo real, los cuales son [1]:

-Requerimiento de Tiempo de Respuesta
-Diseño y Complejidad de Procesamiento
-Funcionalidad
-Número de Tareas Manejadas
-Capacidad de Red

A continuación se mencionarán aplicaciones desde el punto de vista de **Tiempo de Respuesta**, ya que los otros parámetros son específicos de cada sistema [2]:

- Control de Procesos
- Adquisición de Datos
- Procesamiento de Imágenes
- Simulación y Control de Tráfico Aéreo
- Pruebas y Control de Máquinas
- Celdas de Control y Monitoreo
- Comunicación de Datos

Los principales campos donde se aplica este concepto son: industria metalmecánica, industria petroquímica, en equipos de control de alimentos y fármacos, aplicaciones de control de electricidad, sistemas de control y monitoreo de variables del medio ambiente, entre otros.

La clasificación de sistemas de tiempo real se muestra en la figura 2.1.

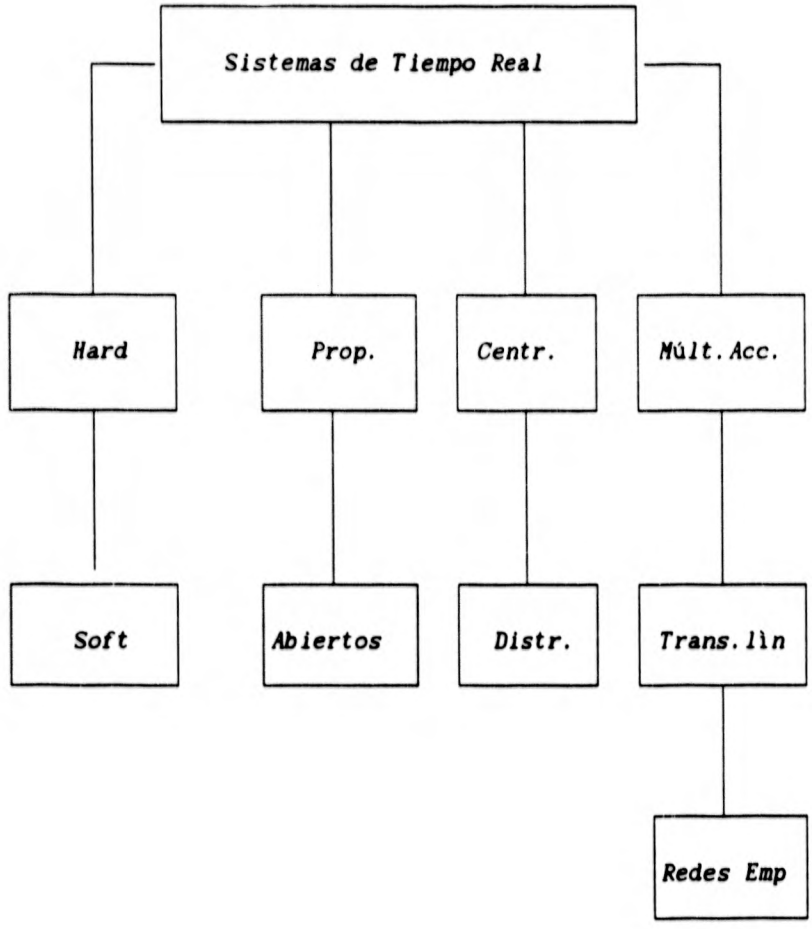



*Fig. 2.1 Clasificación de los Sistemas de Tiempo Real*

## 2.4- CONSIDERACIONES EN EL DESARROLLO Y DISEÑO DE SISTEMAS DE TIEMPO REAL

Con referencia a este subtema se pueden tener básicamente 4 aspectos generales, los cuales contemplan los principales parámetros en el diseño de un sistema en tiempo real, estos son [1]:

a) Especificación, Análisis y Verificación de Sistemas en Tiempo Real: Básicamente en este contexto se tratan metodologías para aplicaciones en Tiempo Real, así como el software desarrollado para tal aplicación.

b) Sistemas Operativos de Tiempo Real: En este punto se contempla la planeación del tiempo real, es decir se refiere a un algoritmo que debe tomar tareas las cuales ejecutará en secuencia ordenada, teniendo en cuenta características básicas para tener procesos eficientes de comunicación y sincronización, garantizando una respuesta de interrupción, rápida y confiable de un administrador de memoria adecuado . Dicho algoritmo puede estar estructurado de dos formas:

b.1) Cohesión: En este tipo de estructura se tiene módulos. Es decir estructuras de programas, donde dichos módulos actuán juntos, en otras palabras existe una interacción entre ellos desde el punto de vista temporal, de procedimiento, de comunicaciones, funcional y lógico.

b.2) Acoplamiento: Este tipo de estructura considera una interdependencia entre módulos.

Por otra parte en la implementación se deberá tener en cuenta el tipo de lenguaje ya sea un lenguaje ensamblador, lenguaje secuencial, ó lenguaje de alto nivel, teniendo en cuenta que el lenguaje ensamblador esta destinado para aplicaciones especificas, en contraste con el lenguaje de alto nivel. Independientemente del lenguaje, este deberá de tener las características de: seguridad, legibilidad, flexibilidad, simplicidad y eficiencia. Por lo que respecta a la memoria, esta será particionada cuando se maneja mas de un proceso, también lo será en tanto se active una bandera para esperar un bus [2].

c) Arquitectura de Tiempo Real: Desde este punto de vista se tiene en consideración lo siguiente:

-Alta velocidad de cálculos
-Alta velocidad en manejo de interrupciones
-Alta densidad de entrada

Como un ejemplo de la arquitectura en tiempo real se pueden tener sistemas que constan de unidades funcionales como las siguientes[2]:

-Procesador de instrucciones
-Controlador de mapa de memoria caché
-Controlador de datos de memoria caché
-Secuenciadores de Entrada/Salida
-Monitoreo de funciones como operador de mantenimiento
-Memoria Expandible
-Acelerador de punto flotante
-CPU que consta de 32 bits, decodificador de 64 bits, instrucciones tipo pipeline.

d) Comunicaciones en Tiempo Real: Desde este punto de vista se tienen que constituir protocólos de transmisión y recepción lo más eficiente posible para hacer de igual calidad el proceso de comunicaciones. De tal forma que en dicho proceso se puede tener una comunicación ya sea asíncrona ó síncrona, entendiendosé que en la síncrona, se espera a transmitir y/ó recibir cuando hay mensaje, asignando cierta prioridad a dicho mensaje, de lo contrario la comunicación será asíncrona.

## 2.5.- CRITERIO PARA REALIZAR MEDICIONES EN TIEMPO REAL

Las principales mediciones que se tienen en un proceso de Tiempo Real son [1]:

- Millones de instrucciones por segundo (MIPS)
- Millones de operaciones de punto flotante por segundo (MFLOPS)

Se pueden tener 3 diferentes formas entre otras de medir los dos párametros anteriores de acuerdo a lo siguiente [1]:

a) Mediciones de Rhealstone: Dentro de este concepto se toma en cuenta lo siguiente:

a.1) Tiempo de Conmutación de Tareas: Es definido como la razón de tiempo que el sistema toma para conmutar entre dos tareas activas e independientes de igual prioridad, como se ilustra en la figura 2.2, básicamente esto depende de la eficiencia del control en la toma de datos y su almacenamiento así como de la arquitectura del CPU y el conjunto de instrucciones.

a.2) Predicción de tiempo: Es el tiempo promedio que toma para transferir el control desde una baja prioridad a una alta prioridad como lo muestra la figura 2.2.

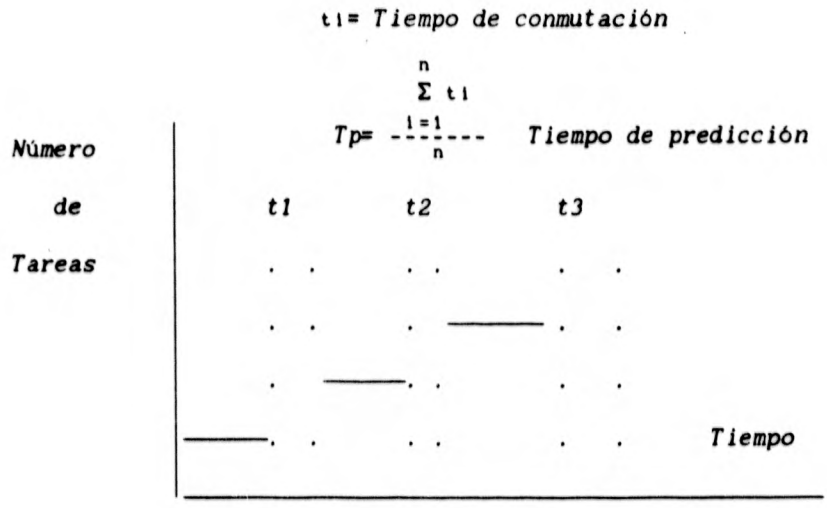
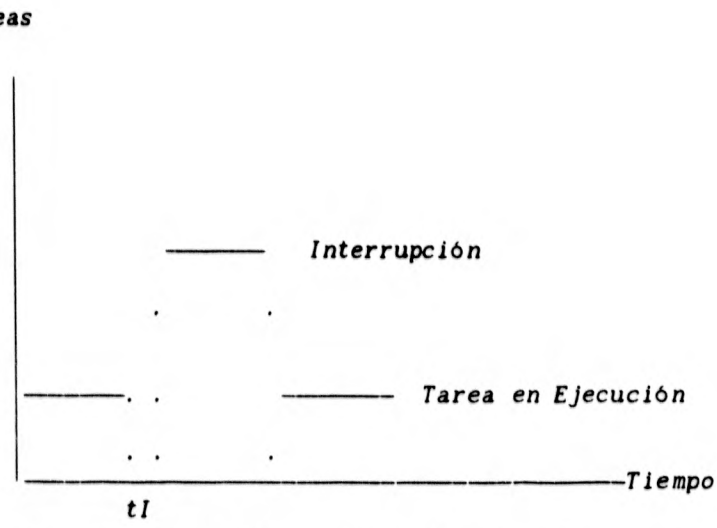



*Fig. 2.2 Tiempos de conmutación de Tareas y de Predicción*

a.3) Tiempo de Interrupción de Latencias: Es el tiempo que se lleva desde que el CPU recibe una interrupción requerida hasta la ejecución de la primera instrucción de servicio como lo muestra la figura 2.3.




*tI= Tiempo de Interrupción de Latencias*

*Fig. 2.3 Tiempo de Interrupción de Latencias*

a.4) Tiempo de Semáforo: Es el tiempo en el que una tarea es detenida hasta su activación por el semáforo como lo muestra la figura 2.4, donde:

*t1= Tarea 1 requiere semáforo  t3= Tarea 2 requiere semáforo*
*t2= Tarea 2 comienza  t4= Tarea 1 deja semáforo*
*t5= Tarea 2 prosigue y obtiene semáforo*
*ts= Tiempo de semáforo= t5-t4*

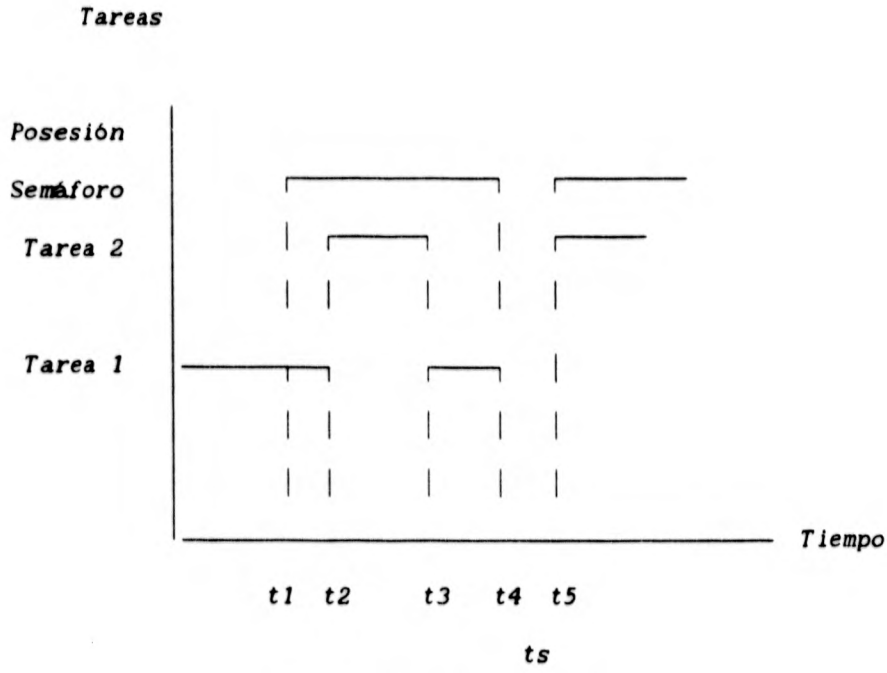



*Fig 2.4 Tiempo de semáforo*

a.5) Tiempo Muerto: Es el tiempo que ocurre desde que requiere la fuente, la última tarea de mayor prioridad, hasta que se termina de ejecutar la tarea de menor prioridad que requirió antes dicha fuente, más el tiempo que toma desde que termina la tarea de baja prioridad la cual primeramente pidió la fuente hasta que la tarea de alta prioridad toma la susodicha fuente, entendiendosé como fuente la utilización de un dispositivo de la computadora, ya sea bus, impresora etc, como lo muestra la figura 2.5.

*t1= Tarea 1 baja de prioridad recibe la fuente*
*t4= Tarea 3 de alta prioridad requiere la fuente*
*t5= Tarea 1 deja la fuente*
*t6= Tarea 3 recibe la fuente*
*t2= Tarea 2 de media prioridad inicia*
*t3= Tarea 3 inicia*
*ta= Tiempo desde que la tarea de alta prioridad 3 requiere la fuente hasta que la tarea 1 de baja prioridad se sigue ejecutando.*
*tb= Tiempo desde que la tarea 1 de baja prioridad dejala fuente hasta que la tarea 3 de alta prioridad recibe la fuente.*
*tdb= Tiempo muerto = ta + tb*

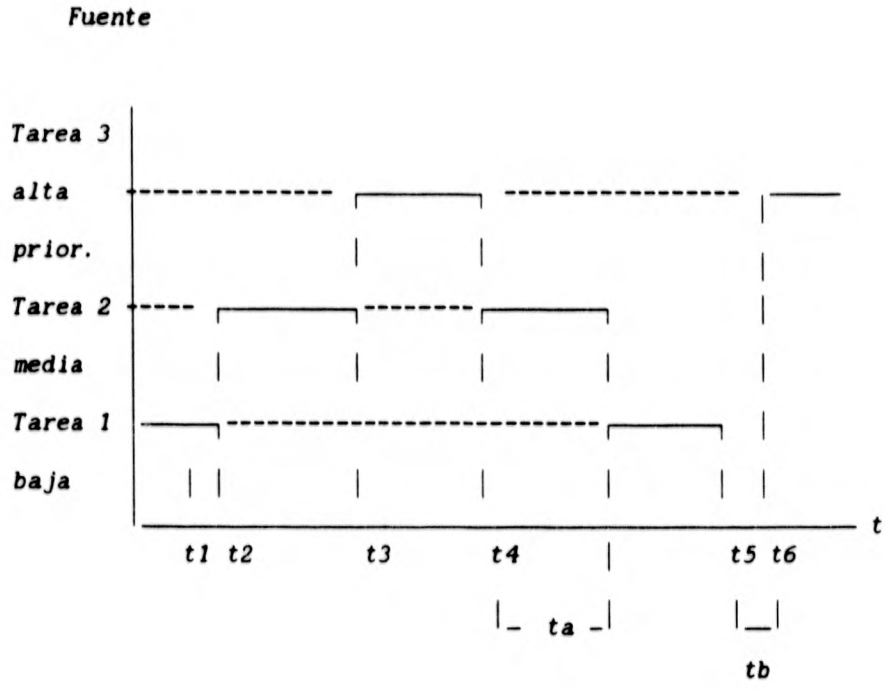



*Fig. 2.5 tdb = tiempo muerto*

a.6) Tiempo de Colocación: Es el número de Kilobytes/seg que una tarea puede mandar a otra, llamando al sistema operativo para poder realizar el proceso, esto se muestra en la figura 2.6.

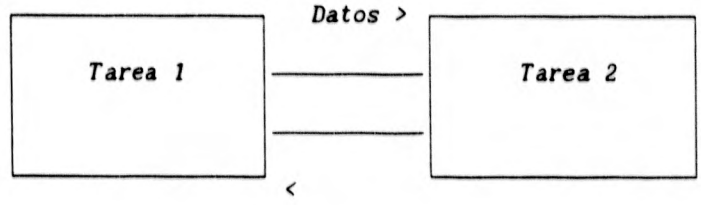



*Fig 2.6*

- Número de Rhealstone: Es el tiempo determinado por los 6 párametros anteriores en el rango de milisegundos a microsegundos. Tomando como referencia la frecuencia f de cada una de ellas, donde el número de Rhealstone estará definido por:

$$R = f1 + f2 + f3 + f4 + f5 + f6 \qquad (2.1)$$

Donde *f1* es el inverso del tiempo de conmutación de tareas, *f2* se refiere a la frecuencia de predicción de tiempo etc, etc,. El número de Rhealstone se puede modificar colocando coeficientes a las *fs*, con la finalidad de asignar determinada importancia a cada uno de dichos párametros de tal forma que el número de Rhealstone queda de la siguiente forma:

$$Rw = c1f1 + c2f2 + c3f3 + c4f4 + c5f5 + c6f6 \qquad (2.2)$$

b) Expedición de Latencias de Tiempo: Este concepto es otra medida comunmente usada en la evaluación de sistemas de tiempo real. Este tiempo puede definirse como el intervalo de tiempo cuando el sistema recibe una interrupción requerida hasta que comienza la ejecución de la aplicación de la tarea desempeñada por el sistema. Como se muestra en la figura 2.7, el tiempo de respuesta de interrupción consiste de latencias de hardware y software. El retardo de hardware es el tiempo requerido por el hardware para notificar al sistema operativo de la interrupción, el retardo restante es de software.

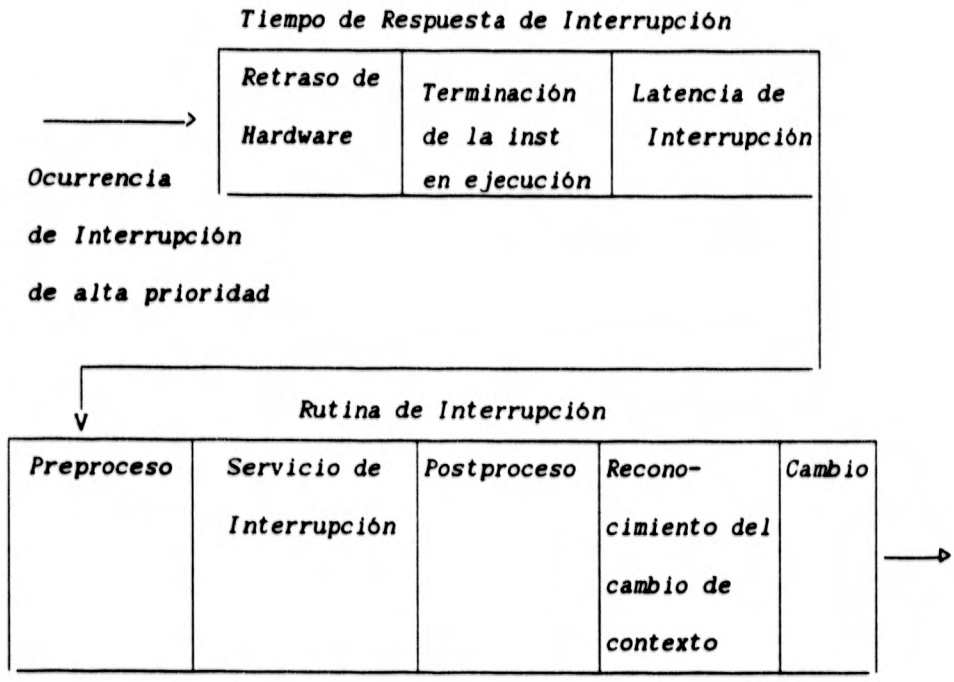



*Fig 2.7 Expedición de Latencias de Tiempo*

c) **Medidas tridimensionales:** Dentro de este concepto se tienen 3 medidas que se conjuntan para visualizar un concepto de velocidad desde 3 diferentes puntos de vista, estas medidas son:

- Velocidad de cálculo del CPU, medido en millones de instrucciones por segundo (MIPS1).

- Capacidad de manejo de interrupciones, medidas en millones de instrucciones por segundo (MIPS2).

- Operaciones de Entrada/Salida, medidas en millones de operaciones de Entrada/Salida por segundo, Mbytes/seg (MIPS3).

Con las medidas MIPS1, MIPS2, MIPS3 se pueden evaluar el desempeño del sistema, realizando un promedio que estará definido de la siguiente forma donde M1, M2, y M3 se refieren a las correspondientes medidas MIPS [1]:

$$Volumen = M1\ M2\ M3 \text{ , y la medida promedio}$$

$$\text{será : } MIPS = \sqrt[3]{Volumen} \qquad (2.3)$$

Dentro de este contexto se tienen los párametros *P* y *S* que se refieren a la eficiencia desde el punto de vista programa y desde el punto de vista sistema. Por lo que respecta al párametro *P* lo podemos definir como el tiempo que toma para completar un programa de aplicación a una interrupción especifica de carga, comparada con el tiempo de no interrupción de carga, se podrá definir de la siguiente manera [3]:

$$P = ( tn - to / to ) \times 100\ \% \qquad (2.4)$$

donde *to* es el tiempo de ejecución de no interrupción de carga y *tn* es el tiempo de ejecución de interrupción de carga. El párametro *S* define que porcentaje del total del tiempo transcurrido es destinado al manejo de interrupciones, donde *S* estará definida como [3]:

$$S = ( tn - to / tn ) \times 100\ \% \qquad (2.5)$$

## 2.6- MODELO DE UN SISTEMA DE TIEMPO REAL ABIERTO

Debido a la gran versatilidad de los sistemas abiertos la siguiente generación de sistemas en tiempo real será de este tipo, ya que esta clase de sistemas reduce el costo, el tiempo de proceso, incrementando su fácil integración, hacia otros sistemas. Como ejemplo [1] mencionaremos las caracteristicas de un sistema UNIX las cuales servirán como modelo para desarrollar el sistema en tiempo real de esta tesis, estas son [11]:

-La operación del sistema es standard basada en un sistema operativo en tiempo real UNIX.

-La arquitectura puede ser distribuida o centralizada dependiendo de la aplicación.

-El sistema abierto esta basado en un sistema general de lenguajes como el compilador standard del sistema.

-Incluye Interfaces de fácil manejo.

-El sistema esta provisto de conectividad hacia otros sistemas.

## 2.7- OPERACIÓN DE SISTEMAS EN TIEMPO REAL

La operación de un sistema en tiempo real se basa en 7 puntos principales ya que el sistema operativo puede ser definido como la colocación de programas, implementados en todo caso por medio de software o firmware, los cuales permiten usar eficientemente el hardware , siendo el sistema operativo la interface entre el hombre y la máquina, dichos puntos serán [11]:

1.- Dentro del punto de **soporte para la planeación de procesos de tiempo real**, el sistema debe ser capaz de soportar la creación, el borrado y la planeación de múltiples procesos. Típicamente un proceso de tiempo real permitirá el uso para definir prioridades e interrupciones para programación.

2.- Por lo que respecta a la **planeación de predicción**, se puede mencionar que el sistema operativo de tiempo real, deberá tener en cuenta el proceso de baja prioridad mientras ejecuta el proceso de alta prioridad.

3.- La **respuesta de interrupción garantizada**, se refiere al reconocimiento de un evento y una tarea determinística basada en dicho evento, dicha tarea estará definida en términos de función y tiempo. El sistema operativo deberá responder tanto al hardware como al software.

4.-El **interproceso de comunicación**, deberá contener semáforos, memoria y mensajes protocolarios.

5.-La **alta velocidad en la adquisición de datos** del sistema deberá tener en cuenta que moverá también a grandes velocidades los datos hacia discos y/o buffers.

6.- Las **entradas/salidas** no soportan operaciones de entrada/salida asíncronas, ni tampoco pueden conectarse directamente a dispositivos de entrada/salida.

7.- **Control de uso de sistemas fuente**,con respecto a este punto se puede mencionar que un sistema de esta naturaleza deberá tener el control de todas las fuentes del sistema incluyendo entradas/salidas, memoria y CPU.

## 2.8.- GENERALIDADES DE LOS SISTEMAS DISTRIBUIDOS

### 2.8.1.- Concepto de Control Distribuido

Por control distribuido se puede tener la siguiente definición [3], la cual dice que es un sistema autónomo de múltiples elementos de procesamiento, cooperando en un propósito común, basado en arquitecturas que conforman dicho sistema, donde dichas arquitecturas estarán sincronizadas entre sí de tal forma que tengan comunicación con un determinado procesador. Dentro de sus principales ventajas estan las siguientes[9]:

- Desempeño en paralelismo
- Incremento de aprovechabilidad en la redundancia
- Dispersión o distribución del poder de cálculo
- La facilidad para establecer redes de procesadores

Cabe hacer enfasis que los procesadores constituyendo un "pipeline", que es una arquitectura de tipo secuencial, en la cual cada procesador que lo constituye debe terminar su respectiva tarea y entregarla al siguiente procesador para utilizar la información y hacer lo propio con ésta, y así sucesivamente. Los "pipeline" no son elementos autónomos por lo tanto no estan contemplados como control distribuido.

### 2.8.2.- Clasificación de Control Distribuido

Dentro de la clasificación del control distribuido se encuentran básicamente lo siguientes [2]:

a) Los sistemas **estrechamente acoplados** basan su sincronización y comunicación en una memoria común

b) Los sistemas **libremente acoplados** necesitan un mensaje de paso y no tienen memoria común.

Otro tipo de clasificación puede basarse en la variedad de procesadores en el sistema, esta sería la siguiente:

a) Sistema homogeneo: Todos los procesadores que constituyen el sistema son del mismo tipo, desde el punto de vista de su arquitectura.

b) Sistema heterogeneo: Todos los procesadores que constituyen el sistema, son de diferente tipo, desde el punto de vista arquitectura.

## 2.8.3.- Caracteristicas del Control Distribuido

Un sistema de tiempo real distribuido está constituido por dos sistemas; un sistema controlado y un sistema controlador, el sistema controlador interactúa con el medio basado en información sobre el mismo medio, dichos sistemas deberán tener sensores los cuales estarán monitoreando cada cierto periodo de tiempo. El sistema controlado estará formado por el proceso a controlar. En resumen los sistemas de tiempo real distribuido difieren de los sistemas tradicionales en que las lineas muertas o otros tiempos restringidos son adheridos al de las tareas. De esta forma los tiempos muertos y restringidos se adicionan a una tarea, la cual tiene una duración determinada. La especificación de un sistema de tiempo real estará determinada por la relación entre el sistema y su medio de donde se obtendrán las especificaciones y en base a esto se tendrá una seguridad del sistema para desempeñarse en su medio [3] y [9].

La exactitud de los sistemas de tiempo real no solo dependen del resultado lógico de los cálculos, si no también del tiempo en el cual los resultados son producidos. Por lo tanto, las caracteristicas de un sistema distribuido en tiempo real, en cuanto a su base de datos, es diferente con referencia a los sistemas de bases de datos convencionales, ya que en un sistema en tiempo real se tienen que desempeñar operaciones estáticas, información continuamente actualizada. En términos generales los sistemas de tiempo real distribuidos pueden ser vistos en tres etapas [3]:

- Adquisición de datos
- Procesamiento de datos
- Entradas/Salidas para actuadores y/ó indicadores

Desde el punto de vista conexión de nodos, se debe de tener en cuenta su homogeneidad, para ubicarlas en cualquier parte del sistema, también escalabilidad, la cual permite rediseñar módulos, supervivencia, la cual permite evitar un desastre cuando dos nodos que deben de estar en comunicación, dejan de estarlo, y por último flexibilidad experimental, la cual permite emular algunos nodos desconectados, en caso de que estos fallen. En un sistema distribuido el proceso de intercomunicación no es insignificante con respecto a la ejecución del proceso. Básicamente un modelo de colocación de tareas para sistemas de cómputo distribuido deberá permitir especificaciones de un gran número de restricciones para facilitar una variedad de requerimientos de aplicaciones de ingeniería, como son: balancear la utilización de procesadores individuales en el sistema distribuido y minimizar el proceso de intercomunicación. Debido a lo anterior los aspectos siguientes son de primordial importancia en un sistema distribuido [2]:

a) Configuración: Se debe tener en cuenta la configuración de tal forma que se pueda tener una eficiencia máxima, que permita minimizar tiempo de ejecución.

a.1) Un simple programa puede configurarse en fragmentos con los cuales se hará la comunicación, dicho programa permitirá la comunicación entre procesadores y su sincronización, donde se puede tener post y prepartición del programa. Por lo que respecta a la postpartición, es una estrategia de configuración después de haber escrito el programa. En cuanto a la prepartición se puede decir que es una estrategia para seleccionar un módulo liga como una sola unidad de configuración, de tal forma que se planea antes de la realización del programa , dentro de este contexto se tiene el concepto de nodo virtual como se muestra en la figura 2.8, dicho nodo es una estructura en la cual se tienen diferentes procesos, variables compartidas y procedimientos compartidos.

a.2) En contraste, un multiprograma radica en uno ó más programas para cada procesador, desarrollando la misma función que el punto anterior.

Para un lenguaje construido basado en nodo virtual este deberá soportar las siguientes características; compilación separada, varios nodos virtuales, reconfiguración dinámica

Existen diferentes tipos de configuraciones por lo que respecta a los nodos virtuales. De tal forma que puede existir una estructura que tenga un nodo o varios nodos virtuales interconectados entre sí, así como las unidades de librería que estarán conectados al o a los nodos virtuales.

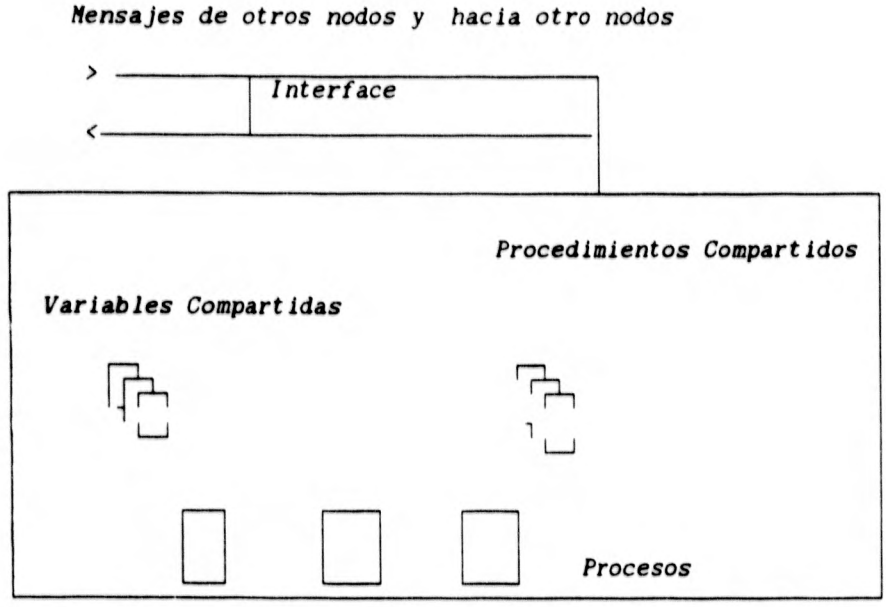



*Fig 2.8 Estructura de un nodo virtual*

b) Confiabilidad: La disponibilidad de múltiples procesos proporciona una habilitación para tolerancia a fallas. En este aspecto un control distribuido debe tener la facilidad para comunicación entre diferentes etapas constitutivas del control distribuido, así como se debe tener en cuenta el hecho de tener un multiprocesador que reconfigure el sistema en caso de falla, en cambio con un solo procesador no sería posible lograr esto.

c) Algoritmos de Control Distribuido: La presencia de algoritmos adecuados para comunicación etc. Dentro de este punto mencionaremos que cuando se tiene un uniprocesador no se presenta dificultad con respecto a los sistemas en cuanto a su acoplamiento ya que comparten la misma memoria. En cambio para sistemas distribuidos, ya que tienen diferente reloj, existe un retraso en la transmisión de mensajes entre procesadores ya que estos sistemas tienen dos importantes características [2]:

- Para cualquier secuencia de eventos dada, es imposible asegurar que dos diferentes eventos observen la misma secuencia.

- Cuando un estado de un proceso cambia esto puede ser visto como un evento

Por eso los procesos en una aplicación distribuida son coordinados y sincronizados. Será necesario colocar en orden estos eventos, en otras palabras si existen dos procesos *P* y *Q* donde son procesos distribuidos se tienen que realizar todos los subprocesos de *Po*-a-*Pn* y de *Qo*-a-*Qn* respectivamente, teniendo en cuenta que los subprocesos *Qs* son independientes de los subprocesos *P*s siempre que no exista comunicación entre ellos, de otra forma si existe comunicación entre ellos los subprocesos entre sí serán dependientes, de tal forma que tenga que haber pasado *Po* ó *Qo* primero, solamente uno antes que el otro, por ejemplo de *P1* a *Q4*, se tiene que pasar de *P1*-a-*Q3*, y de *Q3*-a-*Q4* a estos eventos se les llama concurrentes [2] ( figura 2.9).

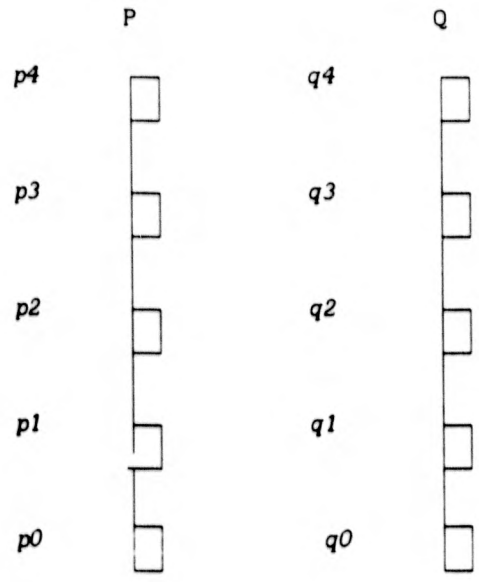



*Fig 2,9 Dos procesos interactuando*

Para ordenar todos los eventos en un sistema distribuido es necesario asociar un tiempo asignado a cada evento. En cuanto a la programación de bajo nivel, básicamente se tienen dos clases de arquitecturas, una con separación del bus de memoria y del bus de entrada/salida, otra compartiendo el mismo bus como se muestran en las figuras 2.10 y 2.11.

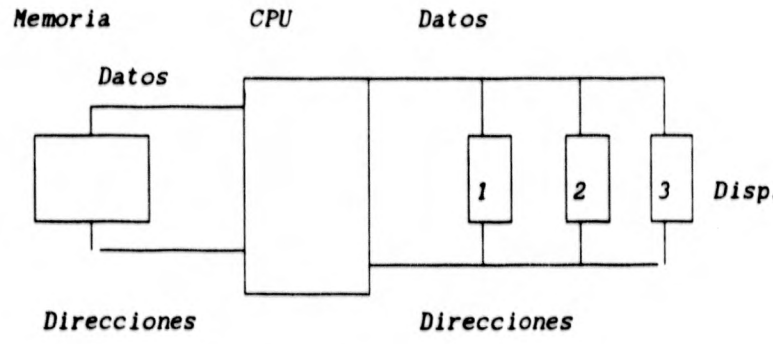



*Fig. 2.10 Arquitectura con Buses Separados*

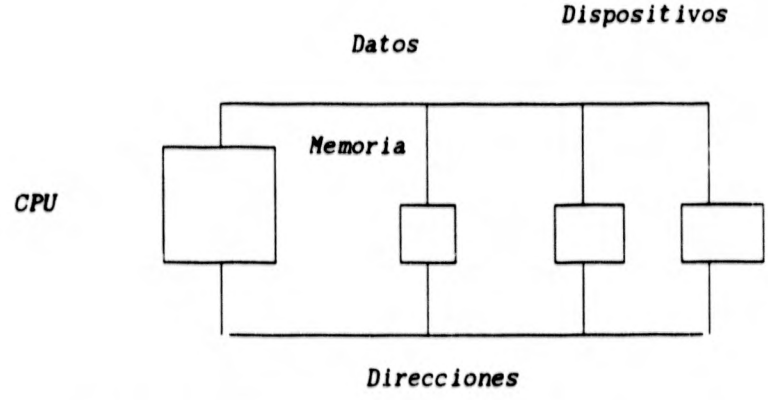



*Fig 2.11 Arquitectura compartiendo la memoria el mismo bus*

En el manejo del estatus, se tienen operaciones de prueba, control de operaciones, operaciones de Entrada/ Salida, por otro lado se tiene el manejo de interrupciones, dentro de este concepto se pueden manejar interrupciones controladas por programa o por hardware. Los elementos requeridos para manejar dispositivos de interrupción son; el control de interrupción, identificación de interrupción, mecanismos de switcheo, [3] etc,..

## 2.9.- CONCLUSIONES

Este capítulo ha dado las pautas que deberá seguir nuestro sistema en cuanto a estructura para que cumpla con las necesidades básicas de un sistema distribuido en tiempo real, de esta forma implementar este sistema para que tenga el mayor grado de aplicación a diferentes tipos de sistemas que necesiten un sistema de esta naturaleza en diferentes áreas.

# 3.- PROGRAMACIÓN EN TIEMPO REAL

## 3.1.- INTRODUCCIÓN

En este capítulo se aprecia la estructura algorítmica de los programas en tiempo real, identificando las características que deben reunir dichos programas. Y así esta forma utilizar en forma eficiente la estructura del hardware donde estarán residentes y así poder estructurarlos para cumplir con la función que se les asigne, como lo es el controlar los procesos y la sincronización y la conmunicación entre los controladores externos y el monitor central.

## 3.2.- DESCRIPCION GENERAL

Los sistemas en tiempo real están caracterizados por el hecho de que resultarían severas consecuencias si el tiempo correcto de las propiedades del sistema no son satisfechas . El cálculo en tiempo real es el tipo de cálculo donde lo correcto del sistema no solamente depende del tiempo del cálculo, si no también del tiempo en el cual los resultados son producidos. Por otra parte, debemos diferenciar los conceptos de cálculo rápido y cálculo en tiempo real, mientras que el objetivo de cálculo rápido es minimizar el promedio de tiempo de respuesta de un conjunto dado de tareas, el objetivo del cálculo en tiempo real es de encontrar el *requerimiento* individual de tiempo de cada una de las tareas, teniendo la característica de *predicibilidad*, característica que el cálculo rápido no tiene. Algo que vale la pena mencionar, es que mientras la programación en lenguaje ensamblador, interrupciones de programación, y escritura a dispositivos manejadores son aspectos de cálculo en tiempo real. De esta forma podemos resumir que la programación en tiempo real deberá cumplir los siguientes requerimientos [2]:

-Especificación de tiempo en el cual las tareas son desempeñadas

-Especificación de tiempo en el cual las tareas son terminadas

-Respuesta a situaciones donde todos los requerimientos no pueden ser conocidos

-Respuesta a situaciones donde los requerimientos son cambiados dinámicamente.

## 3.3.- CARACTERÍSTICAS DE PROGRAMACION Y DE LENGUAJE

Dentro del diseño de un sistema de tiempo real se tomará en cuenta los siguientes aspectos en cuanto a su estructura [2]:

a) Guía de programación: Dentro de este primer aspecto se deberá tener en cuenta lo siguiente:

a.1) Requerimientos de acuerdo a la aplicación que se le deba dar.

a.2) Debe ser funcional de tal forma que se pueda aislar en programas.

a.3) El interproceso de comunicaciones.

a.4) El uso de la tarjeta adecuada de acuerdo al dispositivo periférico accionado [1].

Por otra parte los programas en tiempo real tienen 3 distintas secciones; una que inicializa y preasigna fuentes requeridas por el proceso, una que prepara el proceso para ejecución y coloca la prioridad del proceso y una que contiene la funcionalidad del programa [80].

b) Planeación de Procesos: Se tendrán en cuenta los siguientes dos puntos principales que serán :

b.1) Un proceso corre solamente cuando otro proceso de mayor prioridad no corre.

b.2) El proceso que esta corriendo tiene el control del CPU hasta que llega otro proceso de mayor prioridad y toma el control del CPU.

c) Asignación de Memoria: Para programas en tiempo real se preasigna una cantidad considerable de memoria. En un sistema de operación en tiempo real se podrá escribir e instalar un sistema de llamadas de tal forma que no se tengan que reconstruir y reescribir el sistema operativo.

