UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO VALLE DE CHALCO

SINTONIZACIÓN AUTOMÁTICA DE VELOCIDAD Y POSICIÓN PARA SERVOMOTORES UTILIZANDO CONTROL DIFUSO

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ING. ROBERTO URIBE FLORES

TUTOR ACADÉMICO:

DR. EN C.E. JOSÉ LUIS SÁNCHEZ RAMÍREZ

TUTORES ADJUNTOS:

DRA. EN C.E. CRISTINA JUÁREZ LANDÍN

DR. SAMUEL OLMOS PEÑA

CENTRO UNIVERSITARIO XICO VALLE DE CHALCO

VALLE DE CHALCO SOLIDARIDAD, MÉXICO. ENERO 2015

Valle de Chalco Solidaridad, Edo. de México a 27 de enero de 2015.

Dr. Samuel Olmos Peña
Coordinador de la Maestría en Ciencias de la Computación
Centro Universitario UAEM Valle de Chalco
Presente

Por este medio le comunicamos a usted que la Comisión Revisora designada para analizar la tesis denominada "**Sintonización automática de velocidad y posición para servomotores utilizando control difuso**", que como parte de los requisitos para obtener el grado académico de Maestría en Ciencias de la Computación presenta el **Ing. Roberto Uribe Flores** con número de cuenta **1230663** para sustentar el acto de Recepción Profesional, ha dictaminado que dicho trabajo reúne las características de contenido y calidad necesarios para proceder a la impresión del mismo.

Atentamente

Tutora Adjunta	**Tutor Académico**	**Tutor Adjunto**
Dra. Cristina Juárez Landín	**Dr. En C.E. José Luis Sánchez Ramírez**	**Dr. Samuel Olmos Peña**

Oficio Coord MACSCO 004/2015
Valle de Chalco Solidaridad, Edo. de México a 28 de enero del 2015.

Ing. Roberto Uribe Flores
Candidato al Grado de Maestría en Ciencias de la Computación
Centro Universitario UAEM Valle de Chalco
Presente

De acuerdo con el Reglamento de Estudios Avanzados de la Universidad Autónoma del Estado de México y habiendo cumplido con todas la indicaciones que la Comisión Revisora realizó con respecto a su trabajo de tesis titulado ***"Sintonización Automática de Velocidad y Posición Para Servomotores Utilizando Control Difuso"***, la Coordinación de la Maestría en Ciencias de la Computación del Centro Universitario UAEM Valle de Chalco, concede la autorización para que proceda a la impresión de la misma.

Sin más por el momento, le reitero la seguridad de mi especial consideración y estima.

ATENTAMENTE
"PATRIA, CIENCIA Y TRABAJO"
"2015, Año del Bicentenario Luctuoso de José María Morelos y Pavón"

Centro Universitario UAEM

Dr. Samuel Olmos Peña.
Coordinador de la Maestría en Ciencias de la Computación.
C.U UAEM Valle de Chalco
Universidad Autónoma del Estado de México

Valle de Chalco COORDINACIÓN DE MAESTRÍA

c.c.p. Archivo.
SOP

CARTA DE CESIÓN DE DERECHOS DE AUTOR

El que suscribe **Roberto Uribe Flores** Autor del trabajo escrito de evaluación profesional en la opción de **Tesis** con el título **Sintonización automática de velocidad y posición para servomotores utilizando control difuso**, por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los artículos 35 y 36 fracción II de la Ley de la Universidad Autónoma del Estado de México; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en **CU UAEM, Valle de Chalco Solidaridad, Edo. De México** para ser evaluada con el fin de obtener el Título Profesional de **Maestro en Ciencias de la Computación**.

Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, a la Universidad Autónoma del Estado de México; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura.

Entendiendo que dicha cesión no genera obligación alguna para la Universidad Autónoma del Estado de México y que podrá o no ejercer los derechos cedidos.
Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.

Se firma la presente en la ciudad de **Estado de México**, a los **26** días del mes de **Enero** de 20**15**.

Ing. Roberto Uribe Flores
Nombre y firma de conformidad

DEDICATORIA

Dedico este trabajo a mi hermosa hijita, Ashley Idalys Uribe Galicia, quien con su ternura y cariño, me ha llenado de felicidad y me ha dado la fuerza de voluntad para culminar este ciclo de mi vida profesional.

.

A mi amada esposa Lic. Cony Galicia Sánchez, por haberme brindado su apoyo incondicional, comprensión, paciencia, amor y el ánimo durante toda esta etapa.

A mis amados padres Roque Uribe Araujo y Hortensia Flores Adalid quienes además de darme la vida, me dieron mi carrera y su ejemplo para hacer de mí una persona de bien.

Y por último a mi hermano por todas sus palabras de aliento y consejos invaluables.

AGRADECIMIENTOS

Me gustaría que estas líneas sirvieran para expresar mi más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo, en especial al Dr. en C.E. José Luis Sánchez Ramírez, director de esta investigación, por la orientación, el seguimiento y la supervisión continúa de la misma, pero sobre todo por la motivación y el apoyo recibido a lo largo de estos años.

Especial reconocimiento merece el interés mostrado por mi trabajo y las sugerencias recibidas de la Dra. en C.E. Cristina Juárez Landín, con la que me encuentro en deuda por el ánimo infundido y la confianza en mí depositada. También me gustaría agradecer la ayuda recibida del Dr. Samuel Olmos Peña.

Quisiera hacer extensiva mi gratitud a todos mis profesores, de quienes recibí un apoyo incondicional y quienes me inspiraron y me guiaron a lo largo de esta maestría, siempre demostrando estar dispuestos a ayudar en los momentos más duros sin pedir nada a cambio.

A mis compañeros y amigos que nunca titubearon en darme apoyo y consejos en momentos difíciles de mi vida.

Roberto Uribe Flores agradece el apoyo otorgado por CONACYT mediante el programa PNPC, así también agradece a la Universidad Autónoma del Estado de México, por el apoyo otorgado.

A todos ellos, muchas gracias.

RESUMEN

El presente trabajo de investigación, consiste en el desarrollo de un controlador para servomotor, utilizando lógica difusa para controlar velocidad y posición.

Los servomotores tienen una gran cantidad de aplicaciones de tipo industrial, principalmente en la fabricación de robots industriales, máquinas de control numérico y procesos que requieren control de movimiento preciso. La incorporación de los controladores de lógica difusa para procesos complejos permite al sistema, trabajar más cercano a la forma en que el cerebro humano funciona. La implementación de este sistema en lugar de uno lineal (PID) proporciona robustez y fiabilidad.

Como ya se ha mencionado anteriormente el control de servomotores se lleva a cabo mediante algoritmos de control de tipo lineal, dentro de los que pueden resaltarse, el Proporcional (P), Integral (I), y Derivativo (D), todos estos aplicados de manera independiente según sean los requerimientos del proceso a controlar, o bien; pueden funcionar de manera combinada como el PI, el PD, o el PID, sin embargo ante cambios no lineales, su funcionamiento se ve afectado, pues dejan de trabajar eficazmente.

Tomando como base de conocimiento lo anterior, se diseñó un controlador lógico difuso (fuzzy), de tipo Mamdani, que funciona con dos variables lingüísticas de entrada (velocidad y posición), un motor de inferencia difuso compuesto por un conjunto de reglas de tipo IF, THEN, ELSE, conocido como etapa de fuzzificación. A la salida del controlador se tiene una señal defuzzificada misma que se bifurca para retroalimentar al sistema, cerrando así, su lazo de control.

La construcción del controlador se realizó en Matlab, utilizando para ello su módulo de edición de lógica difusa (Fuzzy Logic Designer), en un ambiente de programación gráfico e intuitivo. Por otra parte la implementación, se realizó en Labview, debido a su facilidad y rapidez para construir la interfaz gráfica y los modelos lineales que se usaron como referencia.

Por lo anterior, el diseño de controladores PID basados en lógica difusa, es motivado por la habilidad de estos de capturar estrategias cualitativas de control y ofrecer un comportamiento altamente flexible.

ABSTRACT

The present research consists in developing a controller for servomotor, using fuzzy logic to control speed and position.

Electric motors have a lot of industrial applications, mainly in the manufacture of industrial robots, CNC machines and processes that require precise motion control.
Incorporating fuzzy logic controllers for complex process allows the system, closer to the way the human brain works. The implementation of this system instead of a linear system (PID) provides robustness and reliability.

As it has already mentioned above servo control is performed by control algorithms of linear type, among which can be highlighted, the proportional (P), Integral (I) and Derivative (D), all these applied independently according to the requirements of process control, or; can perform in combination as the PI, PD, PID, however in front of non-linear changes its operation is affected since they stop working effectively.

Taking this knowledge as the basis, it was designed a fuzzy logic controller (fuzzy) of Mamdani type, which works with two linguistic input variables (velocity and position), a fuzzy inference engine integrated with a set of type rules: IF, THEN, ELSE, known as fuzzification stage. In the controller output, there is a signal which is defuzzificated and this is divided in order to provide feedback to the system, thus closing its control loop.

The construction of the controller is performed in Matlab, using its editing module fuzzy logic (Fuzzy Logic Designer), in an environment of graphical and intuitive programming. On the other hand the implementation, was made in Labview, due to its easiness and fastness to construct the graphical interface linear models that in this case were used just as references.

Therefore, the design based on fuzzy logic, PID controllers is motivated by the ability of these to capture qualitative control strategies and provide a highly flexible behavior.

INDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

1 ANTECEDENTES

La sintonización de servomotores consiste en determinar la estabilidad relativa y afinar la ganancia en el lazo de control. Un sistema de control para servomotores se considera estable si la posición actual del eje es finita cuando la posición que se le ha ordenado también es finita, es decir que la posición que se le ordenó coincide con la posición que el eje del servomotor realizó. Por otra parte se considera inestable cuando la posición que se ordenó resulta en un incremento exponencial de error de posición. En otras palabras cuando un sistema inestable trata de llegar a la posición comandada, esta resulta en una oscilación que nunca termina.

El algoritmo de control PID se encuentra integrado por tres modos básicos P (Proporcional), I (Integral) y D (Derivativo) y cuando se usa esta técnica de control clásico, es necesario decidir cuales modos serán usados (P, I, o D) y especificar los parámetros o ajustes para cada uno. Generalmente los algoritmos básicos usados son P, PI o PID, y existiendo una amplia variedad de formas y métodos de ajuste de este esquema de control (Bolton, 2011).

Sin embargo, a medida que se exige una mayor precisión en el sistema, el ajuste de este tipo de control se hace más difícil sobre todo por el ruido y cuando se presentan retardos, además cuando los procesos a controlar son no lineales y el control debiera tener la capacidad de compensar esas no-linealidades, el control PID no tiene la capacidad de responder a esto porque asume relaciones lineales.

Por otro lado, el control lógico difuso basado en los trabajos desarrollados por Lofti Zadeh (Zadeh, 1965), también presenta características que lo hacen adecuado para él mismo fin, salvo que es capaz de ofrecer un comportamiento de control altamente flexible y ajustarse a condiciones cambiantes como condiciones de desgaste y cambios ambientales.

En este trabajo se diseñó un controlador PID basado en lógica difusa, esto fue motivado por la habilidad que tienen de capturar estrategias cualitativas de control y ofrecer un comportamiento altamente flexible.

1.1 Planteamiento del problema

En la actualidad existen controladores lógicos difusos para el control de velocidad, al igual que existen para el control de posición, cada uno funcionando de manera independiente.

En este trabajo se presenta el diseño y desarrollo de un controlador con la capacidad de sintonizar ambas cualidades de manera automática, utilizando control lógico difuso para sintonizar las funciones de membrecía y reglas.

Para tal efecto se planteó la siguiente pregunta:

¿Será posible la sintonización automática de posición y velocidad de un servomotor con controlador basado en lógica difusa?

1.2 Objetivo General

Diseñar un controlador para servomotores basado en lógica difusa, que sintonice de manera automática su velocidad y posición utilizando lógica difusa para la sintonización de las funciones de membresía.

1.3 Objetivos Específicos

1. Desarrollar el modelo matemático para el control lógico difuso.
2. Simular el modelo matemático en MATLAB.
3. Desarrollar el programa de control.
4. Implementar el sistema propuesto en un simulador para verificar su funcionamiento.

1.4 Justificación

Los servomotores tienen una infinidad de aplicaciones de tipo industrial principalmente en la construcción de robots, máquinas CNC y procesos que requieren control de movimiento exacto. Para el funcionamiento de estos dispositivos se utilizan controladores de tipo PID, los cuales dejan de funcionar eficientemente al responder ante cambios no-lineales provocando así pérdidas económicas. Por esta razón se justifica el desarrollar un controlador que integre la sintonización automática de posición y velocidad de un servomotor, utilizando lógica difusa.

1.5 Delimitación

El controlador lógico difuso utiliza un algoritmo basado en el modelo Mamdani para sintonizar las reglas de control. En cuanto a las funciones de membresía de entrada y salida, fueron propuestas en base a conocimiento del proceso a controlar.

El modelo matemático fue desarrollado en Matlab e implementado en Labview que cuenta con control y adquisición de datos en tiempo real, así como un laboratorio virtual de pruebas mismo que sirvió para la simulación.

El controlador fue desarrollado, implementado y probado solo a nivel de simulación.

1.6 Metodología

Se realizó una profunda investigación en diferentes fuentes como artículos científicos, tesis, libros, revistas, manuales técnicos, páginas de internet reconocidas, con el fin de concentrar la información que es indispensable conocer y tener disponible para el desarrollo de este trabajo de investigación.

Una vez recopilada esta información se utilizó el método retrospectivo para analizar y desarrollar los objetivos de esta tesis.

Para el desarrollo del modelo de inferencia difusa, primero se evaluaron y determinaron los algoritmos necesarios que intervienen en el controlador difuso.

En la siguiente etapa se desarrolló un programa computacional utilizando la herramienta Matlab, en donde se integraron los modelos matemáticos y las variables de inferencia del control.

Los métodos deductivos y experimentales fueron los pasos siguientes en donde, el controlador difuso fue implementado en Labview para realizar pruebas y compararlas con las de un sistema lineal

Por último el método sintético me ayudó a definir y organizar la información obtenida y formular mis conclusiones.

2 MARCO TEÓRICO Y ESTADO DEL ARTE

A continuación se presenta el estado del arte que sustenta este trabajo, comenzando con algunos trabajos realizados en el área de lógica difusa como objeto de investigación. Existen diferentes descripciones acerca del concepto de la lógica difusa. Aceves (2001), menciona que la lógica difusa es una teoría rigurosa, matemáticamente bien fundamentada, en donde se utiliza el concepto de verdad parcial y que además, proporciona una herramienta poderosa para aplicaciones en control y supervisión de procesos industriales. Este trabajo presenta una breve introducción acerca de la lógica difusa además de algunos puntos importantes a tomar en cuenta al momento de decidir utilizarla.

Una descripción más detallada la muestra Jiménez (2000). Ella menciona que la lógica difusa es un concepto de control, que permite deducir una acción a realizar a partir de datos imprecisos, similar al que tiene el razonamiento humano.

Por su parte Kouro & Musalem (2002), mencionan que el control difuso se puede expresar como un controla a través de palabras que interpretan el sentido común, en lugar de números, o bien sentencias en lugar de ecuaciones.

El primer trabajo aplicativo de la lógica difusa fue realizado en 1974 por el Profesor Mamdani de la escuela Queen Mary de Londres, quien aplicó las ideas básicas de Zadeh sobre una caldera de vapor. La primera gran aplicación fue el tren electromagnético de Sendai construido en 1986 por la sociedad Hitachi.

Desde entonces la lógica difusa ha sido implementada en muchas aplicaciones ya que presenta algunas características como lo mencionan Rezoug & Hamerlain (2009), dentro de las cuales las más relevantes son: no necesita de un modelo matemático; es relativamente simple, rápida y adaptable; puede alcanzar los objetivos de diseño. Por lo tanto, el control por lógica difusa es aplicado en sistemas no lineales, donde no es posible hacer la identificación de la planta por métodos convencionales (López & Muñoz, 2007).

Existen en la actualidad diferentes aplicaciones de controladores inteligentes dentro de los cuales está la lógica difusa. Es interesante ver las comparaciones que existen entre estos diferentes algoritmos de control, por ejemplo Alzate & Giraldo (2006), presentan un análisis de la estabilidad de los sistemas no lineales a partir del plano de fase. En esta investigación, se aplican diferentes esquemas de control inteligentes sobre un sistema no lineal, utilizando como técnicas de control el control difuso, las redes neuronales y el control neuro-difuso. Encontrando un mejor resultado al aplicar el control neuro-difuso.

Sala & Ariño (2009), muestran algunas de las desventajas de las metodologías difusas que han sido estudiadas y superadas recientemente en algunos aspectos, sin embargo muchos sistemas no lineales pueden ser modelados como sistemas difusos

de modo que el control difuso puede considerarse como una técnica para el control no lineal.

Si bien, la lógica difusa presenta aplicaciones simples como lo son en motores de CD, es aún más relevante la aplicación en robots de diferentes tipos como: industriales, didácticos, etc.

Yeong & Chung (1993) desarrollaron un controlador por lógica difusa para un robot manipulador basado en esquemas de sistemas de estructura variable.

Las reglas difusas y las funciones de membresía de entrada y salida del controlador difuso, son diseñadas para garantizar la estabilidad total, en este caso, realizan una simulación en un robot de dos grados de libertad. Además, mencionan que un robot manipulador es un caso de dinámica incierta debido a variaciones de carga, fricción y disturbios externos.

Por su parte Diaz & Acevedo (2008), abordan la aplicación de la lógica difusa como técnica de control para ser aplicada a maquinas eléctricas, específicamente en la velocidad de motores de inducción, tratando de mejorar las características de velocidad y recuperación cuando hay inducción de carga al motor, o frente a perturbaciones.

Coronel (2004), presenta el desarrollo de un controlador difuso para el control de la velocidad de un motor de corriente directa ante variaciones en el par de carga. El controlador difuso propuesto, establece un conjunto de reglas de inferencias basadas en el error, y se pudo observar que la recuperación de la velocidad ante la acción de un par de carga fue relativamente muy rápida.

La lógica difusa tiene una historia corta, pero un rápido crecimiento debido a su capacidad de resolver problemas relacionados con la incertidumbre de la información utilizando el conocimiento de los expertos. Además, proporciona un método formal para la expresión del conocimiento en forma entendible por los humanos. Estas cualidades le aseguran un amplio campo de aplicabilidad y un alto interés para las aplicaciones industriales, presentes y futuras.

2.1 Servomecanismo

De acuerdo a la terminología del estándar Industrial Japonés (JIS), un servomecanismo está definido como un mecanismo que usa la posición, dirección, u orientación de un objeto como una variable de proceso para controlar un sistema para seguir cualquier cambio en el valor objetivo.

Simplemente, un servomecanismo es un mecanismo de control que monitorea cantidades físicas como posiciones especificadas.

Un servo sistema se podría definir en más detalle como un mecanismo que:

- Se mueve a una velocidad especificada
- Coloca un objeto en determinada posición

Este sistema de Control Automático puede ser ilustrado con el siguiente diagrama de bloques (figura 2-1).

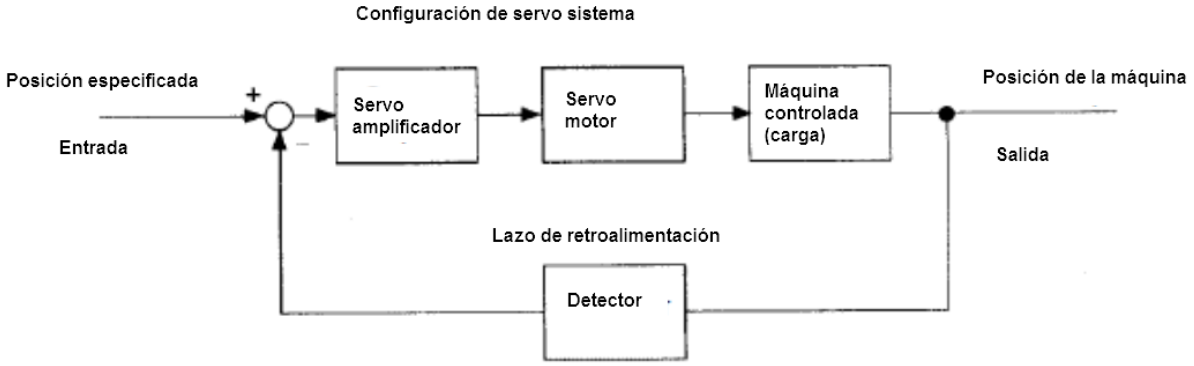


Figura 2-1 Sistema de control realimentado de posición.

Este servosistema es un sistema de control automático que detecta la posición de la máquina (datos de salida), realimenta los datos al lado de entrada, los compara con la posición especificada (datos de entrada) y mueve la máquina por la diferencia entre los datos comparados.

En otras palabras, el servosistema sirve para controlar los datos de salida para que coincidan con los datos de entrada especificados.

Posición, velocidad, fuerza (torque), corriente eléctrica, son valores controlados típicos para un servosistema.

El siguiente diagrama ilustra un servosistema en detalle (figura 2-2).

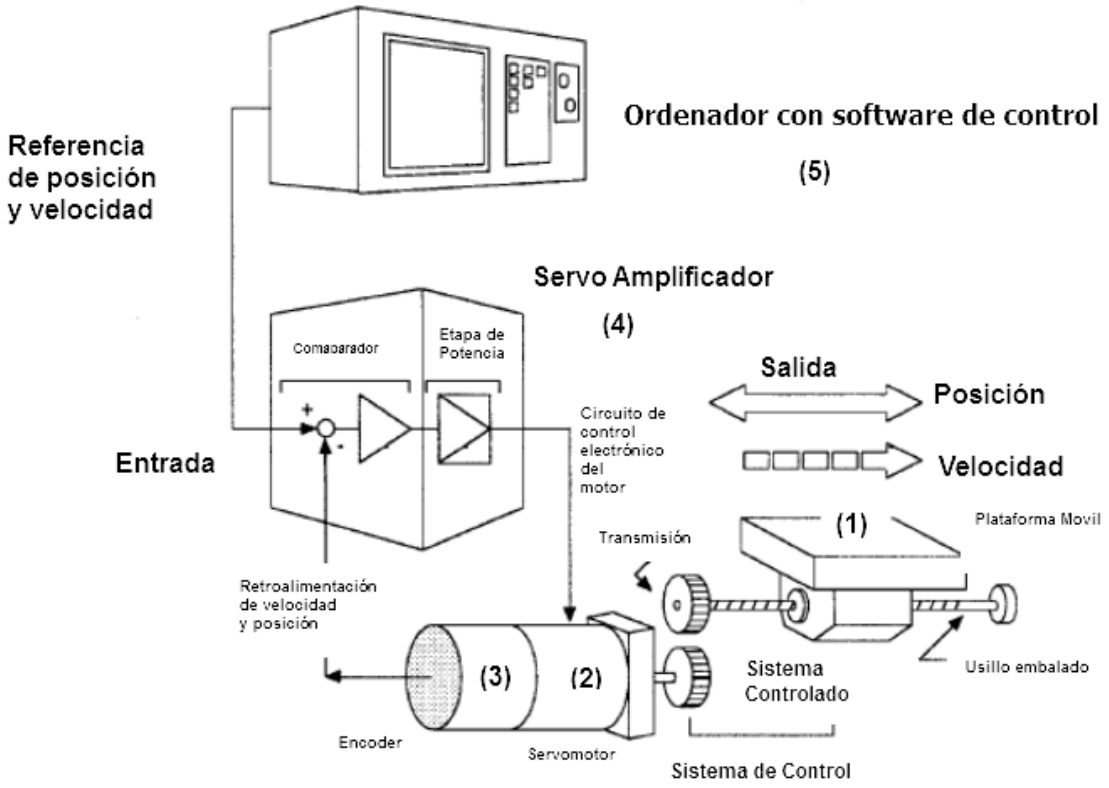


Figura 2-2 Servosistema.

1. Sistema controlado: sistema mecánico para el cual la posición o velocidad será controlada. Esto incluye un sistema de manejo que transmite torque desde un servomotor.

2. Servomotor: un actuador principal que mueve el sistema controlado. Están disponibles en dos tipos: AC y DC.

3. Encoder: un detector de posición o velocidad. Normalmente, un encoder montado en un motor es usado como un detector de posición.

4. Etapa de potencia: un amplificador que procesa la señal de error para corregir la diferencia entre la referencia y los datos de realimentación y opera el servomotor de acuerdo a ello. Un servo amplificador consiste de un comparador, el cual procesa las señales de error y un amplificador de potencia, el cual opera el servomotor.

5. Ordenador con software de control: un dispositivo que controla un servo amplificador especificando una posición o velocidad como un set point.

La figura 2-3 ilustra la estructura de un servomotor de tipo síncrono.

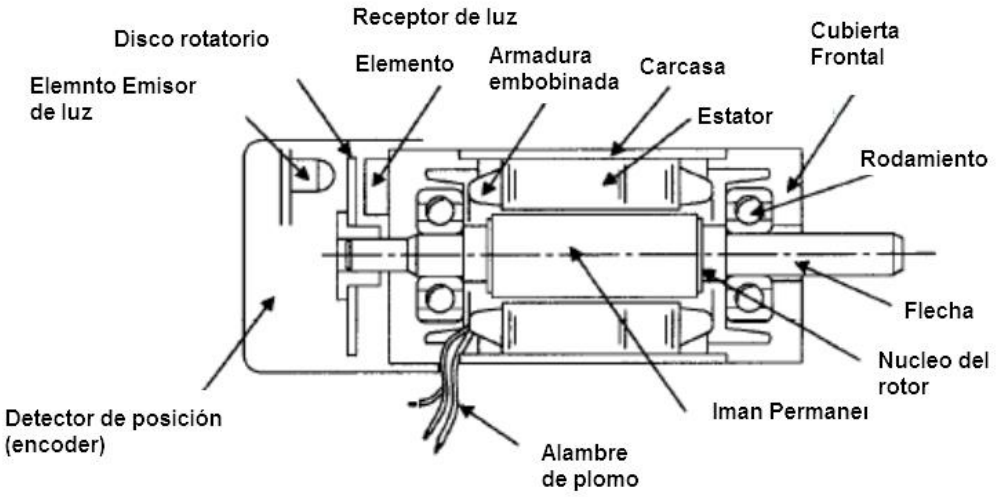


Figura 2-3 Motor AC síncrono.

El encoder es de tipo incremental el cual genera un cierto número de pulsos por revolución. Si este encoder está conectado al sistema mecánico y un pulso está definido como una cierta longitud (por ejemplo. 0.001mm) puede ser utilizado como un detector de posición.

Este encoder no detecta una posición absoluta y meramente da salida a un tren de pulsos. Una operación de retornar a cero debe ser ejecutada antes de posicionar. La siguiente figura ilustra el principio de operación de un generador de pulsos (método óptico) (figura 2-4).

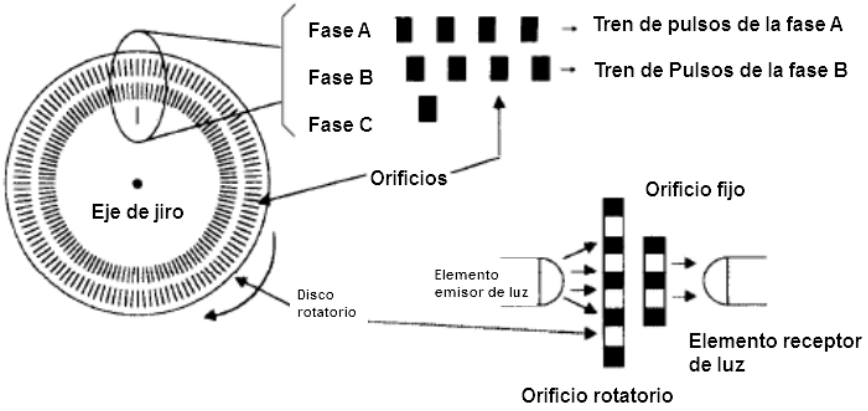


Figura 2-4 Encoder incremental.

Como controlador Huésped puede ser utilizado un PLC, una PC o un sistema basado en algún microcontrolador y además, es el dispositivo que contiene el programa de control y que normalmente es un control del tipo PID.

2.2 La lógica difusa

El primer concepto propuesto en lógica difusa fue en 1965, en la Universidad de California, Berkeley por Lofti A. Zadeh, quien combinó los conceptos de la lógica y de los conjuntos de Lukasiewicz mediante la definición de grados de pertenencia. Así, la Lógica Difusa ha cobrado una gran fama por la variedad de sus aplicaciones las cuales van desde el control de procesos industriales complejos hasta el diseño de dispositivos artificiales de deducción automática, pasando por la construcción de artefactos electrónicos de uso doméstico y de entretenimiento así como también de sistemas de diagnóstico.

La inteligencia artificial tiene varias ramas, entre ellas se encuentran la Lógica Difusa. A diferencia de la Lógica Convencional, en donde sólo son posibles los valores de falso o verdadero, la Lógica Difusa permite definir valores intermedios en un intento por aplicar un modo de pensamiento similar al del ser humano (Ley, Chacón L., & Vázquez , 2000).

El control difuso se utiliza para diseñar controladores para sistemas que son estructuralmente difíciles de modelar, debido a su naturaleza no lineal y a otras complejidades en la obtención del modelo. Durante los últimos años el Control Difuso ha emergido como una de las áreas de mayor investigación (Wang L.-X. , Adaptive Fuzzy System and Control, 1994).

