UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS
Facultad de Ingeniería Eléctrica
Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Título: Aplicación de la Inteligencia Artificial en la domótica.

Autor: Joel Rodríguez Falcón

Tutores: Dr. Jorge Sánchez Bermúdez
MSc. Iván Santana Ching

Santa Clara

2004

“Año del 45 aniversario del triunfo de la Revolución”

Hago constar que el presente trabajo fue realizado en la Universidad Central "Marta Abreu" de las Villas como parte de la culminación de los estudios de la especialidad de Automática autorizando a que el mismo sea utilizado por la Institución, para los fines que estime convenientes, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicado sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor

Firma del Jefe de Dpto.
Donde se defiende el trabajo

_____

Firma del Responsable de
Información Científico-Técnica

Tarea Técnica

Tareas a desarrollar:

1. Realizar una búsqueda de información sobre el tema de la domótica y el control inteligente.
2. Efectuar un estudio de las principales tecnologías existentes en el campo de la domótica.
3. Proponer una variante de control donde se aplique la Inteligencia Artificial en la domótica.
4. Desarrollar un controlador difuso que permita el procesamiento de señales y la manipulación de datos en edificios inteligentes.
5. Confeccionar el informe del trabajo.

_____ _____

Firma del Autor Firma del Tutor

Resumen

Resumen

En el presente trabajo se hizo un estudio de las principales tecnologías existentes en el mercado de la domótica y su aplicación en la actualidad. Se realizó una descripción de todo un conjunto de conceptos técnicos que debemos tener en cuenta para trabajar con sistemas domóticos.

Se describió un sistema multiagente desde un nivel de análisis abstracto. Además se revisó por la parte de la Inteligencia Artificial el control difuso y se implementó mediante el uso del lenguaje de programación Matlab. Se construyó un sistema de toma de decisiones que utiliza variables comunes en edificios, el cual constituye el agente de control multiagente, esto se hace mediante la evaluación de un sistema de inferencia difuso que manipula una base de datos.

Se hizo un servidor de automatización OLE que garantiza el funcionamiento de forma periódica de todos los procedimientos que realiza el controlador difuso.

Se desarrolló una interfaz de usuario gráfica para la observación de las variables y la ejecución del servidor de automatización OLE de Matlab, esta constituye el *RoomAgentDisplay* que es uno de los agentes del sistema.

Se construyó una base de datos que contiene todos los datos de entrada y salida, la cual es el agente más sencillo del sistema multiagente y que llamamos *HistoryAgent*.

Los objetivos principales que se persiguen son hacer un análisis de la aplicación de la Inteligencia Artificial en el campo de la domótica, así como realizar una propuesta específica mediante el uso de la herramienta de programación Matlab, analizando las tecnologías actuales en el mundo de la domótica.

Introducción

La Inteligencia Artificial ha devenido en los últimos años en una de las ramas de las Ciencias de la Computación más difundidas y para las que se ha dedicado un gran esfuerzo, por los beneficios que reporta en la solución de problemas donde es necesaria, principalmente en la manipulación de información simbólica, en aquellos problemas en los que aparece la subjetividad para resolverlos o en los que la información a manipular es incompleta o borrosa, entre otros.

Los lenguajes de programación llamados convencionales permiten la creación de sistemas computacionales para la solución de problemas. En ocasiones es necesario diseñar sistemas que imiten la forma de pensar del hombre para acometer esta tarea, por lo que ha sido necesaria la creación de técnicas de programación que garanticen la solución de los mismos con un mínimo de esfuerzo y recursos.

El ritmo de vida actual ha provocado un fenómeno cultural sin precedentes, nos encontramos inmersos en la Sociedad de la Comunicación de Información, donde aparece una nueva ciencia: la domótica, que se convierte en una necesidad actual y vital. La rápida evolución tecnológica de la electrónica y la informática, ha inundado nuestro entorno con televisores, equipos de fax, módems, redes y sistemas informáticos tanto en centros de trabajo como en viviendas particulares.

Los sistemas domóticos actuales integran automatización, informática, control inteligente y Nuevas Tecnologías de la Información (NTI). Domótica: "tecnología aplicada al hogar", término formado por la raíz latina *domus* (casa), define todas las funciones y servicios proporcionados por una vivienda inteligente. En francés se utiliza un término similar, *domotique* formado por "*domus*" y "*robotique*" (robótica), y en inglés se utiliza la expresión *home systems* o *smart house*.

Con anterioridad se han realizado otros trabajos tanto en el área de la domótica como en el de la Inteligencia Artificial, pero no se había desarrollado un estudio de la vinculación entre ellos.

Mediante el uso de la Inteligencia Artificial como una alternativa para el desarrollo de aplicaciones orientadas al ámbito del hogar podemos garantizar el correcto funcionamiento de estos sistemas. En los edificios inteligentes existen múltiples variables, cuyo comportamiento en algunos casos es impredecible, por lo que su manipulación trae consigo el desarrollo de soluciones con un desenvolvimiento apto para estas circunstancias. Por esta razón se propone la utilización de una ciencia tan efectiva para la toma de decisiones como lo es la Inteligencia Artificial.

La problemática principal con que nos enfrentamos al comienzo de esta tesis es la carencia de un estudio de la aplicación de la Inteligencia Artificial a la domótica. Además la falta de información por parte de los fabricantes de cómo son implementados físicamente estos sistemas.

Luego de efectuado el estudio sobre la aplicación de la Inteligencia Artificial a la domótica, se pudiera proponer como una excelente alternativa. Por otra parte la realización de una aplicación práctica con este objetivo corroboraría lo anterior.

Por todo lo anterior los objetivos principales que se persiguen son hacer un análisis de la aplicación de la Inteligencia Artificial en el campo de la domótica, así como realizar una propuesta específica mediante el uso de herramientas de programación como Matlab, analizando las tecnologías actuales.

Durante la realización del trabajo se utilizó Matlab para el desarrollo de un controlador difuso, la gestión de bases de datos y la implementación de un servidor de automatización OLE (Object Linking and Embedding). También se apoyó del uso del lenguaje de programación Delphi para realizar la interfaz de usuario gráfica, cuyos objetivos son cargar el servidor de Matlab y manipular las variables del sistema. Para el diseño de las bases de datos se utilizó el Microsoft Access el cual mantiene una comunicación efectiva con Delphi y Matlab.

La tesis está estructurada en tres capítulos, en el primero se hace una revisión bibliográfica sobre la domótica, algunas tecnologías y criterios a tener en cuenta para el desarrollo de tales sistemas. En el segundo se realiza una descripción abstracta de un enfoque

multiagente. Además se exponen las características de todos los agentes que forman el sistema así como su diseño, implementación e interacción. En el tercer capítulo se realiza la implementación física de tres de estos agentes, el *ControlAgent*, el *HistoryAgent* y el *RoomAgentDisplay*. También se describe cómo funciona la aplicación en general, además de algunos de los pasos de suma importancia. Por último se exponen las conclusiones y recomendaciones del trabajo.

Capítulo l ¿Qué es la domótica?

La domótica es la integración de sistemas electrotécnicos relacionados con la electricidad, la electrónica, la informática, y las telecomunicaciones en el hogar. Las principales áreas de los sistemas electrónicos y sus funcionalidades relacionadas con la integración son las siguientes:

Automatización y Control que abarca el control de aplicaciones y dispositivos domésticos, con instalaciones, sistemas y funciones para iluminación, climatización, persianas y toldos, puertas y ventanas, cerraduras, electrodomésticos, suministro de agua, suministro de gas, suministro de electricidad, etc.

Seguridad y Alarmas de personas, bienes, incidencias y averías, con instalaciones, sistemas y funciones para alarmas de intrusión, cámaras de vigilancia, alarmas personales, alarmas técnicas (incendio, humo, agua, gas, fallo de suministro eléctrico, fallo de línea telefónica etc.).

Voz y Datos que incluye la distribución de ficheros textos, imágenes y sonidos y compartiendo recursos entre dispositivos, acceso a Internet y a nuevos servicios, con instalaciones, sistemas y funciones como red de telefonía, red local de datos, pasarelas residenciales, *routers* de acceso a Internet.

Capítulo l ¿Qué es la domótica?

Con esta definición queda claro que la domótica no son “servicios” ni “productos” sino la integración y la implementación de los sistemas electrotécnicos en el hogar.

1.1 ¿A quién le puede interesar un sistema domótico?

La respuesta a esta pregunta sería a todo el mundo. Un sistema domótico es flexible, versátil y adaptable a cualquier necesidad, a cualquier tipo de edificio y a cualquier actividad que en él se vaya a desarrollar. Un sistema domótico proporciona un sinfín de beneficios y ventajas inalcanzables mediante una instalación tradicional.

Si tuviésemos que resumir las principales razones para instalar un Sistema Inteligente, sin duda serían éstas: comodidad, seguridad, confort, información, ahorro energético e imagen. Pero sin duda, estas seis razones mencionadas se reducen a una sola: Aumento de la Calidad de Vida.

Una empresa, superficies comerciales, edificios industriales, almacenes, a una oficina o un comercio, sean cuales sean sus dimensiones y sea cual sea su actividad, pueden incrementar el rendimiento de sus instalaciones con un sistema inteligente.

1.2 Control Integrado e Inteligente

Todo sistema domótico consiste en una serie de sensores que automatizan, mediante un control inteligente, a un conjunto de actuadores. Si no tenemos en cuenta el adjetivo,

existen soluciones que hacen lo mismo y que no son domóticas, la diferencia fundamental entre una y otra solución es el control integrado e inteligente.

Por ejemplo, en una vivienda se puede utilizar un sensor de temperatura, para automatizar la climatización. Sin embargo el control de dicha climatización se realiza de forma manual, ya que el usuario debe cambiar continuamente la temperatura de confort deseada en función de la temperatura exterior, de que sea de día o de noche, del número de personas presentes etc.

Si el tema resulta laborioso para un único control manual de temperatura ¿cuánto más difícil será mantener un edificio?, ¡nos pasaríamos el día recorriendo sus distintas plantas para poder ajustar cada termostato! Con un control inteligente el sistema supervisa los parámetros y toma las decisiones adecuadas en cada momento, pudiendo activar luces cuando detecta movimiento y sin embargo no hacerlo cuando sabe que las personas se encuentran durmiendo, o generar alarmas cuando la seguridad esté activada.

Aunque exista control automático, el usuario siempre puede actuar directamente sobre el sistema. Para ello se pueden emplear:

Pulsadores y mando a distancia: desde cualquier lugar de la vivienda o del edificio y sin modificar el cableado se puede actuar sobre el sistema mediante soluciones inalámbricas.

Teléfono: accede a todas las funciones integradas del sistema como si lo hiciera desde un pulsador o un mando a distancia, pero en este caso, una voz guía con claras instrucciones para no tener que recordar las distintas funciones. Existen soluciones que además confirman que la orden ha sido realizada.

1.3 Funciones de un sistema domótico

1.3.1 Seguridad

Los mismos sensores que encienden automáticamente la luz, cuando se activa la opción anti intrusión, sirven para detectar a los intrusos, y si esto sucede, encenderán y apagarán la luz de forma intermitente al tiempo que se produce una alarma sonora para crear una atmósfera de caos que ahuyente a los intrusos.

1.3.2 Simulación de presencia.

El sistema actuando sobre luces y persianas, hará creer a extraños que el lugar se encuentra habitado. Actuará sobre persianas durante el día y luces por la noche, encenderá TV o radio, para lograr una eficaz "Simulación de Presencia" y se desenvolverá siempre de forma distinta.

1.3.3 Seguridad de bienes

Ante una fuga de gas o agua, el sistema debe actuar sobre las correspondientes electroválvulas evitando todo posible daño. Si se produce una emisión excesiva de CO2 o humos, el sistema automáticamente cierra el paso de gas y la alarma actúa de forma inmediata avisando mediante señales acústicas y visuales, dando además aviso con voz a los distintos teléfonos programados.

1.3.4 Gestión de la iluminación

El alumbrado es regulado por el sistema dependiendo de la claridad y de la hora del día. El sistema puede conectar y desconectar automáticamente toda la iluminación del edificio. Lo cual ahorra energía y reduce el gasto.

El sensor de presencia controla que la luz esté solamente conectada cuando realmente hace falta, lo que resulta especialmente apropiado para espacios que se usan sólo por cortos intervalos de tiempo, tales como pasillos, escaleras o baños.

1.3.5 Control de cortinas, persianas y toldos

El sistema debe detectar cambios climáticos, por ejemplo, los toldos se recogen automáticamente en caso de temporal o lluvia, y en cambio se extienden por sí solos

cuando sale el sol, de este modo, el sistema alarga la vida útil del toldo y aumenta el confort del usuario.

Al llegar la noche las persianas se bajan y los toldos se suben. Por la mañana, el proceso será al contrario y el sol entrará en la casa.

Se garantiza confort y ahorro al controlar las pérdidas nocturnas y aprovechar la energía solar del día. En épocas cálidas las persianas interceptan los rayos del sol, evitando sobrecalentamientos y logrando un considerable ahorro climático.

1.4 Conceptos técnicos

Para poder clasificar técnicamente un sistema de automatización de viviendas, es necesario tener claros una serie de conceptos técnicos, como son: tipo de arquitectura, medio de transmisión, velocidad de transmisión y protocolo de comunicaciones.

1.4.1 Tipos de Arquitectura

La arquitectura de un sistema domótico, como la de cualquier sistema de control, especifica el modo en que los diferentes elementos de control del sistema se van a ubicar. Existen dos arquitecturas básicas: la arquitectura centralizada y la distribuida.

Arquitectura centralizada: Es aquella en la que los elementos a controlar y supervisar (sensores, luces, válvulas, etc.) han de cablearse hasta el sistema de control de la vivienda (PC o similar). El sistema de control es el corazón de la vivienda, en cuya falta, todo deja de funcionar, si su instalación no es compatible con la instalación eléctrica convencional en cuanto a que en la fase de construcción hay que elegir esta topología de cableado.

Arquitectura distribuida: Es aquella en la que el elemento de control se sitúa próximo al elemento a controlar.

Hay sistemas que son de arquitectura distribuida en cuanto a la capacidad de proceso, pero no lo son en cuanto a la ubicación física de los diferentes elementos de control y viceversa, sistemas que son de arquitectura distribuida en cuanto a su capacidad para ubicar elementos de control físicamente distribuidos, pero no en cuanto a los procesos de control, que son ejecutados en uno o varios procesadores físicamente centralizados.

1.4.2 Medios de Transmisión

En todo sistema domótico con arquitectura distribuida, los diferentes elementos de control deben intercambiar información unos con otros a través de un soporte físico (par trenzado, línea de potencia o red eléctrica, radio, infrarrojos, etc.).

Líneas de distribución de energía eléctrica (Corrientes portadoras)

Si bien no es el medio más adecuado para la transmisión de datos, sí es una alternativa a tener en cuenta para las comunicaciones domésticas dado el bajo coste que implica su uso.

Para aquellos casos en los que las necesidades del sistema no impongan requerimientos muy exigentes en cuanto a la velocidad de transmisión, la línea de distribución de energía eléctrica puede ser suficiente como soporte de dicha transmisión.

Dada las especiales características de este medio, sobre todo su idoneidad para las instalaciones domésticas, a continuación se detallan sus principales ventajas e inconvenientes:

- Nulo costo de la instalación.
- Facilidad de conexionado.
- Poca fiabilidad en la transmisión de los datos.
- Baja velocidad de transmisión.

El sistema consta de:

- Unidad de control: encargada de gestionar el protocolo, almacenar las órdenes y transmitirlas a la red.
- Interfaz de conexión de los equipos: es el elemento que recibe las órdenes de la unidad de control y las ejecuta.
- Filtro: para evitar que las señales puedan afectar la red eléctrica exterior a la vivienda.

Soportes metálicos

La infraestructura de las redes de comunicación actuales tienen un porcentaje muy elevado de cables metálicos de cobre como soporte de transmisión. En general se pueden distinguir dos tipos:

Par metálico

Los cables formados por varios conductores de cobre pueden dar soporte a un amplio rango de aplicaciones en el entorno doméstico.

Los denominados cables de pares están formados por cualquier combinación de los tipos de conductores que a continuación se detallan:

1. Cables formados por un solo conductor con un aislamiento exterior plástico. (Por ejemplo los utilizados para la transmisión de las señales telefónicas.).

2. Par de cables, cada uno de los cables esta formado por un arrollamiento helicoidal de varios hilos de cobre. (Por ejemplo los utilizados para la distribución de señales de audio.).

3. Par apantallado, formado por dos hilos recubiertos por un conductor trenzado en forma de malla cuya misión consiste en aislar las interferencias electromagnéticas de las señales que circulan por los cables de exteriores. (Por ejemplo los utilizados para la distribución de sonido de alta fidelidad o datos).

4. Par trenzado, está formado por dos hilos de cobre recubiertos cada uno por un trenzado en forma de malla. El trenzado es un medio para hacer frente a las interferencias electromagnéticas. (Por ejemplo los utilizados para la interconexión de computadoras).

Coaxial

Un par coaxial es un circuito físico asimétrico, constituido por un conductor filiforme que ocupa el eje longitudinal del otro conductor en forma de tubo, manteniéndose la coaxialidad de ambos mediante un dieléctrico apropiado.

Este tipo de cables permite el transporte de las señales de video y señales de datos a alta velocidad.

Dentro del ámbito de la vivienda, el cable coaxial puede ser utilizado como soporte de transmisión para:

- Señales de teledifusión que provienen de las antenas (red de distribución de las señales de TV y FM).
- Señales procedentes de las redes de TV por cable. Señales de control y datos a media y baja velocidad.

Fibra Óptica

Capítulo l ¿Qué es la domótica?

La fibra óptica es el resultado de combinar dos disciplinas no relacionadas, como son la tecnología de semiconductores (que proporciona los materiales necesarios para las fuentes y los detectores de luz), y la tecnología de guiado de ondas ópticas (que proporciona el medio de transmisión, el cable de fibra óptica).

A continuación se detallan sus ventajas e inconvenientes:

- Fiabilidad en la transferencia de datos.
- Inmunidad frente a interferencias electromagnéticas y de radiofrecuencias.
- Alta seguridad en la transmisión de datos.
- Distancia entre los puntos de la instalación limitada, en el entorno doméstico estos problemas no existen.
- Elevado coste de los cables y las conexiones.
- Transferencia de gran cantidad de datos:

Conexión sin hilos

Infrarrojos

El uso de mandos a distancia basados en transmisión por infrarrojos está ampliamente extendido para telecomando de equipos de Audio y Video.

La comunicación se realiza entre un diodo emisor que emite una luz en la banda de IR, sobre la que se superpone una señal, convenientemente modulada con la información de control, y un fotodiodo receptor cuya misión consiste en extraer de la señal recibida la información de control.

Los controladores de equipos domésticos basados en la transmisión de ondas en la banda de los infrarrojos tienen las siguientes ventajas:

- Comodidad y flexibilidad.
- Admiten gran número de aplicaciones.

Al tratarse de un medio de transmisión óptico es inmune a las radiaciones electromagnéticas producidas por los equipos domésticos o por los demás medios de

transmisión (coaxial, cables pares, red de distribución de energía eléctrica, etc.). Sin embargo, habrá que tomar precauciones en los siguientes casos:

- Las interferencias electromagnéticas sólo afectarán a los extremos del medio IR, es decir, a partir de los dispositivos optoelectrónicos (diodo emisor y fotodiodo receptor).

Radiofrecuencias

La introducción de las radiofrecuencias como soporte de transmisión en la vivienda, ha venido precedida por la proliferación de los teléfonos inalámbricos y sencillos telemandos.

Este medio de transmisión puede parecer, en principio, idóneo para el control a distancia de los sistemas domóticos, dada la gran flexibilidad que supone su uso. Sin embargo resulta particularmente sensible a las perturbaciones electromagnéticas producidas, tanto por los medios de transmisión, como por los equipos domésticos.

A continuación se detallan las ventajas e inconvenientes de los sistemas basados en transmisión por radiofrecuencias:

- Alta sensibilidad a las interferencias.

- Fácil intervención de las comunicaciones.

- Dificultosas para la integración de las funciones de control y comunicación, en su modalidad de transmisión analógica.

1.4.3 Velocidad de Transmisión

La velocidad a la cual intercambian información los diferentes elementos de control de la red se denomina velocidad de transmisión.

1.4.4 Protocolo de comunicaciones

Una vez establecido el soporte físico y la velocidad de comunicaciones, un sistema domótico se caracteriza por el protocolo de comunicaciones que utiliza, que no es otra cosa que el idioma o formato de los mensajes que los diferentes elementos de control del sistema deben utilizar para entenderse unos con otros y que puedan intercambiar su información de una manera coherente. Dentro de los protocolos existentes, se puede realizar una primera clasificación atendiendo a su estandarización:

Protocolos estándar: Los protocolos estándar son los que de alguna manera son utilizados ampliamente por diferentes empresas y estas fabrican productos que son compatibles entre sí.

Protocolos propietarios: Son aquellos que desarrollados por una empresa, solo ella fabrica productos que son capaces de comunicarse entre sí.

1.4.5 Preinstalación domótica

La preinstalación domótica es la posibilidad de dejar preparada una vivienda para que, con el menor número de actuaciones, se le pueda instalar el sistema domótico en el momento en que el usuario lo demande. Para que un sistema pueda ofrecer una verdadera preinstalación domótica en una vivienda, ha de ser compatible con la instalación eléctrica actual, de tal manera que el usuario pueda, en la fase de construcción, elegir la preinstalación domótica y la instalación eléctrica convencional y con posterioridad, realizar cualquier tipo de automatización de su vivienda.

1.4.6 Descripción del tipo de nodos

Una red domótica de arquitectura distribuida está compuesta por una serie de nodos que se conectan unos con otro a través del bus de comunicaciones, el cual lleva dos hilos para datos y dos para la alimentación. Así tenemos:

Nodos de control estándar: son los encargados de controlar los parámetros de cada estancia. Cada uno soporta dos circuitos independientes de conmutación y dos entradas

extra para sensores. La funcionalidad del nodo depende del programa que se cargue en el nodo.

