Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Metodología para el desarrollo de aplicaciones de control fuzzy en Movicon

Autor: Raidán Rivero López

Tutores: Dr. Roberto Luis Ballesteros Horta

Dr. Boris Luis Martínez Jiménez

Santa Clara

2015

"Año 57 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Metodología para el desarrollo de aplicaciones de control fuzzy en Movicon

Autor: **Raidán Rivero López**

Email: rrlopez@uclv.edu.cu

Tutores: **Dr. Roberto Luis Ballesteros Horta**

Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV

Email: rball@uclv.edu.cu

Dr. Boris Luis Martínez Jiménez

Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV

Email: boris@uclv.edu.cu

Santa Clara

2015

"Año 57 de la Revolución"

PENSAMIENTO

"La incertidumbre es una posición incómoda,

pero la certeza es una posición absurda"

Voltaire.

DEDICATORIA

A mis padres,

en el significado más amplio y sincero de la palabra,

por su apoyo incondicional, sus consejos y guía durante toda mi vida, y sobre todo su cariño.

AGRADECIMIENTOS

A mi abuela Esther, que constituye todo un ejemplo a seguir para mí, por su enseñanza y amor.

A mi tío Armando por ser el hermano que nunca tuve.

A mis tutores Roberto Ballesteros y Boris, por su ayuda y paciencia en todo momento.

A todos los familiares y amigos que forman parte de mi vida y son parte también de este logro.

A todos, sinceramente,

Muchas Gracias.

RESUMEN

El SCADA Movicon, aun cuando es utilizado con regularidad en Cuba, no presenta módulos o herramientas de control fuzzy. En esta problemática se sustenta la presente investigación, que parte de un intenso análisis bibliográfico. En el trabajo se desarrollan procedimientos y herramientas que posibilitan implementar aplicaciones de control fuzzy en Movicon, a partir de lo cual se propone una metodología para el desarrollo de este tipo de aplicaciones. Se toma un proceso como ejemplo para demostrar la aplicación de la metodología. El trabajo sistematiza los conocimientos sobre sistemas SCADA y la implementación de control inteligente en dichos sistemas. Tras ser comparada con otras herramientas de control ya existentes, se observa que al implementar la metodología propuesta se obtienen buenos resultados y demuestra su factibilidad.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El desarrollo tecnológico actual ha sido resultado, en gran medida, de la necesidad de comunicación del hombre con todos los elementos que lo rodean. Los avances tecnológicos aumentan exponencialmente desde el surgimiento del transistor, a finales de la primera mitad del siglo XX, y han revolucionado tanto la forma de vida como la percepción del mundo para los seres humanos. En este entorno surgen los sistemas SCADA (*Supervisory Control and Data Acquisition*) como una excelente vía para lograr un mejor control y supervisión. Estos sistemas son utilizados cada día más en el control y supervisión de procesos, debido a su versatilidad y a las variadas facilidades que proporcionan (Valdés Seoane, 2009).

El software Movicon es uno de los sistemas SCADA con mayor divulgación en Cuba (Valdés Seoane, 2009). Presenta una alta flexibilidad y simplicidad de uso. Movicon es una herramienta general para el desarrollo de softwares SCADA para prácticamente cualquier aplicación de supervisión. En funciones de control, su utilización se ve limitada al no contar con módulos o aplicaciones de control inteligente.

El control inteligente surge como soporte para la sistematización de las tareas complejas realizadas por el operador, con el fin de suplir sus posibles carencias producidas, entre otros motivos, por la sobrecarga de información o la fatiga. Se trata de automatizar en la medida en que sea posible, el análisis de los datos, la detección de fallos, el diagnóstico de los mismos y la toma de decisiones o propuesta de acciones concretas. Es de gran importancia sobre todo cuando la planta a controlar es compleja (Colomer et al., 2000). El control difuso o fuzzy es una de las estrategias de control inteligente más implementadas. Permite definir funciones de control mediante conjuntos de reglas tecnológicas, lo que posibilita modelar de forma relativamente simple el conocimiento de los expertos en supervisión y control de procesos, algo particularmente útil en aplicaciones con plantas problemáticas.

Actualmente no hay una sistematización del conocimiento de cómo desarrollar aplicaciones de control fuzzy en Movicon. No existen procedimientos ni herramientas para el desarrollo de dichas aplicaciones, y por lo tanto se carece de una metodología. Estos elementos traen consigo que los diseñadores de los sistemas automatizados que utilizan el SCADA Movicon, enfrentan el problema de no poder incluir las estrategias de control fuzzy como parte de las soluciones. Las plantas con una dinámica compleja tienen que ser controladas a partir de las estrategias convencionales lo que constituye un trabajo mucho más complicado. Los operadores de planta no se benefician de las ventajas del control fuzzy como estrategia más legible.

La situación anteriormente descrita dio lugar al problema científico siguiente:

¿Cómo desarrollar aplicaciones de control fuzzy en Movicon que permitan un uso más amplio y efectivo de este software?

En consecuencia la hipótesis de la investigación es:

La existencia de una metodología para el desarrollo de aplicaciones de control fuzzy en el software Movicon facilitaría un uso más amplio y efectivo del mismo.

El objetivo general es:

Proponer una metodología para el desarrollo de aplicaciones de control fuzzy en Movicon que permita aprovechar las potencialidades de este tipo de control inteligente y agregar valor al software.

Los objetivos específicos a desarrollar son:

- Analizar la bibliografía existente sobre software SCADA y desarrollo de aplicaciones de control o supervisión fuzzy.
- Determinar en el software Movicon las posibilidades que posee para el desarrollo de aplicaciones.
- Fundamentar una metodología para desarrollar aplicaciones de control fuzzy en Movicon.
- Aplicar la metodología en el control de un proceso específico.
- Validar la metodología para el desarrollo de aplicaciones de control fuzzy en

Movicon.

Con este proyecto se contribuye a la sistematización del conocimiento sobre control inteligente, particularmente control fuzzy; se utiliza la plataforma SCADA/HMI Movicon, la cual es una herramienta muy factible en procesos que requieran de visualización, control y adquisición de datos. Se aportan herramientas desarrolladas específicamente para aplicaciones de control fuzzy. La metodología que se propone facilita a los diseñadores desarrollar aplicaciones de control fuzzy.

Organización del informe

El informe incluye tres capítulos, además de las conclusiones, recomendaciones y referencias bibliográficas. Los temas que se abordan en cada capítulo se estructuran de la forma siguiente:

Capítulo I: En este capítulo se realiza un análisis de la bibliografía existente sobre sistemas SCADA e implementación de control fuzzy en dichos sistemas, conjuntamente con la presentación de los principales conceptos y definiciones que se tratan en este trabajo.

Capítulo II: En esta sección se expone un análisis de las diferentes posibilidades que posee el Movicon para implementar control fuzzy, así como el diseño de herramientas necesarias e imprescindibles para el desarrollo de las aplicaciones. Se fundamenta la metodología para el desarrollo de aplicaciones de control fuzzy en Movicon.

Capítulo III: Finalmente este capítulo muestra un análisis de los resultados que arrojan las diferentes comparaciones que se realizan entre el sistema desarrollado con la metodología propuesta y el sistema implementado por ejemplo en el Simulink de Matlab al utilizar la herramienta de lógica difusa con que cuenta dicho software.

CAPÍTULO 1. Control fuzzy en sistemas SCADA

Con el fin de lograr un mayor entendimiento de lo que significa y la importancia en la actualidad de los sistemas SCADA y el control inteligente, particularmente el control fuzzy, en este capítulo se realiza un análisis de los conceptos asociados a estas temáticas.

Se parte de la descripción de las generalidades de los sistemas SCADA, y se señalan algunas de las principales características del software Movicon así como su utilización en Cuba. Posteriormente se trata el control inteligente; un análisis de sus principales técnicas es llevado a cabo, se enmarcan las bondades de la utilización del control fuzzy. Finalmente se exponen antecedentes de la implementación del control fuzzy en entornos de desarrollo SCADA y se arriba a algunas consideraciones finales.

1.1 Sistemas SCADA

Los sistemas SCADA utilizan la computadora y las tecnologías de comunicación para automatizar la supervisión y control de procesos industriales. Estos sistemas son parte integral de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos ya que pueden recoger la información de una gran cantidad de fuentes muy rápidamente, y la presentan a un operador en una forma amigable. Los sistemas SCADA mejoran la eficacia del proceso de supervisión y control además de proporcionar la información oportuna para poder tomar decisiones operacionales apropiadas (Chacon et al., 2001).

Al decir de (Hentea, 2008) los primeros SCADA eran simplemente sistemas de telemetría que proporcionaban reportes periódicos de las condiciones de campo y vigilaban las señales que representaban medidas y/o condiciones de estado en ubicaciones remotas de campo. Estos sistemas ofrecían capacidades muy simples de monitorización y control; la visión del

operador en el proceso estaba basada en los contadores y las lámparas detrás de paneles llenos de indicadores. Mientras la tecnología se desarrollaba, las computadoras asumieron el papel de manejar la recolección de datos, disponer de comandos de control, y presentar la información sobre una pantalla de Tubo de Rayos Catódicos (*CRT – Cathode Ray Tube*). Las computadoras agregaron la capacidad de programar el sistema para realizar funciones de control más complejas (Chacon et al., 2001, Chavarría Meza, 2007).

Actualmente, los proveedores de SCADA diseñan sistemas pensados para resolver las necesidades de muchas industrias. Existen módulos de software específicos para determinadas industrias que proporcionan las capacidades requeridas comúnmente. Se pueden encontrar softwares SCADA comercialmente disponibles cuya aplicación se observa en industrias tales como: procesamiento de papel y celulosa, industrias de aceite y gas, hidroeléctricas, gerenciamiento, provisión de agua y control de fluidos.

Los sistemas SCADA son parte integral de la estructura de gerenciamiento de la información corporativa. Estos sistemas ya no son vistos por la gerencia simplemente como herramientas operacionales, sino como un recurso importante de información. En este papel sirven como centro de responsabilidad operacional, pero también proporcionan datos a los sistemas y usuarios fuera del ambiente del centro de control que dependen de la información oportuna en la cual basan sus decisiones económicas cotidianas (Izaguirre, 2002, Chacon et al., 2001).

Para alcanzar un nivel aceptable de tolerancia de fallas con estos sistemas, es común tener computadoras SCADA redundantes operando en paralelo en el centro primario del control, y un sistema de reserva del mismo situado en un área geográficamente distante.

Esta arquitectura proporciona la transferencia automática de la responsabilidad del control de cualquier computadora que pueda llegar a ser inasequible por cualquier razón, a una computadora de reserva en línea, sin interrupción significativa de las operaciones.

1.1.1 Elementos del sistema SCADA

Un sistema SCADA está conformado tanto por elementos de hardware como por elementos de software, los cuales posibilitan el funcionamiento íntegro del sistema (Chavarría Meza, 2007, Izaguirre, 2002).

Elementos de hardware:

Unidad Terminal Remota (*RTU – Remote Terminal Unit*):

RTU es un dispositivo instalado en una posición remota que obtiene datos, los descifra en un formato y los transmite a una unidad terminal maestra (MTU). La RTU también recoge la información del dispositivo principal y se encarga de ejecutar las órdenes procedentes de este. La RTU es capaz de ejecutar programas simples autónomos sin la participación de la MTU del sistema SCADA, para simplificar el despliegue y proporcionar la redundancia por razones de seguridad.

Interfaz Operador – Máquinas (*HMI – Human Machine Interface*):

Dispositivo que permite la interacción del ser humano con los medios tecnológicos implementados. Contiene el entorno visual que brinda el sistema para que el operador se adapte al proceso desarrollado por la planta. Esta interfaz incluye generalmente los controles donde el operador se puede interconectar con el sistema SCADA. La HMI es una manera fácil de estandarizar la supervisión de las RTU´s múltiples o de los PLC´s (Controlador Lógico Programable).