Debido a esto, los programas en tiempo real deberán tener características que permitan desempeñarse en forma adecuada, permitiendo optimización en las tareas a realizar así como el tiempo necesario para llevar estas a cabo. Dichas características serán las siguientes [2]:

d) Administración de tiempo: Será capaz de soportar restricciones en tiempo, creando un módulo de software,de tal forma que el programa tenga una estructura donde se puedan guardar, detectar y utilizar módulos de software,[4] una limitación en tiempo impone una restricción en el sistema la cual está dividida en:

d.1) Desempeño de la limitación, la cual coloca límites en la respuesta de tiempo del sistema.

d.2) Conducta de la limitación, que hace demandar la velocidad a la cual el usuario aplica un estímulo al sistema [4]. Estas restricciones desde el punto de vista temporal se pueden clasificar de la siguiente manera:

d.2.1) **Máxima**: No mas que una cantidad de tiempo t que sucede entre un evento y la ocurrencia de otro. Dentro de este aspecto podemos mencionar 4 tipos que son los siguientes:

d.2.1.1) Combinación S-S: Un máximo tiempo es permitido entre las ocurrencias de dos estímulos, ejemplo, después de haber marcado un digito de un número teléfonico , el segundo dígito no debera ser marcado si no hasta dentro de 20 segundos.

d.2.1.2) Combinación S-R: Un máximo de tiempo es permitido entre la llegada de un estímulo y la respuesta del sistema.

d.2.1.3) Combinación R-S: Un máximo de tiempo es permitido entre la respuesta del sistema y el siguiente estímulo.

d.2.1.4) Combinación R-R: Un máximo de tiempo es permitido entre dos respuestas del sistema.

d.2.2) **Mínima**: No menos que una cantidad de tiempo t deberá suceder entre un evento y la ocurrencia de otro.

d.2.3) Duracional: Un evento deberá ocurrir en cierta cantidad determinada de tiempo.

e) Planeabilidad: Deberá tener algoritmos que permitan determinar si los requerimientos de tiempo pueden ser conocidos por el tamaño del set de tareas dado.

f) **Módulos de Software reusables**: Son un conjunto de instrucciones que permiten simplificar el desarrollo de software, así como el tiempo de procesamiento ya que se puede decir, que son módulos que tienen una duración perfectamente definida, y se pueden usar cuantas veces sea necesario.

Por otra parte si se tiene un sistema distribuido, para satisfacer los requerimientos de tiempo, deberán cumplir dos características [6]:

g) Concurrencia: La primera se refiere a que el grado de concurrencia en el desarrollo de procesamiento deberá incrementarse.

h) Integración: Se refiere a la integración de algoritmos concurrentes, de prótocolos de control y planeación.

i) Arquitectura: Este aspecto afecta directamente a la programación de tiempo real ya que depende del tipo de arquitectura donde se implemente, teniendo este aspecto a su vez 3 rubros importantes, que son los siguientes:

i.1) Homogeneidad: Debido a la homogeneidad, las tareas pueden ser colocadas en cualquier nodo, pudiendo cambiar la tarea de un nodo a otro.

i.2) Escalabilidad: Esto permite que la potencia de cálculo de la red pueda ser modificada en cualquier momento sin que se rediseñe cualquiera de los nodos y cause un determinado problema.

i.3) Supervivencia: Dado un par de nodos se deberá tener una configuración de tal forma que la comunicación entre nodos sea una comunicación lo más simple de tal forma que pueda sobrevir esta a una falla.

Con respecto a las características del lenguaje se tendrá que tomar en cuenta los siguientes puntos [6]:

j) Seguridad: Esto básicamente se refiere a que deben ser detectados errores de inmediato en el lenguaje por el bien del programa.

k) Legibilidad: En este punto se definen tipos de módulos y por consiguiente se facilita el programa al modularizarlo. Aún teniendo la desventaja de incrementar la longitud del programa.

l) Flexibilidad: Debe de estar estructurado de tal forma que se puedan insertar comandos o códigos de máquina y restaurar los existentes.

m) Simplicidad: Minimiza el esfuerzo requerido producido por los compiladores, así como la posibilidad de producir errores de programación como resultado de una interpretación desconocida.

n) Portabilidad: Deberá ser independiente del hardware en el cual se ejecute ya que deberá ser transportable a cualquier otra máquina.

o) Aspecto Lexicológico: Se tendrán que tomar en cuenta paréntesis, comas, diferentes letras, etc, para que cada uno de estos términos tenga un significado especifico. Así como el estilo en donde se tomarán en cuenta las declaraciones en el programa, los tipos de datos, los tipos de números, las interrupciones, el comienzo,y la finalización.

## 3.4.- ESTRUCTURACIÓN DE PROGRAMACIÓN EN TIEMPO REAL

La administración de un software en tiempo real permite una optimización adecuada, la cual se reflejará en el desempeño del sistema. Básicamente existen dos formas de estructurar los sistemas en tiempo real, que son las siguientes [2]:

a) Descomposición: Este método sugiere el involucramiento del desglozamiento de sistemas complejos dentro de pequeñas partes hasta llegar a elementos aislados, y de esta forma poder estructurar, de acuerdo al diseño deseado, el programa en tiempo real.

b) Abstracción: Dicho método permite ver con detalle cada uno de los niveles, de tal forma que se desglosa toda la estructura por nivel.

Por otra parte, las dos medidas que relacionan a los módulos que componen el sistema se denominan Cohesión y Acoplamiento que fueron mencionados en el cápitulo 2.

c) Cohesión: Se refiere a los diferentes tipos de enlazamiento que se tienen entre módulos, los cuales pueden ser cualquiera de los siguientes:

c.1) Coincidental: Elementos de módulo que estan ligados por una razón muy superficial, por ejemplo mandar datos al mismo display.

c.2) Lógico: Elementos del módulo son relacionados en términos del sistema en general pero no en términos de software actual, por ejemplo todos los dispositivos de salida.

c.3) Temporales: Elementos de programa que deben ser ejecutados al mismo tiempo.

c.4) Procedimientos: Elementos de módulo que son usados juntos en la misma sección del programa, por ejemplo, componentes usuario- interface.

c.5) Comunicaciones: Elementos del módulo de trabajo en la misma estructura de datos , por ejemplo, algoritmos usados para analizar una señal de entrada.

c.6) Funcionales: Elementos del módulo trabajando juntos para contribuir al desempeño de una función del sistema, por ejemplo, la provisión de un sistema de archivo distribuido.

d) Acoplamiento: Por comparación, es una medida de interdependencia de los módulos del programa. Por ejemplo, si dos módulos cruzan información de control entre ellos, se dice que estan altamente acoplados, en cambio si solamemnte se transmiten datos, alternativamente se pierde el acoplamiento, es decir que tan fácil se puede remover un módulo de tal forma que se reemplace con otro. Dentro de los métodos de diseño, una buena descomposición es el que tiene fuerte cohesión y pérdida de acoplamiento.

Desde el punto de vista de estructura de control, para una determinado fin, se tendrán basicamente 3 diferentes tipos de estructuras [2]:

- Secuenciales: Donde la programación lleva una determinada secuencia en la cual, se va ejecutando instrucción tras instrucción.

- De Decisiones: En este aspecto se tendrán dos o más posibles caminos hacia donde seguir, de tal forma que, el camino a seguir dependerá de las circunstancias de determinadas variables.

- De lazo: Podemos decir en este punto que este tipo de estructura podrá ser de tipo iterativo o recursivo, donde los lazos iterativos van comenzando uno tras otro hasta que el anterior ha terminado, en el recursivo un determinado lazo puede ser interrrumpido por aparición de un segundo lazo de más prioridad.

### 3.4.1.- Interrupciones y su Manejo

Dentro de las estructuras de programación, podemos mencionar también a las interrupciones las cuales deberán cumplir las siguientes caracteristicas [4]:

-Tendrá una facilidad de uso para entenderse y usarse.

-El código de interrupciomnes será de fácil interpretación para evitar errores.

- El mecanismo deberá ser diseñado para que el tiempo de corrimiento general contemple el manejo de interrupciones.

-El mecanismo deberá ser capaz de permitir un tratamiento de interrupciones detectando estas debido al ambiente donde se trabaja y/ó por el programa mismo.

-Dominio de un manejador de interrupciones, este deberá ser especificado de tal forma que cuando la interrupción se presente, el manejador correspondiente deberá ser activado. En el caso de que no exista manejador de interrupciones en un determinado dominio, cuando la interrupción se presente y no encuentre un manejador, esta se irá a buscar a otro manejador.

Para esto se tienen dos modelos, el de Reanudación y el de Terminación que a continuación detallaramos:

- En el de REANUDACION, se contemplan 3 procedimientos P,Q y R en donde P invoca Q el cual invoca R y R levanta una determinada interrupción r la cual es manejada por Q; asumiendo que no es manejador local en R. El manejador para r es Hr. En la intención de manejar r, Hr levanta una interrupción q la cual es manejada por Hq en el procedimiento P (y la llamada de Q), luego esto es manejado por Hr y finalmente continua la ejecución terminando en R, esto se muestra en la figura 3.2.

- En el modelo de Terminación cuando una interrupción ha sido levantada y el manejador ha sido llamado, el control no retorna a el punto donde la interrupción ha ocurrido. En primera instancia el procedimiento contiene el manejador y el control. Un procedimieno invocado, por lo tanto, puede terminar en un determinado número de condiciones. Una de estas es la condición normal mientras que las otras son condiciones de interrupción figura 3.3.

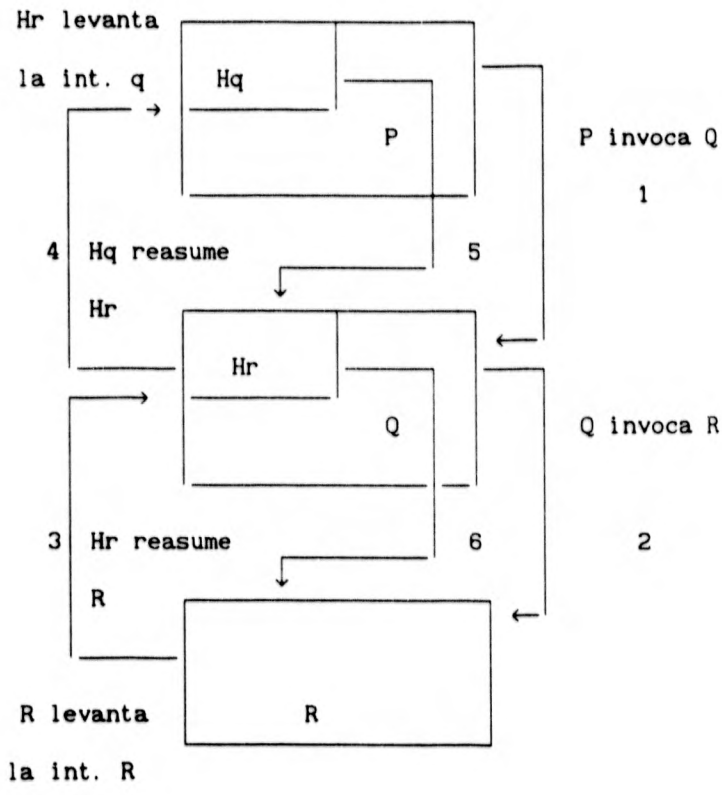
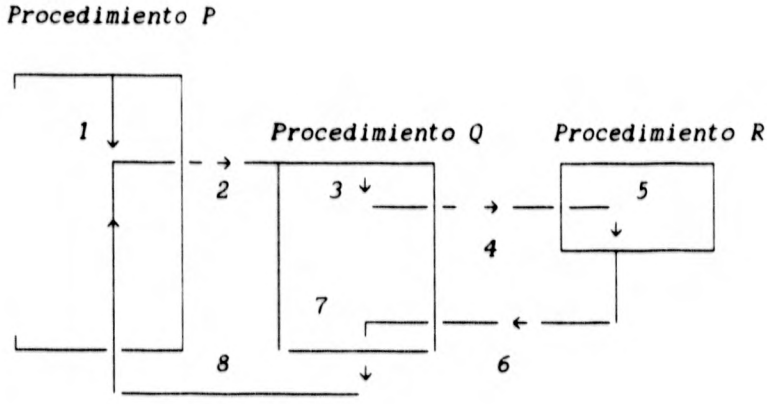



*Fig 3.1 Modelo de Reanudación*




Donde:

*Llamada a Q = 2, Llamada a R = 4*

*Reasume interrupción = 5, Solicitud de Manejador = 6*

*Manejador = 7*

*Fig. 3.2 Modelo de Terminación*

### 3.4.2.- Comportamiento de Programas en Tiempo Real

Dentro de este aspecto podemos mencionar que las estructuras de programación en tiempo real permiten que: el tiempo de ejecución de una simple declaración sea establecido a partir de que el procesador ejecuta la declaración conocida, además de tomar las acciones consecuentes como declaraciones, las cuales son ejecutadas por el procesador sin que exista interrupción, todo esto se debe a un determinado comportamiento en su lenguaje y a las restricciones en tiempo que tienen [6]. Con respecto a su lenguaje podemos mencionar que, virtualmente todos los sistemas de tiempo real son inheremente concurrentes, entendiendosé esto como la notación de programación y técnicas para expresar paralelismo potencial, y resolver problemas teniendo como resultado sincronización y comunicaciones en tiempo real. La construcción de programación concurrrente varía de un lenguaje a otro, estas variaciones podrán ser de acuerdo a lo siguiente [3]:

a) La expresión de ejecución de concurrencia a través de la notación de programación.

b) Comunicación de interprocesos: Ahora los diferentes tipos de procesos deberán estar dentro de cualquiera de los siguientes tipos, estando los procesos constituidos por programas y/o subprogramas:

b.1) Independiente: Son procesos que no se comunican o se sincronizan con otros.

b.2) Cooperativo: Regularmente se comunican y se sincronizan entre sí para desempeñar alguna actividad en común.

b.3) Competición: En este punto se contemplan equitativas particiones de fuentes tales como dispositivos périfericos, memoria y potencia de procesamiento.

Con respecto a la ejecución concurrente habrá posibles variaciones de:

c) La estructura de los procesos puede ser clasificada básicamente como:

c.1) Estática: El número de procesos es fijo y además se conoce el tiempo de compilación.

c.2) Dinámico: Los procesos son creados en cualquier instante, el número de procesos existente es fijado en un determinado intervalo de tiempo.

d) Con respecto a la distinción entre lenguajes a nivel de paralelismo, podemos mencionar que existen básicamente dos diferentes tipos:

d.1) Anidado: Los procesos son definidos en cualquier nivel de texto de programación

d.2) Fijo: El proceso es definido solamente en un nivel de texto de programación.

e) Todo proceso tendrá una inicialización y una terminación dentro de la inicialización se tendrán dos formas de desempeño:

e.1) La primera es para pasar la información en forma de parámetros al proceso.

e.2) La segunda es para comunicarse explícitamente con el proceso después que este ha comenzado la ejecución.

f) Por otra parte la terminación puede ser descrita en una variedad de formas como las siguientes; ejecución complementada del proceso, ejecución por autodeterminación, aborto, y ocurrencia de un error.

### 3.4.3.- Representación de Procesos de Programación

Dentro de la representación de procesos podemos mencionar que existen varios métodos para expresar ejecución concurrente [6] , a continuación se explican:

Corutina: Son un conjunto de subrutinas que permiten pasarse el control entre ellas. De tal forma que se podrá atender a cada subrurtina en forma multiplexada (figura 3.4).

Fork (Bifurcación): Solamente soporta dos subrutinas, la declaración de bifurcación especifica que una rutina diseñada deberá comenzar ejecutandose concurrentemente con el invocador de la bifurcación.

Join (Unión): La declaración de unión permite al invocador sincronizar con la completación de la rutina invocada.

Cobegin: Esta estructura denota la ejecución concurrente de una colección de declaraciones.

## 3.5.- CLASIFICACIÓN DE PROGRAMAS EN TIEMPO REAL

En la implementación, un importante planteamiento entre el nivel tope de especificaciones de requerimientos y la ejecución de códigos de máquina es el lenguaje de programación . Es posible identificar 3 clases de lenguajes los cuales son usados en el desarrollo de sistemas de tiempo real [6]:

-Lenguaje Ensamblador: El cual permite un ahorro considerable en tiempo ya que no pasa por ningún compilador de alto nivel, teniendo la desventaja de que en términos generales son implementados en máquinas orientadas para problemas especificos.

-Lenguaje de Alto Nivel Concurrente: Pueden ser aplicados en diferentes maquinas (transportabilidad) pueden ser cambiados en un determinado momento (modificabilidad).

-Implementación de Sistemas Secuenciales de Lenguaje: Básicamente tiene las mismas ventajas que el anterior pero, con la ventaja de poder insertar códigos de ensamblador.

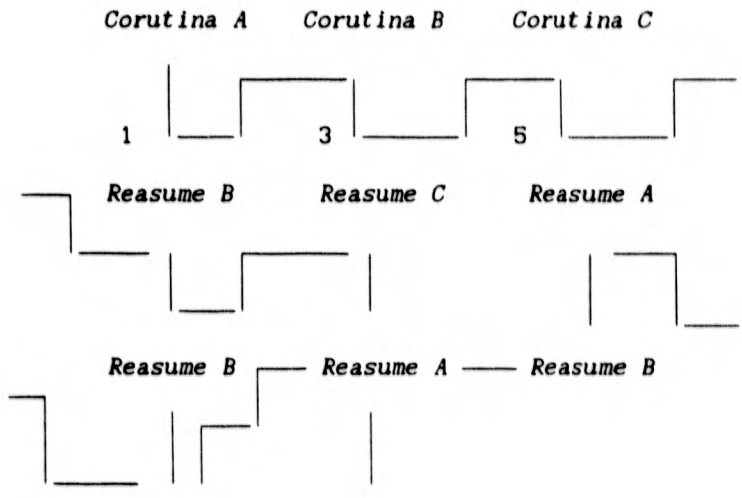



*Fig. 3.3 Flujo de Control de una Corutina*

## 3.6.- CONCLUSIONES

La programación en tiempo real tiene características tales que los algoritmos de control utilizados deben ser totalmente compatibles con la arquitectura en la que son implementados. Por otra parte, la estructura de dichos programas debe ser cuidadosamente estructurada, ya que las llamadas a subrutinas e interrupciones así como la interacción con dispositivos de entrada/salida y con otros procesadores debe estar perfectamente sincronizada para que cumpla con las características de tiempo real.

# 4.- COMUNICACIÓN ENTRE PUESTO CENTRAL Y CONTROLADORES EXTERNOS

## 4.1.- INTRODUCCIÓN

En este capítulo se da un panorama de la interacción entre el monitor central y los controladores externos de tal forma que permita implementar los mecanismos de comunicación adecuados que para realizar sus funciones ya que los sistemas de control distribuidos estan diseñados de modo que puedan controlar diferentes tareas al mismo tiempo en diferentes controladores. Cada tarea deberá ser capaz de reconocer el software y hardware que necesita para poder desempeñar sus funciones, a fin de sólo tomar en cuenta las variables, el software, y los controladores que requiera para desempeñar su labor. Teniendo en cuenta que cada tarea por su naturaleza puede ser diferente, como consecuencia algunas tareas podrán ser activadas por señales externas o por otra tarea, es decir una tarea puede depender directamente de otra, ó bien pueden ser interdependientes entre ellas. En la comunicación entre el puesto central y los controladores externos, los cuales controlarán determinados procesos controlando un proceso por controlador, se tomará en cuenta el tiempo que tarda la comunicación. En un sistema que no esta basado en tiempo real, es suficiente con que se establezca la comunicación no tomando en cuenta el tiempo que tarda esta. Con respecto a las tareas, como se había mencionado, el objetivo es satisfacer los requerimientos de tiempo de cada una de ellas, en cambio en un sistema de cálculo rápido el objetivo es minimizar el promedio de tiempo de respuesta de un conjunto de tareas.

## 4.2.- DESCRIPCIÓN DE LAS COMUNICACIONES

Desde el sistema operativo, podemos mencionar la forma como ve éste la transmisión de un mensaje. Desde este esquema un nodo A manda un mensaje M a un nodo B siguiendo los siguientes pasos [2]:

-Nodo A coloca una determinada cantidad de tiempo para recibir el reconocimiento de mensaje M desde el nodo B.

-Nodo A manda el primer caracter del mensaje M y entonces sobre cada interrupción de salida manda el siguiente cáracter.

-Cuando el nodo A manda el último cáracter, este no comienza a mandar el siguiente segmento completo de mensaje al nodo B hasta que en todo caso recibe de B un reconocimiento y verifica que esto es correcto, en tal caso este procede a mandar el siguiente mensaje, ó retransmite el mismo.

-Nodo B recibe los cáracteres, uno por cada interrupción de entrada y reconstruye el mensaje.

- Cuando B recibe el último cáracter de mensaje, este verifica que el mensaje sea correcto, mandando el nodo B un reconocimiento de mensaje correcto de recepción del nodo A.

- Si el nodo B comienza, recibiendo una retransmisión prioritaria para tener completo el mensaje se resetea para comenzar. Si el nodo B recibió correctamente el mensaje, ignora el mensaje de retransmisión.

- Un reconocimiento se necesita para saber si se trata de mensajes o bien de verificación de mensajes correctos.

Como en cualquier diseño de un sistema de comunicaciones se tendrán que implementar protocolos de comunicación que logren un enlace adecuado entre el Puesto Central y los controladores conectados a este, se tendrá que tomar en cuenta entre otros aspectos la limitación de acceso a los canales de comunicación y los retrasos que se tengan en los mensajes. Entendiendo el Retraso de Canal como el intervalo entre el instante que tarda cuando un mensaje es colocado en el nodo a transmitir y el instante en que es recibido por el nodo receptor, teniendo como consecuencia la suma de los dos tiempos a tomar en cuenta. Con respecto a este factor tiempo, se deberá contemplar en la planeación del sistema, los tiempos que consume cada tarea. En el desempeño del sistema se deberá tomar en cuenta el hecho de una posible falla en la comunicación entre dos nodos. Esto sucede en transmisiones periódicas en el que si un nodo no recibe comunicación desde el nodo transmisor antes de un determinado tiempo, dicho nodo inicia una determinada acción de respuesta a este hecho en la cual se da por entendido que no hay nada por recibir o se inicia una determinada secuencia que permita conocer el mensaje si es que existe, pero que no se pudo mandar. En los Sistemas Distribuidos podemos manejar las tareas básicamente de dos formas; una en la cual considerando el máximo tiempo de retraso que un mensaje puede tener, de esta forma los demás tiempos serán menor a este tiempo considerado el mayor tiempo en retraso que pudiese existir, la otra forma es considerar un prótocolo de comunicaciones en el cual detectemos cuando un mensaje determinado esta listo para ser transmitido [5].

### 4.2.1.- Protocolos de Comunicación

Comunicación entre procesos en diferentes máquinas en un sistema distribuido requieren mensajes ya sea para ser transmitidos y/ó recibidos en el sistema de comunicación fundamental a continuación se darán los principales requerimientos en un sistema de comunicación distribuido [11]:

- Como el sistema tendrá diferentes puntos de acceso se necesitará un protocolo de comunicación.

- Se deberá tener un protocolo apropiado de tal forma que si se necesita retransmitir el mensaje se deberá hacerlo.

- Se dispondrá de un buffer de almacenamiento de transmisión de tal forma que el contenido de este se transmitirá antes de que un nuevo dato pueda ser colocado en éste.

- Deberá ser capaz de soportar diferentes niveles de prioridades para las comunicaciones.

Los prótocolos deberán estar diseñados de tal forma que si se manda un mensaje, todos los controladores estarán enterados de esta situación para evitar conflictos en la transmisión de los mensajes, a este tipo de estructura se le denomina de Múltiple Acceso [5]. Los prótocolos de Múltiple Acceso se divididen en [5]:

a) Base de Argumentos: Los cuales competirán por ganar el acceso al canal, esto opera de tal forma que particiona los controladores de la red formando un conjunto de controladores que tendrán derechos de transmisión y un conjunto de controladores sin derechos de transmisión, los cuales estarán deshabilitados. En caso de que ninguna de las estaciones habilitadas este lista, el canal permanece sin uso y eventualmente una nueva partición podrá ser determinada. En caso de que una estación habilitada este lista, entonces esta transmite el mensaje. Al fin de cada transmisión una nueva partición es determinada. Finalmente si dos o más estaciones habilitadas estan listas, estas estaciones tendrán conflicto con respecto a sus mensajes, si las estaciones pueden detectar y abortar las transmisiones, se aisla una estación de tal forma que solo mande el mensaje una de ellas. En caso de que no se pueda detectar y abortar la transmisión una nueva partición es determinada. Dichas particiones estarán clasificadas como:

a.1) Partición de Tiempo: En este modo se asigna una ventana de tiempo de tal forma que se verifica que estación intenta mandar el mensaje, y si hay más de una se mantendrá un orden determinado previamente asignado, así en la primera ventana, la estación correspondiente mandará su mensaje, dicha ventana se particionará de tal forma que en una segunda ventana creada por la partición de la primera, transmitirá la estación correspondiente.

a.2) Fundamentos Probabilísticos: En este tipo de mecanismo no hay coordinacion entre las estaciones distribuidas, es decir si dos estaciones o más mandan al mismo tiempo su correspondiente mensaje, estos chocaran por lo tanto, cada estación deberá estar programada para retransmitir el mensaje después de una determinada cantidad de tiempo hasta que la transmisión tenga éxito.

- Se deberá tener un protocolo apropiado de tal forma que si se necesita retransmitir el mensaje se deberá hacerlo.

- Se dispondrá de un buffer de almacenamiento de transmisión de tal forma que el contenido de este se transmitirá antes de que un nuevo dato pueda ser colocado en éste.

- Deberá ser capaz de soportar diferentes niveles de prioridades para las comunicaciones.

Los prótocolos deberán estar diseñados de tal forma que si se manda un mensaje, todos los controladores estarán enterados de esta situación para evitar conflictos en la transmisión de los mensajes, a este tipo de estructura se le denomina de Múltiple Acceso [5]. Los prótocolos de Múltiple Acceso se divididen en [5]:

a) Base de Argumentos: Los cuales competirán por ganar el acceso al canal, esto opera de tal forma que particiona los controladores de la red formando un conjunto de controladores que tendrán derechos de transmisión y un conjunto de controladores sin derechos de transmisión, los cuales estarán deshabilitados. En caso de que ninguna de las estaciones habilitadas este lista, el canal permanece sin uso y eventualmente una nueva partición podrá ser determinada. En caso de que una estación habilitada este lista, entonces esta transmite el mensaje. Al fin de cada transmisión una nueva partición es determinada. Finalmente si dos o más estaciones habilitadas estan listas, estas estaciones tendrán conflicto con respecto a sus mensajes, si las estaciones pueden detectar y abortar las transmisiones, se aisla una estación de tal forma que solo mande el mensaje una de ellas. En caso de que no se pueda detectar y abortar la transmisión una nueva partición es determinada. Dichas particiones estarán clasificadas como:

a.1) Partición de Tiempo: En este modo se asigna una ventana de tiempo de tal forma que se verifica que estación intenta mandar el mensaje, y si hay más de una se mantendrá un orden determinado previamente asignado, así en la primera ventana, la estación correspondiente mandará su mensaje, dicha ventana se particionará de tal forma que en una segunda ventana creada por la partición de la primera, transmitirá la estación correspondiente.

a.2) Fundamentos Probabilísticos: En este tipo de mecanismo no hay coordinacion entre las estaciones distribuidas, es decir si dos estaciones o más mandan al mismo tiempo su correspondiente mensaje, estos chocarán por lo tanto, cada estación deberá estar programada para retransmitir el mensaje después de una determinada cantidad de tiempo hasta que la transmisión tenga éxito.

a.3) Dirección asignada: Se habilitará primeramente, mediante las direcciones correspondientes a cada uno de los controladores que esten simetricamente colocados en una de las mitades del "árbol", a fin de que solamente una rama del árbol principal este habilitado y así sucesivamente con las subramas de dicha mitad, de tal forma que solamente la mitad del árbol este habilitado, para posteriormente habilitar la otra mitad fig 4.1, [1].

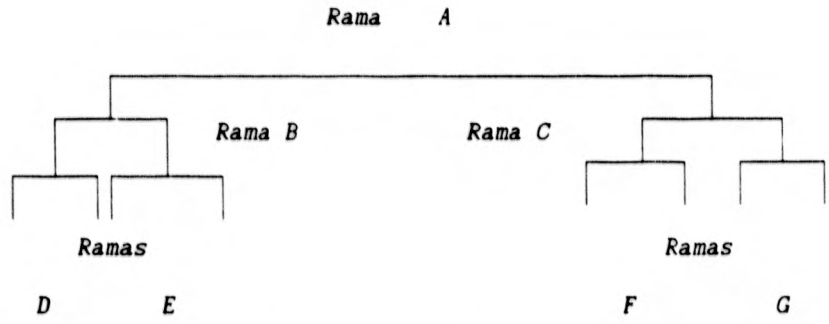



*Fig 4.1 Dirección Asignada*

b) De Acceso Controlado: Los cuales estarán apegados a una determinada asignación, es decir los controladores estarán coordinados de tal forma que nunca intentarán transmitir mensajes simultáneamente. Dicha asignación consiste en imponer un orden en la asignación de derechos de acceso de canal a los controladores, aunque la eficiencia de este tipo de prótocolos se basa en los derechos de transmisión concedidos ó de la opurtunidad de reclamar los derechos de transmisión, esto tendrán que acatar todas las estaciones a pesar de que cualquiera de ellas tenga mensajes que mandar, esta asignación estará basada en :

b.1) Demanda Adaptiva: Esta técnica asigna el canal de una manera más consistente de acuerdo a los requerimientos inmediatos de los controladores. La cual estará regida por dos diferentes aspectos como son: el acceso Reservado y el acceso por Llamada. En este tipo de protocolos, los derechos de acceso de canal son ofrecidos por estaciones acordando algunos accesos ordenados. Este orden puede ser ordenado en prioridad ó puede ser dinamicamente determinado por las estaciones, para esto se utilizan dos mecanismos:

b.1.1) Reservación: Asigna una determinada dirección a cada estación, esta técnica consiste en alterar periódos de post-reservación y transmisión de mensajes.

b.1.2) Símbolos de Pase: En este tipo solamente un controlador tiene un determinado símbolo de pase en cualquier tiempo y por default dicha estación posee los derechos de acceso de canal. De esta forma los controladores tendrán en un momento determinado su símbolo de pase y solamente entonces podrán transmitir.

b.1.3) Método de Emisión de reconocimiento de Acceso: En este método el orden en el cual a las estaciones se les conceden los derechos de transmisión es determinado por el orden número de sus direcciones, es decir la estación 1 inicialmente tiene los derechos de transmisión durante un determinado tiempo, sí la estacion 1 no manda mensaje entonces los derechos de transmisión pasarán a la estación 2. En caso de que comienze a mandar la estación 1 un mensaje la estación 2 aceptará los derechos de transmisión de la estación 1 hasta que dicha estación termine.

b.2) Asignación de Canal Predeterminado: Estos prótocolos se basan en asignación de tiempo, de tal forma que el uso del canal es dividido en intervalos de tiempo de tal forma que cada estación puede usar el canal durante el intervalo determinado, en caso de que el controlador permanesca ocioso un determinado tiempo, es decir no mande ningún mensaje, este tendrá de todos modos asignado un determinado intervalo de tiempo, figura 4.2.

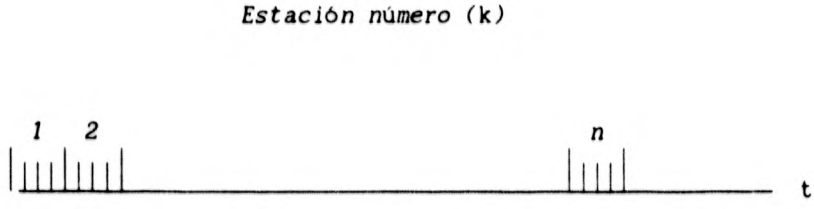



*Figura. 4.2 Asignación de Canal Predeterminado*

Por otra, parte un algoritmo distribuido en el cual los controladores, los cuales son usuarios del canal, particionan a este se denominará canal de prótocolos de acceso. Para esto los controladores deberán mandar sus mensajes ya sea explícita ó implícitamente para coordinar con el resto de los controladores la partición del canal. También deberá tenerse en cuenta que instantáneamente no se podrá conocer el estatus de otras estaciones.

### 4.2.2.- Protocolos de Comunicación Estandard

Basicamente se deben tener características de un sistema de comunicaciones estandard, las cuales permitirán a los usuarios versatilidad en el manejo de la información a transmitir y/ó recibir, las características serán las siguientes: a) Deberán soportar todo tipo de mensajes, de tal forma que pueda existir una transmisión de mensajes

adecuada. b) Los mensajes deberán estar en forma de paquetes o formatos para que al llegar a su destino, cada información dentro del paquete sea identificada por el receptor en forma adecuada, es decir identificar si el mensaje está encriptado, que tipo de bit de paridad usa, que código de destino tiene asignado para que sólo el receptor al cual corresponda dicho código reciba el mensaje, etc,. A continuación daremos las características principales del modelo estandar OSI (Open Systems Interconnections) definida por la Organización de Internacional de Estandares (ISO). Dicho modelo tiene contemplado la posibilidad de conectar diferentes tipos de sistemas, previendo esto se da un modelo estandard que abarque las posibles características que pudiese tener un determinado prótocolo de comunicaciones desde el punto de vista conectividad a una determinada red. Las posibles características son las siguientes [5] Y [7] :

1.- Colocación física: La cual asegura una transmisión correcta del mensaje de tal forma que si se manda un 1 se reciba un 1.

2.- Colocación de datos ligados: Aquí de lo que se trata es de convertir un mensaje dudoso de información correcta en un mensaje confiable, ya sea utilizando la técnica de control de error hacia adelante (forward error control) y de control de error hacia atrás (backward error control), la primera requiere información redundante y la segunda requiere una retransmisión del mensaje completo en el caso de que se detecte un error.

3.- La colocación de red: Esto se refiere a canalizar hacia un determinado destino la información, de acuerdo al código en el paquete de información recibido.

4.- EL Transporte de Colocación: Esto se refiere a proporcionar un determinado tipo de formato de conexión entre transmisión y recepción.

5.- Colocación de sesión: Esto se refiere a proveer un modelo de comunicación entre dos procesos.

6.- Colocación de Presentación: este punto se refiere a la presentación que tendrá el mensaje de comunicación, ya sea texto ó encriptado.

7.- Colocación de la Aplicación: En este punto se pueden aprovechar funciones de alto nivel como bases de datos, correo etc,. [2].

### 4.2.3.- Formatos de Transmisión

Existen principalmente dos formas de transmitir: la asincrona y la síncrona, teniendo la asíncrona la particularidad de transmitir paquetes de tal forma que dentro del mismo formato existan bits que indiquen que parte de la información se refiere al mensaje en sí. Los formatos de transmisión tendrán diferentes características como son las siguientes [7]:

1.- Bit de Paridad: El cual indica si la información es un número par o impar de unos.

2.- Bit de Inicio: Indica cuando comienza la información.

3.- Bit de Paro: Indica cuando termina la información.

4.- Break: Se coloca un retardo de tiempo después del formato con duración de 100 a 600 milisegundos, el cual indicará una interrupción.

5.- Espacio de Paridad: Se utiliza este concepto para completar un octavo bit en formatos de 7 bits, ya que el receptor esta esperando 8 bits, este espacio es representado por nivel bajo.

6.- Marca de Paridad: Igual que en el caso anterior, pero manejado por nivel alto.

7.- Velocidad: Esta indica el número de Bits por segundo a transmitir y/ó recibir (bauds).

Por otra parte la síncrona se refiere a un formato en el cual no existen bits que indiquen cuando inicia la transmisión y cuando termina, esto se hace por medio de contabilización de intervalos de tiempo de tal forma que cada bit de información tendrá una determinada duración, y de esta forma poder conocer la información en el momento preciso.

Dependiendo de las características de transmisión el comienzo de transmisión (handshake), estará definido de acuerdo a dispositivos de hardware ó a prótocolos de software. Dichos prótocolos de software podrán tener formato XON/XOFF y ETX/ACK, que significa respectivamente método de comienzo de transmisión/finalización de transmisión, y método de fin de transmisión/reconocimiento. En el primero se asigna un código para comenzar la transmisión en ASCII(13H) y un código para finalizar la transmisión, en el segundo método se asigna un código ASCII 3 para finalizar la transmisión y un código ASCII 6 para reconocer el mensaje recibido [7] y [8].

## 4.3.- SINCRONIZACIÓN EN LA COMUNICACIÓN DE SISTEMAS MULTIPROCESADORES

Se debe colocar el número de procesos de tal forma que, acorde con el número de procesadores en cuestión, se conozcan perfectamente los tiempos muertos para poder realizar una planeación óptima. Dentro de la colocación de los procesos se deberá tener en cuenta la repartición de los procesos, de tal modo que dichos procesos se distribuirán de cierta manera que consuman un tiempo si no igual por lo menos similar para poder entregar datos en forma periódica al ó a los siguientes procesadores

cuando dichos procesos son PERIODICOS. Cuando los procesos son APERIODICOS es decir, no tienen la misma periocidad, se distribuyen de tal forma que cada proceso arrive a un nodo, y este checa de acuerdo a lo que tiene cargado si lo puede planear, de no ser así lo manda a otro nodo. En los sistemas distribuidos se tiene que tomar en cuenta que todos los nodos del sistema deberán tener códigos de los diferentes procesos ya que cualquiera de ellos podrá en un momento dado recibir un determinado proceso pensando en el peor caso [3].

Por otra parte, se debe tener en cuenta que una secuencia de declaraciones que deberán aparecer juntas, es decir serán indivisibles para ejecutar una determinada operación, será llamada sección crítica, la sincronización requerida para proteger una sección crítica es conocida como exclusión mutua. La condición de sincronización es un requerimiento, el cual es necesario cuando un proceso desea desempeñar una operación que puede solamente sensibilizar ó asegurar, será desempeñada si otro proceso tuvo que tomar la misma acción ó esta en algún estado definido dicha operación, de lo contrario solamente será sensibilizada. Para implementar la sincronización la condición determinada será leída a través de una bandera y si todavía no esta listo el proceso, se estará en un "loop", a esto le llamaremos "busy waiting" (ocupado), otro elemento importante dentro de la programación exclusiva mutua y condición de sincronizaciones es el semáforo el cual tiene los siguientes beneficios [5]:

1.- La simplificación de prótocolos de transmisión

2.- Eliminan la necesidad de loops de "busy waiting".

Un semáforo es una variable entera no negativa que desde la inicialización puede actuar sobre dos procedimientos o más, actuando sobre tres tipos de sincronización [5]:

1.- Asíncrona : Cuando el mensaje se manda a pesar de que sea recibido o no.

2.- Síncrona : Cuando el mensaje sólo se manda si se tiene la seguridad de que será recibido.

3.- Invocación Remota: EL transmisor procede a mandar el mensaje solo cuando se lo pide el receptor [2].

## 4.4.- TÈCNICAS DE CONTROL DISTRIBUIDO

Este subtema trata sobre las diferentes maneras de resolver la problemática que existe en la interacción de los módulos que constituyen el sistema distribuido incluyendo el procesador central. Ya que como habíamos mencionado en párrafos anteriores, es necesaria la comunicación entre módulos debido a diferentes causas como son [10]:

- La falla de un determinado módulo delegando sus funciones entre los restantes.

- La necesidad de comunicarse con otros módulos debido a la utilización de variables en común.

- Cuando los algoritmos por alguna razón son modificados.

A continuación mencionaremos las técnicas más usuales en el manejo de un control distribuido así como sus principales características, haciendo enfasis en el método más adecuado para el desarrollo de este trabajo [9]:

a) Teoría de sistemas Jerárquicos: Este tipo de técnica consiste en buscar la descomposición del problema de control original en estructuras jerárquicas multinivel acorde a las características particulares de cada sistema.

Al tener el control jerárquico perfectamente establecido de cada nivel se tendrá el control del sistema en total. Cada nivel estará constituido por un controlador local con su respectivo controlador-coordinador. Donde el controlador-coordinador gobierna la coordinación de variables *bi* basado en el desempeño de las variables *Ji (bi)* fig.4.3, ya que estas proporcionan al controlador-coordinador la información sobre el desempeño de cada controlador local. El controlador local maneja la variable *ui* de control basado en el estado *Xi*, la coordinación de variables *bi*, y la interacción de variables *mi* como lo muestra la figura 4.3. El desempeño del sistema total estará definido por el desempeño de las estructuras jerárquicas en que dicho sistema fue descompuesto, la interacción de entrada, la cual representa la interacción entre sistemas, es generalmente pequeña entre sistemas excepto para sistemas altamente acoplados [3] Y [10].