Nodos de supervisión: son nodos dedicados a realizar la interfaz con el usuario. Cada función que el usuario necesita para supervisar y controlar el sistema está implementada en el correspondiente nodo. De esta manera, el usuario puede elegir para su vivienda las funciones que considere necesarias.

Nodo de alarmas técnicas: (Agua, Gas, Humo y Fuego)

Nodo de vigilancia de intrusión: (Simulación de presencia, vigilancia)

Nodo de sirena interior: (Prueba de avisador acústico externo y rearme de alarmas)

Nodo de luces exteriores: (Activación manual y automática con el sensor de luz)

Nodo telefónico: Realiza la interfaz entre la red domótica y la red telefónica.

Nodos exteriores: dentro de este tipo de nodos se agrupan aquellos que siendo de uso dedicado se instalan en el exterior de la vivienda. Dentro de ellos podemos destacar el nodo de sirena exterior y el nodo medidor de luz exterior.

Nodos de comunicaciones: estos son nodos dedicados específicamente a soportar la red de comunicaciones de la vivienda. Entre ellos podemos destacar:

- **Nodo repetidor**: Se utiliza para extender en longitud la red de comunicaciones de la vivienda, cuando esta supere los 1000m, o para aislar galvánicamente sectores de la red.

- **Nodos *Routers***: El nodo router realiza una adaptación física y lógica de dos medios de transmisión diferentes.

1.4.7 Criterios

Ante la elección de un sistema de automatización de viviendas, se deben observar dos tipos de criterios.

Criterios de usuario

1. Posibilidad de realizar la preinstalación del sistema en la fase de construcción.

2. Facilidad de ampliación e incorporación de nuevas funciones.

3. Simplicidad de uso.

4. Grado de estandarización e implantación del sistema.

5. Variedad de elementos de control y funcionalidades disponibles.

6. Tipo de servicio posventa.

Criterios técnicos

1. Tipo de arquitectura.

2. Topología.

3. Velocidad de transmisión.

4. Medios de transmisión.

5. Tipo de protocolo.

6. Fabricación de elementos por terceras partes.

1.4.8 Unidad de alimentación

La unidad de alimentación es la encargada de suministrar energía a los diferentes elementos activos (sensores, nodos, electroválvulas, etc.). Estas incorporan una batería (para

vigilancia de intrusión) con autonomía suficiente para varias horas de ausencia de suministro eléctrico.

1.5 El Estándar X-10

La tecnología X-10 de corrientes portadoras fue desarrollada entre 1976 y 1978 por ingenieros en *Pico Electronics Ltd*, en *Glenrothes*, Escocia. Estos desarrollaron el primer módulo que podía controlar cualquier dispositivo a través de la línea de corriente doméstica (120 ó 220 v. y 50 ó 60 hz), modulando impulsos de 120 khz (ausencia de este impulso = 0, presencia de este impulso = 1). Con un protocolo sencillo de direccionamiento se podía identificar cualquier elemento de la red, en total 256 direcciones. El protocolo contemplaba 16 grupos de direcciones llamados "*housecodes*" y 16 direcciones individuales llamadas "*unit codes*".

A este protocolo se le añadieron "tiras" de comandos llamados "*control strings*" que no son más que ceros y unos agrupados formando comandos; en total eran 6: encendido, apagado, reducir, aumentar, todo encendido, todo apagado. Estas señales las podían recibir todos los módulos, pero sólo actuaba sobre aquel al que iba dirigida (los primeros bits de la señal eran el identificador del módulo). La frecuencia de transmisión era la de la corriente eléctrica (50 ó 60 hz), y la señal completa incluyendo dirección y función ocupaba 48 bits., o sea que para mandar una señal a un dispositivo a una frecuencia de 50 hz (en informática

hablaríamos de un ancho de banda de 50 bits por segundo) se tardaría casi un segundo (si enviáramos 50 bits tardaríamos un segundo).

Las transmisiones X-10 se sincronizan con el paso por el cero de la corriente alterna. Las interfaces *Power Line* (tecnología inalámbrica) proporcionan una onda de 50 Hz con un retraso máximo de 100 µseg desde el paso por el cero de la corriente alterna. El máximo retraso entre el comienzo del envío y los pulsos de 120 KHz es de 50 µseg.

Un 1 binario se representa por un pulso de 120 KHz durante 1 milisegundo, en el punto cero, y el 0 binario se representa por la ausencia de ese pulso de 120 KHz. El pulso de 1 milisegundo se transmite tres veces para que coincida con el paso por el cero en las tres fases para un sistema trifásico. La Figura 1.1 muestra la relación entre estos pulsos y el punto cero de la corriente alterna.

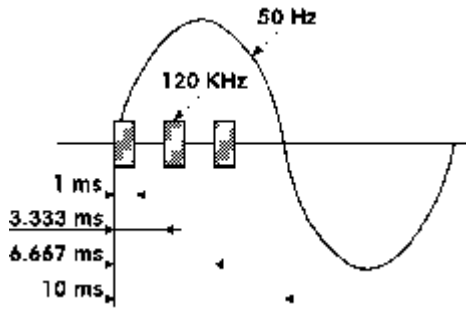

Figura 1.1

Nota: Para una mayor claridad, las señales de la Figura 1.1 se muestran tal como se verían a través de un filtro paso alto. La forma de la curva de 50 Hz. sólo se muestra como referencia. En realidad, las señales van superpuestas con la curva de 50 Hz. y su resultado es más similar al de la Figura 1.2.

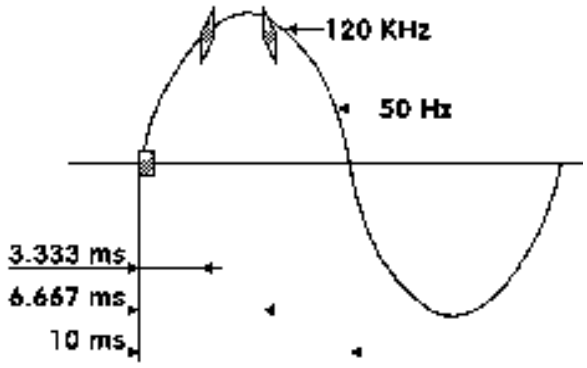

Figura 1.2

La transmisión completa de un código X-10 necesita once ciclos de corriente. Los dos primeros ciclos representan el Código de Inicio. Los cuatro siguientes ciclos representan el Código de Casa (letras A-P), los siguientes cinco representan o bien el Código Numérico

(1-16) o bien el Código de Función (Encender, Apagar, Aumento de Intensidad, etc). Este bloque completo (Código de Inicio, Código de Casa y Código de Función o Numérico) se transmite siempre dos veces, separando cada 2 códigos por tres ciclos de la corriente, excepto para funciones de regulación de intensidad, que se transmiten de forma continua (por lo menos dos veces) sin separación entre códigos.

Las funciones de regulación de intensidad son excepciones a esta regla, y se transmiten de forma continua (al menos dos veces) sin separación entre códigos. Ver figura 1.3

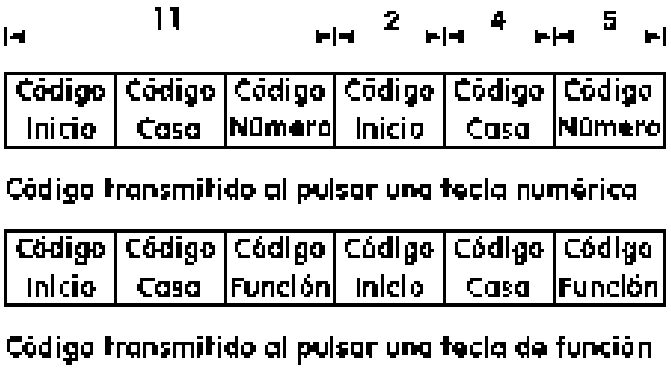

Figura 1.3

Dentro de cada bloque de códigos, cada cuatro o cinco bits de código deben ser transmitidos en modo normal y complementario en medios ciclos alternados de corriente. Por ejemplo, si un pulso de 1 milisegundo se transmite en medio ciclo (1 binario), entonces no se transmitirá nada en la siguiente mitad del ciclo (0 binario). Ver figura 1.4

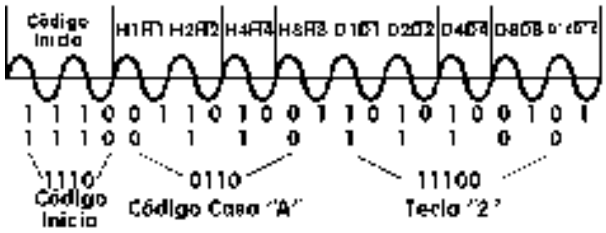

Figura 1.4

X10 es el "lenguaje" de comunicación que utilizan los productos compatibles X10 para hablarse entre ellos y que le permiten controlar las luces y los electrodomésticos del hogar, aprovechando para ello la instalación eléctrica existente de 220V de la casa, y evitando tener que instalar cables.

Cada dispositivo tiene una dirección a la que responde o envía, existiendo un total de 256 direcciones. Todos los productos X10 son compatibles entre sí por lo que se pueden combinar para formar el sistema más adecuado a las preferencias del usuario.

La gran diferencia del sistema X10, es que este es inteligente y los aparatos interactúan entre sí. Por ejemplo al entrar por la tarde, el sensor de presencia puede encender secuencialmente el pasillo, el dormitorio, conectar el calentador y encender dos lámparas de la sala al 50%. Veamos un ejemplo en la figura 1.5 de cómo se pueden aplicar los productos X10 en el hogar.

Con la interfaz para PC y un software específico, se puede controlar toda la casa desde la computadora, programar encendidos, preparar macros de comandos y después volcarlos en la interfaz para que todo siga funcionando incluso con la máquina apagada.

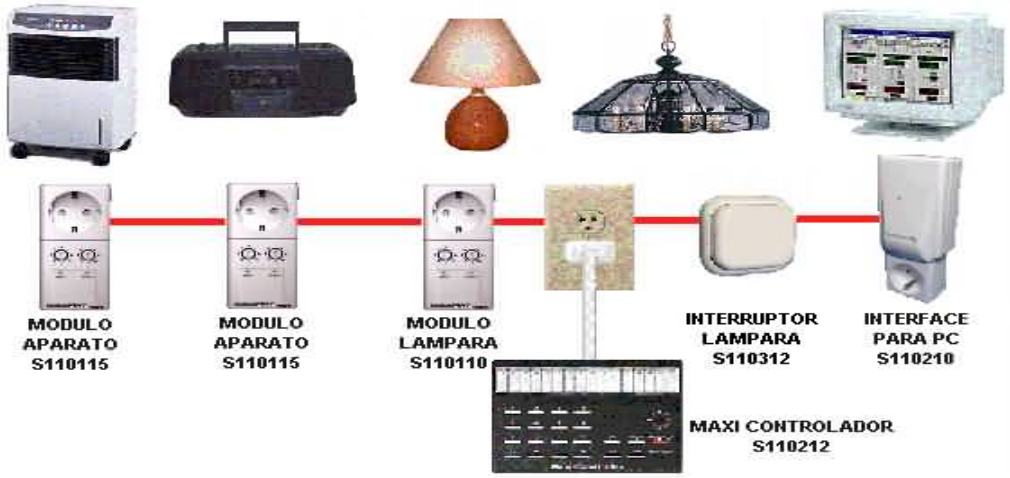

Figura 1.5

1.6 Tecnología LonWorks

Otra tecnología es LonWorks que proporciona una plataforma completa para controlar las conexiones de redes. Esta incluye no sólo un protocolo o un transceptor, sino también los estándares de interoperabilidad y un API de software universal que funciona como un todo. A continuación veremos los diferentes componentes de esta plataforma LonWorks.

1.6.1 Los medios físicos de LonWorks

En un entorno desafiante y variado como el doméstico, cualquier solución de conexión de redes domésticas tiene que ser capaz de adaptarse a las necesidades del dueño de la vivienda y a las condiciones especiales de la propia casa. Por ejemplo, es más rentable para la vivienda actual utilizar la energía eléctrica como medio de conexión, mientras que en las viviendas de nueva construcción un doble cableado entrelazado se puede sacar fácilmente de cualquier parte de la casa.

También sería deseable poder mezclar diferentes tipos de medios dentro de una sola vivienda, así, por ejemplo, una parte de la red doméstica opera con energía eléctrica, otras partes con doble cable, alguna sección a través de la frecuencia de radio y otra incluso por

Infrared. La plataforma LonWorks es independiente de los medios de comunicación. La fiabilidad es por supuesto tan importante como la flexibilidad.

1.6.2 El protocolo LonTalk

La tecnología LonWorks fue diseñada hace acerca de diez años por la Corporación Echelon como una plataforma universal para casi cualquier sistema de control. La tecnología, principalmente el protocolo y el medio de programación fueron diseñados para ocuparse de las idiosincrasias y demandas de las redes de control. El protocolo LonTalk es una completa realización de todas las 7 capas del modelo de referencia 0SI .

El protocolo está abierto al uso de cualquiera en cualquier plataforma (programación) y procesador. Los Chips Neurona realizan el protocolo LonTalk con Sicilio y Toshiba, Cypress Semiconductor y Motorola los manufacturan y los venden por todo el mundo.

Actualmente hay alrededor de 8.5 millones de mecanismos instalados por todo el mundo que llevan incorporado un Chip Neurona.

1.6.3 Asociación de Interoperabilidad Lonmark

La Asociación de Interoperabilidad Lonmark se formó en Mayo de 1994 por 36 compañías y ya tiene unos 230 adeptos inventando el quién es quién en el mundo de las redes de

control. La misión de la Asociación LonMark es permitir la fácil integración de sistemas multivendedor basados en las redes de LonWorks. La Asociación ofrece un forum abierto a las compañías miembro para trabajar conjuntamente en *marketing* y programas técnicos con el fin de fomentar la disponibilidad de mecanismos de control interoperables y abiertos.

Los productos que se han verificado para ajustarse a las pautas de interoperabilidad LonMark son elegibles para llevar el logotipo LonMark. El logotipo LonMark es un indicador de que un producto ha completado las pruebas de conformidad y ha sido diseñado para operar a través de una red LonWorks.

La Plataforma de Software para las Redes de Control, el software *LonWorks Network Services* (LNS) proporciona un API vigoroso para el manejo de las redes, diagnósticos, supervisión y control; LNS ya se utiliza en centenares de empresas en todo el mundo para desarrollar herramientas de software interoperables y fáciles de usar para los que lo instalen y para los usuarios finales.

El soporte IP incorporado de LNS permite una integración sin costuras de una red de control doméstica con Internet. Este es el software que proponemos usar en el Capitulo II cuando hablamos de LonWorks como sistema de bus de campo para el manejo de las variables en el edificio.

1.7 Conclusiones

- En este capítulo pudimos observar las principales áreas de los sistemas electrónicos orientados a edificios inteligentes y sus funcionalidades relacionadas con la integración, vimos que existe un marco muy amplio tanto en las áreas como en las funciones específicas a ejecutar en cada una de estas.

- Desarrollamos una leve descripción de dos de las tecnologías actuales más difundidas y aplicadas en el mundo de la domótica, esfera en la que se lleva a cabo una lucha por la estandarización de los medios y protocolos, donde se hace muy difícil elegir una tecnología, razón por la cual concluimos que hay que tener bien claro todos los conceptos técnicos expuestos en esta sección con el fin de realizar una buena selección.

Capítulo ll Sistema Multiagente

2.1 Introducción al sistema multiagente

En el nacimiento de cualquier nueva tecnología o servicio, el grado de implicación de la parte técnica es alto y se tiende a complicar su uso por la incorporación de cientos de funciones, programaciones, etc. En el caso de servicios o sistemas orientados a usuarios

finales, esta tendencia agrava la situación porque el usuario se encuentra ante un sistema que técnicamente puede ser muy aceptable pero que en la práctica, ante cualquier evento, al usuario le producirá confusión, desconcierto y finalmente rechazo. Por tanto debemos acudir a diferentes formas de enfrentarnos a problemáticas de tal complejidad. En este caso nos referiremos a un enfoque multiagente que es una manera de modularizar las funciones de un sistema y descubrimos que es muy utilizado en el mundo de la domótica, que además presenta a la inteligencia artificial como una solución a estos problemas.

Este capítulo explora un método para desarrollar sistemas de edificios inteligentes, un edificio se considera inteligente si está capacitado para aprender según la experiencia. Demostraremos una avanzada arquitectura multiagente para el control de estos. El sistema multiagente está conectado a sensores y actuadores mediante una red de bus de campo (que pudiera ser LonWorks), basado en datos este sistema toma decisiones sobre las acciones que se quieren ejecutar, estas son tomadas por un sistema de inferencia *fuzzy* sobre la base de un bloque de reglas difusas. El sistema utiliza una estructura de información genérica para resolver las complicadas relaciones entre sensores, actuadores y estructuras estáticas como habitaciones.

Como decíamos anteriormente en este trabajo describimos nuestro enfoque de cómo hacer edificios inteligentes, para lograrlo usamos un algoritmo de aprendizaje específicamente desarrollado para este propósito, nuestro enfoque es aprender sobre reglas difusas adquiriendo datos de los sensores y adaptarlas. Un edificio inteligente, desde un punto de vista computacional, debe tomar decisiones tales que el usuario no tenga que hacerlas y ejecutarlas el mismo. Un usuario no debe notar que el edificio en el que está es en realidad inteligente, lo cual significa que no hay dispositivos especiales de entrada con los cuales el sistema que maneja el edificio pueda interactuar con el usuario. Las únicas herramientas de interacción son los sensores y actuadores presentes en un edificio ordinario, detectores de presencia, sensores de tiempo, interruptores, luces y persianas.

Los usuarios y sus preferencias están en un cambio constante. La consecuencia de esto es que un edificio inteligente tiene que aprender continuamente (*anytime learning*). Algo aprendido sobre el comportamiento de un usuario puede ser válido en tiempo presente pero puede no serlo en el futuro, porque sus preferencias cambiaron, no hay fase de

entrenamiento ni de producción en tal sistema, el sistema tiene que aprender durante toda su vida.

Nuestro punto de vista de un edificio inteligente es muy similar al de los robots, es decir, consideramos un sistema inteligente como un robot. La única diferencia entre un robot ordinario y un edificio inteligente es que personas reales están presentes dentro del edificio. Un sistema autónomo controlando un edificio necesita tomar todas las decisiones acerca del tiempo real, porque un retardo muy notable entre el cambio del sensor y la acción a ejecutar no será aceptado por el usuario.

Tratamos estos retos usando un sistema multiagente, las decisiones se toman por diferentes agentes. Cada agente es responsable de un subconjunto de todo el espacio de estado, este énfasis localizado para la toma de decisiones hace al sistema más rápido. El mismo principio es aplicado al aprendizaje. Cada unidad aprende sobre un pequeño subconjunto de todo el espacio de estado, lo cual garantiza una más rápida convergencia del aprendizaje.

Para juzgar la calidad del sistema se necesita probarlo en la realidad durante un período de tiempo largo, para esto debemos desarrollar un simulador que finge la conducta del edificio con sus usuarios. Este debe incluir la influencia de factores externos como el clima que cambia constantemente. El simulador nos deja probar un sistema específico y sus parámetros de una manera más rápida que en la vida real, porque es capaz de ejecutarse con una mayor velocidad que en el mundo real.

2.2 Motivación

Un ambiente como un edificio es particularmente interesante y un medio retador para aplicar la inteligencia artificial y el aprendizaje. Los edificios son un ambiente usado por personas reales que pudieran manifestarse de una forma inesperada. No es ningún ejemplo diseñado específicamente, los cuales son usados para verificar nuevos algoritmos y arquitecturas. Un ambiente de gente real a diario sufre cambios adicionales, lo cual hace a los algoritmos a aplicar más interesantes y complejos.

Un edificio es un sistema complejo, desde un punto de vista computacional se comporta completamente no determinista. Pensamos que este ambiente se puede controlar con un enfoque de inteligencia artificial distribuida, diferentes agentes no se comunican

directamente con otros, pero ellos tienen metas y acciones que cumplir. La única comunicación indirecta entre los agentes son los propios cambios en el medio.

Un edificio inteligente tiene mucha similitud con la robótica, un buen ejemplo es un robot moviéndose libremente en una habitación, como el robot tiene que lidiar con un medio totalmente no determinista, este no puede influir en lo que pasa a su alrededor de ninguna forma, solo reaccionar a este. Por esto consideramos un edificio inteligente como un robot. La única diferencia entre un robot tradicional y un edificio inteligente es la posición en la cual se encuentran las personas del mundo real, en uno las personas están alrededor y en el otro viven dentro de él mismo. Solo un medio real puede comprobar cuan inteligente es un sistema. El uso de comunicación indirecta entre los agentes a través del medio también depende de la capacidad del medio de ser modificado por los actuadores y de qué puede ser captado por los sensores.

2.3 Arquitectura del sistema

2.3.1 Principios Básicos

Los dispositivos de entrada y salida del sistema son sensores y actuadores los cuales pueden ser accedidos por una red. La figura 2.1 muestra un ejemplo de una arquitectura del hardware. Todos los sensores y actuadores de todas las habitaciones están conectados a la misma red. Esta red es, con la ayuda de un *gateway*, accesible desde una red basada en IP.