La teoría difusa fue introducida por Lofti A. Zadeh en su trabajo de tesis “Conjuntos Difusos", antes de trabajar en la teoría difusa Zadeh se dedicó a la teoría de Control. A principios de los 60s Zadeh pensó que la teoría de control clásico puso demasiado énfasis en la precisión y por tanto no podía manejar los sistemas complejos.

Al comienzo las ideas publicadas por Zadeh no fueron seguidas por la comunidad científica del momento, pero con el tiempo comenzó a tener seguidores lo que produjo que sus teorías fuesen ampliadas y se asentaran sus conocimientos.

La intención de Zadeh, fue la creación de un formalismo para manejar de manera más eficiente la imprecisión del razonamiento humano.

En 1971 realiza la publicación de "Quantitative Fuzzy Semantics”, en donde aparecen los elementos formales que dan lugar a la metodología de la Lógica Difusa y de sus aplicaciones tal y como se conocen en la actualidad.

A partir de 1973 otros investigadores comenzaron a aplicar la Lógica Difusa a diversos procesos haciendo grandes aportaciones tanto al desarrollo de la teoría de Lógica Difusa como al estudio de sus aplicaciones.

En 1974 Assilian y Mamdani en el Reino Unido desarrollaron el primer controlador difuso diseñado para la máquina de vapor. La implementación real de un controlador de este tipo no fue realizada hasta 1980 por F.L. Smidth&Co. en una planta cementera en Dinamarca.

En 1987 Hitachi usa un controlador fuzzy para el control del tren de Sendai, el cual usa uno de los sistemas más novedosos creados por el hombre.

En 1991 Karr, utiliza un algoritmo genético simple para sintonizar un sistema difuso, utilizando un enfoque fuera de línea, sin utilizar la base de "reglas de inferencia" (C.L., 1991)

Zhang y Li en el año 2006 obtuvieron la base de un sistema difuso utilizando un algoritmo genético, para controlar la temperatura de un horno (Jing Yuan & Li, 2006).

En 1993, Fuji aplica la "Lógica Difusa" para el control de inyección química en plantas depuradoras de agua por primera vez en Japón. Ha sido precisamente aquí, en donde más apogeo ha tenido la "Lógica Difusa".

De forma paralela al desarrollo de las aplicaciones de la Lógica difusa, Takagi y Sugeno desarrollan la primera aproximación para construir reglas difusas a partir de datos de entrenamiento (observación).

En 2007, es publicado el artículo: "Control de un motor utilizando lógica difusa con reglas sintonizadas por algoritmos genéticos" (López A., Muñoz A., & Cardona E., 2007). En este documento se describe la implementación de un controlador PI difuso incremental con una base de reglas para un motor de corriente continua controlado por campo. La matriz de reglas es sintonizada a través de algoritmos genéticos. El controlador está diseñado para llevar el motor desde una condición inicial de 0 a 100rpm (revoluciones por minuto). Las simulaciones y los resultados obtenidos muestran un comportamiento óptimo del sistema.

En este mismo año, es presentado el trabajo de tesis: "Diseño de un controlador lógico difuso aplicado al control de posición de un servomotor de CD, utilizando un algoritmo Genético simple" (Madrigal, 2007). En el cual se utilizó un algoritmo genético para obtener la base de reglas de inferencia difusas para obtener la mejor combinación de las variables difusas de entrada y salida, dentro de un rango permisible.

La lógica difusa es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada vaga, ambigua, imprecisa, con ruido o incompleta, en general la lógica difusa imita como una persona toma decisiones basada en información con las características mencionadas. Una de

las ventajas de la lógica difusa es la posibilidad de implementar sistemas basados en ella tanto en hardware como en software o en combinación de ambos.

Se fundamenta en los denominados conjuntos difusos y un sistema de inferencia difuso basado en reglas de la forma “SI…ENTONCES…..”

En contraste con la lógica tradicional, que utiliza conceptos absolutos para referirse a la realidad, la lógica difusa se define en grados variables de pertenencia a los mismos, siguiendo patrones de razonamiento similares a los del pensamiento humano.

Así por ejemplo, mientras dentro del marco rígido de la lógica tradicional o formal un recinto está solamente "oscuro" (0) o claro (1), para la lógica difusa son posibles también todas las condiciones relativas intermedias percibidas por la experiencia humana como "muy claro", "algo oscuro", "ligeramente claro", "extremadamente oscuro", etc. Las condiciones extremas o absolutas asumidas por la lógica tradicional son sólo un caso particular dentro del universo de la lógica difusa. Esta última nos permite ser relativamente imprecisos en la representación de un problema y aun así llegar a la solución correcta.

2.2.1 La lógica Difusa en la Inteligencia Artificial

Método de razonamiento de maquina similar al pensamiento humano, que puede procesar información incompleta o incierta, característico de muchos sistemas expertos.

Con la lógica difusa o borrosa se puede gobernar un sistema por medio de reglas de “sentido común” las cuales se refieren a cantidades indefinidas. En general la lógica difusa se puede aplicar tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología o economía.

2.2.2 Características

1.- Se basa en palabras y no en números, las verdades de los valores son expresados lingüísticamente. Por ejemplo: caliente, muy frío, verdad, lejano, cercano, rápido, lento, medio, etc.

2.- Ésta genera algunos modificadores del predicado como por ejemplo: mucho, más o menos, poco, suficientemente, medio, etc.

3.- También procesa un sistema amplio de cuantificadores, como por ejemplo: pocos, varios, alrededor, generalmente.

4.- Hace uso las probabilidades lingüísticas, como por ejemplo: probable, improbable, que se interpretan como números borrosos y son manipuladas por su aritmética.

5.- Maneja todos los valores entre 0 y 1, tomando éstos como límite solamente.
Así podemos decir que la Lógica Difusa de acuerdo a sus características:
• Usa una representación de conocimiento explícito.
• Realiza verificación y optimización de manera fácil y eficiente.
• No se puede entrenar, esto es que sea capaz de obtener nuevos conocimiento.

2.2.3 Etapas de la Lógica Difusa.

1 Fuzzificación (Fuzzification). Las funciones de pertenencia definidas para las variables de entrada se aplican a sus valores actuales correspondientes, para poder determinar el grado de verdad para cada regla de la premisa.

2 Inferencia Lógica. El valor de verdad para la premisa de cada regla se calcula, y aplica a la parte de conclusiones de cada regla. Este resultado se asigna a un subconjunto difuso para ser asignado a cada variable de salida para cada regla.

3 Defuzzificación (Defuzzification). La cual es usada cuando se desea convertir la salida difusa en un valor puntual numérico. Existen muchos métodos de Defuzzificación (al menos 30).

2.2.4 Aplicaciones

Actualmente existen todo tipo de instrumentos, máquinas y procedimientos controlados borrosamente, adaptándose "inteligentemente" a cada situación particular: acondicionadores de aire, frigoríficos, lavadoras/secadoras, aspiradoras, hornos microondas, mantas eléctricas, ventiladores, autoenfoques fotográficos, estabilizadores de imágenes en grabadoras de vídeo, transmisiones de automóviles, suspensiones activas, controles de ascensores, dispensadores de anticongelantes para los aviones en los aeropuertos, sistemas de toma de decisiones industriales o económicas, y un largo etcétera.

Y así las aplicaciones las podríamos dividir en categorías por ejemplo:

Productos al consumidor:

• Lavadoras
• Hornos de microondas
• Procesadores de arroz
• Limpiadores al vacío
• Cámaras de video
• Televisores

• Sistemas térmicos
• Traductores. Es también utilizada en algunos correctores de voz para sugerir una lista de probables palabras a reemplazar en una mal dicha.

El control difuso, puede ser expresado mejor como un control a través de palabras que interpretan el sentido común, en lugar de números, o bien sentencias en lugar de ecuaciones.

Sin embargo, las variables de los procesos no se miden en sentido común, sino en números. Por lo tanto se hace necesario realizar una adaptación previa antes de introducir el estado de la variable al controlador. Esta etapa es llamada fuzzificación.

En la figura 2-5, se aprecian las distintas transformaciones que sufren las variables y los datos en un lazo de control difuso.

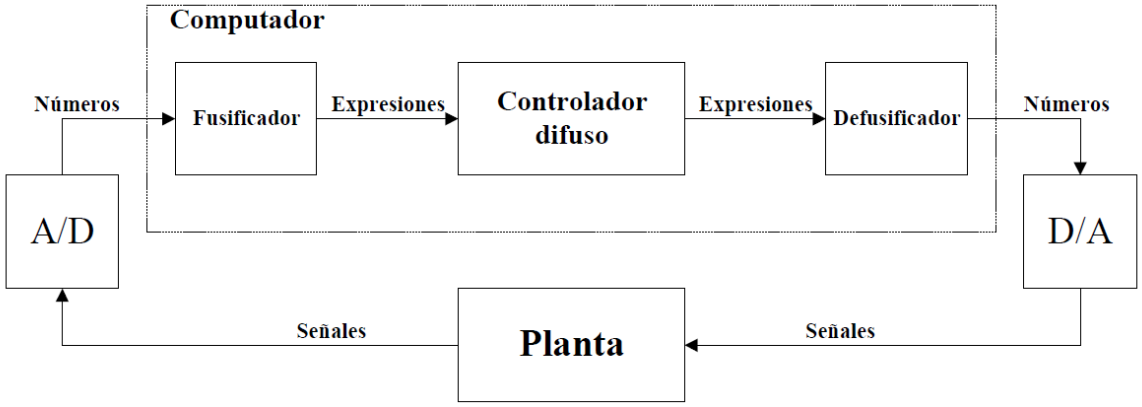


Figura 2-5 Transformaciones de Variables y Datos en Control Difuso.

2.3 Implicación de Mamdani

El método de Mamdani es el más usado en aplicaciones, dado que tiene una estructura muy simple de operaciones “mín-max”.solo utilizando los siguientes pasos.

Paso 1. Evaluación del antecedente en cada regla.

Cuando se tienen entradas (valores numéricos) se pueden obtener distintos valores de pertenencia para cada valor. A esto se le llama “fuzzificación de la entrada”. Si el antecedente de la regla tiene más de un término, a continuación se aplica algún operador (t-norma o t-conorma) obteniendo un único valor de pertenencia. Veamos el ejemplo (figura 2-6).

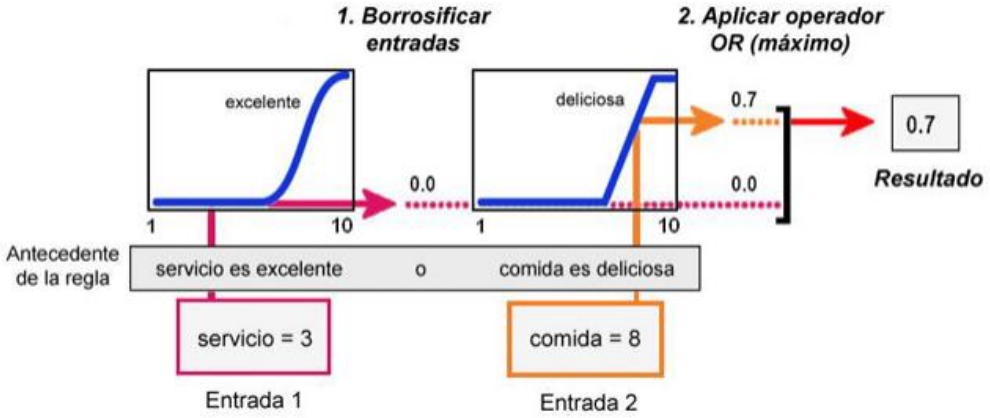


Figura 2-6 Proceso de fuzzificación de los Datos.

Al fuzzificar el primer término del antecedente (servicio es excelente) hemos visto en qué grado el servicio es excelente si al servicio le puntuamos con un 3. Como vemos, un 3 corresponde a un servicio nada excelente, de ahí que obtengamos el valor de pertenencia 0. Al fuzzificar el segundo término del antecedente (comida es deliciosa) hemos visto en qué grado la comida es deliciosa si la puntuamos con un 8. Lógicamente, un 8 corresponde a una comida bastante deliciosa, de ahí que obtengamos el valor de pertenencia 0,7.

Por último, ya que los dos términos del antecedente están unidos por una disyunción (servicio excelente o comida deliciosa), hemos aplicado un operador borroso OR, en este caso el máximo, a los dos valores de pertenencia anteriores obteniendo el valor de pertenencia 0.7. Si los términos del antecedente estuvieran unidos por una conjunción ("y"), habría que aplicar un operador borroso AND, por ejemplo el mínimo.

NOTA: Los operadores de negación y modificador se pueden aplicar a cualquier término.

Paso 2. Obtener la conclusión en cada regla.

A partir del consecuente de cada regla y del valor del antecedente obtenido en el paso 1, aplicamos un operador borroso de implicación obteniendo así un nuevo conjunto borroso.

Dos de los operadores de implicación más usados son el mínimo, que trunca la función de pertenencia del consecuente, y el producto, que la escala. En el siguiente gráfico correspondiente al ejemplo del restaurante, se usa el mínimo (figura 2-7).

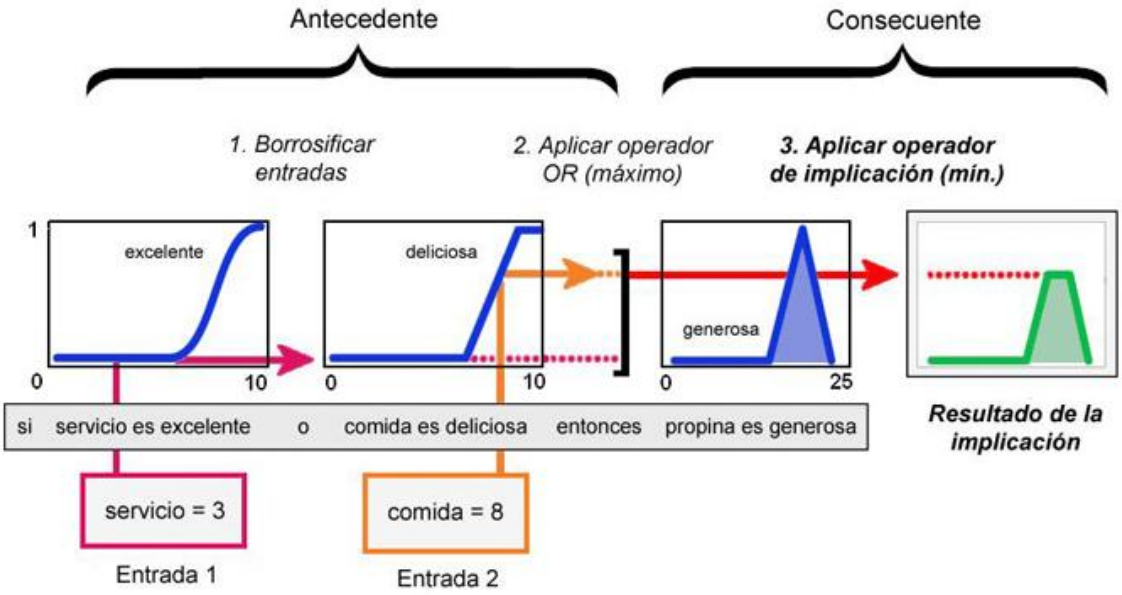


Figura 2-7 Ejemplo de Aplicación del proceso de Fuzzificación.

Paso 3. Agregar conclusiones.

Las salidas obtenidas para cada regla en el paso 2 (obtener conclusión), se combinan en un único conjunto borroso utilizando un operador de agregación borrosa.

Algunos de los operadores de agregación más utilizados son el máximo, la suma o el “or” probabilístico. En el siguiente gráfico correspondiente al ejemplo del restaurante se usa el máximo (figura 2-8).

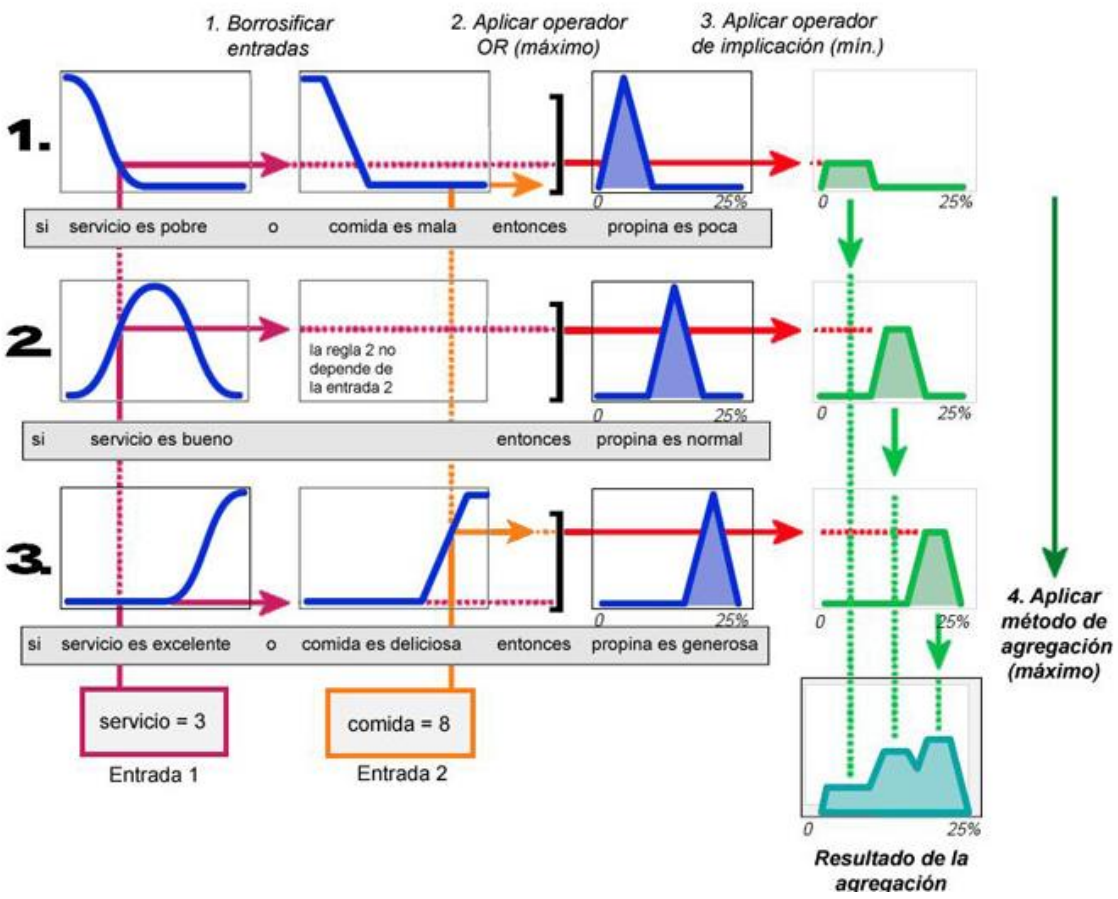


Figura 2-8

Paso 4. Defuzzificación (defuzzify).

Cuando queremos obtener una solución a un problema de decisión, lo que queremos obtener como salida es un número y no un conjunto borroso. Siguiendo el ejemplo del restaurante, no sirve que nos digan que demos una propina generosa, lo que queremos es que nos digan qué propina tenemos que dar. Por tanto, tenemos que transformar el conjunto borroso obtenido en el paso 3 en un número. Uno de los métodos más utilizados es el del centroide, que calcula el centro del área definida por el conjunto borroso obtenido en el paso 3.

El cálculo se muestra en el siguiente gráfico (figura 2-9).

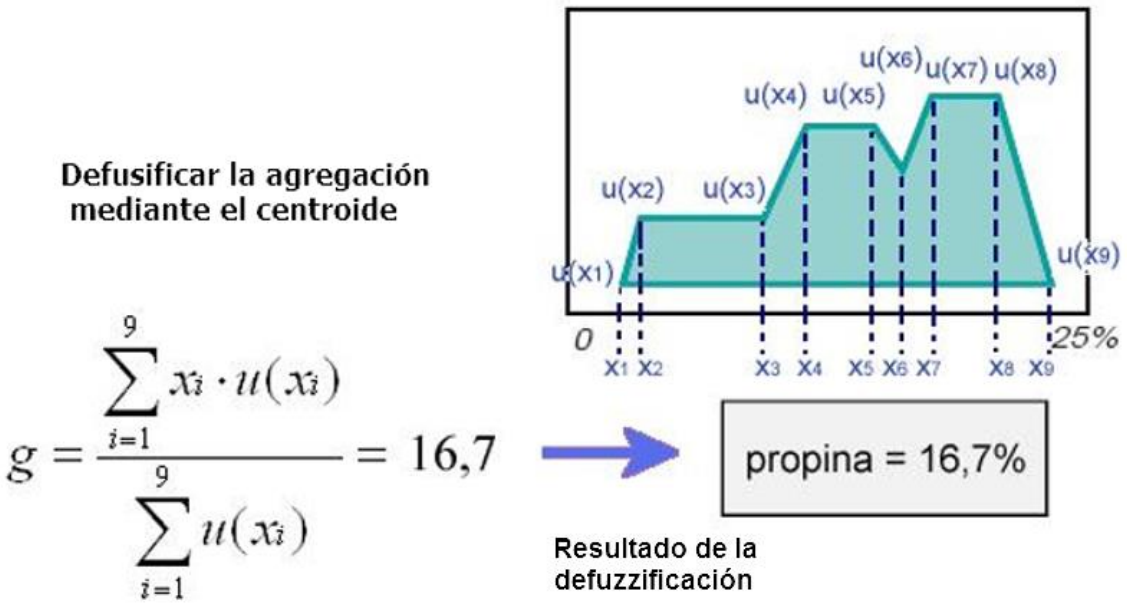


Figura 2-9 Método del Centroide para aplicar la Defuzzificación

En la figura 2-10, podemos ver un ejemplo de todos los pasos del método aplicados a un problema de un restaurante.

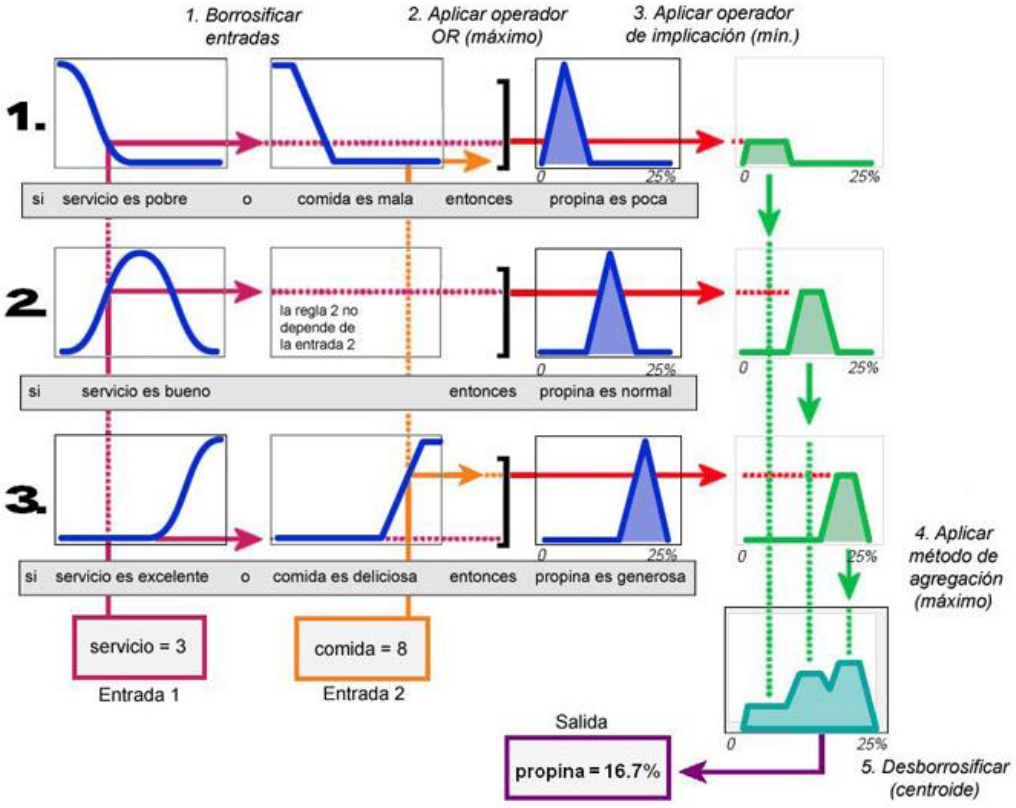


Figura 2-10 Ejemplo práctico de aplicación del Método de Mamdani.

Cabe destacar que el método de Mamdani es útil cuando tenemos un número reducido de variables. En caso contrario, nos encontramos con los siguientes inconvenientes:

- El número de reglas aumenta exponencialmente con el número de variables en la parte premisa.
- Cuantas más reglas haya que construir, más difícil es saber si son adecuadas.
- Si el número de variables en la parte premisa es demasiado grande, será difícil comprender la relación causal entre las premisas y las consecuencias y por tanto, serán difíciles de construir las reglas. (Sanjay , 2014)

2.4 Conjuntos Difusos

La lógica difusa trabaja con conjuntos a los cuales llamamos conjuntos difusos, estos conjuntos están definidos por sus funciones de pertenencia, la cual expresa la distribución de verdad de una variable.

Un conjunto difuso se puede definir matemáticamente al asignar a cada posible individuo que existe en el universo de discurso, un valor que representa su grado de pertenencia o membresía en el conjunto difuso. Este grado de membresía indica cuando el elemento es similar o compatible con el concepto representado por el conjunto difuso.

Ejemplo: Supongamos que tenemos tres conjuntos de tipos de estaturas de personas con valores dentro de los siguientes rangos:

Baja = (130-160 cm.)
Media = (160-180 cm.)
Alta = (180-200 cm.)

En la figura 2-11 a), tenemos una representación utilizando conjuntos clásicos. En esta representación una persona que mida 180 cm. Es considerada de estatura media, y en cambio una persona con 181 cm. Ya no. Vemos que esto no corresponde con la realidad.

En la figura 2-11 b), se observa que existe un grado de pertenencia a los conjuntos difusos. Así, una persona que tenga una estatura de 172 cm. Pertenecería en un 40% (MEDIA (172)=0’4) al conjunto de personas con estatura media y en un 10% (ALTA (172)=0’10) al de estatura alta (Llorens, 2000).

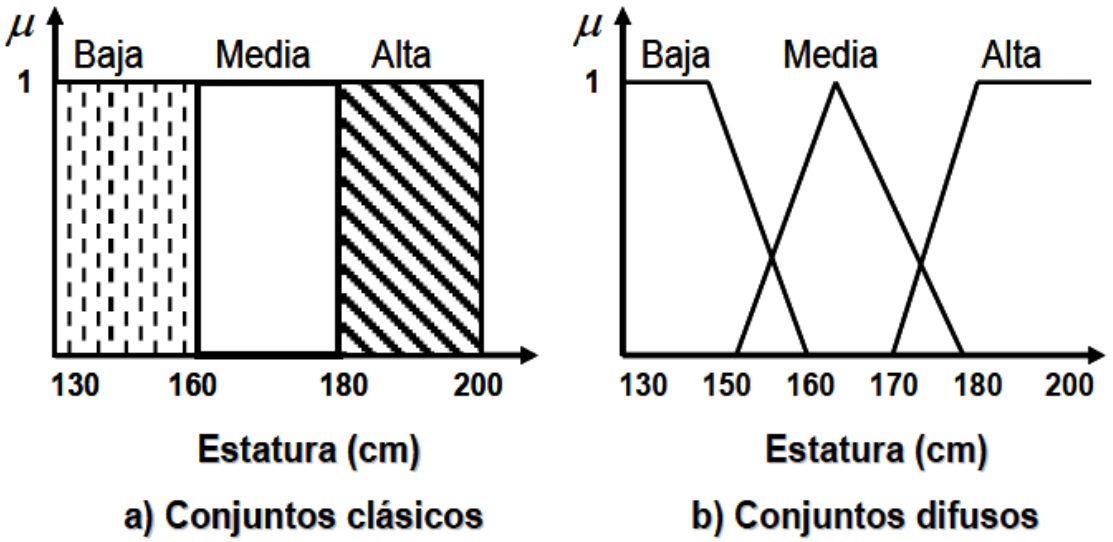


Figura 2-11 Función de tipo escalón

2.5 Conjuntos Clásicos

Los conjuntos clásicos surgen a partir del razonamiento humano por clasificar objetos y conceptos. Por ejemplo la necesidad de clasificar los alimentos en frutas, verduras, carnes rojas, carnes blancas, y de todos los alimentos existentes definir que alimento pertenece a cierto grupo.

Los conjuntos clásicos puedes definirse a partir de varias formas como:

Listado de sus elementos

Fruta: [manzana, Sandia y pera]

Mediante una función de pertenencia μ que toma valores 0 o 1 definida sobre el Universo de discurso U (todos los elementos que pueden o no pertenecer al Conjunto):

Ejemplo: sea U el conjunto de todos los alimentos. Entonces Frutas es un Conjunto tal que μ (manzana)=1, μ (calabaza)=0, etc.

De este modo, para definir un conjunto nítido A podemos utilizar la función de pertenencia dada por:

$$\mu A\,(x) = \begin{cases} 0 \text{ si } x \notin A \\ 1 \text{ si } x \in A \end{cases}$$

Es decir, una función tipo escalón centrada en el valor/valores umbral/umbrales de decisión (Figura 2-12).