Unidad Terminal Maestra (*MTU – Master Terminal Unit*):

El término MTU se refiere a los servidores y el software responsable de comunicarse con los equipos del campo tales como RTU´s y PLC´s. En un sistema SCADA pequeño, la Unidad Terminal Maestra puede estar en un sola computadora pero en un sistema SCADA a gran escala, la MTU puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres.

Esta terminal ejecuta las acciones de mando programadas en base a los valores actuales de las variables medidas. La programación se realiza por medio de bloques de programa en lenguajes de alto nivel como *C* y *Visual Basic*. También se encarga del almacenamiento y procesamiento ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

Sistema de Comunicaciones:

Se encarga de la transferencia de información del punto donde se realizan las operaciones, hasta el punto donde se supervisa y controla el proceso. Lo conforman los transmisores, receptores y medios de comunicación.

Transductores:

Un transductor es un dispositivo capaz de transformar la naturaleza de la variable física del proceso en una señal modificada. Generalmente el transductor es un conversor de una magnitud no eléctrica en una eléctrica. Su calibración es muy importante para que no haya problema con la confusión de los valores de los datos.

Elementos de software:

Configuración:

Permite al usuario definir el ambiente de trabajo del SCADA, adaptándolo a la aplicación particular que se desea desarrollar.

Interface gráfica del operador:

Proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos, gráficos almacenados en la computadora del proceso y generados en el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.

Módulo de proceso:

Ejecuta las acciones de mando pre-programadas desde los valores actuales de las variables leídas.

Gestión y archivo de datos:

Se encarga del almacenamiento y procesamiento ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

Comunicaciones:

Se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y los demás elementos informáticos de gestión.

1.1.2 Funciones principales de los sistemas SCADA

Dentro de las funciones principales realizadas por los sistemas SCADA como se muestra en (Izaguirre, 2002, Chacón Morales, 2012, Chavarría Meza, 2007) están las siguientes:

- Supervisión

  El operador podrá observar desde el monitor la evolución de las variables de control, como cambios que se produzcan en la operación diaria de la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.

- Control

  Mediante el sistema se puede activar o desactivar los equipos remotamente; por ejemplo abrir válvulas, activar interruptores y/o prender motores, de manera automática y también manual. El operador puede ejecutar acciones de control y podrá modificar la evolución del proceso en situaciones irregulares que se generen.

- Adquisición de datos

  Recolectar, procesar, almacenar y mostrar la información recibida en forma continua a partir de los equipos de campo.

- Generación de reportes

  Con los datos adquiridos se pueden generar representaciones gráficas, predicciones, control estadístico, gestión de la producción, gestión administrativa y financiera, etc.

- Representación de señales de alarma

  A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una condición perjudicial o fuera de lo aceptable, estas pueden ser tanto visuales como sonoras.

1.1.3 Ventajas y desventajas de los sistemas SCADA respecto a otras estructuras de sistemas de automatización

Los sistemas SCADA tienen una serie de **ventajas** que contribuyen a su amplia utilización (Izaguirre, 2002, Chavarría Meza, 2007):

- Mejora en la productividad del personal operador, instrumentista y de mantenimiento, así como una operación con mayor seguridad.
- Menor riesgo de contaminación ambiental.

- Reduce costos; menor costo operativo, debido al menor costo de operación y mantenimiento.
- Reducción del personal.
- Reduce requisitos de control futuros.
- Mejora en el factor de servicio de los equipos e instrumentos.
- Reducción de la incidencia de fallas.
- Modernización de sistemas de control obsoletos, o basados exclusivamente en hardware.
- Disponibilidad de la información real para los distintos niveles de la empresa.

Al decir de (Izaguirre, 2002) estos sistemas también conllevan **desventajas** tales como:

- Se requiere de una red industrial fiable, pues resultaría crítico no contar con la misma.
- Alto costo inicial, por concepto de adquisición de los equipos e implantación del sistema acorde a las necesidades y requisitos exigidos.
- Se requiere además realizar gastos en conexión a la red de datos.

1.1.4 Caracterización de los principales softwares SCADA

Según (Valdés Seoane, 2009) el mercado de tecnologías para la informatización industrial presenta un gran número de fabricantes y suministradores de sistemas y entornos de desarrollo SCADA. Los entornos de desarrollo SCADA se orientan en dos variantes:

1. Aplicaciones sectoriales en las que el fabricante es el proveedor y en las que acostumbra a existir un importante acomodamiento a determinados sectores industriales.
2. Aplicaciones de tipo general donde es determinante la facilidad de configuración por parte del proveedor, generalmente distinto al fabricante.

Como característica general hay que destacar la madurez de este tipo de productos. La adopción de estándares de comunicación, tanto para la comunicación entre dispositivos como entre aplicaciones, y la disponibilidad de módulos especializados para el análisis de datos obtenidos en tiempo real así como para la comunicación de los usuarios con el sistema a

través de Internet refieren el desarrollo de estos sistemas en los últimos años. El principal argumento de venta de este tipo de productos plantea la posibilidad de estos sistemas de incorporar las funciones de gestión de producción y/o de comunicación/integración con aplicaciones de planificación y gestión a nivel de empresa. Esto para alcanzar objetivos de control de calidad, mejora de la productividad y toma de decisiones a partir de datos en tiempo real. Y por lo tanto, estas funcionalidades son estratégicas para fabricantes y proveedores que orientan sus políticas de soporte al cliente basadas principalmente en este tipo de funciones.

Las ofertas actuales del mercado tienen como tendencia la utilización de *Windows XP Professional SP2* y *Windows Vista (Business, Enterprise, Ultimate)* como Sistemas Operativos para estaciones monopuesto y estaciones cliente; *Windows Server 2003 R2 SP2* para los servidores de datos y *Windows CE* para sistemas empotrados y sistemas móviles. Varios proveedores suministran entornos soportados sobre *UNIX* y sus versiones *GUI* (*Graphical User Interface*), orientadas para el desarrollo de grandes aplicaciones de ingeniería que requieran elevada potencia de cálculo. En general se mantiene una compatibilidad con los Sistemas Operativos y las tecnologías de *Microsoft* como *DNA* (*Distributed interNet Application*) y *.NET*. Por otro lado, es muy común la posibilidad de configurar las aplicaciones programando en lenguajes estándares como *VBA* (*Visual Basic for Applications*), ó *VB* y *C++*; y se ha generalizado la existencia del lenguaje *VBA* integrado (*embedded*) en el entorno SCADA, lo que permite acceder a los objetos creados en la aplicación SCADA, así como los métodos asociados a cada objeto, y sus propiedades, agilizando el desarrollo y permitiendo obtener la máxima potencia del *VBA*.

Se ha extendido prácticamente en todos los entornos comerciales la utilización de *OPC (OLE for Process Control)* como estándar para las comunicaciones entre aplicaciones y con dispositivos tales como los PLCs. En algunos casos el SCADA solo funciona como cliente *OPC* y en otros puede ser cliente y servidor *OPC*. Resulta común encontrar servidores *WEB* integrados, con tecnología basada en la arquitectura *DNA* de *Microsoft*, que permiten la conexión remota de clientes a través de Internet, donde algunos solo ofrecen la posibilidad de visualizar información poniendo por objeción el problema de la seguridad, mientras que otros ofrecen la posibilidad de tomar el control total de la planta.

En gran parte de los SCADA comerciales se incorporan los protocolos *TCP-IP* para el funcionamiento en red del sistema de supervisión. Además, existe una tendencia a la utilización del estándar *XML* (*Extensible Markup Language*) para el intercambio de información por la red, lo que permite comunicaciones más eficientes. Por otro lado, los controles *ActiveX* son soportados por la mayoría de los entornos de desarrollo. Las aplicaciones clientes de una determinada solución SCADA son en muchos casos contenedoras de objetos *ActiveX*. Frecuentemente se encuentran productos que permiten la utilización de módulos independientes, específicos para determinadas funcionalidades, como por ejemplo, módulos para procesamientos estadísticos (*SPC - Statistical Process Control*) y módulos para la detección y diagnóstico de fallos; que pueden ser desarrollados por el propio fabricante del entorno o por terceros.

Existe una tendencia a la implementación de Arquitecturas Orientadas a Servicios (*SOA - Service Oriented Architecture),* lo que constituye un replanteamiento de los productos e incorpora numerosas ventajas como robustez, facilidad de mantenimiento, independencia de las tecnologías y rapidez.

Dicha arquitectura está basada en servicios de aplicación que se comunican entre sí, y en muchos casos aunque no necesariamente, estos servicios están ubicados en diferentes dispositivos y la comunicación entre servicios se realiza a través de Internet (Servicios Web). Cada servicio ejecuta una función a partir de llamadas recibidas de otros servicios, y como resultado, devuelve una respuesta. La mayor ventaja radica en que la tecnología concreta para brindar el servicio no forma parte de su definición, por lo que una aplicación diseñada como un conjunto de servicios interrelacionados es independiente de la tecnología que se utilice para su ejecución.

En la Tabla 1.1 se muestra un conjunto de los principales entornos de desarrollo SCADA actuales (Valdés Seoane, 2009).

Tabla 1.1 Principales entornos de desarrollo SCADA

Entorno SCADA	**Fabricante**	**País**
Cube Industrial Frameworks	Siemens PS-SH	Alemania

Frameworks CX	Omron	Japón
Supervisor Delta V	Fisher Rosemount	EE.UU
Factory Suite	Wonderware	EE.UU
LabView	National Instruments	EE.UU
Lookout	National Instruments	EE.UU
Monitor Pro	Schneider Electric	EE.UU
Plant Scape	Honeywell Asia Pacific	EE.UU
Movicon	Progea	Italia
iFIX	Intellution	EE.UU
Genesis32	Iconics	EE.UU
I/A	Foxboro	EE.UU
Intouch	Wonderware	EE.UU
RS3	Fisher Rosemount	EE.UU
RSView32	Rockwell Software	EE.UU
Statt Graph 5000	ABB	Inglaterra
WIN CC	Siemens	Alemania
Wiz Factory	Emation	Alemania

Movicon

El software Movicon (*Monitoring, Vision and Control*) es uno de estos sistemas SCADA que destaca su desarrollo y progresión en los últimos años. Actualmente se comercializa la versión Movicon 11.4. Una valoración sobre sus características y prestaciones se realiza a continuación (Suazo Crespo, 2014, Valdés Seoane, 2009):

Simplicidad de uso:

Esta dada gracias al espacio de trabajo cómodo e intuitivo para los usuarios que prefieren Windows como Sistema Operativo, los asistentes integrados para realizar funciones comunes y las herramientas para la importación de variables.

Escalabilidad:

Movicon provee un único entorno de desarrollo para aplicaciones desde *Windows CE* hasta *Windows XP*, por lo que al adquirir un solo programa los desarrolladores pueden implementar pequeñas aplicaciones para terminales *HMI*, así como aplicaciones de mediana y gran complejidad para plantas de procesamiento.

Sistema abierto:

Movicon se diseña como sistema abierto que utiliza completamente los estándares de *Microsoft* tales como:

- *ActiveX* permite insertar los objetos de los terceros.
- *ODBC* (*Open Database Connectivity*) se utiliza para la gerencia de la base de datos.
- *VBA* para la programación en elementos tales como sinópticos y scripts.
- *OPC* para la conectividad y que importe y que exporte los datos o los símbolos.
- *Windows API* (*Application Program Interface*) para garantizar la extensión de sistema a través del *DLL*.
- Movicon es una plataforma basada en *XML*, por tanto, los proyectos realizados sobre este entorno de desarrollo son archivos de texto que pueden ser abiertos y editados en cualquier editor *XML*.