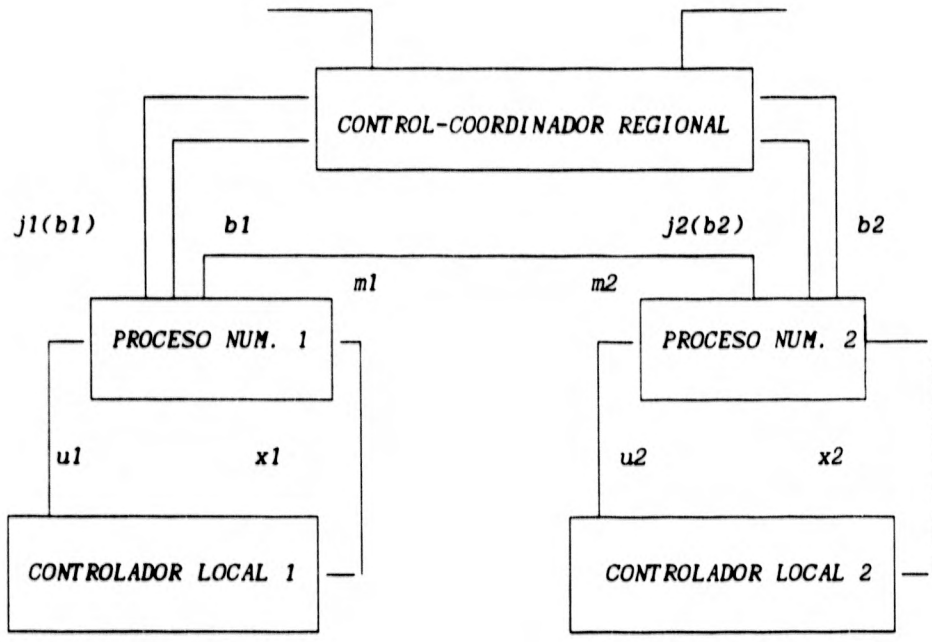



*Figura 4.3 Sistema Jerárquico de dos niveles,* [10]

Al descomponer el sistema en diferentes niveles jerárquicos se tendrá un mecanismo coordinador de dichos niveles. Estos niveles se formarán de acuerdo a los siguientes aspectos [10]:

a.1) Estructura: Se hace una partición del sistema en subsistemas cada uno con una meta individual y con interacción con los otros subsistemas.

a.2) Niveles de Influencia: Para hacer este tipo de partición se tomará en cuenta que cada subsistema tendrá diferentes objetivos o tareas así que la interacción entre ellos es mínima, además de dividirlos por estratos teniendo en cuenta que un estrato alto tendrá mayor prioridad, entendiendosé por estrato el grado de importancia de cada subsistema dentro del sistema.

a.3) Niveles de Control: El sistema se particiona de acuerdo a los diferentes niveles de control, donde cada módulo tiene un determinado objetivo dentro del sistema. Por otra, parte dentro de la partición del sistema se tendrá en cuenta los aspectos anteriores además de considerar un modelo de coordinación ó un modelo de objetivo.

a.4) Modelo de Coordinación: Este modelo considera un sistema *P* con un vector de variables de entrada *m* y un vector de variables de salida *y* compuesto por dos subsistemas donde un determinado par de vectores de variables reflejan la interacción de variables de un subsistema a otro *x1* ó viceversa *x2*, teniendo cada subsistema su respectivo vector de variables de entrada *mi* y su respectiva vector de variables de salida *yi* como lo muestra la figura 4.4. Donde *P* es un sistema estático governado por la ecuación

$$G(m, y, x)=0 \qquad (4.1)$$

En donde la función criterio será descrita por la ecuación

$$P(m, y, x) = P1\ (m1, y1, x1) + P2\ (m2, y2, x2) \qquad (4.2)$$

De tal forma que $z = x$ (4.2a) donde de x es un vector que minimiza la función criterio y la minimización estará descrita por la ecuación $H(z)= \min_{m,y} P(m, y, z)$ (4.2b) sujeta a $G(m, y, z)=0$ (4.2c) donde

$\min_z H(z)$ sobre todos los valores aprovechables.

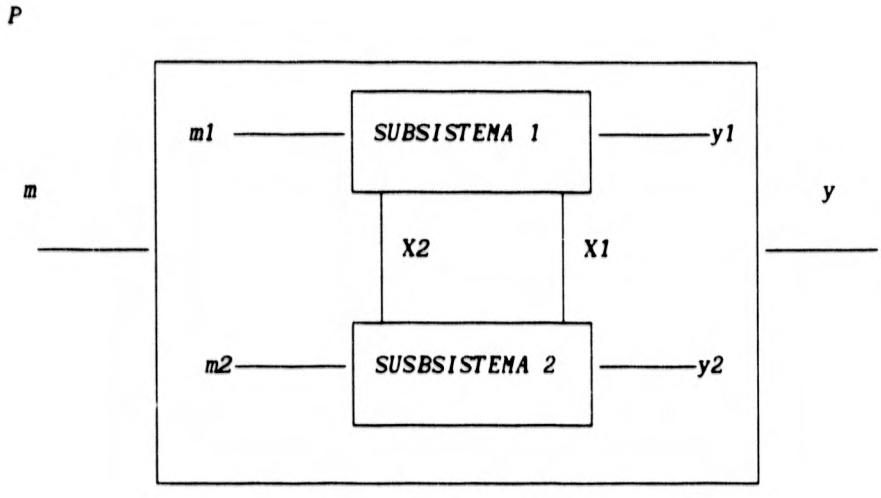



*Fig.4.4 Representación de un Modelo de Coordinación,* [10]

a.5) Modelo de meta: Tiene una estructura similar al anterior donde se tiene que tomar en cuenta que los subsistemas no se pueden separar y tratarlos independientemente si no cumplen ciertas condiciones de optimización. Donde esta optimización dependerá del llamado principio de balance de interacción, dicha estructura es apreciada en la figura 4.5.

Este modelo tiene gran similitud con el anterior, la diferencia radica en que los vectores de las variables *xi* y *wi* no interactúan directamente entre los subsistemas siendo estas ahora cuatro, las cuales representan las "posibles entradas y las posibles salidas" a los subsistemas. Donde dichas variables serán seleccionadas de acuerdo al principio de Interacción de Balance, el cual dice que *xi* y *wi* actúan por igual, donde la función criterio estará determinada por

$$P(m,y,x,w,\lambda) = P1\ (m1,y1,x1) + P2\ (m2,y2,x2) + \lambda(x-z) \qquad (4.3)$$

donde $\lambda$ es un vector que le da peso al factor $x-z$ el cual causará el desbalance, el sistema estará governado por las ecuaciones:

$$G1(m1,y1,x1,z2)=0 \text{ ec. } 4.4 \qquad G2(m2,y2,x2,z1)=0 \qquad (4.5)$$

La solución del sistema se realiza en forma similar al método anterior tomando en cuenta que tendremos dos sistemas independientes.

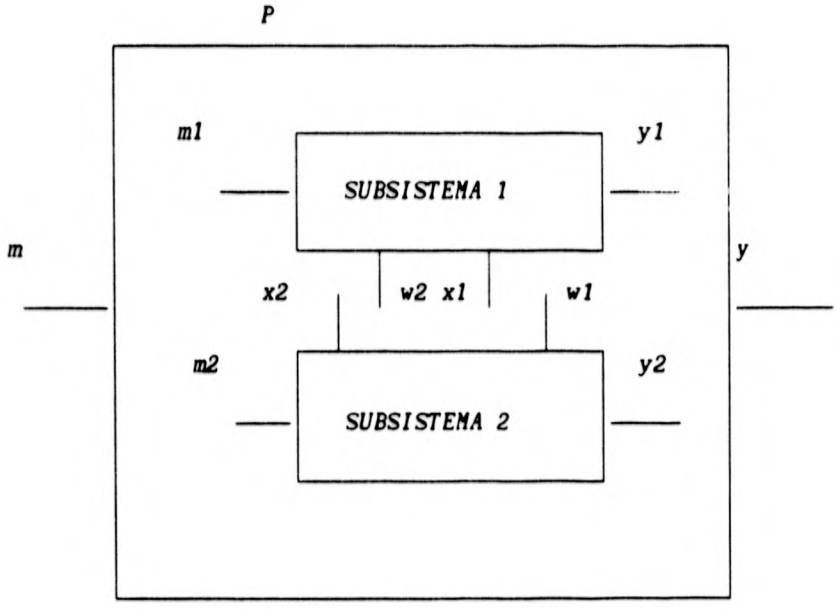



*Fig. 4.5 Representación de un Modelo de Meta,* [10]

b) Programación Dinámica de Aproximaciones Sucesivas: Dicho método consiste en calcular en determinados instantes de tiempo las variables involucradas en los subsistemas teniendo algoritmos de aproximaciones sucesivas para la solución del problema.

c) Técnicas de Redes: Aquí se manejan como nodos siguiendo trayectorias definidas entre ellos .

d) Método de Perturbación: Este método consiste en definir en diferentes escalas de tiempo el modelo en variables de estado que represente cada uno de nuestros susbsistemas.

e) Coordinación Periódica: En este tipo de técnica cada subsistema es representado por sistemas de variables de estado, donde el objetivo es controlar dichos susbsistemas periódicamente, debido a la interacción entre ellos y evitando señales de error.

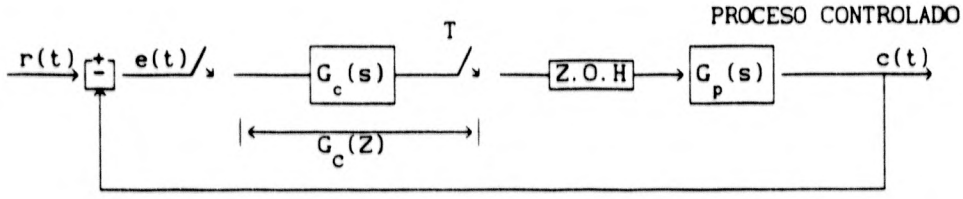
## 4.5. - CONCLUSIONES

La comunicación entre los controladores externos y el monitor central así como la existente entre dichos controladores debe ser de tal modo que permita una interconexión, donde cada controlador tenga un determinado código que permita su identificación. Todos los controladores deberán tener un determinado protocolo de comunicación así como una determinada prioridad de tal forma que se pueda utilizar una de las técnicas de comunicación e implementar el arreglo correspondiente. Además de que el tiempo de procesamiento de cada controlador deberá ser aproximadamente el mismo para tener balanceado el sistema, es decir que en caso de falla un controlador en un determinado momento pueda suplir la función de otro.

# 5.- CONTROL DIFUSO

## 5.1.- INTRODUCCIÓN

Con respecto a este punto en particular debemos considerar que tipos de procesos vamos a controlar de tal forma que utilicemos los algoritmos de control más adecuados. Es decir que nos permitan obtener resultados con una precisión satisfactoria, con el menor tiempo cálculo posible, así como una mínima cantidad de memoria, tomando en cuenta que deberán ser en tiempo real, y su implementación de bajo costo. Existen diferentes tipos de algoritmos de control de los cuales mencionaremos y expondremos las características más importantes, conociendo sus ventajas y desventajas y así estar en posición de escoger el o los algoritmos adecuados acordes con el o los procesos a controlar. Dentro de este aspecto se puede tener una primera clasificación de los algorimtos de control como de lazo abierto o de lazo cerrado dependiendo de la estructura de control que se tenga así como del proceso a controlar. Otra clasificación se puede tener dependiendo del modelo matemático del controlador, ya sea si es un modelo continuo ó un modelo discreto. La siguiente clasificación se puede hacer independientemente si se trata de un modelo discreto ó continuo, es decir se basará en la estructura de control utilizada, como pueden ser controladores proporcionales, integrales, derivativos, adaptables, robustos, autosintonizables, difusos etc. Cada controlador puede tener más de una característica y son utilizados para resolver determinadas tareas, acorde con su diseño, teniendo en cuenta que desde el punto de vista mátematico podrán ser representados por ecuaciones diferenciales para el caso de un sistema continuo y por ecuaciones en diferencias en el caso de un sistema discreto, cayendo cualquier estrategia de control convencional en cualquiera de los modelos a excepción del controlador difuso que no es representado por un modelo matemático convecional si no por reglas de inferencia, aunque hay que hacer notar que ya teniendo dichas reglas de inferencia se pueden transladar a un modelo discreto para efectos de analisis de comportamiento del sistema. Con lo anterior podemos mencionar que un modelo convencional (no difuso) tendría la ventaja de tener una precisión adecuada ya que conoceríamos el modelo matemático del proceso a controlar y la estructura matemática del controlador. Teniendo la desventaja de que se necesita conocer el modelo matemático del proceso a controlar. Por otra parte, un controlador difuso tiene la ventaja de no necesitar conocer el modelo matemático que describa el comportamiento del proceso, aunque tiene la desventaja de necesitar generar en ocasiones un número grande de reglas de inferencia que simulen el comportamiento del proceso en una forma lo más precisa posible. Teniendo en cuenta que entre más reglas existan se necesitará más tiempo de cálculo y más localidades de memoria. A continuación describiremos mediante la figura 5.1 un modelo discreto ya que dicha estructura servirá de base para el diseño tanto del controlador difuso como del controlador on-off utilizados en esta tesis, describiendo el diseño de los controladores más adelante en este cápitulo.




*Control digital con Controlador de lazo en cascada* [22]

*Figura 5.1*

En la figura 5.1 se aprecian tanto la salida como la entrada que son funciones continuas r(t) y c(t) teniendo la salida retroalimentada a la entrada generando una señal de error e(t) la cual será la diferencia entre la señal de entrada y la salida retroalimentada, teniendo el controlador digital representado por una función discreta $G_c(z)$ conectado directamente al proceso $G_p(S)$ a controlar siendo este una función continua [22]. Este esquema en forma básica con las modificaciones adecuadas tendrán las estructuras de control usadas en esta tesis. Tal vez sean los algoritmos de control más sencillos que permiten el tener activado o desactivado un determinado actuador mediante un determinado rango de lectura para una variable. Teniendo en cuenta dichos umbrales mediante sensores, que permitan el mantener en dicho nivel el actuador a través del controlador. Podríamos decir que si la variable sensada se encuentra en un determinado rango se activa el actuador y si no, no se activa el actuador.

## 5.2.- ALGORITMOS DE CONTROL DIFUSO

### 5.2.1.- Descripción de Lógica Difusa

Este tipo de algoritmos estan basados en la logica difusa (fuzzy logic)la cual nace en los Estados Unidos al rededor de 1965, en la Universidad de California, en Berkeley, creada por el profesor **Lofti A. Zadeh**, han sido aplicados en diferentes áreas, como mencionamos más tarde [13]. La lógica difusa llega después de la lógica multivaluada la cual propone 3 niveles lógicos: verdadero (1), falso (0) y neutro (1/2), el cual representa la mitad verdadera o la mitad falsa. Propuesta por Jon Lukasieviicz (1930). La lógica difusa abarca una infinidad de valores los cuales estarán comprendidos entre o (completamente falso) y 1 (completamente verdadero) [20].

Cabe mencionar que para ubicar la lógica difusa en las aplicaciones de la computadora deberemos primero dividir estas aplicaciones en tres grandes áreas como son las siguientes [14]:

a) **Propósitos Numéricos:** Donde se aplican cálculos y análisis números, como solución de problemas matemáticos referidos a diferentes áreas del conocimiento humano.

b) **Propósitos de Bases de Datos:** Donde se almacenan grandes cantidades de información.

c) **Ingeniería de Conocimiento:** Donde se encuentran los sistemas expertos, siendo esta la área más nueva.

Precisamente la lógica difusa tiene aplicaciones tanto en sistemas expertos, como en sistemas de control, aplicaciones en el área número ica, y la de Ingeniería del Conocimiento. Aunque se puede tener otras aplicaciones como se mencionará posteriormente. La primera aplicación industrial en esta área se llevó a cabo en 1977 a la industria del cemento en Dinamarca [24]. Cabe mencionar que los problemas con modelos estructurales son los que usan los valores falso o verdadero para representar vínculos. Por otra parte, en la lógica predictiva las reglas de subjetividad u objetividad, así como los conocimientos que se deberán tener para simular la experiencia adquirida por aprendizaje de un experto para resolver un determinado problema son expresados por sentencias o un conjunto de ellas. Las proposiciones son modelos escritos, así que podemos representar valores ambiguos. En términos generales **los sistemas difusos son convenientes cuando existe incertidumbre o razonamiento aproximado.** Es decir cuando los modelos matemáticos que describen el comportamiento del sistema son demasiado complejos para modelar [20].

### 5.2.2.- Características Principales de la Lógica Difusa.

Dentro de las principales párametros que caracterizen a la lógica difusa podemos mencionar los siguientes:

- Los métodos convencionales son buenos para resolver problemas simples en cuanto a su modelado, mientras que los sistemas basados en lógica difusa son convenientes para problemas complejos o aplicaciones que involucren **descripciones humanas** ó **pesamientos intuitivos**.

- Capacidad de Aprendizaje, donde se añaden ciertos datos para configurar un mejor control.

- Esta compuesta por Reglas de Inferencia las cuales, están hechas en base a un determinado conocimiento del problema y a la experiencia para resolverlo.

De esta forma la lógica difusa, maneja conceptos ambiguos como caliente, muy caliente, viejo, joven etc. llamadas variables linguísticas. Es decir el grado de ocurrencia, no tomando para este fin el paso del tiempo o laspruebas realizadas. La teoría de la probabilidad mide la **probabilidad de ocurrencia** ó **de no ocurrencia** de un evento en tanto que la lógica difusa mide el **grado de ocurrencia** de un evento o de una determinada condición. O de forma sistematizada, podremos decir que la teoría de Probabilidad esta basada en un razonamiento exacto, por manipulación simbólica y cálculos númericos y esto a su vez por predicciones. Mientras que la Lógica Difusa está basada en razonamientos aproximados por manipulaciones simbólicas y cálculos númericos y esto a su vez por condiciones ambigüas. De esta forma podemos formar una estructura constituida por varios pasos para desarrollar un sistema difuso, como el siguiente [13],[14],[20], ejemplo:

- Determinar si la estructura del problema a resolver amerita el uso de lógica difusa.

- Si es así, determinar el rango en donde se manejan las variables de entrada y de salida.

- Definir las funciones que constituyen las salidas y las entradas, siendo estas las posibles opciones que tomarán las entradas y salidas.

- Construir las reglas que relacionarán las entradas y salidas.

- La determinación de dichas reglas requiere de pruebas exahustivas para poder verificar su correcto funcionamiento.

- Los sistemas de control difuso en términos generales son estables, aunque la estabilidad de las reglas nos dan una estabilidad parcial con respecto a cada regla, para que el sistema sea estable, todas sus reglas en conjunto deberán ser estables.

### 5.2.3.- Aplicaciones de la Lógica Difusa

Debido a la gran aceptación en determinadas áreas que presentan las características idóneas para aplicar la lógica difusa, podemos mencionar que el número aproximado de aplicaciones industriales hasta 1993 será de 3000 solamente.En términos generales la lógica difusa o bién encuentra una aplicación en muchas áreas o bién esta por aplicarse. Dichas aplicaciones son tan amplias que pueden ir desde compresión de lenguaje, interpretación de información humana, relaciones humanas, planeación, toma de decisiones hasta elcontrol de robots, reconocimiento de patrones, etc [24].

## 5.2.4.- Fundamentos Matemáticos de la Lógica Difusa

Comenzaremos describiendo los elementos involucrados en la teoría de conjuntos convencionales y posteriormente los de la teoría de conjuntos difusos [12] y [13].

### - Conjuntos Convencionales -

Con respecto a los conjuntos convencionales se puede mencionar que los valores ó están dentro del conjunto ó no lo están, de acuerdo al siguiente modelo [12].

$$X(x) = \begin{cases} 1; & x \in A \\ 0; & x \notin A \end{cases} \qquad (5.2)$$

Como ejemplo, si hablamos de divisiones que caractericen la edad bajo este esquema, el conjunto edad puede ser dividida en por ejemplo 3 subconjuntos [12].

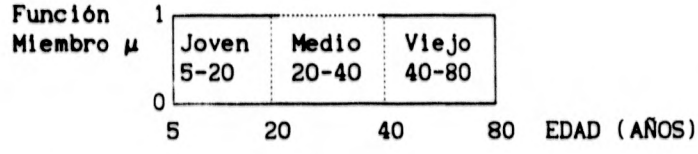



*Fig. 5.2*

A dichas divisiones se les llamará conjuntos crisp. En donde cada conjunto crisp contiene un determinado rango de valores, teniendo en común un solo valor.

En donde se pueden aplicar las operaciones básicas y para cada una de ellas se tendrá su representación gráfica, que en forma general sería la siguiente [12]:

Unión: $\{x/x\varepsilon A \text{ or } x\varepsilon B\} = A \cup B$

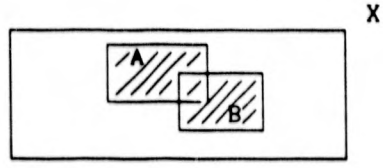



*Fig. 5.3*

Intersección: $\{x/x\varepsilon A \text{ and } x\varepsilon B\} = A \cap B$

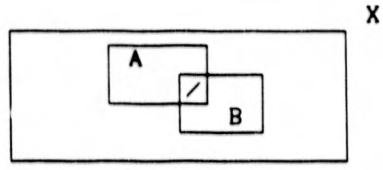



*Fig. 5.4*

Complemento: $\{x/x \notin A,\ x\varepsilon X\} = \bar{A}$

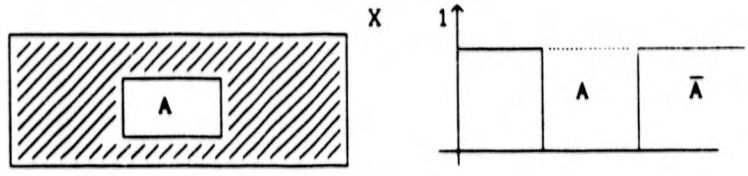



*Fig. 5.5*

Diferencia: $A/B = \{x/x\varepsilon A \text{ and } x \notin B\}$

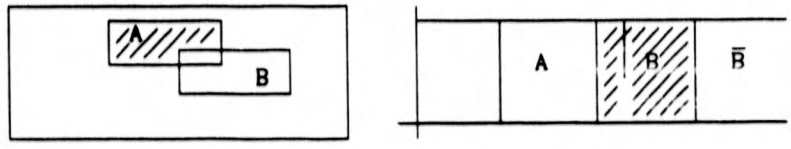



*Fig. 5.6*

Donde se tienen las siguientes propiedades:

Conmutatividad: $A \cup B = B \cup A$
$A \cap B = B \cap A$ ..... (5.3)

Asociatividad: $A \cup (B \cup C) = (A \cup B) \cup C$
$A \cap (B \cap C) = (A \cap B) \cap C$ ..... (5.4)

Distribuitividad: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ .... (5.5)

Idempotencia: $A \cup A = A$
$A \cap A = A$ ..... (5.6)

Identidad: $A \cup \phi = A$
$A \cap X = A$
$A \cap \phi = \phi$
$A \cup X = X$ ..... (5.7)

Transitividad: Si $A \subseteq B \subseteq C$, entonces $A \subseteq C$

Involución: $\bar{\bar{A}} = A$ ..... (5.8)

Ley de la exclusión media:

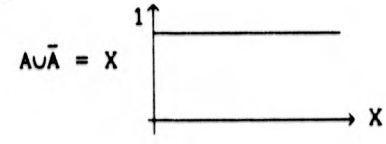
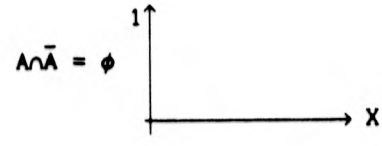



*Fig. 5.7*

Ley de la Contradicción:

*Fig. 5.8*



Leyes de Morgan: $(\overline{A \cap B}) = \bar{A} \cup \bar{B} = \{x/x \notin A \text{ or } x \notin B\}$

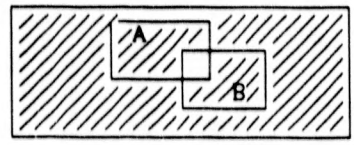



*Fig. 5.9*

$(\overline{A \cup B}) = \bar{A} \cap \bar{B} = \{x/x \notin A \text{ and } x \notin B\}$

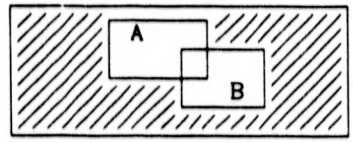



*Fig. 5.10*

Para ver la relación entre dos conjuntos crisp F y G, la unión y la intersección será representada por:

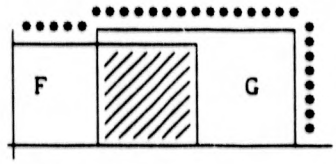



*Fig. 5.11*

* Significa unión
//// Significa intersección

Complemento de F

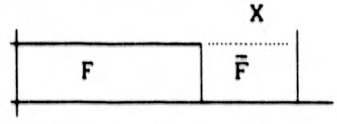



*Fig. 5.12*

Definiendo el criterio máximo y el criterio mínimo por las siguientes ecuaciones (los cuales se verán en forma gráfica en un subtema posterior para conjuntos difusos) :

$$A \cup B = X_{A \cup B}(x) = X_A(x) \cup X_B(x) = \max (X_A(x), X_B(x)) \qquad (5.9.a)$$

$$A \cap B = X_{A \cap B}(x) = X_A(x) \cap X_B(x) = \min (X_A(x), X_B(x)) \qquad (5.9.b)$$

- Conjuntos Difusos-

Su representación estará dada por medio de la siguiente expresión, donde podrán existir una infinidad de valores entre 0 y 1, [12]:

$$\mu_x(x) \in [0,1] \qquad (5.10)$$

Los conjuntos difusos tendrán los siguientes elementos [12]:

- Función Miembro (membersphip function). Es el grado en el cual el elemento $\mu_A(x)$ pertenece al subset A, el cual estará dividido en varios atributos (Fuzzy sets).

$$\mu_A(x) = \text{grado } (x \in A)..... \qquad \text{ec.5.11}$$

-Conjunto (set) X, es el universo, el cual representa todos los valores posibles del sistema.

-Subconjunto (subset) A son valores que se encuentran dentro del conjunto X.

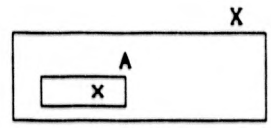



*Fig.5.13*

Donde $\mu$: X → [0,1] podrá tener valores entre 0 y 1

Si hablamos de divisones que caracterizen por ejemplo la edad, como en el caso anterior, el subconjunto edad puede ser dividido en 3 subconjuntos, teniendo estos subconjuntos la forma triangular, trapezoidal ó bien de campana (mostrando la trapezoidal y la triangular) [20].

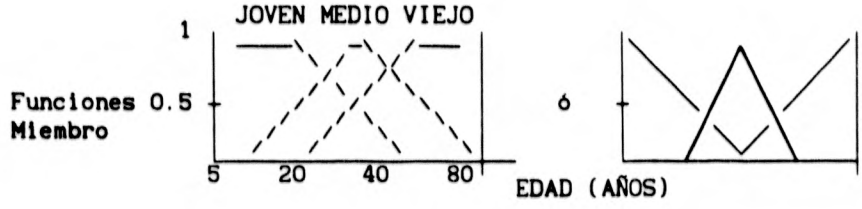



*Fig.5.14*

A dichas divisiones se les llamará conjuntos difusos. Por otra parte definiendo convexo a un conjunto difuso, será aquel que tendrá que cumplir lo siguiente [29]:

$$\mu_A(X) \geq \min [\mu_A(X), \mu_A(Z)] \ldots\ldots\ldots (5.12)$$

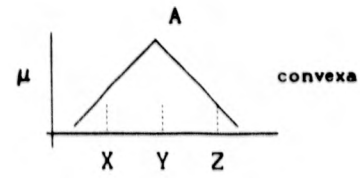



*Fig. 5. 15*

De lo contrario será no convexo:

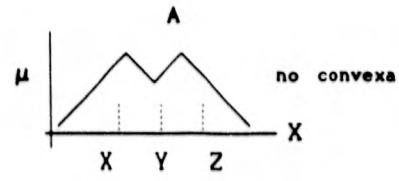



*Fig. 5. 16*

Para ejemplificar la relación de los conjuntos difusos, se tienen los conjuntos difusos A, B, C, cuyas operaciones básicas estarán dadas de la siguiente forma [13]:

Unión:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) \ldots\ldots \quad (5.13)$$


*Fig. 5. 17*

Intersección:

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) \ldots\ldots \quad (5.14)$$


*Fig. 5.18*

Complemento:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \ldots\ldots\ldots (5.15)$$

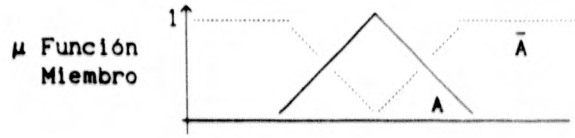



*Fig. 5.19*

Diferencia:

$$\mu_{A\cap\bar{B}}(x) = \mu_A(x) \wedge \mu_{\bar{B}}(x) \ldots\ldots\ldots (5.16)$$

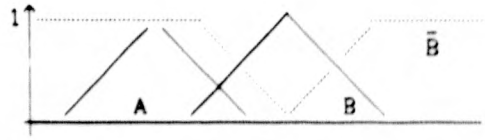



*Fig. 5.20*

Donde se tienen las siguientes propiedades:

Conmutatividad:

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) = \mu_{B\cup A}(x) = \mu_B(x) \vee \mu_A(x) \ldots (5.17.a)$$

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \mu_{B\cap A}(x) = \mu_B(x) \wedge \mu_A(x) \ldots (5.17.b)$$

Las propiedades de Asociatividad, Distributividad, Idempotencia, Identidad, Transmitividad e Involución son exactamente igual para los conjuntos crisp que para los difusos.

Intersección:

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) \ldots\ldots \quad (5.14)$$

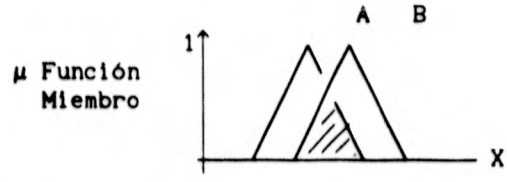



*Fig. 5.18*

Complemento:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \ldots\ldots\ldots (5.15)$$

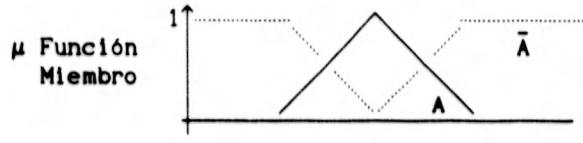



*Fig. 5.19*

Diferencia:

$$\mu_{A\cap\bar{B}}(x) = \mu_A(x) \wedge \mu_{\bar{B}}(x) \ldots\ldots\ldots (5.16)$$

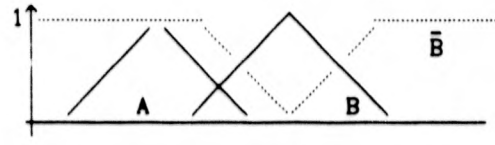



*Fig. 5.20*

Donde se tienen las siguientes propiedades:

Conmutatividad:

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) = \mu_{B\cup A}(x) = \mu_B(x) \vee \mu_A(x) \ldots (5.17.a)$$

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \mu_{B\cap A}(x) = \mu_B(x) \wedge \mu_A(x) \ldots (5.17.b)$$

Las propiedades de Asociatividad, Distributividad, Idempotencia, Identidad, Transmitividad e Involución son exactamente igual para los conjuntos crisp que para los difusos.

Ley de la exclusión media:

$$A \cup \bar{A} \neq X \quad \ldots\ldots\ldots\ldots (5.18)$$

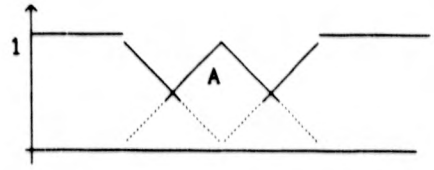



*Fig. 5.21*

Ley de la contradicción:

$$A \cap \bar{A} \neq \phi \quad \ldots\ldots\ldots\ldots (5.13)$$

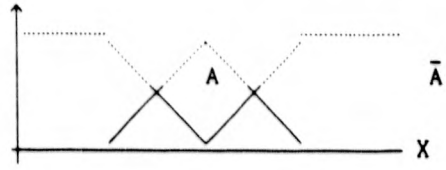



*Fig. 5.22*

Ley de Morgan: $\mu_{\overline{A \cap B}}(x) = \mu_{\bar{A} \cup \bar{B}}(x) = \mu_{\bar{A}}(x) \vee \mu_{\bar{B}}(x) \ldots (5.20)$

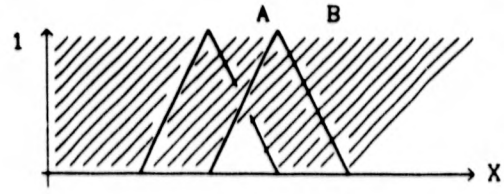



*Fig. 5.23.*

$$\mu_{\overline{A \cup B}}(x) = \mu_{\bar{A} \cap \bar{B}}(x) = \mu_{\bar{A}}(x) \wedge \mu_{\bar{B}}(x) \ldots\ldots\ldots (5.21)$$

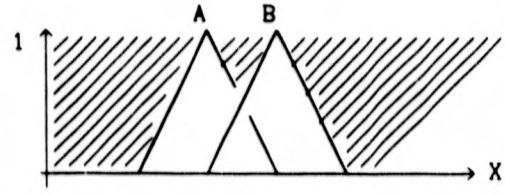



*Fig. 5.24.*

Definiendo $\mu_{A\cup B}(x) \triangleq \mu_A(x) \cup \mu_B(x) = \max(\mu_A(x), \mu_B(x))$ .....(5.22.a)

$$\mu_{A\cap B}(x) \triangleq \mu_A(x) \cap \mu_B(x) = \min(\mu_A(x), \mu_B(x)) \text{.....(5.22.b)}$$

5.2.4.1.- Elementos de una Función Miembro

De acuerdo a la siguiente figura se tendrán los siguientes elementos[12]:

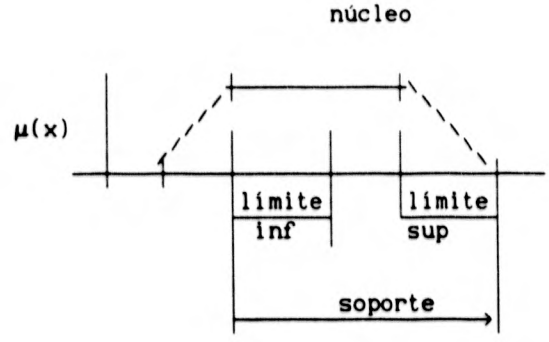



*Fig. 5.25*

Donde el subconjunto A del conjunto X estará definido por:

$$A = \frac{\mu_A(x)}{X_1} + \frac{\mu_A(x)}{X_2} + \ldots = \sum_{i=1}^{n} \frac{\mu_A(x)}{X_i} = \int_x \frac{\mu_A(x)}{X}$$

.....(5.23)

Aquí los elementos de soporte están en el numerador y el grado en el denominador. Usando esta última ecuación podemos expresar la unión, la intersección y el complemento de la siguiente forma [12]:

$$A\cup B = \int_x (\mu_A(X) \vee \mu_B(X))/X \text{ .....(5.24)}$$

$$A\cap B = \int_x (\mu_A(X) \wedge \mu_B(X))/X \text{.....(5.25)}$$

$$\bar{A} = \int_x (1-\mu_A(X))/X \text{.....(5.26)}$$

### 5.2.4.2.- Variables Linguísticas y Razonamiento Aproximado

Una variable lingüística es el nombre que se le da a una característica que representa "algún" conocimiento sobre "algo". Y esta variable no es otra cosa que un subconjunto del conjunto X universal, donde a dicha variable se le podrán cargar ciertos atributos, que no serán otra cosa que los conjuntos difusos, donde las funciones miembro indican $\mu(X)$, que es el grado de pertenencia a cada uno de los conjuntos difusos involucrados [14]. Una variable linguística podrá ser caracterizada por una función con 5 elementos (X, T(X), U,G,M) en donde X es el nombre de la variable; T(X) es la característica o el término set de X, que es el conjunto de nombres de valores linguísticos o características, los cuales describirán algún conocimiento de "algo", siendo cada uno de ellos un conjunto difuso, definidos en el universo U. Donde G es una regla sintáctica para generar nombres de valores de X y M en una regla semántica para asociar con cada valor el significado. De tal forma que M(X) denota un subconjunto de U. Por ejemplo velocidad es interpretada como una variable linguística donde estarán los conjuntos difusos involucrados con esta variable T(velocidad) = {despacio, moderado, rápido, muy despacio etc} donde el universo estará definido por U=[0, 100] definiendo los límites para cada conjunto difuso como sigue [14]:

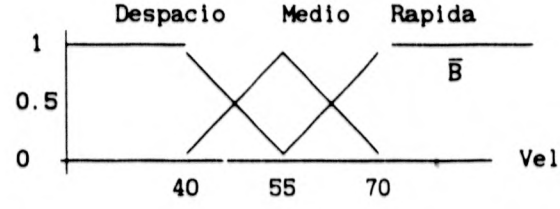



*Fig. 5.26.*

mencionaremos otros ejemplos como pueden ver

T(edad) = joven + no joven + muy joven + no muy joven
+ muy muy joven + ... + viejo + no viejo + muy viejo
+ no muy viejo + .... + no muy joven "y" no muy viejo
+ ... + edad media + no edad media + ... + no viejo y no
edad media + ... + extremadamente viejo + viejo "o" joven + ...
etc. ec.5.27.