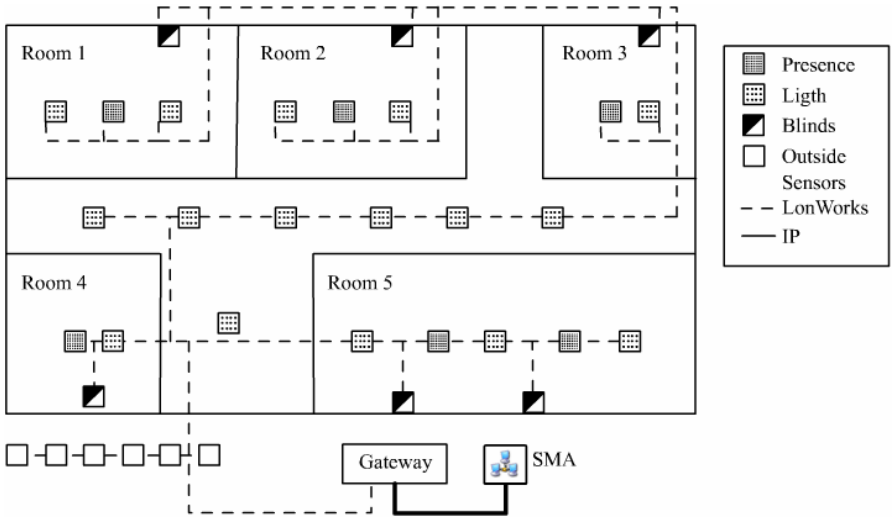

Figura 2.1 El piso de un edificio estructurado en habitaciones

2.3.2 Conectividad

El sistema es una combinación de dos arquitecturas de redes completamente diferentes: red bus de campo y redes de computadora como *Ethernet*. Un edificio de oficinas moderno está típicamente cableado con estos sistemas de bus. El bus de campo (en este caso LonWorks) es usado para comunicarse con los sensores y actuadores del sistema de automatización del edificio y la red de computadoras, para comunicarse entre los diferentes agentes del sistema.

Las dos redes están conectadas con la ayuda de un *gateway* que está conectado a ambas. El *gateway* direcciona los datos entre la red de computadoras y el bus de campo de forma que todas las partes del software pueden ejecutarse en las computadoras sin ningún acceso directo al bus de campo, es decir es un elemento que de cierta forma aísla las dos redes para que cada una realice sus funciones y a la vez las enlaza para permitir la comunicación entre estas.

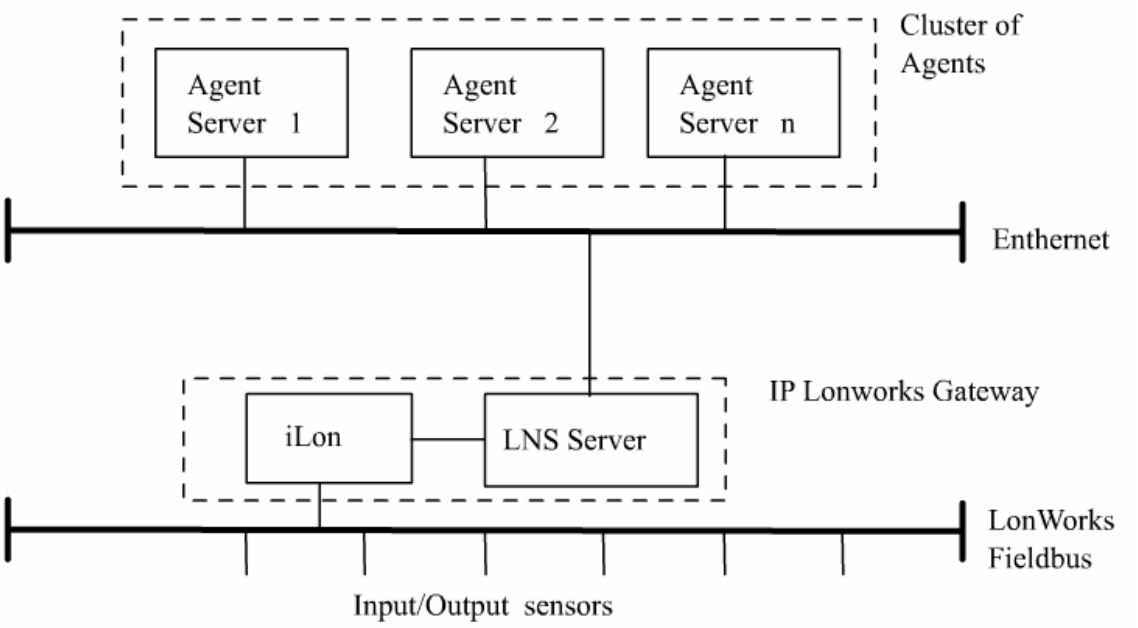

Figura 2.2 Arquitectura de redes

Esta figura muestra la arquitectura de redes, esta es independiente de algún tipo específico de sistema de bus, el bus de LonWorks es usado como una posibilidad, pudiera ser utilizado por ejemplo EIB u otro cualquiera. LNS e iLon son productos específicos de LonWorks que actúan como un **IP-LonWorks Gateway**. La comunicación entre iLon (el cual es el **IP-LonWorks Gateway** real) y el servidor LNS (que es solo un Software) está ya hecho sobre IP, entonces el único dispositivo de hardware que requiere acceso directo al bus LonWorks es el iLon, mediante el cual captaremos todos los datos provenientes de los dispositivos instalados en el edificio.

2.4 Arquitectura del Software

Este sistema está realizado como una colección de agentes los cuales juntos forman el sistema multiagente (SMA). Cada agente es responsable de una tarea específica en el sistema y ofrece esta tarea como un servicio a otros agentes. No existe un agente central que actúe como un coordinador, todos los agentes actúan independientemente de otros y persiguen sus metas especificas.

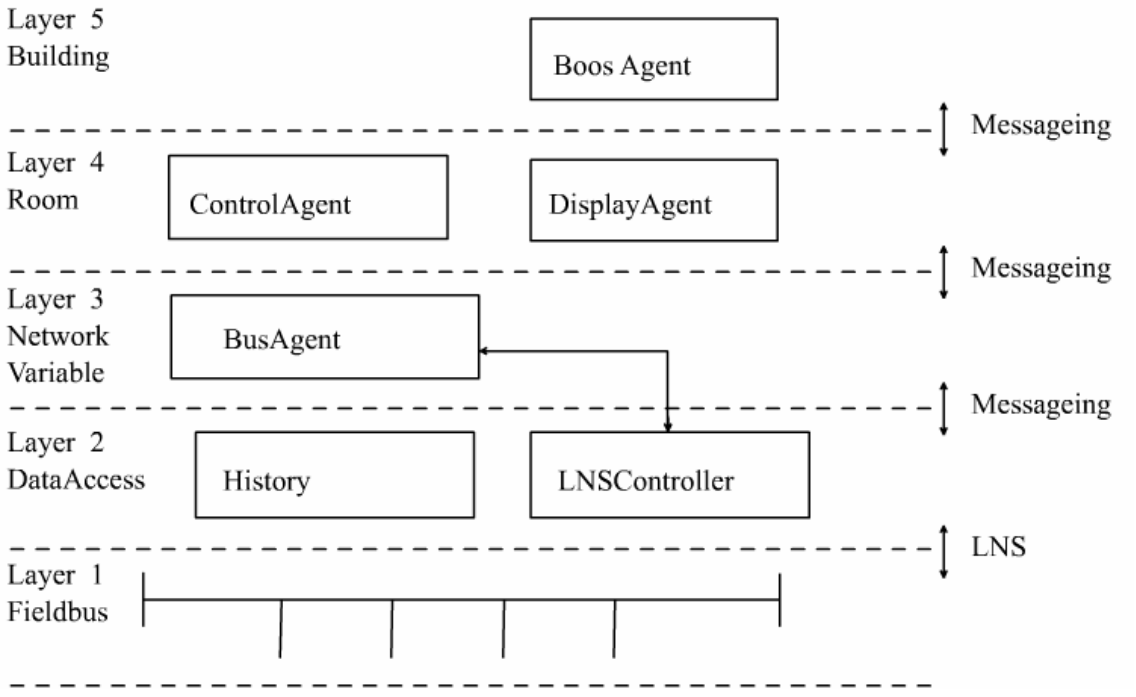

Figura 2.3 Arquitectura del sistema

El SMA está estructurado en cinco Capas. Cada agente en el sistema pertenece a una de estas. Es posible para los agentes que se comuniquen directamente con otros que no están directamente una capa arriba o debajo de la suya. Este modo de señalar es para estructurar el sistema de una forma que los niveles altos tengan trato con menor cantidad de datos que los agentes de bajo nivel.

Las cinco Capas son:

- Capa 1: Edificio, todo lo relacionado al edificio o edificios.
- Capa 2: Habitación, todo directamente relacionado con la habitación.
- Capa 3: Variable de red, todo lo relacionado con una variable de red o propiedad de esta.
- Capa 4: Servicios Base / Acceso a Datos (servicios de soporte, *gateways*)
- Capa 5: Capa de abstracción de la red (hardware, cableado)

Las instancias de los agentes se crean según las reglas siguientes:

- **BusAgent**: Uno por tipo de sistema, abstracción del bus de campo en el nivel de las variables de red.
- **RoomDisplayAgent**: Visualiza toda la información disponible y necesaria sobre una habitación en particular; control manual de todas las salidas de los actuadores.
- **ControlAgent**: Procesa los datos de entrada y toma decisiones; distribuye y ejecuta las decisiones.
- **HistoryAgent**: Lleva la historia de cada uno de los estados del sistema y sus actualizaciones en una base de datos para uso posterior.

- **BossAgent**: Responsable de la creación dinámica de nuevas instancias como sensores, habitaciones o edificios.

Múltiples instancias de estos agentes se están ejecutando al mismo tiempo en el sistema. Todas estas instancias en su conjunto forman lo que se llama sistema multiagente.

- **RoomDisplayAgent**: Arbitrariamente o por la demanda de los usuarios.
- **ControlAgent**: Uno por habitación.
- **HistoryAgent**: Uno por base de datos
- **BossAgent**: Uno por sistema

2.5 Colaboración multiagente

El sistema no puede ser demasiado dependiente porque los agentes tendrían que conocer exactamente a los otros agentes alrededor del sistema y qué hacen. Debido a esto, toda la comunicación entre agentes se hace encima de un intermediario, que es el encargado de distribuir los mensajes a los agentes que están interesados en su volumen. Los agentes se registran a si mismos con otros agentes, de los que ellos quieren recibir mensajes de un cierto tipo en el futuro. El remitente de un mensaje no tiene que saber cuántos (si hay alguno) agentes están interesados en el mensaje que él envía. Ésta es más bien la tarea del intermediario. También es la responsabilidad del intermediario quitar automáticamente agentes que bajaron sin registrar o que ya no están procesando mensajes.

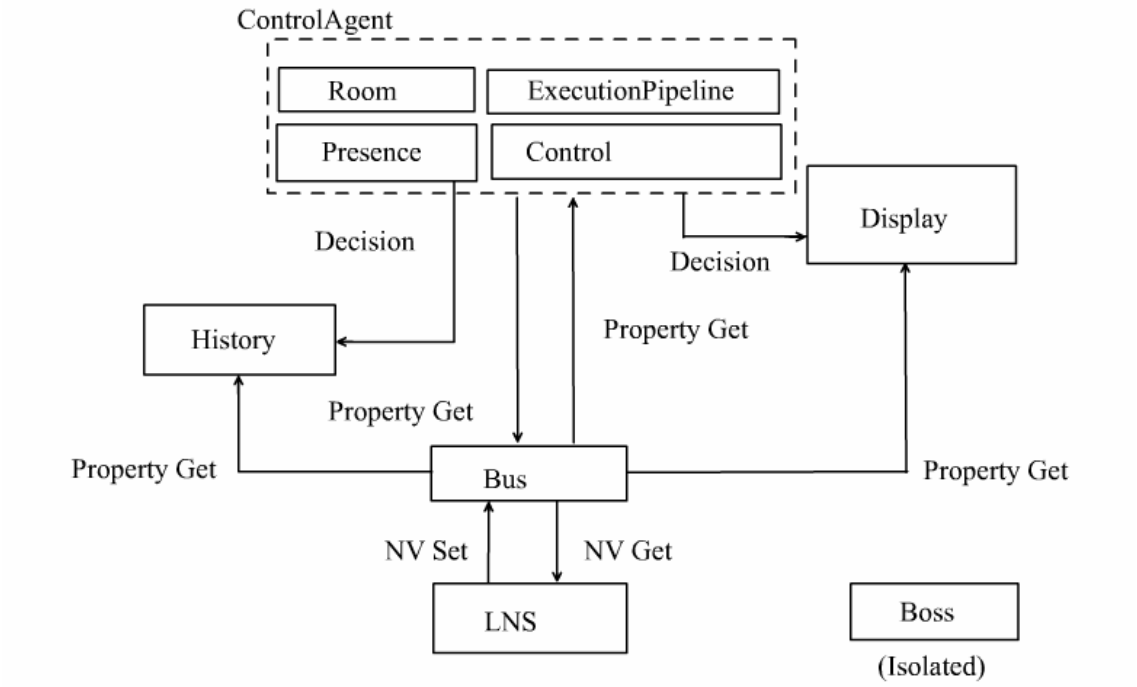

Figura 2.4 Colaboración entre agentes

La figura 2.4 muestra cómo los agentes están intercambiando mensajes. Hay tres tipos de mensajes: VR (variables de la red), *set/get* propiedades y decisiones *set/get*. El nivel más bajo de abstracción es el nivel de variables de la red (VR), se reciben exclusivamente valores de VR y son enviados por el *BusAgent*. La sintaxis de VR es propietaria y depende de la norma de bus de campo usada. La segunda capa de abstracción en comunicación es la capa de propiedades. Las propiedades del sistema son genéricas e independientes en representación de variables de la red. Los Agentes se pueden registrar como oyentes para las propiedades que ellos notifiquen por el *BusAgent* cuando hay un nuevo valor disponible. Los agentes también pueden decirle al *BusAgent* que asigne un nuevo valor a una propiedad. El nivel más alto de abstracción es el nivel de decisiones. Estos mensajes sólo son enviados por el agente de control y pueden ser recibidos por cualquier agente interesado en las decisiones tomadas por el sistema. El agente de control también ejecuta las decisiones tomadas.

2.6 Requisitos de implementación

Los requisitos para la implementación de este sistema son los siguientes:

- Bibliotecas: Allí debe reforzarse las bibliotecas para la construcción de sistemas multiagente disponibles.

- Informática distribuida: los ambientes de ejecución deben apoyar la Informática distribuida (colección de basura distribuida, la invocación del método remota).

- Disponibilidad de API para los sistemas de bus de campo: los API para los sistemas de bus de campo deben estar disponible para el lenguaje de la programación escogido.

El SMA requiere un arquitectura bajo los criterios de evaluación para:

- Apoyo para los agentes físicamente distribuidos (distribuidos entre múltiples *workstations*)

- Mensajería asincrónica (para intercambiar mensajes entre agentes)

- Apoyo para comunicación entre agentes.

- Apoyo para técnicas de toma de decisiones como la inferencia de lógica difusa o ILP (lógica inductiva)

- Apoyo para la programación concurrente.

- El sistema necesita ser estable.

Un lenguaje de programación apropiado, el ambiente de tiempo de ejecución y la arquitectura del SMA serán escogidos según estas condiciones.

2.7 Diseño e implementación

Este epígrafe perfila las tecnologías para construir el SMA. Estas tecnologías serán escogidas debido a los requisitos dados en epígrafe anterior.

Lenguaje de programación y ambiente de tiempo de ejecución

Después de evaluar varias opciones según los requisitos dados en la sección 2.7 llegamos a la conclusión que el lenguaje de programación que utilizaremos es Delphi. Esto es debido a las razones siguientes:

- Bibliotecas: Las bibliotecas reforzadas para la construcción de sistemas multiagente están disponibles
- Soporta el uso de automatización OLE para la comunicación con Matlab que es donde se implementará el controlador difuso.

Nota: Se debe verificar la disponibilidad de API para los sistemas de bus de campo: Hay un API disponible para la mayoría sistema de bus de campo, en este caso LonWorks.

2.8 Toma de decisiones

Una de las tareas principales de un sistema inteligente es tomar decisiones. Todas las otras partes del sistema son apoyar este proceso. Esto incluye adquisición de los datos, procesamiento, almacenamiento e historia (incluso la ejecución de decisiones).

Se toman todas las decisiones en un nivel alto que exige procesar los datos antes de que ellos entren al proceso de inferencia. La toma de decisiones es un proceso secuencial (procesamiento e inferencia). El procesamiento involucra muchas tareas independientes diferentes que necesitan procesar el rendimiento de la inferencia de alguna manera y comunicarlo a otros agentes.

Hay muchas razones de por qué una decisión no podría ejecutarse después de todo, una de las razones más importantes para no ejecutar una decisión es la prevención de la oscilación, anulación de cambios frecuentes de decisiones para dar énfasis a la naturaleza letárgica del sistema.

En nuestro caso la toma de decisiones se lleva a cabo mediante un control difuso, para tener una idea de lo que es este tipo de control diríjase al Anexo C. En el capítulo posterior daremos una explicación más especifica de cómo implementamos este control.

2.9 Agentes

Esta sección sirve para dos propósitos: nos da una pequeña introducción a la teoría de los sistemas multiagente y describe en detalles cada tipo de agentes de nuestra propuesta.

2.9.1 Introducción a los agentes

El siguiente es un sumario de conceptos de sistemas multiagente (SMA) e inteligencia artificial distribuida (IAD).

Lo primero y más importante es el concepto de un agente. ¿Qué exactamente es un agente? No hay ningún acuerdo todavía en una definición formal de un agente pero la definición siguiente generalmente se acepta:

Un agente es un sistema de computadora que se sitúa en algún ambiente, y que es capaz de ejecutar una acción autónoma en este ambiente para encontrar los objetivos del plan.

Una definición más informal de un agente sería que un agente es un fragmento de software que está continuamente revalorizando la entrada que él recibe de su ambiente, para determinar la salida debe reaccionar (acción) al sistema. Un agente generalmente es no determinista lo cual significa que no acaba en cualquier punto de tiempo.

Un agente, como se definió anteriormente, que sólo está reaccionando a su ambiente no puede llamarse inteligente. ¿Cuáles son los diferencias entre agente y un agente inteligente? Un agente inteligente es un agente que tiene las características siguientes.

- **reactividad**: los agentes inteligentes pueden percibir su ambiente, y responden en una forma oportuna a cambios que ocurren en él para satisfacer sus objetivos del plan.

- **pro actividad**: la exhibición de los agentes inteligentes es dirigida a metas tomando la iniciativa en orden a satisfacer sus objetivos del plan.

- **habilidad social**: los agentes inteligentes son capaces de interactuar con otros agentes (y posiblemente los humanos) para satisfacer sus objetivos del plan.

Un agente no puede ser solamente restringido para ser alguna parte de software ,un agente también puede ser una persona o un elemento de hardware (agente incluido) como lo es el iLon en nuestro caso. Este es sobre todo el caso para los edificios inteligentes donde hay las tres clases de agentes: agentes personas, agentes del software y agentes del hardware.

2.9.2 Arquitecturas de un agente

La arquitectura de un agente describe el interior de un agente (los tipos de algoritmos y estructura de datos que un agente usa). Hay cuatro clases diferentes de arquitecturas para los agentes mencionados en la literatura.

- **agentes basados en la lógica**: la toma de decisiones se comprende a través de la deducción lógica.

- **agentes reactivos**: la toma de decisiones se comprende de alguna manera del entrelazado directo de la situación a la acción.

- **agentes de la creencia-deseo-intención**: la toma de decisiones depende de la manipulación de estructuras de los datos que representan las creencias, deseos e intenciones del agente.

- **arquitecturas por capas**: la toma de decisiones se comprende por varias capas del software, cada uno de los cuales son más o menos explícitamente sobre el ambiente a diferentes niveles de abstracción.

Aquí encontramos una mezcla de las arquitecturas anteriores (vea las secciones siguientes para detalles sobre cada agente). Alguno de estos agentes son reactivos porque ellos reaccionan (cercanos al tiempo real) a la entrada del ambiente. Al mismo tiempo ellos también son basados en la lógica porque ellos toman decisiones por inferencia y ellos son agentes de la creencia, deseo e intención porque ellos están esforzándose para lograr las múltiples metas (deseos). Esto claramente muestra que no pueden asociarse aplicaciones del mundo real a muchas de las definiciones cedidas en la teoría, ellos son una mezcla de todas estas definiciones.

2.10 Sistema Multiagente

Un sistema multiagente es un sistema donde agentes múltiples se están ejecutando (actuando) en el mismo ambiente. Esto implica que ellos consiguen las mismas entradas (por lo menos parcialmente) y que allí las acciones afectan el mismo ambiente. Esto lleva automáticamente a los problemas más complejos, como existen agentes múltiples también en el mismo ambiente la acción de uno afecta la acción de todos los otros agentes en el sistema (no episódico). Esto muestra la necesidad claramente de alguna clase de comunicación entre estos agentes. Hay dos modos básicos de sistemas multiagente: cooperación y competición. En un SMA cooperativo, los agentes para lograr alcanzar una meta común cooperan. Los agentes en semejante sistema pueden cada uno por ejemplo tener una misma función especializada para que ellos puedan juntos lograr una meta que requiere alguna de estas funciones especializadas. Por otro lado, en un SMA competitivo, los agentes compiten entre sí por los recursos limitados como el tiempo del cómputo.

Por tanto inferimos que estamos haciendo un SMA cooperativo. Se especializan los agentes en su función, se mantienen cada uno siguiendo sus propias metas. Para alcanzar estas metas (objetivos del plan) la mayoría de estos agentes cooperan con otros agentes del sistema porque ellos no pueden lograr su meta sin la cooperación.

El ambiente de un edificio inteligente es, antes de las asunciones y simplificaciones, inaccesible, no determinista, no episódico , dinámico y continuo que es el ambiente más complejo de todos. Es inaccesible porque el agente no puede observar el estado completo (no hay sensores para todo), no determinista porque un agente que sigue una acción no puede saber con seguridad la afectación que tendrá en el edificio, no episódico porque las metas diferentes actúan recíprocamente, dinámico porque hay muchos otros procesos en el sistema (edificio) que no pueden controlarse (por ejemplo humanos.) y es continuo porque los eventos pueden pasar en cualquier momento.