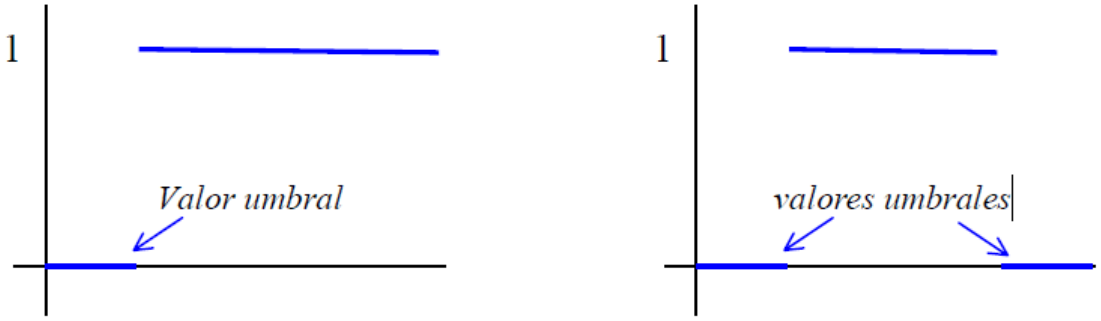


Figura 2-12 Función de tipo escalón

Si se utilizan funciones de pertenencia, la forma de representar el vacío y el conjunto universo será:

El vacío ∅ es un conjunto tal que para todo x de U, μ(x)=0.
El conjunto universo es tal para todo x de U, μ (x)=1.
• *Dando una característica que defina sus elementos*. Esto se puede hacer de varias formas:

Dando directamente la definición:

Fruto = Producto del desarrollo del bulbo de una flor después de la polinización.

Como un subconjunto de un conjunto ya definido:
Frutas = Fruto comestible.

2.6 Sistema de control lineal

Los sistemas controlados han estado evolucionando de forma acelerada en los últimos días y hoy en día pasan desapercibidos para mucha gente pues presentan pocos o ningún problema, las técnicas de control se han mejorado a través de los años, sin embargo es muy importante que se conozca la teoría básica de control. El trabajo pretende formar parte de la educación del alumno en la teoría básica de control siendo una herramienta que puede facilitar el estudio en el laboratorio.

El control automático desempeña una función vital en el avance de la ingeniería y la ciencia, ya que el control automático se ha vuelto una parte importante e integral de los procesos modernos industriales y de manufactura. Por lo cual la teoría de control es un tema de interés para muchos científicos e ingenieros que desean dar nuevas ideas para obtener un desempeño óptimo de los sistemas dinámicos y disminuir tareas manuales o repetitivas.

Las definiciones básicas de los sistemas de control son el punto de partida para comprender el estudio. Estas definiciones surgieron usando como base las ideas de la autora Katsuhiko Ogata en su libro Ingeniería de Control Moderna.

Variable controlada y variable manipulada: la variable controlada es la cantidad o condición que se mide y controla. La variable manipulada es la cantidad o condición que el controlador modifica para afectar el valor de la variable controlada. El objetivo del control es medir el valor de la variable controlada del sistema para aplicar correcciones a través de la variable manipulada para obtener un valor deseado.

Planta: la planta normalmente es un conjunto de partes que trabajan juntas con el objetivo de realizar una operación en particular. Se le llama planta a cualquier sistema físico que se desea controlar.

Proceso: el proceso es cualquier operación que va a ser controlada.

Sistema. Un sistema es un conjunto de componentes que se interrelacionan y trabajan juntos para realizar un objetivo determinado.

Perturbación: una perturbación es una señal que normalmente afecta a la variable controlada del sistema. Las perturbaciones pueden ser internas cuando surgen dentro del sistema, o externas porque se produce fuera del sistema y actúan como otra entrada.

Control realimentado: el control realimentado es un sistema que mantiene una comparación entre la entrada de referencia y la salida deseada, el resultado de la comparación es utilizado para controlar.

Sistema de control en lazo cerrado: los sistemas de control en lazo cerrado alimentan al controlador la señal de error de actuación que es la diferencia entre la señal de entrada y la señal de realimentación, a fin de reducir el error y llevar la salida del sistema a un valor deseado. El término control de lazo cerrado siempre implica el uso de una acción de control realimentado para reducir el error del sistema; es por eso que el termino control de lazo realimentado y de lazo cerrado se usan indistintamente.

Sistemas de control en lazo abierto: en estos sistemas la salida no afecta la acción de control. Es decir que en este tipo de control no se mide la salida ni se realimenta para compararla con la entrada. Por lo tanto, a cada entrada de referencia le corresponde una condición operativa fija; lo que obliga a que la precisión del sistema sea dependiente de la calibración del mismo.

2.7 Control PID

El control PID (Proporcional- Integral- Derivativo). Con casi 60 años de antigüedad sigue siendo usado e aplicaciones industriales, ya que ofrece una alternativa sencilla para controlar procesos dinámicos, ofreciendo tiempos de respuesta rápidos.

A sido muy popular y logrado sobrevivir gracias a su flexibilidad y adaptabilidad a condiciones cambiantes ya sea ambientales o de desgaste como sus siglas lo dicen el control PID consiste en tres modos básicos. El proporcional, el integral y el derivativo. Cuando se requiere usar un control clásico se tiene que decidir cuál de los modos del control PID se necesita, ya sea P, PI, PID. Estas combinaciones de modos surgen de la precisión que exija el sistema.

Una de las problemáticas es que cuando se requiere calibración llega a ser más difícil en sistemas no lineales ya sea por el ruido o por retardos.

El control lógico difuso surge en los años 60 del trabajo de Lofti A. Zadeh como una herramienta para procesos complejos. Ya que permite responder a parámetros variable y no solo a variables exactas. Un parámetro exacto seria frio o caliente, mientras que un intermedio seria medio frio. Esto permitió que la lógica de un proceso se pareciera más a la lógica humana. Esto permitió que los sistemas se ajustaran a medios o variables cambiantes (Lorandi, 2011).

2.8 Labview

Laboratory Virtual Instrument Engineering Workbench (Labview), es un lenguaje de programación grafico que utiliza iconos en vez de líneas de código para crear programas. Es ampliamente usado para hacer pruebas, medición y control en una gran variedad de procesos industriales.

Labview fue lanzado por primera vez en 1986, y ha venido evolucionando hasta alcanzar otros campos relacionados como la utilización de FPGA´s, microcontroladores (PIC, AVR), y tarjetas de desarrollo como la gama ARDUINO.

La instrumentación virtual permite a los diseñadores el desarrollo de diagramas de bloques de manera más fácil y rápida para cualquier tipo de aplicación, ya sea de análisis, adquisición de datos o control (figura 2-13).

Esta característica permite la simplificación del control de procesos, la computación científica, investigación, aplicaciones industriales y medición, por contar con un lenguaje de programación flexible y combinado con una gran cantidad de herramientas especialmente diseñadas para estas aplicaciones.

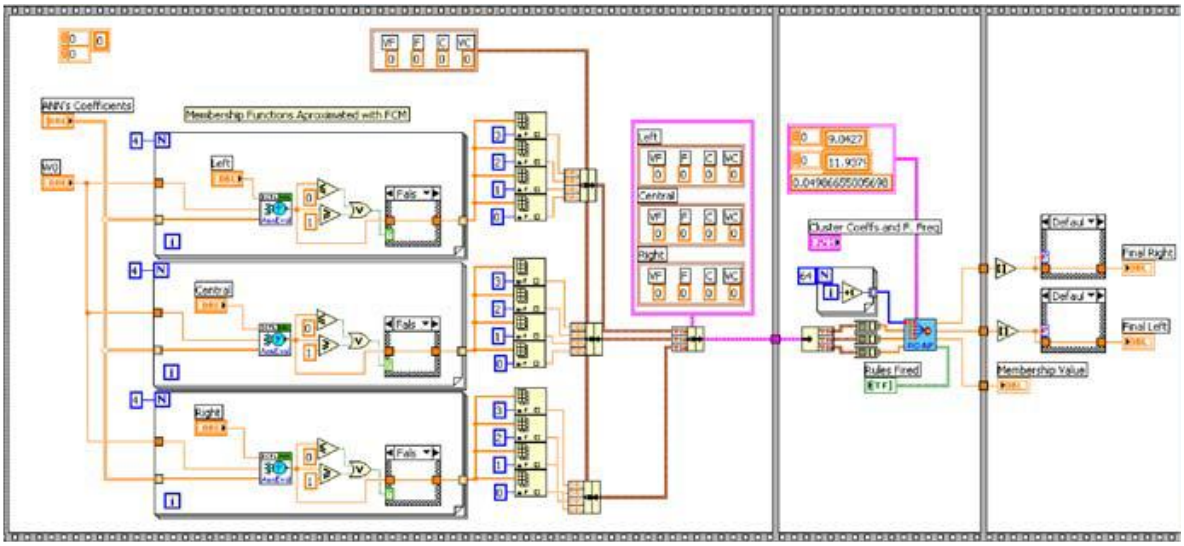


Figura 2-13 Diagrama de bloques del Kit de herramientas de Control Inteligente.

Los programas que simulan instrumentos virtuales creados en Labview se llaman VI (Virtual Instrument). El VI tiene tres componentes básicos: el panel frontal, el diagrama de bloques, y el conector icono. El panel de control es la interfaz de usuario y el código está dentro del diagrama de bloques.

Las figuras 2-14 y 2-15 muestran el diagrama de bloques y el panel frontal de un ejemplo VI de Control Toolkit inteligente.

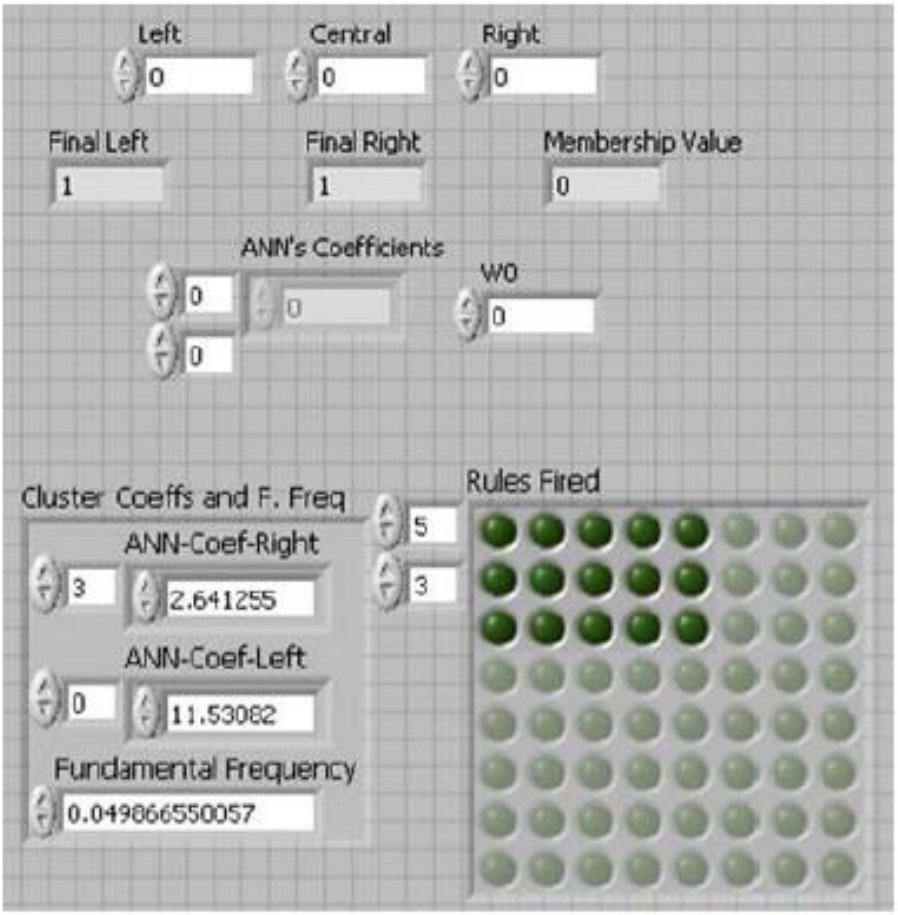


Figura 2-14 Panel Frontal

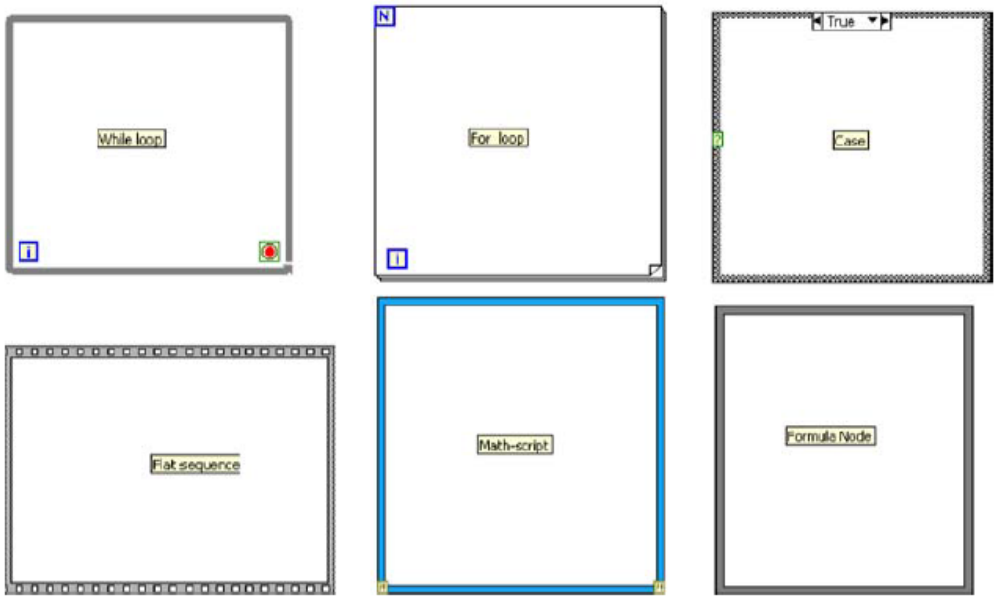


Figura 2-15 Bloque de lazos de control

En el panel frontal, se puede agregar el número de entradas y salidas que el sistema requiere. Los elementos básicos en el interior del panel frontal se pueden clasificar por controles e indicadores. El tipo general de datos numéricos pueden ser enteros, flotantes y números complejos. Otro tipo de datos es el de Boole, útil en sistemas condicionales (verdaderas o falsas), así como cadenas, que son una secuencia de caracteres ASCII.

Con el uso de lazos de control, es posible repetir una secuencia de programas o crear programas condicionantes. Los bucles de control utilizados se muestran en la (figura 2-15).

También es posible analizar las salidas de los sistemas inteligentes mediante un diagrama de forma de onda, por el trazado de los datos de salida.

La Figura 2-16, muestra un diagrama de forma de onda utilizada para el análisis de señales de salida.

En el caso de las señales de entrada, que son representaciones del fenómeno físico, uno podría obtener la información mediante un sistema de adquisición de datos. Uno de los objetivos principales en el sistema de adquisición de datos para la obtención de un sistema exitoso, es la selección del sistema mismo, así como el transductor y los sensores. Los sistemas de adquisición de datos, juegan un papel muy importante en el diseño de sistemas de control inteligente. National Instrument es una de los fabricantes más importante en este ámbito con amplia gama de dispositivos (figura 2-17)

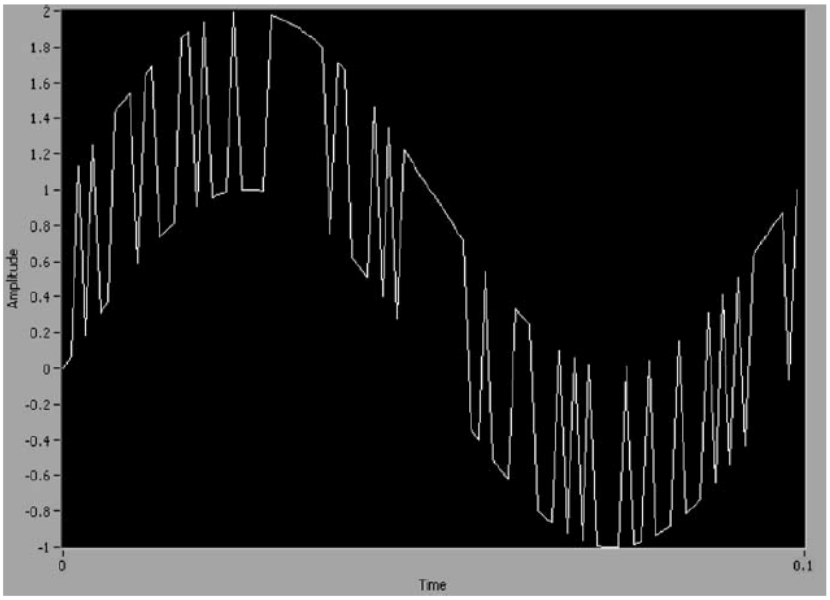


Figura 2-16 Diagrama de forma de onda

a b

Figura 2-17 Sistemas de adquisición de datos desarrollados por NI a CompactRIO. b NIUSB DAQ

2.9 Matlab

Matlab es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros de Control en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa.

La manera más fácil de visualizar Matlab es pensar en él como en una calculadora totalmente equipada, aunque, en realidad, ofrece muchas más características y es mucho más versátil que cualquier calculadora. Matlab es una herramienta para hacer cálculos matemáticos. Es una plataforma de desarrollo de aplicaciones, donde conjuntos de herramientas inteligentes para la resolución de problemas en áreas de aplicación específica, a menudo llamadas toolboxes, se pueden desarrollar con facilidad relativa.

MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

El nombre de MATLAB proviene de la contracción de los términos MATrix LABoratory y fue inicialmente concebido para proporcionar fácil acceso a las librerías LINPACK y EISPACK, las cuales representan hoy en día dos de las librerías más importantes en computación y cálculo matricial.

MATLAB es un sistema de trabajo interactivo cuyo elemento básico de trabajo son las matrices. El programa permite realizar de un modo rápido la resolución numérica de problemas en un tiempo mucho menor que si se quisiesen resolver estos mismos problemas con lenguajes de programación tradicionales como pueden ser los lenguajes Fortran, Basic o C.

MATLAB goza en la actualidad de un alto nivel de implantación en escuelas y centros universitarios, así como en departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales. En entornos universitarios, por ejemplo, MATLAB se ha convertido en una herramienta básica, tanto para los profesionales e investigadores de centros docentes, como una importante herramienta para la impartición de cursos universitarios, tales como sistemas e ingeniería de control, álgebra lineal, proceso digital de imagen, señal, etc. En el mundo industrial, MATLAB está siendo utilizado como herramienta de investigación para la resolución de complejos problemas planteados en la realización y aplicación de modelos matemáticos en ingeniería.

Los usos más característicos de la herramienta los encontramos en áreas de computación y cálculo numérico tradicional, prototipaje algorítmico, teoría de control automático, estadística, análisis de series temporales para el proceso digital de señal.

MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyo especializados denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal.

Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc. Además también se dispone del programa Simulink que es un entorno gráfico interactivo con el que se puede analizar, modelizar y simular la dinámica de sistemas no lineales.

2.9.1 Uso de Matrices

Matlab emplea matrices porque con ellas se puede describir infinidad de cosas de una forma altamente flexible y matemáticamente eficiente. Una matriz de pixeles puede ser una imagen o una película. Una matriz de fluctuaciones de una señal puede ser un sonido o una voz humana. Y tal vez más significativamente, una matriz puede describir una relación lineal entre los componentes de un modelo matemático. En este último sentido, una matriz puede describir el comportamiento de un sistema extremadamente complejo. Por ejemplo una matriz puede representar el vuelo de un avión a 40.000 pies de altura, o un filtro digital de procesamiento de señales.

2.9.2 Origen de Matlab

Matlab fue originalmente desarrollado en lenguaje FORTRAN para ser usado en computadoras mainframe. Fue el resultado de los proyectos Linpack y Eispack desarrollados en el Argonne National Laboratory. Su nombre proviene de MATrix LABoratory. Al pasar de los años fue complementado y re-implementado en lenguaje C. Actualmente la licencia de Matlab es propiedad de MathWorks Inc.

2.9.3 Plataformas

Matlab está disponible para un amplio número de plataformas: estaciones de trabajo SUN, Apollo, VAXstation y HP, VAX, MicroVAX, Gould, Apple Macintosh y PC AT compatibles 80386 o superiores. Opera bajo sistemas operativos UNIX, Macintosh y Windows.

2.9.4 Productos

La empresa MathWorks ofrece Matlab como su principal producto para computación numérica, análisis y visualización de datos. También ofrece Simulink como un anexo a Matlab y que interactúa con él en lenguaje de Matlab y lenguaje de bajo nivel C. Simulink es usado para simulación modelado no lineal avanzado. Se ofrecen además numerosas herramientas especiales en "Toolboxes" para resolver problemas de aplicaciones específicas, por ejemplo control, procesamiento de señales, redes neurales, etc. Estas herramientas son colecciones de rutinas escritas en Matlab.

2.9.5 Librería de Aplicaciones de MATLAB

2.9.5.1 Signal Processing Toolbox

MATLAB tiene una gran colección de funciones para el procesamiento de señal en el Signal Processing Toolbox. Este incluye funciones para:

Análisis de filtros digitales incluyendo respuesta en frecuencia, retardo de grupo, retardo de fase.

Implementación de filtros, tanto directo como usando técnicas en el dominio de la frecuencia basadas en la FFT.

Diseño de filtros IIR, incluyendo Butterworth, Chebyschev tipo I, Chebyshebv tipo II y elíptico.

Diseño de filtros FIR mediante el algoritmo óptimo de Parks-McClellan.

Procesamiento de la transformada rápida de Fourier FFT, incluyendo la transformación para potencias de dos y su inversa, y transformada para no potencias de dos.

2.9.5.2 Desarrollo de aplicaciones utilizando la MATLAB C Math Library

La construcción y desarrollo de aplicaciones utilizando esta librería es un proceso de amplias perspectivas una vez se tiene un dominio adecuado de su operativa. El producto está dividido en dos categorías (como librerías objeto): la librería (built-in library) contiene versiones de las funciones de MATLAB en lenguaje C del tipo numérico, lógico y utilidades. Por otra parte la librería de toolboxes (toolbox library) contiene versiones compiladas de la mayoría de ficheros M de MATLAB para cálculo numérico, análisis de datos y funciones de acceso a ficheros y matrices.

En equipos UNIX estas librerías pueden ser igualmente obtenidas como librerías de tipo estático (static libraries) o bien como librerías compartidas (shared libraries). Respecto al mundo PC, estas librerías pueden obtenerse como DLL's en el entorno Microsoft Windows o como librerías compartidas en equipos Apple MacIntosh.

2.9.6 Utilización de MATLAB y de su compilador

Para construir una aplicación del tipo: “stand alone”, que incorpore código originalmente desarrollado como ficheros M de MATLAB, deberán seguirse los pasos siguientes:

1. Utilizar el compilador de MATLAB para convertir ficheros M en C mediante la utilización de la instrucción mcc -e (la cual es externa a MATLAB).
2. Compilar el código C fuente en código objeto utilizando un compilador ANSI C.
3. Enlazar el código resultante con la MATLAB C Math Library y con cualquier tipo de ficheros y programas específicos que hayan sido previamente definidos por el usuario.

2.9.7 Velocidad y Precisión

Los algoritmos utilizados en la MATLAB C Math Library han sido desarrollados por un grupo de renombrados expertos en programación algorítmica de funciones de tipo matemático (algebra lineal y cálculo numérico). Las funciones de álgebra lineal han sido obtenidas de las librerías mundialmente reconocidas LINPACK y EISPACK. La MATLAB C Math Library contiene más de 300 funciones numéricas, lógicas y de utilidad. Todas estas funciones le permitirán operar en datos de tipo escalar, vectorial o matricial con la misma facilidad sintáctica.

2.9.8 Neural Network Toolbox

Este toolbox proporciona funciones para el diseño, inicialización, simulación y entrenamiento de los modelos neuronales de uso más extendido en la actualidad:

Perceptrón, redes lineales, redes de retropropagación, redes de base radial, aprendizaje asociativo y competitivo, aplicaciones auto-organizativas, aprendizaje de cuantización vectorial, redes de Elman y redes de Hopfield.

Mediante la inclusión de un amplio abanico de funciones y procedimientos escritos para MATLAB, el usuario puede mediante el Neural Network Toolbox efectuar el diseño de arquitecturas complejas, combinando los modelos que ya están proporcionados por defecto en el toolbox. Asimismo, el usuario puede definir sus

propias funciones de transferencia e inicialización, reglas de aprendizaje, funciones de entrenamiento y estimación de error para usarlas posteriormente con las funciones básicas.

El toolbox, aporta las facilidades y prestaciones gráficas de MATLAB para el estudio del comportamiento de las redes: visualización gráfica de la matriz de pesos y vector de desplazamiento mediante diagramas de Hinton, representación de errores a lo largo del entrenamiento, mapas de superficie de error en función de pesos y vector de desplazamiento, etc. Estos gráficos resultan muy útiles en el estudio de la convergencia y estabilidad de los algoritmos de aprendizaje. Este toolbox incluye un manual de introducción al campo de las redes neuronales junto con una colección de demostraciones y aplicaciones muy didácticas, útiles para el estudio y la profundización en las cuestiones fundamentales de los paradigmas de redes neuronales básicos. Asimismo, se proporcionan las referencias bibliográficas más significativas referidas a los distintos modelos que aparecen en la aplicación.

A pesar de que el estudio de las redes neuronales se inició ya hace algunas décadas, las primeras aplicaciones sólidas dentro de este campo no han tenido lugar hasta hace unos doce años y aun ahora constituyen un área de investigación en rápido desarrollo. Este toolbox tiene por tanto una orientación diferente a aquellos destinados a campos como el de sistemas de control u optimización donde la terminología, fundamentos matemáticos y procedimientos de diseño están ya firmemente establecidos y se han aplicado durante años.

Este toolbox pretende que sea utilizado para la valoración y diseño de diseños neuronales en la industria y sobre todo en educación e investigación.

Esta herramienta tiene el soporte de MATLAB 4.2c y SIMULINK. La librería de SIMULINK contiene modelos de capas de redes neuronales de cada tipo de neurona implementada en el toolbox de redes neuronales. Es posible por tanto diseñar sistemas SIMULINK para simular redes neuronales creadas usando esta herramienta. Simplemente, las capas se conectan de acuerdo con la arquitectura de la red y se proporcionan como entrada a la caja de diálogo de cada capa la matriz de pesos apropiada y el vector de desplazamiento. Usando el generador de código C de SIMULINK es posible generar automáticamente el código correspondiente a un diseño neuronal.

Dentro de las aplicaciones básicas de este toolbox, cabe destacar aquellas que están orientadas a aquellas que se enmarcan dentro del campo de la industria aeroespacial y automoción (simulación, sistemas de control, auto pilotaje), banca, defensa (reconocimiento de patrones, procesamiento de señales, identificación de imágenes, extracción de características, compresión de datos), electrónica (control de procesos, análisis de errores, modelado no lineal, síntesis de voz, visión por ordenador), economía (análisis financiero, análisis predictivo), industria (control de procesos, identificación en tiempo real, sistemas de inspección), medicina, robótica (control de trayectorias, sistemas de visión), reconocimiento y síntesis del habla, telecomunicaciones (control de datos e imágenes, servicios de información automatizada, traducción del lenguaje hablado en tiempo real, diagnosis, sistemas de

enrutamiento), etc. El toolbox contiene muchos ejemplos de algunas de estas aplicaciones.

2.9.9 Non Linear Control Design Toolbox

Se trata del primer producto comercialmente disponible en la actualidad para el diseño de controladores automáticos en entornos de sistemas no lineales. Este nuevo toolbox está pensado para ser utilizado exhaustivamente por ingenieros que diseñan controladores para industrias avanzadas, destacando el sector del automóvil, ingeniería aeroespacial, control de procesos y empresas petroquímicas. Según indica Jim Tung, Vicepresidente del área de desarrollo de The Math Works Group, Inc. "El proceso de aproximación tradicional en el diseño de controladores en sistemas no lineales ha sido hasta la fecha linealizarlos de algún modo para aplicar posteriormente un método de diseño lineal que requiere de importantes ajustes manuales. El toolbox NCD permite por primera vez a los ingenieros de control diseñar directamente sus controladores en un ambiente no lineal, obviando la aproximación lineal y otros procedimientos auxiliares que antes se necesitaban de modo imperativo.

Los resultados ahora son de elevada calidad, controladores más robustos y un ciclo de diseño mucho más rápido.

El toolbox NCD extiende, además, las prestaciones que incorpora SIMULINK, el entorno de desarrollo de diagramas de bloques para la modelación y análisis de sistemas dinámicos de The MathWorks, Inc. El usuario puede incluir uno o más bloques NCD en el sistema y describir posteriormente de modo totalmente gráfico las restricciones, tolerancias y límites de permisividad de cada uno de estos bloques. Los métodos avanzados de optimización y la simulación del proceso son posteriormente analizados y ajustados mediante la inclusión de unas ciertas variables de contorno para poder obtener los tiempos de respuesta deseados. Este toolbox puede ser utilizado para ajustar una amplia variedad de controladores que se utilicen en un sistema, destacando los controladores PID, LQR, LQG y estructuras H infinito. El diseñador de sistemas puede utilizar el método de Montecarlo para el diseño y análisis de controladores robustos, 18 siempre que se detecten determinadas variaciones en los componentes del sistema.