Seguridad de los datos:

Asegurada por el uso de algoritmos de encriptación de 128 bits de los archivos de texto y las funciones de administración de usuarios y contraseñas que aseguran control de acceso por niveles y/o áreas.

Uso de tecnologías estandarizadas:

Movicon está completamente basado en estándares: *XML*, *ODBC*, *OPC*, *VBA*, *SOAP (Simple Object Access Protocol)*, Servicios Web, *TCP-IP (Transport Control Protocol-Internet Protocol)*, *UDP (User Datagram Protocol)*, *HTTP* (*Hypertext Transfer Protocol*), *SQL (Structured Query Language)*; que garantizan transparencia, fácil acceso a los datos y seguridad a la inversión del desarrollador en el producto.

Alto rendimiento:

Traducido en un alto rendimiento del *Kernel* (módulo principal o núcleo). Esta mejora se debe a la renovación del concepto de "*thread pooling"* (asociación de hilos) y el nuevo mecanismo de gráficos basado en *SVG* (Gráficos Vectoriales Escalables).

Alta conectividad:

El software ofrece *drivers* de comunicación que incluyen funcionalidades de importación de variables (*tags*), conectividad remota vía modem, conceptos de multiestación para protocolos punto a punto y pruebas de cableado inmediatas. Las bibliotecas de *drivers* se incluyen sin costos adicionales.

Eficiente sistema de redes:

La administración integrada del sistema de redes se basa en las tecnologías multiplataforma *SOAP* y *SOA*. Además soporta los protocolos *UDP*, *HTTP*, *TCP-IP* y conexiones remotas automáticas vía *RAS* (*Remote Access Services*). Ofrece tecnologías de Servicios Web para las cuales la distribución de la información puede soportar redes públicas, como Internet, que garantizan la seguridad sin ser perjudicial para ningún cortafuego.

Arquitectura "*Web-enabled*" (habilitada para la *Web*):

Se integra la tecnología *JAVA* con *XML*, *SVG* y las tecnologías de Servicios Web. Así se permite acceso al servidor usando un navegador de internet sobre cualquier sistema

operativo. Progea asegura posibilidad de conexión de usuarios múltiples, bidireccionalidad, alto rendimiento y seguridad.

1.1.5 Aplicaciones en el ámbito nacional del SCADA Movicon

En Cuba se han implementado y están en ejecución varios proyectos que utilizan sistemas SCADA implementados con Movicon, algunos de ellos son (Montejo Rodríguez, 2014, Valdés Seoane, 2009):

- Imprenta Federico Engels, La Habana. Supervisión de los sistemas de iluminación y clima.
- Central Termoeléctrica Otto Parellada, La Habana. Sistema de monitorización de variables de campo en las calderas de la central.
- Universidad de las Ciencias Informáticas (UCI), La Habana. Supervisión de los sistemas de iluminación y clima.
- Hotel La Estrella 1, Villa Clara. Gestión energética.
- Hospital Ortopédico Fructuoso Rodríguez, La Habana. Supervisión y control de los sistemas de alumbrado, clima, bombeo de agua, fan-coil y extractores.

1.2 Control inteligente

El incremento de las demandas tecnológicas en nuestros tiempos, genera sistemas muy complejos que requieren controladores altamente sofisticados para asegurar alto desempeño dentro de condiciones adversas. Estas y otras condiciones de control no se pueden cumplir con controladores convencionales, debido principalmente a la falta de conocimiento preciso acerca del proceso que se desea controlar. La adquisición de conocimiento adecuado del sistema en ocasiones es problemática o impráctica debido a la complejidad del sistema y al hecho de que la estructura y los parámetros en muchos sistemas cambian de manera significativa e impredecible con el tiempo. Es bajo estas condiciones en donde se utilizan las técnicas del control inteligente (Cotero Ochoa, 2002).

Según (Colomer et al., 2000) el control inteligente es una generalización del concepto de control y se puede ver como un campo dentro de la disciplina del control. El control inteligente es la disciplina donde los métodos de control se desarrollan para emular algunas

características importantes del ser humano. Estas características incluyen adaptación y aprendizaje, planeación bajo gran incertidumbre y el trabajo con gran cantidad de datos.

Las metodologías de control inteligente son aplicadas a la robótica, las comunicaciones, la manufactura, el control de tráfico, por mencionar algunas pocas. Las principales áreas donde se realiza trabajo alrededor del control inteligente según (Cotero Ochoa, 2002) son: redes neuronales, control difuso, algoritmos genéticos, sistemas de planeación, sistemas expertos, y sistemas híbridos.

1.2.1 Principales técnicas de control inteligente

Sistemas expertos

Al decir de (Colomer et al., 2000) pueden encontrarse definiciones muy variadas de Sistemas Expertos (SE). Algunas basadas en su función, otras en la estructura y otras en componentes funcionales y estructurales. A grandes rasgos puede decirse que un SE sirve para codificar conocimiento humano en términos de experiencia, razonamiento aproximado, imprecisión, analogía, razonamiento por defecto y aprendizaje. Específicamente, se trata de representar el conocimiento experto en un sistema basado en reglas tecnológicas para tener una computadora que responda como lo haría el experto humano.

En resumen, puede decirse que un sistema experto contiene un motor de inferencia y una base de conocimiento, esta última a su vez se compone de una base de reglas y una base de hechos.

Una base de reglas es un conjunto de reglas del tipo:

IF [sucede_algo] *THEN* [decide/concluye_algo] *CERTAINTY* [valoración]

La parte *IF* ... *THEN* de la regla, es decir [sucede_algo], se llama premisa de la regla o bien precedente. La parte *THEN* ... *CERTAINTY*, es decir [decide/concluye_algo], se llama conclusión de la regla. La *CERTAINTY*, [valoración], significa la seguridad con que el experto de procesos/control (operario o ingeniero) realiza esta afirmación que está convertida en una regla del SE.

Una base de hechos es el conjunto de evidencias junto con sus certezas asociadas. Por ejemplo, una variable medida repercute en un hecho como “Hoy llueve” o “Posible ALARMA708, con certeza 30%”.

El motor de inferencia se encarga de recorrer las reglas e inspeccionar si las puede aplicar. Es decir, se encarga de ejecutar el razonamiento. El razonamiento consiste en aplicar una base de reglas a una base de hechos para obtener nuevas conclusiones.

Redes neuronales

Las redes neuronales artificiales (RNA) pueden definirse como sistemas de computación constituidos por un gran número de elementos de proceso simples y muy interconectados. La información se procesa como respuesta a entradas externas teniendo en consideración el estado interno de los elementos (Sánchez and Alanis, 2006).

Consisten en unidades de procesamiento densamente interconectadas, llamadas neuronas por su semejanza funcional con las neuronas biológicas. Las unidades de procesamiento reciben, procesan y transmiten señales, tal como las neuronas biológicas.

A diferencia del control fuzzy, la utilización de las redes neuronales está dada sobre todo en tareas de identificación. Este es el caso donde no se tiene conocimiento previo, es decir, experiencia, experticia, heurística; pero se tienen medidas de datos a partir de observaciones. Se establece el concepto de aprendizaje a partir de los datos.

Una neurona es una unidad básica de procesamiento de información, la cual es la base para el diseño de redes neuronales artificiales. En la Figura 1.1 se muestra el modelo respectivo.

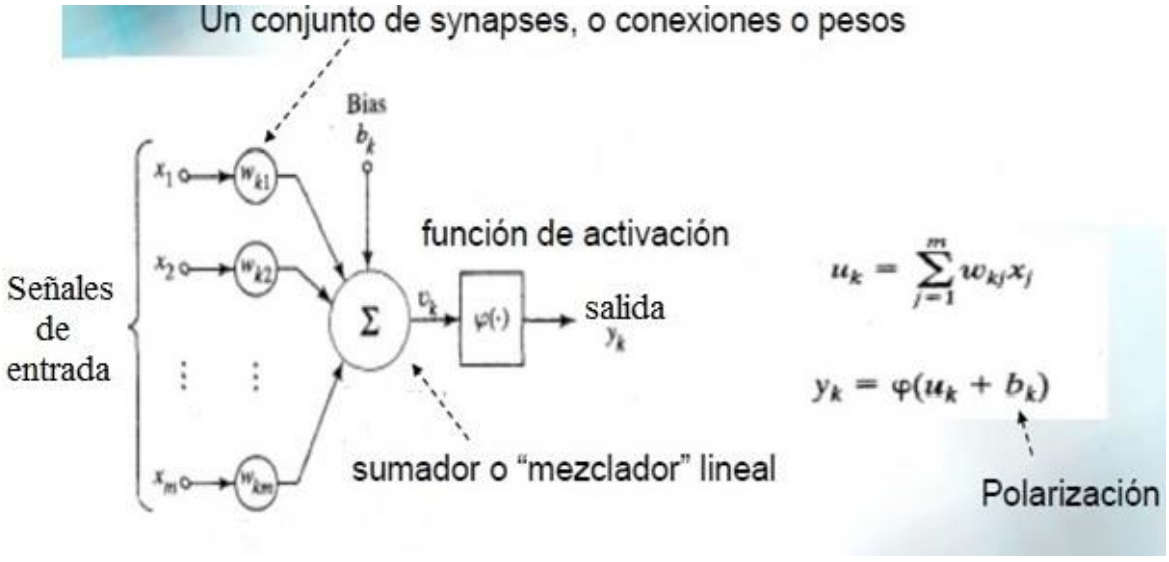

Figura 1.1 Modelo de una neurona

Los nueve componentes principales del funcionamiento de las Redes Neuronales Artificiales son (Sánchez and Alanis, 2006, Colomer et al., 2000):

1) Un conjunto de unidades de procesamiento (neuronas).
2) Un estado de activación (variable de estado).
3) Una función de salida para cada unidad.
4) Un conjunto de conexiones (patrón de conectividad).
5) Un conjunto de reglas de propagación para propagar las señales de salida a través de la RNA.
6) Una regla de combinación.
7) Una regla de activación.
8) Una regla de modificación.
9) Un ambiente en el cual opera la RNA.

Control fuzzy

Al decir de (Kouro and Musalem, 2002) la lógica fuzzy o difusa puede ser descrita como un sistema interpretativo, en el cual los objetos o elementos son relacionados con conjuntos de fronteras no nítidamente definidas, otorgándoles un grado de pertenencia relativa o graduada y no estricta como es de costumbre en la lógica tradicional. En un sentido más amplio se podría decir que existe una especie de interpolación entre una frontera y otra, o bien, entre un conjunto y otro.

Lo anterior permite incorporar sentencias del lenguaje común, las que se caracterizan por ser un tanto indefinidas, para interpretar el estado de las variables de cierto proceso, asignándoles en cada momento un grado de pertenencia a estos conjuntos difusos. Esta interpretación puede ser fácilmente relacionada mediante operadores lógicos tradicionales con ciertas medidas de acción, también de naturaleza no exacta, que son diseñadas de tal manera que produzcan un cambio deseado en las variables de interés. En resumen, se puede diseñar un controlador, que interprete en forma intuitiva y no numéricamente exacta, el estado de ciertas variables, y en base a ello deduzca en forma lógica una actuación posible que permita llevar la variable al estado deseado.