2.10.1 HistoryAgent

Varias razones nos llevan a la decisión de que los datos sobre la actividad en el edificio, su ambiente y la actividad de los sistemas deben guardarse en una base de datos.

Estas razones son:

- es un requisito para hacer pruebas estadísticas sobre el comportamiento del sistema.
- tener una base de aprendizaje, donde el sistema pudiera aprender de experiencia.
- la habilidad de trazar mapas sobre la conducta del sistema.

La frecuencia a la que deben probarse tales datos siempre es una decisión delicada para tomar. La prueba demasiado frecuente requiere mucha actuación y espacio del disco para el proceso de prueba. También reduce la velocidad de todos los datos. Si uno va por una frecuencia de prueba baja podrían faltar crestas importantes. Nosotros pensamos en una frecuencia de muestreo de 1 por minuto. Todavía lo mejor sería hacer esta frecuencia de prueba configurable.

También debemos tener cuidado especial para ser eficientes con los recursos de la CPU. Imagine si un edificio entero con centenares de cuartos fuera controlado y observado por este sistema. La base de datos tendría que guardar miles de cambios por minuto

HistoryAgent sólo guarda valores de propiedades del bus que realmente son supervisadas por algunos agentes en el sistema. El sistema del bus de un edificio ordinario tiene miles y no centenares de variables diferentes, así que la base de datos se bloquearía después de un tiempo corto. Para evitar esto, el *BusAgent* debe guardar todas las actualizaciones inconstantes pendientes hasta que *HistoryAgen*t las saca.

2.10.2 ControlAgent

Responsabilidad

El *ControlAgent* es el representante del concepto de habitación del sistema. Cada caso de este agente cubre una habitación específica. Es responsabilidad del *ControlAgen*t analizar las señales de todos los sensores que pertenecen a la habitación para hacer inferencias de esta información. También es responsable de convertir estas decisiones en acciones. Un caso de un agente control se crea automáticamente para cada cuarto descubierto por el *BossAgent*. Después de comenzar una instancia de un *ControlAgent* se asigna a una habitación particular para su vida entera.

El proceso de toma de decisiones como se describe en detalle en capítulo III se lleva a cabo en el *ControlAgent*. Las decisiones son tomadas por un mando de la lógica *fuzzy* (FIS) como se muestra en Figura 2.5. Las decisiones se revisan como nuevos datos y se pone disponible, eso podría influir en el resultado del FIS.

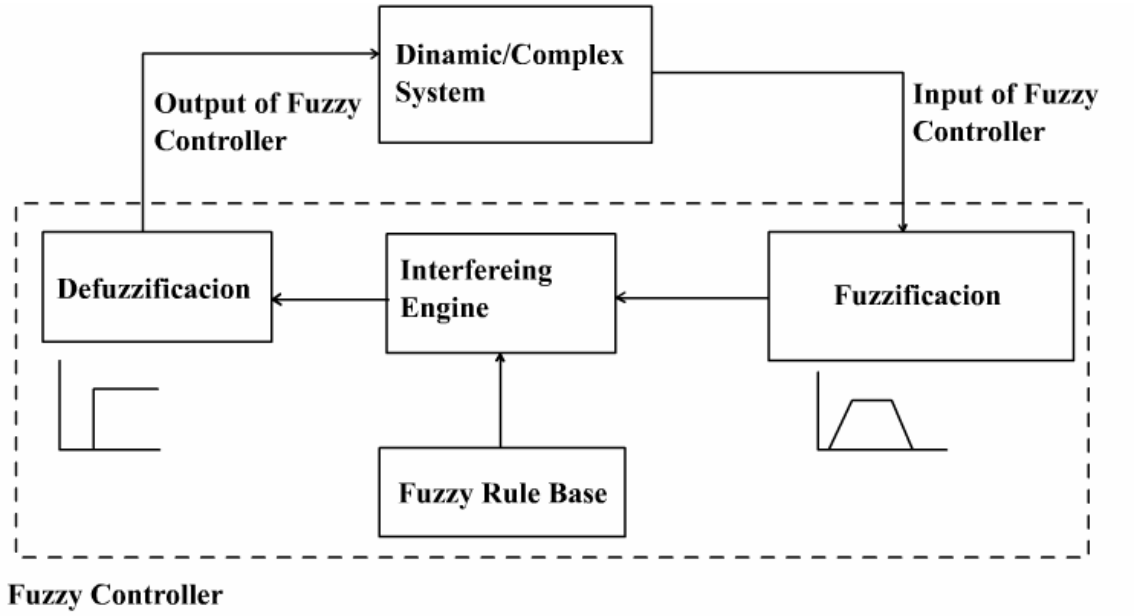

Figura 2.5 La apreciación global de la toma de decisiones, modelada como un controlador fuzzy

Los datos que son usados por el FIS son preprocesados primeramente de forma que sólo datos preparados evalúa el FIS.

2.10.3 BusAgent

El *BusAgent* es el agente de nivel más bajo (arquitectura por capas) en el sistema. Actúa como una entrada entre el SMA y el sistema de bus de campo real.

Responsabilidad

El *BusAgent* proporciona una manera genérica de acceder al control del bus de un edificio. El *BusAgent* lo hace para instanciar una aplicación concreta de la capa de abstracción del bus. El *BusAgent* actúa como un apoderado entre los otros agentes y la propia aplicación de abstracción de bus. Mantiene una lista de agentes interesados para cada variable de red

disponible. Todos estos agentes reciben una notificación del *BusAgent* cuando hay un nuevo valor disponible para una variable de red.

El *BusAgent* proporciona la capacidad además para explorar las variables de la red disponibles en el edificio. Para esto proporciona las maneras de buscar variables específicas con modelos de la búsqueda expresadas como expresiones regulares. *HistoryAgent* debe confiar en *BusAgent* que le proporciona las actualizaciones para todas las variables subscritas. Hace esto localmente guardando todas las actualizaciones conducidas hasta que *HistoryAgent* las saca.

Colaboración entre agentes

El *BusAgent* es el agente más básico. Casi cada uno de los otros agentes usa *BusAgent*. Sobre todo el *HistoryAgent* confía en los servicios que el *BusAgent* proporciona. *BusAgent* no está escuchando ningún mensaje de otros agentes

El siguiente mensaje es enviado por el *BusAgent*: la propiedad de actualización, que se envía a los oyentes para una cierta propiedad de la red

2.10.4 BossAgent

Este agente hace el sistema capaz de automáticamente reaccionar a los cambios en el sistema (configuración dinámica). Este proporciona la funcionalidad de que al sistema automáticamente se agreguen agentes instanciados tales como nuevas habitaciones (conectadas) a la red.

Responsabilidad

BossAgent es el "Jefe" de todos los *ControlAgents*. Conecta al *BusAgent* que tiene que haber empezado antes. *BossAgent* crea un *ControlAgent* entonces para cada habitación que descubre en la red del edificio. El descubrimiento de las habitaciones en la red se hace comparando los nombres de propiedad con un modelo proporcionado en fichero de configuración .

Colaboración entre agentes

BossAgent y los agentes *BusAgent* están relacionados, estos preguntan al *BusAgent* por todas las variables de la red disponibles e intentan extraer nombres del cuarto fuera de los nombres de variables. Empieza (instancia) el número requerido de agentes para cada uno de los cuartos descubiertos mientras investiga la red .

2.10.5 RoomDisplayAgent

Responsabilidad

El *RoomDisplayAgent* es el único agente en el sistema con una interfaz de usuario gráfica. Se diseña para ser usado por usuarios especialistas del sistema para supervisar el sistema o para actuar recíprocamente manualmente con él mediante órdenes del usuario final. El *RoomDisplayAgent* se enfoca en la perspectiva del cuarto (siempre visualiza las variables dependientes de la habitación que pertenecen al cuarto actualmente seleccionado). Él adicionalmente despliega variables globales que están una vez disponible por edificio (las tantas variables de tiempo como la radiación a diferentes lugares, humedad, temperatura).

Este agente no necesariamente tiene que correr, el sistema es funcional sin él. Puede empezarse un arbitrario número de veces, incluso para el mismo cuarto. Se registra como oyente para las decisiones para el cuarto actualmente seleccionado y adicionalmente como oyente para actualizar el valor de propiedad para todas las propiedades disponible para el cuarto seleccionado. En algunos es usado para interactuar de forma directa con el sistema, por ejemplo mediante el mando de forma manual.

Como decíamos los actores siguientes de un cuarto pueden ser controlados manualmente por el *RoomDisplayAgent*: persianas (subir, bajar, detener, ángulo), luz y climatización. Con mucho más funcionalidad estos actores pueden ser controlados, mediante el *RoomDisplayAgent* es posible interactuar directamente con los dispositivos físicos del cuarto. Él además visualiza los datos siguientes: temperatura exterior, radiación exterior (de todos los lados del edificio), humedad exterior, iluminación exterior / luz, el estado de las persianas y estado de las luces.

Colaboración entre agentes

Los siguientes mensajes son enviados por el **RoomDisplayAgent**:

- Asigne un nuevo valor a la propiedad (a *BusAgent*)

El propio *RoomDisplayAgent* se registra como interesado en mensajes de los tipos siguientes:

- Decisiones (enviadas a afuera por el *ControlAgent*)
- La propiedad de actualización (para todas las propiedades que pertenecen al cuarto actualmente seleccionado)
- La propiedad global de actualización (para todas las propiedades globales que pertenecen al edificio en que el actualmente el cuarto seleccionado se localiza físicamente)

2.11 Manejo del detector de presencia

Presencia es la variable que resume un detector de presencia. La salida de un detector de presencia es muy simple (sí o no). Pero decidir si alguien está presente o no es un proceso muy complejo.

El detector de presencia entrega una señal discontinua cuyas crestas siempre tienen que ver con una cierta cantidad de movimiento en el cuarto. Pero que no haya ningún movimiento en el cuarto no significa por supuesto que no hay alguien presente. La consecuencia de esta conducta es que allí tiene que haber un proceso de *smoothening* (para reducir la frecuencia de la señal) que constantemente revaloriza la entrada del detector de presencia en combinación con valores del pasado y tiempo para decidir si hay alguien presente o no.

Resulta que la salida del detector de presencia no es conveniente como una entrada directa a la toma de decisiones. El detector de presencia envía varios *spikes* al sistema según el movimiento en el cuarto. De estos *spikes* el valor real (*yes*/*no* de la presencia, mucho movimiento, movimiento pequeño) necesita ser computado antes de que sea enviado a la toma de decisiones.

2.12 Algunos análisis

Puesto que el *HistoryAgent* anota todos los datos importantes, nosotros podemos hacer algún análisis de las señales de entrada del sensor y el comportamiento del *ControlAgent*. Éstos análisis nos permiten mejorar las reglas *fuzzy* para el *ControlAgent*.

El *HistoryAgent* anota los datos en un formato perfeccionado para la inserción en su base de datos. Esto previene el análisis directo de datos, por tanto necesita ser preprocesado de alguna manera. Nosotros pensamos que usar Matlab es lo ideal para el procesamiento de los datos. Las razones de por qué Matlab son:

- permite acceso directo a la base de datos.
- un standard de la industria.
- implementa (NTI).
- muchos toolboxes disponibles qué podrían usarse en trabajo del futuro.

Los datos en la base de datos no se muestrean a intervalos de tiempo discreto. Así si los datos de algunos sensores tienen que ser analizados ellos tienen que ser explorados primero, tal que haya la misma cantidad de datos de la muestra por cada sensor. Esta exploración no necesita mucho esfuerzo porque los datos en la base de datos ya son normalizados. Estos datos perdidos no se interpolan porque el análisis debe hacerse en los datos de un agente realmente registrado.

El enfoque para la inserción eficiente de datos para *HistoryAgent* es que algunos sensores tienen valores string guardados en la base de datos. Estos valores tienen que ser convertidos a valores numéricos durante el procesamiento de los datos.

2.13 Procesamiento de las salidas

Hay que ciertos estados del sistema en el que tiende a empezar a oscilar entre a los estados diferentes de una variable de salida. Esto pasa cuando el sistema está cambiando un valor de salida o que se remite directamente a un actor como un interruptor ligero. Si la acción de este actor como resultado afecta las entradas al sistema este puede alcanzar un estado

donde decide encender de nuevo la luz. Así empieza en un ciclo interminable de rápido cambio de los valores *input*/*output* del sistema.

Para prevenir esto, el proceso adicional tiene lugar antes de tomar una decisión, realmente se remite a los actores (ejecución). Este proceso constantemente evalúa los valores de la salida pasados, actualiza la frecuencia y las características especiales de cada variable para asegurarse que el sistema no puede alcanzar un estado donde empieza a oscilar. Este método para prevenir al sistema de ponerse inestable ha demostrado ser muy exitoso y no computacionalmente intensivo al mismo tiempo.

2.14 Comunicación entre agentes en general

El intercambio de mensajes entre agentes no proporciona dificultades particulares. Todos los agentes en el sistema deben estar de acuerdo en el volumen de cada mensaje con el que ellos tienen que repartir en la vida. Estas definiciones, particularmente durante el desarrollo, tienden a ser muy volátiles. Con definiciones fijas y codificadas cada cambio en la sintaxis del mensaje o la semántica de este requiere un despliegue completo de todos los agentes.

2.15 Conclusiones

La meta final de lo desarrollado en este capítulo era establecer un enfoque abstracto de lo que consiste un sistema multiagente. Nosotros establecimos una arquitectura del sistema básica, a agentes que proporcionan servicios básicos como anotar, controlar, abstracción del bus y la visualización dinámica y una estructura de agentes que toma decisiones sobre la base de reglas de lógica *fuzzy*.

Por ahora nos limitaremos a implementar este sistema multiagente en los lenguajes de programación Matlab y Delphi, basados en toda esta disertación nos encontramos en la fase de darle vida a algunos de estos agentes mencionados anteriormente con la finalidad de simular el sistema de una manera mucho más dinámica e interactiva, en este se hará uso del sistema de inferencia *fuzzy* implementado en el Matlab y otras herramientas que describiremos más adelante.

Capítulo lll El ControlAgent, HistoryAgent y el RoomDisplayAgent implementados físicamente.

En este capítulo pudiéramos decir que vamos a aterrizar después de una descripción abstracta de lo constituye un sistema multiagente, en este vamos a explicar como implementamos en la práctica el *ControlAgent, HistoryAgent* y el *RoomDisplayAgent*.

3.1 ControlAgent

El *ControlAgent* como decíamos en el capítulo anterior es el agente de control y encargado de la toma de decisiones del sistema domótico, el objetivo principal de la tesis es aplicar la inteligencia artificial en la domótica y este es en sí el agente que cumple con esta función, este es un controlador difuso implementado en el lenguaje de programación Matlab. Para su construcción utilizamos el *Fuzzy Logic Toolbox* que cuenta con una herramienta visual muy útil para la implementación de este tipo de controladores llamada FIS (*Fuzzy Inference System*) Editor. Esta herramienta es un editor que permite establecer las entradas y salidas al sistema y determinar todo un conjunto de reglas difusas, además de todo el proceso de fusificación y defusificación. Los ficheros que este crea son controladores difusos que realizan todas sus funciones y pueden ser utilizados de forma dinámica en el Simulink y también pueden ser llamados desde la ventana de comandos de Matlab, estos ficheros tienen extensión (*.fis).

En este fis de tipo mandami que llamamos BD declaramos seis entradas con varios estados cada una, las cuales enumeraremos a continuación:

1. *Radiation*
   - *A-CompletlyDark, B-LittleLigth, C-Ligth, D-Sun, E-LotsOfSun*
2. *Temperature*
   - *A-VeryCold, B-Cold, C-Warm, D-VeryWarm, E-Hot*
3. *Ilumination*
   - *B-LittleLigth, A-CompletlyDark, C-Ligth, D-Sun, E-LotsOfSun*

4. *LigthStatus*
    - *On, Off*
5. *BlindStatus*
    - *Open, AlmostOpen, AlmostClose, Close*
6. *Presence*
    - *Yes, No, Doubt*

Además definimos tres salidas a los controladores de dispositivos con sus respectivos estados:

1. *LigthController*
    - *On, Off, NoChange*
2. *BlindController*
    - *Open, Close, NoChange*
3. *Climatization*
    - *Cold, Fan, Off, NoChange*

Los rangos de variación de estas variables son elegidos al azar, estos son valores que dependen de los tipos de sensores que tengamos en nuestro sistema, con los cuales no contamos en este momento. De esto se encargará la interfaz física que no se realizará en este proyecto, la cual suponemos que debe actualizar la base de datos con los valores de las señales provenientes de los sensores, que serán los elementos de medición reales que estarán directamente en el campo de trabajo y con cuya estructura física debe relacionarse la interfaz externa que se construya para la terminación de este proyecto. Como ya sabemos solo vamos a implementar tres de los agentes del sistema multiagente. Recordemos que en el capítulo anterior realizamos la propuesta de un bus de campo, para la comunicación entre los elementos de campo que fue el protocolo LonWorks, una tecnología muy difundida

principalmente en Europa que cuenta con elementos de hardware como el **iLon Gateway** y de Software como el LNS, el primero utilizado para el aislamiento de las dos redes (Ethernet y LonWorks) y el segundo para el manejo de todas estas variables de red en la red LonWorks. A cada una de estas variables se le definieron las *membership functions* en su mayoría sigmoidales y triangulares suficientes para describir el comportamiento de estos sistemas.

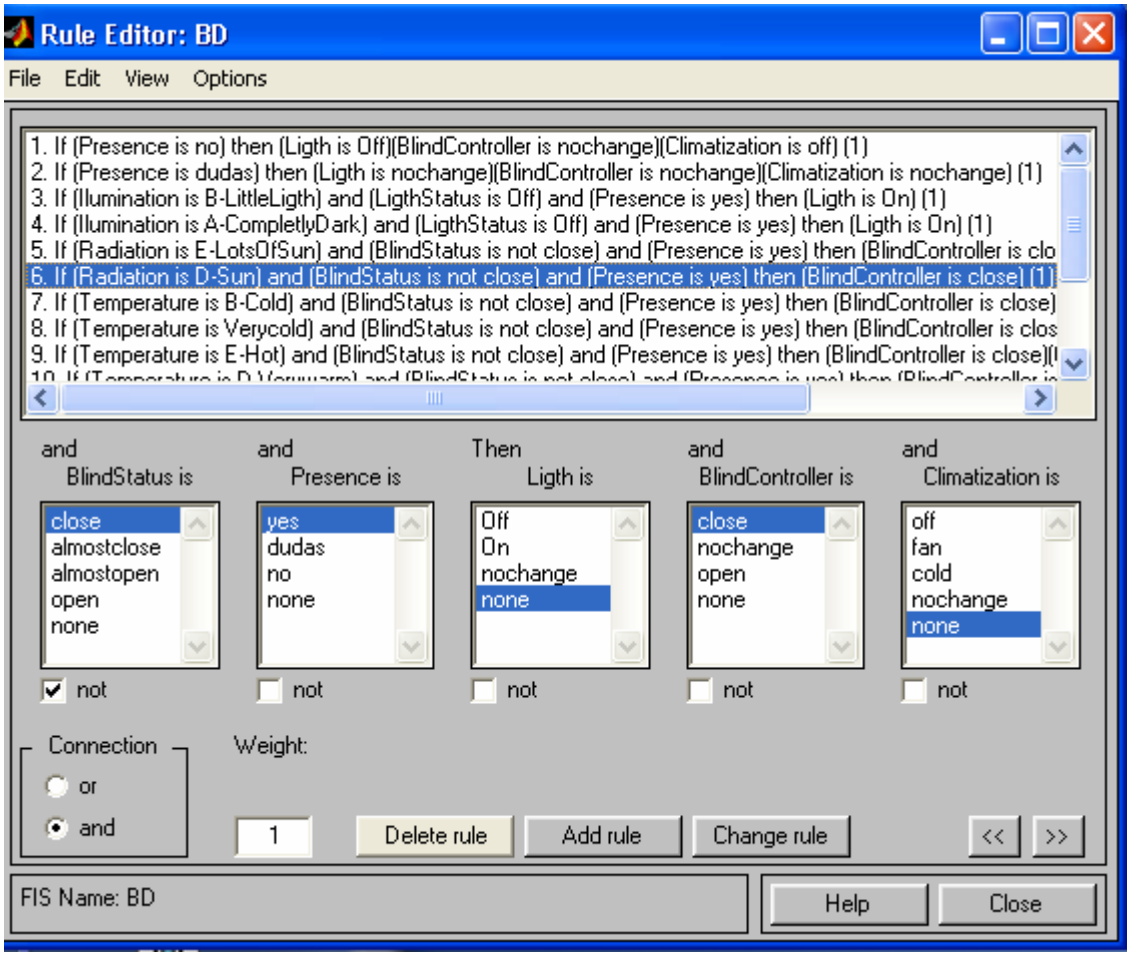

Figura 3.1 Rule Editor

En la figura 3.1 podemos observar el *Rule Editor* así como algunas de las reglas de todo el bloque que implementamos por pura lógica de la realidad y según nuestras preferencias personales. En esta figura podemos ver algunas las entradas y salidas que definimos anteriormente y parte del bloque de reglas por las que se rige este controlador difuso. Esta es una aplicación para utilizar en tiempo de diseño, es decir una vez ejecutada la puesta en

marcha de nuestro *ControlAgent* ya no podemos usarla, pero sí muchas de las propias funciones del *Fuzzy Logic Toolbox* que son las mismas que esta llama.