El toolbox NCD es un componente avanzado del entorno integrado de desarrollo que ofrecen a los especialistas los programas MATLAB y SIMULINK. Por ello, los diseñadores podrán beneficiarse de muchos de los toolboxes desarrollados para este entorno en materia de diseño de sistemas lineales.

Por ejemplo, podrán utilizarse toolboxes para el análisis de sistemas lineales para el diseño inicial; posteriormente, podrán utilizarse modelos no lineales más sofisticados utilizando SIMULINK.

Además, puede invocarse NCD para un mejor ajuste paramétrico y para la optimización de los controladores. Este toolbox se encuentra actualmente disponible

para una amplia variedad de plataformas informáticas, destacando ordenadores personales tipo PC o Apple Macintosh, numerosas estaciones UNIX y ordenadores Digital VAX VMS

2.9.10 Fuzzy Logic Toolbox

Fuzzy Logic Toolbox es una herramienta que proporciona a Matlab, funciones, aplicaciones y un bloque de SIMULINK para analizar, diseñar, y simular sistemas basados en lógica difusa.

Este módulo de Matlab permite modelar el comportamiento de sistemas complejos, usando reglas lógicas muy simples, para luego poder implementar esas reglas en un sistema de inferencia difusa. Esto se puede usar como un motor de inferencia difusa autónomo.

Alternativamente, puede utilizar los bloques de inferencia difusos en Simulink y simular los sistemas difusos dentro de un modelo integral de un sistema dinámico.

2.10 Arduino-Labview

Introducción

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador. Y al ser open-hardware, tanto su diseño como su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia. (San miguel, 2012)

La interfaz de Labview para Arduino (LIFA) Toolkit es una herramienta gratuita que se puede descargar desde el servidor de NI (National Instruments) y que permite a los usuarios de Arduino adquirir datos del microcontrolador Arduino y procesarlos en el entorno de programación gráfica de Labview. (Ruiz, 2012)

El microcontrolador Arduino es una plataforma de bajo costo de electrónica de prototipos. Con la interfaz de Labview para Arduino LIFA se puede aprovechar la

potencia del entorno de programación gráfica de Labview para interactuar con Arduino en una nueva dimensión.

Interface Gráfica de Usuario (Graphical User Interface GUI)

Visualizar los datos Mostrar datos de los sensores en el monitor del ordenador mediante los paneles frontales de Labview. Personalización de la interfaz de usuario permite dar al proyecto un toque profesional con los controles del panel frontal de Labview y los indicadores (figura 2-18).

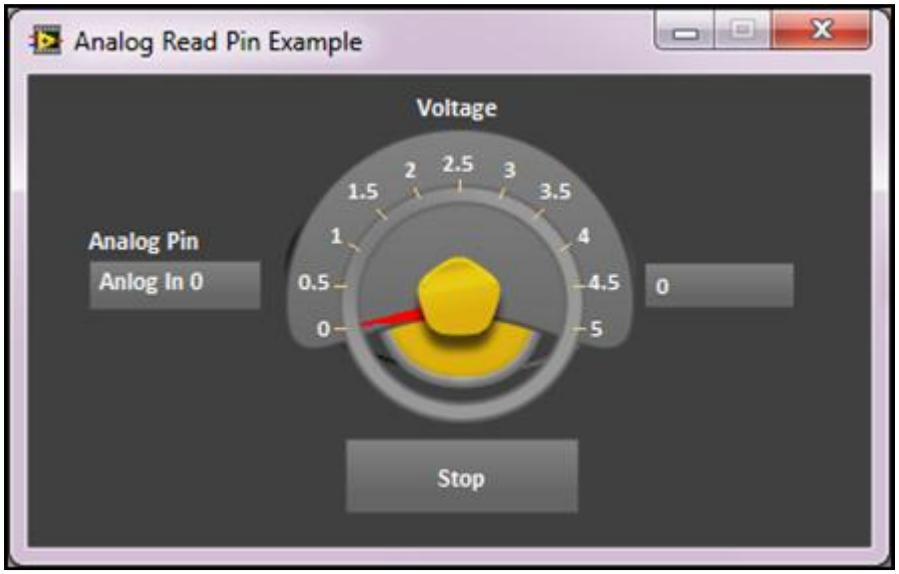


Figura 2-18 Interfaz Gráfica de Usuario.

2.10.1 Programación Grafica

Arrastrar y soltar En lugar de tratar de recordar un nombre de función, se encuentra en la paleta y colóquelo en su diagrama de bloques. Documentación simple Pase el ratón sobre cualquier VI o función con el ratón y ver al instante la documentación con ayuda contextual (figura 2-19).

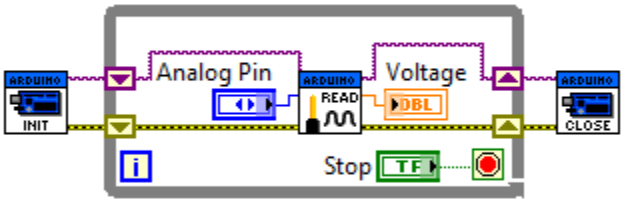


Figura 2-19 Ejemplo de un programa gráfico en el entorno Labview.

2.10.2 Desarrollo interactivo

Animar la ejecución Diagrama de bloques Consulte los valores de datos que se transmiten de una función a otra con resaltado de ejecución. Sondas, puntos de interrupción, y la intensificación Datos de la sonda, la ejecución de una pausa, y el paso a una subrutina sin necesidad de programación compleja (figura 2-20).

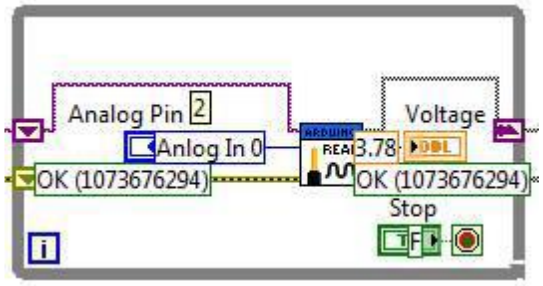


Figura 2-20 Ejemplo de visualización de datos de entrada.

2.10.3 Utilización de librerías

Se pueden aprovechar cientos de bibliotecas integradas de procesamiento de señales, matemáticas y análisis. Bibliotecas Conectividad Interfaz con los servicios web, bases de datos, archivos ejecutables y más con funcionalidad integrada en el núcleo de Labview (figura 2-21). (Ruiz, 2012)

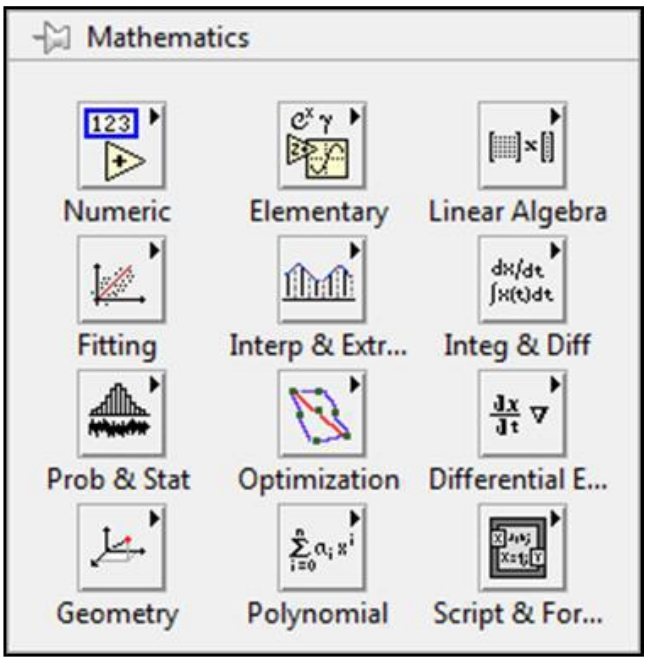


Figura 2-20 Ejemplo de una librería de Labview.

2.10.4 Instalación del Software y el Hardware

A continuación describimos los pasos que se recomiendan para la puesta en marcha de la herramienta LIFA (Labview para Arduino): La configuración de la Interfaz de Labview para Arduino es un proceso de seis pasos que usted sólo tendrá que completar una sola vez. Por favor, siga las siguientes instrucciones para comenzar a crear aplicaciones con la interfaz de Labview para Arduino.

1. Instalar Labview.2. Instale los controladores VISA NI
3. Instalar JKI VI Package Manager (VIPM) Community Edition (gratuito).

4. Instalación de la Interfaz de Labview para Arduino
5. Conectar la placa Arduino a su PC
6. Carga de la interfaz de Labview para firmware Arduino en su Arduino
7. El firmware se puede encontrar en <Labview> \ vi.lib Interface \ Labview para Arduino \ Firmware \ LVIFA_Base. Utilizar el IDE de Arduino para implementar este firmware de la placa Arduino.).

Instalación del Firmware de comunicación entre Labview Interface y Arduino Uno.

Para poder comunicar Labview con Arduino, previamente, debemos instalar en la tarjeta el firmware correspondiente.

Partimos del supuesto de que ya tenemos instalado en nuestro PC el entorno IDE Arduino.

El fichero que debemos cargar en el IDE de Arduino para luego descargar en la tarjeta se encuentra en la carpeta en donde tengamos instalado Labview, ejecutamos el IDE Arduino y cargamos el fichero.

Pasos a seguir:

Abrir el IDE Arduino, pulsando sobre arduino.exe con la opción Fichero->Abrir.
Buscamos el fichero LVIFA_Base.pde (figura 2-22).

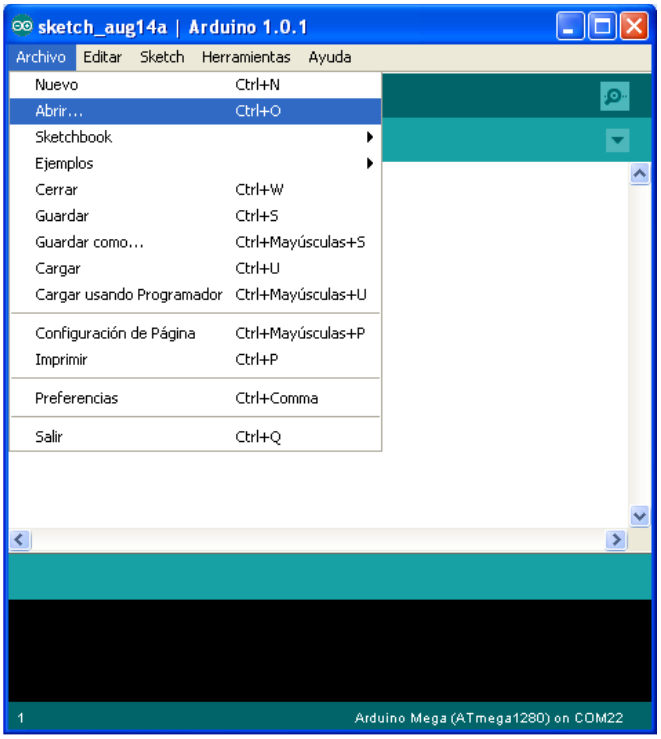


Figura 2-21 Menú principal de la Interfaz.

Una vez cargado el fichero en el IDE Arduino seleccionamos la tarjeta con la que se desea trabajar (figura 2-23).

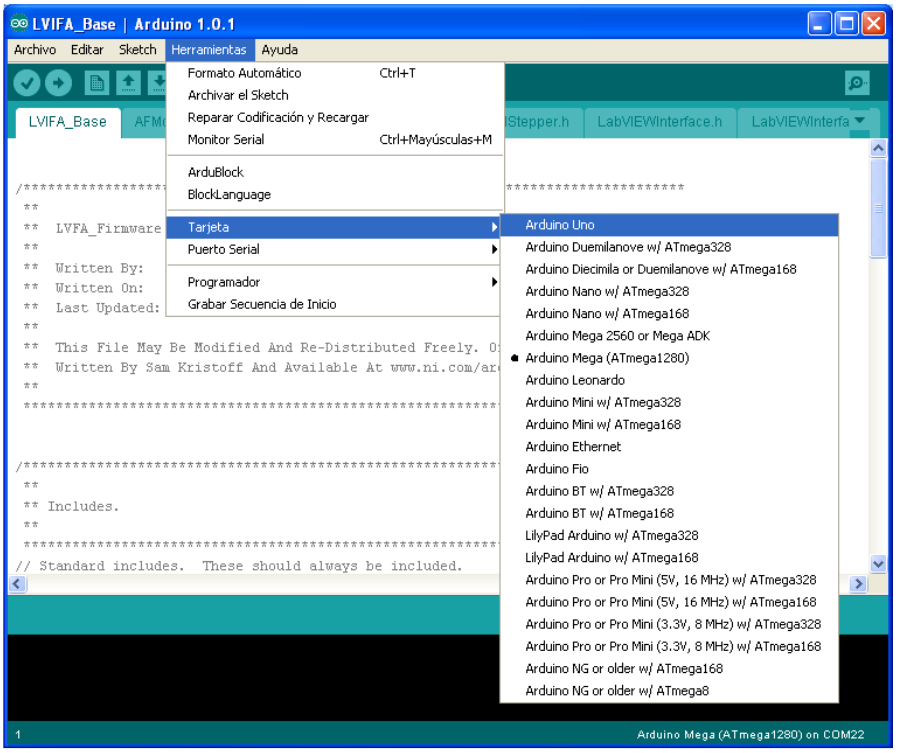


Figura 2-22 Seleccionar la tarjeta adecuada en el menú herramientas.

Seguidamente seleccionamos el puerto con el que realizaremos la descarga del firmware sobre la tarjeta Arduino (figura 2-24).

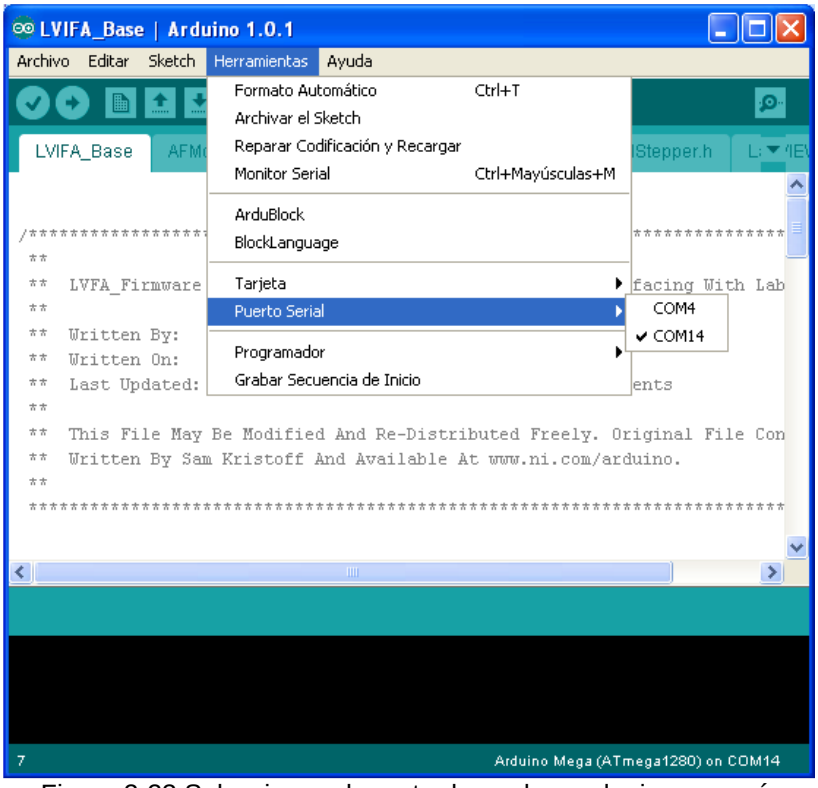


Figura 2-23 Seleccionar el puerto deseado en el mismo menú.

Una vez realizadas estas operaciones basta con que pulsemos el botón de carga de sketch del IDE para que el fichero se transfiera a la tarjeta y, una vez transferido, ya hemos dejado Arduino listo para comunicarse con Labview (figura 2-23).

Figura 2-24 Ventana lista para cargar el sketch del IDE.

2.11 Definiciones Básicas de Sistemas de Control

Las definiciones básicas de los sistemas de control son el punto de partida para comprender el estudio. Estas definiciones surgieron usando como base las ideas de la autora Katsuhiko Ogata en su libro Ingeniería de Control Moderna.

2.11.1 Variable controlada y variable manipulada.

La variable controlada es la cantidad o condición que se mide y controla. La variable manipulada es la cantidad o condición que el controlador modifica para afectar el valor de la variable controlada. El objetivo del control es medir el valor de la variable controlada del sistema para aplicar correcciones a través de la variable manipulada para obtener un valor deseado.

2.11.2 Planta.

La planta normalmente es un conjunto de partes que trabajan juntas con el objetivo de realizar una operación en particular. Se le llama planta a cualquier sistema físico que se desea controlar.

2.11.3 Proceso.

El proceso es cualquier operación que va a ser controlada.

2.11.4 Sistema.

Un sistema es un conjunto de componentes que se interrelacionan y trabajan juntos para realizar un objetivo determinado.

2.11.5 Perturbación.

Una perturbación es una señal que normalmente afecta a la variable controlada del sistema. Las perturbaciones pueden ser internas cuando surgen dentro del sistema, o externas porque se produce fuera del sistema y actúan como otra entrada.

2.11.6 Control realimentado.

El control realimentado es un sistema que mantiene una comparación entre la entrada de referencia y la salida deseada, el resultado de la comparación es utilizado para controlar.

2.11.7 Sistema de control en lazo cerrado.

Los sistemas de control en lazo cerrado alimentan al controlador la señal de error de actuación que es la diferencia entre la señal de entrada y la señal de realimentación, a fin de reducir el error y llevar la salida del sistema a un valor deseado. El término control de lazo cerrado siempre implica el uso de una acción de control realimentado para reducir el error del sistema; es por eso que el termino control de lazo realimentado y de lazo cerrado se usan indistintamente.

2.11.8 Sistemas de control en lazo abierto.

En estos sistemas la salida no afecta la acción de control. Es decir que en este tipo de control no se mide la salida ni se realimenta para compararla con la entrada. Por lo tanto, a cada entrada de referencia le corresponde una condición operativa fija; lo que obliga a que la precisión del sistema sea dependiente de la calibración del mismo.

2.11.9 Función de Transferencia.

El aspecto más importante dentro de los sistemas de control es la estabilidad del sistema.

Un sistema es estable si en ausencia de alguna perturbación la salida permanece en el mismo estado. En un sistema de control se busca que a pesar de las

perturbaciones o entradas el sistema vuelva a un estado de equilibrio. Para este motivo es necesario conocer las características del sistema a través de su función de transferencia.

Los sistemas de control actuales generalmente son no lineales, sin embargo es posible aproximarlos a través de medios matemáticos; es por eso que analizar la respuesta transitoria de la planta es el primer paso a tomar en cuenta para poder implementar las acciones de control. El análisis de la respuesta transitoria nos da como resultado la Función de Transferencia que nos representará la planta que se va a controlar.

2.11.10 Acciones de Control.

La forma en la cual el controlador automático produce la señal de control se llama acción de control. Los controladores automáticos comparan el valor real de la salida de la planta de referencia, lo cual determina la desviación con la que el controlador debe producir una señal de control que reduzca la desviación.

3 DESARROLLO

Una vez realizada la investigación documental, recolectando y clasificando información de libros, revistas, tesis, bases de datos, y la revisión de trabajos relativos a sistemas de control basados en lógica difusa y control lineal, se sintetiza y analiza la información.

La figura 3-1 muestra una aplicación de un motor de corriente directa que tiene acoplada una transmisión compuesta por un mecanismo de usillo-corona, que sirve para transformar un desplazamiento angular a lineal.

Para que este sistema tenga una aplicación práctica es necesario identificar y controlar sus variables de entrada y salida.

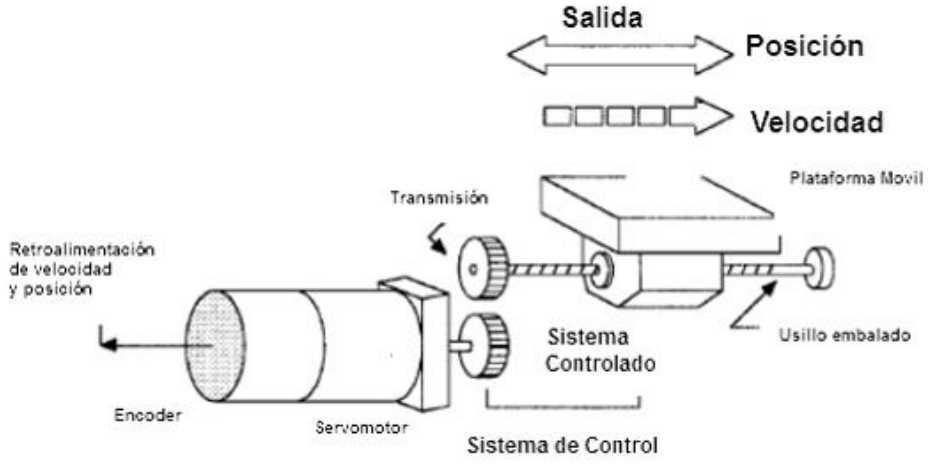


Figura 3-1 Transmisión acoplada con servo motor.

Se puede observar también que la corona se encuentra unida a una mesa móvil, la cual representa a la carga de tal forma que cuando el voltaje del motor sea positivo, la carga se desplaza a la derecha, y cuando el voltaje es negativo, ésta se desplaza hacia la izquierda, debido a que el motor empleado es de corriente continua.

En la figura 3-1, se muestra un diagrama de bloques en el que se observan los elementos que integran el sistema, y su respectivo lazo de control, en él se puede observar un motor de corriente directa que recibe una señal amplificada por un módulo de potencia con un arreglo en puente H para el control bidireccional, así mismo este tiene una señal de entrada con modulación por ancho de pulso (PWM) que es enviada desde el controlador difuso, el cual recibe las señales de retroalimentación del sensor de posición y velocidad, compuesto principalmente por un encoder en cuadratura, la

señal retroalimentada es procesada para realizar la lógica de control necesaria de funcionamiento la cual se puede observar en la figura 3-2.

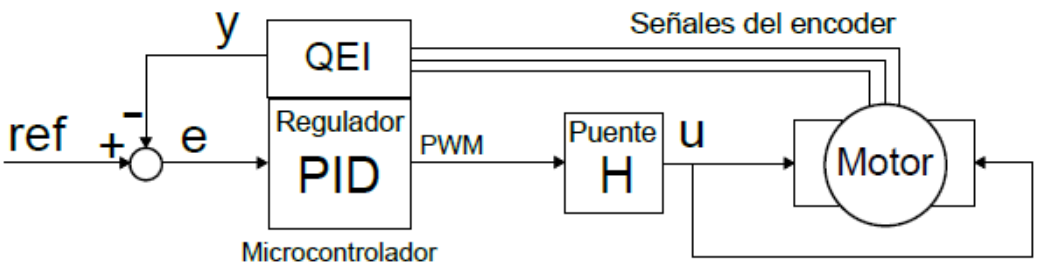


Figura 3-2 Sistema de control del motor de CD.

3.1 Motor de corriente continua

El uso de motores de CC (corriente continua) está muy extendido, hoy en día se pueden encontrar en cualquier dispositivo electromecánico. Controlar la posición y la velocidad de estos motores es muy sencillo convirtiéndose en una buena opción a la hora de crear aplicaciones de automatización y control.
Los motores de corriente continua simplemente necesitan ser conectados a una fuente de corriente continua o una batería compatible para que se pongan en marcha. Es posible modificar la velocidad bajando o subiendo la alimentación, girando más lento o más rápido según el voltaje. Al cambiar la polaridad el motor gira en sentido contrario y si lo desconectas, por inercia sigue girando hasta que se para.

3.2 Modificar la velocidad de giro, PWM

Para regular la velocidad de un motor de continua se usa la modulación por ancho de pulsos (Pulse Width Modulation). El PWM consiste en modificar la cantidad de energía que recibe un motor. Si se conecta un motor a una fuente directamente el motor comienza a acelerar hasta que alcanza cierta velocidad, si en ese momento se desconecta, el motor decelera hasta que se para. As__ pues al conectar y desconectar repetidamente el motor de la fuente en intervalos pequeños el motor no llega a pararse y gira más despacio. De este modo dependiendo del tiempo que se mantenga conectado y desconectado se pueden obtener distintas velocidades, esto se conoce como ciclo de trabajo D.

$$D = \frac{\tau}{T}$$

Donde τ representa el tiempo que está conectado y □□el periodo. Por ejemplo, con un periodo de un segundo, se deja conectado 0.7 segundos, se dice que el ciclo de trabajo es del 70 %. Si el ciclo es 0% se considera apagado y si es 100% se considera siempre conectado (figura 3-3).

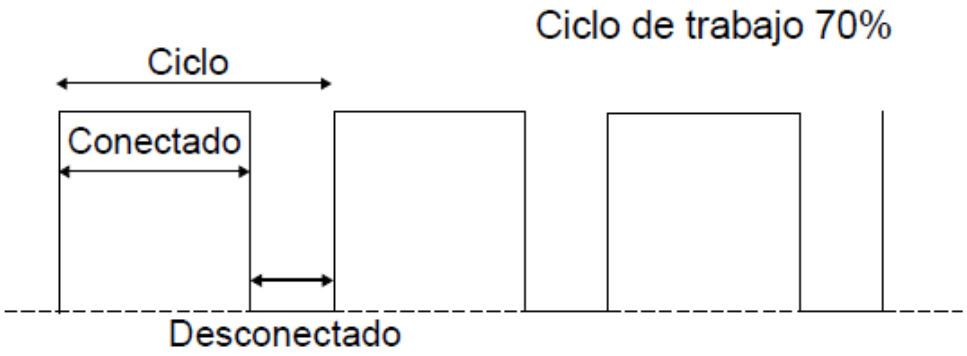


Figura 3-3 Posibles estados que puede adoptar D.

3.3 Posición y Velocidad, Encoders

Ya que el sistema que se pretende controlar es un motor de continua, se puede realizar tanto un control de posición como un control de velocidad. El control de posición mueve el motor una distancia determinada para después detenerse mientras que el control de velocidad establece una velocidad a la cuál debe moverse. Con esta técnica generando una señal con forma de pulso periódico, se puede regular la velocidad y la posición de un motor de corriente directa.

Para medir la velocidad, posición o aceleración del motor se utilizó un encoder digital, el cual consta de un disco codificado con ranuras y unido al eje de tal manera que el disco gira con el eje.

Un sensor detecta el paso de las ranuras y de esta forma sabiendo cuantas ranuras tiene el disco se puede deducir cuantas ranuras son una vuelta, por tanto conociendo el número de vueltas por unidad de tiempo se puede deducir la velocidad o simplemente cuanto se ha movido (figura 3-4).

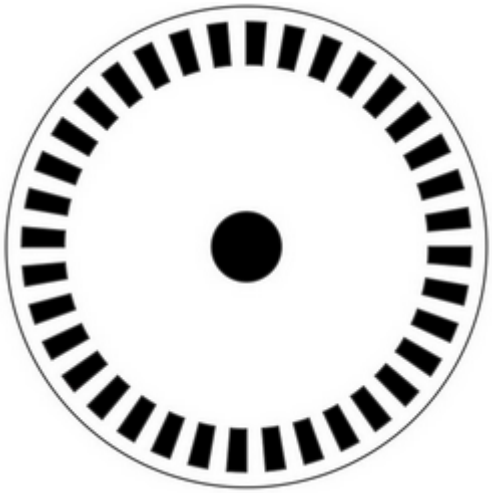


Figura 3-4 Disco de encoder incremental

Normalmente el eje del motor no está conectado directamente al sistema, por ejemplo una rueda, suele utilizar una reductora. Una reductora es un mecanismo compuesto por un conjunto de engranajes que modifica la velocidad del motor, además el eje del motor no debe soportar el peso del sistema y es la reductora junto con un sistema de transmisión el que soporta toda la carga. De este modo conociendo la relación de la reductora y contando las ranuras que se detectan por unidad de tiempo se puede deducir la posición o la velocidad.

3.4 Control de velocidad

El control de velocidad implica obtener el número de pulsos detectados por el paso de las ranuras en cada instante de tiempo. Es decir, pulsos por unidad de tiempo, el valor obtenido será la salida del sistema, en concreto la velocidad actual del motor, que se restará a la referencia para obtener la siguiente acción de control. Así que es importante conocer el tiempo entre acciones de control ya que este periodo servirá para conocer la velocidad del motor entre instantes de tiempo.

3.5 Control de posición

El control de posición procesa los pulsos detectados desde que el sistema se puso en funcionamiento, y los resta con la referencia a medida que avance el tiempo y se ejecuten acciones de control, el error se irá reduciendo pues la cuenta total de pulsos se irá acercando al valor de posición establecido por la referencia conforme gire el motor, y cuando la resta sea cero, el motor se detendrá indicando que ha llegado a la posición deseada.

En la Investigación Documental se realizó la recolección y clasificación de información en libros, revistas, tesis y bases de datos digitales. Y la revisión de trabajos y propuesta de los diferentes sistemas de comunicación existentes.