La Figura 1.2 muestra los elementos que componen un controlador fuzzy. Se detalla cada etapa para un mayor entendimiento a continuación:

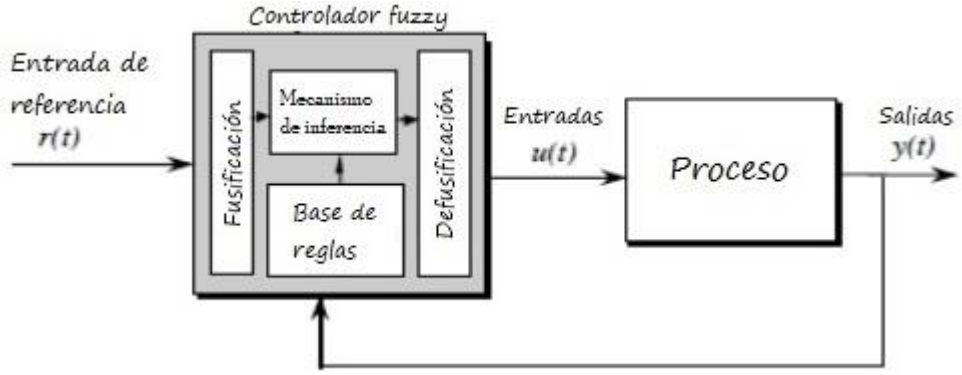

Figura 1.2 Controlador fuzzy

1. Fusificación (*Fuzzification*).

Esta es la primera parte del proceso, donde luego del procedimiento convencional de adquirir los valores de las variables del sistema y calcular otras, se procede a calcular el grado de pertenencia de tales variables a todos los posibles conjuntos fuzzy o funciones de pertenencia que se le han asignado (Passino, 1998, Gómez, 2008).

El objetivo de las funciones de pertenencia es determinar en cada período en que rango se encuentra la variable en cuestión. Estas funciones pueden ser de diferentes tipos: trapezoidales, gaussianas, de picos, triangulares. El tipo a utilizar depende de la aplicación y el diseñador, es decir, para una misma aplicación se pueden utilizar diferentes formas en dependencia del diseñador, este proceso es más subjetivo que objetivo.

2. Base de reglas (*Rule-base*).

A partir de las funciones de pertenencia se pueden crear las reglas difusas.

Si *premisa* **Entonces** *consecuencia*

La cantidad de reglas depende del número de variables de entrada y la cantidad de funciones de pertenencia de cada variable de entrada.

3. Mecanismo de inferencia (*Inference mechanism*).

El mecanismo de inferencia determina en cada momento que reglas están activas a partir de comparar las premisas con las entradas del controlador. También calcula la certeza de activación para cada regla.

Se da el caso en numerosas ocasiones de que exista más de una premisa en una regla y como cada premisa tiene una certeza asociada es necesario utilizar alguna herramienta para establecer la certeza de la regla. Se pueden aplicar entonces cualquiera de las funciones que se ejemplifican a continuación:

- Mínimo: Se define $\mu_{premisa} = \min\{0.5 , 0.25\} = 0.25$
- Producto: Se define $\mu_{premisa} = (0.5)(0.25) = 0.125$
- Máximo: Se define $\mu_{premisa} = máx\{0.5 , 0.25\} = 0.5$

4. Defusificación (*Defuzzification*).

El proceso de defusificación concluye un valor concreto que será el mando a aplicar. Existen diferentes técnicas de defusificación, ejemplo de ellas son las siguientes:

- Centro de gravedad

$$M = \frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i}$$

- Media de centros

$$M = \frac{\sum_i b_i \mu_i}{\sum_i \mu_i}$$

Donde:

M = mando

b_i = valor del centro de la función de pertenencia de la variable de salida

μ_i = certeza del conjunto borroso

La utilización de cada una depende de la aplicación en particular, ambas son comúnmente implementadas por su sencillez y efectividad.

Ventajas en la utilización del control fuzzy

Después de una breve descripción del control difuso se añaden una serie de beneficios que trae consigo utilizar esta técnica de control.

- La lógica difusa tiene la ventaja de incorporar el lenguaje común al diseño de sistemas de control, hecho que se torna muy importante a la hora de incluir el conocimiento empírico de los operadores de procesos.
- Este método no requiere de un modelo riguroso de la planta a controlar.
- Se facilita en gran medida el proceso de diseño del controlador, sobre todo cuando la determinación cuantitativa de los parámetros del sistema se hace compleja. Tal es el caso en sistemas no lineales.

1.2.2 Antecedentes de implementación de control fuzzy en sistemas SCADA

Se seleccionaron para el análisis tres aplicaciones:

1. **Implementación de control de velocidad de un motor DC. Se utiliza lógica difusa en la plataforma de Labview.**

En este trabajo se hace uso del sistema SCADA Labview de *National Instruments*. Se aprovecha que este sistema tiene ya implementado un módulo de diseño de controlador fuzzy, a diferencia de Movicon. Al decir de (Agredo Fajardo, 2012) LabVIEW utiliza el lenguaje G, donde la G simboliza el lenguaje gráfico para hacer diseño, control y pruebas mediante la existencia de los kits de herramientas o mejor llamados *toolkits* para facilitar la programación. El módulo de LabVIEW *Fuzzy Logic Controller Design* presenta de una manera concisa el diseño de las distintas etapas del controlador.

El diagrama de bloques de la Figura 1.3 representa el proceso del lazo de control.

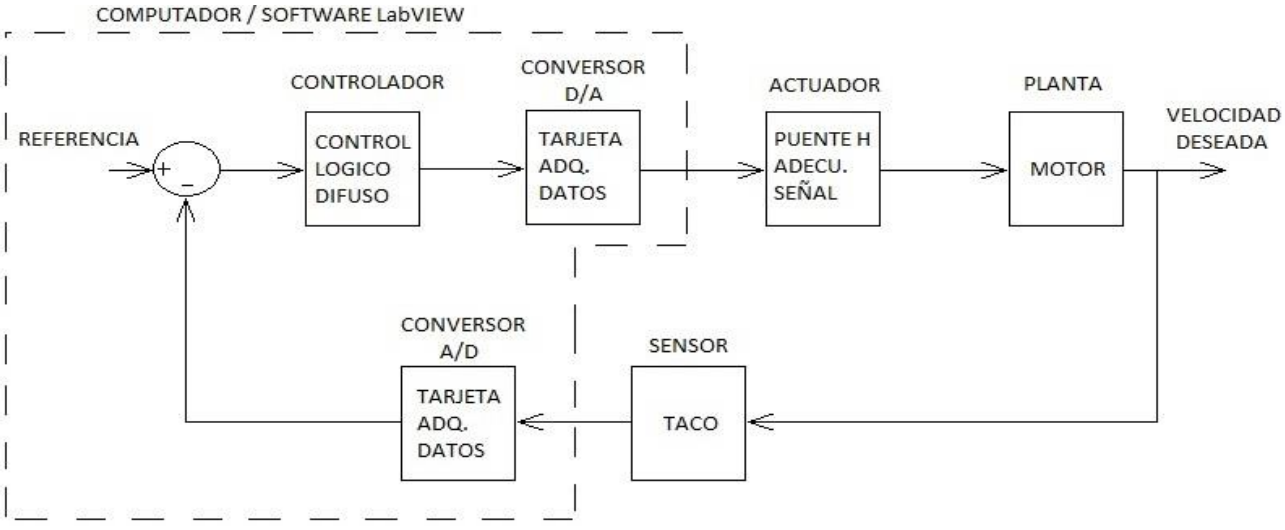

Figura 1.3 Lazo de control

Como se muestra en el esquema anterior el algoritmo de control se ejecuta en el propio software SCADA. Se prescinde entonces de otro medio de cómputo que no sea la propia PC donde actúa el software. En este caso, para controlar la velocidad del motor se hace uso de una tarjeta de adquisición de datos, un puente H como elemento actuador, un tacómetro como elemento sensor y la PC como medio de cómputo. También se realiza una interfaz para el control y supervisión del proceso.

Se puede apreciar que este método es más apropiado para procesos sin una gran cantidad de lazos de control, con un sistema de automatización con mayor integración. Esto debido a que la propia PC donde estaría el sistema SCADA procesaría también el algoritmo del sistema de control.

2. **Desarrollo de un sistema de control y supervisión para un proceso de transporte de piezas basado en lógica difusa utilizando la plataforma de InTouch.**

En esta investigación se complementan las ventajas de los PLC con las ventajas de los sistemas SCADA, en este caso el software InTouch de *Invensys Wonderware*. El autor opta por implementar el algoritmo de lógica difusa a partir de lenguaje de texto estructurado en un PLC; y supervisarlo y controlarlo a través del software InTouch (Smoczek and Sznytko, 2008).

En la plataforma InTouch se crean diferentes interfaces que dan la posibilidad de insertar o actualizar los parámetros del controlador fuzzy desarrollado en el PLC. Estas interfaces son diseñadas en concordancia con las metodologías de diseño de controladores difusos.

Esta opción es la indicada para procesos con mayor cantidad de lazos de control, donde el sistema de automatización es un poco más distribuido. El procesamiento de los algoritmos de control estaría en manos de otros elementos de cómputo tales como PLC y controladores digitales, y no se recargaría a la PC principal.

3. **Implementación en tiempo real de controladores PID y de lógica difusa para el control de nivel de un sistema de tanques utilizando LabView (Mohamed Nasir, 2011).**

El proyecto demuestra cómo se puede implementar el control de nivel en un sistema de tanques, teniendo en cuenta perturbaciones, utilizando tanto PID como control fuzzy. Se realizan varias pruebas con diferentes controladores clásicos (P, PD, PI, PID), y un análisis de las estrategias de control posibles. Posteriormente se decide desarrollar una cascada siendo el PID el controlador del lazo interno para atenuar una perturbación y el controlador difuso en el lazo externo. El controlador difuso se diseña utilizando el módulo de LabView *Fuzzy Logic Controller Design*. El sistema se implementa en Labview y se supervisa y controla desde este.

Esta aplicación demuestra la fiabilidad del control fuzzy para la automatización de procesos. Se observa nuevamente la complementación entre los sistemas SCADA y las estrategias de control inteligente.

1.3 Consideraciones finales del capítulo

Los altos requisitos de los procesos en cuanto a necesidad de gestión, supervisión y realización de tareas de diagnóstico hacen de los sistemas SCADA elementos sumamente importantes hoy en día. El software Movicon, ampliamente utilizado en Cuba, a pesar de tener características que lo convierten en una muy buena opción, necesita del desarrollo de aplicaciones de control fuzzy que le aumente significativamente las prestaciones. Es un hecho además que el control inteligente y específicamente el control fuzzy se hace en muchas ocasiones indispensable por las condiciones y complejidad de los procesos.

Tras observar tres aplicaciones donde se implementa el control fuzzy se puede determinar que la elección del método de desarrollo de las aplicaciones fuzzy depende del proceso en particular, de las necesidades y condiciones existentes. Es evidente entonces que existe la necesidad y sería de mucha utilidad brindar a los desarrolladores o diseñadores una metodología por la cual se pudiera insertar aplicaciones de control difuso en el SCADA Movicon.

CAPÍTULO 2. Fundamentación de la metodología para el desarrollo de aplicaciones de control fuzzy en Movicon

El apartado comienza con la descripción de las características del Movicon que posibilitan el desarrollo de una nueva aplicación, en este caso de control fuzzy. Posteriormente se indica la metodología a seguir de una forma clara y concisa y se arriba a algunas consideraciones finales.

2.1 Posibilidades del SCADA Movicon para el desarrollo de aplicaciones fuzzy

El SCADA Movicon en su versión 11.4 presenta una serie de características que posibilitan el desarrollo y correcto funcionamiento de nuevas aplicaciones enfocadas al control difuso.

2.1.1 Drivers y protocolos de comunicación

Los drivers y protocolos de comunicación son los que como su nombre lo indica se encargan de la comunicación entre los elementos del software y los dispositivos de campo (Progea, 2015). En Movicon se presta especial atención a este tema e incorpora facilidades tales como:

- Enlaces a las direcciones del PLC realizables directamente en las *Tags*, o bien indirectamente a través de 'tareas'.
- Configuración runtime mediante interface Script VBA.
- Importación automática de la base de datos del dispositivo.
- Función Bridging para posibilitar el acceso transparente al dispositivo desde el exterior vía modem. Por ejemplo *teleservice*.

