REPÚBLICA DE CUBA

MINISTERIO DE EDUCACIÓN SUPERIOR

INSTITUTO SUPERIOR MINERO METALÚRGICO DE MOA

"Dr. Antonio Núñez Jiménez"

Trabajo de Diploma

En opción al Título de Ingeniero Eléctrico

TÍTULO: INTERFACE PARA EL CONTROL INTELIGENTE DE INSTRUMENTOS EN SISTEMAS SCADA DE ACCIONAMIENTOS ELECTRICOS

Diplomante: Gerardo Luis Pupo Bacallao

Tutores: Dr. C. Luis Delfín Rojas Purón

Ing. Ernesto Coello Velázquez

Ing. Osmany Pérez Aballe

Curso, 2009-2010

"Año 52 de la Revolución"

INSTITUTO SUPERIOR MINERO METALÚRGICO DE MOA

"Dr. Antonio Núñez Jiménez"

FACULTAD DE METALURGIA ELECTROMECÁNICA

DEPARTAMENTO DE ELÉCTRICA

INTERFACE PARA EL CONTROL INTELIGENTE DE INSTRUMENTOS EN SISTEMAS SCADA DE ACCIONAMIENTOS INDUSTRIALES

Diplomante: Gerardo Luis Pupo Bacallao

Tutores: Dr. C. Luis Delfín Rojas Purón

Ing. Ernesto Coello Velásquez

Ing. Osmany Pérez Aballe

Declaración de autoridad

Yo: Gerardo Luis Pupo Bacallao.

Autor de este trabajo de Diploma tutorado por el Dr. C. Luis Delfín Rojas Purón, el Ing. Osmany Pérez Aballe y el Ing. Ernesto Coello Velázquez, certifico la propiedad intelectual a favor del Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez”, el cual podrá hacer uso del mismo para fines docentes y educativos.

---------- ----------

Gerardo Luis Pupo Bacallao *Dr. C. Luis Delfín Rojas Purón*

(Tutor) *(Tutor)*

---------- ----------

Ing. Osmany Pérez Aballe *Ing.Ernesto Coello Velázquez*

(Tutor) *(Tutor)*

Pensamiento

"Los conceptos y principios fundamentales de la ciencia son invenciones libres del espíritu humano"

Albert Einstein

Dedicatoria

Dedico este trabajo:

A mi Mamá, a mi Papá, a mis Hermanos Luis Gerardo y Sarah Lídia, a mis sobrinas Sahíly y Sayla.

A mis Familiares en general por haberme apoyado en cada momento de mi carrera y depositar toda la confianza, el amor y el cariño en Mí.

A Mi Tutores, por haberme ayudado dando el paso al frente para realizar este hermoso trabajo con su impresionante experiencia.

A mis amigos, que siempre confiaron en mí, y de los cuales recibí un apoyo infinito en los buenos y malos momentos.

Agradecimientos

Agradezco a la Revolución por darme la oportunidad de convertir un sueño hecho realidad, a Fidel y a Raúl Castro.

Agradezco a mi Tutores Dr. C. Luis Delfín Rojas Purón y Osmany Pérez Aballe por su incondicional ayuda en mi formación como profesional.

Agradezco a mis padres y hermanos, por su sacrificio e incondicionalidad.

A mis abuelos maternos y paternos, mis tíos, primos, y familia en general.

También debo mencionar personas que sin su ayuda no hubiese podido elaborar este trabajo, al profesor William Quesada Pupo, a Tony, Jose C. David, Dixan, Daydé, Anita, Julia, Angela, su familia, Eric, y a los que me apoyaron en todo momento.

Agradezco a Yoandris, Maikel L., Yanet, a todos mis vecinos, a mis amigos infinitamente y a los que de una forma u otra depositaron confianza en mí para hacer este trabajo.

A todos:

MUCHAS GRACIAS

Resumen

Debido a la ausencia de una interface que permita el control de instrumentos en los sistemas de monitoreo y supervisión de accionamientos eléctricos de velocidad regulables se hace este trabajo. Para esto se desarrolla un software orientado asistir una tarjeta de adquisición de datos con lectura y escritura de los puertos con los códigos ofrecidos desde el MATLAB O CITECT.

El trabajo incorpora circuitos de mando en accionamientos eléctricos, a partir del protocolo de comunicación de las estaciones de medición existente en el laboratorio de Circuitos Eléctricos y de Mediciones Eléctricas.

Esto constituye una aplicación de mando asistido por PC, en circuitos de control de accionamientos eléctricos, para el caso de estudio de un motor paso a paso (step-step).

Summary

Due to the absence of an interface that allows the control of instruments in the monitoring systems and supervision of adjustable electric workings of speed is made this work. For this an oriented software is developed to attend a data acquisition card with reading and it notarizes of the ports with the offered codes from the MATLAB OR CITECT.

The work incorporates control circuits in electric workings, starting from the protocol of communication of the stations of existent mensuration in the laboratory of Electric Circuits and of Electric Mensurations.

This constitutes a control application attended by PC, in circuits of control of electric workings, for the case of study of a motor step to step (step-step).

Índice

Declaración de autoridad II

Pensamiento III

Dedicatoria IV

Agradecimientos V

Resumen VI

Summary VI

Índice VII

Introducción General 1

Situación problémica 1

Hipótesis 1

Objetivo 2

Resultados esperados 2

CAPÍTULO I 2

1.1 Introducción 3

1.2 Fundamentos de adquisición de señales 4

1.2.1 Características básicas de un convertidor A/D 6

1.2.2 Errores en los convertidores analógico/digital. 8

1.2.3 Etapa de acondicionamiento de la señal 10

1.2.4 Muestreo de la señal 12

1.3 Entorno de trabajo MATLAB y herramientas para la adquisición de datos. 15

1.4 Interface del MATLAB para capturar y generar señales. 16

1.5 Elementos de software 17

1.6 Configuración de los elementos Hardware 18

1.7 Diseño de controladores 19

1.8 Conclusiones del capítulo I 24

CAPÍTULO II 1

2.1 Introducción 26

2.2 Descripción de GPIB 27

2.3 Hardware disponible para sistemas GPIB 30
2.4 Software existente para GPIB 39
2.5 Tarjeta de adquisición de datos DaqBoard/1000 IOtech 41
2.6 Conclusiones del capítulo II 51
CAPÍTULO III 52
3.1 Introducción 52
3.2 Conceptos y reglas básicas 56
3.3 Diseño del controlador difuso a través de la herramienta fuzzy de MATLAB 58
3.4 Implementación de una red neuronal básica 65
3.5 Conclusiones del capítulo III 71
CAPÍTULO IV 52
4.1 Introducción 67
4.2 Motores paso a paso (step- step). 68
4.3 Adquisición de datos por el puerto paralelo de la PC 69
4.4 Elaboración de la interface para el control de un motor paso a paso. 71
4.5 Propuesta de aplicación para un sistema de control inteligente utilizando un motor paso a paso. 79
4.6 Conclusiones del capítulo IV 80
CAPÍTULO V 67
5.1 Introducción 80
5.2 Costos en la explotación del accionamiento industrial 80
5.3 Gastos en inversiones 81
5.4 Impacto económico de la interface para la supervisión 81
5.5 Conclusiones del capítulo V 84
Conclusiones Generales 85
Recomendaciones 86
Bibliografía 86
Anexos 80

Introducción General

La supervisión es el hecho de controlar a distancia los procesos industriales, de forma remota y computarizada un usuario o una maquina controla los diferentes procesos que se dan en una fábrica y su principal función es la centralización del control de procesos fuera del área de control o fuera de máquina a controlar. En la misma actúan tanto las personas como las máquinas. Tan importante como la supervisión son las jerarquías de control que establecen una red industrial de procesos, existen en el mercado diferentes sistemas para el control, estos se denominan sistemas SCADA y su nomenclatura indica un acrónimo por Supervisory Control And Data Acquisition (control y adquisición de datos de supervisión). Los sistemas SCADA utilizan el computador y las tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales.
Con ésta línea de investigación se pretende elaborar una interface que permita el control de instrumentos en los sistemas de monitoreo y supervisión de accionamientos eléctricos a partir de sistemas SCADA desde MATLAB.

Situación problémica

La ausencia de una interface para el control de instrumentos en los sistemas de monitoreo y la supervisión de accionamientos eléctricos, donde se necesita el mando de sus variables, basado en el control desde PC.

Hipótesis

Con la identificación de los protocolos de comunicación de las estaciones de medición disponibles en los laboratorios de Circuitos Eléctricos y Mediciones Eléctricas, del Dpto. de Eléctrica del ISMM de Moa, para los accionamientos eléctricos, es posible elaborar una interface de comunicación que permita la supervisión y control de estos sistemas.

Objetivo

Elaborar una interface para el control de un sistema de medición destinado al monitoreo y supervisión de accionamientos eléctricos a partir de sistemas SCADA desde MATLAB.

Resultados esperados

- Construir una interface para el control de instrumentos en tareas de monitoreo y supervisión de los accionamientos eléctricos a partir de sus protocolos de comunicación.
- Brindar un software que asista la interface para el control de instrumentos en un sistema para la supervisión de accionamientos.
- Desarrollar un caso de estudio para la validación de la tarjeta a través de un esquema de control de instrumentos de medición desde MATLAB.

CAPÍTULO I

FUNDAMENTOS SOBRE LA ADQUISICIÓN DE DATOS DESDE MATLAB

1.1 Introducción

Las señales son generadas o producidas de forma natural o inducida, de esta forma se pueden aplicar en el área de actividad del ser humano, por medio de la debida manipulación de esta, siendo así impulsora del desarrollo de la tecnología, civilización, cultura y economía, la cual nos permite transformar, controlar y adecuar el medio en que habitamos, dando solución a sus problemas, necesidades o aspiraciones individuales y colectivas, mediante la construcción de sistemas y procesos técnicos en donde se emplean los recursos inmersos en la sociedad que vivimos.

Los avances tecnológicos se pueden considerar como "la aplicación sistemática del conocimiento científico y organizado a las tareas prácticas", a la resolución de problemas específicos. La esencia de la tecnología radica en la utilización de teorías, métodos científicos y su adaptación para conseguir determinados fines, utilizando las fuentes de la experiencia e inspiración de investigación, para dar así aportaciones prácticas y especificas para las diversas áreas de conocimiento y desarrollo.