3.6 Sentido de giro, puente H

El sentido de giro de un motor de corriente continua está relacionado con la polaridad de la corriente que se aplica en los bornes. Un puente H consiste de cuatro transistores y un motor en medio, este diseño se asimila a la letra H y de ahí su nombre.

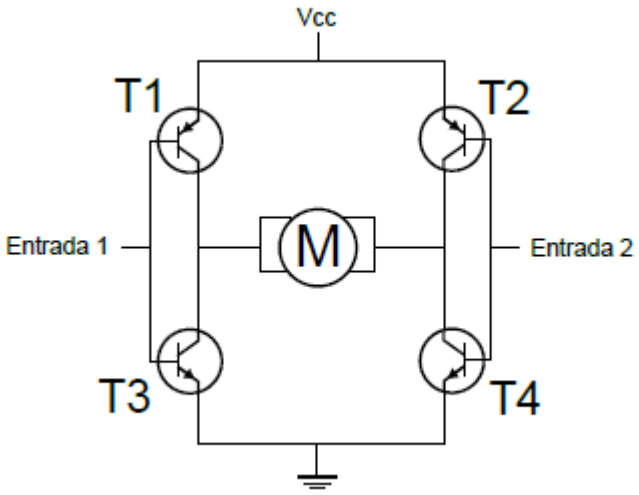


Figura 3-5 Diagrama electrónico de un puente H con transistores.

Al aplicar un *cero* lógico en la entrada 1 y un uno lógico en la entrada 2, la corriente pasará desde el transistor T1 hacia el T4, lo cual excitará a la bobina del motor haciéndolo girar en un sentido, si cambiamos las entradas, un *uno* lógico en la entrada 1 y un *cero* lógico en la entrada 2, la corriente circulará desde el transistor T2 hacia el T3 excitando nuevamente la bobina del motor pero con polaridad inversa y por lo tanto el motor girará en sentido contrario.

Si se aplica *cero* lógico en ambas entradas el motor no estará polarizado quedando *liberado*, esto quiere decir que se puede hacer girar con la mano en ambos sentidos. Del mismo modo al poner *uno* lógico en las dos entradas se quedará *fijo* y no se podrá girar manualmente en ningún sentido (figura 3-5).

3.7 Diseño del controlador difuso

El controlador lógico difuso se basa en el modelo lingüístico de la estrategia del operador humano, es decir, tiene la capacidad para operar con conceptos propios del

razonamiento cualitativo, fundado sobre un soporte matemático que permite extraer conclusiones a partir de un conjunto de observaciones y reglas cualitativas. La esencia de tal modelo es un programa basado en reglas, por lo que clasifica entre los llamados sistemas expertos (García, 2003).

Un controlador difuso se compone de cuatro elementos: el fuzzificador, la base de conocimiento, la toma de decisiones (mecanismo de inferencia) y defuzzificador.

Fuzzificado. La primera tarea del controlador lógico difuso (FLC) es traducir el valor medido en términos de valores lingüísticos. El proceso consiste en la medición de las variables de entrada al controlador realizando un mapeo a escala que transforma el rango de valores de las variables de entrada en los correspondientes valores lingüísticos en el universo discurso (Wang L. , 1997)

En la *fuzzificación* de las entradas se hace la lectura de las señales de nivel y flujo, las cuales se codifican de tal forma que queden en términos de los conjuntos difusos, como se muestra en la figura 6. Así, en esta etapa se determina su correspondiente grado de pertenencia.

La creación del algoritmo se basa exclusivamente en descripciones lingüísticas de los operadores acerca de su actuación en la dirección del proceso (Vadiie & Jamshidi, 1993)

Tomando como punto de partida el planteamiento del problema, se diseñó un controlador basado en lógica difusa, para ello se ha tomado el modelo propuesto por el Dr. Mamdani, el cual nos brinda las siguientes ventajas:

Es computacionalmente eficiente.
Trabaja bien con técnicas lineales (por ejemplo como lo disponible para controladores PID).
Trabaja bien con técnicas de optimización y control adaptable.
Tiene garantizada una superficie de control continua.

Está bien adaptado al análisis matemático.

3.8 Base de conocimiento.

La base de conocimiento consiste en una base de datos y otra de reglas. La base de datos proporciona las definiciones necesarias para las reglas de control y la manipulación de los datos difusos. El número máximo de reglas viene dado por el producto de números de particiones de todas las variables lingüísticas de entrada al FCL. Para hablar de una base de reglas, es preciso elegir cuáles variables se tomaran como entrada y cuáles como salida del FCL.

En la figura 3-6, 3-7, 3-8, se puede observar cómo se define la variable entrada DISTANCIA, VELOCIDAD Y VOLTAJE DE SALIDA respectivamente.

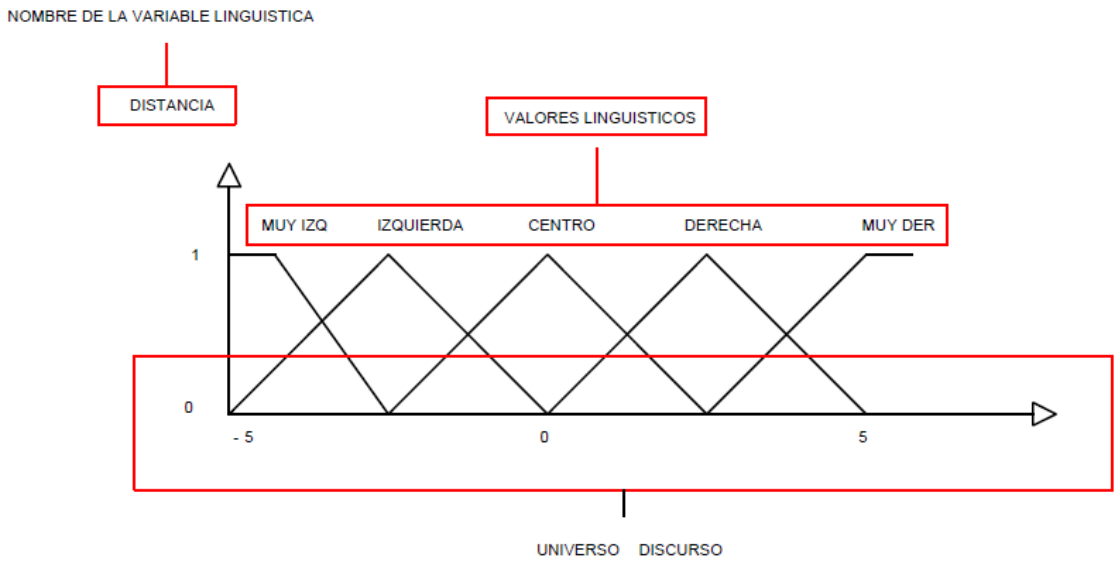


Figura 3-6 Definición de variable DISTANCIA.

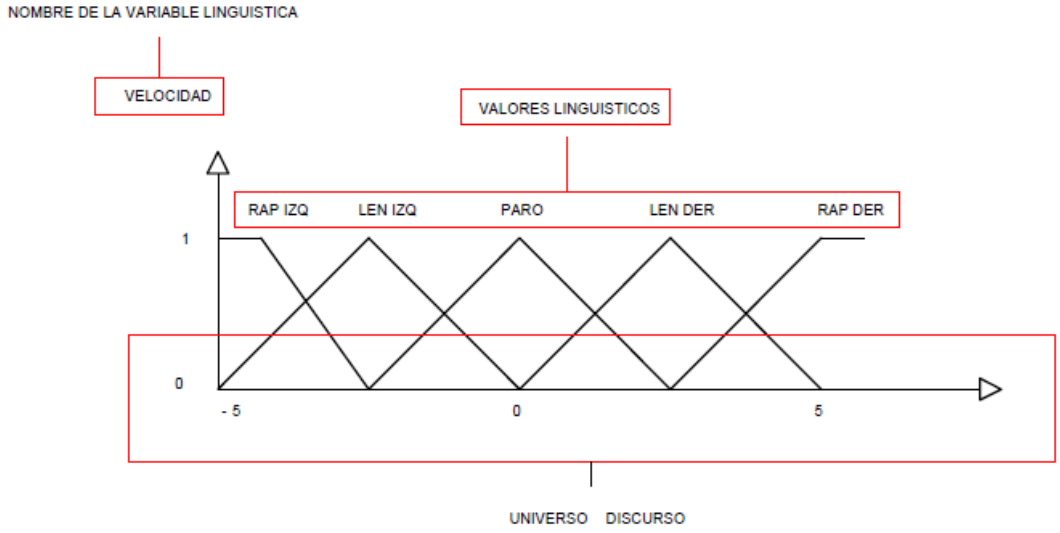


Figura 3-7 Definición de variable VELOCIDAD.

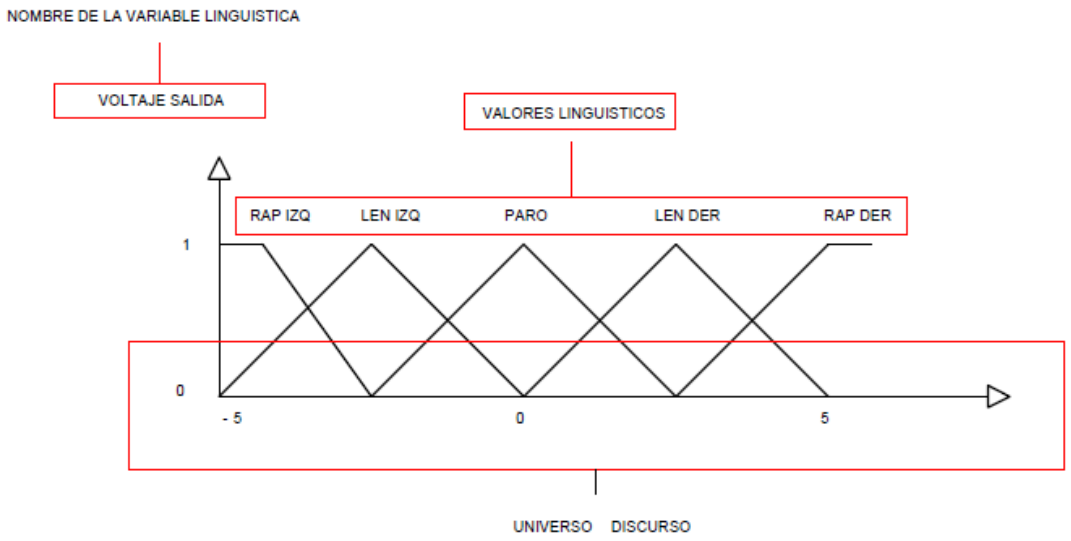


Figura 3-8 Definición de variable VOLTAJE DE SALIDA.

Una vez identificadas las variables lingüísticas, se integran al controlador difuso, para el diseño de la planta (figura 3-9).

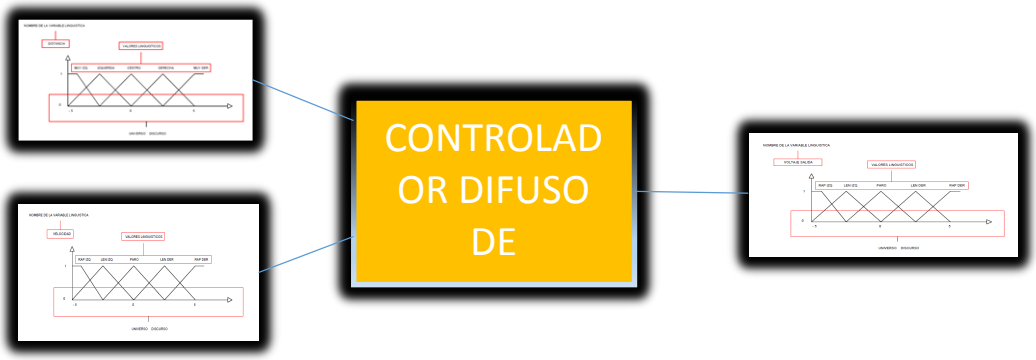


Figura 3-9 Planta con controlador difuso.

3.9 Toma de decisiones

La toma de decisiones es la médula del controlador difuso (mecanismo de inferencia). Tiene la capacidad de simular la toma de decisiones humanas basada en conceptos y acciones de control y empleando implicaciones y reglas de inferencia de la lógica difusa. Estas tareas se resuelven utilizando las operaciones lógicas de disyunción,

conjunción e implicación, de tal forma que la evaluación de una regla vendrá dada por un *antecedente* y un *consecuente.*

Los sistemas difusos tipo Mamdani tienen la siguiente estructura:

Si $X_1 \in a_1 \ \Lambda \ x_2 \in a_2 = \mu_1 \in b_1$

Donde:

$x_1 \in a_1 \ \Lambda \ x_2 \in a_2$ *es el antecedente*

$\mu_1 \in b_1$ *es el consecuente*

El *antecedente* y *consecuente* son ambos difusos; se construyen con variables difusas y funciones de pertenencia.

Donde

$x_1 \ x_2 \ \mu_1$ *son variables difusas.*

$a_1 \ a_2 \ a_3$ *son funciones de pertenencia.*

Es importante conocer las etapas de activación y desactivación de las señales para la toma de decisiones. Supondremos en lo que sigue que la base de reglas es de tipo clásico, y las reglas se realizaran con base en la figura 3-10.

Si< estado del proceso>entonces <acción de control>

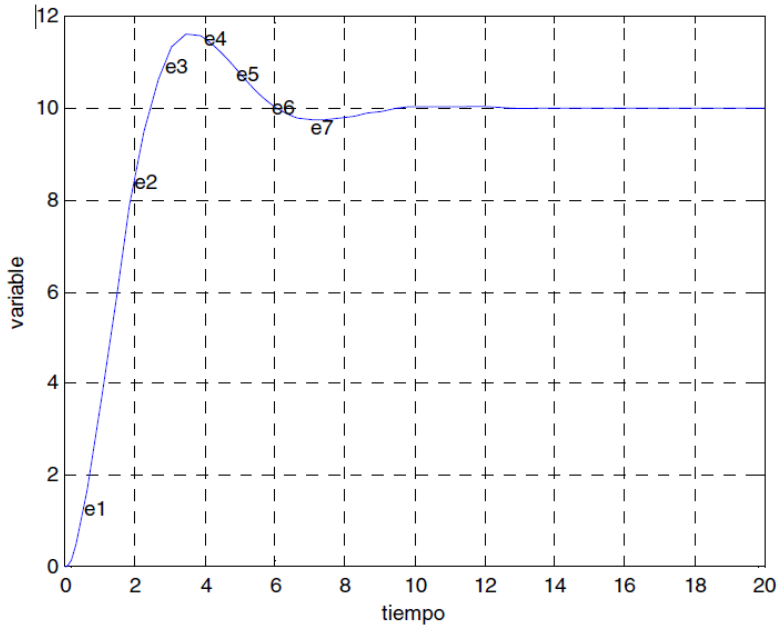


Figura 3-10 Activación de reglas.

En los puntos e1 y e2 el error es positivo y la derivada del error es también positiva, para lo que se generará una salida positiva.

En el punto e3 el error es negativo y la derivada del error es positiva, para lo que se generará una salida cero.

En los puntos e4 y e5 el error es negativo y la derivada del error es también negativa, para lo que se generará una salida negativa.

En el punto e6 el error es cero y la derivada del error es negativa, para lo que se generará una salida cero.

En el punto e7 el error es positivo y la derivada del error es cero, para lo que se generará una salida positiva.

De esta manera se construye la tabla 3-1 FAM (*fuzzy asociative memory*) para tres particiones, como se puede observar en la tabla 1, que describe la relación entre las variables de entrada y las variables de salida.

Tabla 3-1 FAM (fuzzy asociative memory) de tres particiones

e\eº	Negativo	Cero	Positivo
Negativo	Negativo	Negativo	Cero
Cero	Negativo	Cero	Positivo
Positivo	Cero	Positivo	Positivo

Entonces, un estado dado dará lugar a la “activación” de varias reglas y se activarán solamente aquellas en las que todos los conjuntos difusos del antecedente resulten ser no nulos para el estado dado (Vadiie & Jamshidi, 1993).

Para el diseño de este controlador se consideró una distancia de -0.5 a 0.5 como límites y el punto de equilibrio en cero como se indica en la figura 3-11.

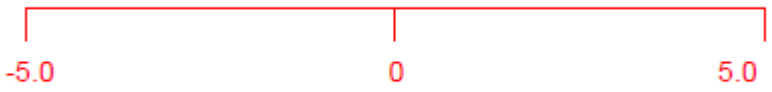


Figura 3-11 Distancia.

Estos límites representan el universo en discurso, y en él, se determinaron las variables lingüísticas, como se muestra en la Figura 3-6. (DISTANCIA) que corresponden a: muy a la izquierda (MUY IZQ), izquierda (IZQ), centro (CENTRO), derecha (DERECHA), muy a la derecha (DERECHA). Los nombres de las variables se asignaron de esta manera para que resulten familiares y de cuerdo a la experiencia

del operador humano es decir que rengos se consideran muy a la izquierda o muy a la derecha y siempre debe estar combinados.

Haciendo esto mismo a la entrada VELOCIDAD se tiene que: rápido a la izquierda (RAP IZQ), lento a la izquierda (LEN IZQ), motor en paro (PARO), lento a la derecha (LEN DER), rápido a la derecha (RAP DER), (figura 3-7).

La salida VOLTAJE, adopto las siguientes variables lingüísticas: positivo grande (PG), positivo pequeño (PP), sin voltaje (ZERO), negativo pequeño (NP), negativo grande (NG), (figura 3-8).

Una vez determinado los conjuntos difusos, se proponen las siguientes reglas representadas en forma matricial en una tabla denominada matriz de asociación difusa o FAM (tabla 3-2).

Tabla 3-2 Matriz de Asociación Difusa FAM.

		DISTANCIA				
		MUY IZQ	IZQ	CENTRO	DERECHA	MUY DER
VELOCIDAD	RAP IZQ	PG	PP	ZE	NP	NG
	LEN IZQ	PG	PP	ZE	NP	NG
	PARO	PG	PP	ZE	NP	NG
	LEN DER	PP	PP	ZE	NP	NG
	RAP DER	PP	ZE	ZE	ZE	NP

Las reglas pueden verificarse mediante un plano de fase, graficando al error y a la derivada del error, donde el segundo cuadrante corresponde al decrecimiento del tiempo de impulso (figura 2-12).

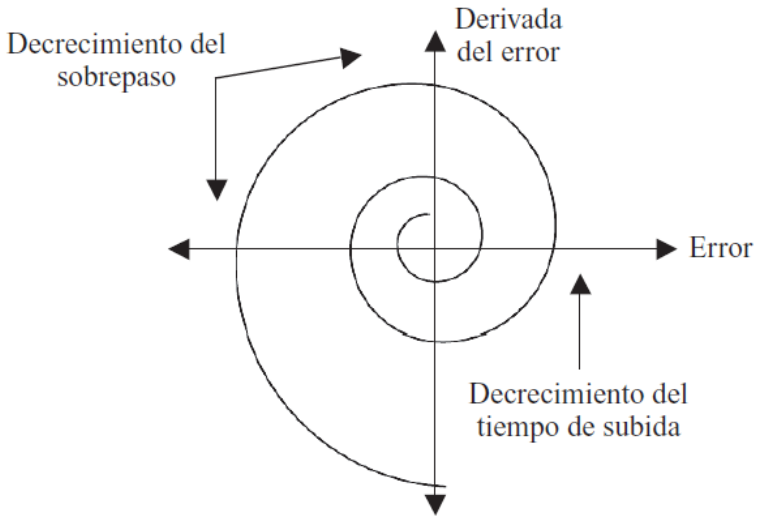


Figura 3-12 Plano de fase.

De lo anterior, se dedujo el siguiente plano de fase lingüístico para el controlador (figura 3-13).

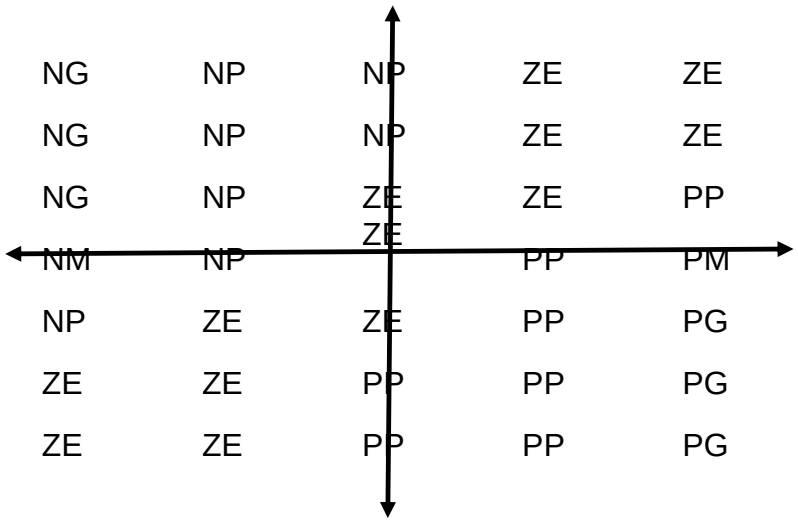


Figura 3-13 Plano de fase Lingüístico.

A continuación establecemos las reglas lingüísticas tal como se indica en la siguiente tabla.

Tabla 3-3 Reglas lingüísticas del controlador difuso.

1	si	DISTANCIA	esta	MUY A LA IZQUIERDA	Y	VELOCIDAD	esta	RAPIDO A LA IZQUIERDA	entonces	VOLTAJE	esta	POSITIVO GRANDE
2	si	DISTANCIA	esta	IZQUIERDA	Y	VELOCIDAD	esta	RAPIDO A LA IZQUIERDA	entonces	VOLTAJE	esta	POSITIVO GRANDE
3	si	DISTANCIA	esta	CENTRO	Y	VELOCIDAD	esta	RAPIDO A LA IZQUIERDA	entonces	VOLTAJE	esta	ZERO
4	si	DISTANCIA	esta	DERECHA	Y	VELOCIDAD	esta	RAPIDO A LA IZQUIERDA	entonces	VOLTAJE	esta	NEGATIVO PEQUEÑO
5	si	DISTANCIA	esta	MUY A LA DERECHA	Y	VELOCIDAD	esta	RAPIDO A LA IZQUIERDA	entonces	VOLTAJE	esta	NEGATIVO GRANDE
6	si	DISTANCIA	esta	MUY A LA IZQUIERDA	Y	VELOCIDAD	esta	LENTO A LA IQUIERDA	entonces	VOLTAJE	esta	POSITIVO GRANDE
7	si	DISTANCIA	esta	IZQUIERDA	Y	VELOCIDAD	esta	LENTO A LA IQUIERDA	entonces	VOLTAJE	esta	POSITIVO PEQUEÑO
8	si	DISTANCIA	esta	CENTRO	Y	VELOCIDAD	esta	LENTO A LA IQUIERDA	entonces	VOLTAJE	esta	ZERO
9	si	DISTANCIA	esta	DERECHA	Y	VELOCIDAD	esta	LENTO A LA IQUIERDA	entonces	VOLTAJE	esta	NEGATIVO PEQUEÑO
10	si	DISTANCIA	esta	MUY A LA DERECHA	Y	VELOCIDAD	esta	LENTO A LA IQUIERDA	entonces	VOLTAJE	esta	NEGATIVO GRANDE
11	si	DISTANCIA	esta	MUY A LA IZQUIERDA	Y	VELOCIDAD	esta	EN PARO	entonces	VOLTAJE	esta	POSITIVO GRANDE
12	si	DISTANCIA	esta	IZQUIERDA	Y	VELOCIDAD	esta	EN PARO	entonces	VOLTAJE	esta	POSITIVO PEQUEÑO

13	si	DISTANCIA	esta	CENTRO	Y	VELOCIDAD	esta	EN PARO	entonces	VOLTAJE	esta	ZERO
14	si	DISTANCIA	esta	DERECHA	Y	VELOCIDAD	esta	EN PARO	entonces	VOLTAJE	esta	NEGATIVO PEQUEÑO
15	si	DISTANCIA	esta	MUY A LA DERECHA	Y	VELOCIDAD	esta	EN PARO	entonces	VOLTAJE	esta	NEGATIVO GRANDE
16	si	DISTANCIA	esta	MUY A LA IZQUIERDA	Y	VELOCIDAD	esta	LENTO A LA DERECHA	entonces	VOLTAJE	esta	POSITIVO PEQUEÑO
17	si	DISTANCIA	esta	IZQUIERDA	Y	VELOCIDAD	esta	LENTO A LA DERECHA	entonces	VOLTAJE	esta	POSITIVO PEQUEÑO
18	si	DISTANCIA	esta	CENTRO	Y	VELOCIDAD	esta	LENTO A LA DERECHA	entonces	VOLTAJE	esta	ZERO
19	si	DISTANCIA	esta	DERECHA	Y	VELOCIDAD	esta	LENTO A LA DERECHA	entonces	VOLTAJE	esta	NEGATIVO PEQUEÑO
20	si	DISTANCIA	esta	MUY A LA DERECHA	Y	VELOCIDAD	esta	LENTO A LA DERECHA	entonces	VOLTAJE	esta	NEGATIVO GRANDE
21	si	DISTANCIA	esta	MUY A LA IZQUIERDA	Y	VELOCIDAD	esta	RAPIDO A LA DERCHA	entonces	VOLTAJE	esta	VOLTAJE PEQUEÑO
22	si	DISTANCIA	esta	IZQUIERDA	Y	VELOCIDAD	esta	RAPIDO A LA DERCHA	entonces	VOLTAJE	esta	ZERO
23	si	DISTANCIA	esta	CENTRO	Y	VELOCIDAD	esta	RAPIDO A LA DERCHA	entonces	VOLTAJE	esta	ZERO
24	si	DISTANCIA	esta	DERECHA	Y	VELOCIDAD	esta	RAPIDO A LA DERCHA	entonces	VOLTAJE	esta	ZERO
25	si	DISTANCIA	esta	MUY A LA DERECHA	Y	VELOCIDAD	esta	RAPIDO A LA DERCHA	entonces	VOLTAJE	esta	NEGATIVO PEQUEÑO

3.10 Construcción del sistema en Matlab

El sistema se construyó en MATLAB versión 2014, en el módulo de diseño de lógica difusa (Matlab Fuzzy Logic Toolbox). El uso de este software ofrece un amplio soporte matemático, alta precisión, rápido prototipado, integración con dispositivos hardware.

Para utilizar el Toolbox de lógica difusa de MATLAB, tecleamos el comando fuzzy, desde la línea de comandos de Matlab, aparece una ventana como en la figura 3-14.

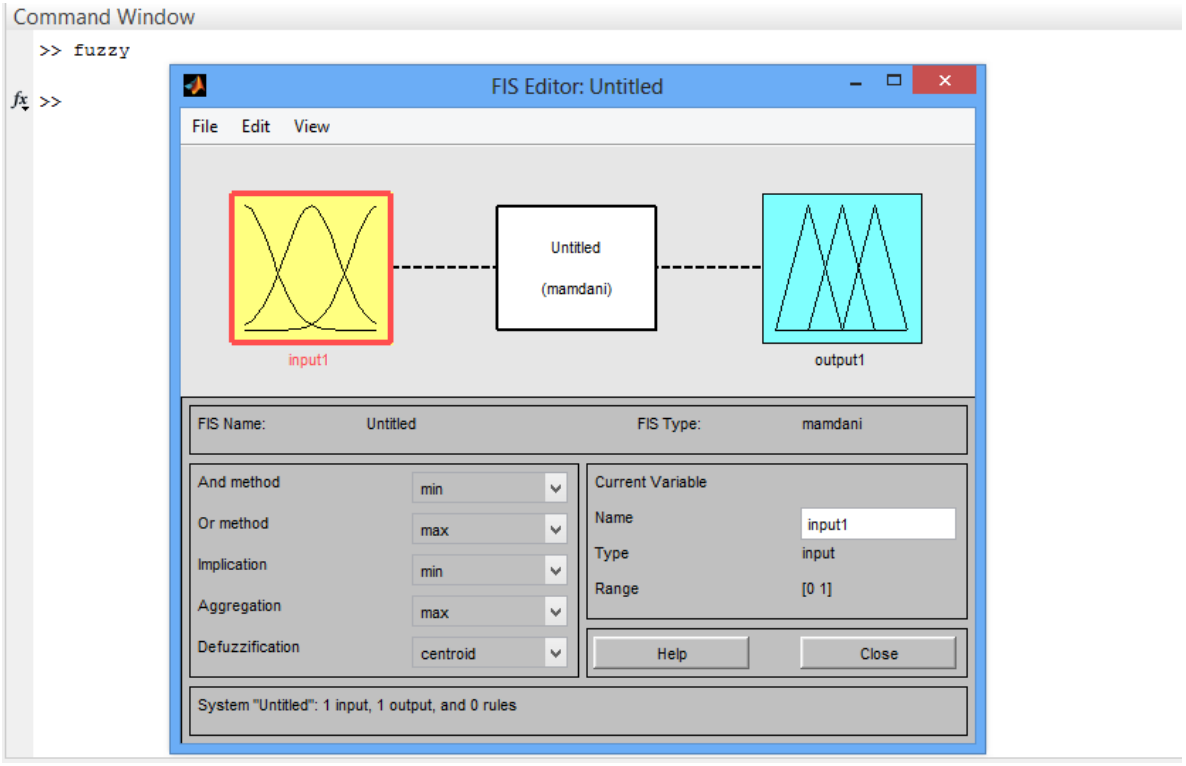


Figura 3-14 Ventana de edición de lógica difusa.