- Funciones TAPI *(Telephony Application Programming Interface)* para posibilitar la llamada automática a dispositivos seriales remotos vía modem.
- Funciones RAS para posibilitar la llamada automática a dispositivos remotos Ethernet vía modem.
- Prueba de cableado y comunicación inmediato y directo.

De acuerdo al esquema de la Figura 2.1, el driver ejecuta a bajo nivel el protocolo de comunicación:

1. A través de la "Estación", el driver precisa definir los parámetros principales en la comunicación. Según se trate de drivers seriales o de red, se deberá definir los parámetros correspondientes.
2. A través de la *Task* (Tarea), el driver permite definir la asociación indirecta entre las direcciones del dispositivo y las variables del proyecto del Supervisor. Las *Tasks* ofrecen la posibilidad de definir cómo comunicar por bloques de datos, al incluir una variable o un grupo de variables en asociación a una dirección o dirección de partida del dispositivo. De este modo, el usuario puede configurar de forma directa y por lo tanto independiente del proyecto, los enlaces a las áreas de memoria del dispositivo.
3. A través de las "Direcciones Dinámicas" el driver posibilita la asociación directa de la dirección de memoria en las propiedades de la variable en el proyecto (*Tag*). De esta manera, la variable apunta directamente a la dirección en el dispositivo, deja al driver la tarea de crear "dinámicamente" las tareas de comunicación que serán efectuadas siempre de forma optimizada.
4. El driver siempre está estrechamente relacionado con la *Realtime Database* (Base de Datos de Tiempo Real) del proyecto. Por esta razón las variables se pueden asociar directamente a partir de las propiedades de las *Tags* o bien indirectamente al utilizar las *Tasks*. En todo caso, la comunicación es optimizada según el concepto de "variables en uso".
5. La configuración del driver es salvada en los archivos *XML* correspondientes en la carpeta "Recursos" del proyecto. Los archivos se basan en el lenguaje *XML*, al igual

que el entero proyecto, para ofrecer la máxima transparencia. Los archivos del driver son:

<nome_driver>.drvsettings = archivo que contiene las configuraciones generales del driver.

<nome_driver>.dynsettings = archivo generado al inicio del runtime con las características de las tareas dinámicas calculadas.

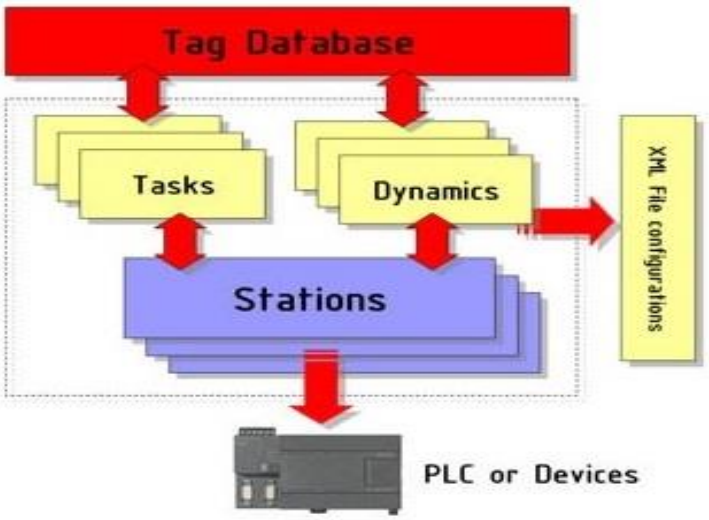

Figura 2.1 Esquema de comunicación

Además el software presenta una gran cantidad de drivers por defecto, resaltan los siguientes:

- Endress+Hauser
- Mitsubishi
- Omron
- Schneider Group
- Siemens

Asimismo se muestran los protocolos de comunicación asociados a cada driver.

2.1.2 Basic Script

Dentro de un proyecto es posible disponer de funciones basic script en diversas circunstancias y modalidades. Es conveniente utilizarlas en aquellos casos en los cuales las mismas

operaciones no estén disponibles en otros recursos o métodos: Lógica General, Propiedades de Ejecución de los Dibujos, Eventos. El código basic puede ser utilizado en varios puntos del proyecto: como recurso, directamente en las propiedades de ejecución de un objeto dibujo, como código asociado a los eventos de un objeto (alarmas, dibujo o símbolo), en los sinópticos. Movicon viene integrado en su interior con un componente de software, *WinWrap Basic*, que permite integrar dentro de la aplicación las rutinas en lenguaje Basic.

A continuación se mostrará una lista con algunas de las principales características del *WinWrap Basic* (Progea, 2015):

- Permite crear rutinas en lenguaje VBA *(Visual Basic for Application)* compatibles con el código BASIC *(Beginners All-purpose Symbolic Instruction Code)*
- Permite extender el set de instrucciones con funciones y métodos personalizados
- Permite crear dialog box al utilizar los controles standard de *Windows* (botones, checkboxes, groupboxes, listboxes, option buttons, imágenes y casillas de texto)
- Soporta los controles ActiveX
- Permite declarar y llamar las funciones API y WMI de *Windows*.

Soporte de VB.NET

Usando la llave especial '#Language "WWB.NET" se podrá pasar a la modalidad de programación de *.NET*. Esto permite acceder directamente a las *.NET* assemblies con el código VB. La ventana de diálogo para agregar los objetos de referencia a utilizarse en el código tiene en cuenta esta llave. En caso que se haya configurado el uso del lenguaje *.NET* se podrá ver la lista de todas las assemblies seleccionables.

Soporte Unicode

El editor del código permite introducir strings en formato Unicode. Por lo tanto, es posible visualizar en las ventanas de diálogo basic script líneas en Unicode o bien asignar a variables string del proyecto textos en Unicode.

Existe la posibilidad de leer y/o escribir archivos Unicode UTF8 o UTF16. A tal fin han sido agregadas dos nuevas constantes "vbUTF8BOM" y "vbUTF16BOM". Al escribir una de

estas constantes en el primer caracter de un archivo de texto, será determinada su codificación.

Bibliotecas Basic Script

Además de las funciones básicas puestas a disposición por el *WinWrap Basic*, dentro de los Basic Script es posible utilizar una serie de funciones suplementarias que forman parte de las bibliotecas de Movicon que permiten interactuar con el proyecto. Estas funciones permiten por ejemplo leer y escribir las variables de la *Real Time DB* de Movicon, efectuar cambios de página e interactuar con las propiedades de los símbolos de Movicon. Estas bibliotecas de funciones de Movicon son definidas como "Bibliotecas de Interfaz Basic" y cada interfaz abarca una serie de funciones específicas para un determinado componente del proyecto.

2.1.3 Interfaz gráfica

Los recursos Sinópticos de Movicon son los elementos a través de los cuales es posible realizar la interfaz gráfica Hombre-Máquina usando los instrumentos para la edición del dibujo y para la animación gráfica. Movicon permite realizar el dibujo mediante dos posibilidades distintas: el mismo editor gráfico interno o la inserción de dibujos bmp, jpg, gif, wmf y emf. Es posible trabajar en el diseño y utilizar ambas posibilidades, que pueden coexistir.

Los objetos y controles de Movicon, que pueden insertarse en los sinópticos son los que están disponibles en la "Ventana Objetos". Estos componentes pueden llevar a cabo diversas funciones, desde simples figuras geométricas hasta controles avanzados para la ejecución de comandos o visualización de datos.

Los objetos y controles de Movicon han sido realizados en formato vectorial propietario, y es posible también realizar símbolos compuestos de varios elementos y asociarles propiedades de animación. Tal formato es exportable o importable del formato Metafile (WMF, EMF). Mediante la "Biblioteca de Símbolos" es posible salvar y reutilizar los propios dibujos vectoriales de Movicon.

2.1.4 Eventos

Los eventos son recursos que permiten ejecutar comandos en función del cambio de una variable de Movicon. Este recurso ofrece grandes ventajas operativas en cualquier aplicación en la cual es necesario mantener controlado el valor de una o más variables a fin de ejecutar uno o más comandos al variar un determinado valor. Esto evita que el programador tenga que escribir el código necesario para el control de las variables en cuestión.

2.2 Metodología para el desarrollo de aplicaciones de control fuzzy

Para desarrollar una aplicación de control fuzzy en Movicon es necesario seguir una serie de procedimientos y tener en cuenta algunas herramientas imprescindibles que proporciona el software (Borbón, 2011). La metodología desarrollada es la siguiente:

1. Determinar las señales a medir en el proceso.

Después de estudiar las características del proceso a controlar se determina las señales a medir.

2. Determinar las señales de mando necesarias en el proceso.

La señal de mando de igual forma dependerá de las características del proceso. Esta será la salida de la aplicación fuzzy.

3. Determinar si la aplicación será para un sistema que funcione en tiempo real o para un sistema simulado.

    $\Rightarrow$ Sistema en tiempo real.

Para un sistema que funcione en tiempo real lo primero que sería necesario hacer es configurar las propiedades del driver de comunicación. En Movicon después de creado un proyecto se configura el driver en la *Real Time DB,* específicamente en la sección *Comm. Drivers.* Se configuran parámetros tales como velocidad de transmisión, paridad, datos, bits de stop y las tareas a realizar; sin los cuales una correcta comunicación sería imposible.

    $\Rightarrow$ Sistema simulado.

Para un sistema simulado lo primero que sería necesario hacer es crear un script y programar el modelo de la planta en cuestión. El modelo de la planta se programa en potencias de z, para discretizar se recomienda utilizar herramientas como la función c2d() del Matlab que facilitan el trabajo.

4. Determinar y programar las variables de entrada del controlador fuzzy.

No siempre las señales que se miden en el proceso son las entradas del controlador fuzzy. En ocasiones las entradas son variables como por ejemplo el error, en estos casos se crea un script para programar los algoritmos necesarios para obtener las variables de entrada del controlador fuzzy.

5. Programar los valores de inicialización de la aplicación.

Se crea un nuevo script y se programan los valores de inicialización de la aplicación. En la *Real Time DB* se crean las variables, esta acción se realiza en cualquier momento del desarrollo de la aplicación en el que sea necesario insertar una nueva variable. Este script se ejecutará solo la primera vez que se ejecute la aplicación con el objetivo de inicializar las variables.