En este primer capítulo, se hace una fundamentación teórica sobre la adquisición de señales, el entorno de trabajo en MATLAB, y sus herramientas para la adquisición de datos. También se hace referencia a los elementos tanto de software como de hardware, y su respectiva configuración.

1.2 Fundamentos de adquisición de señales

Un sistema de adquisición de datos es un equipo que nos permite tomar señales físicas del entorno y convertirlas en datos que posteriormente podremos procesar y presentar. A veces el sistema de adquisición es parte de un sistema de control, y por tanto la información recibida se procesa para obtener una serie de señales de control.

Estructura de un sistema de adquisición de datos: En la figura 1.1 podemos ver los bloques que componen nuestro sistema de adquisición de datos:

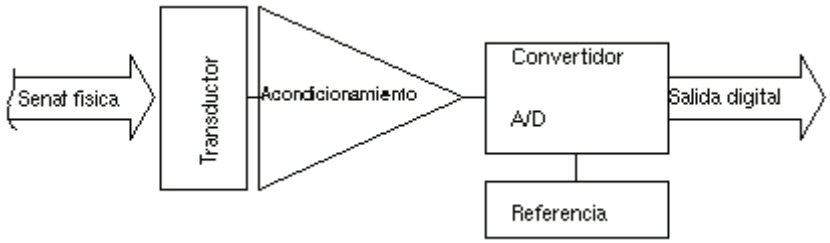


Figura 1.1. Esquema de bloques de un sistema de adquisición de datos.

Como vemos, los bloques principales son estos:

- El transductor
- El acondicionamiento de señal
- El convertidor analógico-digital
- La etapa de salida (interfaz con la lógica)

El transductor es un elemento que convierte la magnitud física que vamos a medir en una señal de salida (normalmente tensión o corriente) que puede ser procesada por nuestro sistema. Salvo que la señal de entrada sea eléctrica, podemos decir que el transductor es un elemento que convierte energía de un tipo en otro. Por tanto, el transductor debe tomar poca energía del sistema bajo observación, para no alterar la medida.

El acondicionamiento de señal es la etapa encargada de filtrar y adaptar la señal proveniente del transductor a la entrada del convertidor analógico / digital. Esta adaptación suele ser doble y se encarga de:

- Adaptar el rango de salida del transductor al rango de entrada del convertidor. (Normalmente en tensión).
- Acoplar la impedancia de salida de uno con la impedancia de entrada del otro.

La adaptación entre los rangos de salida del convertidor y el de entrada del convertidor tiene como objetivo el aprovechar el margen dinámico del convertidor, de modo que la máxima señal de entrada debe coincidir con la máxima que el convertidor (pero no con la máxima tensión admisible, ya que para ésta entran en funcionamiento las redes de protección que el convertidor lleva integrada).

Por otro lado, la adaptación de impedancias es imprescindible ya que los transductores presentan una salida de alta impedancia, que normalmente no puede excitar la entrada de un convertidor, cuya impedancia típica suele estar entre 1 y 10 k.

El convertidor Analógico / Digital es un sistema que presenta en su salida una señal digital a partir de una señal analógica de entrada, (normalmente de tensión) realizando las funciones de **cuantificación** y **codificación.**

La cuantificación implica la división del rango continuo de entrada en una serie de pasos, de modo que para infinitos valores de la entrada la salida sólo puede presentar una serie determinada de valores. Por tanto la cuantificación implica una pérdida de información que no podemos olvidar.

La codificación es el paso por el cual la señal digital se ofrece según un determinado código binario, de modo que las etapas posteriores al convertidor puedan leer estos datos adecuadamente. Este paso hay que tenerlo siempre en cuenta, ya que puede hacer que obtengamos datos erróneos, sobre todo cuando el sistema admite señales

positivas y negativas con respecto a masa, momento en el cual la salida binaria del convertidor nos da tanto la magnitud como el signo de la tensión que ha sido medida.

La etapa de salida es el conjunto de elementos que permiten conectar el s.a.d con el resto del equipo, y puede ser desde una serie de buffers digitales incluidos en el circuito convertidor, hasta un interfaz RS 232, RS 485 o Ethernet para conectar a un ordenador o estación de trabajo, en el caso de sistemas de adquisición de datos comerciales.

1.2.1 Características básicas de un convertidor A/D

A continuación describiremos las características esenciales que hemos de tener en cuenta para realizar nuestras medidas de un modo decente. No mencionaremos todas, sino las más básicas, dejando un estudio en profundidad de los convertidores para otro documento. Las características que no debemos olvidar son éstas:

- Impedancia de entrada
- Rango de entrada
- Número de bits
- Resolución
- Tensión de fondo de escala
- Tiempo de conversión
- Error de conversión

Hay una serie de características que son comunes a otros tipos de circuitos que no detallaremos, aunque siempre hay que tener en cuenta, como la impedancia de entrada, fan-out, etc. **Número de bits:** Es el número de bits que tiene la palabra de salida del convertidor, y por tanto es el número de pasos que admite el convertidor. Así un convertidor de 8 bits sólo podrá dar a la salida 2^8=256 valores posibles **Resolución:** Es el mínimo valor que puede distinguir el convertidor en su entrada analógica, o dicho de otro modo, la mínima variación, Vi, en el voltaje de entrada que se necesita para cambiar en un bit la salida digital. En resumen, tenemos que:

$$Vi = \frac{Vfe}{(2^n - 1)}$$

donde n es el número de bits del convertidor, y Vfe la tensión de fondo de escala, es decir, aquella para la que la salida digital es máxima. La tensión de fondo de escala depende del tipo de convertidor, pero normalmente se fija a nuestro gusto, en forma de una tensión de referencia externa, (aunque en algunos casos, como el del convertidor ADC 0804 la tensión de fondo de escala es el doble de la tensión de referencia).Por ejemplo, un convertidor de 8 bits con una tensión de fondo de escala de 2V tendrá una resolución de:

$$\frac{2V}{2^8 - 1} = 7.84 \frac{mV}{paso}$$

En cambio, para el mismo convertidor, si cambiamos la tensión de referencia, y por tanto la de fondo de escala, la resolución será de:

$$\frac{5V}{2^8 - 1} = 19.6 \frac{mV}{paso}$$

Tiempo de conversión:

Es el tiempo que tarda en realizar una medida el convertidor en concreto, y dependerá de la tecnología de medida empleada. Evidentemente nos da una cota máxima de la frecuencia de la señal a medir.

Este tiempo se mide como el transcurrido desde que el convertidor recibe una señal de inicio de conversión (normalmente llamada SOC, Start of Conversión) hasta que en la salida aparece un dato válido. Para que tengamos constancia de un dato válido tenemos dos caminos:

- Esperar el tiempo de conversión máximo que aparece en la hoja de características.

- Esperar a que el convertidor nos envíe una señal de fin de conversión.

Si no respetamos el tiempo de conversión, en la salida tendremos un valor, que dependiendo de la constitución del convertidor será:

- Un valor aleatorio, como consecuencia de la conversión en curso
- El resultado de la última conversión

1.2.2 Errores en los convertidores analógico/digital.

Un convertidor no es un circuito perfecto, sino que presenta una serie de errores que debemos tener en cuenta. Algunos de los que más importancia tiene son los siguientes:

Error de offset: El error de offset es la diferencia entre el punto nominal de offset (cero) y el punto real de offset. Concretamente, para un convertidor A/D este punto es el punto central de todos aquellos valores de la entrada que nos proporcionan un cero en la salida digital del convertidor. Este error afecta a todos los códigos de salida por igual, y puede ser compensado por un proceso de ajuste, como se muestra en la figura 1.2.

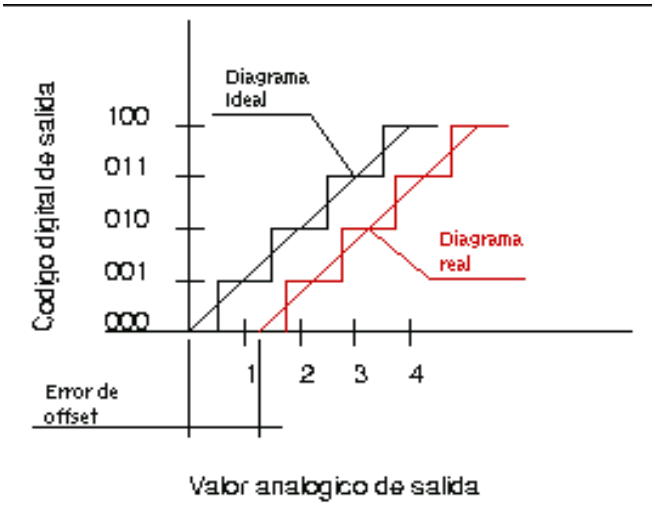


Figura 1.2. Esquema del error de offset

Error de cuantificación:

Es el error debido a la división en escalones de la señal de entrada, de modo que para una serie de valores de entrada, la salida digital será siempre la misma. Este valor se corresponde con el escalonado de la función de transferencia real, frente a la ideal. Podemos verlo en esta figura: como vemos, cada valor digital tiene un error de cuantificación de +- ½ LSB (Bit menos significativo). Por tanto, cada código digital representa un valor que puede estar dentro del ½ LSB a partir del punto medio entre valores digitales continuos.

Error de linealidad (linealidad integral):

Este error es la manifestación de la desviación entre la curva de salida teórica y la real, de modo que para iguales incrementos en la entrada, la salida indica distintos incrementos.

Error de apertura:

Es el error debido a la variación de la señal de entrada mientras se está realizando la conversión. Este error es uno de los más importantes cuando se están muestreando señales alternas de una frecuencia algo elevada, (como por ejemplo el muestreo de voz) pero tiene poca importancia cuando medimos señales cuasi-continuas, como temperatura, presión, o nivel de líquidos. Para minimizar este tipo de error se usan los circuitos de muestreo y retención.

Este error es importante, ya que si no lo tenemos en cuenta raramente podemos digitalizar adecuadamente señales alternas.

Si consideramos un error que no afecte a la precisión total de la conversión, (por lo que habrá de ser menor que ½ LSB) la frecuencia máxima de muestreo deberá ser:

$$Fmax = \frac{1}{Ta * \pi * 2^{n+1}}$$

En esta fórmula Ta es el tiempo de apertura del circuito de muestreo y retención, o bien el tiempo total de conversión si el anterior no existe, y n el nº de bits del convertidor.