De acuerdo a los requerimientos del controlador, anexamos una entrada más al controlador como se muestra en la figura 3-15.

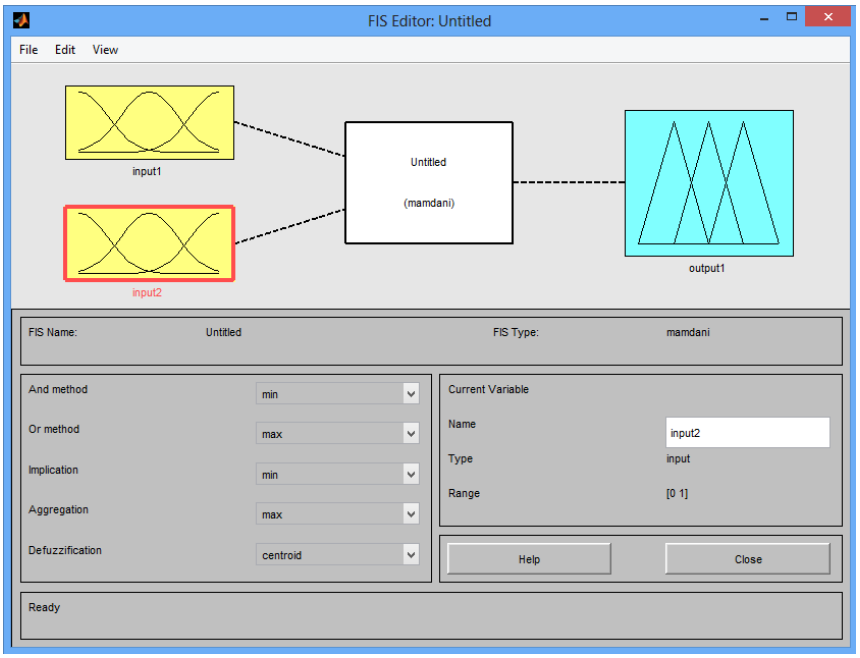


Figura 3-15 Controlador *con dos entradas una salida*

Seleccionamos cada entrada, salida y escribimos el nombre de cada una de ellas en la ventana “Name” (figura 3-16).

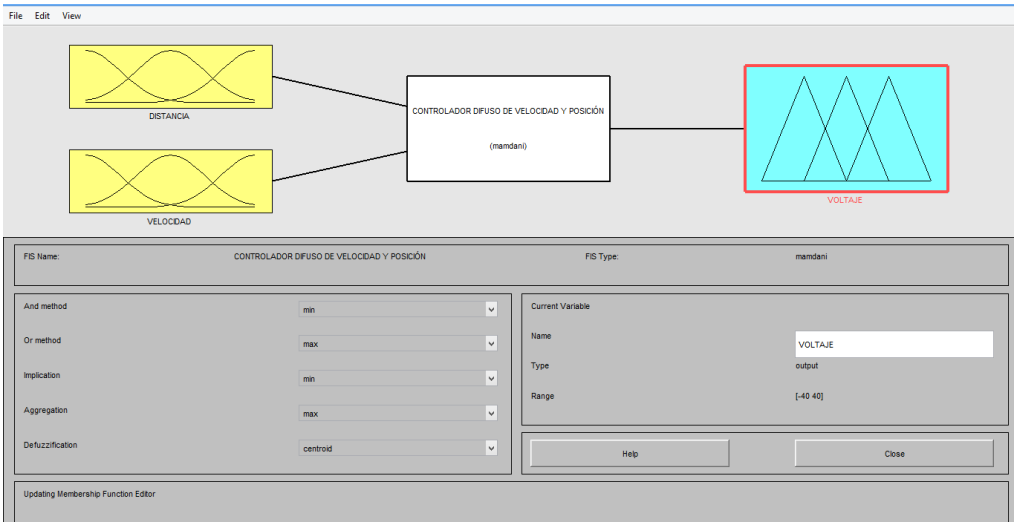


Figura 3-16 Asignación de nombres a entradas y salidas del sistema.

Para editar el conjunto difuso correspondiente a la variable DISTANCIA, primero la seleccionamos, y establecemos los límites en la ventana de rango (figura 3-17).

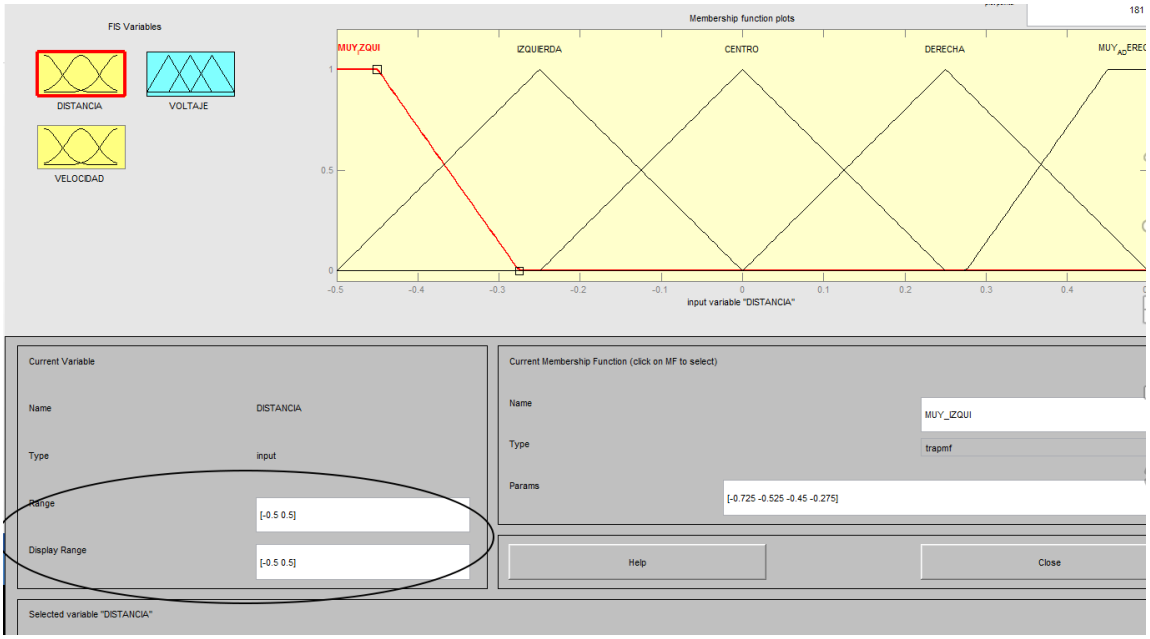


Figura 3-17 Establecer rango.

En esta ventana podemos establecer las funciones de membresía, el tipo función, el nombre, y los parámetros para cada conjunto difuso (figura 3-18).

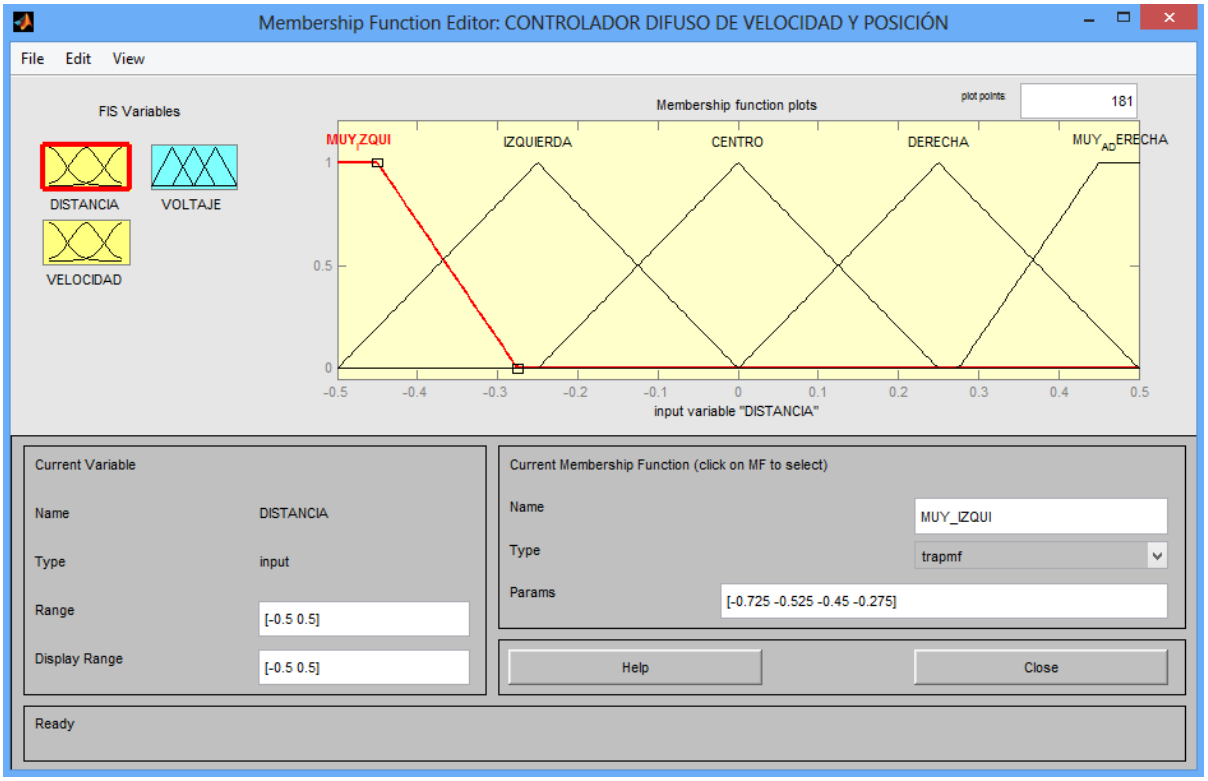


Figura 3-18 Asignación de las funciones de membresía para la entrada distancia.

Se establecen las funciones de membresía para la entrada VELOCIDAD (figura 3-19).

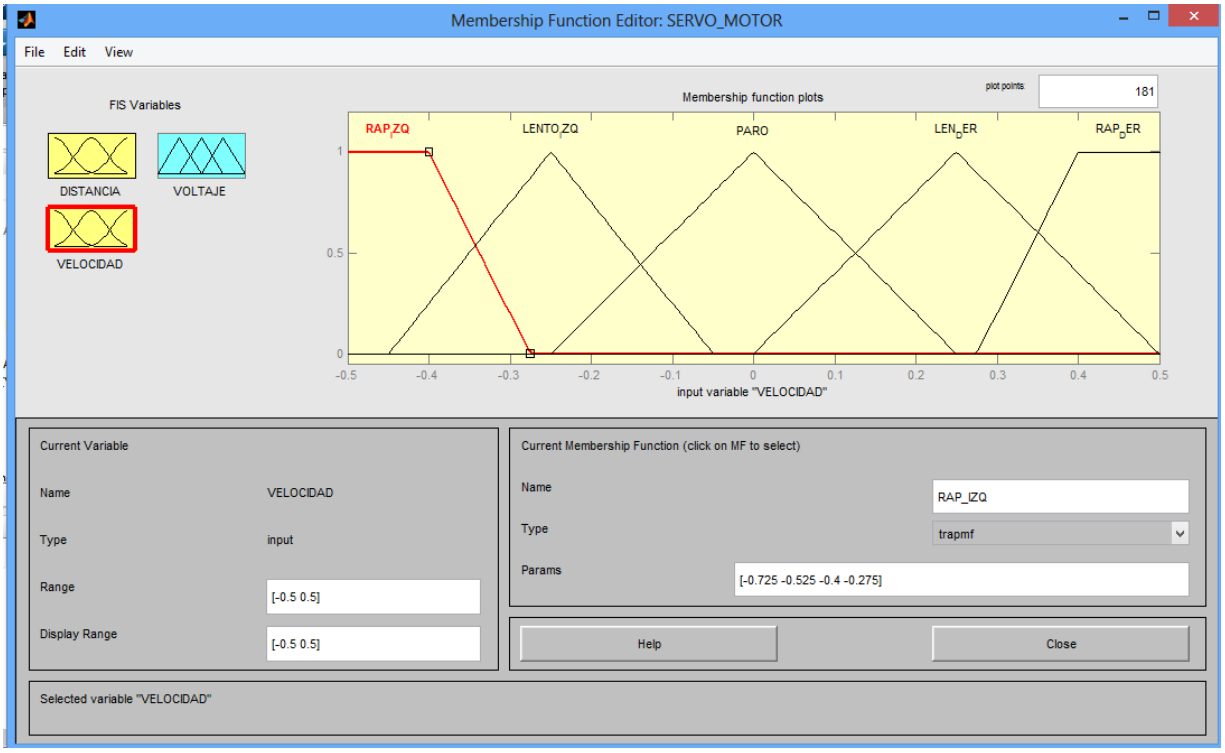


Figura 3-19 Funciones de membresía para la variable VELOCIDAD.

Se establecen las funciones de membresía para la entrada VOLTAJE (figura 3-20).

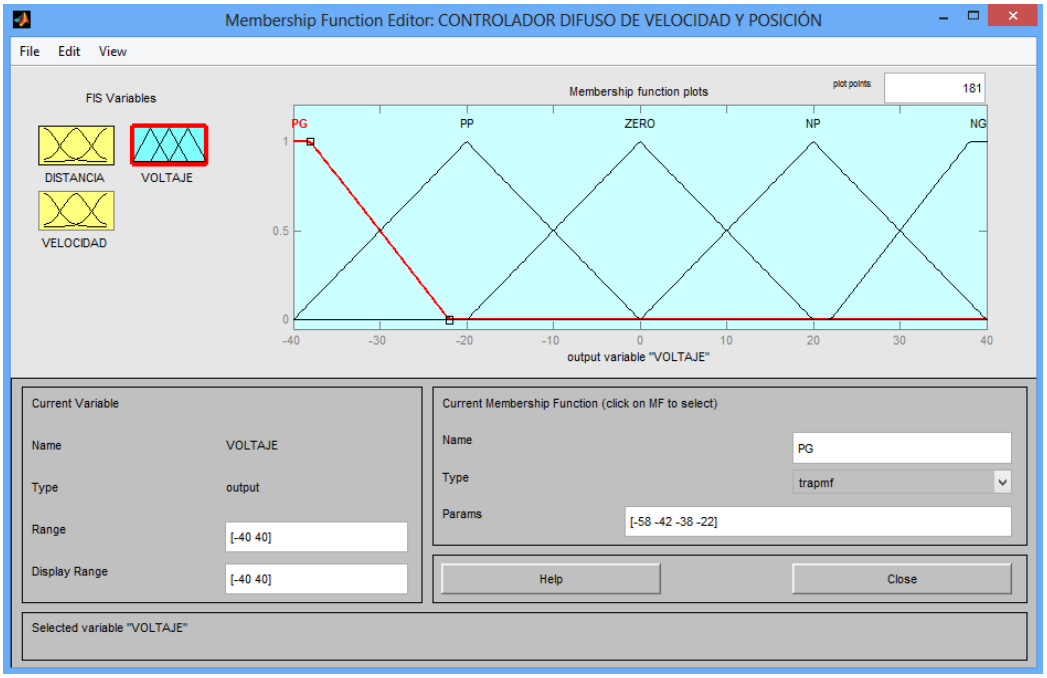


Figura 3-20 Funciones de membresía de salida VOLTAJE.

Ahora creamos las reglas de membresía, para lo cual hacemos clic en EDIT y luego seleccionamos la opción Rules (figura 3-21).

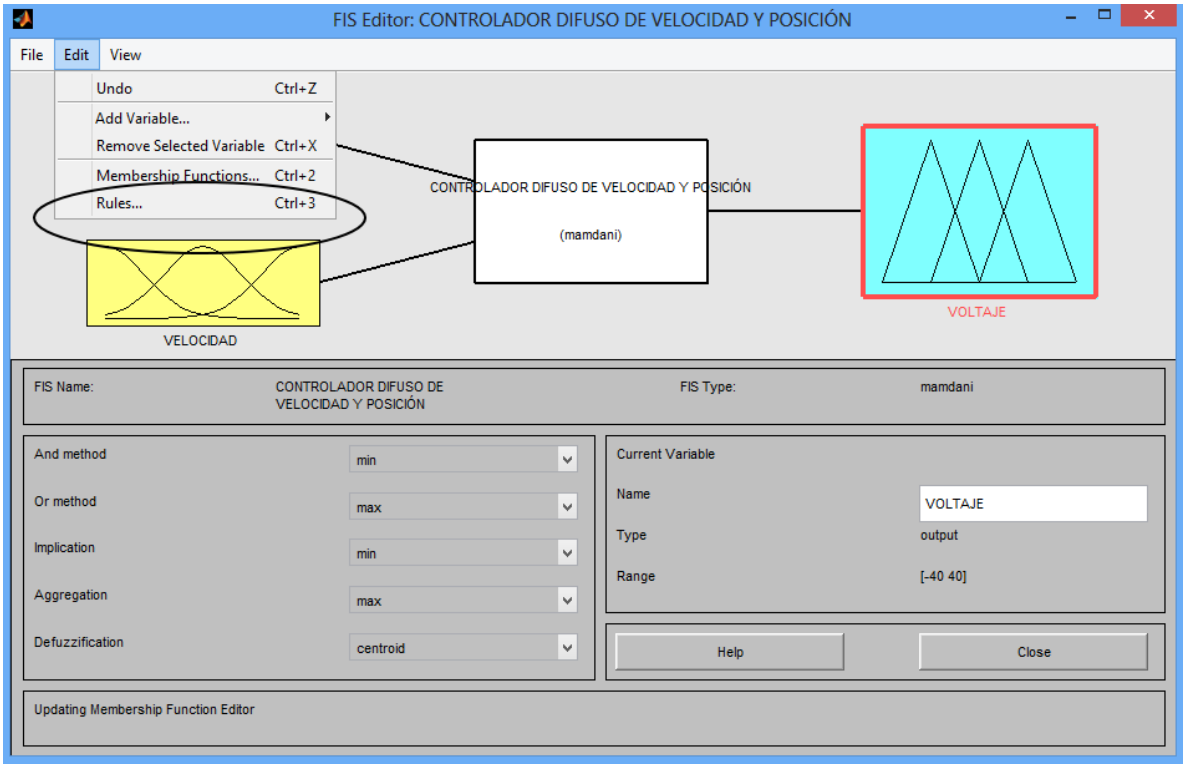


Figura 3-21 Edición de reglas lingüísticas.

En la ventana de edición de reglas lingüísticas, podemos establecer el motor de inferencia difusa con simples reglas condicionantes del tipo IF, THEN, ELSE, y conectivos del tipo “AND”, y “OR” (figura 3-22).

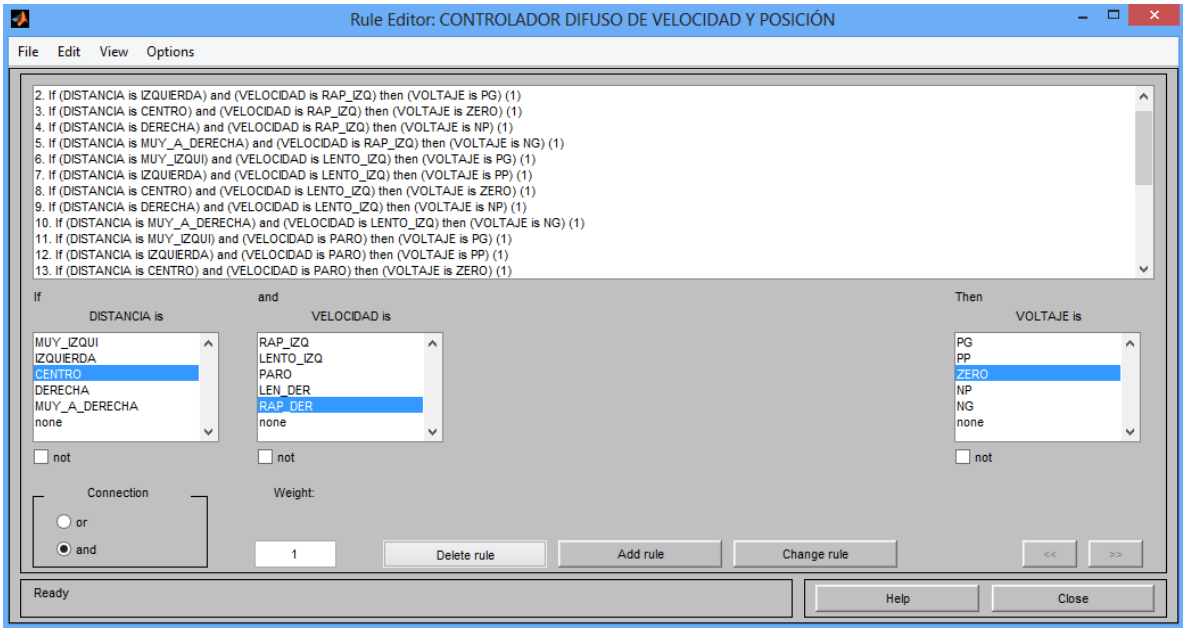


Figura 3-22 Establecimiento de reglas lingüísticas.

4 RESULTADOS EXPERIMENTALES

La acción sobre el proceso exige que el controlador brinde a su salida una magnitud dada, no difusa, por ejemplo, una tensión o una presión neumática dada, por lo que es necesario realizar una *Defuzzificación*, que realiza un mapeo a escala que convierte el rango de valores de las variables de salida al correspondiente universo discurso, y *defuzzifica* también la acción de control difusa inferida en una acción de control concreta.

En la práctica, para adoptar cierta acción de control, hay que tener muy en cuenta la posición del elemento final de control. Ésta necesariamente tiene que influir en el algoritmo, considerándola como parte de estado del proceso.

Utilizando nuevamente el editor de inferencia difuso FIS EDITOR de MATLAB, se simula el controlador difuso. Para tal efecto accedemos a la ventana RULE VIEWER, oprimiendo las teclas Ctrl + 5 como se muestra en la figura 4-1.

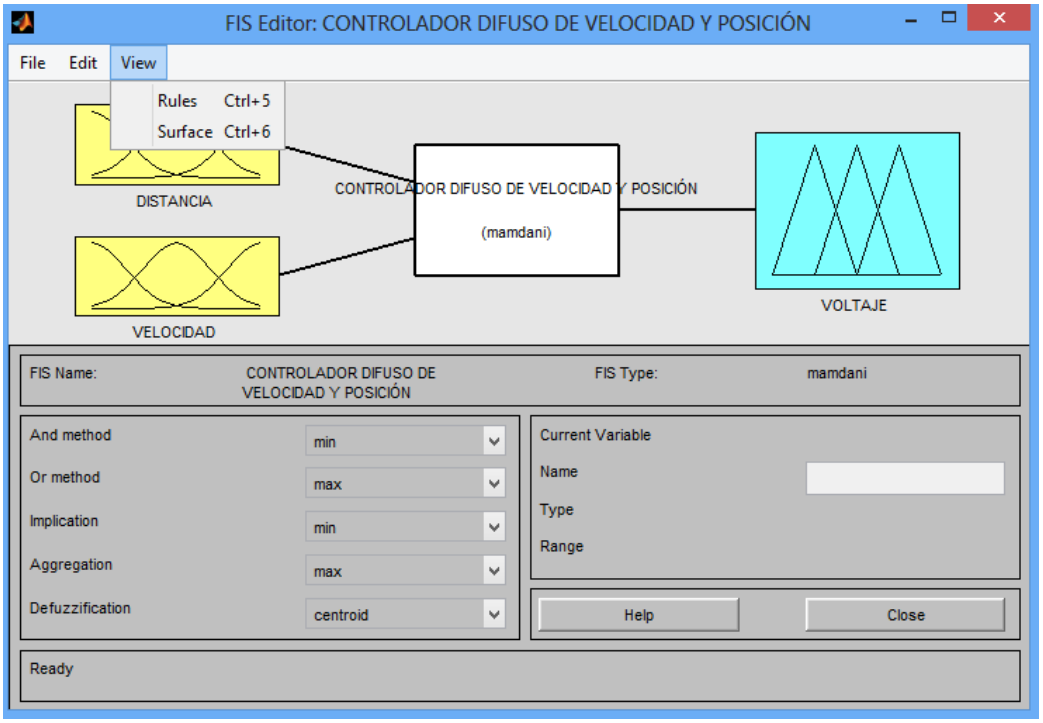
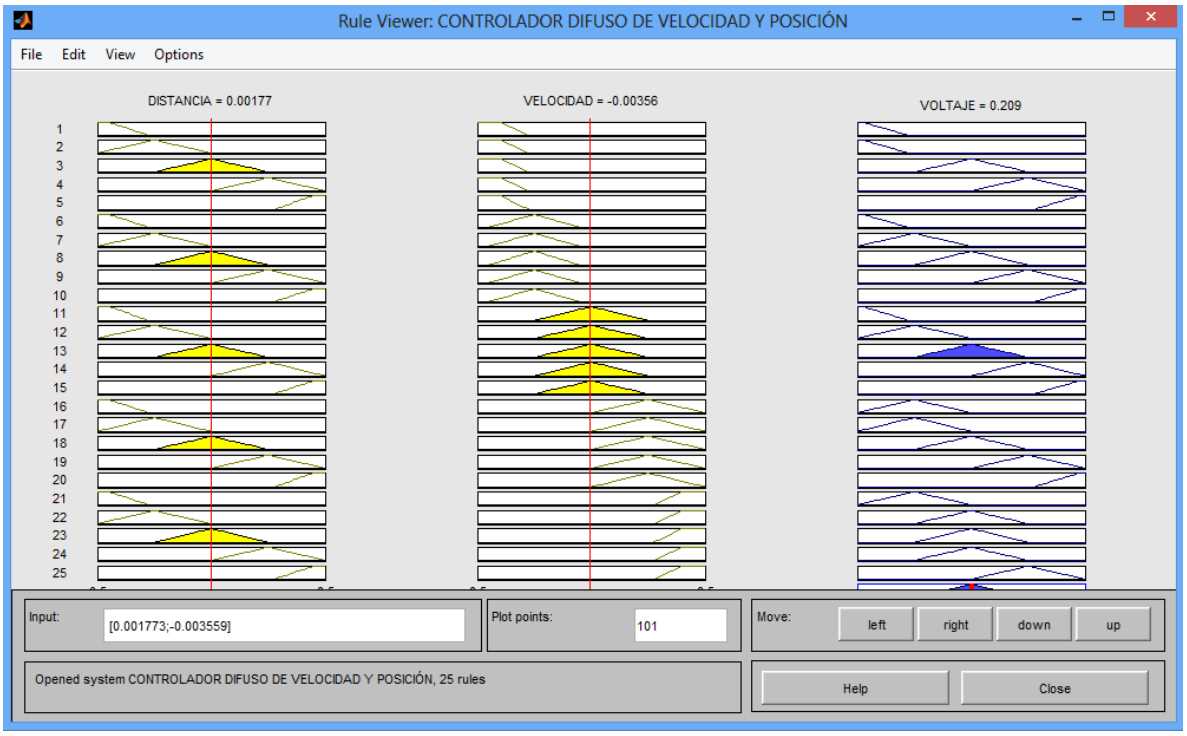


Figura 4-1 Método de acceso a ventana de Defuzzificación.

En esta ventana podemos observar de manera gráfica las 25 reglas lingüísticas ingresadas al motor de inferencia difusa.

Las entradas (DISTANCIA Y VELOCIDAD), pueden ser manipuladas de manera indistinta para poder observar la salida defuzzificada para diversos estados de estas mismas (Graficas 4-1).



Gráfica 4-1 Reglas y Defuzzificación.

Por otra parte podemos hacer uso de la opción Surface, (figura 4-2).