T(apariencia) = bonita + atractiva + fea + ... + más o menos
bonita + horrible + ... + etc, etc, etc..... ec.5.28

Por otra parte, es conveniente ejemplificar el concepto de característica mencionado anteriormente, esto puede hacerse de la siguiente forma: supongamos que hablamos de las carácteristicas más comunes del hombre que quedarían conformadas de la siguiente forma:

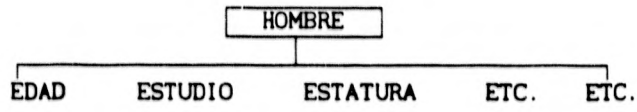



*Fig. 5.27*

Dichas características serán los conjuntos difusosdel conjunto hombre. Por otra parte, la estructura jerárquica de una variable lingüística se visualiza a continuación,se puede apreciar que una subcaracterística puede pertenecer a más de una característica fig.5.28:

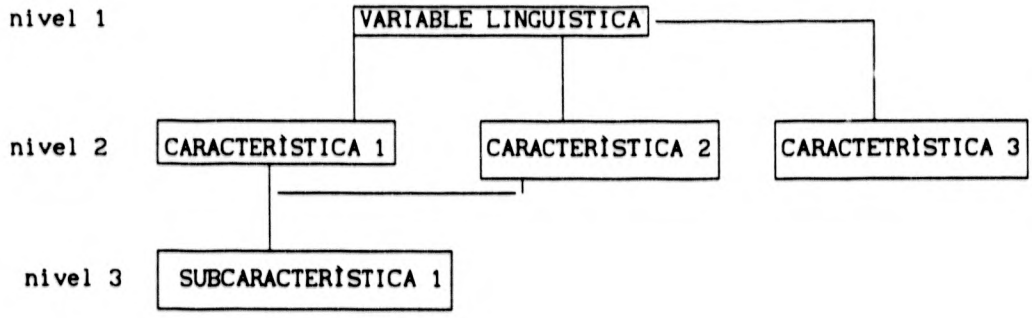



*Fig. 5.28*

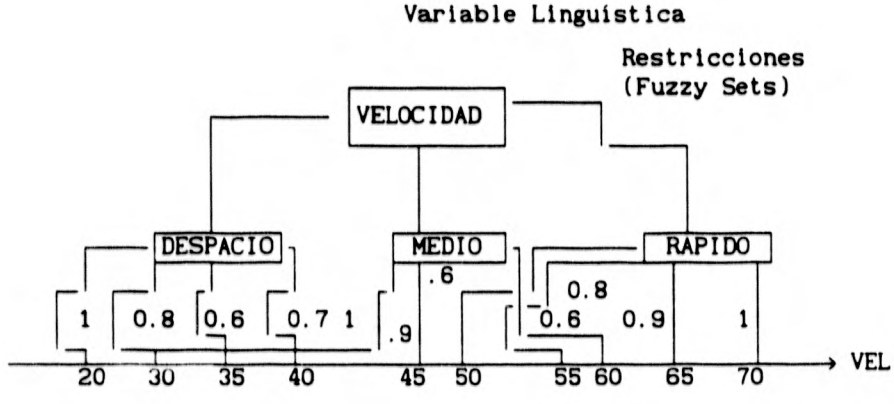



*Fig. 5.29*

En la figura 5.29 se puede apreciar que hay valores de velocidad que estan en más de un solo subconjunto difuso con diferentes grados de pertenencia dentro del subconjunto difuso correspondiente (funciones miembro). Con referencia a esto la función de compatibilidad asocia cada valor de la variable en el intervalo entre [0,1] que es el rango de las funciones miembro, por ejemplo las de la siguiente figura [14]:

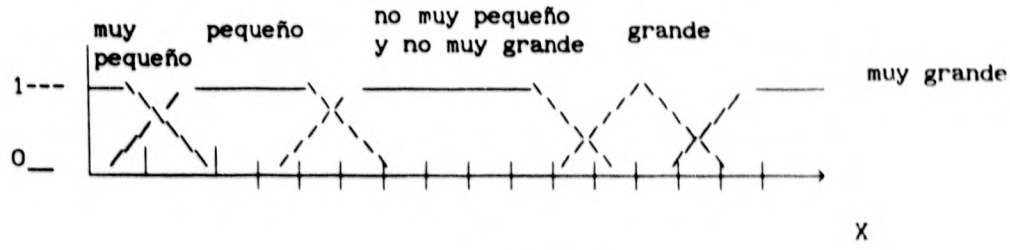



*Fig. 5.30*

Donde cada conjunto difuso tiene un rango en X.
Si tenemos un conjunto difuso A, $\mu_A$: U $\longrightarrow$ [0,1] el cual asocia con cada elemento de y de U un número $\mu_A(y)$ en el intervalo [0,1], el cual representa el grado de pertenencia de y en A. donde como se habia definido:

$$A = \int_U \mu_A(y)/y \ldots\ldots (5.29) \qquad \text{ó}$$

$$A = \mu_1/y_1 + \ldots + \mu_n/y_n; \quad A = \sum_{i=1}^{n} \mu_i/y_i \ldots\ldots\ldots (5.30)$$

en donde $\mu_i$ i = 1,...,n es el grado de pertenencia de $y_i$ en A en

donde $U = y_1 + y_2 + \ldots + y_n$ o $U = \sum_{i=1}^{n} y_i$ ..... (5.31)

ó también $U = 1/y_1 + 1/y_2 + \ldots + 1/y_n$ o $U = \sum_{i=1}^{n} 1/y_i \ldots$ (5.32)

De este modo para los números reales a, b podemos definir

$$a \vee b = \max(a,b) \triangleq \begin{cases} a, & \text{si } a \geq b \\ b, & \text{si } a < b \end{cases} \quad \ldots (5.33)$$

$$a \wedge b = \min(a,b) \triangleq \begin{cases} a, & \text{si } a \leq b \\ b, & \text{si } a > b \end{cases} \quad \ldots (5.34)$$

y por lo tanto las relaciones de conjunto difusos simples (no pares ordenados) quedarian:

En la figura 5.29 se puede apreciar que hay valores de velocidad que estan en más de un solo subconjunto difuso con diferentes grados de pertenencia dentro del subconjunto difuso correspondiente (funciones miembro). Con referencia a esto la función de compatibilidad asocia cada valor de la variable en el intervalo entre [0,1] que es el rango de las funciones miembro, por ejemplo las de la siguiente figura [14]:

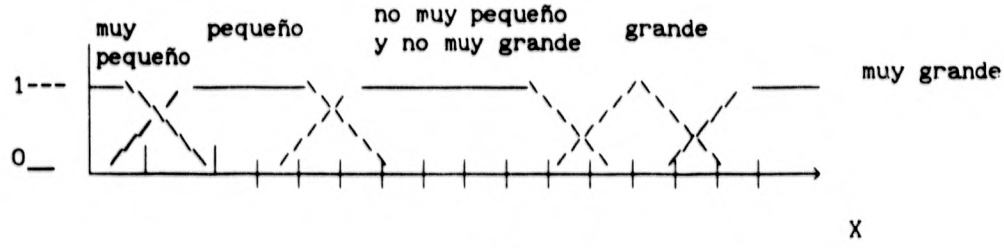



*Fig. 5.30*

Donde cada conjunto difuso tiene un rango en X.
Si tenemos un conjunto difuso A, $\mu_A: U \longrightarrow [0,1]$ el cual asocia con cada elemento de y de U un número $\mu_A(y)$ en el intervalo [0,1], el cual representa el grado de pertenencia de y en A. donde como se había definido:

$$A = \int_U \mu_A(y)/y \ldots\ldots (5.29) \qquad \text{ó}$$

$$A = \mu_1/y_1 + \ldots + \mu_n/y_n; \quad A = \sum_{i=1}^{n} \mu_i/y_i \ldots\ldots\ldots (5.30)$$

en donde $\mu_i$ $i = 1,\ldots,n$ es el grado de pertenencia de $y_i$ en A en

donde $U = y_1 + y_2 + \ldots + y_n$ o $U = \sum_{i=1}^{n} y_i \ldots\ldots\ldots (5.31)$

ó también $U = 1/y_1 + 1/y_2 + \ldots + 1/y_n$ o $U = \sum_{i=1}^{n} 1/y_i \ldots$ (5.32)

De este modo para los números reales a,b podemos definir

$$a \vee b = \max(a,b) \triangleq \begin{cases} a, & \text{si } a \geq b \\ b, & \text{si } a < b \end{cases} \quad \ldots (5.33)$$

$$a \wedge b = \min(a,b) \triangleq \begin{cases} a, & \text{si } a \leq b \\ b, & \text{si } a > b \end{cases} \quad \ldots (5.34)$$

y por lo tanto las relaciones de conjunto difusos simples (no pares ordenados) quedarían:

Unión:

$A \cup B \triangleq \int_U (\mu_A(y) \vee \mu_B(y))/y$, de este modo u "ó" v $\triangleq$ u $\cup$ v...(5.35)

Intersección:

$A \cap B \triangleq \int_U (\mu_A(y) \wedge \mu_B(y))/y$, de este modo u "y" v $\triangleq$ u $\cap$ v...(5.36)

Complemento: $\bar{A} \triangleq \int_U (1-\mu_A(y))/y$.....(5.37)

Producto: $AB \triangleq \int_U \mu_A(y)\ \mu_B(y)/y$.....(5.38)

Potenciación: $A^{\alpha} \triangleq \int_U (\mu_A(y))^{\alpha}/y$.....(5.39)

Multiplicación por un número: $\alpha\ A \triangleq \int_U \alpha\ \mu_A(y)/y$.....(5.40)

Concentración: CON (A) $\triangleq A^2$.....(5.41)

Una implicación entre dos proposiciones es combinada con una segunda proposición y ambas son usadas para implicar una tercera proposición. Esta operación tiene la caracteristica tal que la reducción en la magnitud de grado de pertenencia de"y" en el subset A es relativamente pequeño para "y" la cual tiene un alto grado de pertenencia en A y relativamente grande para "y" con baja pertenencia.

Dilatación: DIL(A) $\triangleq A^{0.5}$.....(5.42), esto es lo opuesto a la concentración. A continuación con valores numericos tendremos ejemplos:

Si tenemos U = 1+2+...+10 (Universo)

u = 0.8/3 + 1/5 + 0.6/6 subconjuntos difusos de u

v = 0.7/3 + 1/4 subconjuntos difusos de v

u "o" v = 0.8/3 + 1/4 + 1/5 + 0.6/6

u "y" v = 0.7/3 + 0.6/6 lo que representa unión e inter sección respectivamente.

Si tenemos A = .8/2 + .9/5; B = 0.6/2 + 0.8/3 +0.6/5

$\therefore$ AB = .48/2 + .54/5

ahora $A^2$ = .64/2 + .81/5

.5A = .4/2 + .45/5

Por último daremos una definición de muy $X \triangleq X^2$ y para ilustrar esto daremos el siguiente ejemplo:

$$U = 1 + 2 + 3 + 4 + 5$$

pequeño: $1/1 + .8/2 + .6/3 + .4/4 + .2/5$

muy pequeño = $1/1 + .64/2 + .36/3 + .16/4 + .04/5$

5.2.4.3.- Razonamiento Aproximado

Razonamiento aproximado, este tipo de razonamiento esta basado en la lógica tradicional y se divide básicamente en dos tipos. Modus Ponens Generalizado (GMP), el cual esta caracterizado por lo siguiente [12]:

| | |
|---|---|
| Premisa 1: X es A' | Aquí se encuentra el valor verdadero de una consecuencia |
| Premisa 2: Si X es A en entonces y es B | una producción de una regla, dando el valor verdadero del antecedente en la regla |
| Consecuencia: y es B' | |

Modus Tollens Generalizado (GMT), esta caracterizado por lo siguiente:

| | |
|---|---|
| Premisa 1: y es B' | Dadas dos proposiciones A "y" B' implica B ambas verdaderas, entonces |
| Premisa 2: Si X es A entonces y es B | la verdad de la simple proposición B es automáticamente inferida. |
| Consecuencia: X es A' | |

Estos tipos de Razonamientos tienen las siguientes caracteristicas, es GMP cuando A' = A "y" B' = B. Es GMT cuando $B' = \bar{B}$ "y" $A' = \bar{A}$. A continuación mostraremos en tèrminos de diagrama de Venn algunas premisas, por ejemplo: $P \rightarrow Q = \bar{A} \cup B$, es verdadero = en todo caso "no en A" o "en B". Así que $(P \rightarrow Q) \leftrightarrow (\bar{P} \vee Q)$ porque "P implica Q es verdadero" cuando en todo "no A" o "B" es verdadero. De esta forma la diferencia A/B es la región donde la implicación "P implica Q" es falso.

La región sombreada es donde la implicación sea verdadera [13].

$\overline{A\,\bar{B}} = \bar{A} \cup B = \overline{(A \cap \bar{B})}$...(5.43); Si X esta en A "y" X no esta en B entonces

$A \rightarrow B$ es falso = $A|B$(diferencia)

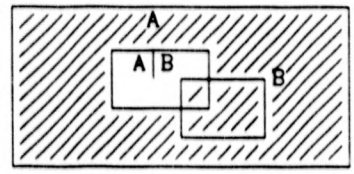



*Fig. 5.31*

Ahora supongamos que la operación de implicación involucra dos diferentes universos. Donde P es una proposición descrita por un set A, el cual es definido en un universo X, y Q en una proposición descrita por un set B, el cual es definido en un universo Y. Entonces la implicación P implica Q" puede estar representada en tèrminos de una relación R, donde R es definida por [13]:

$$R = (A \times B) \cup (\bar{A} \times Y) = \text{Si A, entonces B} \quad ...(5.44)$$

es decir Si $X \in A$ donde $X \in X$, $A \subset X$ entonces $y \in B$ donde $y \in y$, $B \subset y$

Es decir $P \rightarrow Q$ Si $X \in A$, entonces $y \in B$, ó $P \rightarrow Q = \bar{A} \cup B$... (5.45)

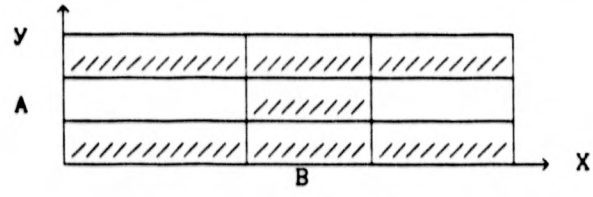



*Fig. 5.32*

Por otra parte para la premisa: B Si A entonces B de otro modo

$C = (A \times B) \cup (\bar{A} \times C)$...(5.46); que es lo mismo que las proposiciones siguientes:

Si A, entonces B, ó Si $\bar{A}$, entonces C

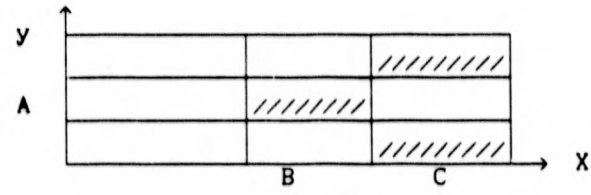



*Fig. 5.33*

A continuación listaremos las deducciones de Modus Ponens [12]:

$(A \wedge (A \rightarrow B)) \rightarrow B$

$(A \wedge (\bar{A} \cup B)) \rightarrow B$

$((A \wedge \bar{A}) \cup (A \wedge B)) \rightarrow B$ ecs. (5.47)

$(\phi \cup (A \wedge B)) \rightarrow B$

$(A \wedge B) \rightarrow B$

$\overline{(A \wedge B)} \cup B$

$(\bar{A} \vee \bar{B}) \cup B$

Por otra parte las deducciones de Modus Tollens serán [12]:

$(\bar{B} \wedge (A \rightarrow B)) \rightarrow \bar{A}$

$(\bar{B} \wedge (\bar{A} \cup B)) \rightarrow \bar{A}$ ecs. (5.48)

$((\bar{B} \wedge \bar{A}) \cup (\bar{B} \wedge B)) \rightarrow \bar{A}$

$((\bar{B} \wedge \bar{A}) \cup \phi) \rightarrow \bar{A}$

$(\bar{B} \wedge \bar{A}) \rightarrow \bar{A}$ ecs. (5.49)

$\overline{(\bar{B} \wedge \bar{A})} \cup \bar{A}$

$(\bar{\bar{B}} \vee \bar{\bar{A}}) \cup \bar{A}$

$B \cup (A \cup \bar{A})$

### 5.2.4.4.- Estructura de un algoritmo Basado en Lógica Difusa

La estructura básica de un algoritmo basado en lógica difusa estará constituido por 5 bloques principales relacionados de tal forma que operen en forma sistemática para poder desempeñar una determinada función. Dicha estructura estará conformada de la siguiente manera (24):

1) Fusificación, el cual desempeña las siguientes funciones:
   a) Mide los valores de las variables de entrada
   b) Ubica en la escala de valores del correspondiente universo a dichas variables.

   c) Convierte los datos de entrada en un grado de pertenencia a uno o más conjunto difusos.

2) Base de Conocimientos o FAM (Fuzzy Asociative Memories): Contiene parte de las reglas en los cuales se basará el sistema para desempeñar su trabajo, constituida de premisas, ubicando los conjuntos difusos de las variables involucradas en cierto arreglo que le permita al bloque siguiente usar dichas reglas.

3) Bloque de toma de Decisiones: Este bloque y el bloque 2 son los que simulan la experiencia, intuición y aprendizaje de un operador humano, o bien que sustituye el modelo o modelos matemáticos los cuales permitirán tener un determinado control del sistema. Esta basado en reglas de inferencia, que permitirán la toma de una decisión para el control de una acción en base a los conjunto difusos de las variables involucradas. En forma genérica se tendrán reglas del siguiente tipo: Si un conjunto de condiciones son satisfechas entonces un conjunto de consecuencias pueden ser inferidas. Estas reglas pueden tomar la siguiente forma que podrá ser modificada de acuerdo a cada sistema.

$R_1$ Si X es $A_1$ 'y' y es $B_1$ entonces Z es $C_1$

$R_2$ Si X es $A_2$ 'y' y es $B_2$ entonces Z es $C_2$

$R_n$ Si X es $A_n$ 'y' y es $B_n$ entonces Z es $C_n$

4. Interfase [illegible] desempeña las siguientes funciones:

a. Se encarga de tomar el conjunto de resultados de la inferencia para determinar estos mediante las reglas de inferencia establecidas y tener un conjunto de valores a las salidas inferidas.

### 5.2.4.4.- Estructura de un algoritmo Basado en Lógica Difusa

La estructura básica de un algoritmo basado en lógica difusa estará constituido por 5 bloques principales relacionados de tal forma que operen en forma sistemática para poder desempeñar una determinada función. Dicha estructura estará conformada de la siguiente manera [24]:

1) Fusificación, el cual desempeña las siguientes funciones:
   a) Mide los valores de las variables de entrada
   b) Ubica en la escala de valores del correspondiente universo a dichas variables.

   c) Convierte los datos de entrada en un grado de pertenencia a uno o más conjunto difusos.

2) Base de Conocimientos o FAM (Fuzzy Asociative Memories): Contiene parte de las reglas en los cuales se basará el sistema para desempeñar su trabajo, constituida de premisas, ubicando los conjuntos difusos de las variables involucradas en cierto arreglo que le permita al bloque siguiente usar dichas reglas.

3) Bloque de toma de Decisiones: Este bloque y el bloque 2 son los que simulan la experiencia, intuición y aprendizaje de un operador humano, o bien que sustituye el modelo o modelos mátematicos los cuales permitirán tener un determinado control del sistema. Esta basado en reglas de inferencia, que permitirán la toma de una decisión para el control de una acción en base a los conjunto difusos de las variables involucradas. En forma gènerica se tendrán reglas del siguiente tipo: Si un conjunto de condiciones son satisfechas entonces un conjunto de consecuencias pueden ser inferidas. Estas reglas pueden tomar la siguiente forma que podrá ser modificada de acuerdo a cada sistema.

$R_1$ Si X es $A_1$ "y" y es $B_1$ entonces Z es $C_1$
$R_2$ Si X es $A_2$ "y" y es $B_2$ entonces Z es $C_2$
.
.
.
$R_n$ Si X es $A_n$ "y" y es $B_n$ entonces Z es $C_n$

4) Defusificación, desempeña las siguientes funciones:

a) Se encarga de tomar el conjunto de valores fusificados para procesar estos mediante las reglas de inferencia establecidas y dar un conjunto de valores a las conclusiones inferidas.

b) Dicho conjunto de valores se mapean, de tal forma que indiquen a que subconjunto difuso de las variables de salida pertenecen y dar un determinado valor como resultado del proceso.

En forma de diagramas de Bloques el sistema quedaría:

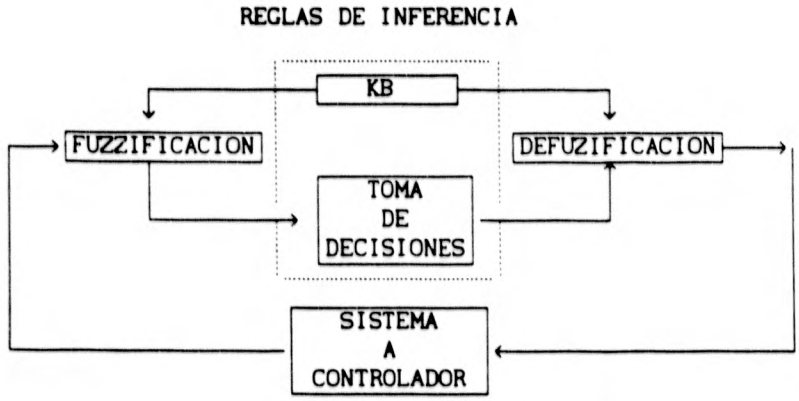



*Fig. 5.34*

## 5.3.- PÁRAMETROS DE DISEÑO DE UN ALGORITMO BASADO EN LÓGICA DIFUSA

Dichos parámetros son puntos fundamentales en el diseño del algoritmo ya que cada cada uno de estos tendrá diferente número de entradas y salidas de variables así como diferentes variables a manejar, tanto de entrada como de salida, a continuación mencionaremos dichos parámetros indicandosu o sus objetivos con su correspondiente clasificación en el caso de que la tengan [29] y [30]:

1.- Estrategias de fusificación: Basicamente, como lo habìamos mencionado la fusificación consiste en convertir un conjunto crisp en un conjunto difuso. Para poder representar los conjuntos difusos se utilizan básicamente 3 tipos de funciones las cuales permiten describir la variable linguística en términos de valores, estas funciones difusas, dependiendo de si el universo es continuo o discreto serán las siguientes:

a) Definición Funcional: Este tipo es para universos continuos, dicha definición permite conocer todas las funciones miembro involucradas en el subconjunto correspondiente, matemáticamente cada función miembro estará representada por:

$$\mu_f(x) = e^{\left\{\frac{-(X-\mu_f)^2}{2\sigma_f^2}\right\}} \quad (5.50)$$

donde $\mu_f$ y $\sigma_f$ son parámetros de normalización que sirven para determinar las funciones miembro.

b) Definición numérica: Este tipo de definición se útiliza para universos discretos, donde los conjunto difusos estarán representados por trapecios o por triángulos donde matemáticamente las funciones miembro estarán representadas por:

$$\mu_f(u) = \sum_{i=1}^{n} ai/\mu_i \quad \ldots(5.51)$$

donde ai son los posibles valores entre 0 y 1 que se toman dentro de los subconjuntos, estas tendrán la siguientes formas, como ejemplo:

Donde se pueden dividir en forma burda

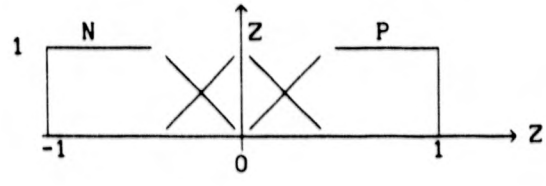
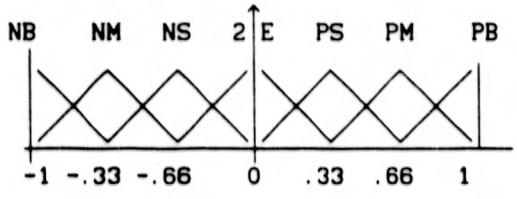



*Fig. 5.35*




En forma fina se puede dividir de la siguiente forma.

*Fig. 5.36*

De esta forma podemos representar conjunto difusos en forma general mediante ecuaciones como las siguientes:

$$A(X) = \begin{cases} \phi_1(x), & x \in [P_0, P_1] \\ \vdots & \\ \phi_S(x), & x \in [P_{S-1} P_S] \end{cases} \quad \text{donde } \phi_i(x) \in [0,1] \text{ para } x \in [P_{i-1}, P_i] \qquad (5.52)$$

$$\phi_i(x) = \sum_{k=0}^{n} C_k^i \, X^k \ldots . (5.53)$$

donde; $i = 1,2,\ldots,S$. y $-\infty = P_0 < P_1 < \ldots < P_{S-1} < P_S = \infty$. Los coeficientes $C_k^{is}$ son párametros de el polinomio $\phi_i(x)$.

Como ejemplo tomamos dos conjunto difusos, los cuales estarán representados por la siguientes figuras:

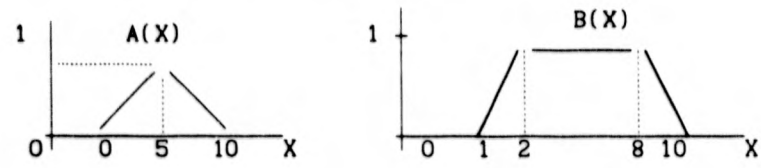



*Fig. 5.37*

De tal forma que:

(5.54)

$$A(X) = \begin{cases} 0 & X \in (-\infty, 0] \\ 0.2X & X \in [0,5] \\ -0.2X + 2, & X \in [5,10] \\ 0 & X \in [10,\infty) \end{cases}$$

(5.55)

$$B(X) = \begin{cases} 0 & X \in (-\infty, 0] \\ 0.5X & X \in [0,2] \\ 1 & X \in [2,8] \\ -0.5X + 5 & X \in [8,10] \\ 0 & X \in [10,\infty) \end{cases}$$

Vale la pena mencionar cierta caracteristica que se tienen con este tipo de conjunto difusos

$$A^1(x) = \begin{cases} \phi_1^1(X), & X \in [P_0^1, P_1^1] \\ \vdots & \\ \phi_{s1}^1,(X), & X \in [P_{s1-1}^1, P_{s1}^1] \end{cases} ; \qquad (5.56)$$

$$A^2(x) = \begin{cases} \phi_1^2(X), & X \in [P_0^2, P_1^2] \\ \vdots & \\ \phi_{s2}^2,(X), & X \in [P_{s2-1}^2, P_{s2}^2] \end{cases} ; \qquad (5.57)$$

Asumiendo que los conjunto difusos tienen funciones miembro continuas.
Entonces:

$A^1(X)\ X\ A^2(X) = \phi_1^1(X)\ X\ \phi_1^2(X)$....(5.58). Como ejemplo tendremos [31].

$$A^1(X) = \begin{cases} 0, & X \in (-\infty, 0], \\ 0.2X, & X \in [0,5], \\ 1, & X \in [5,10], \\ -0.067x + 1.67, & X \in [10,25], \\ 0, & X \in [25,\infty), \end{cases} \qquad (5.59)$$

$$A^2(X) = \begin{cases} 0, & X \in (-\infty, 5], \\ 0.1x - 0.5, & X \in [5,10], \\ 0.5, & X \in [10,15], \\ 0.1x - 1, & X \in [15,20], \\ -0.2x + 5, & X \in [20,25], \\ 0, & X \in [25,\infty). \end{cases} \qquad (5.60)$$

por lo tanto

$$A^1(X)\ XA^2(X) = \begin{cases} 0, & X \in (-\infty, 5], \\ 0.1x - 0.5, & X \in [5,10], \\ -.034x + .84, & X \in [10,15], \\ -.0067x^2 + .234x - 1.67, & X \in [15,20], \\ .0134x^2 - 0.669x + 8.35, & X \in [20,25], \\ 0, & X \in [25,\infty). \end{cases} \qquad (5.61)$$

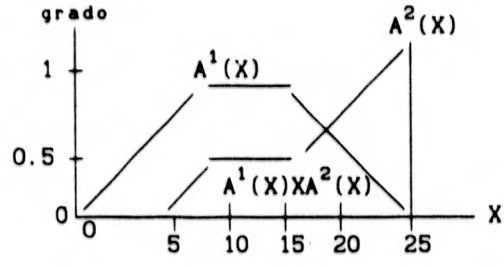



*Fig. 5.38*

De esto podemos concluir que el producto de dos funciones miembro continuas establecen una función miembro continua [31].

2.- Bases de datos: En este punto tendremos en cuenta los siguientes aspectos:

a) Discretización: Donde asignamos valores numéricos a cada función miembro, de acuerdo al valor de la variable tratada.

b) Se colocan los subconjuntos difusos tomando en cuenta el rango de cada subconjunto difuso.

c) Tipos de conjuntos difusos

3.- Implementación de reglas:

a) Selección de Entradas y Salidas y conformando matrices de reglas.

b) Interpretación de reglas de conectividad: si entonces, no, "y", 'o', entonces muy, más o menos etc.

$FAM_s$ (Fuzzy Asociative memory). Estas FAM asocian la regla ó asociación ($A_i$, $B_i$), donde se asocian los conjuntos difusos $B_i$ con los conjunto difusos $A_i$.

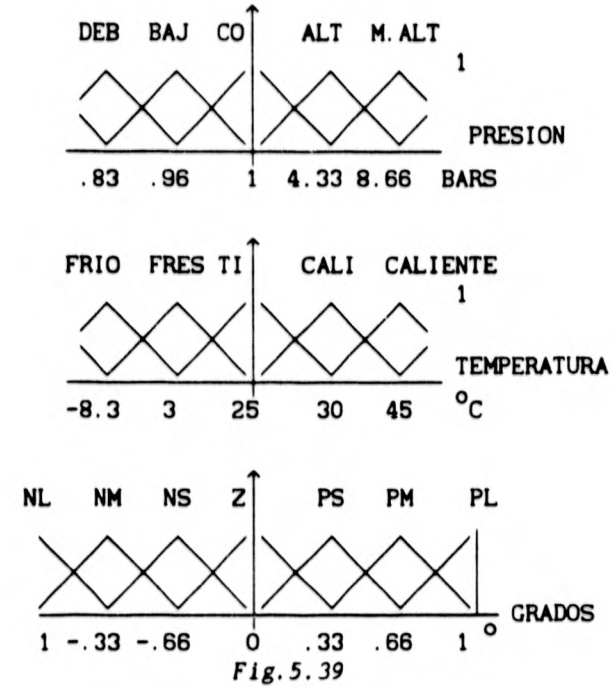



*Fig. 5.39*

En la figura 5.53 se puede apreciar lo que se refiere al punto 3.

**TEMPERATURA**

| PRESION | FRIO | FRES | TIBIO | CAL | CALIENTE |
|---|---|---|---|---|---|
| DEB | | | PM | | |
| BAJA | | | PS | | |
| CORR | PS | PS | Z | NS | NL |
| ALTA | | | NS | | |
| M.AL | | | NM | | |

BANCO DE FAM referido al punto 3a

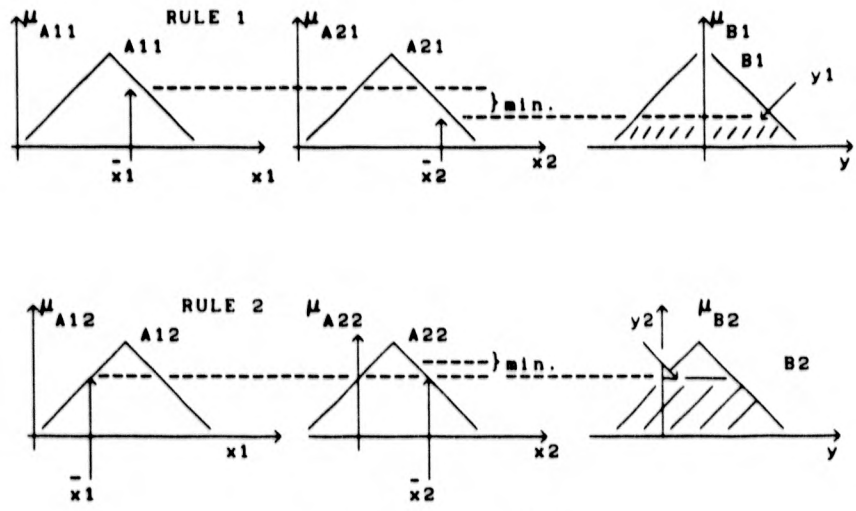
*Fig. 5.40*

Esta matriz es un ejemplo de como las posibles regiones que pudiesen tomar las variables de entrada, así como la intersección de estas tendrá como consecuencia posibles regiones que tomará la variable de salida, las entradas serán la presión cuyas regiones serán: Debil, Baja, Correcta, Alta y Muy Alta. Por otra parte la otra entrada será la temperatura cuyas regiones serán las siguientes: Frío, Fresco, Cálido y Caliente. Lo que dará como salida un determinado angulo de abertura de una válvula, el cual podrá estar en las siguientes regiones: PS (Positivo Pequeño), PM (Positivo Mediano), NS (Negativo Pequeño), NM (Negativo Mediano) y NL (Negativo Grande).

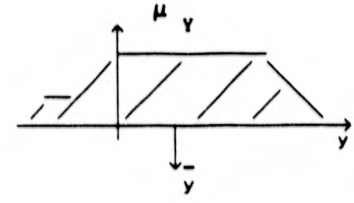
4. Toma de Decisiones, este punto se referirá a los mecanismos de inferencia, de acuerdo a la definición de la implicación difusa. De acuerdo a los diferentes procedimientos de reglas de inferencia [13] se puede tener el método gráfico max-min de inferencia, y el gráfico de producto máximo, entre otros, mostraremos el primero de ellos:

Metodo Gráfico MAX-MIN de inferencia

*Fig. 5.41*




Agregado de reglas




$$\mu_y(y) = MAX_k \{MIN [\mu_{A_1^k}(\bar{X}_1), \mu_{A_2^k}(\bar{X}_2), \mu_{B^k}(y)\} \quad \bar{y} = Defuzzy [\mu_y(y)] \ldots (5.62)$$

Con respecto a las reglas podemos dividirlas en:

4.1) Reglas de Evaluación de Estado que se describen a continuación, con un esquema de múltiples entradas y múltiples salidas (MIMO).

$R_1$: Si X es $A_1$,..., "y" y es $B_1$ entonces Z es $C_1$,..., "y" w es $D_1$

$R_2$: Si X es $A_2$,..., "y" y es $B_2$ entonces Z es $C_2$,..., "y" w es $D_2$

.

.

.

$R_n$: Si X es $A_n$,..., "y" y es $B_n$ entonces Z es $C_n$,..., "y" w es $D_n$

Con un esquema de múltipes entradas y salidas simple (MISO)

$R_1$: Si X es $A_1$,..., "y" y es $B_1$ entonces Z es $C_1$

$R_2$: Si X es $A_2$,..., "y" y es $B_2$ entonces Z es $C_2$

.

$R_n$: Si X es $A_n$,..., "y" y es $B_n$ entonces Z es $C_n$

4.2) Reglas de evaluación de objeto: También llamadas de predicción propuestas por Miyamoto y Ihara [29], donde la tipica regla será $R_i$: Si ($\mu$ es $C_i \rightarrow$ (X es A, "y" y es $B_i$)) entonces $\mu$ es $C_i$ En otras palabras "Si la representación de X es $A_i$, "y" X es $B_i$, cuando un comando de control $\mu$ es seleccionado para ser $C_i$, entonces esta regla es seleccionada y el comando de control $C_i$ es tomado para ser la salida de el controlador, con el contenido de este ejemplo podremos ejemplificar el punto 3b y 4. Donde será la conversión de una cantidad difusa en una cantidad númerica.

5. Estrategias de Defusificación: Se tienen 3 diferentes métodos para poder lograr la fuzzificación de $\mu$ la ó las variables involucradas de salida del sistema, estos métodos son los siguientes:

a) El método de Criterio máximo: Este produce la salida correspondiente al punto en el cual la posibilidad de distribución del control de la acción alcanza el máximo valor.

$$\bar{y} = \text{Defuzzy}\,[\mu_y(y)] \text{ donde } \mu_y(\bar{y}) = \underset{y \in Y}{\text{MAX}}\,[\mu_y(y)] \quad \ldots(5.63)$$

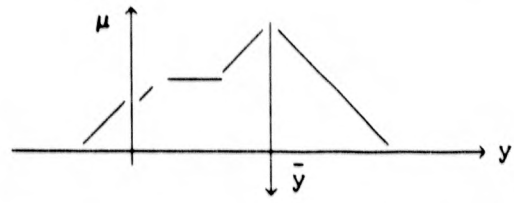



*Fig. 5.42*

b) El significado del método máximo (MOM) Esta estrategia genera una acción de control la cual representa el valor significativo de todas las acciones de control locales cuyas funciones miembro alcanzan el máximo. Esto puede ser expresado por:

$$Z_0 = \sum_{j=1}^{l} \frac{W_l}{l} \qquad (5.64)$$

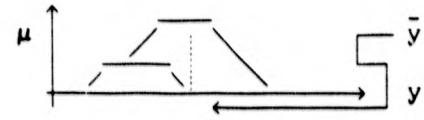



*Fig. 5.43*

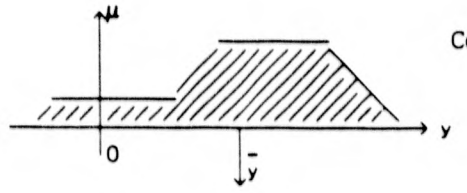
$$\mu_y(y) = \text{MAX}\,[\mu_{y1}(y),\ \mu_{y2}(y), \ldots, \mu_{yn}(n] \quad \text{ec. 5.65}$$

donde $W_j$ es el valor en el cual la función miembro alcanza el máximo valor $\mu_z(W_j)$ y l es el número de valores .

c) El mètodo del Centroide del área: En este tipo de estrategia genera el centro de gravedad de la posibilidad de distribución de una acción de control, donde n es el número de niveles cuantizados de la salida.

$$Z_0 = \frac{\sum_{j=1}^{n} \mu_z(w_j)\, w_j}{\sum_{j=1}^{n} \mu_z(w_j)} \; ; \; \mu_y(y) = \text{MAX}\,[\mu_{y1}(y),\ \mu_{y2}(y), \ldots, \mu_{yr}(r)]$$

ecs. (5.66)




Centro de gravedad.

*Fig. 5.44*

## 5.4.- INTERPRETACIÓN DE SISTEMAS DIFUSOS COMO SISTEMAS DISCRETOS

En este subtema se determinan las ecuaciones que se utilizarán para determinar la estabilidad de un sistema difuso, manejándolo como un sistema discreto, es decir, de las reglas de inferencia involucradas en el sistema difuso, generaremos sus correspondientes ecuaciones en diferencias. Dadas las reglas de la siguiente estructura [31]:

Si $X(k)$ es $A_1^l$ AND....AND $X(k-n+1)$ es $A_n^l$ AND

$\mu(k)$ es $B_1^l$ AND....AND $\mu(k-m+1)$ es $B_m^l$ Entonces

$$X^l(k+1) = a_o^l + a_1^l x(k) + \ldots + a_n^l x(k-n+1) + b_1^l \mu(k) + \ldots + b_m^l \mu(k-m+1) \quad \ldots(5.67)$$

por lo tanto

$$X(k+1) = \sum_{i=1}^{\ell} w^l X^l(k+1) / \sum_{i=1}^{\ell} w^l; \quad w^l = \prod_{p=1}^{n} A_p^l(X(k-p+1)) \; X \prod_{q=1}^{m} B_q^l(\mu(k-q+1)) \quad \ldots(.5.68)$$

$X(k)$ es $P^l$ y $\mu(k)$ es $Q^l$

Entonces $$X^l(k+1) = a_o^l + \sum_{p=1}^{n} a_p^l \, X(k-p+1) + \sum_{q=1}^{m} b_q^l \mu(k-q+1) \ldots(5.69)$$

ecs. (5.70)

$$X(k) = [X(k), X(k-1), \ldots, X(k-n-1)]^T; \quad \mu(k) = [\mu(k), \mu(k-1), \ldots, \mu(k-m+1)]^T$$

donde:

$$P^l = [A_1^l, A_2^l, \ldots, A_n^l]^T, \quad Q^l = [B_1^l, B_2^l, \ldots, B_m^l]^T, \; G^J = [C_1^J, C_2^J, \ldots C_n^J]^T$$

$$H^J = [D_1^J, D_2^J, \ldots, D_m^J]^T$$

De acuerdo a lo anterior existen dos formas de relacionar $\mu(k)$ con las X, a estas formas se les llama **CONECTIVIDAD** tipo A y tipo B, que tendrán las siguientes estructuras [31]:

**TIPO A**

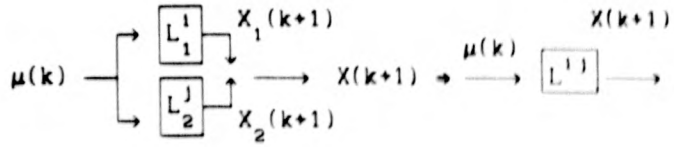



*Fig. 5.45*

$L_1^I$ Si $\underline{X(k)}$ es $P^I$ and $\underline{\mu(k)}$ es $Q^I$ ENTONCES $X_1^I(k+1) = a_0^I + \sum_{p=1}^{n} a_p^I X(k-p+1)$

$+ \sum_{q=1}^{m} b_q^I \mu(k-q+1)$ ....(5.71)

$L_2^J$ Si $\underline{X(k)}$ es $G^J$ and $\underline{\mu(k)}$ es $H^J$ ENTONCES $X_2^J(k+1) = c_0^J + \sum_{p=1}^{n} c_p^J X(k-p+1)$

$+ \sum_{q=1}^{m} d_q^J \mu(k-q+1)$ .....(5.72)

$L^{IJ}$ Si $\underline{X(k)}$ es ($P^I$ and $G^J$) and $\underline{\mu(k)}$ es $Q^I$ and $H^J$)

ENTONCES

$$X^{IJ}(k+1) = (a_0^I + c_0^J) + \sum_{p=1}^{n} (a_p^I + c_p^J) X(k-p+1) + \sum_{q=1}^{m} (b_q^I + d_q^J) \mu(k-q+1)$$

(5.73)

**TIPO B**

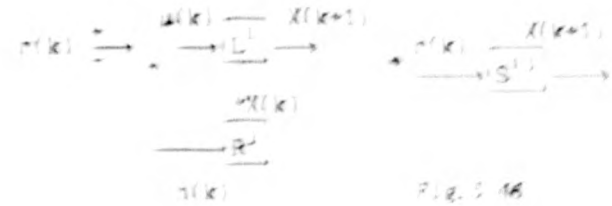


*Fig. 5.46*

De acuerdo a lo anterior existen dos formas de relacionar $\mu(k)$ con las X, a estas formas se les llama **CONECTIVIDAD** tipo A y tipo B, que tendrán las siguientes estructuras [31]:

**TIPO A**

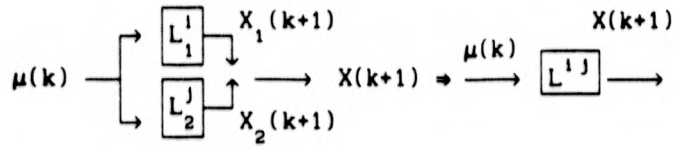



*Fig. 5.45*

$L_1^i$ Si $\underline{X(k)}$ es $P^i$ and $\underline{\mu(k)}$ es $Q^i$ ENTONCES $X_1^i(k+1) = a_0^i + \sum_{p=1}^{n} a_p^i X(k-p+1)$

$+ \sum_{q=1}^{m} b_q^i \mu(k-q+1)$ ....(5.71)

$L_2^j$ Si $\underline{X(k)}$ es $G^j$ and $\underline{\mu(k)}$ es $H^j$ ENTONCES $X_2^j(k+1) = c_0^j + \sum_{p=1}^{n} c_p^j X(k-p+1)$

$+ \sum_{q=1}^{m} d_q^j \mu(k-q+1)$ .....(5.72)

$L^{ij}$ Si $\underline{X(k)}$ es ($P^i$ and $G^j$) and $\underline{\mu(k)}$ es $Q^i$ and $H^j$)
ENTONCES

$$X^{ij}(k+1) = ( a_0^i + c_0^j) + \sum_{p=1}^{n}( a_p^i + c_p^j) X(k-p+1) + \sum_{q=1}^{m} (b_q^i + d_q^j) \mu(k-q+1) \quad .....(5.73)$$

**TIPO B**

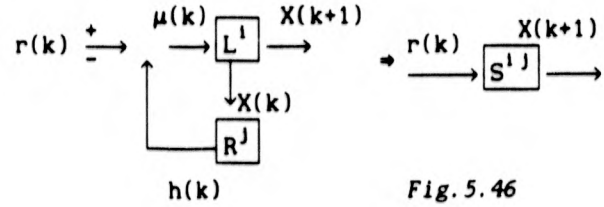



*Fig. 5.46*

$L^1$ Si X(k) es $P^1$ and $\mu(k)$ es $Q^1$

$$\text{ENTONCES } X^1(k+1) = a^1_0 + \sum_{p=1}^{n} a^1_p X(k-p+1) + b^1\mu(k)$$

$$= a^1_0 + \sum_{p=1}^{n} a^1_p X(k-p+1) + b^1(r(k) - h(k)) \quad \ldots(5.74)$$

$R^J$ Si X(k) es $G^J$ and $\mu(k)$ es $H^J$

$$\text{ENTONCES } h^J(k) = c^J_0 + \sum_{p=1}^{n} c^J_p X(k-p+1) \quad \ldots\ldots(5.75)$$

$S^{IJ}$ Si X(k) es ($P^I$ and $G^J$) and $v^*(k)$ es ($Q^I$and $H^J$)

$$\text{ENTONCES } X^{IJ}(k+1) = a^I_0 - b^I c^J_0 + b^I r(k) + \sum_{p=1}^{n} \{a^I_p - b^I c^J_p\} X(k-p+1) \quad \ldots(5.76)$$

ecs. (5.77)

donde $v^*(k)=[r(k)-e^*(X(k)), r(k-1)-e^*(X(k-1), \ldots, r(k-m+1)-e^*(X(k-m+1))]^T$,

$h(k)=e^*(x(k))$

De esta forma podremos representar un sistema difuso en ecuaciones en diferencias y aplicar los conceptos de sistemas discretos a sistemas difusos.