El circuito de muestreo y retención puede estar a veces integrado dentro de la misma cápsula del convertidor, lo que nos puede simplificar el diseño enormemente.

1.2.3 Etapa de acondicionamiento de la señal

Con más detalle, en una etapa de acondicionamiento podemos encontrar estas etapas, aunque no todas están siempre presentes:

- Amplificación
- Excitación
- Filtrado
- Multiplexado
- Aislamiento
- Linealización

Amplificación Es el tipo más común de acondicionamiento. Para conseguir la mayor precisión posible la señal de entrada deber ser amplificada de modo que su máximo nivel coincida con la máxima tensión que el convertidor pueda leer.

Aislamiento - Otra aplicación habitual en el acondicionamiento de la señal es el aislamiento eléctrico entre el transductor y el ordenador, para proteger al mismo de transitorios de alta tensión que puedan dañarlo. Un motivo adicional para usar aislamiento es el garantizar que las lecturas del convertidor no son afectadas por diferencias en el potencial de masa o por tensiones en modo común.

Cuando el sistema de adquisición y la señal a medir están ambas referidas a masa pueden aparecer problemas si hay una diferencia de potencial entre ambas masas, apareciendo un "bucle de masa", que puede devolver resultados erróneos.

Multiplexado - El multiplexado es la conmutación de las entradas del convertidor, de modo que con un sólo convertidor podemos medir los datos de diferentes canales de entrada. Puesto que el mismo convertidor está midiendo diferentes canales, su frecuencia máxima de conversión será la original dividida por el número de canales muestreados.

Filtrado - El fin del filtro es eliminar las señales no deseadas de la señal que estamos observando. Por ejemplo, en las señales cuasi-continuas, (como la temperatura) se usa un filtro de ruido de unos 4 Hz, que eliminará interferencias, incluidos los 50/60 Hz de la red eléctrica.

Las señales alternas, tales como la vibración, necesitan un tipo distinto de filtro, conocido como filtro antialiasing, que es un filtro pasabajo pero con un corte muy brusco, que elimina totalmente las señales de mayor frecuencia que la máxima a medir, ya que se si no se eliminasen aparecerían superpuestas a la señal medida, con el consiguiente error.

Excitación - La etapa de acondicionamiento de señal a veces genera excitación para algunos transductores, como por ejemplos las galgas extesométricas, termistores o RTD, que necesitan de la misma, bien por su constitución interna, (como el termistor, que es una resistencia variable con la temperatura) o bien por la configuración en que se conectan (como el caso de las galgas, que se suelen montar en un puente de Wheatstone).

Linealización: Muchos transductores, como los termopares, presentan una respuesta no lineal ante cambios lineales en los parámetros que están siendo medidos. Aunque la linealización puede realizarse mediante métodos numéricos en el sistema de adquisición de datos, suele ser una buena idea el hacer esta corrección mediante circuitería externa.

1.2.4 Muestreo de la señal

El muestreo de la señal implica pérdida de información respecto a la señal de entrada, ya que de un número infinito de valores posibles para la entrada sólo tenemos un valor finito de valores posibles para la salida. Por tanto es fundamental saber cuántas muestras hemos de tomar.

La respuesta a esta pregunta depende del error medio admisible, el método de reconstrucción de la señal (si es que se usa) y el uso final de los datos de la conversión.

Independientemente del uso final, el error total de las muestras será igual al error total del sistema de adquisición y conversión más los errores añadidos por el ordenador o cualquier sistema digital.

Para dispositivos incrementales, tales como motores paso a paso y conmutadores, el error medio de los datos muestreados no es tan importante como para los dispositivos que requieren señales de control continuas.

Para ver el error medio de muestreo en los datos, consideremos el caso en el que se toman dos muestras por ciclo de señal sinusoidal, y la señal se reconstruye directamente desde un convertidor D/A sin filtrar (reconstrucción de orden cero). El error medio entre la señal reconstruida y la original es la mitad de la diferencia de áreas para medio ciclo, que es un 32% para una reconstrucción de orden cero, o del 14 % para una reconstrucción de orden uno.

De cualquier modo, la precisión instantánea en cada muestra es igual a la precisión del sistema de adquisición y conversión, y en muchas aplicaciones esto puede ser más que suficiente.

La precisión media de los datos muestreados puede mejorarse con estos métodos:

- Aumentar el número de muestras por ciclo
- Filtrado previo al multiplexado
- Filtrar la salida del convertidor digital / analógico

La mejora en la precisión media es espectacular con un pequeño aumento en el número de muestras por ciclo, como podemos ver en la figura 1.3

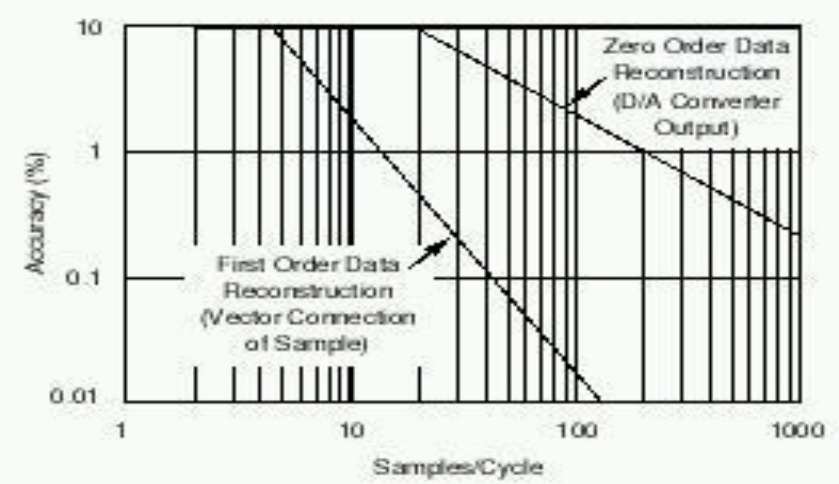


Figura 1.3. Mejora de la precisión media.

Para una reconstrucción de orden cero, podemos ver que con más de 10 muestras por ciclo de señal, podemos conseguir precisiones del 90 % o mejor. Normalmente se usan entre 7 y 10 muestras por ciclo.

El teorema de Nyquist o teorema de muestreo

El objetivo fundamental de la adquisición es el poder reconstruir la señal muestreada de una manera fiel. Este teorema nos dice que la frecuencia **mínima** de muestreo para poder reconstruir la señal ha de ser el doble de la frecuencia de la señal a medir. Pero ojo, para que la reconstrucción sea fiable, deberemos tomar muestras a una frecuencia unas 10 veces superior a la de la señal a evaluar.

En la figura 1.4 podemos ver una señal sinusoidal, que es muestreada con dos medidas por ciclo y su reconstrucción mediante los dos métodos que más se usan (reconstrucción de orden cero y reconstrucción de orden uno).

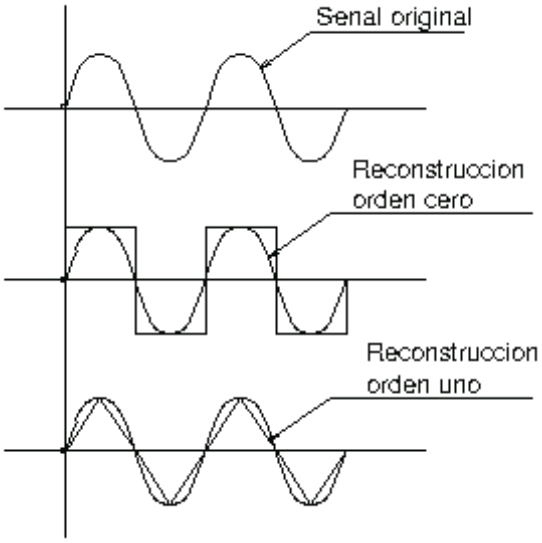


Figura 1.4. Muestreo de señales y su reconstrucción por métodos orden cero y orden uno.

Como se ve, aplicando el teorema de Nyquist podemos saber al menos la frecuencia de la señal medida, aunque no su tipo, ni si el muestreo es eficaz o no.

Por último comentar que la reconstrucción de orden cero es la salida directa de un convertidor analógico digital, mientras que la de orden uno es la interpolación simple mediante rectas, de modo que la señal se aproxima más a la original.

Efectos de Aliasing:

El aliasing se produce cuando la frecuencia de muestreo es menor que la de la señal que se muestrea, y se refiere al hecho de que podemos interpretar de una manera no exacta la señal, apareciendo un "alias" de la señal (de ahí el término). Este efecto se pone de manifiesto en la siguiente figura:

Figura 1.5. Esquema del efecto Alias

Como se aprecia, al tomar varias muestras con un periodo de muestreo superior al de la señal medida, llegamos a creer que la señal tiene una frecuencia mucho menor de la que realmente tiene. En este efecto también influyen las señales armónicas que interfieran con la señal a medir, de modo que pueden aparecer señales de alta frecuencia superpuestas, como ruido, y otras senoidales, que aparentemente no son ruido, pero que también afectan a la señal bajo medida. Por tanto, cualquier frecuencia de muestreo excesivamente baja nos da información falsa sobre la señal.

1.3 Entorno de trabajo MATLAB y herramientas para la adquisición de datos.

El nombre de MATLAB proviene de la contracción de los términos **MAT**rix **LAB**oratory (*Laboratorio Matricial*) y fue concebido para el fácil acceso a las librerías que son de gran importancia en el campo de la computación y el cálculo matricial.
MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para el desarrollo de proyectos con elevados cálculos matemáticos y la visualización gráfica de estos. MATLAB integra análisis numérico, cálculo matricial, procesado de señal, todo ello en un entorno fácil para el usuario.