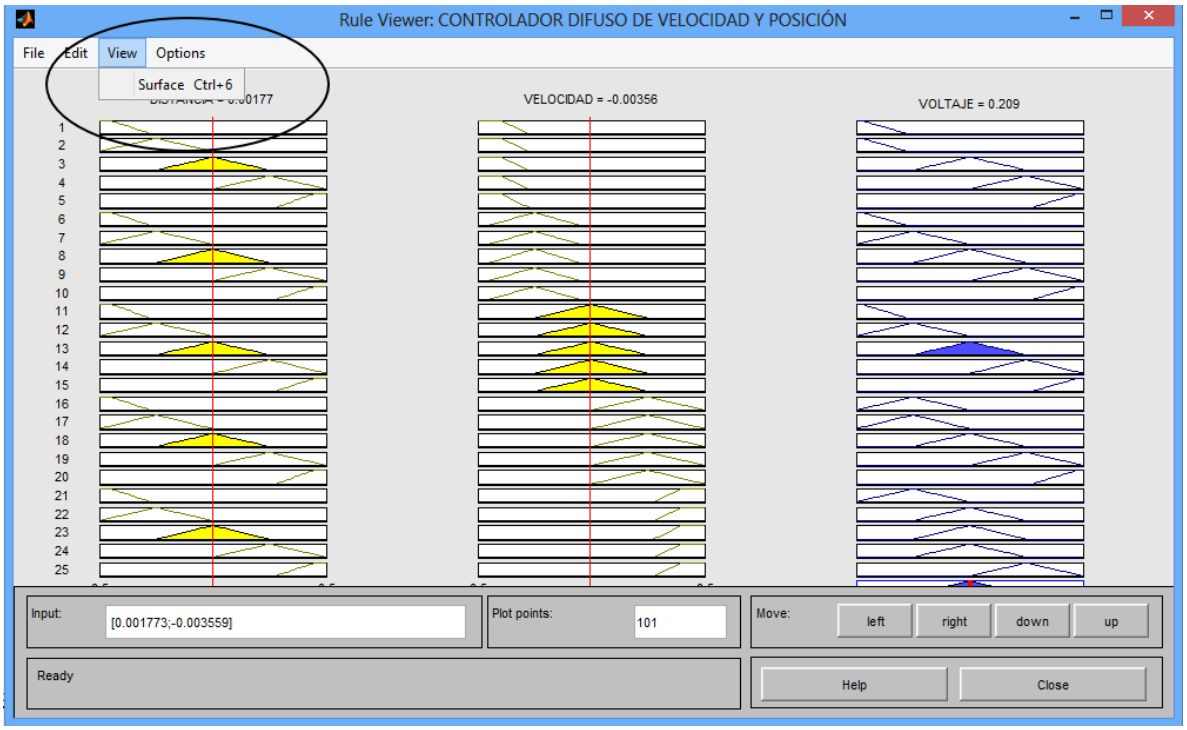
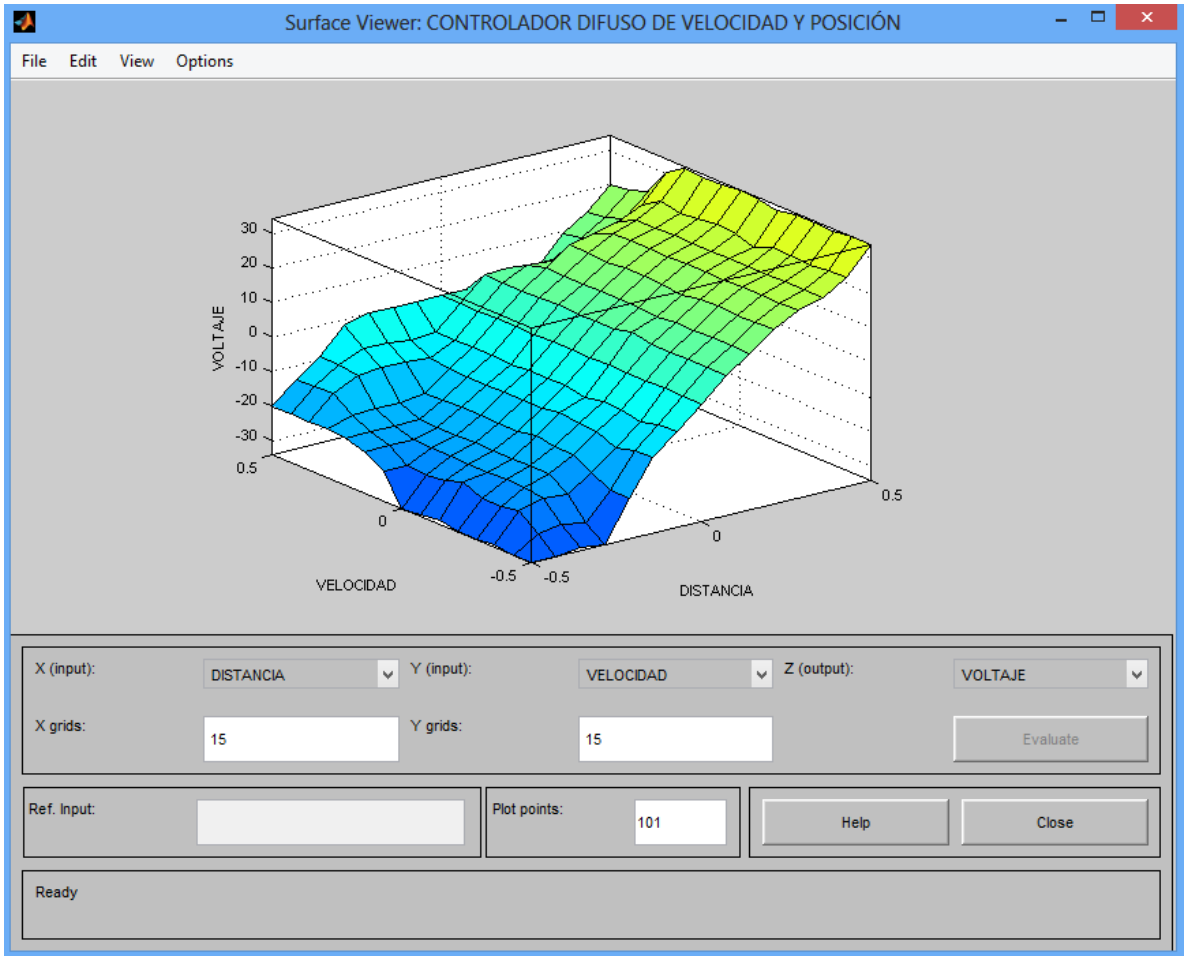


Figura 4-2

En la ventana Surface Viewer podemos observar el comportamiento de la salida defuzzificada con respecto a sus entradas difusas en 3D (gráfica 4-2).



Gráfica 4-2 Surface Viewer.

El diseño e implementación del software de control se hizo en la plataforma de programación gráfica Labview, la cual es idónea para aplicaciones que involucren adquisición, control, análisis y presentación de datos.

Uno de los factores que llevó a la utilización de esta plataforma de programación, fue su similitud con el Simulink de MATLAB (figura 4-3).

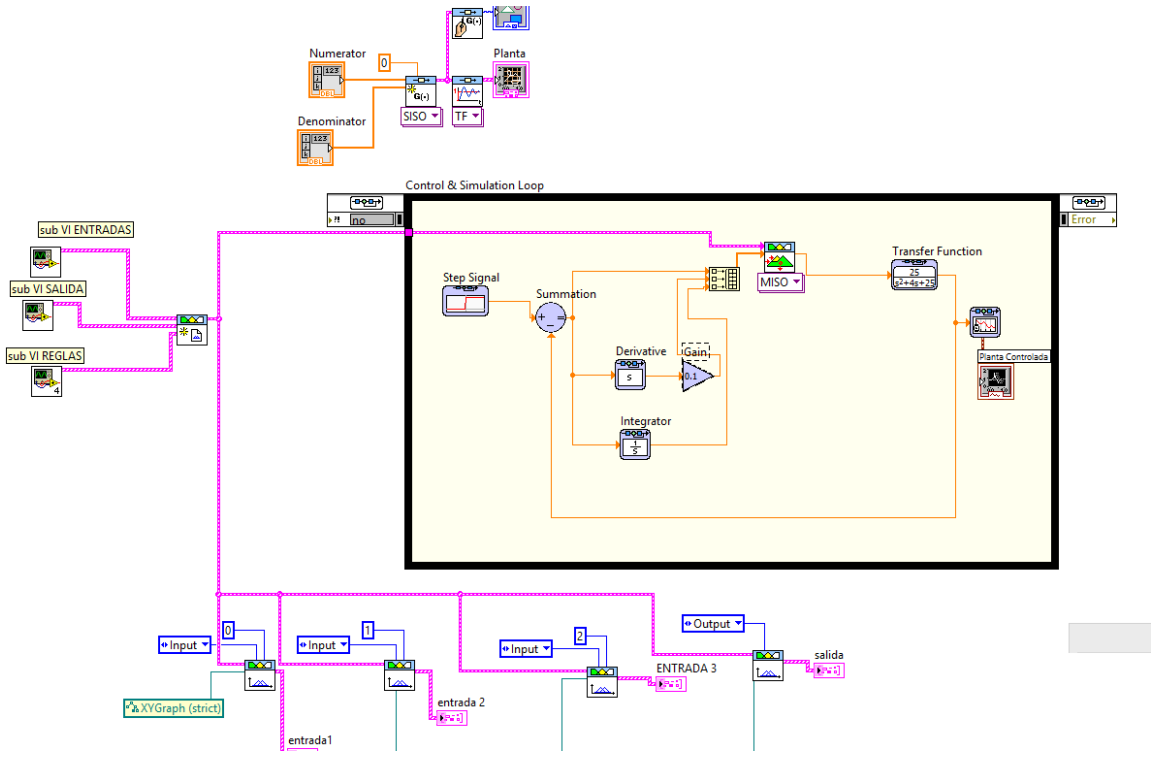
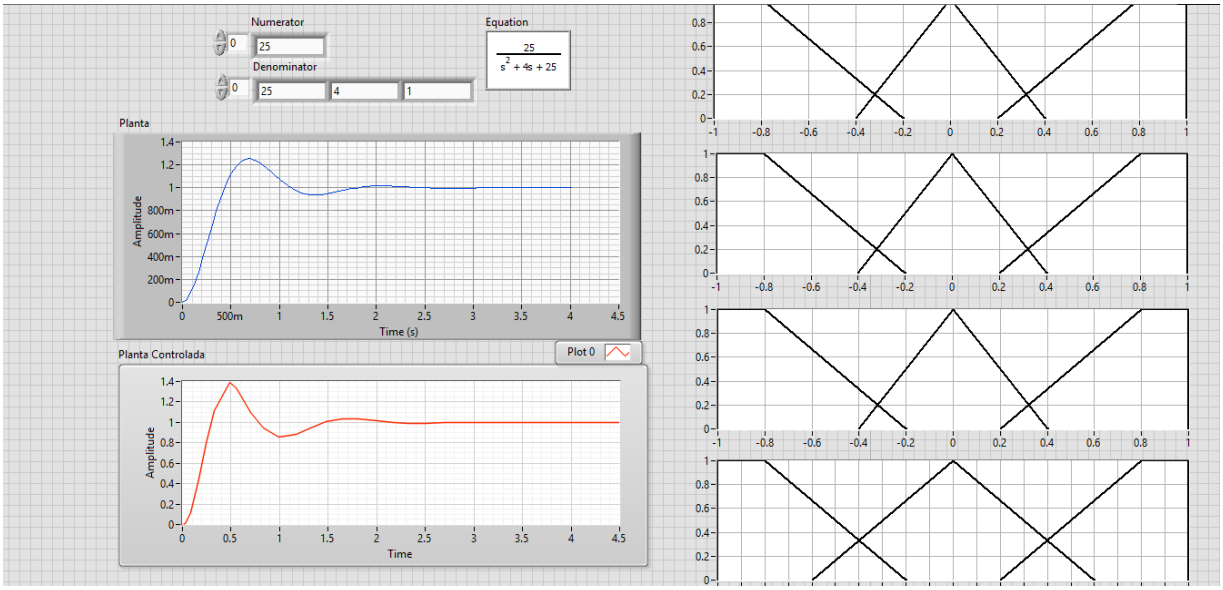


Figura 4-3 Implementación del algoritmo en Labview.

Finalmente se compara la planta basada en un controlador difuso, con otra planta de tipo lineal y obtenemos sus graficas correspondientes (gráfica 4-3).



Gráfica 4-3 Graficas obtenidas de planta controlada con lógica difusa y control lineal.

Como se puede observar, en la Gráfica 4-3, la curva obtenida en la planta controlada por lógica difusa tenemos un sobre paso. Para modificarla solo se modifica la constante derivativa, la consecuencia es que el sistema se vuelve más lento, sin

embargo esto es parte del proceso de sintonización, por lo que se modifica esta constante y se obtiene el siguiente resultado (figura 4-4).

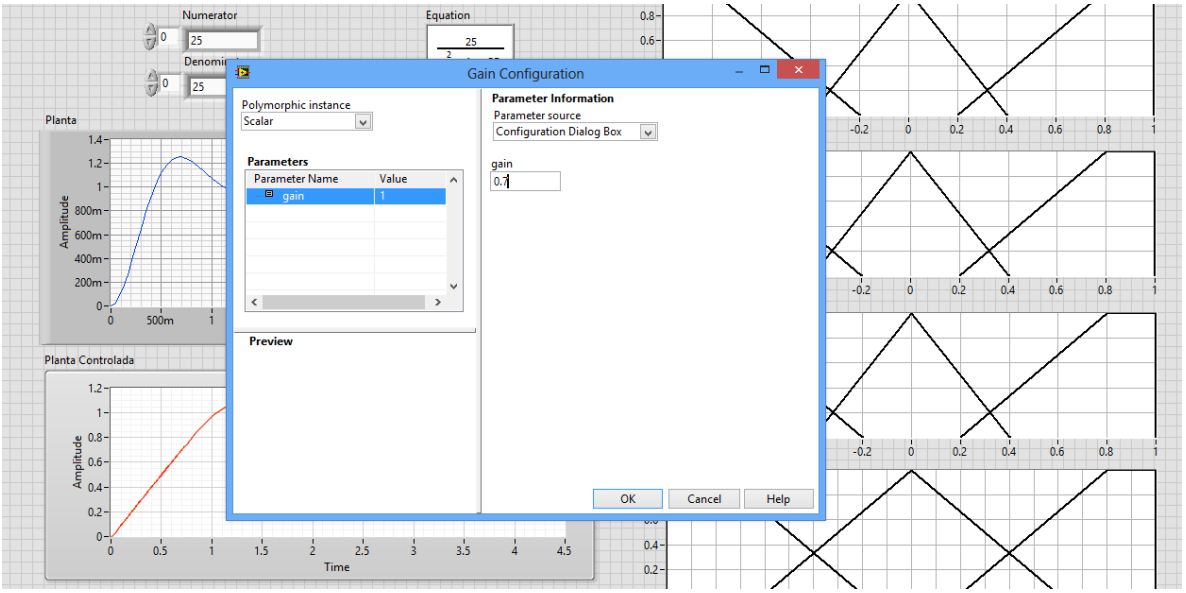
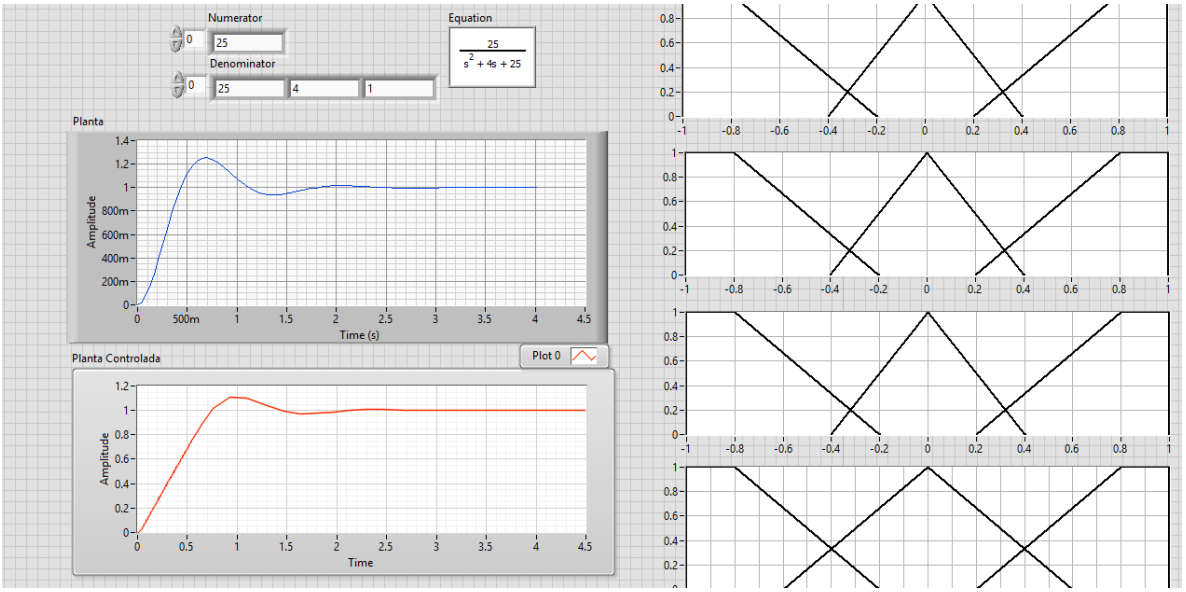


Figura 4-4 Modificación de la ganancia.

Finalmente en la gráfica 4-4, podemos observar que disminuyo en gran cantidad el sobrepaso, pero como ya se mencionó antes se volvió más lento.



Gráfica 4-4 Gráfica con ganancia modificada.

5 CONCLUSIONES

Al aplicar el controlador difuso en tiempo real, se comparó su respuesta con la entregada por el sistema convencional, observando que las señales de control correspondientes a cada controlador difuso presentan un mínimo de error en el objetivo del control y logran la estabilidad del elemento final de control. Por tanto, de esta forma se concluye que los sistemas difusos eliminan el efecto timbre sin perder velocidad de respuesta.

Las ventajas más importantes de los controladores difusos que se pudieron comprobar por medio de la realización de este trabajo, es que permiten incorporar en la automatización esquemas de razonamiento cualitativos, típicamente humanos.

Otra ventaja confirmada de los controladores difusos radica en que son menos sensibles a cambios en los parámetros o perturbaciones en comparación con los controles convencionales, ya que los controles difusos han demostrado ser más robustos que los tradicionales controles P, PI y PID. Además, los controles difusos tienen la ventaja de que sus parámetros pueden actualizarse de manera sencilla si los puntos de operación de la planta cambian. En muchos casos, incluso un operador no especializado en control puede mantener la base de reglas del control, dado que no es difícil de entender dicha base, porque las reglas utilizan variables lingüísticas en vez de variables numéricas.

La estabilidad del sistema se asegura al utilizar controladores en cascada. Esto hace que las perturbaciones en el lazo interno o secundario sean corregidas por el segundo controlador, antes de que puedan afectar la variable primaria; cualquier variación en la ganancia estática de la parte secundaria del proceso se compensa por su propio lazo y las constantes de tiempo asociadas al proceso secundario son reducidas drásticamente por el lazo secundario.

6 RECOMENDACIONES, LINEAS ABIERTAS O TRABAJOS FUTUROS

Es posible y altamente recomendable la utilización de este tipo de controladores en aplicaciones industriales. Tal es el caso de los actuadores usados en la robótica, o bien las máquinas de control numérico por computadora.

Los controladores para este tipo de dispositivos resultan ser muy caros, y a veces inalcanzables para las micro y pequeñas empresas, de manera que un trabajo a futuro es, el desarrollar un programa que emule un engrane electrónico como el que actualmente se utiliza con el control lineal, y que tenga la capacidad de recibir código “G”, con el fin de interpretarlo, procesarlo y ejecutar las secuencias de movimientos programadas por algún software de diseño y mecanizado.

También podría tener la capacidad de funcionar con tarjetas de control y adquisición de datos, Open Source, como Arduino one, o Arduino Mega, o directamente en un microcontrolador de tipo AVR.

7 APENDICES Y ANEXOS

ANEXO A.- Código para la creación del controlador difuso en Matlab.

```
[System]
Name='SERVO_MOTOR'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=25
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='DISTANCIA'
Range=[-0.5 0.5]
NumMFs=5
MF1='MUY_IZQUI':'trimf',[-0.75 -0.5 -0.25]
MF2='IZQUIERDA':'trimf',[-0.5 -0.25 0]
MF3='CENTRO':'trimf',[-0.25 0 0.25]
MF4='DERECHA':'trimf',[0 0.25 0.5]
MF5='MUY_A_DERECHA':'trimf',[0.25 0.5 0.75]

[Input2]
Name='VELOCIDAD'
Range=[-0.5 0.5]
NumMFs=5
MF1='RAP_IZQ':'trapmf',[-0.725 -0.525 -0.4 -0.275]
MF2='LENTO_IZQ':'trimf',[-0.45 -0.25 -0.05]
MF3='PARO':'trimf',[-0.25 0 0.25]
MF4='LEN_DER':'trimf',[0 0.25 0.5]
MF5='RAP_DER':'trapmf',[0.275 0.4 0.525 0.725]

[Output1]
Name='VOLTAJE'
Range=[-40 40]
NumMFs=5
MF1='PG':'trapmf',[-58 -42 -38 -22]
MF2='PP':'trimf',[-40 -20 0]
MF3='ZERO':'trimf',[-20 -2.22e-16 20]
MF4='NP':'trimf',[0 20 40]
MF5='NG':'trapmf',[22 38 42 58]

[Rules]
1 1, 1 (1) : 1
2 1, 1 (1) : 1
3 1, 3 (1) : 1
4 1, 4 (1) : 1
5 1, 5 (1) : 1
1 2, 1 (1) : 1
2 2, 2 (1) : 1
```

```
3 2, 3 (1) : 1
4 2, 4 (1) : 1
5 2, 5 (1) : 1
1 3, 1 (1) : 1
2 3, 2 (1) : 1
3 3, 3 (1) : 1
4 3, 4 (1) : 1
5 3, 5 (1) : 1
1 4, 2 (1) : 1
2 4, 2 (1) : 1
3 4, 3 (1) : 1
4 4, 4 (1) : 1
5 4, 5 (1) : 1
1 5, 2 (1) : 1
2 5, 3 (1) : 1
3 5, 3 (1) : 1
4 5, 3 (1) : 1
5 5, 4 (1) : 1
```

Anexo B.- Implementación de Controlador Fuzzy en Matlab

```
open_system('sllookuptable')
open_system('sllookuptable/Fuzzy PID')
open_system('sllookuptable')
open_system('sllookuptable/Conventional PID')
C0 = pid(1,1,1,'Ts',Ts,'IF','B','DF','B'); % define PID structure
C = pidtune(Plant,C0) %#ok<*NOPTS> % design PID
[Kp, Ki, Kd] = piddata(C); % obtain PID gains

C =

             Ts*z           z-1
  Kp + Ki * ------ + Kd * ------
              z-1          Ts*z

  with Kp = 30, Ki = 28.6, Kd = 6.9, Ts = 0.1

Sample time: 0.1 seconds
Discrete-time PID controller in parallel form.

FIS = newfis('FIS','mamdani','prod','probor','prod','sum');
FIS = addvar(FIS,'input','E',[-10 10]);
FIS = addmf(FIS,'input',1,'Negative','trimf',[-20 -10 0]);
FIS = addmf(FIS,'input',1,'Zero','trimf',[-10 0 10]);
FIS = addmf(FIS,'input',1,'Positive','trimf',[0 10 20]);
FIS = addvar(FIS,'input','CE',[-10 10]);
FIS = addmf(FIS,'input',2,'Negative','trimf',[-20 -10 0]);
FIS = addmf(FIS,'input',2,'Zero','trimf',[-10 0 10]);
FIS = addmf(FIS,'input',2,'Positive','trimf',[0 10 20]);
FIS = addvar(FIS,'output','u',[-20 20]);
FIS = addmf(FIS,'output',1,'LargeNegative','trimf',[-20 -20 -20]);
FIS = addmf(FIS,'output',1,'SmallNegative','trimf',[-10 -10 -10]);
FIS = addmf(FIS,'output',1,'Zero','trimf',[0 0 0]);
FIS = addmf(FIS,'output',1,'SmallPositive','trimf',[10 10 10]);
FIS = addmf(FIS,'output',1,'LargePositive','trimf',[20 20 20]);
ruleList = [1 1 1 1 1;...   % Rule 1
            1 2 2 1 1;...   % Rule 2
            1 3 3 1 1;...   % Rule 3
            2 1 2 1 1;...   % Rule 4
            2 2 3 1 1;...   % Rule 5
            2 3 4 1 1;...   % Rule 6
            3 1 3 1 1;...   % Rule 7
            3 2 4 1 1;...   % Rule 8
            3 3 5 1 1];     % Rule 9
FIS = addrule(FIS,ruleList);
gensurf(FIS)
GE = 10;
GCE = GE*(Kp-sqrt(Kp^2-4*Ki*Kd))/2/Ki;
GCU = Ki/GE;
GU = Kd/GCE;
Step = 10; % use 3 break points for both E and CE inputs
E = -10:Step:10;
CE = -10:Step:10;
N = length(E);
LookUpTableData = zeros(N);
```

```
for i=1:N
   for j=1:N
      % compute output u for each combination of break points
      LookUpTableData(i,j) = evalfis([E(i) CE(j)],FIS);
   end
end
open_system('sllookuptable/Fuzzy PID using Lookup Table')
open_system('sllookuptable/Scope')
sim('sllookuptable')
```