### 5.4.1.- Estabilidad de Sistemas Difusos

Este concepto es de gran importancia para determinar la confiabilidad de los sistemas de este tipo. Para poder determinar la estabilidad de un Sistema Difuso es necesario conocer el comportamiento parcial del sistema, es decir conocer como se comporta el sistema al aplicar cierta regla de inferencia, determinando la estabilidad de cada una de las reglas que controlan el sistema. Si todas las reglas del sistema son estables entonces se deben encontrar ciertos párametros que satisfagan a todas ellas y poder tener un sistema totalmente estable. Dichos parámetros son constantes, producto de un conjunto de ecuaciones las cuales son generadas en base al punto anterior, y que al fin y al cabo representarán las reglas de inferencia. Con la finalidad de aplicar criterios de estabilidad que se usan para sistemas discretos, se generarán dichas ecuaciones. El criterio de estabilidad utilizado será el de LYAPUNOV debido a su simplicidad y semejanza, entre sistemas difusos y discretos, desde el punto de vista de sus ecuaciones.
Por otra parte el equilibrio de un sistema difuso, es globalmente ásintoticamente estable si existe una matriz común positiva definida P para todos los subsistemas difusos tal que [31]:

$$A_i^T P A_i - P < 0 \; ; \quad i \in \{1,2,\ldots,\ell\}. \ldots\ldots (5.78)$$

∴ Si se tiene una función V(X(K) tal que

$V(X(K)) = X^T(K)\, PX(K)$ donde P es una matriz positiva definida ..(5.79)

a) $V(0) = 0$

b) $V(X(K)) > 0$ para $X(K) \neq 0$

c) $V(X(K)) \longrightarrow \infty$; $\|X(K)\| \longrightarrow \infty$

d) $\Delta(X(K) < 0$

V(X(K)) es una función Lyapunov y por lo tanto el sistema difuso es ásintoticamente estable. Por otra parte si $A_i A_j$ es una matriz no estable entonces no existe una matriz común P donde $A_i$ es la matriz correspondiente al subsistema difuso i y $A_j$ al subsistema j, esto se ilustra con el siguiente ejemplo [31] :

$$L^1\ X^1(K+1) = 2.178X(K) - 0.588X(K-1) + .603\ \mu(K)$$
$$L^2\ X^2(K+1) = 2.256X(K) - 0.361\ X(K-1) + 1.120\ \mu(K)$$
$$R^1\ \mu(K) = KX(K) \qquad \text{ecs.(5.80)}$$

∴ $S^1\ X^1(K+1) = (2.178 - .603K)\ X(K) - .588\ X(K-1)$

$S^2\ X^2(K+1) = (2.256 - 1.120K)\ X(K) - .361\ X(K-1)$ ecs.(5.81)

Determinando primeramente la función de transferencia de Lazo Abierto y posteriormente la de lazo Cerrado tenemos que [22]:

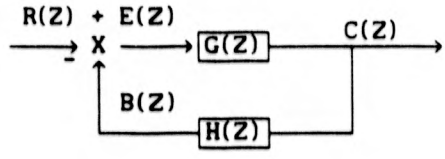



*Fig. 5.47*

La funcion de transferencia de lazo abierto quedaria de la siguiente forma [22]:

$$B(Z) = H(Z)C(Z);\ldots \quad (5.184)$$

$$B(Z) = H(Z)G(Z)(E(Z)) \ldots . (5.82)$$

y por lo tanto

$$\boxed{\frac{B(Z)}{E(Z)} = H(Z)G(Z);} \quad (5.83)$$

La función de transferencia de lazo cerrado quedaria como [31]:

$$C(Z)=G(Z)\ E(Z) \quad (5.84)$$

$$E(Z)=R(Z)-B(Z);\ C(Z)=G(Z)[R(Z)-B(Z)] \quad (5.85)$$

$$C(Z)=G(Z)R(Z)-G(Z)H(Z)C(Z) \quad (5.86)$$

$$C(Z)[1+G(Z)H(Z)]=G(Z)R(Z) \quad (5.87)$$

$$\boxed{\frac{C(Z)}{R(Z)} = \frac{G(Z)}{1+\ G(Z)H(Z)}} \quad (5.88)$$

Para $S^1$ la transformada en Z

$$G(Z) = \frac{C(Z)}{R(Z)}\ ;\quad X(K+1) - (2.178 - .603K)\ X(K) + .588\ X(K-1) = 0 \quad ..(5.89)$$

La ecuacuación característica será C(Z) + R(Z) = 0.....(5.90)

$$X(K) - (2.178 - .603K)X(K-1) + .588X(K-2)=0$$

$$X(Z) - (2.178 - .603K)Z^{-1}X(Z) + .588\ Z^{-2}\ X(Z)=0$$

$$Z^2 - (2.178 - .603K)Z^{-1} + .588 = 0 \qquad \text{ecs.(5.91)}$$

$$Z^2 + (.603K - 2.178)\ Z^{-1} + .588 = 0$$

Utilizando el método del lugar de las raices

$$2\xi W_n = (.603K - 2.178) \qquad (5.92)$$

$$W_n^2 = .588\ ;\quad W_n = .7668$$

Sust en ec.(5.92)

$$\frac{1.53\xi + 2.178}{(.603)} = K$$

Para valores de $-1 < \xi < 1$ se tendra $.980 < K < 6.25$

Para $S^2$

$$X(K+1) - (2.256 + 1.120K)X(K) + 361\ X(K-1) = 0 \qquad (5.93)$$

$$X(K+1) + (1.120K - 2.256)X(K) + .361\ X(K-1) = 0 \qquad (5.94)$$

$$X(K) + (1.120K - 2.256)X(K-1) + .361\ X(K-2) = 0 \qquad (5.95)$$

$$X(Z) + (1.120K - 2.256)Z^{-1}X(Z) + .361\ Z^{-2}X(Z) = 0 \qquad (5.96)$$

$$Z^2 + (1.120K - 2.256)Z^{-1} + .361 = 0 \qquad (5.97)$$

Utilizando el metodo del lugar de las raices

$$2\xi W_n = 1.120K - 2.256 \qquad (5.98)$$

$$W_n^2 = .361 \ ; \quad W_n = .60$$

$\frac{1.2\xi + 2.256}{1.120} = K$ Para valores de $-1 < \xi < 1$ se tendrá

$$.80 < K < 3.2$$

$\therefore$ Por lo tanto los valores que satisfacen las dos ecuaciones en diferencias serán:

$$.98 < K < 3.28$$

Entonces tendremos las matrices $A_1$ y $A_2$ respectivamente de las ecuaciones $S^1$ y $S^2$ respectivamente.

$$A_1 = \begin{bmatrix} 2.178 - 603K & -.588 \\ 1 & 0 \end{bmatrix}; \quad A_2 = \begin{bmatrix} 2.256 - 1.120K & -.361 \\ 1 & 0 \end{bmatrix}$$

Si escogemos un valor de $K = 1.12$

$$A_1 = \begin{bmatrix} 1.50 & .588 \\ 1 & .0 \end{bmatrix}; \quad A_2 \begin{bmatrix} 1. & -.361 \\ 1 & 0 \end{bmatrix}$$

$A_1A_2 = \begin{bmatrix} .912 & -.54 \\ 1 & -.361 \end{bmatrix}$ utilizando la ecuación $A^TPA - P = -Q$ para $X_e = 0$

donde $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ y $P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$

ó $A_1^T PA_1 - P < 0$ y $A_2^T PA_2 - P < 0$

ó $A_1^T PA_2 + A_2^T PA_1 - 2P < 0 \Rightarrow -(A_1-A_2)^T P(A_1-A_2) \leq 0$ ecs.(5.99)

Estas ecuaciones son del teorema de Estabilidad de Lyapunov.

Resumiendo: Un sistema difuso es globalmente asintoticamente estable si sus subsistemas difusos son estables, (para estos, sus matrices $A_i$ tendrán que ser estables, esto será determinado por el lugar de las raices K, $\xi$ etc, ó por sus valores caracteristicos) y además el producto de esas matrices $A_iA_j$ es una matriz estable, entonces existirá una matriz común P tal que cumple la siguiente ecuación para todas las matrices $A_i^T PA_i - P<0$ ó tomando un valor inicial $X_e=0$ (estado de equilibrio) <0, tomamos:

$$-Q = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$ por lo tanto la ecuación será

$$A_i^T P\, A_i - P = -Q \quad \ldots.(5.100)$$

por lo tanto sabemos que $A_1A_2$, para este caso son estables, entonces se tendrá que cumplir que:

$A_1^T P\, A_1 - P<-Q$ ...(5.101) y $A_2^T P\, A_2 - P <-Q$ .....(5.102)

De esta forma para la ec.(5.102), tendremos:

$$\begin{bmatrix} 1.5 & 1 \\ -.588 & 0 \end{bmatrix}\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\begin{bmatrix} 1.5 & -.588 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

(5.103)

Para la ec.5.101:

$$\begin{bmatrix} 1 & 1 \\ -.361 & 0 \end{bmatrix}\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\begin{bmatrix} 1 & -.361 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$ ec.5.104

De la ecuación (5.103)

$$\begin{bmatrix} 1.5P_{11}+P_{21} & 1.5P_{12}+P_{22} \\ -.588P_{11} & -.588P_{12} \end{bmatrix}\begin{bmatrix} 1 & .5 & -.588 \\ 1 & & 0 \end{bmatrix}-\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}=\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 2.25P_{11}+1.5P_{21}+1.5P_{12}+P_{22} & -.882P_{11}-.588P_{21} \\ -.882P_{11}+.588P_{12} & .34P_{11} \end{bmatrix}-\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}=\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$2.25P_{11}+1.5P_{21}+1.5P_{12}+P_{22}-P_{11}=-1$ a)

$-.882P_{11}-.588P_{21}-P_{12}=0$ b)

$-.882P_{11}-.588P_{12}-P_{21}=0$ c)

$.34P_{11}-P_{22}=-1$ d)

De la ecuación 5.104.

$$\begin{bmatrix} P_{11}+P_{21} & P_{12}+P_{22} \\ -.361P_{11} & -.361P_{12} \end{bmatrix}\begin{bmatrix} 1 & -.361 \\ 1 & 0 \end{bmatrix}-\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}=\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} P_{11}+P_{21}+P_{12}+P_{22} & -.361(P_{11}+P_{21}) \\ -.361P_{11} & -.361P_{12} \quad .13P_{11} \end{bmatrix}-\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}=\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$P_{11}+P_{21}+P_{12}+P_{22}-P_{11}-1$ a')

$-.361(P_{11}+P_{21})-P_{12}=0$ b')

$-.361(P_{11}+P_{12})-P_{21}=0$ c')

$.13P_{11}-P_{22}=-1$ d')

de 4 $P_{22}=.13P_{11}+1$ y sust en a')

$P_{11}+P_{21}+P_{12}+.13P_{11}-P_{11}=-2$

$P_{21}+P_{12}+.13P_{11}=-2$ e')

de 2 $-.361P_{11}-.361P_{21}=P_{12}$ f')

de 3 $-.361P_{11}-.361P_{12}=P_{21}$ g')

De la ecuación (5.103)

$$\begin{bmatrix} 1.5P_{11}+P_{21} & 1.5P_{12}+P_{22} \\ -.588P_{11} & -.588P_{12} \end{bmatrix}\begin{bmatrix} 1 & .5 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 2.25P_{11}+1.5P_{21}+1.5P_{12}+P_{22} & -.882P_{11}-.588P_{21} \\ -.882P_{11}+.588P_{12} & .34P_{11} \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$2.25P_{11}+1.5P_{21}+1.5P_{12}+P_{22}-P_{11}=-1$ a)

$-.882P_{11}-.588P_{21}-P_{12} = 0$ b)

$-.882P_{11}-.588P_{12}-P_{21} = 0$ c)

$.34P_{11} - P_{22} = -1$ d)

De la ecuación 5.104.

$$\begin{bmatrix} P_{11}+P_{21} & P_{12}+P_{22} \\ -.361P_{11} & -.361P_{12} \end{bmatrix}\begin{bmatrix} 1 & -.361 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} P_{11}+P_{21}+P_{12}+P_{22} & -.361(P_{11}+P_{21}) \\ -.361P_{11} & -.361P_{12} \quad .13P_{11} \end{bmatrix} - \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$P_{11}+P_{21}+P_{12}+P_{22}-P_{11}-1$ a')

$-.361(P_{11}+P_{21}) - P_{12} = 0$ b')

$-.361(P_{11}+P_{12}) - P_{21} = 0$ c')

$.13P_{11} - P_{22} = -1$ d')

de 4 $P_{22}=.13P_{11}+1$ y sust en a')

$P_{11}+P_{21}+P_{12}+.13P_{11}-P_{11} = -2$

$P_{21}+P_{12}+.13P_{11} = -2$ e')

de 2 $-.361P_{11} -.361P_{21}=P_{12}$ f')

de 3 $-.361P_{11} -.361P_{12}=P_{21}$ g')

Sust g') en f')

$$-.361P_{11}-.361[-.361P_{11}-.361P_{12}]=P_{12}$$

$$-.361P_{11}+.13P_{11}+.13P_{12}=P_{12}$$

$$-.231P_{11}=.87P_{12} \qquad h')$$

sust g') en e')

$$-.361P_{11}-.361P_{12}+P_{12}+.13P_{11}=-2$$

$$-.231P_{11}+.639P_{12}=-2 \qquad i')$$

sust h') en i')

$$.87P_{12}+.639P_{12}=-2 \therefore \boxed{P_{12}=-1.3}$$

De igual forma se obtienen el resto de las $P_s$ de las ecuaciones (5.101) y (5.102), una vez teniendo todass las $P_s$ se comparan y se escogen de tal forma que cumpla con las siguientes desigualdades:

$$2.25P_{11}+1.5P_{21}+1.5P_{12}+P_{22}-P_{11}<0$$

$$-.882P_{11}-.588P_{21}-P_{12}<0$$

$$-.882P_{11}-.588P_{12}-P_{21}<0$$

$$.34P_{11}-P_{22}<0 \qquad \text{ecs. (5.105)}$$

$$P_{11}+P_{21}+P_{12}+P_{22}-P_{11}<0$$

$$-.361(P_{11}+P_{21})-P_{12}<0$$

$$-.361(P_{11}+P_{12})-P_{21}<0$$

$$-.13P_{11}-P_{22}<0$$

O bien cualquier valor que cumpla con las desigualdades simultáneas anteriores para cada $P_s$

### 5.4.2.- Diseño de un Controlador Difuso

De acuerdo a párrafos anteriores podemos determinar un procedimiento para el diseño de un controlador difuso y su representación sería similar al de un controlador convencional como lo muestra la siguiente figura [33] .

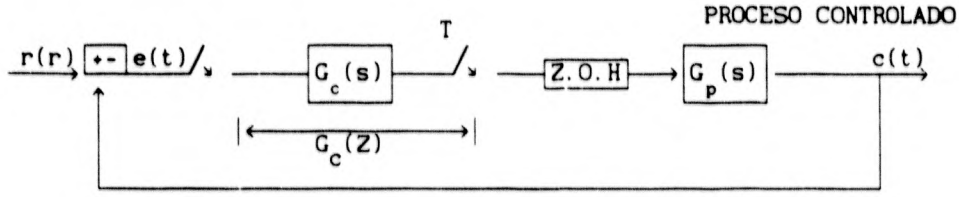



*Fig. 5.48 Controlador Digital con controlador de lazo en cascada*

De acuerdo a esto podemos implementar el diagrama de un controlador difuso considerando la estructura vista en puntos anteriores, esto se muestra en la figura.5.58.

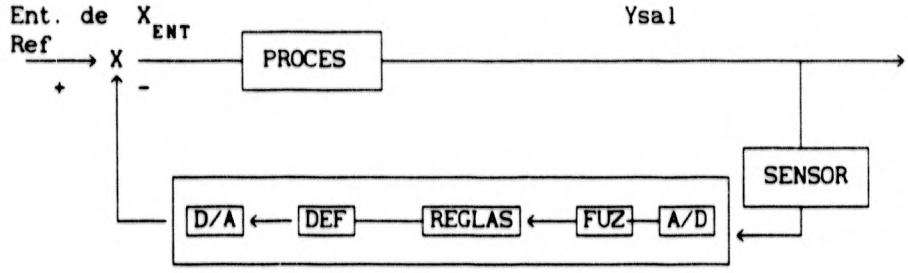



*Fig. 5.49 Diagrama a bloques de un controlador difuso*

El procedimiento de diseño de un controlador difuso estára basado en la siguiente estructura [31]:

$L^1$ Si X(K) es $A^1$ y $\mu(K)$ es $B^1$entonces $X^1(K+1) = a^1 X(K) + b^1\mu(K)$

$L^2$ Si X(K) es $A^2$ y $\mu(K)$ es $B^2$entonces $X^2(K+1) = a^2 X(K) + b^2\mu(K)$

Como ejenmplo para el modelo difuso, nosotros podemos diseñar el siguiente controlador [31].

$R^1$ Si X(K) es $A^1$ y $\mu(K)$ es $B^1$entonces $\mu^1(K) = f^1X(K)$ .....(5.106a)

$R^2$ Si X(K) es $A^2$ y $\mu(K)$ es $B^2$entonces $\mu^2(K) = f^2X(K)$.....(5.106b)

Para el tipo de correctividad B

$S^{11}$ Si X(K) es ($A^1$ y $A^1$) y $\mu(K)$ es ($B^1$ y $B^1$)
entonces $X^{11}(K+1) = (a^1-b^1f^1)X(K)$.....(5.106c)

$2S^{12*}$ Si X(K) es ($A^1$ y $A^2$) y $\mu(K)$ es ($B^1$ y $B^2$)
entonces $X^{12}(K+1) = (a^1+a^2-b^1f^2-b^2f^1)X(K)/2$.....(5.107)

$S^{22}$ Si X(K) es ($A^2$ y $A^2$) y $\mu(K)$ es ($B^2$ y $B^2$)
entonces $X^{22}(K+1) = a^2-b^2f^2)X(K)$.....(5.108)

De donde la salida será

$$\frac{W^1 X^{11}(K+1) + 2W^2 X^{12*}(K+1) + W^3X^{22}(K+1)}{W^1 + 2W^2 + W^3} \quad \text{.....(5.109)}$$

### 5.4.3.- Autosintonización de un Controlador Difuso

Este tipo de autosintonizador es basado en la estructura matemática de los puntos anteriores. Consistiendo en tomar las reglas de inferencia como ecuaciones en diferencias. De esta forma se puede "estimar" un determinado comportamiento de las variables involucradas de acuerdo al controlador difuso usado, este a su vez será diseñado acorde a las reglas de inferencia involucradas. Tratando de la acción de control más acertada en cuanto a su exactidud, de tal forma que el resultado que proporcione el control permita aproximarse poco a poco a una solución óptima basado, en un conjunto de resultados obtenidos. Es decir al generar un resultado se tendrá que tomar en cuenta el resultado inmediato anterior y compararlos, con un resultado propuesto como final, y el que resulte más apegado al propuesto, será el resultado final. Esta optimización dependerá de varios factores de acuerdo a las necesidades de diseño . El modelo esquemático de un controlador difuso autosintonizable podrá representarse mediante la siguiente figura [32],[13] y [36].

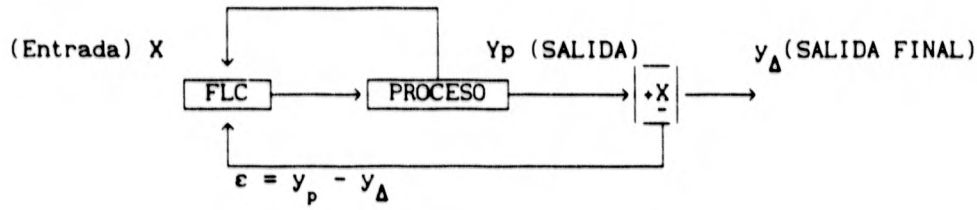



*Fig.5.50 Estructura General de un Sistema Autosintonizable*

La autosintonización se hará estimando párametros de acuerdo al error de salida que proporcionan las entradas actuales. Se estiman las posibles salidas de acuerdo a la minimización del error.

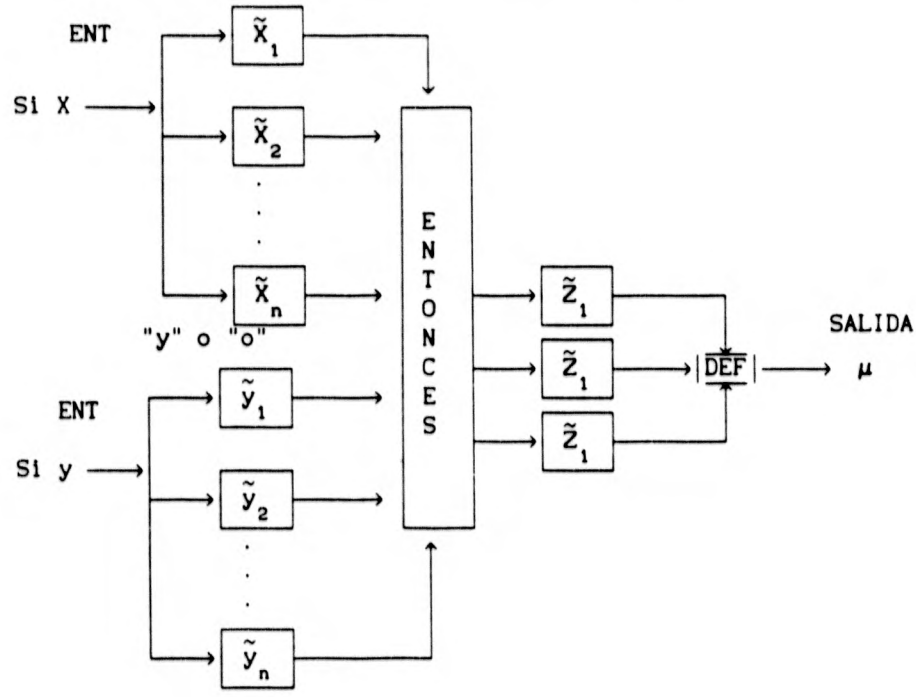



*Fig.5.51 Sistema Autosintonizable*

Donde $\mu = \sum_{i=1}^{n} a_i C_i$ ...(5.110a) será la salida calculada como cualquier centroide de defuzzificación, para el operador "o" se tendrá:

$$\mu_{RM}(X, Y) = V(\mu_{\bar{X}_j}(X), \mu_{\bar{y}_j}(Y)) \quad .....(5.110b)$$

y para el operador "y" se tendrá

$$\mu_{RM}(X, Y) = A(\mu_{X_j}(X), \mu_{y_j}(Y)) \quad \ldots\ldots(5.111)$$

En otras palabras considerando las ecuaciones , $\mu^1(k)$ "y" $\mu^2(k)$, que darán las salida $\mu$ en el caso de tomar solo tomar dos valores de $\mu$. Ya que X "y" Y serán los valores de $A^1$ y $B^1$ para el caso de la regla 1, para el caso de la regla 2, $A^2$y $B^2$, dichos coeficientes corresponden a las ecuaciones del tema anterior. Para un sistema no sintonizable el diagrama de bloques sería de la siguiente forma:

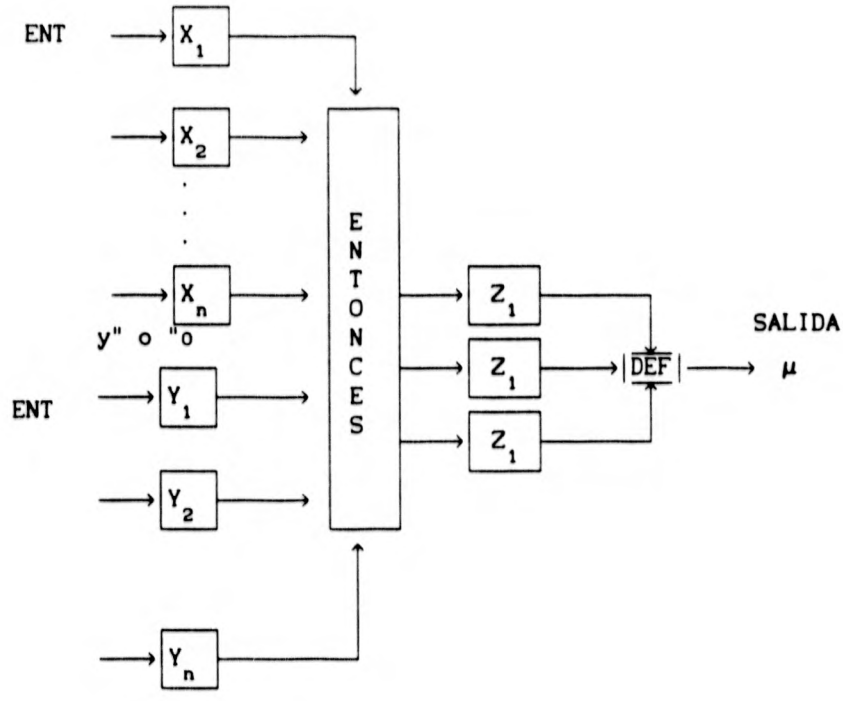



*Fig. 5.52 Sistema no Sintonizable*

Se debe notar que para detectar una señal de salida se deberá tener un parámetro de comparación, es decir una señal de referencia que permita conocer un determinado error entre la señal de referencia ó estimada y la salida real. Esta diferencia es generada por la señal estimada y la salida real en un determinado instante de tiempo, es decir va a ser la diferencia de dos señales. Si almacenamos dicha diferencia y esperamos

tener otra muestra de señal de salida y de valor estimado, podremos generar una segunda diferencia, y con las dos diferencias obtener un resultado que sería la diferencia de la primera diferencia y la segunda diferencia. Tomando en cuenta que el valor estimado utilizado puede ser el mismo o puede ser diferente. Con esta diferencia de diferencias se tiene una diferencia de segundo orden y del mismo modo se pueden obtener diferencias de ordenes mayores. Todo esto con la finalidad de generar una señal de error más exacta. Los controladores difusosautosintonizables pueden estar diseñados de varias formas, y esto varía de acuerdo a las necesidades que se tengan. En términos generales dichos controladores estarán constituidos por:

a) Comparador de Entrada.- Es el que genera la señal de error en un determinado instante

b) Retenedores de orden n.- Donde se almacenan las diferencias de un determinado orden.

c) Prioritizadores - Donde se designa que grado de importancia va a tener un determinado valor número. Los prioritizadores ó ajustadores serán los coeficientes de dichos valores.

d) Retardadores.- Son elementos que permiten el obtener una determinada señal un instante después exactamente como los son los factores $Z^n$ en ecuaciones en diferencias. Ya que al fin y al cabo las reglas de inferencia serán transformadas en ecuaciones en diferencias.

e) Bloque de control difuso.- Estará constituido por tres sub-bloques que serán el bloque de variables linguísticas, donde se maneja un determinado número de dichas variables , de las cuales solo estarán en disponibilidad las que se vayan a usar, asignando un determinado rango numérico por el bloque de asignación de rango número, y por último el subbloque de Reglas difusas, que es donde se almacenan las reglas en uso.

f) Comparador de salida.- Compara dos salidas en dos diferentes instantes de tiempo y entrega una determinada diferencia.

g) Defusificador.- Asigna un determinado valor numérico a las salidas ó salida involucradas.

h) Modificador de Controlador Difuso.- Modifica de acuerdo al tipo de salida, colocando nuevos elementos y/ó inhibiendo algunos que están en cualquiera de los tres subbloques del controlador difuso, es decir modifica y/ó agrega una variable linguística, modifica el rango número y cambia y/ó agrega las reglas.

Cabe mencionar que se pueden combinar dichos bloques de manera de tener un diseño específico para cada controlador en especial. En la figura 5.53 se representa un posible diseño de un controlador difuso autosintonizable de orden 2.

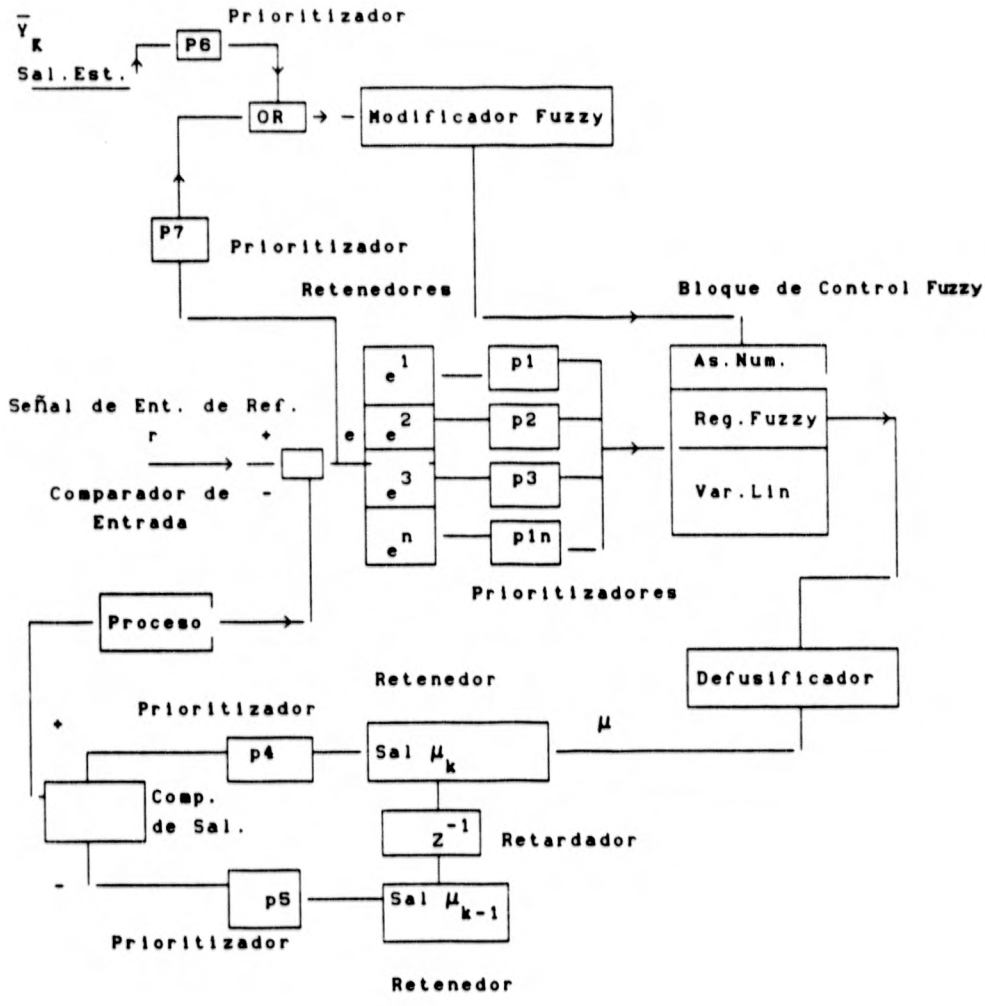



*Fig.5.53 Propuesta de un Sistema Autosintonizable*

En el capitulo 6 se explica el tipo de controlador difuso autosintonizable utilizado en este trabajo, tanto en el procesador central como en los controladores.

## 5.5.- CONCLUSIONES

Los algoritmos basados en lógica difusa son apropiados para controlar procesos, cuyo modelo matemático es muy complejo y/ó el modelo del controlador también lo es. Teniendo en cuenta que es ideal para representaciones intuitivas ya que simula la experiencia y los conocimientos de un operador humano.

Por otra parte para analisis sobre estabilidad, controlabilidad etc., se pueden representar esta clase de controladores mediante ecuaciones discretas lo que da un herramienta adecuada al analisis y diseño de esta clase de sistemas, e incluso poder diseñar controladores con caracteristicas hibridas, de tal forma que puedan combinar algoritmos difusos con algoritmos tradicionales y de esta forma incrementar su eficiencia, ya que para ciertas aplicaciones será mejor usar algoritmos difusos que tradicionales y viceversa.

# 6.- APLICACION DE LOS SISTEMAS DE TIEMPO REAL AL MONITOREO Y CONTROL DE ALARMAS

## 6.1.- INTRODUCCIÓN

En este cápitulo se utilizan los fundamentos teóricos para dar los lineamientos referentes al diseño del sistema y de esta forma implementar acorde con los aspectos de tiempo real, paralelismo, y control difuso, el sistema de control distribuido en tiempo real usando controladores implementados con arquitecturas paralelo y pipeline, y algoritmos de control difuso.

## 6.2.- DESCRIPCIÓN GENERAL DEL SISTEMA

El objetivo de este sistema es controlar procesos de diferente naturaleza, mediante un controlador para cada uno de ellos, teniendo la posibilidad de conectarse mediante un bus hacia un comando central, el cual puede monitorear, registrar informaciones, enviar señales hacia puntos distantes, y controlar diferentes procesos en tiempo real. En el presente trabajo se manejan diferentes áreas del conocimiento, tales como la electrónica, el control y el cálculo distribuido, en las cuales utilizamos dispositivos semiconductores, sistemas digitales, interfaces etc..Se puede decir que los procesos a controlar aunque son diferentes en su naturaleza, pero que en un determinado momento, pueden tener comunicación entre sí, haciendo versátil el uso de este tipo de implementación. Como aplicación consideramos un sistema de seguridad automático de locales, el cual involucra control de procesos mediante sensores y actuadores. Como lo muestra la figura 6.1, cada proceso se conectará a un controlador diferente el cual accionará determinados actuadores "A" de acuerdo a las variables determinadas por los sensores "S". Cada controlador tendrá diferentes tipos de actuadores y sensores e indicadores de acuerdo al proceso a manejar, dichos procesos se referirán a procesos que intervienen en un sistema de seguridad como son detección de movimiento, ruido, calor, etc. ya sea usando cámaras de video, sensores ácusticos, sensores de proximidad, sensores infrarojos etc, y como actuadores motores, válvulas solenoide, bocinas, focos etc. Dichos controladores estarán conectados a una PC, donde dicha PC será el procesador central a través de un controlador de tráfico (CTSP), la cual operará con el Sistema Operativo en tiempo real QNX 4, cuyas descripción haremos más adelante. Dicho procesador central asignará cierto código a cada uno de los controladores, que al recibir la petición de requisición de información y/ó bien órdenes, provenientes de la PC, dichos controladores se interrumpirán pero solo el del código seleccionado acatará el mandato, el resto al ver que no se trata del código correspondiente a ellos seguirá trabajando normalmente. De este modo la PC podra monitorear y controlar los procesos con la información proveniente de los controladores a través de su puerto serie, conectandose de antemano el controlador de tráfico. Dicha información

proveniente de los controladores a la PC serán números en hexadecimal que entrarán en forma serial, los cuales representarán posición de un objeto, activación de un sensor en una determinada área del local ya sea ácustico, infrarrojo, etc. La PC acomadará los números en la pantalla de tal forma que se pueda saber que tipo de sensor se activo y en que área del local, ya sea en forma gráfica o númerica donde se aplica dicho sistema. Deberá ser fundamental que dicho sistema sea dentro de los límites permisibles, adaptable a cualquier tipo de proceso, de tal forma que pueda usarse en la industria, hospitales, centros comerciales, oficinas etc.

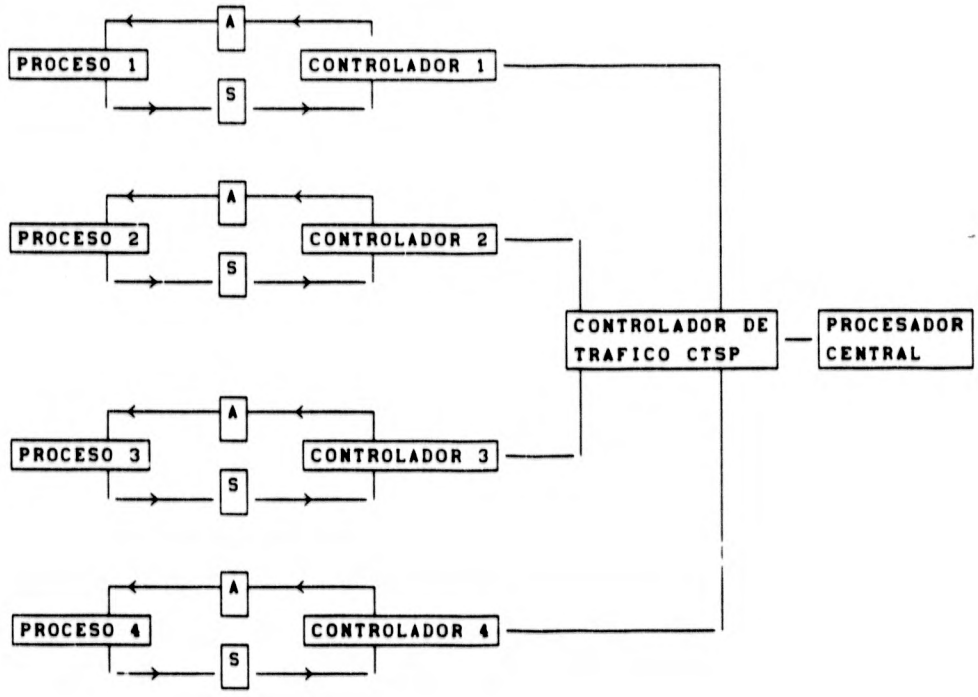



*Fig.6.1 Arquitectura del Sistema Distribuido*

## 6.3.- MANEJO DE LOS PROCESOS INVOLUCRADOS

Independientemente de los procesos que intervengan en el control del sistema, cada proceso tendrá un número determinado de variables de entrada y un número determinado de variables de salida. Aunque se deberá tomar en cuenta que el máximo número de entradas a procesar será de 4 y el número de salidas procesadas será de 4, en otras palabras se podrán tener en un mismo proceso más de un sensor del mismo tipo y más de un actuador del mismo tipo en el mismo proceso, pero sólo se utilizarán uno o dos sensores y un actuador para el control y monitoreo de una ó dos variables del mismo proceso en el mismo instante de muestreo. Se debe hacer notar que cada proceso será diferente, y por lo tanto los controladores, desde el punto de vista arquitectura, y actuadores serán diferentes, aunque tengan similitud en algo. Dichos procesos tendrán como característica que todos ellos serán controlados por algoritmos difusos implantados en las arquitecturas correspondientes, estando acorde dichos algoritmos con las arquitecturas utilizadas.

### 6.3.1.- Descripción de los procesos involucrados en el sistema

El proceso 1 estará constituido por la detección de objetos móviles dentro de una determinada área, los sensores usados serán sensores de proximidad, detectando movimiento, estos activarán un motor que a su vez controlará una cámara de video para seguir la imagen del objeto dentro de determinada área, se tendrán dos sensores de proximidad de tal forma que proporcionan cada uno de ellos la información correspondiente a X y a Y, procesando esta información en todos los instantes de muestreo.