MATLAB presenta una aplicación para hacer simulaciones en tiempo real, la *toolbox* **R**eal **T**ime **W**indows **T**arget. Esta herramienta permite realizar aplicaciones de control y simulaciones en tiempo real para plantas físicas, como puede ser el caso que nos ocupa: un motor de corriente continua. (Cilento,2007)

1.4 Interface del MATLAB para capturar y generar señales.

Real Time Windows Target es una herramienta de MATLAB que permite capturar y generar señales en tiempo real mediante diagramas de bloques generados con Simulink. Además, se pueden visualizar estas señales, cambiando y controlando parámetros, todo en tiempo real. Para hacerlo posible tiene que haber un elemento físico que interactúe entre simulink y el elemento exterior que queremos controlar, recoger señales,… este elemento es la placa adquisición de datos DAQ, que es la que permite operar con señales de entrada y/o salidas analógicas y digitales.

Un componente clave del Real Time Windows Target es el Kernel en tiempo real que hace de interfaz con el sistema operativo Windows para asegurar que la aplicación en tiempo real se está ejecutando en el tiempo de muestreo seleccionado. Las características más distintivas del sistema son:

- El kernel gestiona las interrupciones de tiempo
- Las interrupciones se utilizan como base de tiempo para crear un sencillo planificador.
- Hace una interfaz, gestionando las entadas y salidas de tiempo de adquisición de datos.
- Comunicación del kernel a través de Simulink, en modo externo.
- Necesidad de un compilador C

Una aplicación en tiempo real tiene las siguientes características:

- Código complicado: el resultado de complicar el modelo utilizado y la aplicación en tiempo real.
- Relación con el modelo de Simulink: la aplicación estará relacionada a través de las interconexiones entre bloques, dependencias de tiempo, etc.
- Relación con el kernel: Si no está instalado no podremos ejecutar el código.

- Checksum: El kernel utiliza este valor para comparar el modelo y el ejecutable, si estos son coherentes permitirá realizar la ejecución.

Elementos físicos de la aplicación en Tiempo Real

El conjunto del hardware que forma la aplicación en Tiempo Real se puede desglosar en seis bloques de elementos que interactúan entre sí.

- PC
- Entradas/Salidas, o placa de adquisición
- Driver del motor de corriente continua
- Motor de corriente continua
- Alimentación externa
- Realimentación

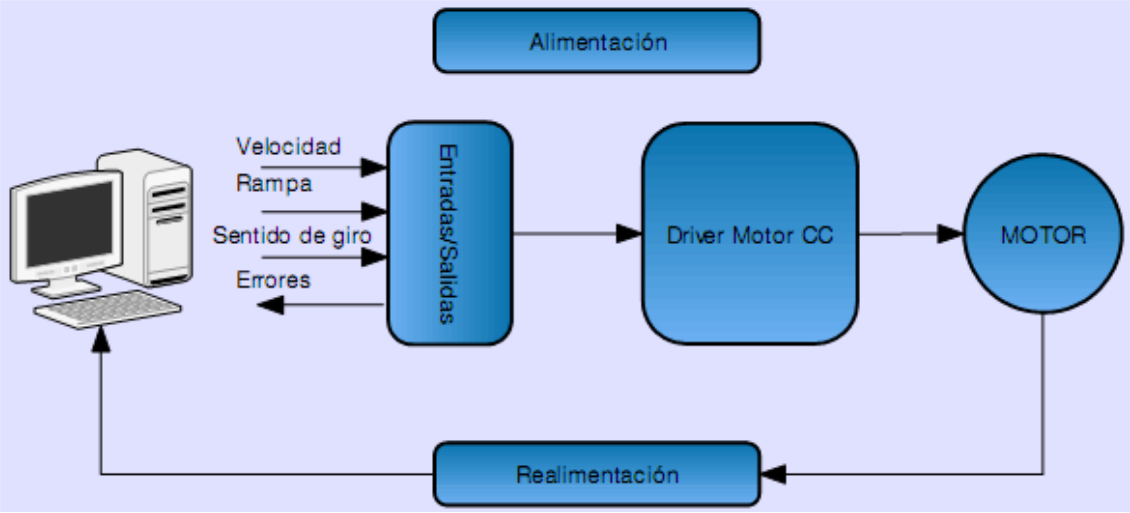


Figura 1.6. Esquema en bloques de los componentes del controlador

1.5 Elementos de software

Para el desarrollo de la aplicación en tiempo real, se trabaja con una herramienta de MATLAB llamada GUIDE (Graphical Use Interface Development Environment). Esta herramienta está pensada para desarrollar GUIs (Graphical User Interfaces) fácil y

rápidamente haciendo sencillo el diseño y presentación de los controles de la interfaz reduciendo la labor en el momento de seleccionar, deshacer, arrastrar y central controles, así como la personalización de las propiedades de estos.

El proceso a seguir para el desarrollo de un programa mediante GUIDE es que una vez se tienen todos los controles en posición, se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. GUIDE está diseñado APRA hacer menos tedioso el proceso de desarrollo de la interfaz gráfica, para ello cuenta con un editor de propiedades (property editor) con el que se podrá modificar en cualquier momento los nombres, valores por defecto y las propiedades de los elementos.

Cuando se crea una GUI se generan dos ficheros:

- Un archivo .FIG, que es el que contiene los elementos gráficos así como las propiedades de la interfaz.
- Un archivo .M que es el que contiene el código con las correspondencias de los botones de control de la interfaz. Cada vez que se introduzca un elemento gráfico en el .FIG se generará unas líneas de programa automáticamente asociadas a ese tipo de control. Estas líneas de programas son vacías, es decir, es como un contenedor que necesita ser llenado para llevar a cabo alguna acción durante la ejecución del programa. Este modo de trabajo es como el de LABWINDOWS, VISUAL BASIC, etc.

1.6 Configuración de los elementos Hardware

Para poder hacer un control de velocidad sobre un sistema real, el caso que nos ocupa es un motor de corriente continua, se tendrá que:

- Establecer un sistema que permita implantar y lazo de control.
- Modelar la planta del sistema real para poder diseñar el control.

El sistema que permitirá llevar a cabo el control de velocidad es el siguiente:

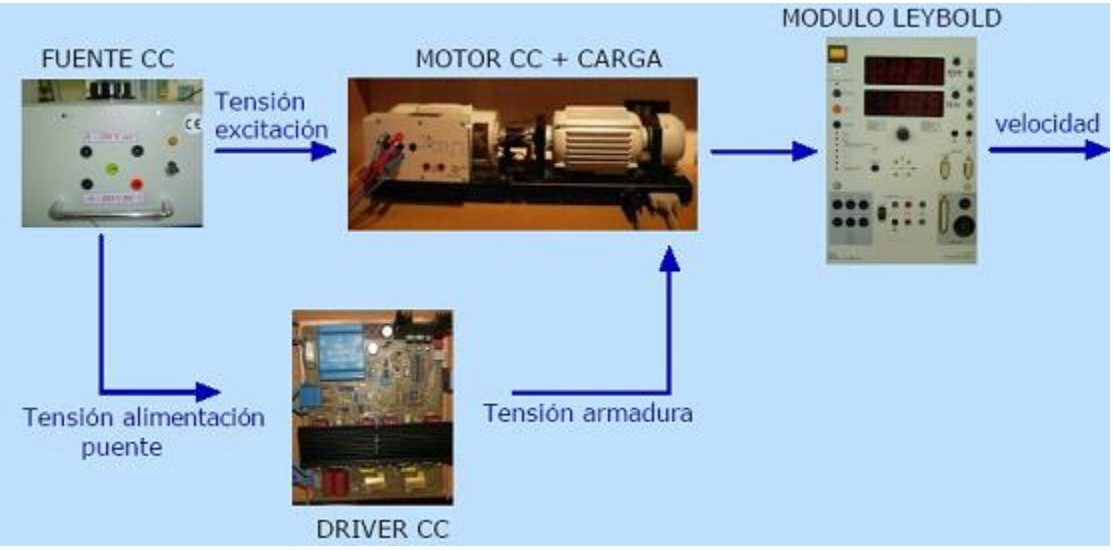


Figura 1.7. Esquema del diagrama de bloques del sistema

Para el modelado de la planta se deberá estudiar la respuesta del sistema en lazo abierto a una entrada escalón, y por el tipo de respuesta aproximarla a una función de primer orden.

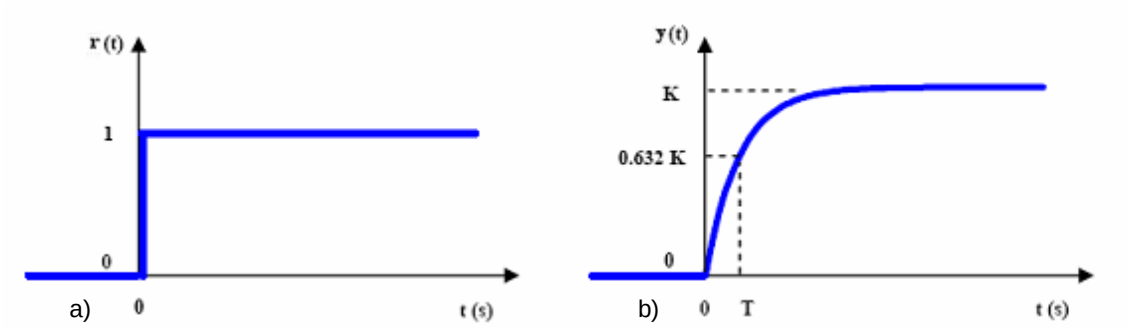


Figura 1.8. a) y b) Respuesta de un sistema de primer orden a una entrada de tipo escalón unitaria.

Una vez obtenida la planta en continuo, se selecciona un tiempo de muestreo y se discretiza por el método de la transformada Z.

1.7 Diseño de controladores

Con la planta discretizada se busca un control que pueda satisfacer nuestras exigencias de diseño. Para ello se han dimensionado dos tipos de controles: el P y el PI.

Controlador P

La estructura de un controlador P introduce una ganancia a la entrada de la planta. La estructura es la que se muestra en la figura 1.9.