Mientras nos encontrábamos en la etapa de diseño pensamos en cómo podríamos solucionar el problema planteado anteriormente, lo cual hacía nuestra aplicación muy estática, mas si analizamos que para ejecutarla tendríamos que disponer de la ventana de comandos de Matlab. Además necesitábamos que el controlador difuso muestreara las entradas de forma periódica para que así pudiera quedarse funcionando dinámicamente durante tiempo indefinido y verificara todas las actualizaciones en la base de datos cada vez que cambiaran los valores de los sensores. Para esto entonces utilizamos el *Runtime Server Toolbox* el cual describiremos más adelante.

Parte del *ControlAgent* son también los ficheros *InOut*.m y *addr*.m cuya función es establecer la conexión con las tablas de la base de datos db1.mdb, la cual diríamos que es el *HistoryAgent* y añadir en caso de ser necesario reglas al controlador.

3.2 HistoryAgent

El *HistoryAgent* quizás es el más sencillo de todos lo agentes, pero no por esto podemos decir que su función no es importante, este consta de una base de datos llamada db1.mdb que tiene varias tablas. Por ejemplo para cada habitación tiene una tabla con todas las entradas provenientes de los sensores que son evaluadas por el controlador difuso, más tarde este exporta a otra tabla que contiene los valores de salida hacia los controladores de dispositivos. En este caso realizamos la implementación para una sola habitación pero no resultaría difícil realizarlo para un edificio completo, de esto se encargaría el *BossAgent* que es el encargado de crear instancias de los agentes en un sistema multiagente (este agente no está implementado en este proyecto). Los valores de la tabla de entradas son aleatorios, realmente deben provenir de la interfaz física que se realice para hacer las lecturas de los sensores que instalemos en nuestro sistema. Esta base de datos está confeccionada en Microsoft Access, programa que tiene una magnífica comunicación con Matlab y para el cual este lenguaje tiene una herramienta que es *Database Toolbox* que provee funciones con este fin.

Para la comunicación entre el *HistoryAgent* y *ControlAgent* creamos el fichero *InOut*.m que es quien establece la conexión y realiza las consultas SQL (*Structured Query Language*) a la tabla de entrada, mediante estas consultas el *ControlAgent* conoce los valores de entrada, evalúa el controlador difuso y modifica la tabla de salida. Para establecer la conexión debemos definir el *timeout* para en caso de que esta falle el sistema no se quede colgado, para ello utilizamos la función **logintimeout** (*param*) donde *param* es el número de veces que el sistema intentará establecer la conexión. También en el fichero *InOut*.m se especifica el origen de datos que en nuestro caso se llama domótica y cuya función es definir el directorio donde se encuentra la base de datos para realizarle la encuesta posteriormente, por supuesto que este es un alias que debe instalarse anteriormente. También aquí se realizan las consultas SQL, Matlab nos da posibilidad de utilizar este lenguaje con todas sus potencialidades y de este nos auxiliamos para conocer los registros de entrada de la tabla *InOut*, esta información se guarda en una variable en forma de matriz y puede ser utilizada para varias finalidades, por ejemplo para hacer un análisis de cómo se esta comportando el sistema o para evaluar el controlador difuso llamando al BD.fis. Luego de realizar esta operación el controlador nos devuelve los valores de salida correspondiente al juego de datos de entrada y estos son insertados en la tabla de salida *OutPut*.

En la base de datos también hay un tabla llamada *Users* con los datos fundamentales de todos los huéspedes que están en la habitación seleccionada, esta se actualiza de forma manual mediante un registro de carpeta, en la interfaz existe una posibilidad de llenarla aleatoriamente para facilitar el trabajo en la fase de prueba de la aplicación.

3.3 Runtime Server

Las variables comunes de sistemas domóticos cambian cada determinado período, por tanto para mantener nuestro sistema actualizado necesitamos una aplicación que se mantenga alerta en todo momento, realizando lecturas de las variables de entrada y por supuesto actualizando las salidas hacia los controladores de dispositivos. Este era el problema del cual habíamos hablado anteriormente, como habíamos dicho el fichero BD.fis es un fichero estático, es decir solo se ejecuta cuando lo mandamos y por tanto para ello necesitamos otra aplicación que sea la que se encargue de esta función, algo así como un servidor.

Aquí es donde entra a jugar su papel el *Runtime Server*, para su construcción y el logro de este objetivo trabajamos con el *Runtime Server Toolbox*, este es un toolbox con sus características peculiares pues debemos configurarlo, instalarlo y renombrar el *license*.dat de Matlab a *license*.tmp, este toolbox se sale de la aplicación convencional de Matlab y lo convierte en una aplicación servidora.

El *Matlab Runtime Server* es una variante de Matlab que los desarrolladores de Software pueden unir con aplicaciones que usan Matlab. El *Matlab Runtime Server* mantiene todas las capacidades computacionales y gráficas de Matlab convencional pero está diseñado para aplicaciones *standard-alone* basadas en Matlab. Los usuarios finales de este tipo de aplicaciones no necesitan tener conocimiento alguno sobre Matlab y ni siquiera pueden acceder al código fuente. Cuando digo usuarios finales me refiero a los que vayan a hacer uso del servidor incluyendo los programadores que desarrollen la interfaz de usuario que no tiene que ser necesariamente en Matlab, por ejemplo en nuestro caso la desarrollamos en Delphi. También especificamos que la ventana de comandos de Matlab no esta disponible durante la ejecución del servidor, lo cual debemos tener en cuenta a la hora de construirlo, porque este solo recibe la información necesaria en forma de parámetros desde la aplicación que lo cargue.

3.3.1 La aplicación RtSetup.exe

Antes de usar el servidor debemos instalarlo y ponerle un *password* (*stamp*) que nosotros elegimos. Para esto debemos realizar los siguientes pasos:

1. Ir a la ventana de comandos del sistema
2. Si *matlabroot* es el directorio donde instalamos Matlab entonces navegar hasta:
   - matlabroot\toolbox\runtime\bin\win32 (PC)
   - matlabroot/toolbox/runtime (UNIX)
3. En la ventana del sistema debemos escribir
   - rtsetup -f matlabroot\bin\win32\matlab.exe -s string (PC)

- rtsetup (UNIX)

Donde *string* es el el *password* que seleccionamos. La utilidad *rtsetup* pone sello a la copia de Matlab con este *password* y este procedimiento se puede hacer una sola vez. No se puede *restamp* una copia de Matlab.

3.3.2 Los ficheros matlabrt.m y pathdefrt.m

Además debemos crear dos ficheros que son el matlabrt.m y el pathdefrt.m que son *startup files* de la aplicación servidora y debemos ponerlos en el siguiente directorio matlabroot\toolbox\runtime\local, estos ficheros son los que llaman al Interruptor.m y definen todos los directorios de Matlab donde hay aplicaciones usadas por nuestro sistema. Cuando el servidor se ejecuta por primera vez llama a estos ficheros que son versiones de los ficheros matlabrc y pathdef que la versión comercial de Matlab invoca cuando comienza.

Propiedades de la función matlabrt.m

- Debe residir en el directorio toolbox\local
- Desempeña las tareas ordinarias de matlabrc, tales como llamar la función de definición de caminos (pathdefrt).
- Carga el resto de la aplicación porque el matlart.p es el único fichero que el *Matlab Server* llama directamente, por ejemplo en nuestra aplicación el fichero matlabrt.m contiene la línea que contiene el comando Interruptor que llama a la aplicación *top level* que es el fichero Interruptor.m.
- También podemos definir el comportamiento de tareas de forma opcional, como el comportamiento global ante diferentes tipos de errores, en nuestro caso lo definimos con la opción por defecto que es *dialog* la cual le pregunta al usuario si desea continuar ignorando el error o salir de la aplicación.

Propiedades de la función pathdefrt.m

La utilidad pathdefrt es esencial para la aplicación *Matlab Runtime Server*, es una variación de la función pathdef que usa la versión comercial de Matlab, esta debe residir en el directorio toolbox\local y guarda la información de todos los directorios. Esta función es llamada por la función matlabrt.m cuando el servidor comienza a funcionar. Todos los ficheros que el *Runtime Server* utiliza deben estar en estos directorios que pathdefrt define para que este los encuentre. Este trayecto de búsqueda solo debe incluir directorios por debajo del directorio *toolbox*.

Para encontrar todos estos directorios debemos ejecutar en la ventana de comandos de Matlab las funciones **depfun** o **depdir**, si una de estas funciones indican que alguna aplicación usa algún fichero de directorios privados o de clases no podemos incluirlo completamente en nuestro trayecto de búsqueda, sin embargo el directorio anterior si debemos incluirlo, por ejemplo si la función **depdir** lista:

C:\matlab\toolbox\matlab\funfun\@inline

Entonces el trayecto de búsqueda debe incluir el directorio toolbox\matlab\funfun

Para crear el fichero pathdefrt debemos seguir los siguientes pasos:

1. Copiar el fichero pathdefrt_template.m que está en el directorio toolbox\Runtime para toolbox\local.

2. Usar la función **depdir** como explicábamos anteriormente.

   list = **depdir**('matlabrt');

3. Omitir los directorios privados de list.

4. Pegar el contenido de list en el fichero pathdefrt que estamos construyendo y editar las líneas según sea necesario para ajustarnos al formato, por ejemplo los elementos deben tener esta forma.

   '$toolbox/matlab/general:'

y no algo como 'C:\Apps\matlab\toolbox\matlab\general'.

3.3.3 La compilación

Debemos tener presente que para el funcionamiento del servidor es necesario convertir todos los ficheros (*.m) en (*.p) mediante la función **buildp**, este es un proceso de compilación, los ficheros punto p son algo parecido a ejecutables. La función **buildp** determina los ficheros que debe compilar y los compila, por tanto el programador no debe compilar los ficheros uno por uno sino que se compila el fichero matlabrt y así la función sabe cuales son los otros ficheros de la aplicación y los convierte en punto p. Antes de compilar los ficheros y buscar dependencia debemos ejecutar la función **cleanp** para eliminar del camino todos los ficheros con extensión .p. Aunque si no se hace, la función **buildp** los sobrescribe, pero no es una práctica recomendable, por lo menos a nosotros nos fue mejor usando la función **cleanp** (es más seguro y no cuesta ningún trabajo).

Este enfoque es una herramienta muy poderosa porque utiliza a Matlab como un *engine aplication*, es decir explota de este su mejor característica que el procesamiento de datos, más cuando se trata de datos de sistemas de control.

3.3.4 Confirmación

Cuando el servidor está funcionando correctamente observamos un icono de Matlab en la barra de tareas de *Windows* con la etiqueta (MATLAB SERVER), esto nos indica que nuestro servidor está listo para ser llamado a realizar alguna función de los ficheros punto p.

¿Cómo llamar el servidor? El servidor es para ser ejecutado como un objeto ActiveX u OLE *Automation* desde cualquier lenguaje de programación orientado a objetos o una GUI (*Grafics User Interface*) del propio Matlab. En nuestro caso elegimos una GUI implementada en Delphi sobre la cual hablaremos posteriormente.

3.3.5 Matlab y ActiveX

ActiveX es un conjunto tecnologías y herramientas orientadas a objetos que permiten a los desarrolladores de Sofware integrar componentes específicos de aplicaciones de diferentes fabricantes dentro de su propia solución. La automatización con empleo de ActiveX le

permite a Matlab controlar y ser controlado por otros componentes ActiveX. ActiveX es un protocolo de *Microsoft Windows* para integración de componentes. Usando ActiveX los desarrolladores y usuarios finales pueden seleccionar aplicaciones específicas, por ejemplo una aplicación puede requerir acceso a bases de datos, análisis matemáticos y presentación de información. Usando ActiveX el programador puede elegir un componente de acceso a datos de un vendedor, un componente de creación de interfaces visuales de otro fabricante e integrarlos todos en un componente de análisis matemático hecho por un tercero.

Las tecnologías de ActiveX tienen una raíz común llamada *Component Object Model*, o COM. Cada lenguaje orientado a objetos tiene un modelo de objetos que definen ciertas características en ese medio tales como la localización de esos objetos, sus instancias y su identificación.

Cada objeto ActiveX soporta una o varias interfaces, una interfaz está relacionada con una colección de métodos, propiedades y eventos. Los métodos son llamadas a funciones que a su vez son solicitudes a objetos para que realicen alguna función. Las propiedades son variables de estado que mantiene el objeto, tales como el color de un texto o el nombre de un fichero. Los eventos son notificaciones que un controlador envía a su cliente.

Matlab soporta dos tecnologías ActiveX: contención de controles y automatización ActiveX, esta última es la que nos interesa para nuestro estudio y es la que permite que Matlab sea controlado por otros componentes ActiveX, cuando Matlab es controlado por otro componente decimos que actúa como un servidor de automatización. Dentro de las capacidades del servidor de automatización del Matlab se incluyen la habilidad de ejecutar comandos en el *Matlab workspace*, y tomar o poner matrices en este.

3.4 RoomAgentDisplay

Por toda esta explicación en la sección anterior ya conocemos que Matlab puede ser cargado por otra aplicación y controlado por esta que sería un controlador de automatización, con este objetivo creamos la interfaz de usuario para el controlador difuso que es el *RoomAgentDisplay*. La interfaz de automatización ActiveX de Matlab soporta múltiples métodos, por ejemplo en nuestro caso cuando lo cargamos desde la aplicación de Delphi (lenguaje en el cual se realizó esta interfaz) primero creamos un objeto de tipo

Variant llamado Matlab, el nombre no es importante, pudiera ser cualquiera, lo que realmente interesa es el tipo de variable que también pudo haber sido una *Idispatch*, luego creamos un objeto ole mediante el uso de la función:

Matlab := **CreateOleObject** (‘Matlab.application6’);

A esta función se le pasa como parámetro el ProID del Matlab que es *Matlab.application6*, el ProID es un identificador único para cada programa. Este se reconoce porque el objeto ActiveX de Matlab se localiza así en el registro de *Windows*, exactamente cómo se invoque el servidor depende de qué controlador estemos usando pero lo que sí el controlador requiere es este identificador para identificar el servidor.

Dentro de los métodos que invoca el servidor está el método *execute*, este toma como argumento una *command string* y devuelve una *string* también como resultado. La *command string* podría ser cualquier comando que normalmente escribiríamos en la ventana de comandos de Matlab y el resultado lo que normalmente pudiera retornar en la ventana de comandos. Mediante el método *execute* llamamos a los ficheros de Matlab.

La forma en que se presenta esta interfaz es como una ventana habitual de *Windows*, en esta podemos seleccionar la base de datos a la cual deseamos acceder. Dentro de esta base de datos elegimos una tabla específica que también podemos seleccionar en un **listbox,** que visualiza todas las tablas existentes en la base de datos. El contenido de la tabla marcada en el **listbox** se puede ver en un **DBGrid** y mediante un **DBNavigator** podemos navegar a través de la tabla y realizar algunas opciones de edición. En la ventana principal encontramos el botón **Select Fields**, que visualiza otra ventana donde podemos seleccionar los campos de tabla que nos interese visualizar en un momento determinado.

También tenemos el botón **Users** cuya función es visualizar la tabla con los datos de los usuarios que estén en registrados en la habitación. Pudiéramos decir que el botón más importante en la ventana principal es el botón **Start Server** que como su nombre lo indica lo que hace es llamar al servidor por primera vez, una vez ejecutado el botón **Evaluate Controller** si damos clic en el cuadro de confirmación **Periodic**, el servidor se mantendrá realizando su función de forma periódica.

Cuando presionamos el botón **Simulation** estamos llamando un método que genera aleatoriamente valores de entrada en la tabla *Input*s, de esta manera simulamos de forma desordenada un medio determinado, cabe aclarar que estos valores pudieran variar en rangos ilógicos pero esto no es algo de vital importancia en el programa pues en la realidad esto no sucede así.

Si pulsamos el botón **Remote** visualizamos unas de las aplicaciones más interesantes de la interfaz, en esta ventana podemos incidir manualmente sobre las variables de salida de la tabla *Outputs*. Por supuesto que una decisión como esta podría traer consigo una contradicción entre el usuario y las reglas por las que se rige el controlador difuso, por lo que si este lo desea pudiera convertir su decisión en una preferencia si selecciona **Apply Preferences**. La aplicación es capaz de registrar las entradas que en ese momento existían y de esa forma añadir una regla más al controlador difuso, ejecutando el fichero addr.p de Matlab.

3.5 Conclusiones

- Pensamos que para el desarrollo de este tipo de sistemas es necesario basarnos en un conjunto de utilidades, que logren en su totalidad la integración de todas las funcionalidades que ellos requieren, para lo cual es necesario un riguroso estudio de tales herramientas y así poder lograr la complementación de la aplicación.

- Reafirmamos que el uso de la inteligencia artificial en sistemas de alta complejidad como lo son los sistemas domóticos es una técnica muy efectiva, que simplifica y soluciona problemas que por otros métodos serian extremadamente engorrosos.

Conclusiones

A través de este trabajo hemos llegado a las siguientes conclusiones:

- Del estudio realizado se constató que un sistema domótico es flexible, versátil y adaptable a cualquier necesidad, a cualquier tipo de edificio y a cualquier actividad que en él se vaya a desarrollar; lo que proporciona un sinfín de beneficios y ventajas inalcanzables mediante una instalación tradicional.

- Existe una lucha por la estandarización de los medios y protocolos, lo cual implica el conocimiento de una serie de conceptos técnicos con el objetivo de realizar una buena selección de la tecnología.

- El enfoque multiagente permite modularizar las funciones de un sistema y la utilización de la inteligencia artificial, por lo que se convierte en una vía muy utilizada en el mundo de la domótica.

- La inteligencia artificial admite el manejo de múltiples variables así como la realización de operaciones de complejidad superior, manifestándose como una exitosa alternativa para el control de sistemas domóticos

- Matlab es una herramienta de programación poderosa para el desarrollo de controladores difusos, que permitan el procesamiento de señales y la manipulación de datos en edificios inteligentes.

Recomendaciones

Luego de finalizar esta tesis se quisiera recomendar:

- Desarrollar un estudio más intenso sobre la propuesta específica del hardware (LonWorks) para la implementación de los elementos de campo, tales como los sensores y los actuadores, así como del driver (iLon) para la comunicación con estos dispositivos.

- Realizar los ajustes necesarios para actualizar esta aplicación según las características de la propuesta anterior, por ejemplo los rangos de las variables deben coincidir con los valores de la realidad.

- Incluir dentro de las asignaturas de control los temas de la domótica y la Inteligencia Artificial, dado el grado de desarrollo, aplicación e impacto que tienen ambas esferas en el ámbito de los sistemas de control actual y con el objetivo fundamental de preparar a los futuros ingenieros en estas disciplinas que prometen grandes avances para el futuro de la humanidad.

- Debemos pensar en el desarrollo de proyectos para la automatización de algunas instalaciones turísticas en nuestro país, no solo porque reviste un notable ahorro energético, sino también por el desarrollo alcanzado por esta rama a nivel mundial, lo cual trae consigo la afluencia de un turismo más exigente en este sentido.

Bibliografía

[1] Abaires. “Conecte su hogar al futuro”

http://www.abaires.comDomotica1.htm. Marzo, 28 ,2001

[2] Ae-t Ingeniería. “Instalaciones de domótica”

http://www.ae-t.com/domotica.htm. Marzo, 27, 2001

[3] Ayuda del Delphi

[4] Ayuda del Matlab

[5] Calvo Soteló. “Camba oferta para la domótica”

http://www.cambas.com. Marzo, 13, 2001

[6] Catálogo

http://www.echelon.com/Catalogos. Mayo, 4, 2001

[7] Catalogo. Sistemas de automatización del hogar.

http://www.superinventos.com. Marzo, 17, 2001

[8] Domodesk. “Todo en domótica”

http://www.Domodesk.com/Productos. Abril, 2, 2001

[9] “Domótica”

http://www.geocities.com/NapaValley/4376/Domotica.htm. Abril, 12, 2001

[10] Domoval. “Productos”

http://www.domoval.com. Marzo, 18,2001

[11] Expósito Diez, José Javier. "¿Qué son los sistemas de domótica?"

http://www.imeyca.com/Domotica.htm. Marzo, 22, 2001

[12] Gemma. "Una perspectiva generalista de LonWorks"

http://www.domotica.net/news/pg00306.htm. Marzo, 22, 2001

[13] Giga. Revista cubana de computación. Colombus conectividad. N° 1, 1998.p29-35.