El proceso 2 consistirá en monitorear 4 sensores ópticos "S" y activar 4 actuadores "A" de tal forma que dichos sensores estarán colocados en diferentes lugares del local a vigilar de tal forma que cuando se detecte en cada sensor una interrupción se activará un determinado actuador cerrando el respectivo acceso como puede ser ventana o puerta.

El proceso 3 estará determinado por el control y monitoreo de presión y temperatura de la zona, usando para esto cuatro sensores, 2 de temperatura y 2 de presión, los sensores de temperatura activarán dos actuadores los cuales estarán conectados a los encendidos de un calefactor y un ventilador respectivamente, por otra parte los sensores de presión activarán dos actuadores conectados a un compresor y a una válvula, todo esto para mantener la presión y la temperatura en determinados rangos.

El proceso 4 estará constituido por la determinación de niveles de ruido, o bien de niveles de líquidos donde se utilizarán 4 sensores los cuales serán amplificadores de bajo ruido de tal forma que al detectar un umbral determinado activará 4 indicadores respectivamente. Dichos sensores y actuadores estarán colocados en diferentes lugares del local.

### 6.3.2.- Algoritmos Difusos utilizados en el Monitoreo y Control de los Procesos.

Los algoritmos usados en los controladores difusos utilizados en la implementación de este trabajo tendrán de 2 a 4 entradas y de una salida a 4, dependiendo del proceso de que se trate. Dichos algoritmos estarán basados en la información proveniente de variables diferentes de entrada y que darán como resultado, variables de salida, dichos algoritmos se manejarán de diferente forma por lo que respecta al software y a la arquitectura utilizada, es decir dependerá de cada controlador local y del procesador central. Es importante mencionar que el método utilizado para la fusificación será el de triángulos y el mètodo utilizado para la defusificación es el de centroides, los cuales se explicaron en capítulos anteriores. A manera de descripción mencionaremos en forma general como actúan las variables de entrada y las de salida, como X, Y y $\beta$ respectivamente. En el caso de la ubicación del objeto móvil (proceso 1), [83], las variables de entrada serían las que indicarían la posición lineal siendo estas la abcisa y la ordenada y el ángulo que girará el motor para ubicar la cámara con respecto al objeto, para el caso de los procesos 2 ,3 [82] y 4 [84] se tienen solamente dos posibles valores que pueden tener los actuadores. Estos algoritmos tendrán exactamente la misma estructura que la del proceso 1, con la excepción que sólo usarán dos reglas, aplicando esto a una sola variable de entrada y a una de salida. Con respecto al algoritmo usado en el procesador central, deberá ser un algoritmo donde queden involucrados todos los anteriores para el control y monitoreo de cada proceso, conociendo el estado de cada una de las variables y el control de estas, además de un algoritmo adicional que le permita activar o desactivar los controladores externos ya que dicho procesador deberá manejar todos los procesos en tiempo real.

#### 6.3.2.1.- Generación de Reglas de Inferencia

Dichas reglas son generadas en base a la forma en que se desea que se maneje el comportamiento de la salida, además de la experiencia que se tenga con respecto al manejo de la variable de salida. A manera de ejemplo veremos la estructura de las reglas del proceso 1 [83], las cuales se conforman como lo indica la fig 6.2, para el resto de los procesos ver las referencias respectivas .

| | | Y | | | | |
|---|---|---|---|---|---|---|
| | | I | IC | C | DC | D |
| X | AR | NG | NM | NM | NM | NP |
| | ARC | NM | NP | NP | NP | NP |
| | C | ZE | ZE | ZE | ZE | ZE |
| | ABC | PM | PP | PP | PP | PP |
| | AB | PG | PM | PM | PM | PP |

*Fig 6.2 Matriz de Reglas de Inferencia*

Para los demás procesos se utilizará en forma independiente cada variable de entrada con su respectiva salida en cada uno de los controladores, Fig.6.3

| Var.Ent. | Var.Sal |
|---|---|
| activo | activo |
| inactivo | inactivo |

*Fig.6.3*

Donde las variables de entrada X y Y podrán tomar las siguientes posibilidades:

| Para X | Para Y |
|---|---|
| AR=ARRIBA | I=IZQUIERDA |
| ARC=ARRIBA CENTRO | IC=IZQUIERDA CENTRO |
| C=CENTRO | C=CENTRO |
| ABC=ABAJO CENTRO | DC=DERECHA CENTRO |
| AB=ABAJO | D=DERECHA |

Para la salida $\beta$:

NG=NEGATIVO GRANDE

NM=NEGATIVO MEDIO

NP=NEGATIVO PEQUEÑO

ZERO=CERO

PP=POSITIVO PEQUEÑO

PM=POSITIVO MEDIO

PG=POSITIVO GRANDE

Dichas reglas para la ordenada X están divididas en zonas donde cada zona abarca una determinada superficie, las cuales a su vez están limitadas por medio de valores números por ejemplo, la zona centro tanto para ordenada como para la abcisa estará limitada entre -1 y 1, Fig.6.4. Como podemos apreciar algunas zonas se traslapan de tal forma que habrá valores que corresponderán tanto a una zona como a otra, esto visto desde el punto de vista geomètrico, pero que tendrá como consecuencia desde el punto de vista de exactitud un incremento de esta, ya que en el momento de obtener el centroide, (Fig.6.4), tendremos un número mayor de datos a procesar, y con esto tener una mayor precisión en el valor número de la salida. Para la variable Y sucede exactamente lo mismo que para la anterior. Con el valor de las dos variables de entrada de acuerdo a la matriz de reglas de inferencia tendremos un correspondiente valor número de la salida, la cual estará dividida por sectores de tal forma que dichos sectores estarán limitados por valores angulares. Como en el caso de las variables de entrada dichos sectores se traslaparán, teniendo mayor cobertura en la variable de salida. Se debe hacer notar que entre más translapes existan se tendrá más exactitud del valor de salida, aunque el cálculo se complica, debido a que en el momento de obtener el centroide se tendrán que considerar más elementos en la ecuación utilizada para esto. Además de que desde el punto de vista de la experiencia se genera una mayor experiencia incrementando el número de reglas de inferencia ya que se aumenta el número de posibilidades. En la figura 6.2 esto se vería reflejado por el aumento de cuadros, reglas de inferencia , incrementandosé el número de posibilidades de una salida. Esto también se reflejaría como un incremento en este desde el punto de vista del aprendizaje.

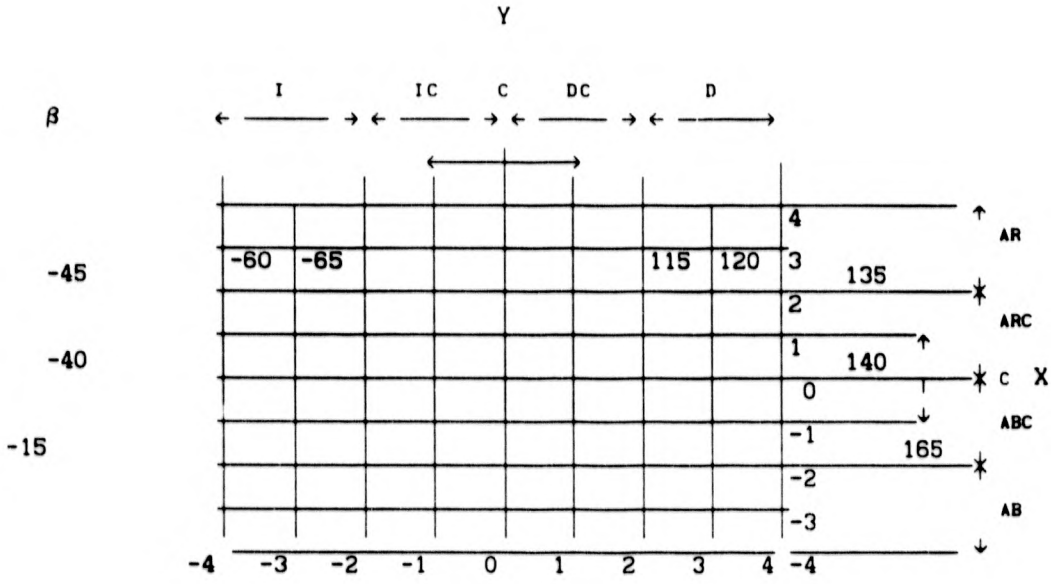



*Fig.6.4 Regiones de Reglas de inferencia*

### 6.3.2.2.- Determinación de las alturas en base a la Fusificación de las entradas

La metodología a seguir se basa en utilizar el criterio de mínimos visto en el capítulo 5, el cual indica que escojamos el mínimo valor de las alturas correspondientes al valor número de las variables de entrada. Con el valor numérico de las variables de entrada intersectamos a que altura corresponde o alturas corresponde dicho valor numérico (en el caso de que toque una zona con traslape). Por último, dichas alturas se comparan entre sí (las correspondientes alturas de una variable con las alturas correspondientes de la otra variable), Fig.6.5.

### 6.3.2.3.- Determinación del valor numérico de la salida en base a la Defusificación de las alturas de las entradas

Ya identificando, las alturas mínimas, se procede a determinar, de acuerdo a las reglas de inferencia, a que sector de salida corresponde la variable de salida asignando, la altura determinada en el punto anterior al sector correspondiente en su respectivo triángulo, de estas alturas asignadas en los triángulos de salida se compararán aquellos que pertenezcan al mismo sector. Solo se trabajará con los mínimos y ya con estos se determinará el centroide que no será otra cosa que el valor número de salida del algoritmo, Fig.6.5.

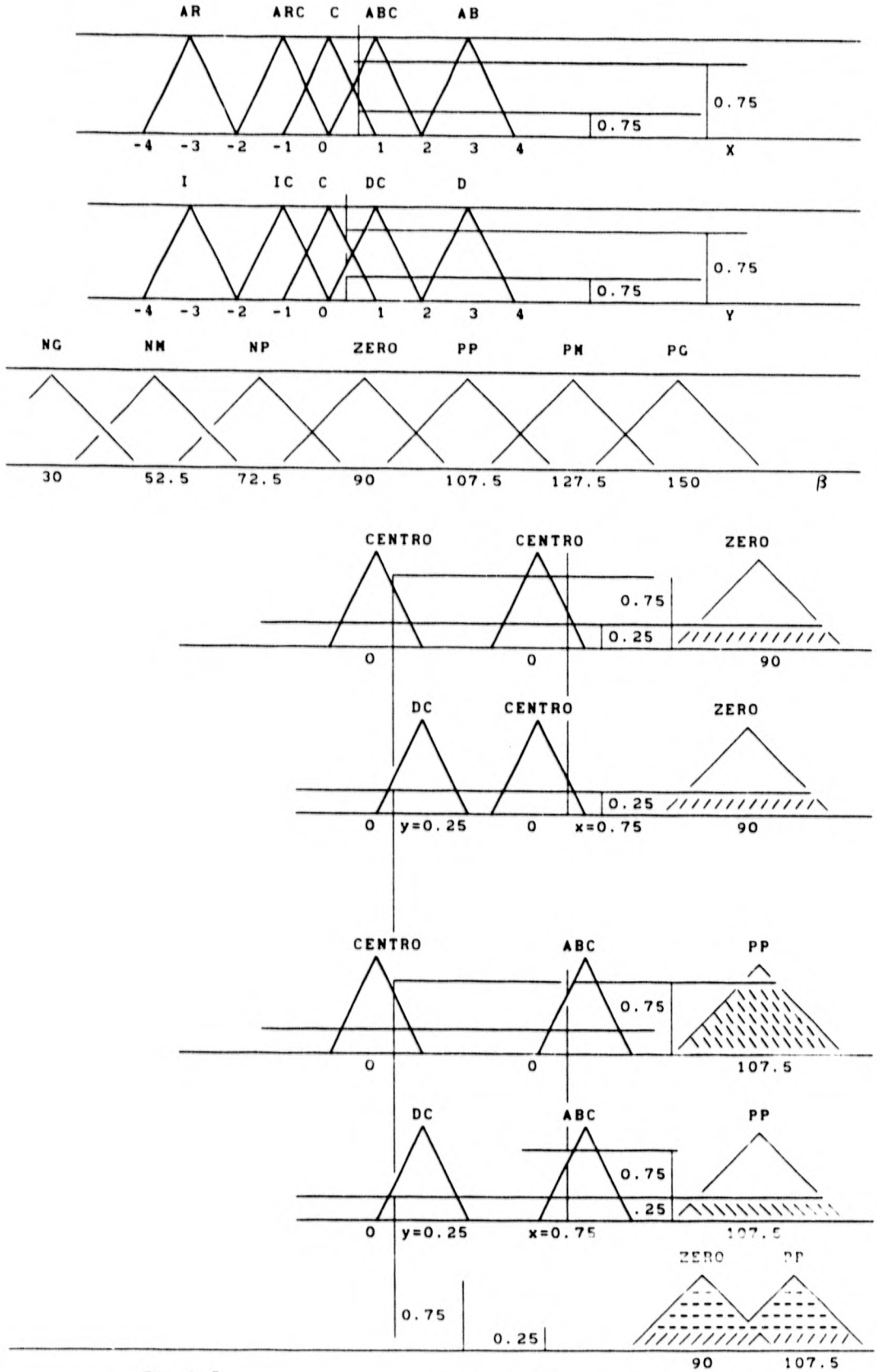



*Fig.6.5*

$$I = \frac{[0.25(90) + 0.25(107.5)]}{(0.25 + 0.25)} = 98.75$$

### 6.3.2.4.- Estabilidad del Controlador Difuso

Basicamente, la estabilidad de los controladores difusos estará determinada por la estabilidad de cada una de las reglas de inferencia que intervienen en el algoritmo difuso, de esta forma como se mencionó en el capitulo 5, se hizo en forma génerica el analisis para el resto de las reglas, y así el analisis para la estabilidad total del sistema. Para mostrar lo anterior a continuación utilizaremos una coordenada referente al proceso 1 y de este modo tendremos:

Conversión de Reglas de Inferencia a Ecuaciones en Diferencias

Tomando valores $y = .25$; $x = .75$

$L^1$ Si $x(K)$ es $A_1^c$ y $u(K)$ es $B_1^c$

$x(K) = f(X)$; $\mu(K) = f(y)$; $x' = (K+1)\theta_1$

Entonces $x'(K+1) = a_0^c + a_1^c x(K) + b_1^c \mu(K)$

$x = .75$; $a_1^c = .25$ "y" $y = .25$; $b_1^c = .75$

$$\therefore x^1(K+1) = .25\, x(K) + .75\, \mu(K)$$

$L^2$ Si $x(K)$ es $A_1^c$ y $\mu(K)$ es $B_1^{1c}$

$$\text{entonces } x^2(K+1) = a_0^c + a_1^c x(K) + b_1^{Lc} \mu(k)$$

$x = .75$; $a_1^c = .25$ "y" $y = .25$; $b_1^{Lc} = .25$

$$\therefore x^2(K+1) = .25\, x(K) + .75\, \mu(K)$$

$L^3$ Si $x(K)$ es $A_1^{Dc}$ y $\mu(K)$ es $B_1^c$

$$\text{entonces } x^3(K+1) = a_0^{Dc} + a_1^{Dc}\, x(K) + b_1^c \mu(k)$$

$x = .75$; $a_1^{Dc} = .75$ "y" $y = .25$; $b_1^c = .75$

$$\therefore x^3(K+1) = .75\ x(K) + .75\ \mu(K)$$

$L^4$ Si $x(K)$ es $A_1^{Dc}$ $\underline{y}$ $\mu(K)$ es $B_1^{Lc}$

$$\text{entonces } x^4(K+1) = a_0^{Dc} + a_1^{Dc}\ x(K) + b_1^{Lc}\mu(k)$$

$x = .75$; $a_1^{Dc} = .75$ "y" $y = .25$; $b_1^{Lc} = .25$

$$\therefore x^4(K+1) = .75\ x(K) + .25\ \mu(K)$$

Donde $L^1$ sera:

Si $x(X)$ es $A_1^1$ "y".... "y" $x(K-n+1)$ es $A_n^1$ "y" $\mu(K)$ es $B_1^1$ "y" ..."y" $\mu(k-m+1)$ es $B_m^1$ donde:

$x(X) = [x(K), x(K-1)...(K-n+1)]^T$

De esta forma generaremos las ecuaciones en diferencias correspondientes a cada una de las reglas y generaremos las matrices $_1A$ y $_1B$ vistas en el cápitulo anterior.

$$A_1 = \begin{bmatrix} a_1^1 & a_2^1 & \dots & a_{n-1}^1 & a_m^1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} b^1_1 & b^1_2 & \dots & b^1_{n-1} & b^1_m \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

Ya con esto se debe tener primero que garantizar la estabilidad de los subsistemas $L_1$, $L_2$ etc. de tal forma que exita P para $A_1$, $A_2$ etc. si se cumple, entonces analizar estabilidad para el sistema total de tal forma que exista P para $A_1$ $A_2$ $A_n$ etc.: si se cumple estabilidad para sistemas y subsistemas el sistema será globalmente asintoticamente estable [31].

### 6.3.2.5.- Autosintonización del Controlador Difuso

Como se mencionó en el capítulo 5 el analísis de autosintonización se realizó en base a párametros estimados, la autosintonización del controlador estará determinada por la autosintonización de cada una de sus reglas que en forma genérica se planteó en el capítulo 5, aquí analizaremos la autosintonización de una de las reglas de inferencia, siendo una autosintonización de primer orden ya que solo se tomará la diferencial de la señal obtenida y la señal estimada, esto funciona de la siguiente forma: La variable de salida $\beta$ despuès del proceso de defuzzficación tendrá un valor numérico el cual será nuestra señal de salida, esta variable se comparará con dos valores numéricos que serán nuestra señal estimada, dichos valores serán el centro de dos sectores uno a la derecha y otro a la izquierda del valor de salida, de tal forma que un valor será mayor y el otro menor. Con esto se trata de encontrar las dos diferencias como resultado de las comparaciones entre el valor de salida y los valores mencionados, la diferencia menor en valor absoluto es la que se tomará en cuenta ya que nuestra señal estará más cerca del valor correspondiente y tomará éste como la verdadera salida y de esta forma autosintonizandosé, con el objeto de dar el valor exacto del centro del sector correspondiente ya que el sistema sólo tiene contemplados los centros de los sectores para el proceso de defuzzificación. En la figura 6.6 se podrá apreciar la comparación de la regla de salida N, siendo N la ubicación de la regla dentro del cuadrante donde se encuentran los sectores. Donde d1 y d2 representan los resultados de las comparaciones antes mencionadas.

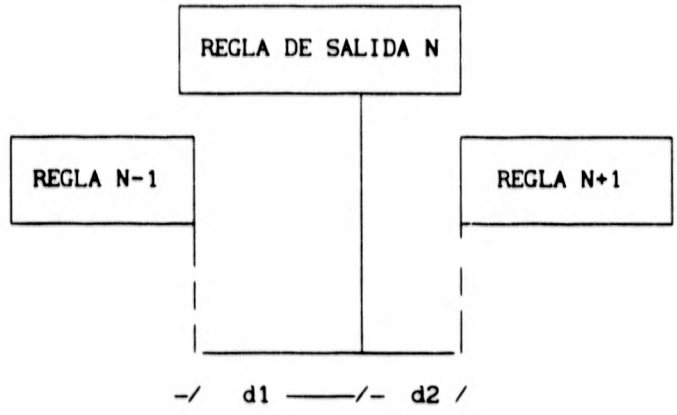



*Fig 6.6 Autosintonización de primer orden en la salida*

6.3.2.6.- Algoritmo de Monitoreo y Control residente en el Procesador central

Dicho algoritmo es un algoritmo constituido por algoritmos exactamente igual que los residentes en los controladores externos de tal forma que comparán el valor entrante con un conjunto de valores previamente establecidos dentro del procesador central donde los valores establecidos deberán mantener un determinado comportamiento, es decir, conoceremos de antemano un comportamiento aproximado de las variables de entrada como es característica de los algoritmos difusos. Es decir, se tendrá en cuenta el rango ó bien los valores que pueden tomar las variables involucradas en determinado proceso controlado este por un controlador específico. Además de contar con un algoritmo de comunicación y un algoritmo de habilitación y deshabilitación de los controladores externos para quien un determinado momento pueda controlar los procesos dicho procesador central ó bien en forma manual, Fig 6.7, y sólo utilizar a los controladores externos como monitores.

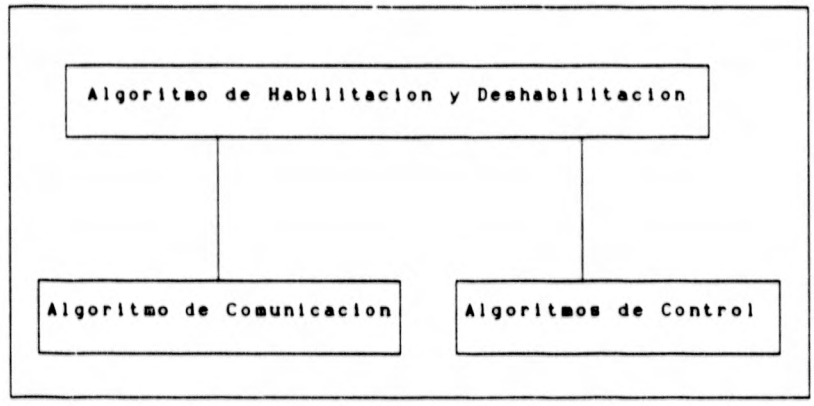



*Fig.6.7 Algoritmo en el Procesador Central*

## 6.4.- PARALELISMO EN CONTROL DISTRIBUIDO

En este subtema identificaremos el paralelismo que existe a nivel de control distribuido [11] y a nivel de controladores externos en esta tesis [82], [83], [84]. De esta forma podemos mencionar que desde el punto de vista Sistema Distribuido será una Arquitectura del tipo de Red de Procesadores y Cálculo Distribuido en un Arreglo Vectorial donde los controladores que constituyen el sistema tienen las características de no estar estrechamente acoplados, no comparten memoria, dichos controladores se comunican con el Procesador Central. Con respecto a los controladores dos de estos serán arquitecturas del tipo concurrente (de bus compartido)[85], ya que se comunicarán entre ellos formando un arreglo paralelo en un nivel con respecto a dos de los CPUs que constituyen dicho controlador y estos dos CPUs formarán un arreglo secuencial con respecto a un tercer CPU que también es parte constitutiva del controlador, los tres CPUs compartiendo un mismo bus de datos [83] y [84]. Otro de los controladores será de tipo pipeline, estando constituido este por tres CPUs que se transmitirán datos secuencialmente [82]. Y por último se tendrá en el cuarto controlador un monoprocesador que no será más que un control on-off, tendrán una comunicación cada uno de los controladores externos con el monitor central a través de un puerto serial, teniendo cada uno de ellos un determinado código para que el procesador central así como entre ellos se diferencien y no sea causa de posibles errores en la comunicación, esto se muestra en la Fig.6.8.a.

En este diseño un controlador pueden en un momento tomar las tareas de otro debido a una posible falla, ó bien al responder a un comando del procesador central, aunque no es un sistema tolerante a fallas se puede implementar el software correpondiente residente tanto en el monitor central como en los controladores. Además de poder conectarse a otros sistemas distribuidos compatibles con este sistema a través de sus controladores, y de esta forma tener un sistema versátil.

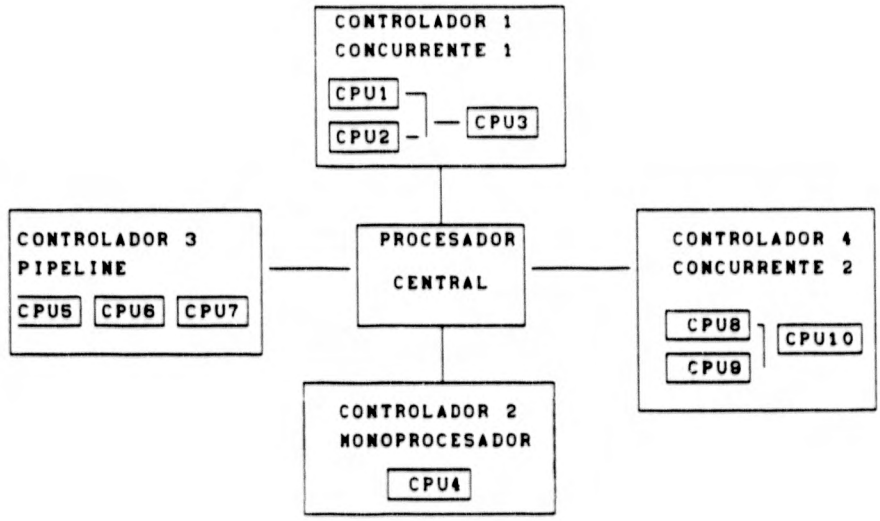



*Fig.6.8.a. Arquitectura tipo red de Arreglo Vectorial*

## 6.5.- ARQUITECTURA DEL SISTEMA DISTRIBUIDO

Del sistema distribuido, en cuanto a su arquitectura, podremos mencionar de acuerdo a las caracterizticas mencionadas, que la PC tendrá prioritizados a los procesos de tal forma que si más de un controlador requiere mandar información a la PC al mismo tiempo, está atenderá al proceso de mayor prioridad, y posteriormente al ó a los de menor prioridad, de esta forma la PC al sensar diferentes procesos tendrá la información necesaria para en un momento dado mandar determinadas órdenes a los controladores en el caso de que sea necesario Fig.6.8.b. Dicha PC estará operada por un sistema operativo en tiempo real (SOTR) que administrará los dispositivos de la computadora, tomando en cuenta la

planeación de la aplicación de programas, escrituras a archivos o discos y transmisión de datos a través de una red. El sistema operativo (SOTR) tiene las siguientes caracteristicas [80] :

- Procesamiento Multitarea
- Manejo de Interrupciones planeadas preventivas
- Rápido Switcheo de Contextos

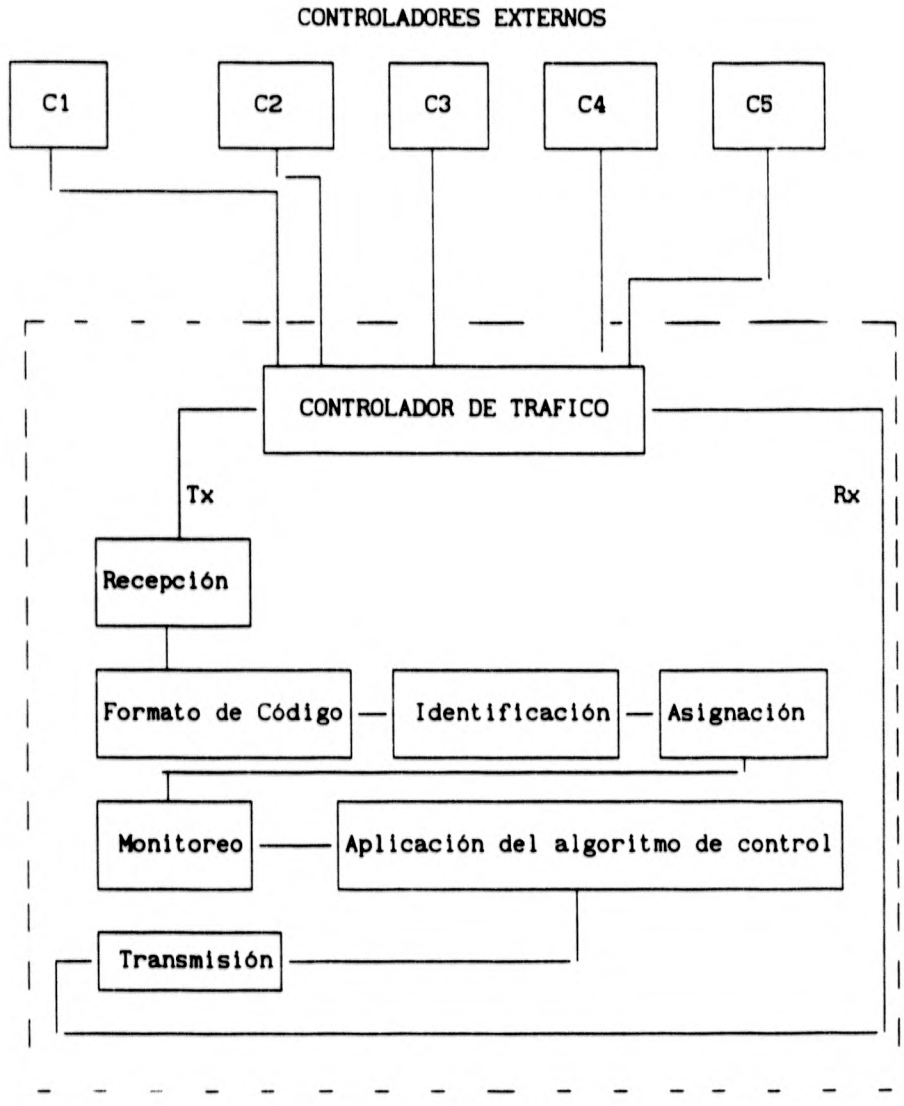



*Fig. 6.8.b Arquitectura del Sistema Distribuido*

El SOTR puede estandarizar para conocer las necesidades de cada aplicación, como configurar desde un elemento básico hasta una red de usuarios. Dicho sistema logra esto en base a una arquitectura de microkernel (microelemento) y a sus mensajes basados en interprocesos de comunicación. Ya que se trabaja en modo multitarea y permite interactuar entre los procesos de cooperación. Dicho microkernel estará constituido por un grupo de cuatro procesos de cooperación y un nodo de control los cuales serán los siguientes Fig.6.9 :

-Administración de Procesos

-Administración de Dispositivos

-Administración de Sistemas de Archivos

-Administración de Redes

-Nodo de Control: El cual tiene dos funciones, el de pase de mensajes y el de planeación. Donde se maneja el direccionamiento de los mensajes y la planeación de cuando un proceso cambia de estado como resultado de un mensaje o una interrupción, respectivamente.


*Fig.6.9 Arquitectura del microkernel (microelemento) del Sistema Operativo QNX*

Dicho procesador central proporcionará en la pantalla una consola de control y monitoreola situación de cada uno de los procesos proporcionando datos en forma gráfica y en forma numérica, así como la posibilidad de interrupción en un momento dado para pedir nueva información y/ó transmisión de órdenes [81].

### 6.5.1.- Arquitecturas de Controladores

Los controladores que constituyen dicho sistema tendrán dentro de su programación algoritmos de control difuso para controlar cada uno de los procesos, basandose en reglas de inferencia las cuales nos permiten controlar los procesos del sistema sin tener que usar algún modelo matemático del proceso a controlar, como se había hecho mención. Los controladores como tal estarán constituidos por un chip de determinadas características (microcontrolador ó microprocesador), o bièn estarán constituidos por más de un chip trabajando en paralelo, esto dependerá de la naturaleza del algoritmo difuso y del proceso a controlar. Aunque basicamente la decisión se basa en el factor tiempo del proceso, y de esta forma utilizar chips trabajando en paralelo los cuales incrementarán en conjunto la rápidez del algoritmo.

Para los procesos se utilizarán Microcontroladores Motorola MC68HC11, PICS16C74 y /ó, Microcontroladores Intel 8031, dependiendo del proceso. Las características de velocidad e integración de unidades funcionales y costo permite tener desde el punto de vista hardware arquitecturas pequeñas, mientras que el segundo se seleccionó por la simplicidad, velocidad y costo del primero. Esta selección se hizo de acuerdo a las características de cada uno de los procesos a manejar, ya que dichos procesos son relativamente lentos, aunque el proceso 1 es el más rápido de ellos, la descripción detallada de cada uno de los controladores se hace en las respectivas referencias [82], [83] y [84]. Los controladores tienen su diagrama de tiempos y en el cual se puede apreciar como se manejan las tareas, el diagrama de las latencias, la eficiencia y el ancho de banda, así como su diagrama de flujo algoritmico,[82], [83] y [84].
Con respecto al proceso 1, la arquiterctura que conforma el controlador para el control y monitoreo de estos será una arquitectura en paralelo de tal forma que dicha arquitectura estará constituida por dos microcontroladores HC11 y un microcontrolador 8031. Los HC11 harán el proceso de fuzzificación tomando cada HC11 una de las variables de entrada y entregando los datos correspondientes a las variables fusificadas al 8031, siendo este el encargado de hacer el proceso de defusficación. Mientras se desarrolla el proceso de defusificación por el 8031, entregando la variable de salida, los HC11 desarrollarán la fusificación, teniendo los datos correspondientes de ésta, disponibles en cuanto el 8031 los requiera, de tal forma que si los HC11 tienen datos disponibles, y el 8031 no está listo los HC11 procederán a fusificar las variables nuevamente, en la siguiente muestra, como se puede apreciar en la figura Fig.6.10.

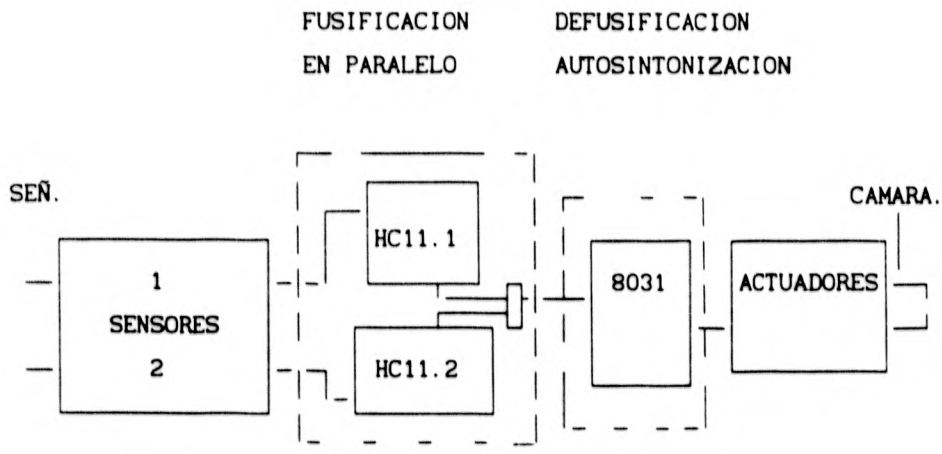



*Fig.6.10 Arquitectura del controlador 1*

En el caso del proceso 4 la arquitectura es similar, a diferencia de que en lugar de HC11 se usan pics y en lugar del 8031 un HC11 [84]. Igual que en el caso anterior se pueden apreciar los párametros correspondientes a este tipo de arquitectura, así como el tipo de procesamiento paralelo utilizado [83]. De manera similar para el proceso 2 se tendrá un controlador constituido por un microcontrolador 8031 como lo muestra la Figura 6.11, teniendo las características de latencias etc, como en los controladores anteriores [83].

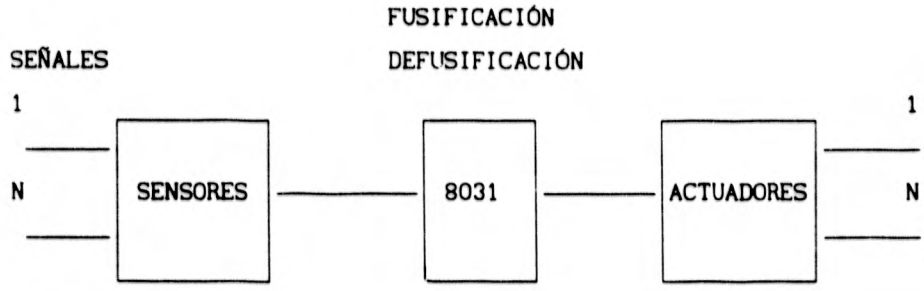



*Fig.6.11 Arquitectura controlador 2*

Para el caso del proceso 3 la arquitectura utilizada, es una arquitectura tipo pipeline donde el controlador esta constituido por dos HC11 y un microprocesador 8088, donde el primer HC11 recibe las señales analógicas y realiza la fusificación asignando grados de pertenencia, transmitiendo al segundo HC11 los datos para que este microcontrolador realice la defusificación, y de esta forma el 8088 realiza la autonsintonización y se comunica con el monitor central, como lo muestra la fig..6.12.

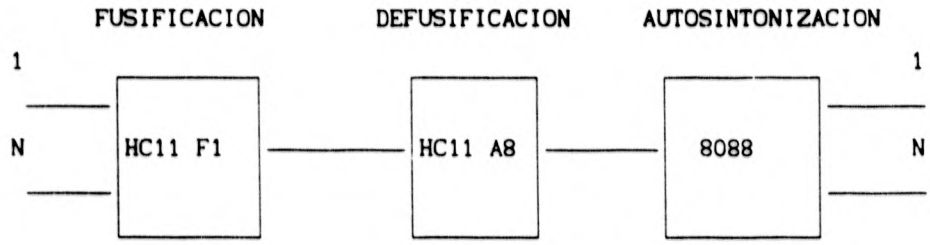



*Fig. 6.12 Arquitectura del controlador 3*

6.5.1.1.- Arquitectura del Controlador de Tráfico Serie-Paralelo (CTSP)

Este circuito es la interfaz entre el procesador central y los controladores externos y tiene las siguientes características principales [82]:

- Dar paso a la información proveniente del controlador seleccionado por el procesador central.

- Conmutar de Procesador a Controladores y viceversa para el control de los procesos.

-Interfaz entre Controladores y Procesador Central.

Dicho controlador de tráfico estará constituido por un bloque de control, un bloque de acoplamiento RS232, un bloque transmisión serial, y un bloque de conmutación, a continuación se mencionan la finalidad de cada uno de ellos:

-El bloque de control es operado por el Procesador Central el cual maneja el resto de los bloques.

-El bloque de acoplamiento RS232, es para conectarse al puerto serial del Procesador Central.

-El bloque de Transmisión Serial, permite elegir que controlador estará en comunicación con el Procesador Central.

-El bloque de conmutación sirve para conectar el proceso ó bien al Procesador Central ó bien a los Controladores.

### 6.5.2.- Protocolos

Con referencia a este punto podemos mencionar que el protocolo de comunicacion será un protocolo simple que permita probar el control distribuido con todos los controladores externos involucrados, teniendo en cuenta que para un uso industrial se utilizará un prótocolo más adecuado a la aplicación se recomienda el uso de un prototipo industrial que permita empaquetar información de tal forma que tengamos una transmisión más eficiente. El prótocolo usado será un formato a 9600 bauds, bit de inicio, paquete de datos de 8 bits, y un bit de paro (stop), sin tener bit de paridad, este formato incluye códigos que permitan identificar hacia que controlador se debe enviar la información, así como de que controlador proviene la información, ya que tanto el procesador central como los controladores externos tendrán los códigos correspondientes. El código utilizado no tiene ningún formato de encriptado pero sí posee la característica de poder identificar y verificar mediante algoritmos difusos de comparación la información recibida como lo hacen algunos prótocolos comerciales, cabe mencionar que para una aplicación industrial, con productos de tipo comercial, se usarán protocolos que sean más versátiles en cuanto a su manejo de tal forma que se pueda empaquetar información codificada dentro de la cual tendrá definido su destino así como los códigos de acceso correspondientes, encriptación, texto etc. A continuacion se describirá la estructura de los prótocolos de comunicación. El proceso 1 tendrá asignado el código 01H, para el envió de información hacia el procesador central y los códigos 03h y 09h para la desactivación y reactivación del controlador 1, para el proceso 2 los códigos 02H, 04h y 0Ah. Para el proceso 3 serán los códigos 05, 07 y 0B, para el proceso 4 serán 06, 08, y 0C respectivamente actuando de igual modo cada uno en su respectivo controlador. Para el proceso n los códigos n, p, y q donde n, p y q significan los enèsimos números en hexadecimal correspondientes al enèsimo controlador. De esta forma, se transmite el código, ya sea del procesador central o hacia el procesador central, en el formato antes mencionado, ya que los controladores al recibir el código correspondiente responden con el mismo número de código contestando al procesador central, seguido de los datos correspondientes. Dichos datos acomodandolos en forma de pila para poder comparar con datos residentes cuando van hacia el procesador central y así estar en posibilidad de aplicar el algoritmo difuso para poder determinar la autenticidad y exactitud de los datos y con esto en una escala numérica y una gráfica

poder monitorear los procesos y en un momento dado poder mandar una orden de control si así se amerita, al o a los controladores externos teniendo como base un algoritmo en el procesador central igual a los residentes en los controladores externos, esto se muestra en la Fig. 6.13.

El formato de los protocolos será de la siguiente forma:

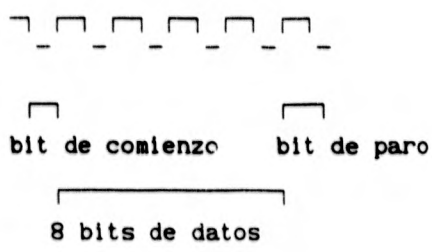



*Fig.6.13 Formato de protocolos de comunicación*

## 6.6.- DESCRIPCIÓN DE LA PROGRAMACIÓN

Dentro de los programas utilizados, a nivel de controladores se tendrá en forma de bloques la secuencia que siguen los programas, primeramente se describirá el que utiliza el controlador simple, como lo indica la figura 6.14. En la primera parte de la programación se fusifican las variables de entrada, en la segunda parte se defusfican, y en la tercera se autosintonizan.