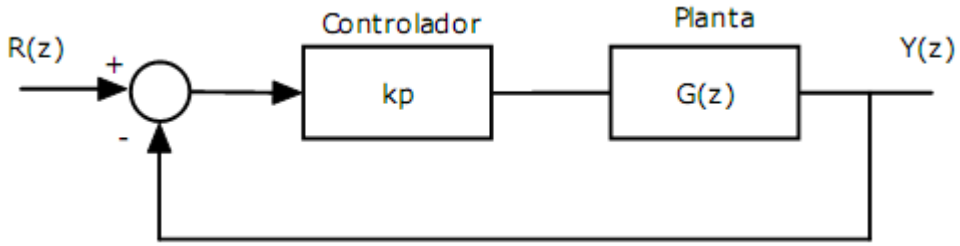


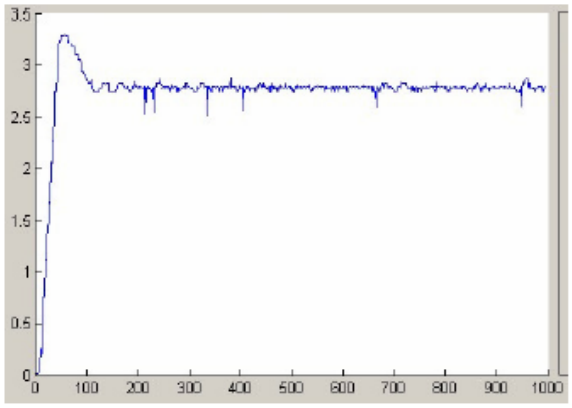
Figura 1.9. Diagrama de bloques del controlador P con la planta del sistema.

Los valores calculados para las dos frecuencias y que aseguran la estabilidad se reflejan en la siguiente tabla:

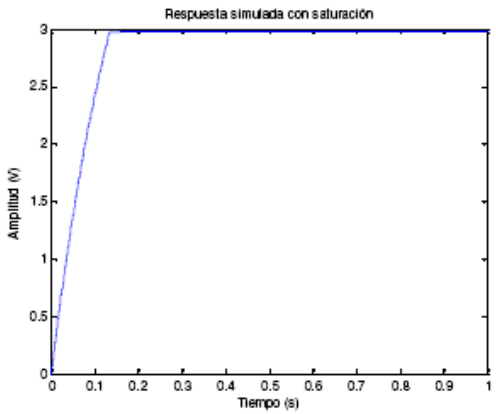
Tiempo de muestreo (s)	Valor constante proporcional kp
0,01	15
0,01	93,4

Tabla 1.1. Valores kp en función del tiempo de muestreo

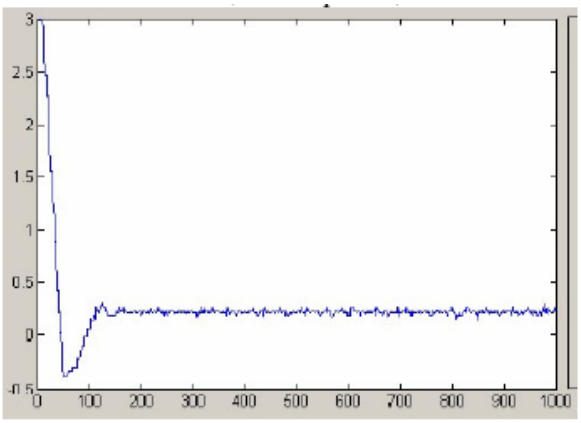
Para el tiempo de muestreo correspondiente a 0,001 segundos y con una consigna de 3

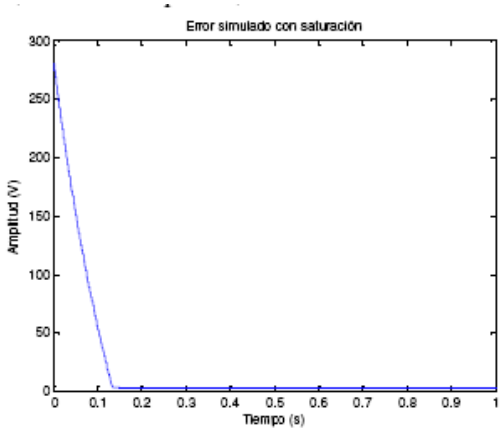


(a) Representación tensión de salida en función del número de muestras. Respuesta real controlador a Ts=0,001s kp=93,4



(b) Representación tensión de salida en función del tiempo. Respuesta simulación controlador a Ts=0,001s con kp=93,4 con saturación





(c) Representación tensión de salida en función del número de muestras. Error real Controlador a Ts=0,001s

(d) Modelo con controlador proporcional y saturación a Ts=0,001s con saturación

Figura 1.10.a), b), c), d) Comparación entre el controlador P real y el simulado

Para poder eliminar el sobre pico habrá que pensar en otro tipo de control, que añada una parte integral.

Controlador PI

El controlador PI, al contener la parte integral, a diferencia del anterior incrementa el tiempo de establecimiento y elimina el error en estado estacionario. Con la acción combinada de la parte integral y proporcional habrá que reducir el sobrepico y el tiempo de establecimiento.

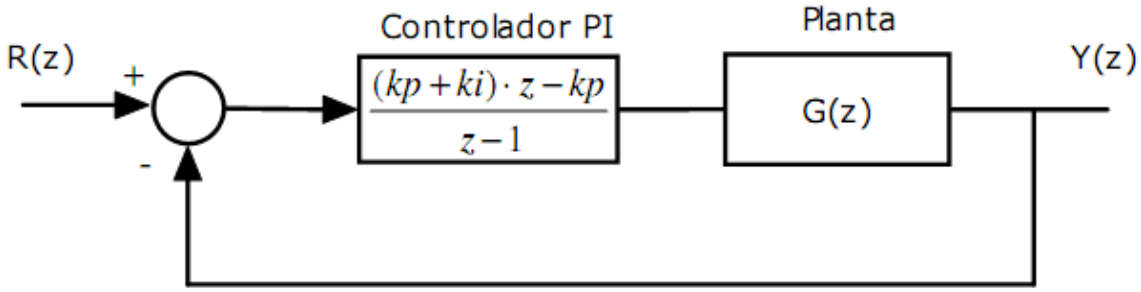
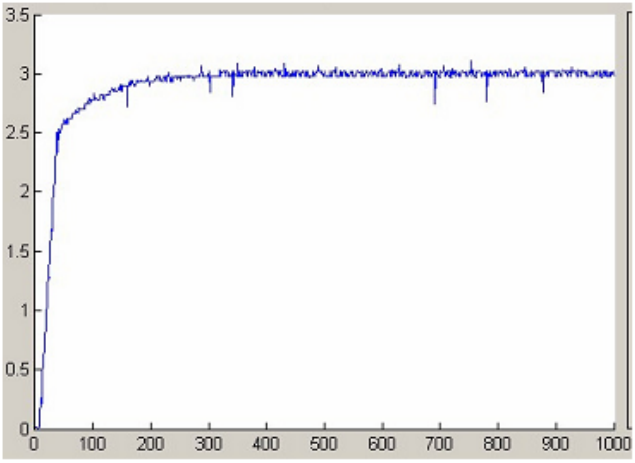
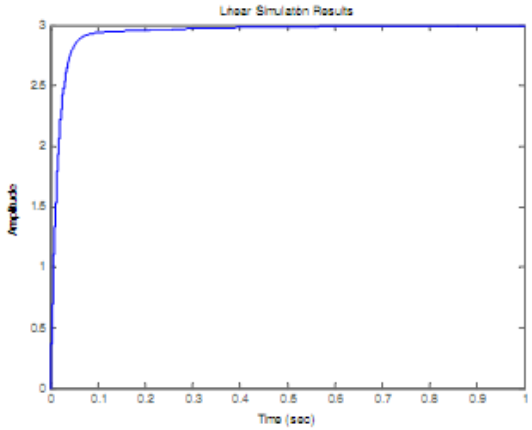


Figura 1.11. Diagrama de bloques del controlador PI con la planta del sistema.

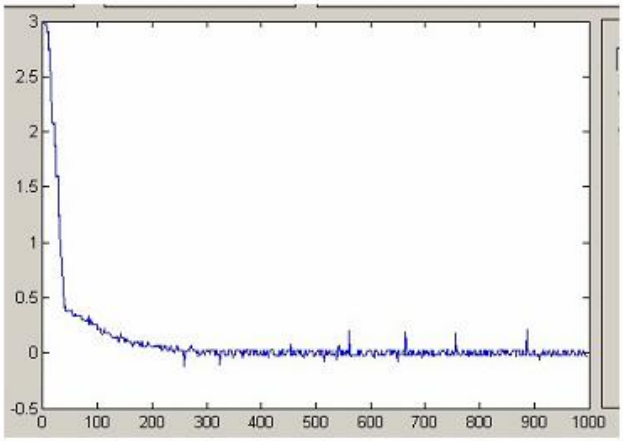
Las respuestas obtenidas en la simulación y en la aplicación Real Time para el tiempo de muestreo de 0,001 segundos e introduciendo en una consigna escalón de 3 se representan en la siguiente ilustración.



(a) Representación tensión de salida en función del número de muestras. Respuesta real controlador a Ts=0,001s



(b) Representación tensión de salida en función del tiempo. Respuesta simulación controlador a Ts=0,001s



(c) Representación tensión de salida en función del número de muestras. Error real Controlador a Ts=0,001s

Figura 1.12.a), b), c) Comparación entre controlador PI real y simulado

Se puede comprobar en las figuras a) y b), que el diseño se ha realizado correctamente, aunque en la ejecución real se alcanza el valor final unas décimas de segundo antes que en la simulada. Respecto al diseño del controlador P, se ha conseguido reducir el

error y el sobre pico. En la c se puede apreciar como el error en la entrada del controlador decrece hasta alcanzar un valor próximo a cero.

Las funciones de transferencia de los controladores se reflejan en la siguiente tabla:

Tiempo de muestreo (s)	Función de Transferencia Discreta (Z)
0,01	$D(z) = \frac{z - 0{,}996}{z - 1}$ (1)
0,001	$D(z) = \frac{z - 0{,}98}{z - 1}$ (2)

Tabla 1.2. Funciones de transferencia en función del tiempo de muestreo

1.8 Conclusiones del capítulo I

- Con el uso de las herramientas del MATLAB: Real Time Windows Target es posible el desarrollo de un sistema SCADA y el control de instrumentos.
- La utilización de un motor paso a paso, permite la implementación de un esquema de control de posición para cuatro señales de control: velocidad, rampa, sentido de giro y detección de errores, tal como se ilustra en la figura 1.6.