[14] Giga. Revista cubana de computación. Colombus conectividad. N° 4, 2000.p29-35

[15] http://www.sainel.es/menuhot.htm

[16] "Ingeniería de Sistemas Domóticos y Electrónicos"

http://www.Ide-ing.com. Marzo, 29, 2001

[17] Inmoclick. "Podemos controlar nuestro hogar desde un ordenador"

http://www.inmoclick.com/b2c/pages/estatic/domoindex.htm. Marzo, 8, 2001

[18] JorgeCD. "Domótica, Sistemas de Control"

http://personal.redestb.es/JorgeCD/Domótica.html. Marzo, 12, 2004

[19] "¿Que es LonWorks ? "

http://www.numatico.com/Information/LonWorks.html. Marzo, 15, 2004

[20] Ramis, Juan. "La casa conectada"

http://www.urbaniza.com/hogar/domotica/lacasaconectada.html. Abril, 15 2004

[21] Sinde Ingeniería. "Domótica o control inteligente de casas y edificios"

http://www.serconet.com/com/usr/cosecas/domotica.htm. Marzo, 18, 2004

[22] Tinacria. "Diseño de una red abierta y distribuida de control "

http://www.tinacria.com/LonWorks.htm. Marzo, 14, 2004

Anexo A

Código fuente del fichero Interruptor.m

```
function interruptor(action,rule)
% Top-level M-file for an Engine-based Visual Basic application.

% Copyright 1984-2001 The MathWorks, Inc.
% $Revision: 1.8 $

switch action

case 'InOut'
  InOut

case 'addr'
  addr(rule)

end

% Other application M-files, not called from interruptor.m
if 0
  matlabrt
end
```

Código fuente del fichero InOut.m

```
function InOut
logintimeout(5);
```

```
conn3=database('domotica','','');
curs = exec (conn3,'select Radiation,Temperature,Ilumination,LigthStatus,BlindStatus,Presence from Inputs');
setdbprefs ('DataReturnFormat','numeric');
curs = fetch (curs);
InputRow = curs.data;
colnames = columnnames (curs);

fismat = readfis ('BD');

out = evalfis (InputRow,fismat);

get (conn3,'Autocommit');
colnames = {'Ligth','BlindController','Climatization'};
insert (conn3,'Outputs',colnames,out);
close (conn3);
close (curs);
```

Código fuente del fichero matlabrt.m

```
%MATLABRT_TEMPLATE Template Runtime startup M-file.
%   MATLABRT is automatically executed by the MATLAB Runtime Server during
%   startup.  MATLABRT must be placed either in the toolbox/local directory
%   or the directory from which MATLAB is started (for example, the Start in
%   directory on PC platforms).
%
%   Make a copy of MATLABRT_TEMPLATE file and rename it to MATLABRT.M
%   and place this renamed copy in your toolbox/local directory.  You can
%   then modify toolbox/local/matlabrt.m to suit your application.
%
%   See an example MATLABRT in toolbox/runtime/examples/gui for a typical
```

```
%   application.

%   Copyright 1984-2001 The MathWorks, Inc.
%   $Revision: 1.16 $  $Date: 2001/01/18 21:01:18 $

% The RecursionLimit forces MATLAB to throw an error when the specified
% function call depth is hit.  This protects you from blowing your stack
% frame (which can cause MATLAB and/or your computer to crash).  Set the
% value to inf if you don't want this protection.
cname = computer;
if strncmp(cname,'GLNX',4)
  set(0,'RecursionLimit',100)
elseif strncmp(cname,'ALPHA',5)
  set(0,'RecursionLimit',200)
else
  set(0,'RecursionLimit',500)
end

%% For Japan, set default fonts
lang = lower(get(0,'language'));
if strncmp(lang, 'ja', 2)
   if strncmp(cname,'PC',2)
      set(0,'defaultuicontrolfontname',get(0,'factoryuicontrolfontname'));
      set(0,'defaultuicontrolfontsize',get(0,'factoryuicontrolfontsize'));
      set(0,'defaultaxesfontname',get(0,'factoryuicontrolfontname'));
      set(0,'defaultaxesfontsize',get(0,'factoryuicontrolfontsize'));
      set(0,'defaulttextfontname',get(0,'factoryuicontrolfontname'));
      set(0,'defaulttextfontsize',get(0,'factoryuicontrolfontsize'));
   end
end
```

```
% Set up the path for the Runtime Application by calling pathdefrt.
if exist('pathdefrt')
   matlabpath(pathdefrt);
end

% Set the errormode.  Possibilities are
%   'quit'     -- When an error occurs display the error. Force users to quit.
%   'dialog'   -- When an error occurs display the error. Allow users to
%                 choose between continue and quit.
%   'continue' -- When an error occurs ignore it silently.

if (ispc)
  runtime('errormode','dialog');
else
  runtime('errormode','continue');
end

% Now call your application.  Put it in a try-catch construct so you
% can handle any errors that may occur.
  % Code segment to invoke your application.
InOut
```

Código fuente del fichero pathdefrt.m

```
function p = pathdefrt
%PATHDEFRT_TEMPLATE Template for search path defaults.
%   PATHDEFRT returns a string that can be used as input to MATLABPATH
%   in order to set the path.  You would typically call this file
%   from MATLABRT like this:
%
%   matlabpath(pathdefrt);
```

```
%
%   This file works on all platforms
%
%   This is an example file that contains only the path to the MATLAB
%   toolbox.  You must edit it and append any other directories you have for
%   your m-files.  Note that to be platform independent the path
%   must be relative to MATLABROOT ($toolbox in the string below).
%
%   Make a copy of PATHDEFRT_TEMPLATE file and rename it to PATHDEFRT.M
%   and place this renamed copy in your toolbox/local directory.  You can
%   then modify toolbox/local/pathdefrt.m to suit your application.

%   Copyright 1984-2001 The MathWorks, Inc.
%   $Revision: 1.14 $ $Date: 2001/01/18 21:01:18 $

% PATH DEFINED HERE
p = ['$toolbox\local:',...
   '$toolbox\DomoticaDir:',...
   '$toolbox\database\database:',...
   '$toolbox\fuzzy\fuzzy:',...
   '$toolbox\matlab\datafun:',...
   '$toolbox\matlab\datatypes:',...
   '$toolbox\matlab\elfun:',...
   '$toolbox\matlab\elmat:',...
   '$toolbox\matlab\funfun:',...
   '$toolbox\matlab\general:',...
   '$toolbox\matlab\graphics:',...
   '$toolbox\matlab\iofun:',...
   '$toolbox\matlab\lang:',...
   '$toolbox\matlab\ops:',...
   '$toolbox\matlab\specfun:',...
```

```
    '$toolbox\matlab\strfun:',...
    '$toolbox\matlab\uitools:',...
    '$toolbox\symbolic:',...
];

% Note: toolbox/local is required to allow MATLAB to run MATLABRT.

p = translate(p); % Translate p to a platform specific string

%--translate ----------------------------------------------------
function p = translate(p);
%TRANSLATE Translate unix path to platform specific path

cname = computer;
if strncmp(cname,'PC',2)
  p = strrep(p,'/','\');
  p = strrep(p,':',';');
  p = strrep(p,'$',[matlabroot '\']);
else
  p = strrep(p,'$',[matlabroot '/']);
end
Codigo fuente del fichero addr.m
function addr(rule);
BD = readfis ('BD');
BD = addrule (BD,rule);
```

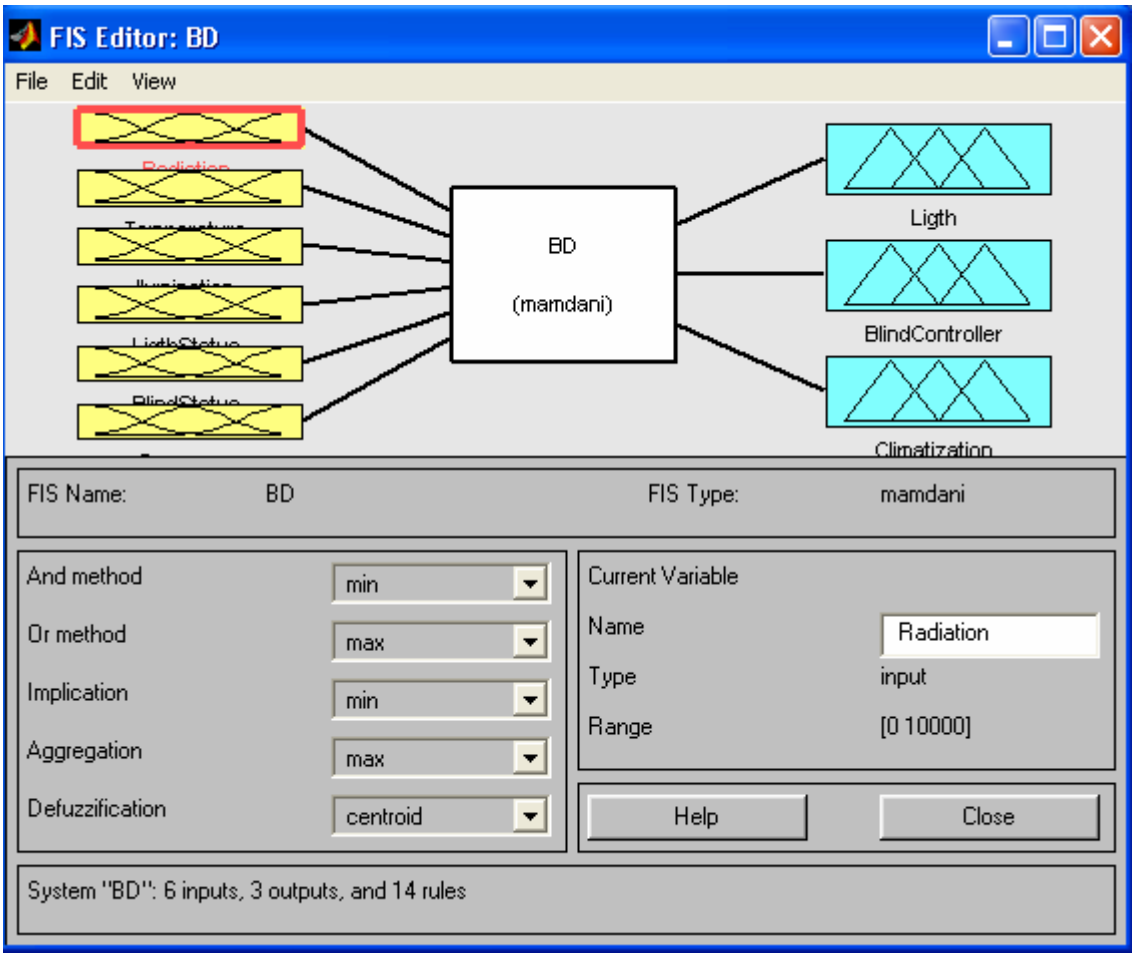

Figura 1 FIS Editor del BD.fis

Propiedades de BD

name: 'BD'

type: 'mamdani'

andMethod: 'min'

orMethod: 'max'

defuzzMethod: 'centroid'

impMethod: 'min'

aggMethod: 'max'

input: [1x6 struct]

output: [1x3 struct]

rule: [1x15 struct]