6. Programar las funciones de pertenencia.

Crear un nuevo script, el cual será el script del controlador fuzzy, y programar las funciones de pertenencia para cada variable de entrada y salida. Se pueden utilizar diversas funciones como las triangulares y trapezoidales. Se establecen los puntos para cada función de pertenencia como se muestra en el ejemplo de una función de pertenencia triangular a continuación:

```
Function Función_Ejemplo() As Double
If variable > a And variable < b Then
    Función_Ejemplo = ( variable – a ) / ( b – a )
Else
        If variable >= b And variable <= c Then
            Función_Ejemplo = ( c – variable ) / ( c – b )
        Else
            Función_Ejemplo = 0
        End If
```

```
End If
End Function
```

Donde: *Función_Ejemplo – Cualquier función de pertenencia*

a, b, c – puntos que describen la función triangular

variable – variable de entrada o salida de la aplicación fuzzy

7. Crear e inicializar vectores de índice para las funciones de pertenencia.

En cada período se evalúa la pertenencia de cada variable y estos valores se pasan a los vectores de índice para su posterior manejo.

```
vector(0) = función 1 ()
vector(1) = función 2()
```

▪

▪

```
vector(n-1) = función n()
```

Donde: *vector – vector que almacena los valores de las funciones de pertenencia de una variable de entrada o salida*

función – función de pertenencia de una variable de entrada o salida

n – número total de funciones de pertenencia de una variable de entrada o salida

8. Definir y programar las reglas difusas.

Las reglas difusas son parte esencial del controlador fuzzy. Al igual que las funciones de pertenencia, para definir las reglas difusas es necesario el conocimiento experto del proceso. Para programar las reglas difusas se utiliza una matriz. La cantidad de reglas depende de las combinaciones posibles entre el número de funciones de pertenencia por cada variable de entrada. Un ejemplo del algoritmo de programación se observa a continuación (ver epígrafe 1.2.1):

```
Dim m(var1,var2) As Double
```

```
m(0,0) = a
m(1,0) = b
```

▪

▪

Donde: *m – matriz de reglas difusas*

var1 – valor que representa la cantidad de funciones de pertenencia para una variable de entrada

var2 – valor que representa la cantidad de funciones de pertenencia para una variable de entrada

a – variable que representa un índice del vector de salida

b - variable que representa un índice del vector de salida

9. Programar rutinas para determinar qué funciones de pertenencia están activas y sus respectivas reglas difusas.

Se trabaja con ciclos y estructuras *if-else* para determinar en cada momento qué funciones de pertenencia están activas y las reglas que se aplican. Se almacena la información utilizando matrices y contadores. Se muestra un ejemplo del algoritmo a implementar a continuación:

```
For i=0 To n-1
  If vector1(i)>0 Then
    For j=0 To n-1
      If vector2(j)>0 Then
      marcador(markCont,0)=i
      marcador(markCont,1)=j
      marcador(markCont,2)=m(j,i)
      markCont=markCont+1
      End If
    Next j
  End If
Next i
```

Donde: *marcador – matriz que almacena las posiciones de las funciones de pertenencia y las reglas que están activas*

markCont – contador

i – variable interna del ciclo for

j – variable interna del ciclo for

vector1 – vector que almacena los valores de las funciones de pertenencia de una variable de entrada

vector2 – vector que almacena los valores de las funciones de pertenencia de una variable de entrada

10. Aplicar criterios como los de máx-mín.

Los criterios como los de máx-mín posibilitan establecer el conjunto borroso de salida (ver epígrafe 1.2.1)

```
For i=0 To markCont
 num3=vector1(marcador(i,0))
 num4=vector2(marcador(i,1))
 num1=vector3(marcador(i,2))
   If num3<num4 Then
        num2 = num3
   Else
        num2 = num4
   End If

  If num1>num2 Then
       vector3(marcador(i,2))= num1
  Else
       vector3(marcador(i,2)) = num2
  End If
Next i
```

Donde: *vector3 – vector que almacena valores probabilísticos de salida*

11. Defusificar el conjunto de salida.

Para defusificar el conjunto borroso de salida se pueden utilizar diferentes algoritmos como el de media de centros que se muestra a continuación:

```
 For i=0 To n-1
   num+=vector3(i)*vector4(i)
   den+=vector3(i)
Next i
Salida = num/den
```

Donde: *vector4 – vector que almacena los posibles valores de la variable de salida*

Salida – variable de salida, el mando

12. Aplicar el mando al proceso.

    ⇒ Si el proceso es real dar salida a través del driver.

    ⇒ Si el proceso es simulado aplicar el mando al modelo de la planta programado en un script con anterioridad.

13. Visualizar en sinópticos el funcionamiento del sistema.

Crear un sinóptico, insertar y configurar alguna herramienta de visualización como el *Trend* para observar la respuesta del sistema.

2.3 Consideraciones finales del capítulo.

El SCADA Movicon posee características que posibilitan y facilitan el desarrollo de nuevas aplicaciones. Para implementar una aplicación de control fuzzy es imprescindible que el software cuente con flexibilidad a la hora de insertar nuevos algoritmos de programación o editar los ya existentes. Movicon en este sentido permite insertar y editar algoritmo en lenguaje VBA en diversos puntos del proyecto: como recurso, en sinópticos, en objetos.

Se arribó a una metodología que desde el punto de vista de algoritmos de programación no es complicada de aplicar. Para implementar la metodología se observa la imperiosa necesidad de tener amplios conocimientos sobre el proceso a controlar, aspecto fundamental para desarrollar cualquier aplicación de control fuzzy.

CAPÍTULO 3. Aplicación y validación de la metodología

En el capítulo se parte de implementar la metodología propuesta en el control de un motor de C.D. Se realiza una comparación entre el resultado que se alcanza con la aplicación de la metodología y el mismo sistema desarrollado al utilizar otras herramientas profesionales que brindan fiabilidad a la investigación. El objetivo es apreciar si la aplicación de la metodología cumple con los requisitos, es decir, realizar un análisis al tomar como base la comparación que se establece con los módulos profesionales. Tanto la herramienta de lógica difusa del Matlab (*Fuzzy Logic Toolbox*) primeramente, como el PID del Movicon en segunda instancia, sirven de patrones para las comparaciones que se realizan. Se efectúa una valoración económica y finalmente se exponen algunas consideraciones finales.

3.1 Aplicación de la metodología en el control de velocidad de un motor de corriente directa

La planta a controlar es un motor FRACMO. Se utiliza un amplificador, con ganancia igual a 25, que antecede al motor y posibilita que la señal de mando necesaria no sea excesivamente grande. El modelo del sistema, es decir, del amplificador y el motor conjuntamente, según (Fernández, 2008) es:

$$\frac{Vel(s)}{Va(s)} = \frac{K}{\tau s+1} \tag{3.1}$$

Donde: *Vel – velocidad del motor*.

Va – voltaje de armadura aplicado al motor.

*K – ganancia del sistema = 25 rad*s^{-1}*V^{-1}*.

τ – constante de tiempo del sistema = 0.25 s.

Procedimiento 1. Determinar las señales a medir en el proceso.

Como se observa el proceso es sencillo y por demás basta con medir la velocidad del motor.

Procedimiento 2. Determinar las señales de mando necesarias en el proceso.

Para controlar la velocidad del motor se manipula el voltaje de armadura del mismo, esta variable es el mando de la aplicación.

Procedimiento 3. Determinar si la aplicación será para un sistema que funcione en tiempo real o para un sistema simulado.

El proceso se simulará en el Movicon, por tanto, como se expone en la metodología es necesario programar el modelo de la planta en un script.

La constante de tiempo de la planta es muy pequeña y en lazo cerrado el proceso se hará mucho más rápido, por lo tanto, es conveniente normalizar el tiempo para que al simular se pueda apreciar con claridad el funcionamiento del sistema. En este caso, al ser un sistema de primer orden, normalizar el tiempo sería básicamente multiplicar la constante de tiempo por un factor N=320 y obtenemos una $\tau^{'}$=80 s (ver ecuación 3.2). Con este cambio solo hay que tener en cuenta que al simular el proceso el tiempo real es el que se muestra dividido por 320s.

$$\frac{Vel(s)}{Va(s)} = \frac{25}{80s+1} \tag{3.2}$$

Movicon no trabaja con valores en el campo s, es por esto que es necesario discretizar la ecuación 2.2. Para discretizar se utilizó la función c2d() del software Matlab y la planta queda como muestra la ecuación 3.3.

$$\frac{Vel(z)}{Va(z)} = \frac{0.3106}{z-0.9876} \tag{3.3}$$

La función de transferencia anterior se transforma a potencias negativas de z y queda como muestra la ecuación 2.4.

$$\frac{Vel(z)}{Va(z)} = \frac{0.3106z^{-1}}{1-0.9876z^{-1}} \tag{3.4}$$

Se multiplica cruzado y se despeja la velocidad (ver ecuación 3.5).

$$Vel(z) = 0.9876z^{-1}Vel(z) + 0.3106z^{-1}Va(z) \quad (3.5)$$

Finalmente la ecuación 3.6 muestra cómo queda programada la planta en un script creado para este propósito:

$$Vel = 0.9876Vel + 0.3106Va \quad (3.6)$$

Hay que tener en cuenta que las variables *Vel* y *Va* que forma parte de la suma contienen los valores del instante de muestreo anterior en consecuencia con las potencias de z negativas de la ecuación 3.5.

Procedimiento 4. Determinar y programar las variables de entrada del controlador fuzzy.

Las variables de entrada escogidas son el error y la derivada del error (Shakya, 2014, Vaishnav, 2007).

Se muestrea la velocidad y en un script creado para este propósito se determina el error como la diferencia entre una velocidad de referencia y la velocidad actual. La derivada del error se plantea como la diferencia entre el error actual y el anterior (ver ecuaciones 3.7 y 3.8).

$$e = Velr - Vel \quad (3.7)$$

Donde: *e – variable que representa el valor del error.*

Velr – Velocidad de referencia.

$$p = e - e_{ant} \quad (3.8)$$

Donde: *p – variable que representa el valor de la derivada del error.*

e_{ant} *– variable que representa el valor del error en el instante de muestreo anterior.*

Procedimiento 5. Programar los valores de inicialización de la aplicación.

Se creó un script que se va a ejecutar de primero una sola vez en cada período que contiene la inicialización a 0 de las principales variables del proceso tales como velocidad y error. Esto con el objetivo de reiniciar cada vez que se simule.

Procedimiento 6. Programar las funciones de pertenencia.

Se creó un script para programar el controlador fuzzy. En este script lo primero es programar las funciones de pertenencia para cada variable de entrada y salida. Todas las funciones de pertenencia se seleccionaron del tipo triangular. La función de pertenencia que describe el error en un rango deseado se muestra como ejemplo a continuación:

```
Function e_OK() As Double
If e>-0.1 And e<0 Then
    e_OK = (e+0.1)/0.1
Else
        If e>=0 And e<=0.1 Then
           e_OK = (0.1-e)/0.1
       Else
           e_OK = 0
        End If
End If
End Function
```

Donde: *e_OK – función de pertenencia que describe el comportamiento del error dentro de un rango de valores deseado.*

Procedimiento 7. Crear e inicializar vectores de índice para las funciones de pertenencia.

Se crearon vectores de índice a los cuales se les pasa los valores de las funciones de pertenencia.

```
uE(0)=e_OK()          uP(0)=p_OK()
uE(1)=e_PP()          uP(1)=p_PP()
```

. .

. .

Donde: *e_PP – función de pertenencia que describe el comportamiento del error dentro de un rango de valores positivos pequeños.*

p_OK – función de pertenencia que describe el comportamiento de la derivada del error dentro de un rango de valores deseado.

p_PP – función de pertenencia que describe el comportamiento de la derivada del error dentro de un rango de valores positivos pequeños.

uE – vector que almacena los valores de las funciones de pertenencias del error.

uP – vector que almacena los valores de las funciones de pertenencias de la derivada del error.

Procedimiento 8. Definir y programar las reglas difusas.

Se programó una matriz de 7x7, ya que son dos entradas cada una con 7 funciones de pertenencia, que contendrá las reglas difusas.

```
Dim m(7,7) As Double
  m(0,0) = 3
  m(1,0) = 4
```

Donde: *m – matriz de reglas difusas.*

Procedimiento 9. Programar rutinas para determinar qué funciones de pertenencia están activas y sus respectivas reglas difusas.

Se programaron rutinas para determinar que funciones de pertenencia están activas así como sus respectivas reglas difusas. Se almacenaron las posiciones en una matriz para posteriormente continuar el trabajo.

```
For i=0 To 6
  If uE(i)>0 Then
    For j=0 To 6
      If uP(j)>0 Then
      marcador(markCont,0)=i 'POS Error
      marcador(markCont,1)=j 'POS Derivada
      marcador(markCont,2)=m(j,i) 'PARTICION DE SALIDA
      markCont=markCont+1
      End If
```

```
    Next j
  End If
Next i
```

Donde: *marcador – matriz que almacena las posiciones de las funciones de pertenencia y las reglas que están activas.*

markCont – contador

Procedimiento 10. Aplicar criterios como los de máx-mín.