CAPÍTULO II

CONTROL DE INSTRUMENTOS EN SISTEMAS SCADA

2.1 Introducción

GPIB es un bus y un protocolo estándar para el control y comunicación con instrumentos de medida, como polímetros digitales, osciloscopios, etc., que permite configurar sistemas automáticos en el laboratorio y en la industria con gran flexibilidad y potencia. En este trabajo describimos el estándar GPIB, incluyendo sus características eléctricas, lenguaje de comunicación y las herramientas existentes para la programación.

2.2 Descripción de GPIB

GPIB es un estándar de conexión que permite la comunicación de un ordenador con instrumentos electrónicos de medida, como pueden ser generadores de funciones, osciloscopios, etc. Las siglas corresponden a *General Purpose Interface Bus*, pero a pesar de este nombre, fue diseñado específicamente para la conexión de instrumentos de medida. Fue creado en 1965 por la compañía Hewlett-Packard, que lo denominó originalmente HP-IB, y se popularizó con rapidez, debido a sus altas tasas de transferencia de datos (8 Mbytes/s). Para evitar la dispersión de características, los principales fabricantes acordaron la estandarización del GPIB en 1975 (IEEE 488.1), centrándose en las características eléctricas y mecánicas del bus. Una segunda estandarización (IEEE 488.2 de 1987) delimitó de forma más concreta la programación del GPIB, definiendo comandos de aparatos, formato de mensajes y estado de los instrumentos. El siguiente paso de importancia fue la adopción del formato de comandos SCPI, que estructura las órdenes a los aparatos de forma coherente, permitiendo (hasta cierto punto), la sustitución de instrumentos de distintos fabricantes con mínimos cambios (Seco, 2004)

Estructura de un sistema GPIB

Un sistema GPIB consiste en una serie de instrumentos de medida conectados a un bus, y controlados, normalmente, por un PC dotado de una tarjeta GPIB. Existe bastante libertad en la configuración topológica del bus, que, en general, es una combinación de disposiciones lineales y en estrella, como se muestra en la figura 2.1

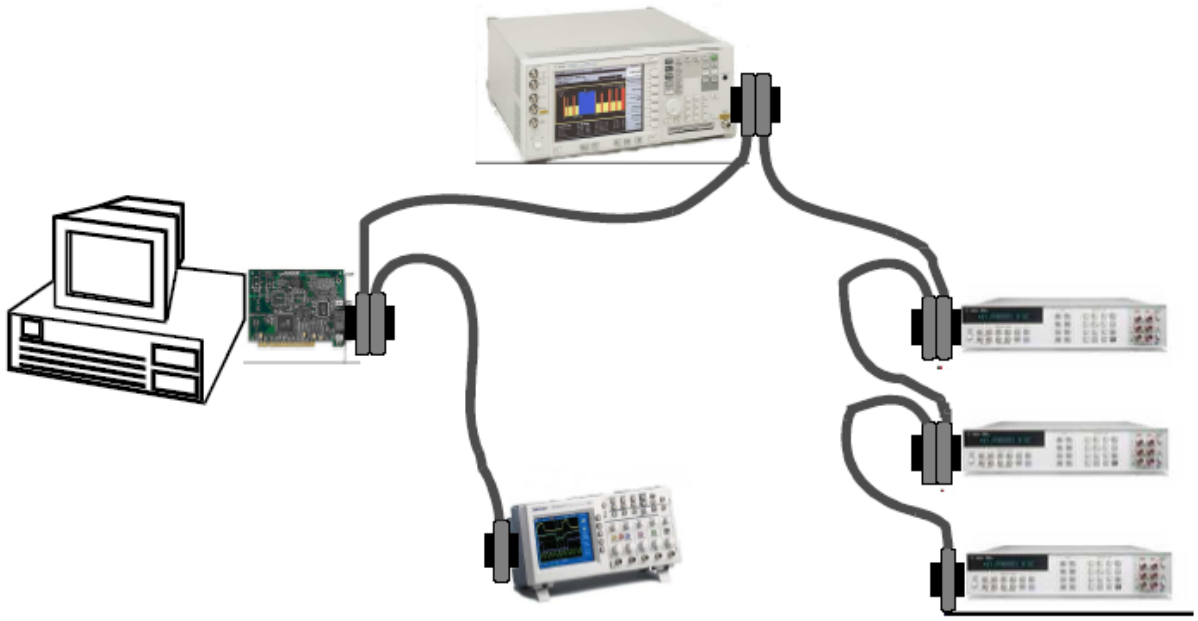


Figura 2.1. Esquema de configuración de un sistema GPIB.

El dispositivo controlador (normalmente un PC dotado de una tarjeta GPIB) gestiona el flujo de datos y comandos a los diferentes elementos del sistema.

Características eléctricas del GPIB

El bus de transmisión de datos de GPIB es de 8 bits en paralelo, y lógica negativa con niveles TTL estándar (cierto si el voltaje es £ 0.8 V y falso si el voltaje es ³ 2.0 V). Los cables y conectores tienen el aspecto típico mostrado en la figura 2.2:

Figura 2.2. Cable de conexión GPIB: aspecto físico y distribución de señales.

El bus consta de 24 pines, repartidos de la siguiente forma:

- 8 líneas de transmisión de datos (DIO1-DIO8)
- 3 líneas para el control asíncrono de la comunicación (NRFD, NDAC y NRDAV).

Mediante estas líneas se verifica la correcta transmisión de los datos, que es una de las fortalezas del GPIB.

- 5 líneas que gestionan la transmisión de comandos (ATN, IFC, REN, SRQ y EOI).
- El resto componen las tierras de las diferentes líneas.

Para que el bus GPIB alcance la velocidad de transmisión para el que fue diseñado (hasta 8 Mbytes/s), deben cumplirse los siguientes **requisitos**:

- Puede haber un máximo de 15 dispositivos conectados al bus, y al menos dos tercios de ellos deben estar encendidos.
- La separación máxima entre dos dispositivos es 4 m, y la separación promedio en toda la red debe ser menor de 2 m.
- La longitud total de la red no debe1 exceder los 20 m.

2.3 Hardware disponible para sistemas GPIB

Un sistema típico constará de un ordenador con una tarjeta controladora GPIB, más los instrumentos (compatibles con IEEE 488, obviamente). Existen tarjetas GPIB para prácticamente todos los ordenadores presentes en el mercado (PC, Macintosh, estaciones Sun, Silicon Graphics, DEC Alpha, HP RS/6000, etc). En el caso concreto del PC, las controladoras GPIB pueden conectarse al bus ISA, PCI, PCMCIA (portátiles), USB, Ethernet, Firewire, y los puertos serie y paralelo. Existen asimismo adaptadores para los estándares de comunicación RS-232 y RS-485.

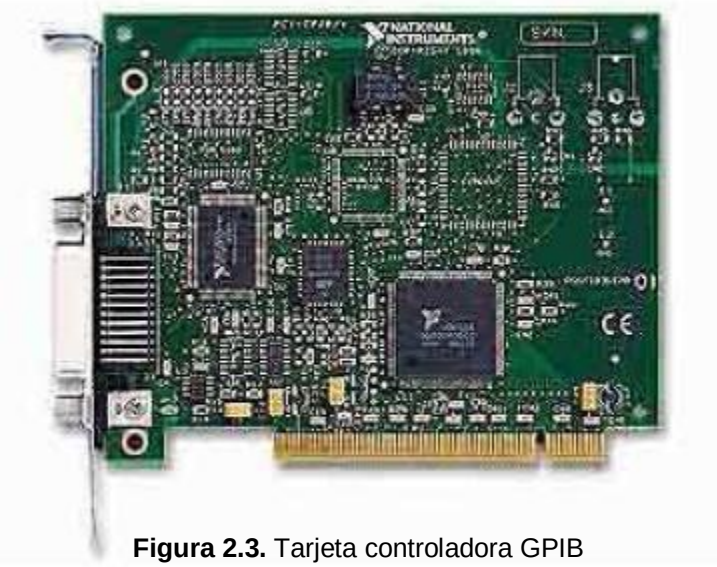


Figura 2.3. Tarjeta controladora GPIB

En la mayoría de los sistemas industriales, un PC con el sistema operativo DOS es suficiente para controlar el bus GPIB. La tarjeta GPIB se instala en un slot ISA o PCI libre. Si tanto la tarjeta como el sistema operativo son plug and play, en principio será reconocida y configurada automáticamente. En caso contrario (por ejemplo, bajo DOS), debemos elegir de forma manual:

- El puerto de entrada/salida, entre los valores 100h y 3F8h.
- El canal DMA, si se va a utilizar.

- La línea de petición de interrupción (IRQ), entre 2 y 7, también si queremos utilizarla.

Estos parámetros deben tomarse de forma que no produzcan conflictos con otras tarjetas instaladas en el PC.

Programación en GPIB

La estructura genérica de programación de un sistema GPIB se muestra en la figura 2.4:

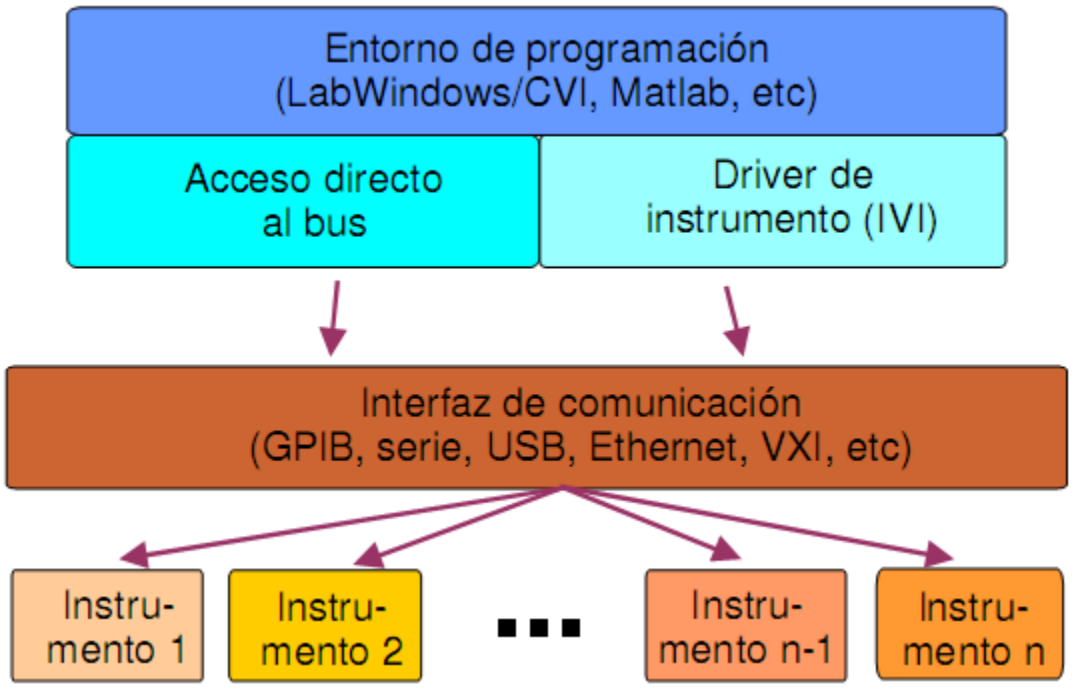


Figura 2.4. Estructura de programación de un sistema GPIB

La programación de los sistemas GPIB se realiza por intercambio de mensajes entre los dispositivos que pueden ser de dos tipos:

- De interfaz, para la gestión del bus: inicialización y direccionamiento de los aparatos.
- De datos, dirigidos a un dispositivo específico, para decirle que tome una medida, cambiar su configuración, etc.