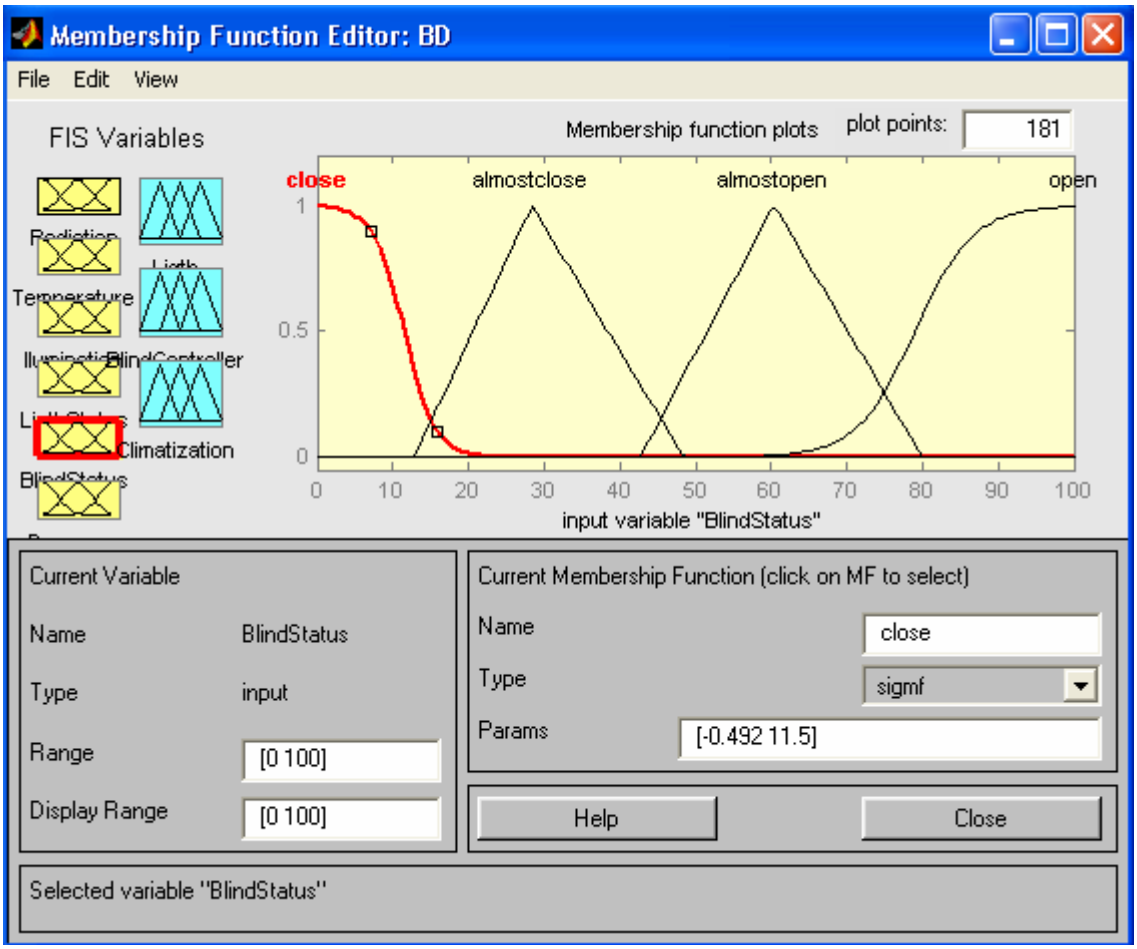

Figura 2 Membership Function Editor de la variable BlindStatus

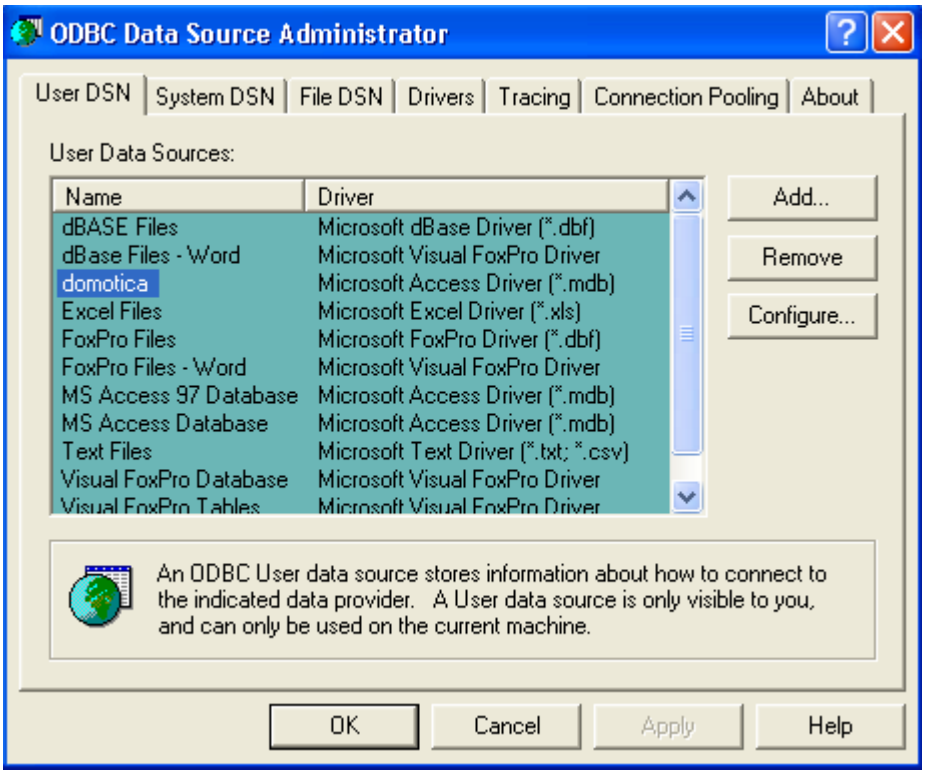

Figura 3 Data Source Administrator

Anexo B

Códigos fuente

```
program Domotica;
uses
  Forms,
  MainUnit in 'MainUnit.pas' {MainForm},
  GridUnit in 'GridUnit.pas' {GridForm},
  FieldsUnit in 'FieldsUnit.pas' {FieldForm},
  UsersUnit in 'UsersUnit.pas' {UsersForm},
  RemoteUnit in 'RemoteUnit.pas' {RemoteForm};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TGridForm, GridForm);
  Application.CreateForm(TFieldForm, FieldForm);
  Application.CreateForm(TUsersForm, UsersForm);
  Application.CreateForm(TRemoteForm, RemoteForm);
  Application.Run;
end.

unit MainUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, ADODB, Grids, DBGrids, StdCtrls,DBTables, ExtCtrls, DBCtrls,
   Buttons;
```

```
type
  TMainForm = class(TForm)
    ListBox1: TListBox;
    DBGrid1: TDBGrid;
    ADOConnection1: TADOConnection;
    ADOTable1: TADOTable;
    DataSource1: TDataSource;
    Panel1: TPanel;
    ComboBox1: TComboBox;
    DBNavigator1: TDBNavigator;
    Label1: TLabel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    Panel2: TPanel;
    Button1: TButton;
    Timer1: TTimer;
    Label2: TLabel;
    Button2: TButton;
    Timer2: TTimer;
    Button3: TButton;
    BitBtn1: TBitBtn;
    Button4: TButton;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ListBox1DblClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
```

```
    procedure SpeedButton2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    procedure AddRandomData;
  end;

var
  MainForm: TMainForm;
  matlab : Variant;

implementation

//uses FieldsUnit;
  uses

  GridUnit,FieldsUnit,UsersUnit,RemoteUnit, ComObj;

{$R *.dfm}
const
  Radiation   = 10000;
  Temperature = 100;
  Ilumination = 10000;
  LigthStatus = 100;
  BlindStatus = 100;
```

```
  Presence    = 100;


procedure TMainForm.FormCreate(Sender: TObject);
begin

    Session.GetDatabaseNames(ComboBox1.Items);
    //Forza un valor inicial
    ComboBox1.Text := 'domotica';
    ComboBox1Change(Self);
    //selecciona el primer item
    ListBox1.ItemIndex := 0;
    ListBox1Click(Self);

end;

procedure TMainForm.ComboBox1Change(Sender: TObject);
begin
  Session.GetTableNames(ComboBox1.Text,'',True,False,ListBox1.Items);
end;

procedure TMainForm.ListBox1Click(Sender: TObject);
begin
  ADOTable1.Close;
  ADOTable1.TableName :=
    ListBox1.Items[ListBox1.ItemIndex];
    try
      ADOTable1.Open;
      ADOTable1.Active;
    Except
      ADOTable1.Close;
```

```
    end;

  Label2.Caption := Format('Table:%s ',[ADOTable1.TableName]) ;
end;

procedure TMainForm.ListBox1DblClick(Sender: TObject);
var
  GridForm : TGridForm;
begin
  GridForm := TGridForm.Create(Self);
  GridForm.ADOTable1.TableName :=
    ListBox1.Items[ListBox1.ItemIndex];
    try
      GridForm.ADOTable1.Open;
      GridForm.ADOTable1.Active;
      GridForm.Show;
    Except
      GridForm.Close;
end;
end;

procedure TMainForm.SpeedButton1Click(Sender: TObject);
var
  I :integer;
begin
  FieldForm.FieldsList.Clear;
  for I := 0 to ADOTable1.FieldCount - 1 do
  begin
    FieldForm.FieldsList.Items.Add(
      ADOTable1.Fields[I].FieldName);
    FieldForm.FieldsList.Selected[I] :=
```

```
    ADOTable1.Fields[I].Visible;
  end;
  if FieldForm.ShowModal = mrOK then
    for I := 0 to ADOTable1.FieldCount - 1 do
      ADOTable1.Fields[I].Visible :=
        FieldForm.FieldsList.Selected[I];

end;

procedure TMainForm.SpeedButton2Click(Sender: TObject);
var
  UsersForm : TUsersForm;
begin
    UsersForm := TUsersForm.Create(Self);
  UsersForm.ADOTable1.TableName := 'Users';

    try
      UsersForm.ADOTable1.Open;
      UsersForm.Show;
      UsersForm.ADOTable1.Active := true;
    Except
      UsersForm.Close;
    end;
end;
procedure TMainForm.Button1Click(Sender: TObject);

begin

  try
  matlab := CreateOleObject('Matlab.Application.6');
```

```
  except
  ShowMessage('Can not open Matlab');
  end;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
  Timer2.Enabled := True;
  if  ADOTable1.TableName <>'Inputs'  then
    ADOTable1.Active := false;
    ADOTable1.Close;

  AddRandomData;
end;

procedure TMainForm.AddRandomData;

begin
  ADOTable1.TableName := 'Inputs';
  ADOTable1.Active;
  ADOTable1.Open;
  Randomize;

    ADOTable1.InsertRecord ([
      Random (Radiation) + 1,
      Random (Temperature) + 1,
      Random (Ilumination) + 1,
      Random (LigthStatus) + 1,
      Random (BlindStatus) + 1,
      Random (Presence) + 1]);
```

```
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
  Timer2.Enabled := False;
end;

procedure TMainForm.BitBtn1Click(Sender: TObject);
var
  RemoteForm : TRemoteForm;
begin
  RemoteForm := TRemoteForm.Create(Self);
  RemoteForm.Show;
end;

procedure TMainForm.Button4Click(Sender: TObject);
var
  s: WideString;
begin
  if CheckBox2.Checked = True then
  begin
  Timer1.Enabled := CheckBox1.Checked;
  s := matlab.Execute('interruptor("InOut")')
  end
  else
  ShowMessage('To evaluate the controller you should activate the server');
end;

procedure TMainForm.CheckBox1Click(Sender: TObject);
begin
  Timer1.Enabled := True;
```

```
end;

end.

unit RemoteUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, DB, ADODB;

type
  TRemoteForm = class(TForm)
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    RadioGroup3: TRadioGroup;
    Panel1: TPanel;
    Image1: TImage;
    Button1: TButton;
    ADOTable1: TADOTable;
    ADOConnection1: TADOConnection;
    Panel2: TPanel;
    CheckBox1: TCheckBox;
    Panel3: TPanel;
    CheckBox2: TCheckBox;
    procedure Button1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  RemoteForm: TRemoteForm;
  matlab : Variant;

implementation

uses ComObj ;

{$R *.dfm}
const
  ONc  = 30;
  OFFc = 100 ;

procedure TRemoteForm.Button1Click(Sender: TObject);
var
   LigthStatus, BlindStatus,Climatizacion : Integer;
   Rad, Temp,Ilum, Ligth, Blind, Presence : Variant;
   s   : WideString;
   mfrad, mftemp, mfilum, mfligth, mfblind, mfpres  : Integer;
   mfligthout, mfblindout, mfclimatizationout : Integer;
   rule : Variant;


begin
  if  ADOTable1.TableName <> 'Outputs'  then
    ADOTable1.Active := false;
    ADOTable1.Close;
```

```
ADOTable1.TableName := 'Outputs';
ADOTable1.Open;
ADOTable1.Active;
if RadioGroup1.Items[RadioGroup1.ItemIndex] = 'ON' then
  LigthStatus := random (30) + 1
else
  LigthStatus := random (40) +61 ;
if RadioGroup2.Items[RadioGroup2.ItemIndex] = 'CLOSE' then
  BlindStatus := random (30) + 1
else
  BlindStatus := random (30) + 71;
if RadioGroup3.Items[RadioGroup3.ItemIndex] = 'OFF' then
  Climatizacion := random (20) + 1
else
  if RadioGroup3.Items[RadioGroup3.ItemIndex] = 'FAN' then
    Climatizacion := random (40) + 1
else
  Climatizacion := random (40) + 61;

  ADOTable1.InsertRecord([LigthStatus,BlindStatus,Climatizacion]);

  if CheckBox1.Checked = true then
  begin
    if ADOTable1.TableName <> 'Inputs' then
      ADOTable1.Active := False;
      ADOTable1.Close;
      ADOTable1.TableName := 'Inputs';
      ADOTable1.Open;
      ADOTable1.Active;
      ADOTable1.FindLast;
      Rad := ADOTable1.fieldbyname ('Radiation').Value;
```

```
Temp := ADOTable1.fieldbyname ('Temperature').Value;
Ilum := ADOTable1.fieldbyname ('Ilumination').Value;
Ligth := ADOTable1.fieldbyname ('LigthStatus').Value;
Blind := ADOTable1.fieldbyname ('BlindStatus').Value;
Presence := ADOTable1.fieldbyname ('Presence').Value;

case Rad of
  1000..3499: mfrad := 1;
  15..199   : mfrad := 2;
  200..999  : mfrad := 3;
  3500..5000: mfrad := 4;
  0..14     : mfrad := 5;
end;
case Temp of
  40..100   : mftemp :=1;
  18..39    : mftemp :=2;
  8..17     : mftemp :=3;
  4..7      : mftemp :=4;
  0..3      : mftemp :=5;
end;
case Ilum of
  900..3499 : mfilum :=1;
  15..249   : mfilum :=2;
  250..899  : mfilum :=3;
  0..14     : mfilum :=4;
  3500      : mfilum :=5;
 end;
 case Ligth of
   0..49   : mfligth :=1;
   50..100 : mfligth :=2;
 end;
```

```
case Blind of
  0..11  : mfblind :=1;
  12..39 : mfblind :=2;
  40..79 : mfblind :=3;
  80..100: mfblind :=4;
end;
case Presence of
  0..49  : mfpres :=1;
  50..100: mfpres :=3;
end;
case LigthStatus of
  0..49   : mfligthout :=1;
  50..100 : mfligthout :=2;
end;
case BlindStatus of
  0..5  : mfblindout:=1;
  6..10 : mfblindout:=2;
end;
case Climatizacion of
  0..19  : mfclimatizationout :=1;
  20..39 : mfclimatizationout :=2;
  40..100: mfclimatizationout :=3;
end;


rule := VarArrayCreate([0, 10],varVariant);
  rule [0]:= mfrad;
  rule [1]:= mftemp;
  rule [2]:= mfilum;
  rule [3]:= mfligth;
  rule [4]:= mfblind;
```

```
        rule [5]:= mfpres;
        rule [6]:= mfligthout;
        rule [7]:= mfblindout;
        rule [8]:= mfclimatizationout;
        rule [9]:=1;
        rule [10]:=1;

            s := matlab.Execute ('interruptor ("addr",rule)');

   end;
   end;

procedure TRemoteForm.CheckBox2Click(Sender: TObject);
begin
  matlab :=CreateOleObject('Matlab.Application.6');
end;

procedure TRemoteForm.CheckBox1Click(Sender: TObject);
begin
  if CheckBox2.Checked = False then
    CheckBox2.Checked := True;
end;

end.

unit UsersUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
  Dialogs, DB, ADODB, ExtCtrls, Grids, DBGrids, DBCtrls, Buttons, StdCtrls;

type
  TUsersForm = class(TForm)
    DBGrid1: TDBGrid;
    Panel1: TPanel;
    ADOTable1: TADOTable;
    DataSource1: TDataSource;
    DBNavigator1: TDBNavigator;
    Button1: TButton;
    procedure SpeedButton1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    procedure AddRandomData;
  end;

var
  UsersForm: TUsersForm;

implementation

{$R *.dfm}
const
  FirstName : array [1..10] of string =
    ('Joel', 'Javier', 'Adolfo', 'Clara', 'Marisel',
    'Rafael', 'Daniel', 'Azucena', 'Yolanda', 'Nery');

  LastName : array [1..10] of string =
    ('Rodriguez', 'Falcon', 'Permuy', 'Morales', 'Acosta',
```

```
    'Bormey', 'Torres', 'Perez', 'Garcia', 'Bermudez');

  Country : array [1..10] of string =
    ('Cuba', 'Venezuela', 'Brazil', 'EUA', 'Canada',
    'Colombia', 'Suecia', 'Alemania', 'Italia', 'Francia');

  Passport = 1000;
  RoomNumber = 100;
  NewRecords = 10;

procedure TUsersForm.AddRandomData;
var
  I: Integer;
begin
  Randomize;
  for I := 1 to NewRecords do
    ADOTable1.InsertRecord ([
      FirstName [Random (High (FirstName)) + 1],
      LastName [Random (High (LastName)) + 1],
      Country [Random (High (Country)) + 1],
      Random (Passport) + 1,
      Random (RoomNumber) + 1,
      Date - Random (10000)]);

  ShowMessage (IntToStr (NewRecords) + ' added');
end;

procedure TUsersForm.SpeedButton1Click(Sender: TObject);
begin
  //ADOTable1.Close;
  UsersForm.Close;
```

```
end;

procedure TUsersForm.Button1Click(Sender: TObject);
begin
  AddRandomData;
end;
end.

unit FieldsUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;

type
  TFieldForm = class(TForm)
    FieldsList: TListBox;
    Label1: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FieldForm: TFieldForm;
```

```
implementation

{$R *.dfm}

end.

unit GridUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, DB, ADODB, DBCtrls, ExtCtrls, Buttons;

type
  TGridForm = class(TForm)
    DBGrid1: TDBGrid;
    ADOTable1: TADOTable;
    DataSource1: TDataSource;
    Panel1: TPanel;
    DBNavigator1: TDBNavigator;
    SpeedButton1: TSpeedButton;
    procedure SpeedButton1Click(Sender: TObject);
    procedure DataSource1DataChange(Sender: TObject; Field: TField);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  GridForm: TGridForm;

implementation

uses FieldsUnit, StdCtrls;

{$R *.dfm}

procedure TGridForm.SpeedButton1Click(Sender: TObject);
  var
  I :integer;
begin
  FieldForm.FieldsList.Clear;
  for I := 0 to ADOTable1.FieldCount - 1 do
  begin
    FieldForm.FieldsList.Items.Add(
      ADOTable1.Fields[I].FieldName);
    FieldForm.FieldsList.Selected[I] :=
      ADOTable1.Fields[I].Visible;
  end;
  if FieldForm.ShowModal = mrOK then
    for I := 0 to ADOTable1.FieldCount - 1 do
      ADOTable1.Fields[I].Visible :=
        FieldForm.FieldsList.Selected[I];

end;


procedure TGridForm.FormCreate(Sender: TObject);
begin
```

```
  GridForm.Caption := Format('Table: %s ',[ADOTable1.TableName]) ;
end;

end.
```

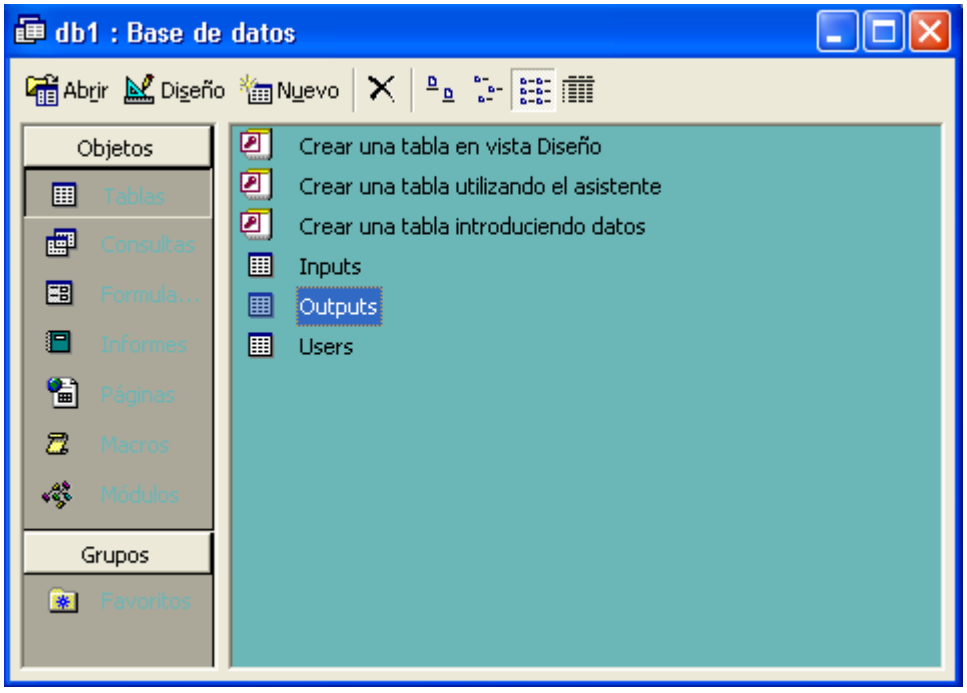

Figura 1 Bases de Datos db1.mdb en Microsoft Access

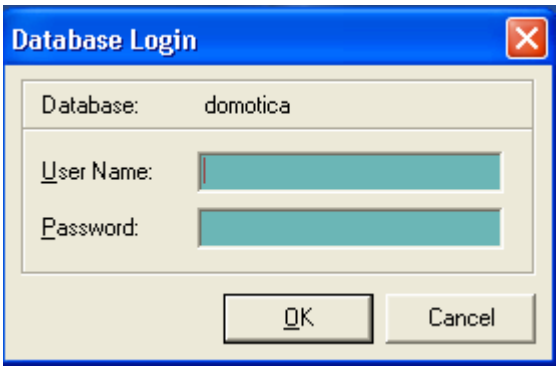

Figura 2 Database Login

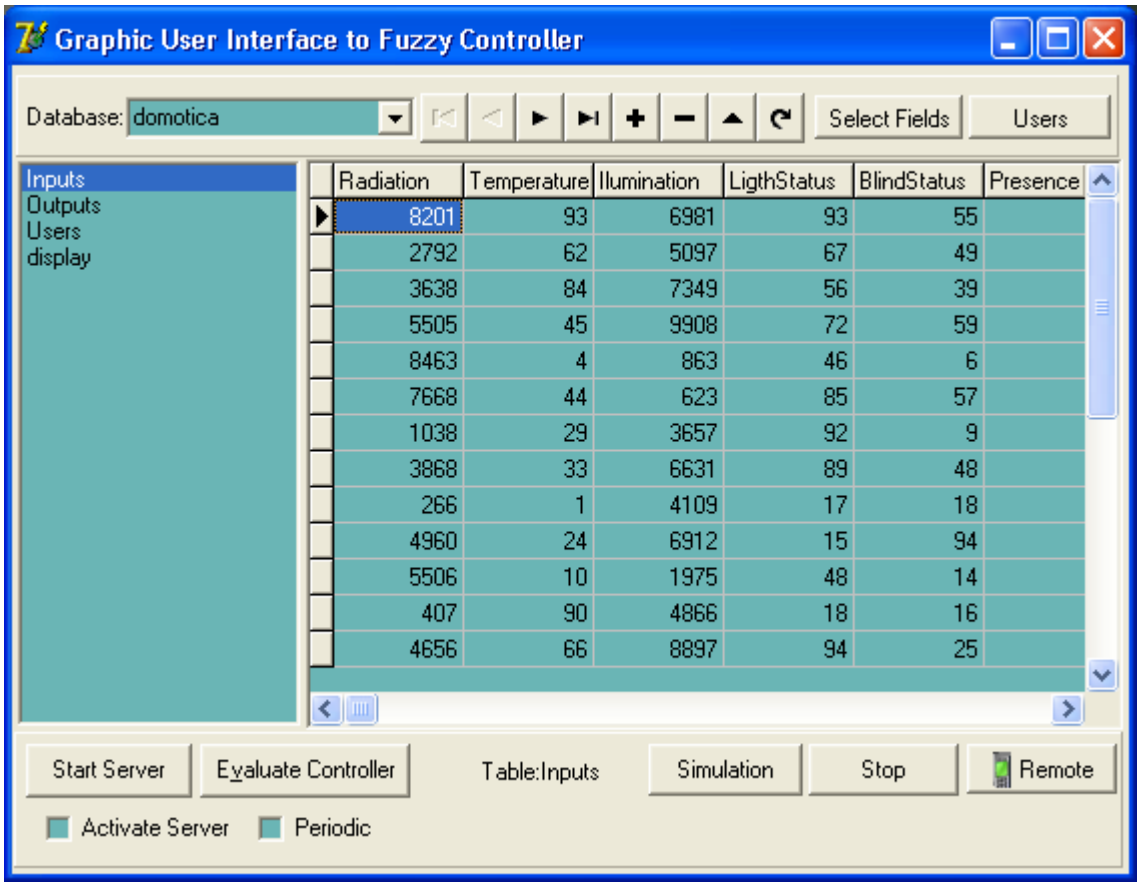

Figura 3 RoomAgentDisplay

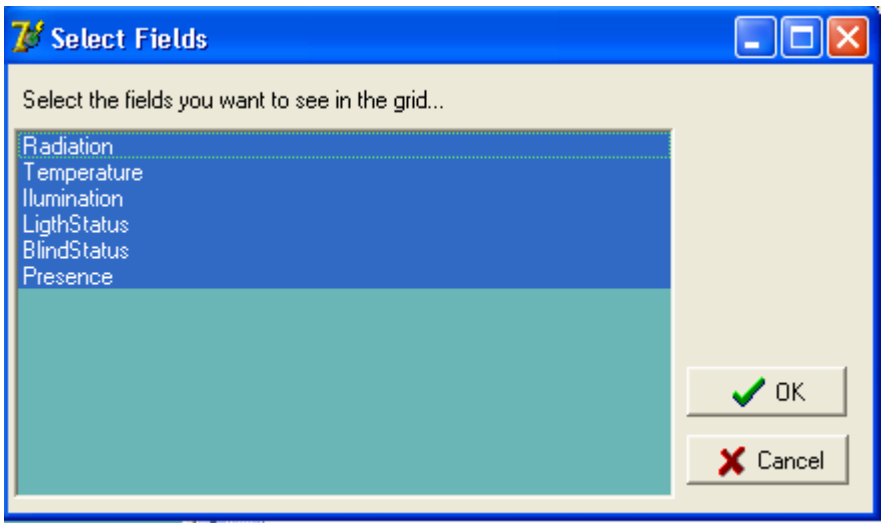

Figura 4 Select Fields

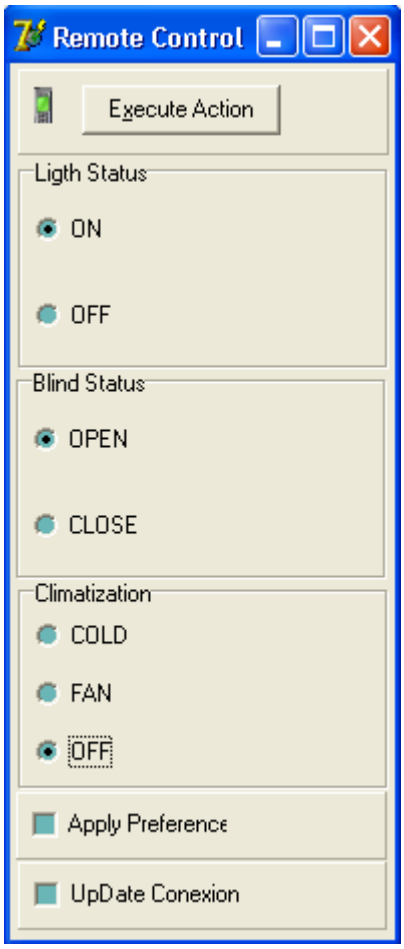

Figura 5 Control Remoto

Anexo C

Lógica difusa

Básicamente, la lógica difusa proporciona un medio efectivo de concebir la naturaleza aproximada e imprecisa del mundo real. Visto desde otra perspectiva, la parte esencial del control y supervisión difusos es un conjunto de reglas lingüísticas relacionadas por los conceptos duales de implicación difusa y regla de inferencia composicional.

En esencia, el control y supervisión difusos proporcionan mecanismos que permiten convertir una estrategia lingüística basada en el conocimiento experto en una estrategia automática. La experiencia muestra como los resultados de aplicación de control difuso resultan mejores que los obtenidos por técnicas convencionales, en particular cuando los procesos son demasiado complejos para ser analizados por técnicas cuantitativas convencionales, o cuando las fuentes disponibles de información son interpretadas de modo cualitativo, impreciso o incierto.

Conjuntos Difusos.

Un conjunto clásico, a diferencia de uno difuso, está constituido por un grupo de elementos que se caracterizan por poseer algo en común y la frontera que determina si un elemento pertenece o no a dicho conjunto está perfectamente definida, de modo que se puede afirmar o negar categóricamente dicha pertenencia.

Si definimos un conjunto **U** de objetos (universo de discurso) cuyos elementos genéricos se representan como *u*, la pertenencia a un subconjunto **A** de **U** puede verse como una cierta función característica μ_a de μ {0, 1} tal que:

$\mu_a(x) = 1$ si x $\in$ A

$\mu_a(x) = 0$ si x $\notin$ A

Esta lógica convencional puede ser interpretada como un caso particular de la Lógica Difusa, digamos que μ es el grado de pertenencia a un conjunto que puede tomar valores

entre 0 y 1 y representa el grado en que determinada proposición es verdadera. Si hacemos que μ sea igual a 0 indicaremos no pertenencia, si la igualamos a 1 indicaremos pertenencia total.

Definición 1: Conjunto difuso: Un conjunto difuso **A** en un universo de discurso **U** está caracterizado por una función de pertenencia μ_a la cual toma valores en el intervalo [0, 1] y se representa de la forma μ_a :**U**$\rightarrow$[0, 1].

Definición 2: Un conjunto difuso **A** es nulo o vacío si y solo si su función de pertenencia es cero $\mu_a = 0$.

Definición 3: Dos conjuntos difusos **A** y **B** son iguales y se denominan **A**=**B**, si y solo si $\mu_a(u) = \mu_b(u), \ \forall u \in U$

Definición 4: Un conjunto difuso **A** es contenido en otro conjunto **B** (**A** $\subset$ **B**) si $\mu_a(u) \leq \mu_b(u), \ \forall uU$

Propiedades de los conjuntos difusos.

Sean **A** y **B** dos conjuntos difusos en **U** con funciones de pertenencia μ_a y μ_b respectivamente. El conjunto de las operaciones teóricas para estos conjuntos queda definido a través de sus funciones de pertenencia en la forma siguiente:

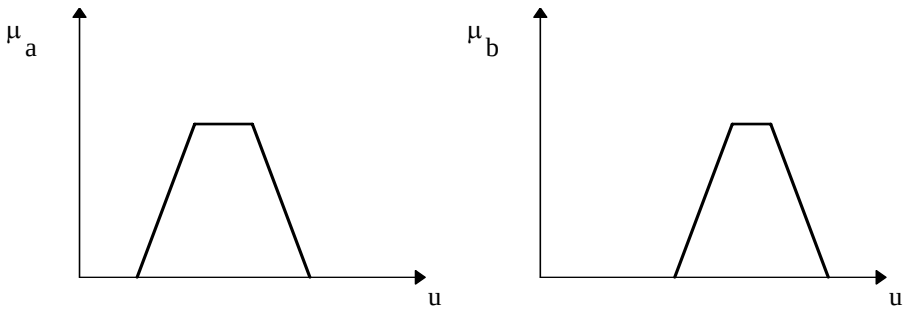

Figura 1 Funciones de pertenencia de los conjuntos difusos A y B.

Definición 5: Unión: La función de pertenencia $\mu_{a\cup b}$ resultante de la unión de los conjuntos difusos **A** y **B** está definida punto a punto para todo $u \in U$ por $\mu_{a\cup b}(u)=\max\{\mu_a(u),\mu_b(u)\}$.

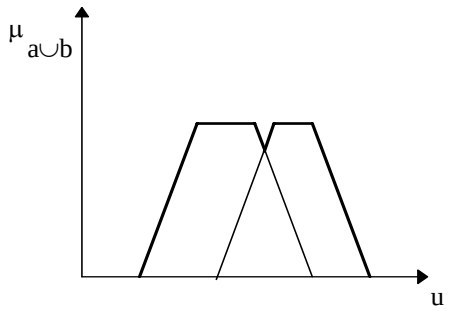

Figura 2

Definición 6: Intersección: La función de pertenencia $\mu_{a\cap b}$ resultante de la intersección de los conjuntos difusos **A** y **B** está definida punto a punto para todo $u \in U$ por $\mu_{a\cap b}(u)=\min\{\mu_a(u),\mu_b(u)\}$.