Anexo C.- Codigo del firmware LVIFA que proporciona las funciones de interfaz para Arduino ONE.

```
/************************************************************************
******
 **
 **
 **
 **  Written By:    Sam Kristoff - National Instruments
 **  Written On:    November 2010
 **  Last Updated:  Dec 2011 - Kevin Fort - National Instruments
 **
 **  This File May Be Modified And Re-Distributed Freely. Original File
Content
 **  Written By Sam Kristoff And Available At www.ni.com/arduino.
 **

************************************************************************
*****/

#include <Wire.h>
#include <SPI.h>
#include <LiquidCrystal.h>

//Includes for IR Remote
#ifndef IRremoteInt_h
#include "IRremoteInt.h"
#endif
#ifndef IRremote_h
#include "IRremote.h"
#endif



/************************************************************************
******
 ** Optionally Include And Configure Stepper Support

************************************************************************
*****/
#ifdef STEPPER_SUPPORT

  // Stepper Modifications
  #include "AFMotor.h"
  #include "AccelStepper.h"

  // Adafruit shield
  AF_Stepper motor1(200, 1);
  AF_Stepper motor2(200, 2);

  // you can change these to DOUBLE or INTERLEAVE or MICROSTEP
  // wrappers for the first motor
  void forwardstep1() {
      motor1.onestep(FORWARD, SINGLE);
  }
  void backwardstep1() {
```

```
      motor1.onestep(BACKWARD, SINGLE);
  }
  // wrappers for the second motor
  void forwardstep2() {
      motor2.onestep(FORWARD, SINGLE);
  }
  void backwardstep2() {
      motor2.onestep(BACKWARD, SINGLE);
  }

  AccelStepper steppers[8];  //Create array of 8 stepper objects

#endif

// Variables

unsigned int retVal;
int sevenSegmentPins[8];
int currentMode;
unsigned int freq;
unsigned long duration;
int i2cReadTimeouts = 0;
char spiBytesToSend = 0;
char spiBytesSent = 0;
char spiCSPin = 0;
char spiWordSize = 0;
Servo *servos;
byte customChar[8];
LiquidCrystal lcd(0,0,0,0,0,0,0);
unsigned long IRdata;
IRsend irsend;

// Sets the mode of the Arduino (Reserved For Future Use)
void setMode(int mode)
{
  currentMode = mode;
}

// Checks for new commands from LabVIEW and processes them if any exists.
int checkForCommand(void)
{
#ifdef STEPPER_SUPPORT
  // Call run function as fast as possible to keep motors turning
  for (int i=0; i<8; i++){
      steppers[i].run();
  }
#endif

  int bufferBytes = Serial.available();

  if(bufferBytes >= COMMANDLENGTH)
  {
      // New Command Ready, Process It
      // Build Command From Serial Buffer
      for(int i=0; i<COMMANDLENGTH; i++)
      {
        currentCommand[i] = Serial.read();
```

```
      }
      processCommand(currentCommand);
      return 1;
  }
  else
  {
      return 0;
  }
}

// Processes a given command
void processCommand(unsigned char command[])
{
  // Determine Command
  if(command[0] == 0xFF && checksum_Test(command) == 0)
  {
      switch(command[1])
      {
      /*************************************************************************
************
      ** LIFA Maintenance Commands
      *************************************************************************
***********/
      case 0x00:      // Sync Packet
        Serial.print("sync");
        Serial.flush();
        break;
      case 0x01:     // Flush Serial Buffer
        Serial.flush();
        break;

      /*************************************************************************
************
      ** Low Level - Digital I/O Commands
      *************************************************************************
***********/
      case 0x02:     // Set Pin As Input Or Output
        pinMode(command[2], command[3]);
        Serial.write('0');
        break;
      case 0x03:     // Write Digital Pin
        digitalWrite(command[2], command[3]);
         Serial.write('0');
        break;
      case 0x04:     // Write Digital Port 0
        writeDigitalPort(command);
         Serial.write('0');
        break;
      case 0x05:     //Tone
        freq = ( (command[3]<<8) + command[4]);
        duration=(command[8]+  (command[7]<<8)+
(command[6]<<16)+(command[5]<<24));

        if(freq > 0)
        {
            tone(command[2], freq, duration);
        }
```

```
        else
        {
            noTone(command[2]);
        }
         Serial.write('0');
        break;
      case 0x06:    // Read Digital Pin
        retVal = digitalRead(command[2]);
        Serial.write(retVal);
        break;
      case 0x07:    // Digital Read Port
        retVal = 0x0000;
        for(int i=0; i <=13; i++)
        {
            if(digitalRead(i))
            {
              retVal += (1<<i);
            }
        }
        Serial.write( (retVal & 0xFF));
        Serial.write( (retVal >> 8));
        break;

      /************************************************************************
************
      ** Low Level - Analog Commands
      ************************************************************************
***********/
      case 0x08:    // Read Analog Pin
        retVal = analogRead(command[2]);
        Serial.write( (retVal >> 8));
        Serial.write( (retVal & 0xFF));
        break;
      case 0x09:    // Analog Read Port
        analogReadPort();
        break;

      /************************************************************************
************
      ** Low Level - PWM Commands
      ************************************************************************
***********/
      case 0x0A:    // PWM Write Pin
        analogWrite(command[2], command[3]);
         Serial.write('0');
        break;
      case 0x0B:    // PWM Write 3 Pins
        analogWrite(command[2], command[5]);
        analogWrite(command[3], command[6]);
        analogWrite(command[4], command[7]);
         Serial.write('0');
        break;

      /************************************************************************
************
      ** Sensor Specific Commands
```

```
      ************************************************************************
***********/
      case 0x0C:    // Configure Seven Segment Display
        sevenSegment_Config(command);
         Serial.write('0');
        break;
      case 0x0D:    // Write To Seven Segment Display
        sevenSegment_Write(command);
         Serial.write('0');
        break;

      /************************************************************************
***********
      **  I2C
      ************************************************************************
***********/
      case 0x0E:    // Initialize I2C
        Wire.begin();
         Serial.write('0');
        break;
      case 0x0F:    // Send I2C Data
        Wire.beginTransmission(command[3]);
        for(int i=0; i<command[2]; i++)
        {
            #if defined(ARDUINO) && ARDUINO >= 100
              Wire.write(command[i+4]);
            #else
              Wire.send(command[i+4]);
            #endif

        }
        Wire.endTransmission();
         Serial.write('0');
        break;
      case 0x10:    // I2C Read
        i2cReadTimeouts = 0;
        Wire.requestFrom(command[3], command[2]);
        while(Wire.available() < command[2])
        {
            i2cReadTimeouts++;
            if(i2cReadTimeouts > 100)
            {
              return;
            }
            else
            {
              delay(1);
            }
        }

        for(int i=0; i<command[2]; i++)
        {
            #if defined(ARDUINO) && ARDUINO >= 100
              Serial.write(Wire.read());
            #else
              Serial.write(Wire.receive());
            #endif
```

```
        }
        break;

      /************************************************************************
************
      ** SPI
      ************************************************************************
***********/
      case 0x11:    // SPI Init
        SPI.begin();
         Serial.write('0');
        break;
      case 0x12:    // SPI Set Bit Order (MSB LSB)
        if(command[2] == 0)
        {
            SPI.setBitOrder(LSBFIRST);
        }
        else
        {
            SPI.setBitOrder(MSBFIRST);
        }
         Serial.write('0');
        break;
      case 0x13:    // SPI Set Clock Divider
        spi_setClockDivider(command[2]);
         Serial.write('0');
        break;
      case 0x14:    // SPI Set Data Mode
        switch(command[2])
        {
        case 0:
            SPI.setDataMode(SPI_MODE0);
            break;
        case 1:
            SPI.setDataMode(SPI_MODE1);
            break;
        case 2:
            SPI.setDataMode(SPI_MODE2);
            break;
        case 3:
            SPI.setDataMode(SPI_MODE3);
            break;
        default:
            break;
        }
         Serial.write('0');
        break;
      case 0x15:  // SPI Send / Receive
        spi_sendReceive(command);
        break;
      case 0x16:  // SPI Close
        SPI.end();
         Serial.write('0');
        break;
```

```
      /************************************************************************
************
      ** Servos
      ************************************************************************
***********/
      case 0x17:  // Set Num Servos
        free(servos);
        servos = (Servo*) malloc(command[2]*sizeof(Servo));
        for(int i=0; i<command[2]; i++)
        {
            servos[i] = Servo();
        }
        if(servos == 0)
        {
            Serial.write('1');
        }
        else
        {
             Serial.write('0');
        }
        break;
      case 0x18:  // Configure Servo
        servos[command[2]].attach(command[3]);
         Serial.write('0');
        break;
      case 0x19:  // Servo Write
        servos[command[2]].write(command[3]);
         Serial.write('0');
        break;
      case 0x1A:  // Servo Read Angle
        Serial.write(servos[command[2]].read());
        break;
      case 0x1B:  // Servo Write uS Pulse
        servos[command[2]].writeMicroseconds( (command[3] + (command[4]<<8))
);
         Serial.write('0');
        break;
      case 0x1C:  // Servo Read uS Pulse
        retVal = servos[command[2]].readMicroseconds();
        Serial.write ((retVal & 0xFF));
        Serial.write( (retVal >> 8));
        break;
      case 0x1D:  // Servo Detach
        servos[command[2]].detach();
         Serial.write('0');
        break;

/************************************************************************
******
            **                                          LCD

************************************************************************
*****/
      case 0x1E:  // LCD Init
            lcd.init(command[2], command[3], command[4], command[5],
command[6], command[7], command[8], command[9], command[10], command[11],
command[12], command[13]);
```

```
    Serial.write('0');
    break;
case 0x1F:  // LCD Set Size
    lcd.begin(command[2], command[3]);
    Serial.write('0');
    break;
case 0x20:  // LCD Set Cursor Mode
    if(command[2] == 0)
    {
      lcd.noCursor();
    }
    else
    {
      lcd.cursor();
    }
    if(command[3] == 0)
    {
      lcd.noBlink();
    }
    else
    {
      lcd.blink();
    }
    Serial.write('0');
    break;
case 0x21:  // LCD Clear
    lcd.clear();
    Serial.write('0');
    break;
case 0x22:  // LCD Set Cursor Position
    lcd.setCursor(command[2], command[3]);
    Serial.write('0');
    break;
case 0x23:  // LCD Print
    lcd_print(command);
    break;
case 0x24:  // LCD Display Power
    if(command[2] == 0)
    {
      lcd.noDisplay();
    }
    else
    {
      lcd.display();
    }
    Serial.write('0');
    break;
case 0x25:  // LCD Scroll
    if(command[2] == 0)
    {
      lcd.scrollDisplayLeft();
    }
    else
    {
      lcd.scrollDisplayRight();
    }
```

```
            Serial.write('0');
            break;
        case 0x26: //  LCD Autoscroll
            if(command[2] == 0)
            {
              lcd.noAutoscroll();
            }
            else
            {
              lcd.autoscroll();
            }
            Serial.write('0');
            break;
        case 0x27: // LCD Print Direction
            if(command[2] == 0)
            {
              lcd.rightToLeft();
            }
            else
            {
              lcd.leftToRight();
            }
            Serial.write('0');
            break;
        case 0x28: // LCD Create Custom Char
            for(int i=0; i<8; i++)
            {
              customChar[i] = command[i+3];
            }
            lcd.createChar(command[2], customChar);

            Serial.write('0');
            break;
        case 0x29:  // LCD Print Custom Char
            lcd.write(command[2]);
            Serial.write('0');
            break;


      /*************************************************************************
************
      ** Continuous Aquisition
      *************************************************************************
***********/
        case 0x2A:  // Continuous Aquisition Mode On
            acqMode=1;
            contAcqPin=command[2];
            contAcqSpeed=(command[3])+(command[4]<<8);
            acquisitionPeriod=1/contAcqSpeed;
            iterationsFlt =.08/acquisitionPeriod;
            iterations=(int)iterationsFlt;
            if(iterations<1)
            {
              iterations=1;
            }
            delayTime= acquisitionPeriod;
         if(delayTime<0)
```

```
          {
              delayTime=0;
          }
             break;
         case 0x2B:  // Continuous Aquisition Mode Off
             acqMode=0;
             break;
        case 0x2C:  // Return Firmware Revision
              Serial.write(byte(FIRMWARE_MAJOR));
              Serial.write(byte(FIRMWARE_MINOR));
             break;
        case 0x2D:  // Perform Finite Aquisition
              Serial.write('0');

finiteAcquisition(command[2],(command[3])+(command[4]<<8),command[5]+(comman
d[6]<<8));
             break;
       /************************************************************************
************
       ** Stepper
       ************************************************************************
***********/
       #ifdef STEPPER_SUPPORT
         case 0x30:  // Configure Stepper
             if (command[2] == 5){    // Support AFMotor Shield
               switch (command[3]){
               case 0:
                   steppers[command[3]] = AccelStepper(forwardstep1,
backwardstep1);
                   break;
               case 1:
                   steppers[command[3]] = AccelStepper(forwardstep2,
backwardstep2);
                   break;
               default:
                   break;
               }
             }
             else if(command[2]==6) {                    // All other stepper
configurations
               steppers[command[3]] = AccelStepper(1,
command[4],command[5],command[6],command[7]);
             }
             else{
               steppers[command[3]] = AccelStepper(command[2],
command[4],command[5],command[6],command[7]);
             }
              Serial.write('0');
             break;
        case 0x31:  // Stepper Write
             AccelStepper_Write(command);
              Serial.write('0');
             break;
        case 0x32:  // Stepper Detach
             steppers[command[2]].disableOutputs();
              Serial.write('0');
             break;
```

```
        case 0x33:  // Stepper steps to go
            retVal = 0;
            for(int i=0; i<8; i++){
              retVal += steppers[i].distanceToGo();
            }
            Serial.write( (retVal & 0xFF) );
            Serial.write( (retVal >> 8) );

            break;
      #endif

      /************************************************************************
***********
      ** IR Transmit
      ************************************************************************
**********/
      case 0x34:  // IR Transmit
        IRdata = ((unsigned long)command [4] << 24) | ((unsigned
long)command [5] << 16) | ((unsigned long)command [6] << 8) | ((unsigned
long)command [7]);
        switch(command[2])
        {
            case 0x00:  // NEC
              irsend.sendNEC(IRdata, command[3]);
              break;
            case 0x01:  //Sony
              irsend.sendSony(IRdata, command[3]);
              break;
            case 0x02:  //RC5
              irsend.sendRC5(IRdata, command[3]);
              break;
            case 0x03:  //RC6
              irsend.sendRC6(IRdata, command[3]);
              break;
        }
        Serial.write((IRdata>>16) & 0xFF);
        break;
      /************************************************************************
***********
      ** Unknown Packet
      ************************************************************************
**********/
      default:      // Default Case
        Serial.flush();
        break;
      }
  }
  else{
      // Checksum Failed, Flush Serial Buffer
      Serial.flush();
  }
}

/************************************************************************
******
**  Functions
```

```
*****************************************************************************
*****/

// Writes Values To Digital Port (DIO 0-13).  Pins Must Be Configured As
Outputs Before Being Written To
void writeDigitalPort(unsigned char command[])
{
  digitalWrite(13, (( command[2] >> 5) & 0x01) );
  digitalWrite(12, (( command[2] >> 4) & 0x01) );
  digitalWrite(11, (( command[2] >> 3) & 0x01) );
  digitalWrite(10, (( command[2] >> 2) & 0x01) );
  digitalWrite(9, (( command[2] >> 1) & 0x01) );
  digitalWrite(8, (command[2] & 0x01) );
  digitalWrite(7, (( command[3] >> 7) & 0x01) );
  digitalWrite(6, (( command[3] >> 6) & 0x01) );
  digitalWrite(5, (( command[3] >> 5) & 0x01) );
  digitalWrite(4, (( command[3] >> 4) & 0x01) );
  digitalWrite(3, (( command[3] >> 3) & 0x01) );
  digitalWrite(2, (( command[3] >> 2) & 0x01) );
  digitalWrite(1, (( command[3] >> 1) & 0x01) );
  digitalWrite(0, (command[3] & 0x01) );
}

// Reads all 6 analog input ports, builds 8 byte packet, send via RS232.
void analogReadPort()
{
  // Read Each Analog Pin
  int pin0 = analogRead(0);
  int pin1 = analogRead(1);
  int pin2 = analogRead(2);
  int pin3 = analogRead(3);
  int pin4 = analogRead(4);
  int pin5 = analogRead(5);

  //Build 8-Byte Packet From 60 Bits of Data Read
  char output0 = (pin0 & 0xFF);
  char output1 = ( ((pin1 << 2) & 0xFC) | ( (pin0 >> 8) & 0x03) );
  char output2 = ( ((pin2 << 4) & 0xF0) | ( (pin1 >> 6) & 0x0F) );
  char output3 = ( ((pin3 << 6) & 0xC0) | ( (pin2 >> 4) & 0x3F) );
  char output4 = ( (pin3 >> 2) & 0xFF);
  char output5 = (pin4 & 0xFF);
  char output6 = ( ((pin5 << 2) & 0xFC) | ( (pin4 >> 8) & 0x03) );
  char output7 = ( (pin5 >> 6) & 0x0F );

  // Write Bytes To Serial Port
  Serial.print(output0);
  Serial.print(output1);
  Serial.print(output2);
  Serial.print(output3);
  Serial.print(output4);
  Serial.print(output5);
  Serial.print(output6);
  Serial.print(output7);
}

// Configure digital I/O pins to use for seven segment display
void sevenSegment_Config(unsigned char command[])
```

```
{
  // Configure pins as outputs and store in sevenSegmentPins array for use
in sevenSegment_Write
  for(int i=2; i<10; i++)
  {
      pinMode(command[i], OUTPUT);
      sevenSegmentPins[(i-1)] = command[i];
  }
}

//  Write values to sevenSegment display.  Must first use
sevenSegment_Configure
void sevenSegment_Write(unsigned char command[])
{
  for(int i=1; i<9; i++)
  {
      digitalWrite(sevenSegmentPins[(i-1)], command[i]);
  }
}

// Set the SPI Clock Divisor
void spi_setClockDivider(unsigned char divider)
{
  switch(divider)
  {
  case 0:
      SPI.setClockDivider(SPI_CLOCK_DIV2);
      break;
  case 1:
      SPI.setClockDivider(SPI_CLOCK_DIV4);
      break;
  case 2:
      SPI.setClockDivider(SPI_CLOCK_DIV8);
      break;
  case 3:
      SPI.setClockDivider(SPI_CLOCK_DIV16);
      break;
  case 4:
      SPI.setClockDivider(SPI_CLOCK_DIV32);
      break;
  case 5:
      SPI.setClockDivider(SPI_CLOCK_DIV64);
      break;
  case 6:
      SPI.setClockDivider(SPI_CLOCK_DIV128);
      break;
  default:
      SPI.setClockDivider(SPI_CLOCK_DIV4);
      break;
  }
}

void spi_sendReceive(unsigned char command[])
{
  if(command[2] == 1)         //Check to see if this is the first of a series
of SPI packets
  {
```

```
      spiBytesSent = 0;
      spiCSPin = command[3];
      spiWordSize = command[4];

      // Send First Packet's 8 Data Bytes
      for(int i=0; i<command[5]; i++)
      {
        // If this is the start of a new word toggle CS LOW
        if( (spiBytesSent == 0) || (spiBytesSent % spiWordSize == 0) )
        {
            digitalWrite(spiCSPin, LOW);
        }
        // Send SPI Byte
        Serial.print(SPI.transfer(command[i+6]));
        spiBytesSent++;

        // If word is complete set CS High
        if(spiBytesSent % spiWordSize == 0)
        {
            digitalWrite(spiCSPin, HIGH);
        }
      }
  }
  else
  {
      // SPI Data Packet - Send SPI Bytes
      for(int i=0; i<command[3]; i++)
      {
        // If this is the start of a new word toggle CS LOW
        if( (spiBytesSent == 0) || (spiBytesSent % spiWordSize == 0) )
        {
            digitalWrite(spiCSPin, LOW);
        }
        // Send SPI Byte
        Serial.write(SPI.transfer(command[i+4]));
        spiBytesSent++;

        // If word is complete set CS High
        if(spiBytesSent % spiWordSize == 0)
        {
            digitalWrite(spiCSPin, HIGH);
        }
      }
  }
}

// Synchronizes with LabVIEW and sends info about the board and firmware
(Unimplemented)
void syncLV()
{
  Serial.begin(DEFAULTBAUDRATE);
  i2cReadTimeouts = 0;
  spiBytesSent = 0;
  spiBytesToSend = 0;
  Serial.flush();
}
```

```
// Compute Packet Checksum
unsigned char checksum_Compute(unsigned char command[])
{
  unsigned char checksum;
  for (int i=0; i<(COMMANDLENGTH-1); i++)
  {
      checksum += command[i];
  }
  return checksum;
}

// Compute Packet Checksum And Test Against Included Checksum
int checksum_Test(unsigned char command[])
{
  unsigned char checksum = checksum_Compute(command);
  if(checksum == command[COMMANDLENGTH-1])
  {
      return 0;
  }
  else
  {
      return 1;
  }
}

// Stepper Functions
#ifdef STEPPER_SUPPORT
  void AccelStepper_Write(unsigned char command[]){
      int steps = 0;
      int step_speed = 0;
      int acceleration = 0;

      //Number of steps & speed are a 16 bit values, split for data
transfer. Reassemble 2 bytes to an int 16
      steps = (int)(command[5] << 8) + command[6];
      step_speed = (int)(command[2] << 8) + command[3];
      acceleration = (int)(command[7] << 8) + command[8];

      steppers[command[4]].setMaxSpeed(step_speed);

      if (acceleration == 0){
        //Workaround AccelStepper bug that requires negative speed for
negative step direction
        if (steps < 0) step_speed = -step_speed;
        steppers[command[4]].setSpeed(step_speed);
        steppers[command[4]].move(steps);
      }
      else {
        steppers[command[4]].setAcceleration(acceleration);
        steppers[command[4]].move(steps);
      }
  }
#endif


void sampleContinously()
{
```

```
  for(int i=0; i<iterations; i++)
  {
        retVal = analogRead(contAcqPin);
        if(contAcqSpeed>1000) //delay Microseconds is only accurate for
values less that 16383
        {
          Serial.write( (retVal >> 2));
          delayMicroseconds(delayTime*1000000); //Delay for neccesary amount
of time to achieve desired sample rate
        }
        else
        {
             Serial.write( (retVal & 0xFF) );
             Serial.write( (retVal >> 8));
             delay(delayTime*1000);
        }
  }
}
void finiteAcquisition(int analogPin, float acquisitionSpeed, int
numberOfSamples)
{
  //want to exit this loop every 8ms
   acquisitionPeriod=1/acquisitionSpeed;

  for(int i=0; i<numberOfSamples; i++)
  {
        retVal = analogRead(analogPin);

        if(acquisitionSpeed>1000)
        {
          Serial.write( (retVal >> 2));
          delayMicroseconds(acquisitionPeriod*1000000);
        }
        else
        {
          Serial.write( (retVal & 0xFF) );
          Serial.write( (retVal >> 8));
          delay(acquisitionPeriod*1000);
        }
  }
}

void lcd_print(unsigned char command[])
{
  if(command[2] != 0)
  {
      // Base Specified By User
      int base = 0;
      switch(command[2])
      {
        case 0x01:  // BIN
            base = BIN;
            break;
        case 0x02:  // DEC
            base = DEC;
            break;
```

```
        case 0x03:  // OCT
            base = OCT;
            break;
        case 0x04:  // HEX
            base = HEX;
            break;
        default:
            break;
      }
      for(int i=0; i<command[3]; i++)
      {
        lcd.print(command[i+4], base);
      }
  }
  else
  {

      for(int i=0; i<command[3]; i++)
      {
        lcd.print((char)command[i+4]);

      }
  }
  Serial.write('0');
}
```

Anexo D.- 8° Coloquio de Investigación de la Maestría en Ciencias de la Computación

Universidad Autónoma del Estado de México

Centro Universitario UAEM Temascaltepec

Otorga la presente

CONSTANCIA

A: Ing. Roberto Uribe Flores

Por su participación en el

VIII Coloquio de investigación 2012-B de la MACSCO

Llevado a cabo el 13 de Diciembre del 2012, en el Centro Universitario UAEM Temascaltepec

U A E M
CENTRO UNIVERSITARIO
TEMASCALTEPEC
DIRECCIÓN

U A E M
CENTRO UNIVERSITARIO
TEMASCALTEPEC
SUBDIRECCIÓN ACADÉMICA

CU Temascaltepec

Dr. En Edu. Manuel Antonio Pérez Chávez
Encargado del Despacho del Centro Universitario UAEM Temascaltepec y su Extensión Tejupilco

Dr. En Edu. Daniel Cardoso Jiménez
Subdirector Académico

Anexo E.- 14° Seminario de Investigación en la Universidad Autónoma de Aguascalientes

14° SEMINARIO DE INVESTIGACIÓN

La Universidad Autónoma de Aguascalientes
a través de la Dirección General de Investigación y Posgrado
otorga la presente

Constancia

A: **Roberto Uribe Flores, José Luis Sánchez Ramírez, Cristina Juárez Landín**

Por su participación como Ponentes con el trabajo
Sintonización automática de velocidad y posición para servomotores utilizando control difuso y redes neuronales artificiales
en la Mesa de **Ingenierías y Tecnologías**

M. en Admón. Mario Andrade Cervantes
RECTOR

Dr. en C. Fernando Jaramillo Juárez
DIRETOR GENERAL DE INVESTIGACIÓN Y POSGRADO

universidad autónoma de aguascalientes

"Se Lumen Proferre"

aniversario uaa

Aguascalientes, Ags., mayo 2013

Anexo F.- 9° Coloquio de Investigación de la Maestría en Ciencias de la Computación

Universidad Autónoma del Estado de México
Centro Universitario UAEM Atlacomulco

S E O T O R G A L A P R E S E N T E

CONSTANCIA

A: ROBERTO URIBE FLORES

Por su ***PARTICIPACIÓN*** *en el "IX Coloquio de Investigación de la Maestría en Ciencias de la Computación 2013-A", llevado a cabo en el Centro Universitario UAEM Atlacomulco, el día 30 de mayo del 2013.*

"2013, 50 Aniversario Luctuoso del Poeta Heriberto Enríquez"
PATRIA, CIENCIA Y TRABAJO

CENTRO UNIVERSITARIO UAEM ATLACOMULCO
DIRECCIÓN

M. EN A. FIDENCIO OCHOA FLORES
Encargado del Despacho de la Dirección del
Centro Universitario UAEM Atlacomulco

Anexo G.- 9° Coloquio de Internacional de Cómputo e Informática

UAEM | Universidad Autónoma del Estado de México

CENTRO UNIVERSITARIO UAEM VALLE DE CHALCO

Centro Universitario UAEM Valle de Chalco

4° Coloquio Internacional de Cómputo e Informática

13 al 15 de noviembre de 2013

Otorga la presente

Constancia

Al Ing. Roberto Uribe, Dr. José Luis Sánchez Ramírez y a la Dra. Cristina Juárez Landín

Por su participación con la Ponencia "Sintonización Automática de Velocidad y Posición para Servomotores Utilizando Control Difuso" en el marco del 4to. Coloquio Internacional de Computación e Informática

Valle de Chalco, Estado de México a 13 de Noviembre de 2013.

Dr. René Guadalupe Cruz Flores	*Dra. Magally Martínez Reyes*	*M. en Ed. Anabelem Soberanes Martín*
Coordinador de Investigación	*Directora del Centro Universitario*	*Líder de C. A. Cómputo Aplicado*

Anexo H.- 10° Coloquio de Investigación de la Maestría en Ciencias de la Computación 2013-B

UAEM | Universidad Autónoma del Estado de México

CUTex
Centro Universitario UAEM Texcoco

Otorga a

Uribe Flores Roberto

el presente

Reconocimiento

Por disertar su tema de investigación en este Campus Universitario el 28 de noviembre de 2013, en el marco del X Coloquio de Investigación de la Maestría en Ciencias de la Computación 2013-B.

Patria, Ciencia y Trabajo

"2013, 50 Aniversario Luctuoso del Poeta Heriberto Enríquez"

Dr. en Ed. Carlos G. Vega Vargas
Encargado del Despacho de la Dirección
DIRECCIÓN

Anexo I.- 4° Foro Científico Interdisciplinario de la DES Oriente

UAEM | Universidad Autónoma del Estado de México

Otorga a

Roberto Uribe Flores

el presente

RECONOCIMIENTO

Como ponente disertando el tema intitulado

Sintonización automática de velocidad y posición para servomotores utilizando Control Difuso

llevada a cabo en este campus universitario
el 5 de noviembre de 2014
en el marco cultural del IV Foro Científico
Interdisciplinario de la DES oriente

"Patria, Ciencia y Trabajo"

Dr. en D. Ricardo Colín García
Director

CENTRO UNIVERSITARIO UAEM TEXCOCO
DIRECCIÓN

M. en C. Ed. Viridiana Banda Arzate
Subdirectora Académica

8 BIBLIOGRAFIA

Aceves, L. A. (2001). Usos y abusos de la lógica difusa. *Con mantenimiento productivo,* 12-17.

Alzate, A., & Giraldo, S. E. (2006). APLICACIÓN DE CONTROLES INTELIGENTES SOBRE SISTEMAS NO LINEALES Y. *Scientia et Technica*, 1-6.

Bolton, W. (2011). *Ingenieria de control.* México: Alfaomega.

C.L., K. (1991). Design of a adaptative fuzzy logic controller using a genetic algorithm. *Proceedigns of the fourth international conference, Belew, R. and Brooker, L.* San Mateo California: Morgan Kaufmann.

Coronel , L. M. (2004). Simulación de sistema difuso para el control de velocidad de un motor de c.d. *Memorias del tercer congreso de cómputo AGECOM*, 1-10.

Diaz , R. J., & Acevedo, G. T. (2008). Controlador Lógico Difuso Aplicado al Control de Motores De. *Revista Colombiana de tecnologías de avanzada*, 1-5.

Garcia, A. (2012). *Inteligencia Artificial.* México: Alfaomega.

García, J. L. (2003). Control Digital. *Capitulo 5 Identificación de Sistemas*.

Intech. (03 de 06 de 2009). *Computational Intelligence and Modern Heuristics*. Recuperado el 30 de 04 de 2013, de http://www.intechopen.com/books/computational-intelligence-and-modern-heuristics

Jimenez, B. L. (2000). *Algunas Aplicaciones de Lógica Difusa a Teoría de Control.* Tijuana B.C.: IPN.

Jing Yuan, Z., & Li, Y. D. (2006). Application of genetic algorithm in optimization of fuzzy control rules. *Proceedings of the sixth international conference on intelligent systems design and aplications.* ISDA06.

Kouro, S. R., & Musalem, R. M. (2002). Control Mediante Lógica Difusa. *Tecnicas Modernas en Automatica*, 1.

Ley, M., Chacón L., O., & Vázquez , E. (2000). *Control de Voltaje de sistemas de potencia utilizando lógica difusa.* México: Universidad Autonoma de Nuevo León .

Llorens, F. L. (06 de 2000). Lógica Multivaluadas o Polivalentes. *Universidad Politécnica Superior de Alicante España*. Obtenido de http://www.dccia.ua.es/~faraon/

López A., J., Muñoz A., P., & Cardona E., J. (2007). Control de un motor utilizando lógica difusa con reglas sintonizadas por algoritmos genéticos. *Scientia et Technica Año XIII, No 37 Universidad Tecnológica de Pereira. ISSN 0122-1701*, 121-125.

López, J. A., & Muñoz, P. A. (2007). CONTROL DE UN MOTOR UTILIZANDO LÓGICA DIFUSA CON REGLAS. *Scientia et Technica*, 1-5.

Lorandi, M. A. (2011). Controladores PID y controladores difusos. *Revista de la Ingeniería Industrial*. Obtenido de Revista de la IngenieriaIndustrial: http://academiajournals.com/downloads/Lorandi2011IE.pdf

Mackworth, A., & Poole, D. (2010). *Artificial Intelligence*. Recuperado el 05 de 02 de 2013, de http://artint.info/html/ArtInt.html

Madrigal, R. (2007). *Diseño de un controlador lógico difuso aplicado al control de posicón de un motor de cd.* Veracruz, ver.: Universidad Veracruzana.

Morales L., G. (17 de 2 de 2002). *Itroduccion a la Lógica Difusa*. Recuperado el 05 de 03 de 2013, de http://delta.cs.cinvestav.mx/~gmorales/ldifll/ldifll.html

Rezoug, S. B., & Hamerlain, F. (2009). Fuzzy Logic Control for Manipulator Robot. *Journal of electrical systems*, 1-6.
Ruiz, J. (Agosto de 2012). Labview + Arduino. España: Creative Commons Attribution 3.0 Unported License.
Sala, A., & Ariño, C. V. (2009). Reduciendo distancias entre el control borroso y el. *Revista Iberoamericana de Automática e Informatica Industrial*, 1-2.
San miguel, C. (julio de 2012). Diseño y desarrollo de un sistema de uso doméstico. Cantabria, España: Universidad de Cantabria.
Sánchez, C. O. (2006). *Curso de Utilización Práctica de Matlab.* Madrid: España.
Sanjay , K. A. (06 de 2014). *eMathTeacher: Método de Mamdani de Inferencia Borrosa.* Recuperado el 19 de 10 de 2014, de http://www.dma.fi.upm.es/research/fundmatsoftcomputing/fuzzyinf/mamdanir.htm
UNIVERSIDAD DE LAS AMERICAS-MEXICO . (s.f.). *powershow.com.* Recuperado el 19 de 10 de 2014, de http://www.powershow.com/view1/281cb9-ZDc1Z/FUNDAMENTOS_DE_LOGICA_DIFUSA_powerpoint_ppt_presentation
Vadiie, N., & Jamshidi, M. (1993). Timothy J. Fuzzy logic and control. Prentice Hall.
Wang, L. (1997). *A course in fuzzy systems and control. Prentice Hall*.
Wang, L.-X. (1994). *Adaptive Fuzzy System and Control.* Englewood Cliffs: Prentice Hall .
Wang, L.-X. (1997). A course in fuzzy systems and control. Prentice Hall.
Zadeh, L. (1965). *The Berkeley Initiative in Soft Computing*. Recuperado el 16 de 02 de 2013, de http://www.cs.berkeley.edu/~zadeh/
Zatarain V., O. (18 de 5 de 2012). *Lógica Difusa*. Recuperado el 12 de 02 de 2013, de http://www.uaeh.edu.mx/docencia/P_Presentaciones/icbi/asignatura/logicaDifusa.pdf