Los dispositivos se identifican en la red GPIB por su dirección numérica (asignada en el instrumento y leída por el driver GPIB instalado en nuestro sistema), aunque mediante

el lenguaje de programación le podremos dar un handler alfanumérico más descriptivo. Por ejemplo, podemos direccionar un osciloscopio presente en la dirección 8 del bus GPIB con:

```
int osciloscopio;
osciloscopio = ibdev (0, 8, NO_SAD, T10s, 1, 0);
if (osciloscopio<0) error();
```

donde los dos primeros argumentos de entrada son la dirección de la tarjeta GPIB (en este caso 0, porque sólo hay una) y del propio osciloscopio. La instrucción ibdev además realiza la configuración de algunos parámetros de este instrumento: no hay dirección secundaria, se fija un tiempo de timeout de 10 s, se elige enviar la señal END al final de cada operación de escritura y se elige 0 como carácter EOS (final de cadena). La función devuelve, en la variable osciloscopio, el handle que usaremos en adelante para referirnos al instrumento. Los handles válidos son enteros mayores que cero.

El estándar GPIB define tres variables globales de control (a través del archivo gpib.h) que podemos leer para comprobar que todo ocurrió como se esperaba. Cada función GPIB que se ejecuta modifica dichas variables:

• Variable de estado ibsta. Cada uno de sus bits contienen información sobre diferentes aspectos de la ejecución del comando. Por ejemplo, si (ibsta & ERR) es cierto, se produjo un error.

• Variable de error iberr. Contiene el código del error que se ha producido en la ejecución de un comando GPIB.

• Variable de cuenta ibcnt. Devuelve el número de bytes transferidos en la operación de comando, escritura o lectura más reciente.

Es una buena práctica limpiar el aparato de medida antes de empezar a enviarle órdenes:

```
ibclr(osciloscopio);
```

Las órdenes básicas para escribir y leer en los aparatos son ibwrt e ibrd, respectivamente. Cada instrumento GPIB posee un manual de programación en el que se explican los diferentes comandos que entiende el aparato. Gracias a los sucesivos estándares, se ha conseguido cierta homogeneidad entre los fabricantes, tanto en las órdenes concretas como en su sintaxis (ver el apartado Comandos SCPI más adelante). Las órdenes usadas en este apartado corresponden al osciloscopio TDS1012 de Tektronix.

Con las siguientes órdenes hacemos que el osciloscopio muestre los canales 1 y 2, y fijamos el modo de adquisición en promedio:

```
ibwrt (osciloscopio, "SEL:CH1 ON;CH2 ON");
ibwrt (osciloscopio, "ACQ:MOD AVE");
```

Algunas versiones de la función ibwrt (por ejemplo, la de LabWindows/CVI) exigen como tercer argumento el número de bytes que hay que transmitir; en este caso habría que usar:

```
strcpy (comando, "SEL:CH1 ON;CH2 ON");
nbytes = strlen (comando);
ibwrt (osciloscopio, comando, nbytes);
```

Como medida de seguridad, se pueden detectar posibles errores después de la ejecución de cada comando, con la comprobación:

```
if (ibsta & ERR) printf("Ocurrió un error\n");
```

En este caso, la variable iberr indica el error cometido; por ejemplo:

```
if (iberr & EARG) printf("Argumento no válido\n");
```

Para recibir datos del osciloscopio (por ejemplo, medir la amplitud pico a pico de la señal del canal 2), usaríamos:

Algunas versiones de la función ibwrt (por ejemplo, la de LabWindows/CVI) exigen como tercer argumento el número de bytes que hay que transmitir; en este caso habría que usar:

```
strcpy (comando, "SEL:CH1 ON;CH2 ON");
nbytes = strlen (comando);
ibwrt (osciloscopio, comando, nbytes);
```

Como medida de seguridad, se pueden detectar posibles errores después de la ejecución de cada comando, con la comprobación:

```
if (ibsta & ERR) printf("Ocurrió un error\n");
```

En este caso, la variable iberr indica el error cometido; por ejemplo:

```
if (iberr & EARG) printf("Argumento no válido\n");
```

Para recibir datos del osciloscopio (por ejemplo, medir la amplitud pico a pico de la señal del canal 2), usaríamos:

```
ibwrt(osciloscopio, "MEASU:IMM:SOU CH2");
ibwrt(osciloscopio, "MEASU:IMM:TYP PK2");
ibwrt(osciloscopio, "MEASU:IMM:VAL?");
```

La primera línea indica que vamos a tomar una medida cuya fuente (source) es la señal del canal 2; la segunda que el osciloscopio debe medir la amplitud pico a pico (PK2PK) de la señal y la tercera envía al osciloscopio la orden para tomar la medida y devolver el resultado al ordenador. Las órdenes que exigen respuesta del instrumento acaban con signo de interrogación. Por último leemos la respuesta del osciloscopio y la mostramos por pantalla.

```
char medida[100];
ibrd(osci,rd,20);
printf("Voltaje = %f V\n", atof(rd));
```

Como se puede ver, el aparato devuelve el resultado de la medida como una cadena ASCII en el array medida.

La mayoría de comandos GPIB son comunicados a los instrumentos de forma prácticamente instantánea, con la excepción notable de la transmisión y captura de formas de onda. Por ejemplo, la captura de una forma de onda de 2500 puntos, en modo ASCII, desde un osciloscopio Tektronix, tarda 3.5 segundos. Puesto que las funciones ibwrt e ibrd son síncronas, el fujo del programa se interrumpe hasta completar la operación de I/O, aunque realmente el tráfico de la información es gestionado por la controladora GPIB y no por el procesador. Si durante ese tiempo se quiere, por ejemplo, atender al interfaz de usuario, pueden emplearse las versiones asíncronas ibwrta e ibrda de las funciones anteriores, y disponer del procesador mientras se completa la operación de I/O. En este caso, GPIB proporciona una rutina para la sincronización:

```
ibwait(osciloscopio, mascara);
```

donde mascara es un entero de 16 bits con la misma estructura que la variable ibsta. Los bits más útiles en este caso son CMPL (operación de I/O completada) y TIMO (timeout); este último debería ser habilitado para evitar que el programa se cuelgue. La función ibstop puede usarse para detener forzosamente las transmisiones asíncronas.

Comandos SCPI

A pesar de los estándares IEEE 488 y 488.2, existía libertad para que cada fabricante eligiera los comandos de sus instrumentos. En 1990 un grupo de empresas fabricantes de instrumentos acordaron crear un conjunto de órdenes con una sintaxis común, que fue llamada SCPI (Comandos Estándar para Instrumentos Programables). Lógicamente, SCPI se construyó respetando los principios del anterior 488.2.

Si dos instrumentos (por ejemplo, dos osciloscopios), de fabricantes distintos, se adhieren al estándar SCPI, es teóricamente posible intercambiarlos con mínimas modificaciones en el programa de control.

Los comandos SCPI se escriben como texto ASCII, y tienen una estructura jerárquica por niveles, separados por dos puntos:

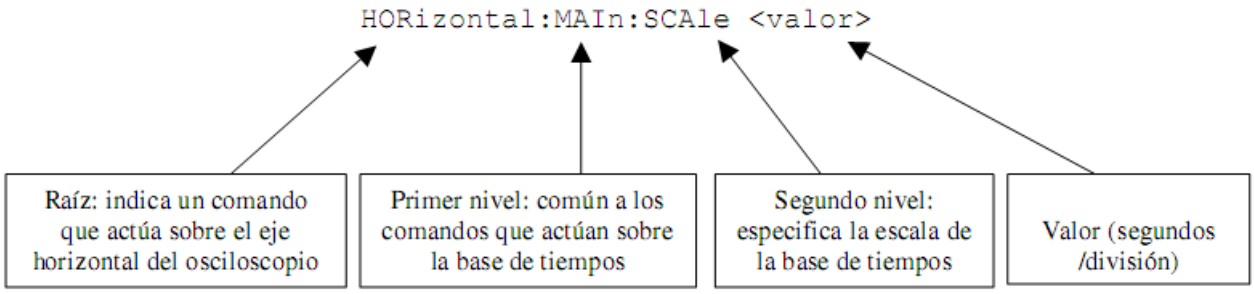


Figura 2.5. Estructura de los comandos estándar para instrumentos programables

Los caracteres en mayúsculas son necesarios para especificar la orden, mientras que los que están en minúsculas pueden suprimirse, sirviendo sólo para facilitar la lectura de programas por usuario. Los comandos en sí pueden ser escritos indistintamente en mayúsculas o minúsculas. Así, SCALE, sca y scale representan todos al mismo comando. Por ejemplo:

HOR:MAIN:SCA 5.0E-4

Establece la escala de la base de tiempos en 500 µs/división.

El signo de dos puntos (:) separa los niveles de la jerarquía. Si quisiéramos preguntar al osciloscopio por la escala actual de la base de tiempos habría que escribir:

HOR:MAIN:SCA?