Se aplicó el criterio máx-mín para determinar el conjunto borroso de salida.

```
For i=0 To markCont
 num3=uE(marcador(i,0))
 num4=uP(marcador(i,1))
 num1=uO(marcador(i,2))
    If num3<num4 Then
        num2 = num3
    Else
        num2 = num4
    End If

   If num1>num2 Then
       uO(marcador(i,2))= num1
   Else
       uO(marcador(i,2)) = num2
   End If
Next i
```

Donde: *uO – vector que almacena valores probabilísticos de salida.*

Procedimiento 11. Defusificar el conjunto de salida.

Se defusificó el conjunto de salida a través del algoritmo de media de centros.

```
For i=0 To 6
  num+=uO(i)*out(i)
```

```
  den+=uO(i)
Next i
Va=num/den
```

Procedimiento 12. Aplicar el mando al proceso.

Como el sistema es simulado el mando se aplicó en el script donde se programó el modelo de la planta.

Procedimiento 13. Visualizar en sinópticos el funcionamiento del sistema.

Se construyó una interfaz para visualizar la respuesta del sistema. La velocidad de referencia del motor se estableció en 200 rad/s. A continuación se muestra el gráfico en la Figura 3.1 donde se observa el correcto funcionamiento del sistema.

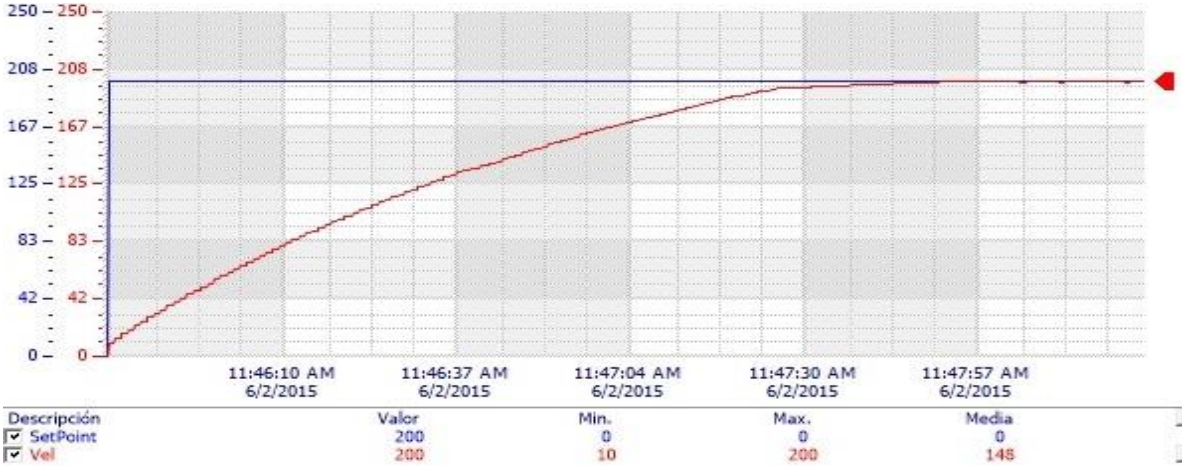

Figura 3.1 Respuesta del sistema en lazo cerrado.

3.2 Desarrollo de la aplicación en Matlab

El software Matlab es una poderosa herramienta matemática que posibilita también el diseño de controladores. Presenta numerosos módulos que facilitan el trabajo ante cualquier problemática. Uno de esos módulos es la herramienta de lógica difusa *Fuzzy Logic Toolbox* que posibilita el diseño de controladores difusos de una forma muy práctica y eficaz.

Se diseñó con esta herramienta un controlador fuzzy para el control del motor de la aplicación en análisis. Por supuesto este diseño se realizó con la implementación del mismo conocimiento experto con que se desarrolló el controlador fuzzy en el Movicon. Esto significa que las funciones de pertenencia, las reglas difusas y los métodos matemáticos

escogidos son similares a los anteriormente utilizados. Los controladores fuzzy y el controlador PI que se muestra posteriormente fueron diseñados para que el mando se mantenga en todo momento entre -11 V y 11 V.

En el Simulink de Matlab se creó el diagrama que representa el sistema como muestra la Figura 3.2.

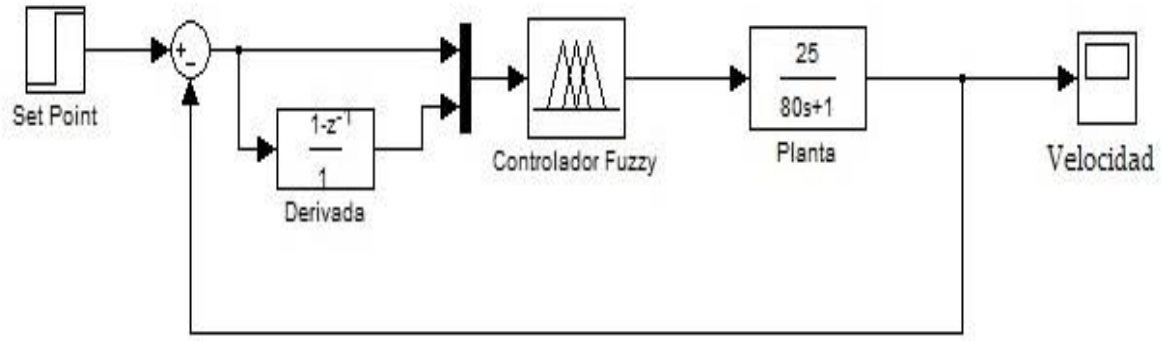

Figura 3.2 Diagrama del sistema en lazo cerrado.

La velocidad de referencia de igual forma se colocó en 200 rad/s. La figura 3.3 muestra el resultado de la simulación.

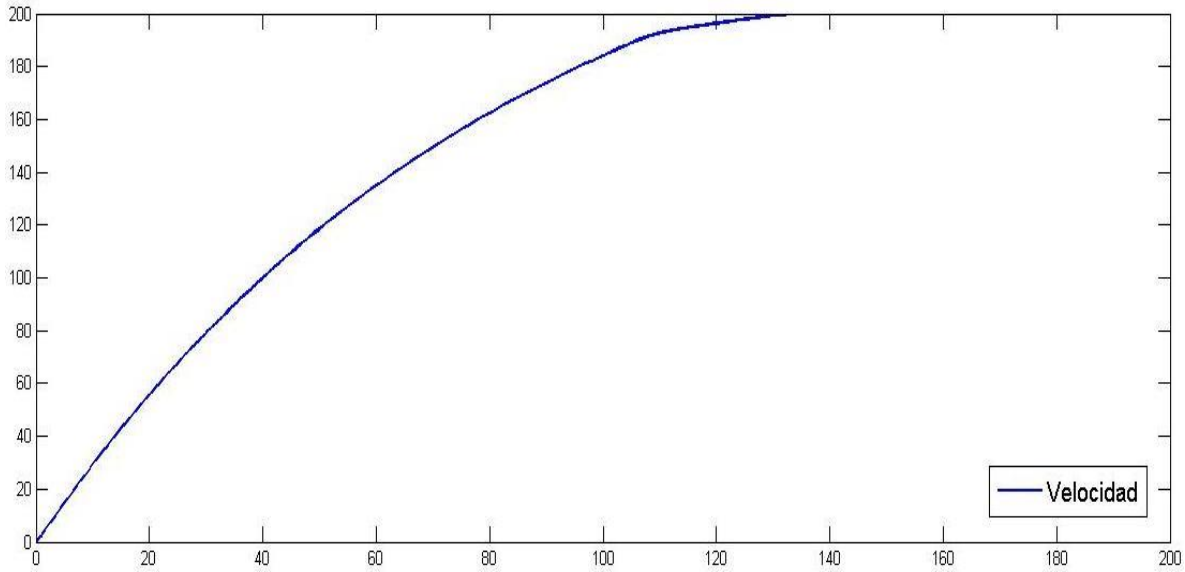

Figura 3.3 Respuesta del sistema en lazo cerrado con el controlador fuzzy de Matlab.

Como se aprecia el sistema se estabiliza aproximadamente a los 130 s. No presenta sobreimpulso ni error en estado estable. Cabe señalar que la línea temporal real es la que se muestra dividida por 320, ya que el sistema está normalizado.

Al comparar esta respuesta con la de la Figura 3.1, es decir, a la respuesta del sistema con el controlador fuzzy en Movicon, se puede apreciar que son similares. En Movicon el tiempo de establecimiento fue de aproximadamente 125 s, no presentó igualmente ni sobreimpulso ni error en estado estable.

3.3 Desarrollo de la aplicación en Movicon con el PID

Movicon dispone de un potente motor para elaborar las lógicas dentro de los sinópticos, las Sinapsis. Estas son instrumentos puestos a disposición por el programador para introducir este tipo de lógica. Su empleo permite conectar gráficamente los objetos en el sinóptico entre sí y programar las lógicas mediante códigos basic script que se asocian con cada objeto.

En la biblioteca de símbolos de Movicon existen componentes que contienen lógica de sinapsis y permiten crear un PID. La Figura 3.4 muestra el diseño de un PID en el ambiente de desarrollo del Movicon.

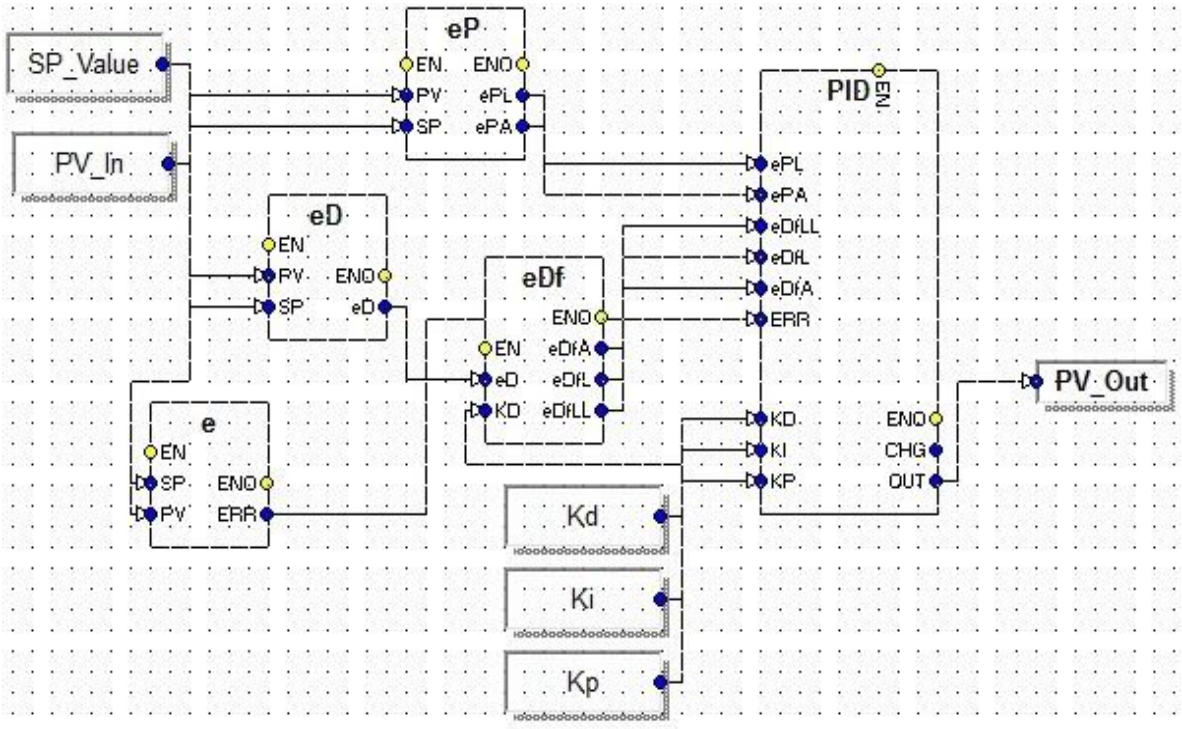

Figura 3.4 Diseño de PID en Movicon.