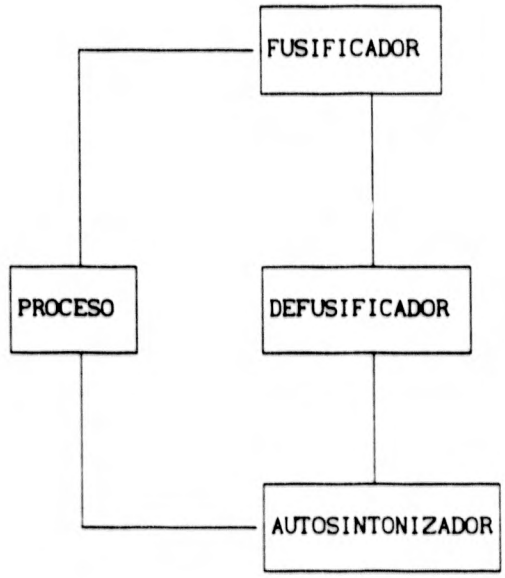



*Fig.6.14 Controlador Simple*

En el caso de los controladores paralelo la estructura de programación quedaría, como lo indica la Figura 6.15, donde se pueden apreciar los bloques antes mencionados.

Con respecto a la programación del procesador central se tendrá la secuencia para determinar la aplicación del algoritmo difuso en el procesador central Fig.6.16.

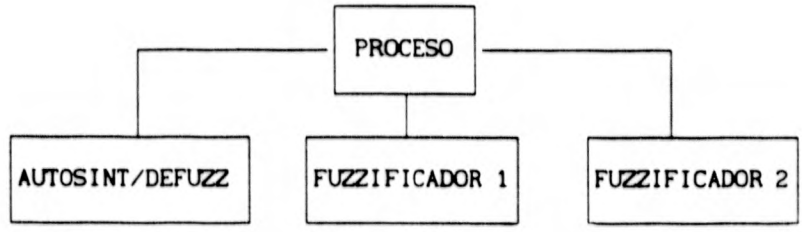



*Fig.6.15 Controlador Paralelo*

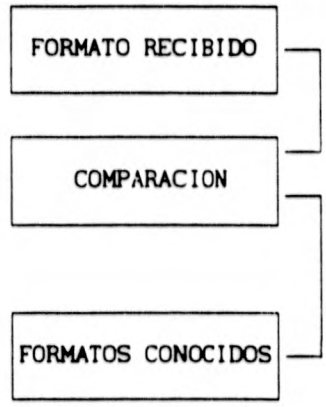



***Fig.6.16 Secuencia de aplicación del algoritmo difuso en el procesador central***

De tal forma que se identifique secuencialmente formando un conjunto de reglas de acuerdo al siguiente esquema:

- Si se conoce el código entrante se comparará con todos los códigos conocidos uno a uno en orden de menor a mayor.

- Si el resultado es igual a uno de los códigos establecidos se pasará a la siguiente fase.

-Si el resultado fue diferente a todos los códigos se ignorará dicho código y seguirá normalmente su labor el control distribuido.

- Al ser igual a uno de los códigos correspondientes el sistema procederá a verificar los datos almacenados con respecto a los datos nuevos que este recibirá en ese instante con respecto al controlador externo en cuestión. Entonces el procesador central procederá a determinar si aplica algún comando de control o simplemente monitorea, de a acuerdo a las reglas y algoritmos que se tienen implantados en los controladores externos, Fig.6.17

CONJUNTO DE REGLAS DE COMPARACION

| | Cod1 | Cod2 | Cod3 | Cod4 | Cod5 |
|---|---|---|---|---|---|
| Cod1 | C1 | S | S | S | S |
| Cod2 | S | C2 | S | S | S |
| Cod3 | S | S | C3 | S | S |
| Cod4 | S | S | S | C4 | S |
| Cod5 | S | S | S | S | C5 |

*Fig.6.17*

## 6.7.- CONCLUSIONES

Podemos mencionar que por sus características el sistema propuesto es ideal para diferentes tipos de procesos ya que usamos algoritmos difusos, y es muy versátil por las diferentes familias de microprocesadores y microcontroladores de los cuales esta constituido, además de las características del sistema operativo que usa.

De esta forma se puede tener el mayor grado aplicación posible y con esto integrar este tipo de control a equipos comerciales, reduciendo el costo de instalación.

# 7.- PRUEBAS Y VALIDACIÓN DEL SISTEMA

## 7.1.- INTRODUCCION

En este cápitulo describiremos las pruebas hechas al sistema así como a cada una de sus partes indicando los parámetros que sirvieron como referencia para determinar el desempeño del sistema. Se puede comentar que para poder determinar las especificaciones del sistema es necesario conocer los siguientes parámetros:

- Versatilidad

Así párametros de relevancia que reflejan el desempeño de cada uno de los controladores, así como el desempeño en general del sistema.

- Latencias de cada arquitectura así como del sistema global.

- Tiempo de muestreo, a nivel de controladores como a nivel sistema.

- Tiempo de respuesta de cada uno de los controladores con respecto al sistema.

- Velocidad de transmisión entre el procesador central y cada uno de los controladores.

-Diagrama de tareas

- Tiempos de procesamiento de cada variable de entrada así como de salida y de cada interproceso.

Cabe hacer notar que dicho sistema fue probado utilizando señales provenientes de generadores de funciones de tal forma que pudiesemos manipular a voluntad los parámetros de las señales de entrada y de esta forma estar en posibilidad de estudiar las variaciones tanto de las señales de salida como la interacción entre los controladores y el sistema, y así poder conocer los parámetros antes mencionados. Los parámetros antes mencionados se encuadrarán con respecto a cada controlador [82], [83] y [84] y posteriormente se hará un encuadre a nivel sistema global incluyendo el sistema operativo utilizado [80] y [81].

## 7.2.- VERSATILIDAD

Este aspecto abarca la grado de aplicabilidad que posee el sistema, y para esto se analizan tanto el número de variables de entrada como el de salida, la naturaleza y velocidad del proceso. Para esto se muestran esquemas donde se pueden evaluar los rangos hacia donde se moverían otro tipo de variables tanto de entrada como de salida, y de esta forma ubicar otro tipo de aplicación.

### 7.3.- LATENCIAS DE CADA ARQUITECTURA ASI COMO DEL SISTEMA GLOBAL

En este punto se analiza la cantidad de tiempo que cada controlador que permanece ocioso, así como la cantidad de tiempo que este trabaja y de esta manera verificar cual es la cantidad de tiempo muerto que tiene el sistema en general.

### 7.4.-TIEMPO DE MUESTREO, A NIVEL DE CONTROLADORES COMO A NIVEL SISTEMA

Con referencia a este punto se determina cuantas variables se pueden procesar en cada controlador y por lo tanto cuantas a nivel de todo el sistema.

### 7.5.-TIEMPO DE RESPUESTA DE CADA UNO DE LOS CONTROLADORES CON RESPECTO AL SISTEMA

Se analiza cuanto tiempo tardan cada uno de los controladores del sistema en responder a los comandos del monitor central, así como la respuesta de los controladores, de tal forma que se pueda comunicar dicho monitor con todos sus controladores y pueda monitorear todos los procesos involucrados.

### 7.6.-VELOCIDAD DE TRANSMISION DEL PROCESADOR CENTRAL Y DE CADA UNO DE LOS CONTROLADORES

Este punto podría considerarse parte del punto anterior ya que este nos dice a que velocidad se comunican los controladores con el monitor central.

### 7.7.- TAREAS

Esto se refiere a cada una de las fases de todos los procesos involucrados en el sistema, Figura 7.1.

### 7.8.- TIEMPOS DE PROCESAMIENTO DE CADA VARIABLE DE ENTRADA ASI COMO DE SALIDA Y DE CADA INTERPROCESO

Aquí basicamente se toman los tiempos de cada una de las fases del proceso.

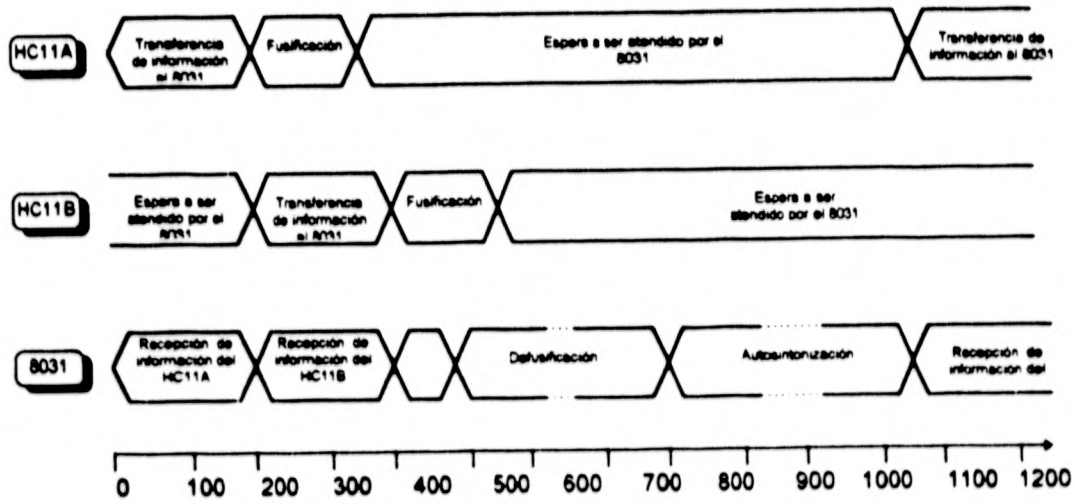



*Fig. 7.1. Cronograma de las etapas del proceso del controlador 1*

En la figura 7.1 aparece el cronograma del controlador 1 en donde se aprecia una secuencia de procesos entre una iteración y otra [83].

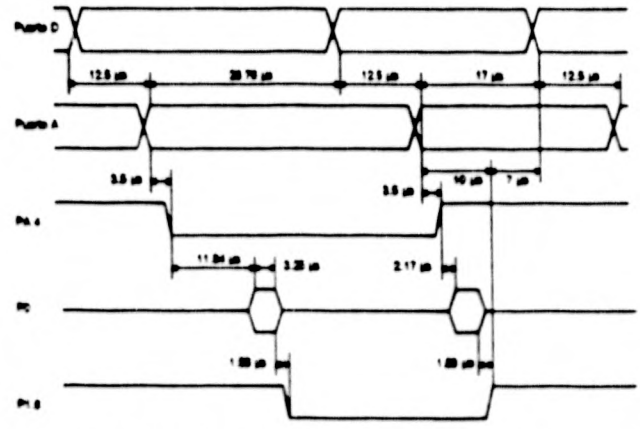
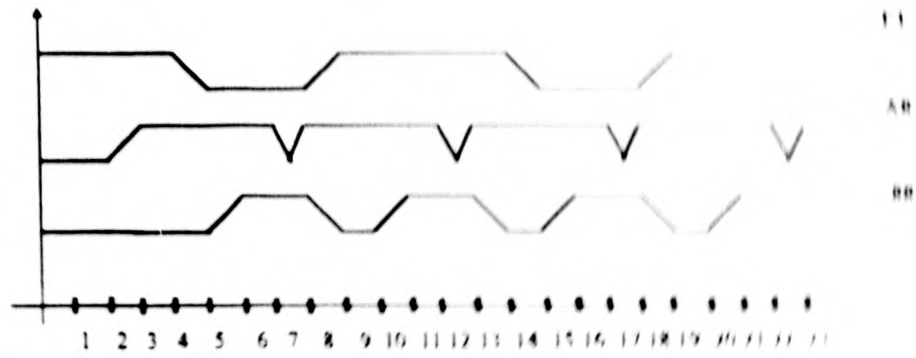
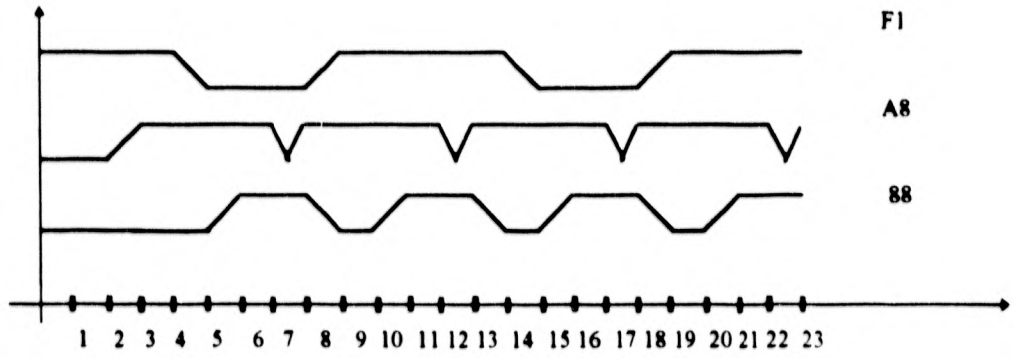


*Fig. 7.2 Cronograma de las señales involucradas entre el microcontrolador 8031 y uno de los microncontroladores HC11 en el controlador 1*

En la figura 7.2, del controlador 1, se aprecian los primeros 6 bits de datos que provienen del puerto D y los dos más significativos del puerto A del HC11, estos van al puerto 0 del 8031 el cual durante una lectura a memoria externa de datos captura la información. El bit PA.4 del HC11 indica que los datos en los puertos A y D son válidos y los bits P1.6 y P1.7, avisan a los respectivos HC11 que se leyó el puerto [83].


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




1) F1) -Realiza la búsqueda de los datos para fusificación
A8) -Pregunta al F1 si ya tiene un dato
88) -Pregunta si el A8 tiene un dato

2) F1) -Realiza la determinación de los minimos
A8) -Pregunta al F1 si ya tiene un dato
88) -Pregunta si el A8 tiene un dato

3) F1) -Avisa que tiene datos de fusificación para $T_1$
A8) -Identifica que ya hay un dato a la entrada
88) -Pregunta si el A8 tiene un dato

4) F1) - Envia los datos relacionados con $T_1$
A8) - Recibe los datos relacionados con $T_1$
88) - Pregunta si el A8 tiene un dato

5) F1) - Avisa que tiene datos de fusificación para $P_1$
A8) -Inicia el proceso de defusificación para $T_1P_1$
88) - Pregunta si el A8 tiene un dato

6) F1) - Avisa que tiene datos de fusificación para $P_1$
F1) - Avisa que tiene un dato defusificado
88) - Identifica que ya hay un dato en la entrada

7) A8) - Avisa que tiene datos de fusificación para $P_1$
F1) - Transmite los datos defusificados para $T_1$
88) - Recibe los datos relacionados con $T_1$

8) A8) - Avisa que tiene datos de fusificación para $P_1$
F1) - Identifica que ya hay un dato en la entrada
88) - Realiza autosintonizacion y manipulacion de válvula $T_1$

9) A8) - Envia los datos relacionados con $P_1$
F1) - Recibe los datos relacionados con $P_1$
88) - Pregunta si el A8 tiene un dato

10) A8) - Realiza la búsqueda de los datos para fusificar
F1) - Inicia el proceso de defusificación para $P_1T_1$
88) -Pregunta si el A8 tiene un dato

11) A8) - Realiza la determinación de los minimos
F1) - Avisa que tiene un dato defusificado
88) - Identifica que hay un dato en la entrada

12) A8) - Avisa que tiene un dato de fusificación para $T_2$
F1) -Trasmite los datos defusificados para $P_1$
88) - Recibe los datos relacionados con $P_1$

13) A8) - Avisa que tiene un dato de fusificación para $T_2$
F1) - Identifica que ya hay un dato en la entrada
88) - Realiza autosintonizacion y manipulación de válvula $P_1$

14) A8) - Envia los datos relacionados con $T_2$
F1) - Recibe los datos relacionados con $T_2$
88) - Pregunta si el A8 tiene un dato listo

15) A8) - Avisa que tiene un dato listo
F1) - Inicia el proceso de defusificación para $T_2$
88) - Pregunta si el A8 tiene un dato listo

16) A8) - Avisa que tiene un dato listo
F1) - Avisa que tiene un dato defusificado para $T_2$
88) - Identifica que hay un dato a la entrada

17) A8) - Avisa que tiene un dato listo
F1) - Envia los datos de defusificación para $T_2$
88) - Recibe los datos relacionados con $T_2$

18) A8) - Avisa que tiene un dato listo
F1) - Identifica que hay un dato a la entrada
88) - Realiza autosintonización y manipulación de válvula $T_2$

19) A8) - Envia los datos relacionados con $P_2$
F1) - Recibe los datos relacionados con $P_2$
88) - Pregunta si el A8 tiene un dato listo

20) F1) - Inicia el proceso de defusificación para $P_2$
88) - Pregunta si el A8 tiene un dato

21) F1) - Avisa que tiene un dato defusificado $P_2$
88) - Identifica que hay un dato a la entrada

22) F1) - Envia los datos defusificados para $P_2$
88) - Recibe los datos relacionados con $P_2$

23) 88) - Realiza autosintonización y manipula Válvula

*Fig. 7.3. Cronograma de las etapas del proceso del controlador 3*

En la figura 7.3 se aprecian las diferentes etapas del proceso de una iteración en el controlador 3 [82].

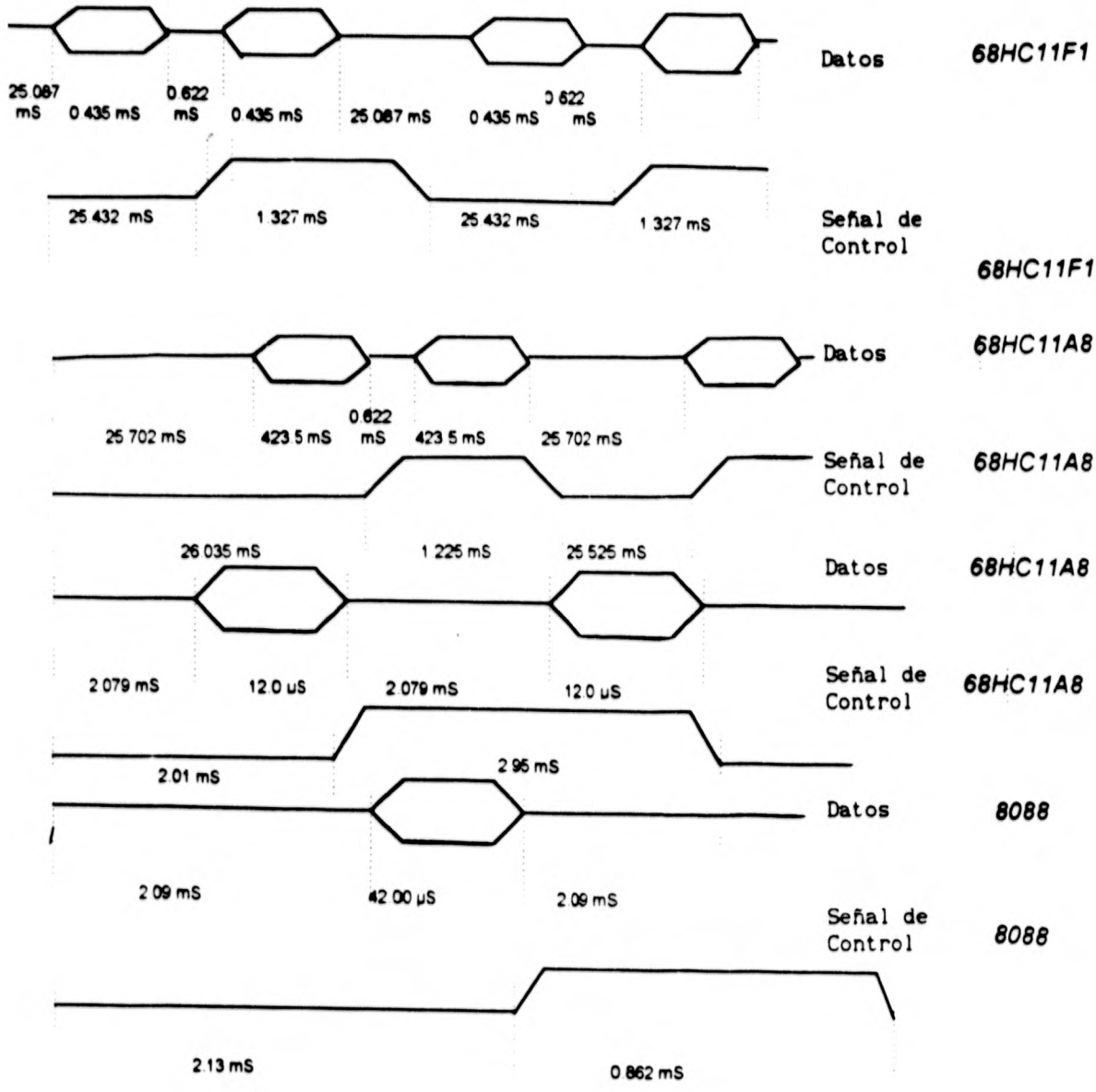
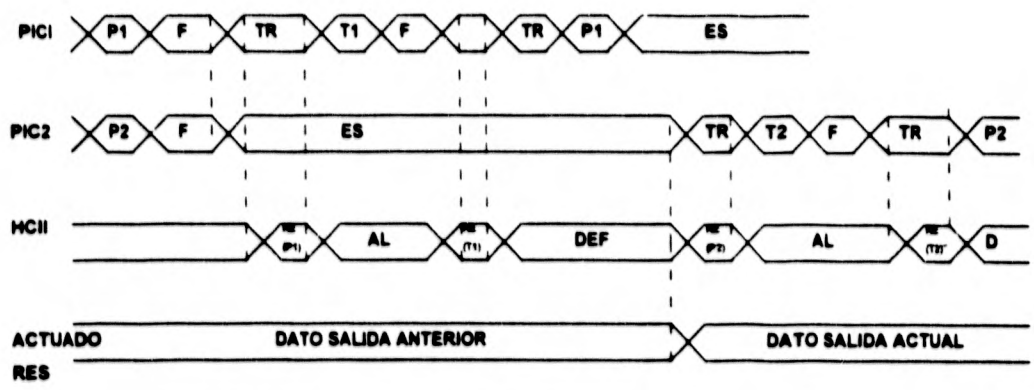



*Fig. 7.4 Cronograma de las señales involucradas del controlador 3*

En la figura 7.4 se aprecian las señales de transferencia de datos y sincronización entre los microcontroladores del controlador 3 [82].




P1 Conversión A/D de Presión 1
P2 Conversión A/D de Presión 2
T1 Conversión A/D de Temp. 1
T2 Conversión A/D de Temp. 2
F Fusificación
TR Transfiere los primeros cinco datos

ES Espera ser atendido
RE Recibe datos
DEF Defusifica
AL Espera los cinco datos restantes

*Fig. 7.5. Cronograma de las etapas del proceso del controlador 4*

En la figura 7.5 aparece el cronograma del controlador 4 en donde se aprecia una secuencia de procesos entre sus microcontroladores, en una iteración y otra [84].

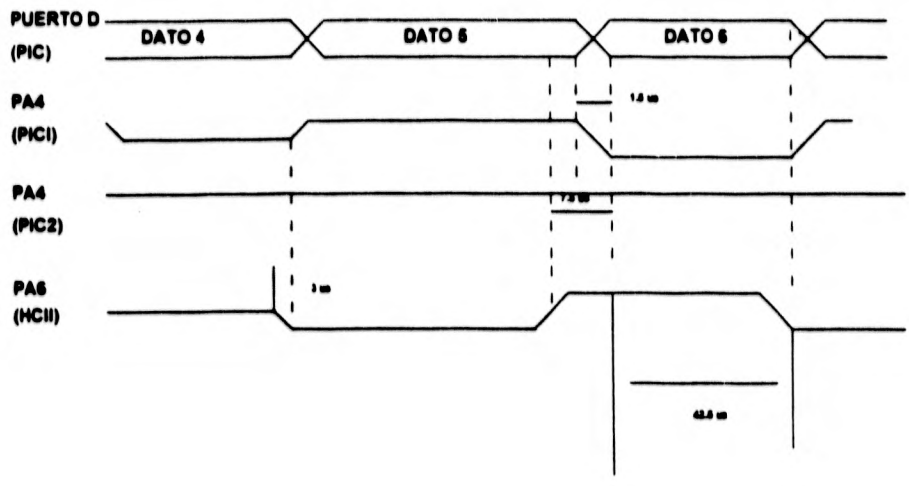



*Fig. 7.6 Cronograma de las señales involucradas del controlador 4*

En la figura 7.6 se aprecian las señales de transferencia de datos y sincronización entre los microcontroladores del controlador 4 (84).

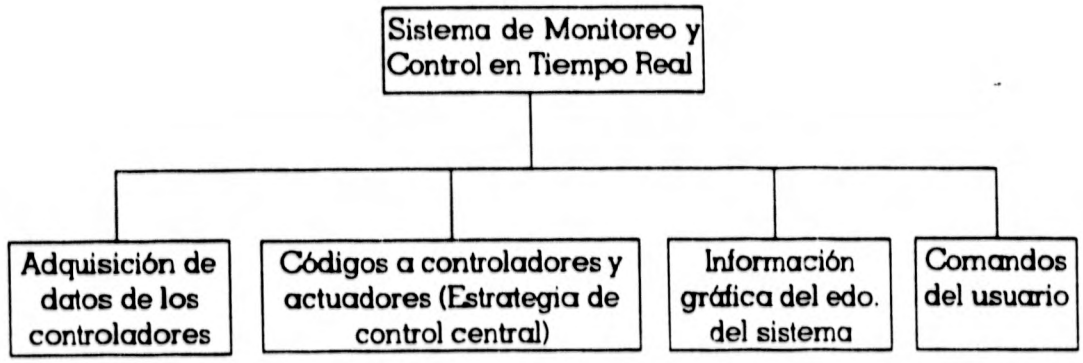
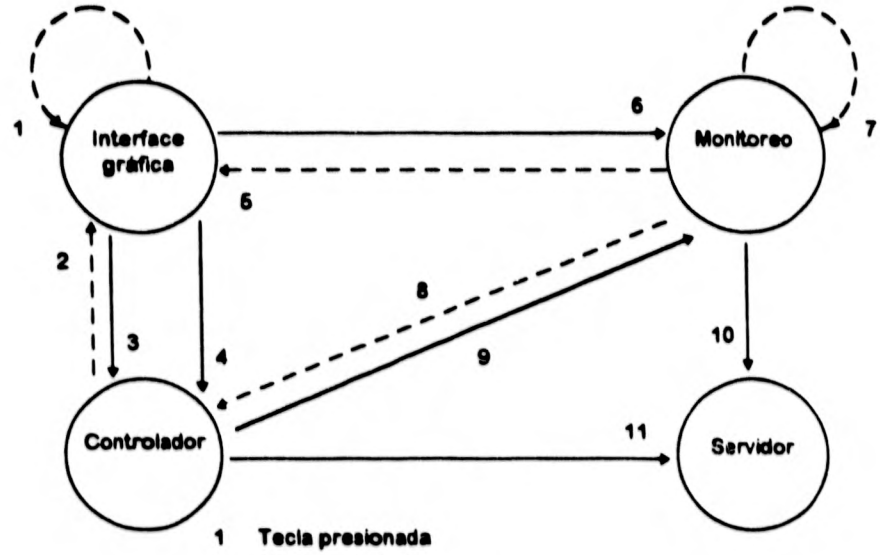



*Fig. 7.7. Funciones del sistema operativo*

En la figura 7.7 a manera de bloques se muestra las diferentes funciones del sistema de monitoreo y control en tiempo real [81].




1 Tecla presionada
2 Acciones tomadas
3 Solicita información sobre acciones tomadas
4 Solicita la ejecución de una petición del usuario
5 Cambio de estado
6 Solicita información sobre cambio de estado
7 Timer
8 Cambio de estado
9 Solicita información cambio de estado
10 Solicita muestreo
11 Solicita acción sobre controladores o actuadores

Mensajes tipo proxy

Mensajes que bloquean al proceso en espera de una respuesta del receptor

*Fig. 7.8. Mensajes entre diferentes etapas del sistema operativo*

Diferentes mensajes entre diferentes etapas del sistema de monitoreo en tiempo real [81].

Tiempo de muestreo del puesto central y tiempo de consumo de cada uno de los controladores. Así como el tiempo total del proceso.

| Controlador 1 | Controlador 2 | Controlador 3 | Controlador 4 | M. Central |
|---|---|---|---|---|
| 3539.55 µs | 1179 µs | 11594 µs | 3556.5 µs | 500 ms |
| | | | | Tiempo total de proceso |
| TOTAL Controladores 19869.05 µs + Total muest. (600 ms x 4)=1219.86 ms | | | | |

Tabla 7.1

Para apreciar con mayor detalle el desglose de tiempos ya sea tanto en en el monitor central y los controladores referirse a [81], [82], [83] y [84] respectivamente.

## 7.9.- CONCLUSIONES

La información obtenida en este cápitulo, es útil para tener las características de desempeño del sistema, y estar en posibilidad de comparar con otros sistemas además de conocer el grado de adaptación que puede tener dicho sistema en el control de diferentes procesos.

# 8.- CONCLUSIONES Y PERSPECTIVAS

Dentro de las principales características de este cápitulo se encuentran el resumen de la aportación de este trabajo haciendo referencia sobre los puntos relevantes del mismo, asi podemos hablar de las siguientes conclusiones:

## 8.1.- Aplicaciones de Algoritmos Difusos:

El uso de los algoritmos difusos se vislumbra como un panorama alentador para muchos casos, los cuales tienen la característica de no utilizar alguna descripción matemática del proceso a controlar, y tambièn de prescindir de un modelo matamático de algún tipo de controlador, aunque frecuentemente se hacen combinaciones de controles difusos y controladores convencionales de algún tipo, que bien necesitan la descripción matemática del proceso y el modelo matemático del controlador, en la parte del proceso donde se requiere el controlador convencional, podemos mencionar como ventajas el ahorro para muchos casos de tiempo de procesamiento, no necesitamos conocer el modelo mátematico del proceso y solamente necesitamos una idea de que como funciona. Aunque como desventaja se tendría que conocer de antemano el comportamiento del proceso y hacer pruebas cambiando reglas de inferencia hasta que el funcionamiento del controlador sea aceptable. Otra desventaja es el que cuando se describe el comportamiento del proceso, el controlador puede llegar a tener un número muy grande de reglas y por ende puede llevar un tiempo de procesamiento grande.

## 8.2.- Criterio para el uso de un algoritmo difuso.

Concluimos de acuerdo a lo anterior que el uso de un algoritmo difuso se basa fundamentalmente, en la complejidad del modelo mátematico del proceso y del controlador mismo, asi como del tiempo de cálculo que se lleva controlar dicho proceso, además de la precisión en los resultados obtenidos mediante el conocimiento y la experiencia en el manejo del proceso a controlar sustituyendo la descripción matemática del proceso, asi como caracteristicas que permitan modificar la respuesta a nuestro antojo, en resumen concluimos que se deberá tener:

- Experiencia en el manejo del proceso ó modelar mátematicamente el proceso.

- Evaluar el número de reglas de inferencia o bién complejidad del modelo matemático que describen el proceso y el controlador en el caso de que se conozcan.

- Tiempo de ejecución para ambos casos.

- Precisión en la respuesta de los controladores.

- Estabilidad dentro de los rangos determinados.

- Características que permitan sintonizar el controlador.

- Arquitecturas adecuadas para este tipo de algoritmos así como el software necesario para su aplicación.

## 8.3.- Arquitecturas

En este aspecto concluiremos sobre el tipo de arquitecturas adecuadas para la utilización de este tipo de algoritmos. Dentro de un algoritmo difuso se podrá tener un cierto número de reglas de inferencia que mediante un número determinado de variables de entrada se tendrá un número de variables de salida.
En el desarrollo del algoritmo de control difuso se pueden apreciará basicamente 4 fases, que son las siguientes:

- Identificación de variables

- Defusificación

- Fusificación

- Autosintonización

Dependiendo principalmente de las arquitecturas a nivel de controladores, serán arquitecturas acorde a los algoritmos de control difuso, dichos algoritmos tienen caracteristicas, de tal forma que se pueden implantar en arquitecturas paralelo ya que mientras se identifican las variables de entrada al instante n, se pueden estar fusificando las variables de la muestra anterior, en este caso si de dos ó más variables se trata, se puede utilizar una determinada arquitectura para la fusificación de la variables de entrada al mismo tiempo, además de defusificar al mismo tiempo las variables del instante n-2, y por último una etapa para la autosintonización de la variables de salida, autosintonizando en el mismo tiempo todas las variables de salida correspondientes al instante n-3. Por último, se determinará el tipo de procesamiento paralelo que se utilizará. Esto se usará basandosé en el tipo de proceso a controlar, es

decir dependiendo de la cantidad y del tipo de datos a procesar, pudiendo existir un paralelismo de memoria compartida cuando por el tipo de proceso lo amerite, es decir que se usen datos almacenados en la misma región de memoria para ahorrar tiempo consumido por ciclos de lectura. Este tipo de paralelismo se aplica en algoritmos cuya estructura tenga más de una variable de entrada de tal forma que teniendo una de ellas, las alturas correspondientes pueden almacenar las alturas a la entrada en la misma región y posteriormente leerlos. Por otra parte se deberá tomar en cuenta el paralelismo de bus compartido, el cual tiene aplicación en algoritmos que necesitan los mismos datos en el mismo instante de tiempo ó bien que pueden compartir el mismo bus para transmitir y/ó recibir datos, este tipo de algoritmos tienen la característica de usar un mismo flujo de datos como lo utilizaría un defusificador, al cual entraría el flujo de datos correspondientes a dos ó más variables a fusificar, de este modo mientras se transmite el flujo de datos correspondiente a la primera entrada a tráves del bus compartido, los datos correspondientes a la segunda variable serán procesados y viceversa. Existen tal vez otros razonamientos en los cuales se puede aplicar determinado tipo de arquitectura acorde con los algoritmos a utilizar.

## 8.4.- Control Distribuido

Podemos concluir que el tipo de control distribuido empleado depende igualmente de los procesos a monitorear y controlar, aunque en tèrminos generales los sistemas más aceptados por la versatilidad en cuanto a su operación son los sistemas abiertos ya que pueden ser escalables y practicamente adaptarse a cualquier clase de sistema, y así ser modular el sistema, de tal forma que dicho sistema se irá transformando de acuerdo a los requerimientos de los procesos a controlar. Todo esto tiene la gran ventaja de que el control distribuido solo utilizará lo necesario ya sea hardware ó software sin desperdiciar módulos, y por lo tanto ahorrar consumo de tiempo, energía y dinero. Adicionando a esto, podemos concluir que el sistema distribuido de tipo jerárquico incrementa la versatilidad y la adecuación del control distribuido al tipo de procesos, jerarquizandolos de acuerdo a las necesidades de control y monitoreo que se tenga. Ya que en un momento dado habrá algunos procesos de mayor prioridad que otros. De acuerdo a las características del sistema estará clasificado acorde a los siguientes puntos de vista :

-Tiempo de Respuesta: SRT
-De Grupo: Abierto.
-De Arquitectura: Distribuido.
-De Comunicación:Sistema de Múltiple Acceso.
-De Sistema Operativo: Cohesión.
-De Protocolos: Acceso Controlado (Demanda Adaptiva).
-Técnica de Control Distribuido: Jerárquico con Niveles de Influencia, utilizando Modelo de Coordinación.
-Programación en Tiempo Real: No portable.

## 8.5.- Arquitectura y protocolos

Desde este punto de vista se puede concluir que los controladores harán un procesamiento paralelo en tiempo real conectandosé vía un bus serie para transmitir y/ó recibir información estandard por medio de un bus serial RS232 ó a tráves de un multipuerto que permite conectar varios controladores al procesador central para recibir y/ó mandar información, que utilizará el proceso, esta característica a nivel de control distribuido relacionada con el paralelismo y la asignación jerárquica permite en un momento dado asignar un nuevo proceso a un determinado controlador prioritizando los procesos de tal forma que si un nodo tuviera una falla, pueda ser sustituido por otro controlador. Siendo este último controlador el que lleve a cabo un proceso de mayor prioridad que el que se estaba realizando ó bien realizará el proceso anterior y el proceso asignado.
Con respecto al formato de los prótocolos usados, se puede mencionar que el formato OSI es bien aceptado, apegandose a las caracteristicas de cada transmisión, de tal forma que dichos protocolos sean versátiles y se use solo lo necesario de estos para asi poder realizar una comunicación más eficiente. Para esto se plantea el uso de un algoritmo difuso, lo mismo que para la prioritización jerárquica de los controladores.

## 8.6.- Perspectivas

A continuación mencionaremos las perspectivas de este trabajo asi como algunas propuestas con respecto al mismo. Basandosé en las conclusiones anteriores se puede vislumbrar que la aplicación de algoritmos difusos puede ser aceptable, dependiendo de las caracteristicas del proceso a controlar, tomando las consideraciones antes mencionadas, de tal forma que podemos mencionar que para aplicaciones en tiempo real u otra donde se necesita rapidez y predictivos, los algoritmos difusos serán bien aceptados de tal forma que la restricción más grande para su aplicación es el conocimiento y/ó experiencia sobre dicha aplicación, en lugar de utilizar un modelo mátematico complejo de tal forma que se puede ahorrar tiempo de cálculo. La siguiente perspectiva se refiere al tipo de arquitecturas utilizadas de tal forma que el paralelismo ya sea en un sentido u otro es aplicable principlamente para optimizar el desarrollo del sistema de tal forma que se ahorraria tiempo de cálculo, incrementando la velocidad del proceso. Debido a esto hay infinidad de aplicaciones que tendrian una arquitectura paralela, principalmente donde se necesite velocidad en los cálculos. Adicionado a esto, las arquitecturas paralelo con algoritmos difusos podrán tener una perspectiva de uso bastante aceptable ya que dicha combinación incrementa su eficiencia. Habrá casos en que para algunas aplicaciones en un determinado proceso no se necesite velocidad, o bien no se requieran sistemas en tiempo real pero fuera de esto se tendrán perspectivas de aplicación muy buenas. Esto se involucra desde el punto de vista de control distribuido y a nivel de controladores externos, de acuerdo a esto las propuestas sobre el uso de paralelismo y algoritmos difusos asi como el tipo de control distribuido serán las siguientes:

- Determinar cuantas variables de entrada se utilizarán en el proceso.

- Determinar que precisión se requiere en el control y monitoreo del proceso.

- Conformar las reglas en que se basará el manejo del proceso, sustituyendo el modelo mátematico que describe el proceso así como el modelo que describe el control del sistema.

- Determinar el número de reglas que están involucradas en el proceso, así como el número de salidas y obviamente el de entradas.

- Desarrollar una arquitectura, ya sea paralelo o no, de acuerdo a la aplicación de los algoritmos y pensando que tenga el mayor grado de aplicabilidad, esto a nivel de controladores.

- Implementar el tipo de control distribuido de tal forma que se utilice para una amplia gama de aplicaciones.

- Minimizar el tiempo de cálculo, así como el costo, de tal forma que podamos combinar el factor economía y la versatilidad.

- Verificar la estabilidad de las reglas mediante el mètodo descrito con anterioridad ó bien cualquier otro.

- Así como hacer autosintonizable el algoritmo de tal forma que permita incrementar la precisión del sistema, mediante el mètodo descrito.

- Con respecto a la comunicación utilizar un algoritmo difuso, descrito con anterioridad, utilizando parcialmente o totalmente el formato OSI para así estandarisar protocolos e incremnentar la eficiencia sobre todo lo que se refiere a la parte de identificación y prioritización.

Estas son propuestas basadas en las conclusiones y los mètodos antes descritos, aunque existe la posibilidad de detallarlas más y posiblemente mejorarlas. Dichas propuestas son utilizadas dentro de este trabajo.

## BIBLIOGRAFÍA


-[1] REAL-TIME UNIX SYSTEMS Design and Application Guide Borko Furth, Dan Grostick, David Gluch, Guy Rabbat, John Parker and Meg McRoberts. KLUWER ACADEMIC PUBLISHERS, Massachussets USA, 1991

-[2] REAL TIME SYSTEMS AND THEIR PROGRAMMING LANGUAGES Alan Burns and Andy Wellings, ADISON WESLEY PUBLISHING COMPANY, Great Britain, 1991.

-[3] CLOCK SINCHRONICATION IN DISTRIBUITED REAL TIME SYSTEMS H. Kopetz and W. Ochsenreiter (IEEE Transaction on Computers) TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering by John A. Stankovic and Krithi Ramamritham), Washington, D C USA, 1987.

-[4] TIMING CONSTRAINTS OF REAL-TIME SYSTEMS CONSTRUCTS FOR EXPRESSING THEM, METHODS OF VALIDATING THEM, B. Dasarathy (TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering by John A. Stankovic and Krithi Ramamritham), Washington, D C USA, 1985.

-[5] MULTIPLE-ACCESS PROTOCOLS AND TIME-CONSTRAINED COMMUNICATION, James F. Kurose, Mischa Schwartzand W. Ochsenreiter(IEEE Transaction on Computers) (TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering by John A. Stankovic and Krithi Ramamritham), Washington, D.C USA, 1984.