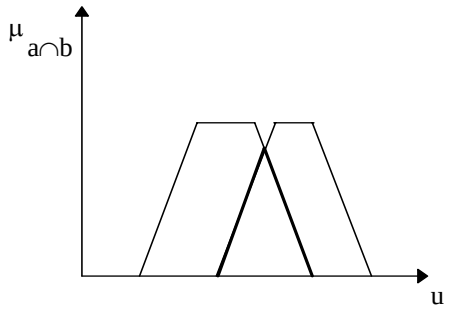

Figura 3

Definición 7: Complemento: La función de pertenencia $\mu_{\bar{a}}$ del complemento de un conjunto difuso **A** está definida punto a punto para todo $u \in U$ por $\mu_{\bar{a}}(u)=1-\mu_a(u)$.

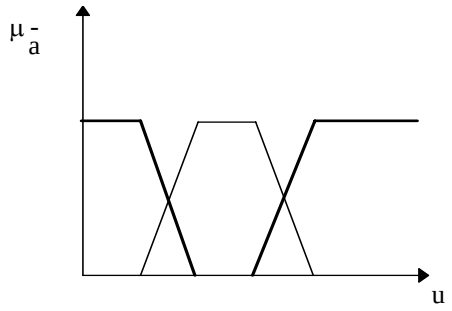

Figura 4

Definición 8: Suma cartesiana: Si $\mathbf{A}_1$, $\mathbf{A}_2$, …$\mathbf{A}_n$ son conjuntos difusos en $\mathbf{U}_1$, $\mathbf{U}_2$, …$\mathbf{U}_n$, respectivamente, la suma cartesiana de $\mathbf{A}_1$, $\mathbf{A}_2$, …$\mathbf{A}_n$ es un conjunto difuso con función de pertenencia igual a:

$$\mu_{a1+a2+\cdots+an}(u_1,u_2,\ldots,u_n) = \mu_{a1}(u_1)+\mu_{a2}(u_2)+\ldots+\mu_{an}(u_n) - \mu_{a1}(u_1)\cdot\mu_{a2}(u_2)\cdot\ldots\cdot\mu_{an}(u_n)$$

Definición 9: Producto cartesiano: Si $\mathbf{A}_1$, $\mathbf{A}_2$, …$\mathbf{A}_n$ son conjuntos difusos en $\mathbf{U}_1$, $\mathbf{U}_2$, …$\mathbf{U}_n$, respectivamente, el producto cartesiano de $\mathbf{A}_1$, $\mathbf{A}_2$, …$\mathbf{A}_n$ es un conjunto difuso con función de pertenencia igual a:

$$\mu_{a1\times a2\times\cdots\times an}(u_1,u_2,\ldots,u_n) = \min\{\mu_{a1}(u_1),\ldots,\mu_{an}(u_n)\}$$

o

$$\mu_{a1\times a2\times\cdots\times an}(u_1,u_2,\ldots,u_n) = \mu_{a1}(u_1)\cdot\mu_{a2}(u_2)\cdot\ldots\cdot\mu_{an}(u_n)$$

Variable lingüística.

Definición 10: Una variable lingüística esta caracterizada por una quíntupla (x, T(x), U, G, M). en la cual:

x: es el nombre de la variable.

T(x):Conjunto de valores lingüísticos de x, donde cada valor lleva asignado su significado.

U: Es el universo de discurso de definición de los valores lingüísticos.

M: Son las reglas semánticas de asignación de significados. Se le asocia cada valor a su significado.

G: Son las reglas sintácticas de generación de términos compuestos, a partir de los términos atómicos que configuran las sentencias que dan lugar a cada valor lingüístico.

Estos términos pueden ser de cuatro tipos:.

1. Términos primarios (viejo, joven, alto etc.) que son en si mismo elementos del conjunto T(x).

2. Operadores lógicos (AND, OR, NOT)

3. Modificadores lingüísticos difusos (muy, ligeramente, más o menos).

4. Marcadores para la delimitación de los resultados (paréntesis).

Para ilustrar el concepto de variable lingüística utilizaremos un ejemplo, donde se pretende describir la temperatura de un proceso, no por sus valores numéricos, sino por términos lingüísticos. En este caso el nombre de la variable (x) es obviamente, temperatura, y el conjunto de términos (T(x)) se diseñará tan amplio como sea necesario para especificar mejor el problema, o sea:.

T(temperatura)={Alta, no Alta, Baja, Media, más o menos Alta....}

Con respecto al universo de discurso (U), este comprenderá un rango de temperatura apropiado para definir los valores difusos significativos para el problema en toda su extensión, (digamos entre 0 y 500 ^{0}C). Lo que hemos denominado reglas sintácticas, (de construcción de términos, compuestos a partir de los atómicos) asumen como primitiva tres términos primarios: Alta, Baja y Media. Términos como no Alta y más o menos Alta surgen de la gramática que conforma este conjunto de reglas sintácticas.

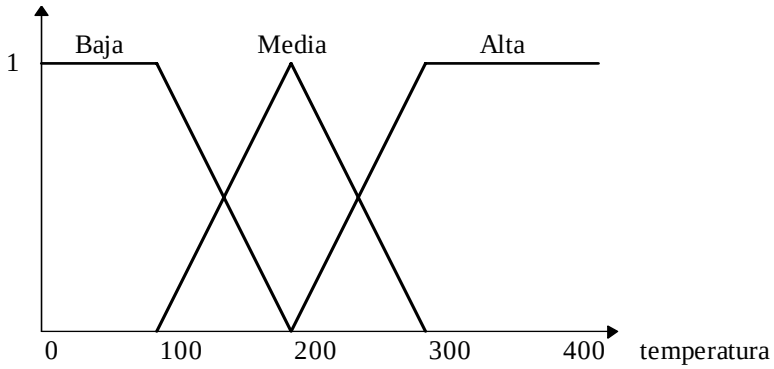

Figura 5

Razonamiento Aproximado.

En los sistemas basados en la lógica difusa y razonamiento aproximado, el método de inferencia es conocido como regla composicional de inferencia, sugerido por Zadeh en 1973, la esencia de este modelo es un programa basado en reglas (clasifica entre los llamados sistemas expertos) donde la antecedencia y la consecuencia de estas reglas están asociadas a variables lingüísticas. En nuestra terminología, una regla de inferencia difusa no es más que una sentencia condicional en la cual la antecedencia es una condición en el dominio de la aplicación, mediante variables lingüísticas, y la consecuencia es la acción. también lingüística., que se le hace corresponder por el experto.

Reglas

La escritura de reglas depende en última instancia del conocimiento del experto, así como de las características generales del sistema en si. Los especialistas han coincidido en dividirlas en los siguientes tipos fundamentales:

Reglas clásicas

Esta es quizás la regla más utilizada, en forma general adopta la siguiente estructura :

si:	(premisa 1) es (arreglo Difuso) y (premisa 2) es (arreglo Difuso)
entonces:	(consecuencia) es (arreglo Difuso).

Estructura de un controlador difuso.

La configuración básica de un controlador difuso se indica en la Figura 6

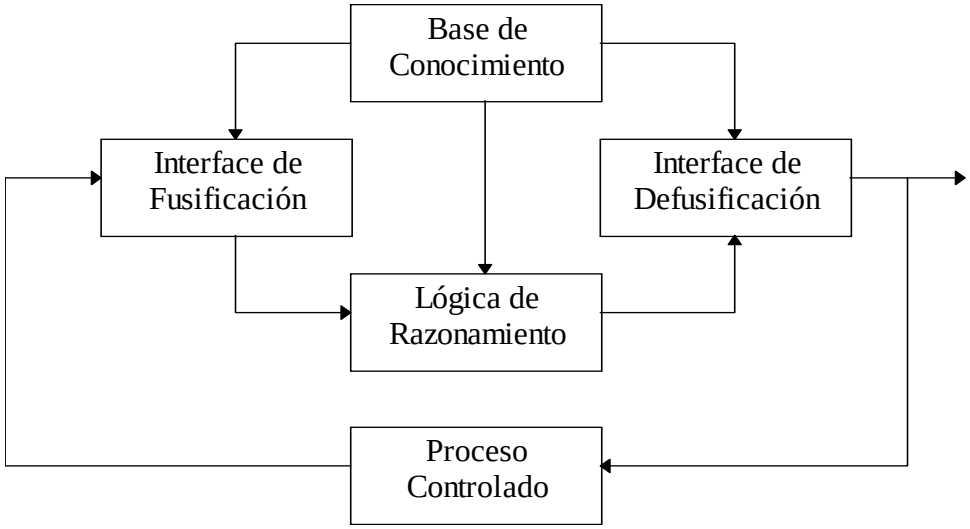

Figura 6

En el caso de que el sistema difuso se dedique a la supervisión y no al control directo, el esquema es similar, salvo que el bloque "Proceso" ya no representa la planta aislada, sino, la planta con sus controladores convencionales algunos de cuyos parámetros se ajustan desde el controlador difuso.

La interfaz de fusificación

La interfaz de fusificación deberá realizar las siguientes tareas:

- Medida de las variables del proceso .
- Escalado de las mismas, que transfieran su rango de valores a los correspondientes universos de discurso.

- Fusificación, esto es , conversión a valores lingüísticos que pueden ser considerados como etiquetas de conjuntos difusos.

El operador humano analiza el estado del proceso (sus salidas), de manera cualitativa, aun cuando cada variable tiene un valor dado en cada instante de tiempo. Un instrumento de temperatura podría indicar al C.L.F. que la temperatura es de 310 ^{0}C. Lo primero que deberá hacer este controlador es llevar este valor en términos de la variable lingüística temperatura. Si el universo de discurso fuera el de la fig.2.2, dicho valor determinístico podría ser expresado como 0.8 "Medio" y 0.2 "Alto" y 0 "Bajo".

Del número de particiones, o sea el número de elementos de T(x), influye decisivamente el método que se emplee para la creación del algoritmo del C.L.F., aunque debe quedar claro que el número de particiones es arbitrario así como la forma que tendrán las funciones de pertenencia.

Quizás la forma más elemental de crear un algoritmo es la denominada verbalización, se basa exclusivamente en descripciones verbales del experto humano que no pocas veces están cargadas de subjetivismo. Puede suceder que el mismo proceso sea dirigido de manera satisfactoria, aun cuando dos expertos no coincidan en que la cantidad de particiones de la variable lingüística sea las mismas.

La base de conocimientos

La base de conocimientos contiene el conocimiento específico de la aplicación y está formada por:

- Una base de datos con las definiciones utilizadas en las reglas lingüísticas.

- La base de reglas lingüísticas que caracteriza los objetivos del control.

Se incluyen en este bloque funcional los datos e informaciones suministrados al C.L.F. tales que permitan el procesamiento matemático de las variables medidas del proceso a fin de generar las acciones de control.

Forman parte de esta base, los parámetros necesarios para la discretización de las variables. El número de niveles de cuantificación resulta de un compromiso entre la precisión la sensibilidad a que se aspire, la memoria computacional y la velocidad de cómputo de que se dispone. Dado que la matemática difusa define operaciones entre variables de diferente entidad física, se hace imprescindible la normalización de los universos de discurso respectivos. Para ello hay que indicarle al C.L.F. el o los intervalos que se usarán. Los más frecuentes son [0,1] y [-1,1].

A esta altura se evidencia una vez más la importancia de las reglas para el C.L.F.. La base de conocimientos comprende pues el conjunto de reglas de producción que deben serle prefijadas. El número máximo de reglas viene dada por el de los números de particiones de todas las variables de entrada al C.L.F.. Para hablar de una base de reglas, es necesario definir cuales son las variables de entrada y las de salida. No está de más señalar que una selección adecuada de las variables de estado del proceso y las de acción resultan esencial para la caracterización del sistema difuso y tendrá un efecto sustancial en el ulterior comportamiento del C.L.F.. Ejemplos típicos de entrada a un C.L.F. son el estado, el error del estado, suma del error del estado etc.

Propiedades de la base de conocimiento.

1. Completamiento: Aptitud de un algoritmo de control de poder inferir una acción de control adecuada para cada estado del proceso. El comportamiento de un C.L.F. se relaciona tanto con su base de datos como con su base de reglas.

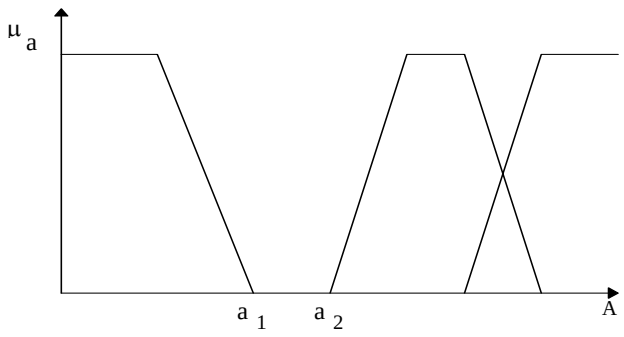

Figura 7

Entre a_1 y a_2 no hay pertenencia a ningún conjunto difuso y por lo tanto le corresponde un valor de pertenencia nulo. Con esta partición cuando el estado caiga en ese intervalo, la acción será cero, cualquiera sea el valor de las restantes variables. Por su parte el completamiento de la base de reglas es más fácil de garantizar introduciendo en el antecedente reglas que provean todas las posibles combinaciones de atributos de las variables de entrada al controlador, a costa del gravamen de la memoria .

2. Consistencia: Como las reglas generalmente se obtienen de fuentes humanas, las mismas tienen posibilidad de errores subjetivos que dan lugar a inconsistencia.

Lógica de la toma de decisiones

Este proceso es el núcleo del sistema difuso. Tiene la capacidad de simular el razonamiento humano utilizando las reglas de inferencia y la lógica difusa.

Si partimos de que nuestra base de reglas son del tipo clásico o sea de la forma:

si:	[Estado del proceso]
entonces:	[Acción de control]

para un estado inicial x_0, se activa un determinado número de reglas, aquellas en las que todos los conjuntos difusos del antecedente resultan ser para el estado dado, no nulas. Generalmente no se hace necesario desarrollar un algoritmo en el controlador para la desactivación de las reglas, se verá posteriormente, las operaciones matemáticas a realizar con las reglas, en particular con la intersección se encargará de anular la partición en la acción de control de aquellas que debieron ser desactivadas.

Proceso de inferencia

El proceso de inferencia se desarrolla en varios pasos.

1. Para un regulador que toma en cuenta n variables de entrada, aparecerán n grados de pertenencia o membresía. Esto se trata con el nombre de medida de compatibilidad a través de un operador de compatibilidad.

2. Cada regla de control maneja en principio varias variables lingüísticas, a las cuales está asociado un valor determinado de grado de pertenencia. Sin embargo a la hora de tomar una decisión tendrá que darse como salida una determinada señal de mando, que como es lógico está asociada a un único valor de μ. De aquí que el paso que le continua, es un proceso donde se lleva a cabo una agregación de los grados de pertenencias que aparecen en la parte izquierda de las reglas de control, (premisas), en un solo valor de μ. Para llevar a cabo esa operación de agregación se emplean las llamadas normas y co-normas triangulares , o sea , a cada regulador se la asignan los arreglos difusos de entrada que le corresponde, y con ello, sus respectivos grados de pertenencia.

Definición 9: Normas triangulares: La norma triangular * es una función en dos planos de [0,1]x[0,1] a [0,1] o sea, * :[0,1]x[0,1]$\rightarrow$[0,1], las cuales incluyen intersección, producto algebraico, producto acotado y producto drástico. Las operaciones asociadas con las normas triangulares están definidas para toda x, y $\in$ [0,1] como:

intersección: $x \wedge y = \min\{x, y\}$

producto algebraico: $x \cdot y = x\,y$

producto acotado: $x \odot y = \max\{0, x + y - 1\}$

producto drástico: $x \overset{\bullet}{\cap} y = \begin{cases} x & y = 1 \\ y & x = 1 \\ 0 & x, y < 1 \end{cases}$

En este caso las normas más utilizadas son: la intersección (regla del mínimo de Mandani) y el producto algebraico (regla del producto de Larsen).

Definición 10: Co-normas triangulares: La co-norma triangular $\dot{+}$ es una función en dos planos de [0,1]x[0,1] a [0,1] o sea, $\dot{+}:[0,1]x[0,1]\rightarrow[0,1]$, las cuales incluyen unión, suma algebraica, suma acotada, suma drástica y suma disyuntiva.

unión: $x \vee y = \max.\{x, y\}$

suma algebraica: $x \hat{+} y = x + y - x\,y$

suma acotada: $x \oplus y = \max\{0, x + y - 1\}$

suma drástica: $x \tilde{+} y = \begin{cases} x & y = 1 \\ y & x = 1 \\ 0 & x, y < 1 \end{cases}$

suma disyuntiva: $x \Delta y = \max\{\min(x, 1 - y) \cdot \min(1 - x, y)\}$

3. Una vez que se tiene el grado de pertenencia de todas las condiciones previas, agrupadas en un valor μ, teniendo presente que la salida del regulador deberá presentar un valor, cuyo grado de pertenencia no sobrepase al de sus condiciones previas ($y \leq \mu$), la salida se modificará en cada ciclo de trabajo con la ayuda de un operador de inferencia. De forma general pueden emplearse los operadores mínimo, producto algebraico o producto drástico,. siendo los dos primeros los más utilizados en la práctica.

En el caso de utilizar una operación de agregación intersección o unión, se recomienda utilizar el operador mínimo. Si se emplea una operación de agregación producto, se recomienda entonces, la utilización del operador producto algebraico.

La diferencia entre el uso de estos dos operadores radica en que el producto algebraico da como resultado un área que es menor o igual a la del operador mínimo ().

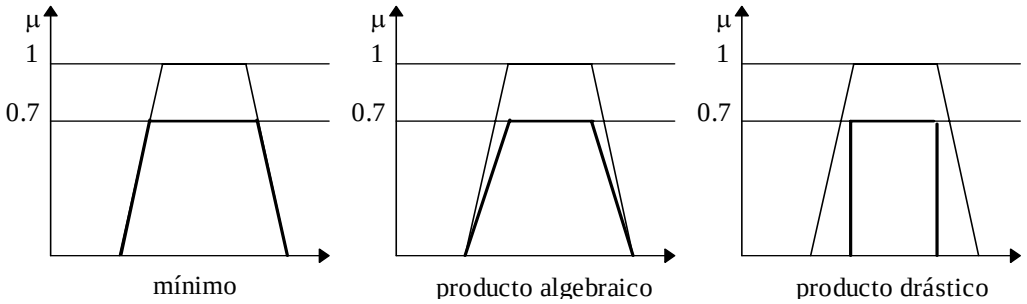

Figura 8

4. Si estamos en presencia de un sistema que incluya j reguladores, aparecería como resultado de las operaciones un número igual de arreglos difusos, los cuales deben ser llevados a un solo arreglo de salida general utilizando un operador acumulación. En este caso, dentro de los operadores más conocidos se sitúan las co-normas triangulares y en la práctica el más encontrado es el operador máximo, o sea, la unión.

Si se utiliza un operador máximo de acumulación y como operador de inferencia el mínimo, o el producto algebraico, se habla. de un procedimiento de inferencia máx-mín o de un máx-prod.

Defusificación.

Como resultado del proceso de inferencia queda como resultado un arreglo difuso, o una determinada área, que a fin de cuentas, representa el valor de salida en ese instante. Sin embargo, los elementos de mando y regulación necesitan entregar al proceso una variable de control escalar(por lo general, un número real), por lo tanto, la función de esta etapa es obtener ese valor real, a partir del arreglo difuso que se obtiene durante la inferencia. generando las acciones de control deterministas a partir de la distribución probabilística resultante de la unión, aplicando para ello tres métodos fundamentales:

- Método del centro de gravedad.
- Método del máximo.

- Método de la media del máximo.

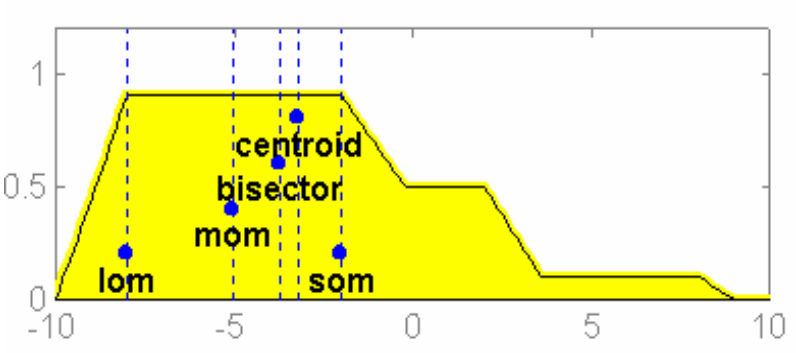

Figura 9

Método del centro de gravedad.

Para el cálculo del centro de gravedad S_x, de una función continua f en la dirección x y en un intervalo determinado se puede utilizar la ecuación:

$$S_x = \frac{\int_0^b x\, f(x)dx}{\int_0^b f(x)dx}$$

que resulta en este caso de la forma:

$$S_y = \frac{\int_0^b y\, \mu(y)dy}{\int_0^b \mu(y)dy}$$

siendo y es el valor de la salida.

De forma general esta ecuación es aproximada según la ecuación:

$$S_y = \frac{\sum_{i=1}^{n} \mu(y_i) y_i}{\sum_{i=1}^{n} \mu(y_i)}$$

donde y_i son los valores mínimos de salida para los cuales en cada arreglo el valor de μ=1.

Ventajas presentes al utilizar controladores difusos.

- Son bastantes insensibles al ruido de alta frecuencia debido al número de reglas y al efecto promedio que realizan.
- Son comparables en precisión a los analíticos.
- Consiguen buen control sobre amplio rango de condiciones iniciales.
- Son más tolerantes a fallas en los sensores.

Desventajas.

- No existe metodología general que permita la implementación.
- Necesitan expertos para extraer la estrategia de control.
- Tiempo superior de respuesta que un controlador analítico.
- Más difíciles de mantener y modificar.