La planta del proceso en estudio es de primer orden, por esta causa no es necesario controlar con un PID sino que con un PI es suficiente. Para el ajuste del controlador PI se aprovecharon las facilidades de la herramienta del Matlab *PID Tuner*.

Los parámetros del controlador quedaron:

Kp (Ganancia proporcional) = 0.4

τ_i (Tiempo de integración) = 13.3

En la Figura 3.5 se observa la respuesta del sistema trabajando con el controlador PI diseñado.

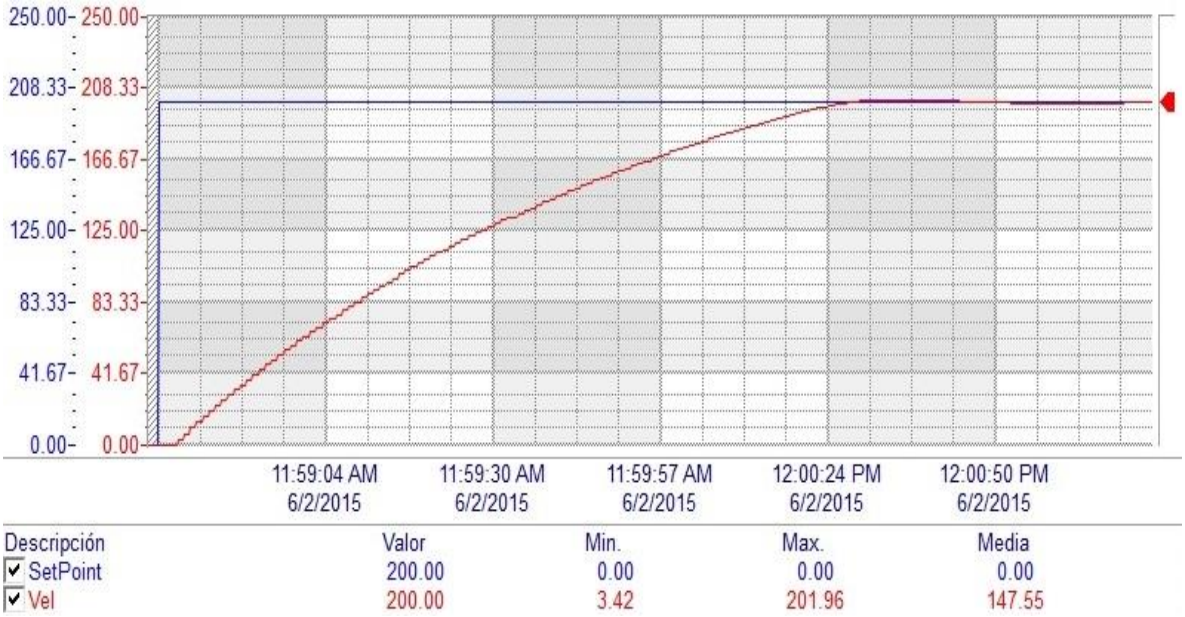

Figura 3.5 Respuesta del sistema en lazo cerrado controlando con el PI del Movicon.

Como se aprecia la respuesta del sistema con el controlador PI es más rápida, con un tiempo de establecimiento de aproximadamente 110 s. Esto se debe a que, al entrar en funcionamiento el sistema, la salida del controlador PI inmediatamente comienza a ser de 11 V hasta que se acerca al valor de establecimiento; mientras que la señal de mando del controlador fuzzy nunca llega a ser de 11 V por el propio algoritmo de diseño del controlador que hace que el mando límite superior esté entre los valores de 10 V y 11V. Evidentemente al ser la salida del controlador PI un poco mayor que la salida del controlador difuso se logra arribar al valor de establecimiento más rápido.

Al analizar otros parámetros se observa que el sistema con el controlador PI presenta un sobreimpulso mínimo, la velocidad alcanza los 201.96 rad/s, es decir, 0.98% por encima del valor de establecimiento; mientras que la aplicación de control fuzzy de la metodología no impone sobreimpulso alguno. Las gráficas muestran que ninguna de las dos variantes arroja error en estado estable.

3.4 Comparación entre los distintos controladores

Después de desarrollar los distintos controladores se observa la paridad que existe entre las respectivas respuestas, lo que demuestra que es posible implementar la metodología que se propone obteniendo buenos resultados. Para facilitar el análisis se muestra la Tabla 3.1 que contiene una comparación entre los principales elementos a evaluar para cada sistema.

Tabla 3.1 Comparación entre los controladores diseñados.

	Sistema basado en la metodología	**Sistema diseñado en Matlab**	**Sistema con PID de Movicon**
Ess (error en estado estable)	-	-	-
Mp (sobreimpulso)	-	-	0.98%
ts (tiempo de establecimiento)	125 s	130 s	110 s

La metodología es efectiva, como se observa en la respuesta del sistema tras su aplicación. Demuestra ser otra opción fiable en cuanto a estrategias de control se trata.

3.5 Valoración económica

La viabilidad y fiabilidad económica de esta investigación está estrechamente ligada a las ventajas desde el punto de vista económico que posee la utilización de los SCADA en los sistemas de automatización. Los SCADA, por su capacidad de realizar a distancia operaciones de control, supervisión y registro de datos, resultan sumamente eficientes. En un principio la puesta en marcha de un sistema de este tipo puede traer consigo gastos

significativos por concepto de adquisición del equipamiento e implantación acorde a las necesidades y requisitos exigidos; pero con el tiempo esta inversión inicial es recuperada ya que los SCADA reducen en gran medida los costos de producción, operación y mantenimiento.

Además los SCADA por su flexibilidad permiten la diversificación de la producción y su aumento. Todo esto repercute en el saludable estado económico de la empresa cliente.

El hecho de dotar al SCADA Movicon de una aplicación de control fuzzy aumenta la posibilidad de su utilización. En casos en que sea necesario implementar esta estrategia de control inteligente ya no se desechará el Movicon y/o se buscarán otras alternativas como los PLC. La metodología propuesta complementa entonces el uso de este excelente software que por su eficiencia y bajo costo es cada día más utilizado en Cuba.

3.6 Consideraciones finales del capítulo

La metodología que se propone se aplicó y quedó verificada al ser comparada con herramientas profesionales como son *Fuzzy Logic Toolbox* de Matlab y el PID de Movicon y obtener buenos resultados. A pesar de que el proceso de estudio es de relativa simplicidad se constató el acierto de la metodología desarrollada. El análisis económico fundamentó la importancia del uso de esta metodología como herramienta que favorece la utilización del software Movicon.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Con la culminación de la investigación y el análisis de los resultados alcanzados se arriba a las siguientes conclusiones:

1. El análisis bibliográfico determinó que los sistemas SCADA por sus características de capacidad de gestión y supervisión son cada día más utilizados en el mundo y en Cuba particularmente. La bibliografía consultada nos permite afirmar que la utilización de control fuzzy en sistemas SCADA es cada vez más frecuente dado por la factibilidad de dicha estrategia de control en numerosos procesos.
2. La presente investigación permitió constatar la flexibilidad del software Movicon para la incorporación de nuevas aplicaciones lo que hace del mismo un software con diversas posibilidades de uso.
3. El estudio realizado resaltó la importancia del conocimiento experto del proceso en cuestión para el correcto desarrollo de la metodología.
4. La aplicación de la metodología desarrollada proporcionó buenos resultados tanto en el estado estable como en el estado transitorio de los sistemas a controlar y la comparación realizada testifica la fiabilidad y viabilidad del sistema de procedimientos propuesto.

Recomendaciones

Terminado el trabajo se realizan las siguientes recomendaciones:

- Utilizar la metodología propuesta en el desarrollo de aplicaciones de control fuzzy para la automatización de procesos que así lo requieran.
- Continuar el estudio del desarrollo de aplicaciones de control inteligente en sistemas SCADA.

REFERENCIAS BIBLIOGRÁFICAS

AGREDO FAJARDO, J. 2012. Implementación de control de velocidad de un motor DC utilizando lógica difusa en la plataforma de LAbview.

BORBÓN, P. 2011. *Implementación de un controlador difuso en un PIC16F877A (CCS PIC-C)* [Online]. Available: www.PabloBorbón.html 2015].

COLOMER, J., MELÉNDEZ, J. & AYZA, J. 2000. Sistemas de supervisión. *Cuadernos CEA-IFAC*.

COTERO OCHOA, J. 2002. Control clasico/moderno y control inteligente. *Control clasico y control inteligente***,** 1-35.

CHACON, D., DIJORT, O. & CASTRILLO, J. 2001. Supervisión y control de procesos. *EUPVG-UPC (2001-2002).[En línea][http://ocw. upc. edu/sites/default/files/materials/15012628/40194-3452. pdf, 31 de mayo de 2014]*.

CHACÓN MORALES, R. S. 2012. *Simulación SCADA (Control, supervisión y adquisición de datos) de una planta generadora de energía eléctrica a base de energía geotérmica.* Universidad de El Salvador.

CHAVARRÍA MEZA, L. E. 2007. SCADA System's & Telemetry.

FERNÁNDEZ, R. M. 2008. Control de velocidad de un servomotor de corriente continua.

GÓMEZ, J. C. 2008. Fuzzy Control. *Buenos Aires: Editorial de la Universidad Tecnologica Nacional*.

HENTEA, M. 2008. Improving security for SCADA control systems. *Interdisciplinary Journal of Information, Knowledge, and Management,* 3**,** 73-86.

IZAGUIRRE, E. 2002. *Sistemas de Automatización*.

KOURO, S. & MUSALEM, R. 2002. Control mediante lógica difusa. *Técnicas modernas automaticas,(1-7)-7*.

MOHAMED NASIR, N. 2011. *Real time implementation of PID and fuzzy logic controller for single tank system using LABVIEW.* Universiti Tun Hussein Onn Malaysia.

MONTEJO RODRÍGUEZ, L. 2014. *Propuesta de SCADA de Gestión Energética para el Hotel La Estrella 1*.

PASSINO, K. Y., STEPHEN 1998. *Fuzzy Control*.

PROGEA. 2015. Available: www.progea.com [Accessed 02/03/2015.

SÁNCHEZ, E. N. & ALANIS, A. Y. 2006. Redes neuronales: conceptos fundamentales y aplicaciones a control automático. *Cinvestav Unidad Guadalara. Editorial Prentice Hall*.

SHAKYA, R. R., KRITIKA; PATEL, SANSKRITI; DINKAR, SMITA 2014. Design and Simulation of PD, PID and Fuzzy Logic Controller for Industrial Application. *International Journal of Information and Computation Technology,* 4**,** 363-368.

SMOCZEK, S. & SZNYTKO, J. 2008. The HMI/SCADA in control systems and supervision processes of manufacturing transport. *Journal of KONES,* 15**,** 499-507.

SUAZO CRESPO, M. A. 2014. *Metodología para implementar Sistemas de Alarmas en Movicon.* Universidad Central "Marta Abreu" de Las Villas.

VAISHNAV, S. R. K., Z.J. Design and Performance of PID and Fuzzy Logic Controller with Smaller Rule Set for Higher Order System. Proceedings of the World Congress on Engineering and Computer Science 2007, 2007 San Francisco, EE.UU

VALDÉS SEOANE, J. I. 2009. *Sistema de guías sobre el entorno de desarrollo SCADA MOVICON X2.*