-[6] REAL TIME BEHAVIOR OF PROGRAMS, V. H Hasse (TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering by John A. Stankovic and Krithi Ramamritham), Washington, D C USA, 1981.

-[7] MASTERING SERIAL COMMUNICATION, Peter W. Gofton, Sibex, California USA, 1986.

-[8] COMUNICACIONES SERIE, GUIA DE REFERENCIA DEL PROGRAMADOR EN C, Joe Campbell, Editorial Anaya, Barcelona España, 1987.

-[9] A SURVEY OF DISTRIBUTED CONTROL TECHNIQUES, Tutorial Distributed Control, Robert E. Larson, IEEE Catalog No EH0153-7

-[10] Guaranteed Response Times in a Distribuited Hard Real-Time Environment D. W Leinbaugh and M. R Yamini( TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering, ), Washington, D.C USA, 1986.

-[11] DISTRIBUTED COMPUTER SYSTEMS FOR INDUSTRIAL PROCESS CONTROL, J. D. Schoeffler ( TUTORIAL HARD REAL-TIME SYSTEMS IEEE Transactions on Software Engineering), Washington, D.C USA, 1984.

-[12] FUZZY SYSTEMS THEORY AND ITS APPLICATIONS, Toshiro Terano, Kiyoji Asai, Michio Sugeno, Academic Press, Inc, San Diego California, 1987.

-[13] FUZZY LOGIC AND CONTROL, Mohammad Jamshidi, Nader Vadiee, Timothy J. Ross, Prentice- Hall, Inc,Englewood Cliffs N J, USA, 1993.

-[14] FUZZY SETS AND APPLICATIONS: Selected Papers by L.A.Zadeh, Edited by R.R Yager, S.Ovchinnikov, R.M.Tong,H.T.Nguyen,J ohn Wiley y Sons, USA, 1987.

-[15] AUTOMATIC TRAIN OPERATION SYSTEM BY PREDICTIVE FUZZY CONTROL, S.Yasunobu and S.Miyamoto, Industrial Applications of Fuzzy Control, Edited by Michio Sugeno, Elseiver Science Publishing Company Inc, New York, N Y, USA, 1992.

-[16] A FUZZY LOGIC CONTROLLER FOR AIRCRAFT FLIGHT CONTROL, L.I.Larkin, Industrial Applications of Fuzzy Control,Edited by Michio Sugeno, Elseiver Science Publishing Company Inc, New York, N Y,USA, 1992.

-[17] AUTOMOBILE SPEED CONTROL SYSTEM USING A FUZZY LOGIC CONTROLLER, S.Murakami and M.Maeda, Industrial Applications of Fuzzy Control, Edited by Michio Sugeno, Elseiver Science Publishing Company Inc, New York, N Y, USA, 1992.

-[18] AN EXPERIMENTAL STUDY ON FUZZY PARKING CONTROL USING A MODEL CAR, M.Sugeno and K.Murakami, Industrial Applications of Fuzzy Control, Edited by Michio Sugeno, Elseiver Science Publishing Company Inc, New York, N Y, USA, 1992.

-[19] A MICROPROCESSOR BASED FUZZY CONTROLLER FOR INDUSTRIAL PURPOSES, T.Yamazaki and M.Sugeno, Industrial Applications of Fuzzy Control,Edited by Michio Sugeno, Elseiver Science Publishing Company Inc, New York,N Y, USA, 1992.

-[20] NEURAL NETWORKS AND FUZZY SYSTEMS, Bart Kosko, Prentice-Hall, Englewood Cliffs, New Yersey, 1992.

-[21] DIGITAL CONTROL ENGINEERING, M.Gopal, John Wiley and Sons, New Delhi, India, 1988.

-[22] DIGITAL CONTROL SYSTEMS, Benjamin C. Kuo, Holt Rinehart and Winston Inc, 1980.

-[23] FUZZY MODEL REFERENCE LEARNING CONTROL FOR CARGO SHIP STEERING, Jeffery R.Layne and Kelvin M.Passino, 1993.

-[24] FUZZY SYSTEMS AN OVERVIEW, Toshinori Munakata, Yashvant Jani, Communications of the ACM, 1994.

-[25] FUZZY LOGIC, NEURAL NETWORKS, AND SOFT COMPUTING, Lofti A.Zadeh, Communications of the ACM, 1994.

-[26] FUZZY LOGIC, Bart Kosko and Satoru Isaka, Scientific American, 1993.

-[27] FUZZY FUNDAMENTALS, Earl Cox, Spectrum, 1992.

-[28] DEVELOPMENT OF FUZZY ALGORITHMS FOR SERVO SYSTEMS, Y.F. Li and C.C Lau, IEEE International Conference on Robotics and Automation, Philadelphia, Pennsylavania, 1988.

-[29] FUZZY LOGIC IN CONTROL SYSTEMS: FUZZY LOGIC CONTROLLER-PARTS 1 AND 2, Chuen Chien Lee, University of California, Berkeley California, IEEE, 1990.

-[30] THEORY OF THE FUZZY CONTROLLER: AN INTRODUCTION, James J.Buckley, Elsevier Science Publishers, 1992.

-[31] STABILITY ANALYSIS AND DESIGN OF FUZZY CONTROL SYSTEMS, Kazuo Tanaka and Michio Sugeno, Elsevier Science Publishers, 1992.

-[32] A SELF-TUNNING FUZZY CONTROLLER, Mikio Maeda and Shuta Murakami, Elsevier Science Publishers, 1992.

-[33] A STABILITY APPROACH TO FUZZY CONTROL DESIGN FOR NONLINEAR SYSTEMS, Guang-Chyan Hwang and Shih-Chang Lin, Elseiver Science Publishers, 1990.

-[34] THE STATE OF KNOWLEDGE.BASED SYSTEMS, Frederick Hayes-Roth, Neil Jacobstein, Communications of the ACM, 1994.

-[35] INTRODUCTION TO THE THEORY OF FUZZY SUBSETS VOLUME 1, A. Kaufmann, Academic Press, New York, N Y, 1975.

-[36] FUZZY CONTROL AND FUZZY SYSTEMS, Witold Pedrycz, JohnWiley & Sons Inc, Winnipeg Canada, 1989.

-[37] LENGUAJE ENSAMBLADOR PARA MICROCOMPUTADORAS IBM PARA PRINCIPIANTES Y AVANZADOS, J.Terry Godfrey, Prentice Hall, Prentice Hall Hispanoamericana, México 1991.

-[38] 80386/80286 PROGRAMACION EN LENGUAJE ENSAMBLADOR, William H.Murray. III y Chris H.Pappas, Osborne/McGraw-Hill, México, 1987.

-[39] THE 8088 AND 8086 MICROPROCESSORS PROGRAMMING, INTERFACING, SOFTWARE, HARDWARE, AND APPLICATIONS, Walter A.Triebel, Avtar Singh, Prentice Hall, Englewood Cliffs, New Yersey, 1991.

-[40] THE INTEL MICROPROCESSORS 8086/8088, 80186, 80286, 80386, AND 80486, ARCHITECTURE, PROGRAMMING AND INTERFACING, Barry B.Brey, Merrill USA, 1991.

-[41] OPTOELECTRONICS DATA BOOK, Texas Instruments.

-[42] ARQUITECTURA DE COMPUTADORAS Y PROCESAMIENTO PARALELO, K.Hwang, Mc Graw Hill.

-[43] MPSIM USER'S GUIDE, Microchip Technology Inc, 1993.

-[44] MPASM USER'S GUIDE, Microchip Technology Inc, 1993.

-[45] MICROCHIP DATABOOK, Microchip Technology Inc, 1993.

-[46] THE 8051 MICROCONTROLLER, I.Scott Mackenzie, Merrill, New York, N Y, 1992.

-[47] MCS-51 FAMILY OF SINGLE MICROCOMPUTERS USER'S MANUAL, Intel, Santa Clara, California USA, 1981.

-[48] 8080A/8085 ASSEMBLY LANGUAGE PROGRAMMING, Lance Leventhal, Osborne/McGraw-Hill, Berkeley, California USA, 1978.

-[49] 8085A COOKBOOK,Christopher A.Titus, Jonathan A.Titus, and David G.Larsen, Howard W.Sams and Co, Inc, Indianapolis, Indiana USA, 1981.

-[50] AVMAC 805 1 FAM I LY USER'S MANUAL, by Avocet Systems Inc, Rockport, Maine USA, 1988.

-[51] MACRO MAGIC WITH TURBO ASSEMBLER, Jim Mischel, John Wiley and Sons Inc, USA.

-[52] DEVELOPMENT SYSTEMS, Motorola Microprocessor Products Group, Austin Texas, USA, 1989.

-[53] M68HC11 REFERENCE MANUAL, Motorola Inc, 1991.

-[54] M68000 FAMILY REFERENCE, Motorola Inc, USA, 1990.

-[55] THE 68000 MICROPROCESSOR ARCHITECTURE,SOFTWARE, AND INTERFACING TECHNIQUES, Walter A.Triebel, Avtar Singh, Prentice Hall, Englewood Cliffs, New Jersey USA, 1986.

-[56] 16-AND 32-BIT MICROCOMPUTER INTERFACING PROGRAMING EXAMPLES IN C AND M68000 FAMILY ASSEMBLY LANGUAGE, G.J.Lipovski, Prentice Hall, Englewood Cliffs, New Yersey, USA, 1990.

-[57] THE M68000 MICROPROCESSOR FAMILY FUNDAMENTALS OF ASSEMBLY LANGUAGE PROGRAMMING AND INTERFACE DESIGN, Yu-Cheng Liu, Prentice Hall, Englewood Cliffs, New Yersey USA, 1991.

-[58] INTERFACING TO THE IBM PERSONAL, by Lewis C.Eggebrecht, Howard W.Sams and company, Indianapolis, Indiana USA, 1983.

-[59] THE COMPLETE GUIDE TO IBM PC AT ASSEMBLY LANGUAGE, Harley Hahn, Scott Foresman and Company.

-[60] MICROPROCESSORS AND INTERFACING PROGRAMMING AND HARDWARE, Douglas V.Hall, McGraw Hill.

-[61] IBM PC JR ASSEMBLER LANGUAGE, David C.Willen, Howard W.Sams and Co.Inc.

-[62] IBM PS/2 A REFERENCE GUIDE TJ BYERS, McGraw Hill.

-[63] FUNCIONES DE DOS Y BIOS MANUAL DE BOLSILLO, Adison Wesley Iberoamericana.

-[64] ASSEMBLY LANGUAGE QUICK REFERENCE, Allen L. Wyatt, Sr, Que Corporation, Carmel Indiana, 1989.

-[65] LINEAR DATABOOK NATIONAL SEMICONDUCTOR, USA, 1980.

-[66] THE TTL DATA BOOK FOR DESIGN ENGINEERS, Texas Instruments Incorporated, USA, 1981.

-[67] DINAMICA DE SISTEMAS, Francisco J.Rodríguez Ramírez, Trillas, México D.F, 1989.

-[68] CONTROL DE MOTORES DE PASOS, Luis Agustín Alvarez-Icaza Longoria, Instituto de Ingeniería, 1992.

-[69] CONVERSION DE ENERGIA ELECTROMECANICA, Representaciones y Servicios de Ingeniería, Gourishankar México, 1975.

-[70] MAQUINAS ELECTRICAS, George J.Thaler y Milton L. Wilcox, Limusa México, 1979.

-[71] CONTROL NUMERICO, Jose Ramón Alique Lopez, Marcombo, Barcelona 1981.

-[72] CURSO DE ROBOTICA, Jose Ma. Angulo Usategui, Rafael Avilez González, Paraninfo Bilbao 1984.

-[73] THE Z80 MICROPROCESSOR ARCHITECTURE, INTERFACING, PROGRAMMING AND DESIGN, Ramesh Gaonkar, Merrill USA, 1993

-[74] MICROPROCESSOR ARCHITECTURE, PROGRAMING, AND APLICATION WITH THE 8085/8080A, Ramesh Gaonkar, Merrill USA, 1984.

-[75] TMS34020 USER'S GUIDE, Texas Instruments, 1990.

-[76] MICROCOMPUTER PRODUCTS DATABOOK, Nec Electronics Inc.

-[77] DSP96002 IEE FLOATING-POINT DUAL-PORT PROCESSOR USER'S MANUAL, Motorola Inc, 1989.

-[78] DSP56000/DSP56001 DIGITAL SIGNAL PROCESSOR USER'S MANUAL, Motorola Inc, 1989.

-[79] ADAPTIVE FUZZY SYSTEMS AND CONTROL DESIGN AND STABILITY ANALYSIS, Li-Xin Wang, Prentice Hall, New Jersey, 1994.

-[80] QNX 4 OPERATING SYSTEM /SYSTEM ARCHITECTURE, QNX SOFTWARESYSTEMSLTD, Ontario, Canada 1993.

-[81] DISEÑO Y DESARROLLO DE UN SISTEMA PARA EL CONTROL DISTRIBUIDO DE PROCESOS USANDO UN SISTEMA OPERATIVO EN TIEMPO REAL, Tesis de Licenciatura, Facultad de Ingenieria, UNAM, López Rodriguez Alejandra, México D.F. 1995.

-[82] CONTROL PARA AUTOCLAVE USANDO UN ALGORITMO DIFUSO AUTOSINTONIZABLE, Tesis de licenciatura, Facultad de Ingenieria UNAM, Diaz José Emilio y Martinez Victor M, México, D.F, 1995.

-[83] LOCALIZACION Y DIRECCIONABILIDAD DE UN VEHICULO USANDO UN CONTROLADOR DIFUSO IMPLEMENTADO CON MICROCONTROLADORES Tesis de Licenciatura, Facultad de Ingenieria UNAM, Priego Juan C y Salazar Erick A. México , D.F, 1995

-[84] CONTROLADOR DIFUSO PARA NIVEL DE LÍQUIDOS IMPLEMENTADO CON MICROPROCESADORES, Tesis de Licenciatura, Facultad de Ingenieria UNAM, Ramos Vargas Floricel, México ,D.F, 1995.

-[85] Memorias Taller de Procesamiento Paralelo, IIMAS, UNAM, Agosto 1994, México, D.F.

# ANEXOS

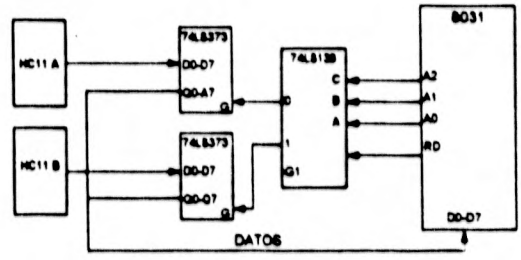



*Fig. A.7.1 Diagrama de Bloques del controlador 1* [83]

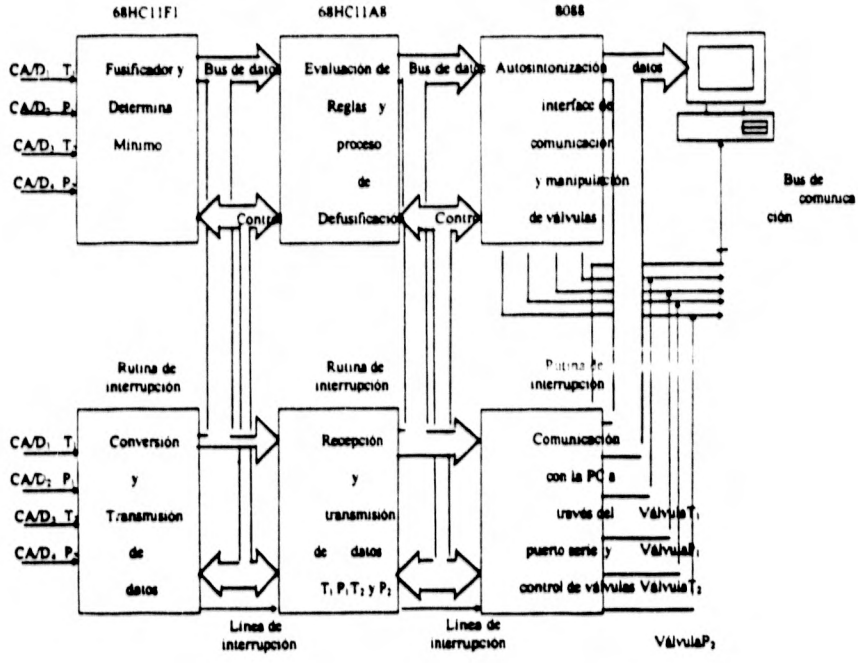



*Fig. A.7.2 Diagrama de Bloques del Controlador 3* [82]

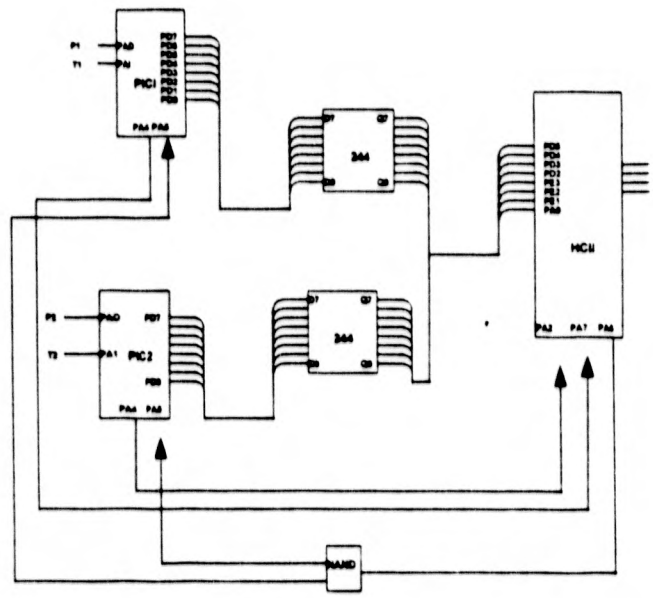



*Fig. A.7.3. Diagrama de bloques del Controlador 4* [84]

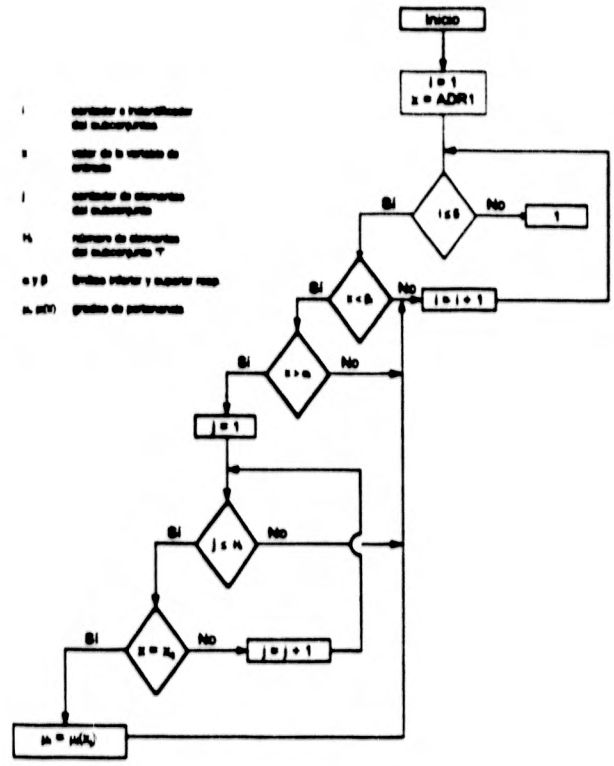



*Fig. A.7.4. Diagrama de flujo del fusificador del controlador 1* [83]

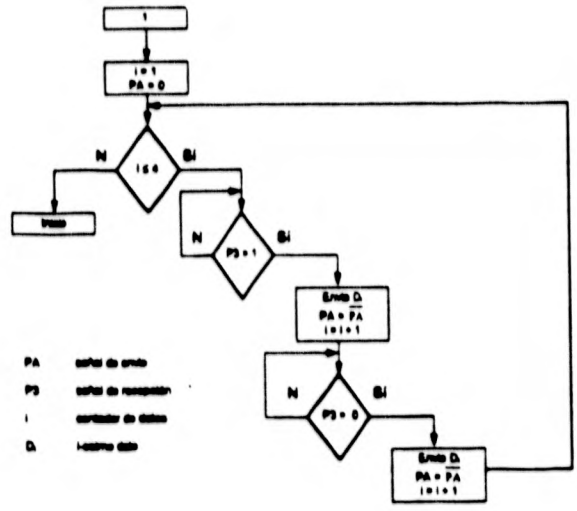


*Fig. A.7.5. Diagrama de flujo de la transmisión de datos entre los microcontroladores del controlador 1* [83]

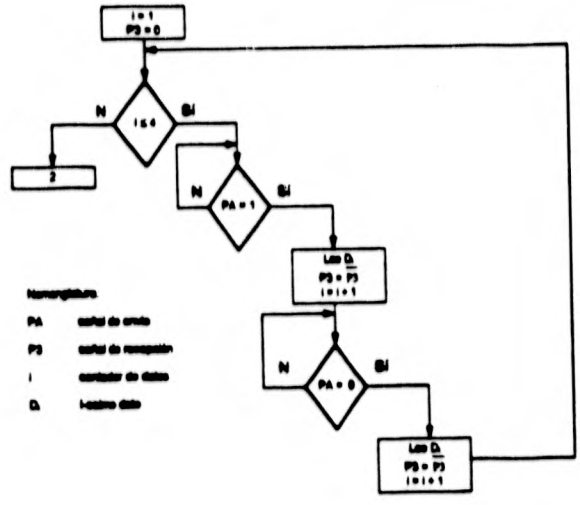


*Fig. A.7.6. Diagrama de flujo de recepción de datos entre los microcontroladores en el controlador 1* [83]

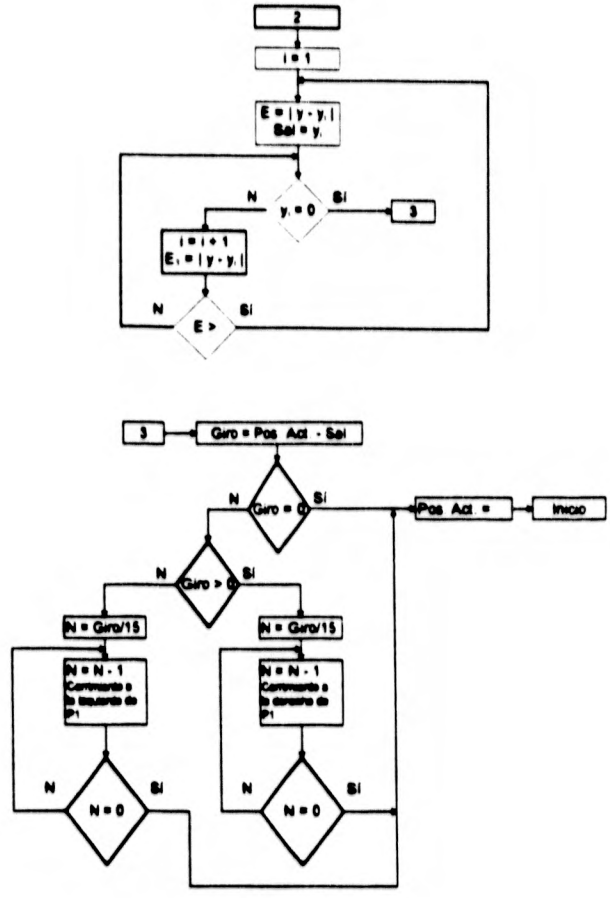



*Fig. A.7.7. Diagrama de flujo de activación del motor del controlador 1* (83)

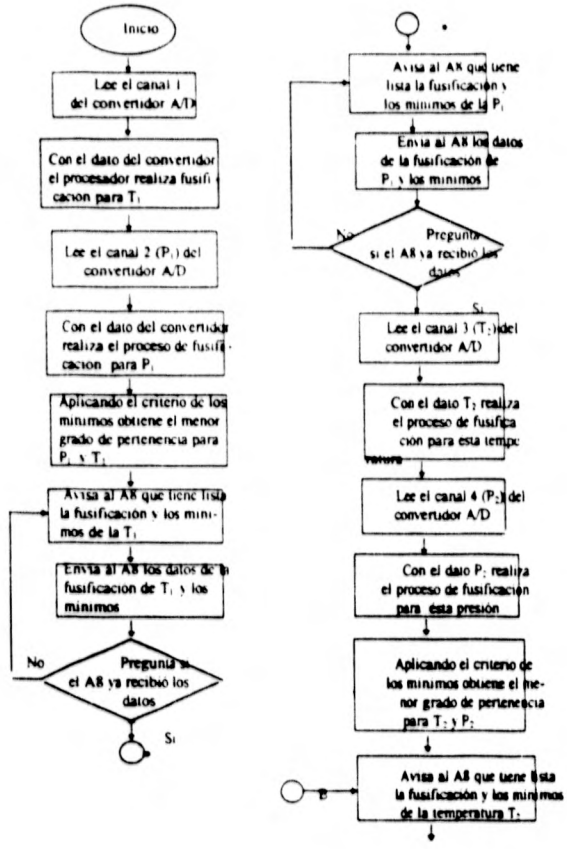



*Fig. A.7.8. Diagrama de flujo de acciones del 68hc11f1 en el controlador 3* [82]

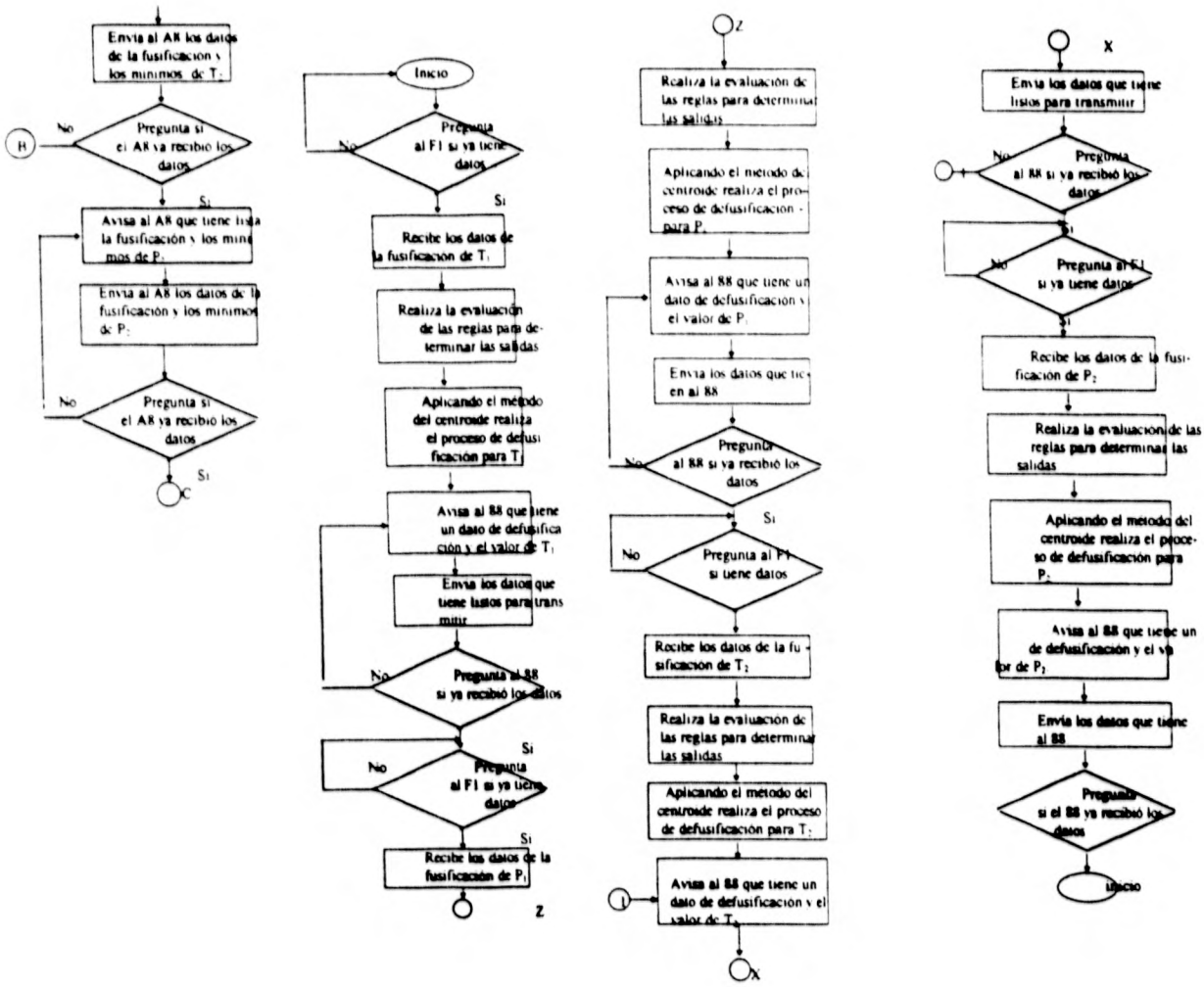



*Fig. A.7.9. Diagrama de flujo de acciones del 68hc11A8 en el controlador 3* [82]

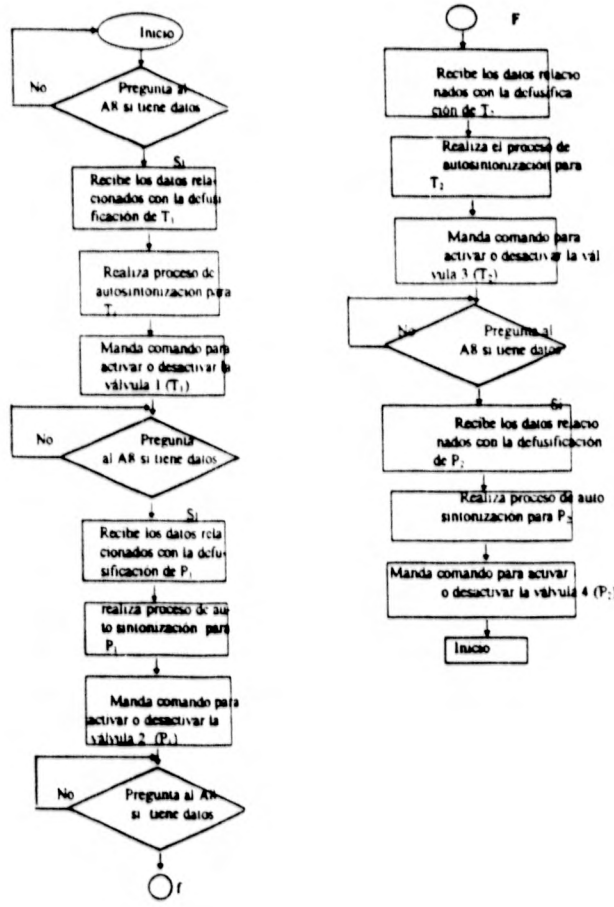
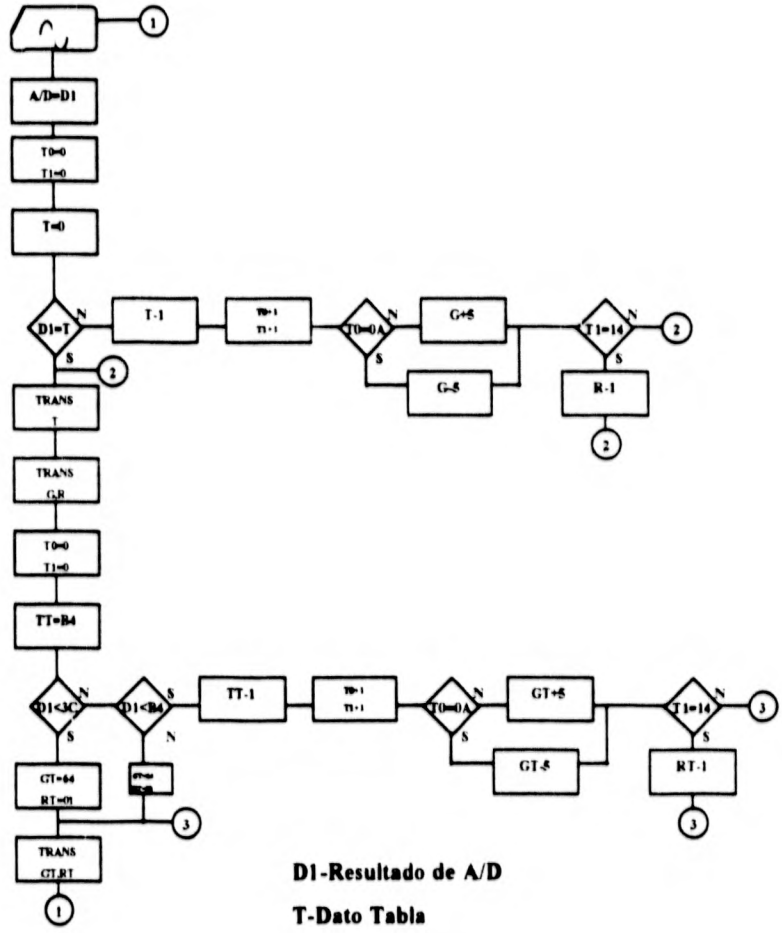



*Fig. A.7.10. Diagrama de flujo de acciones del 8088 en el controlador 3* [82]




**D1-Resultado de A/D**

**T-Dato Tabla**

**T0-Contador 0**

**T1-Contador 1**

**G-Grado**

**R-Región**

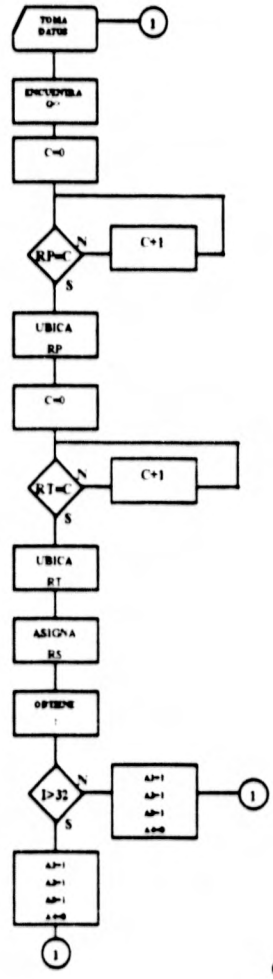
**TT-Dato Tabla Traslape**

**GT-Grado Traslape**

**RT-Región Traslape**

**TRANS-Transfiere**

*Fig. A.7.11. Diagrama de flujo de acciones del pic16c74 en el controlador 4* [84]




**C-Contador**

**RP-Región Presión**

**G-Grado**

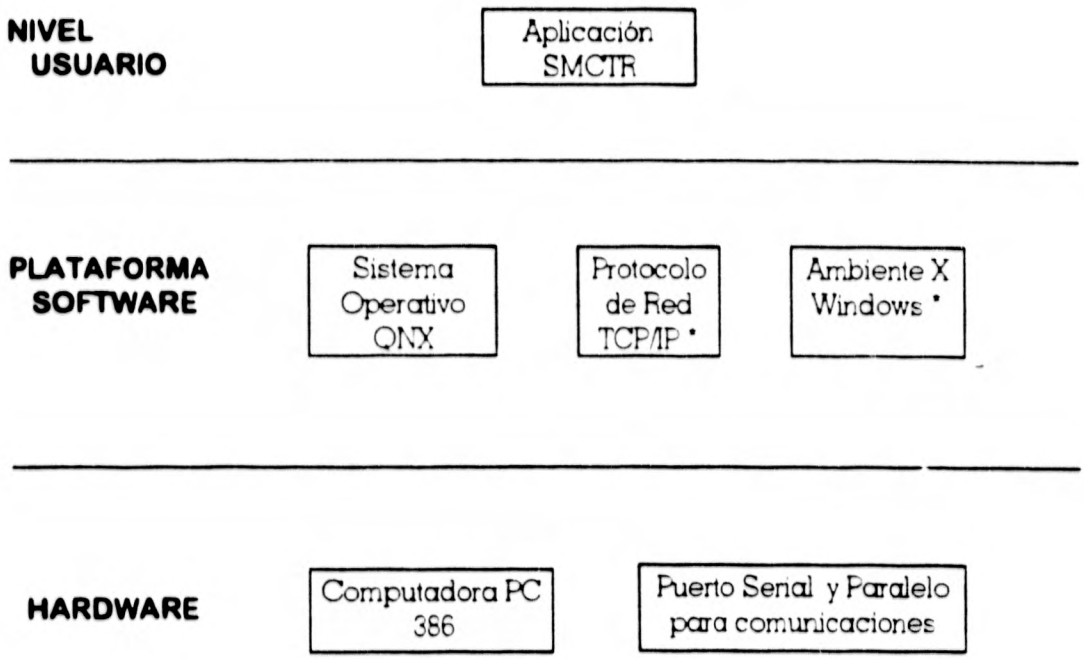
**RT-Región Temparatura**

**RS-Región Salida**

**I-Centroide**

**An-Actuadores**

*Fig. A.7.12. Diagrama de flujo de acciones de los 68hc11A8 en el controlador 4* [84]




*** Módulos no usados en esta etapa del proyecto**

***Fig. A.7.13. Arquitectura de Software del sistema de monitoreo de control en tiempo real*** [81]

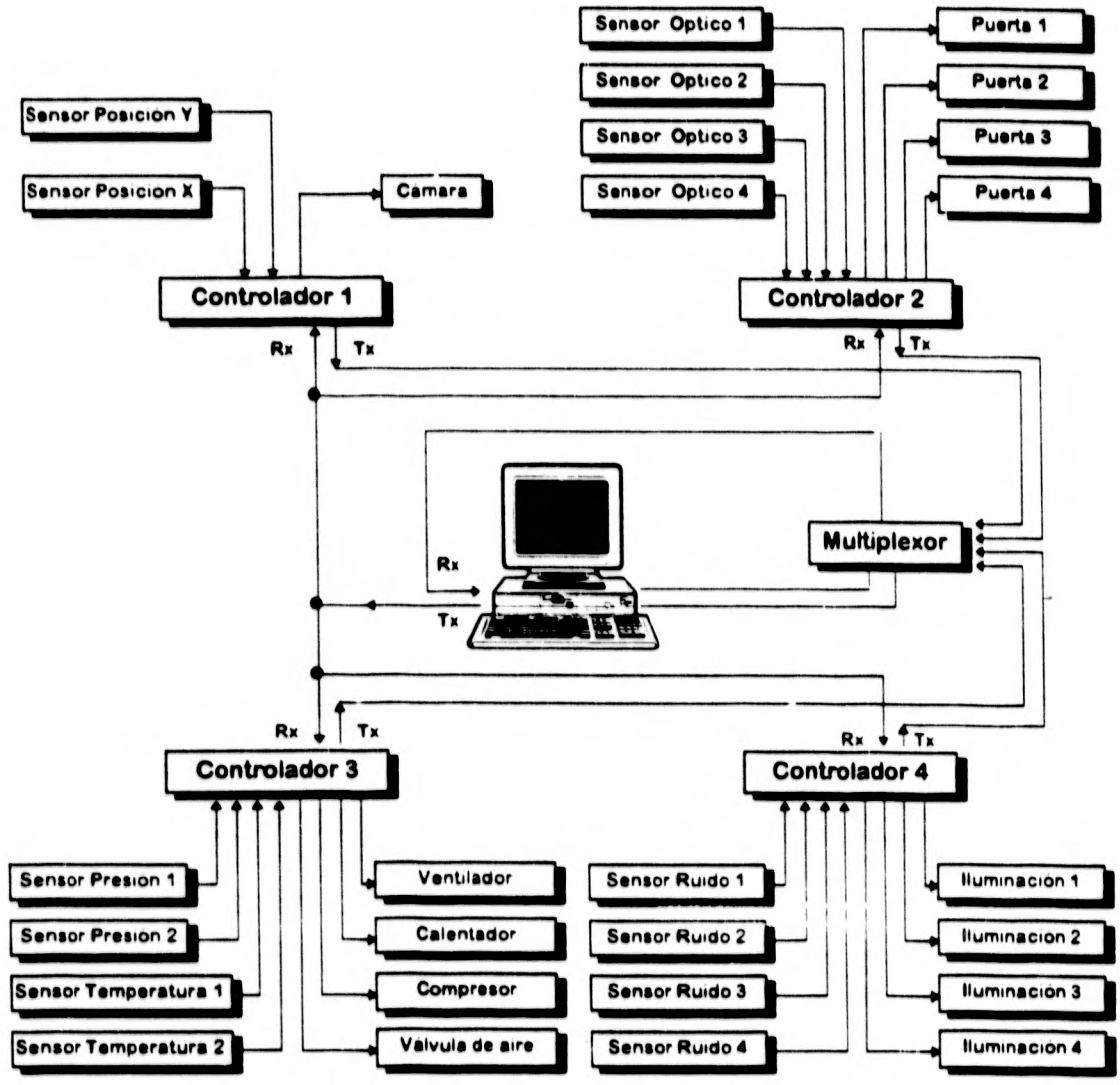



*Fig. A.7.14. Arquitectura a bloques del sistema de monitoreo de control en tiempo real*