Los comandos se pueden concatenar utilizando un punto y coma (;). Por ejemplo, se puede escribir la orden:

ACQuire:MODe AVE; NUMAVg 16

que indica al osciloscopio que la adquisición va a ser en modo promedio (average) y que va a constar de 16 muestras. Nótese que los niveles jerárquicos ACQuire:MODe, al ser comunes, sólo necesitan ser indicados una vez; es decir, la orden anterior equivale a:

ACQuire:MODe AVE; ACQuire:MODe NUMAVg 16

Una ventaja del estándar SCPI es la definición homogénea de comandos para todos los aparatos de una misma clase; por ejemplo, para un osciloscopio, las principales categorías de raíz se muestran en la tabla 1:

ACQuire	Adquisición de señales (muestra, promedio, etc)
CALibrate	Calibración y diagnóstico del aparato
CURSor	Control del cursor
DISplay	Control de la pantalla (formato YT o XT, contraste, etc)
HORizontal	Control de la base de tiempos (posición, escala, etc)
MATH	Operaciones matemáticas sobre las ondas (suma, resta, FFT)
MEASUrement	Medidas sobre las formas de onda (amplitud, frecuencia, etc)
TRIGger	Control del disparo (fuente, nivel, pendiente, etc)
CH	Control vertical de las señales (posición, amplitud, etc)
CURV, DAT y WFMP	Captura de ondas del osciloscopio

Tabla 1: principales apartados de la jerarquía de comandos de un osciloscopio, según el estándar SCPI.

VISA e IVI

Como se ha podido ver, la programación de un sistema GPIB a base de comandos SCPI es bastante laboriosa. Aunque hace años era la única opción disponible; hoy en día disponemos de drivers para los principales entornos de programación que permiten el acceso a los instrumentos a más alto nivel.

Históricamente, el primer paso hacia la estandarización fue VISA (Virtual Instruments Software Architecture), un convenio de Agilent y NI para acceder a los instrumentos de la misma forma independientemente de la interfaz física (GPIB, puerto serie, etc).

En 1998 surgió el consorcio IVI (Interchangeable Virtual Instruments) entre una treintena de compañías, incluyendo usuarios de sistemas como Boeing, y fabricantes de hardware como Agilent, Tektronix, NI, etc, con el objetivo de alcanzar una estandarización de los drivers de los instrumentos. En concreto, IVI aporta las siguientes novedades:

- Adopción de VISA (independendizando la programación respecto al bus de interfaz).
- Posibilidad del intercambio de instrumentos, incluso de distintos fabricantes.
- Posibilidad de trabajar con instrumentos simulados durante el desarrollo de aplicaciones, cuando la disponibilidad de los equipos está restringida.
- Acceso a los instrumentos mediante una caché de estado, que optimiza el tráfico del bus cambiando el estado del instrumento sólo de forma incremental.
- Posibilidad de programación multihilo.

Los drivers IVI están clasificados en ocho clases o plantillas, correspondientes a los siguientes instrumentos de medida:

- Fuentes DC
- Multímetros digitales
- Generadores de funciones
- Osciloscopios
- Medidores de potencia

- Generadores de RF
- Analizadores de espectros
- Conmutadores de señales (switches)

Con una librería IVI, el programador de C puede emplear rutinas de alto nivel sin necesidad de conocer el conjunto de comandos SCPI que el instrumento entiende. Por ejemplo, la rutina C usada para configurar la salida de un generador de funciones de Agilent mediante el driver IVI es:

```
hp33120a_ConfigureStandardWaveform (Generador, "1",
    HP33120A_VAL_WFM_SINE, amplitud, 0.00, frecuencia*1e3, 0.00);
```

que es mucho más compacta y comprensible que la secuencia de comandos GPIB necesarios para obtener el mismo efecto. Como consecuencia, el desarrollo de aplicaciones de esta forma se ha agilizado considerablemente respecto al uso de comandos SCPI.

2.4 Software existente para GPIB

Programas propietarios

La ventaja evidente de estos programas es que pueden ser empleados nada más conectar los instrumentos, y proporcionan ya hechas las funciones más comunes que uno puede desear realizar, sin necesidad de programar.

Las desventajas son también claras: por tratarse de software cerrado, sólo puede ser usado para la tarea para la que fue diseñado, y además son imposibles de integrar con otros programas (Seco, 2004)

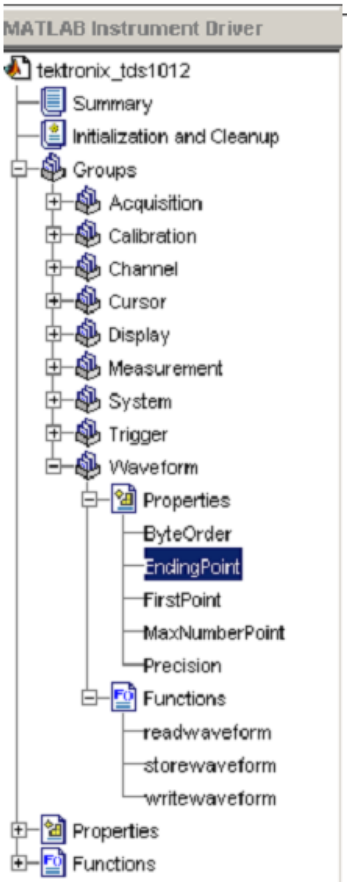
Caja de herramientas de control de instrumentos en MATLAB. (Instrument Control Toolbox)

Matlab, que fue en origen un conjunto de rutinas para manipulación de matrices, ha evolucionado con el tiempo para convertirse en un entorno de programación de

propósito general con gran potencia matemática y aplicabilidad en muchos ámbitos de la ciencia y la ingeniería, gracias a sus módulos de extensión (toolboxes) de procesamiento de señales, control, ecuaciones diferenciales, y un largo etcétera. En muchos casos las cajas de herramientas representan el estado del arte en programación numérica en sus respectivos campos.

La caja de herramienta ha sido diseñada de forma que el acceso a los instrumentos de medida se realiza mediante objetos, que pueden ser de dos clases: interfaz o de dispositivo (ver figura 2.6). Mientras que el primer método equivale a la programación con comandos SCPI, el segundo enfoque permite explotar toda la potencia de IVI. Para ello es preciso disponer del driver de Matlab correspondiente (extensión .mdd), obtenible en la página de Mathworks o bien usar una utilidad llamada makemid para convertir (o "envolver") el driver IVI ya instalado en el sistema. La utilidad midedit permite ver y modificar todos los objetos definidos en los aparatos, sus propiedades, y las funciones ejecutables (Seco, 2004)

La figura 2.6 muestra el árbol con los objetos predefinidos por el driver IVI sobre un osciloscopio Tektronix y un ejemplo de código para la captura de una forma de onda desde el mismo.




```
% Conexión al osciloscopio (dirección GPIB 8)
g_osci = gpib('ni', 0, 8);

% Construye un objeto de dispositivo con el driver
% indicado usando el objeto de interfaz g_osci
d_osci = icdevice('tektronix_tds1012', g_osci);

% Creamos un objeto 'Canal' en el osciloscopio que
% contiene los dos canales
canales=get(d_osci,'Channel');
% Configuramos el canal 2
set(canales(2),'State','on')
set(canales(2),'Position',0)
set(canales(2),'Scale',0.1)  % 100 mV/div

% Crea un objeto del tipo 'Forma de onda' en el
% osciloscopio
captura_onda=get(d_osci,'Waveform');

% Invoca la función 'readwaveform' definida sobre
% el objeto 'ondacapturada' (ver el árbol a la
% izquierda) para capturar la onda del canal 2
[v,t]=invoke(captura_onda,'readwaveform',
                    'channel2');

% Dibuja la onda
figure(1),plot(t*1e6,v),xlabel('Tiempo (\mus)')
```

Figura 2.6. Estructura del driver IVI de un osciloscopio y ejemplo de uso de la caja de herramientas de control de instrumentos en Matlab para capturar una forma de onda a través del bus GPIB.

2.5 Tarjeta de adquisición de datos DaqBoard/1000 IOtech

Figura 2.7. Características de algunos elementos de la Daq Board/1000

DaqBoard/1000 es una tarjeta de alta velocidad, multi-función, plug-and-play de adquisición de datos para PC con bus PCI. Cuenta con una de 16-bit, 200-kHz convertidor A / D, calibración digital, bus mastering DMA, dos de 16-bit, 100-kHz D / A,

24 líneas digitales I / O, los contadores de cuatro, y dos temporizadores. Hasta cuatro tableros pueden ser instalados en un PC.

Un 68-pin conector SCSI III en el tablero proporciona acceso a todas las señales de entrada y de salida. El DaqBoard/1000 acomoda todas las E / S con un cable y una ranura PCI. El 68-pines I / O conector es lógicamente dividido en tres funciones:

- Entrada analógica for16 una sola terminal o diferencial de 8 entradas analógicas con 7 rangos de software programable bipolar (± 10 V a ± 156 mV a gran escala).
- 24 líneas de propósito general E / S digital.
- 4 entradas de contador, 2 salidas de temporizador, y 2 salidas analógicas.

La exploración a bordo secuenciador le permite seleccionar hasta 512 combinaciones de canal / rango. El secuenciador escanea todos los canales de la detección en 5µs ó 10 mS / canal.

Bus mastering analógico y digital permite que datos de entrada del contador, así como analógica y digital de datos de salida, el flujo entre el PC y el DaqBoard sin consumir tiempo de CPU.

DaqBoard/1000 soporta modos de disparo que incluyen:

• Digital y el patrón de activación - Los foros se han separado línea digital de entrada de disparo, lo que permite TTL-nivel de disparo y las latencias de menos de 5 ms. El disparador puede ser programado para el nivel de la lógica o el borde de disparo. En el patrón de activación, cualquiera de los puertos de entrada digital actúa como el puerto de disparo. Usted puede programar el patrón digital.

• Software basado en disparo - El PC detecta el evento de disparo del sistema analógico, digital, o las lecturas mostrador. Seis modos de disparo pre y post-son compatibles.

Los dos de 16-bit, los canales analógicos de 100 kHz de salida tienen una salida de -10 V a +10 V. Utilizar Bus Mastering DMA, D, cada uno / una salida puede una forma de onda.

Otras características de la DaqBoard/1000 incluyen:

• 24 puntos digitales de nivel TTL líneas E / S. Se dividen en tres puertos de 8-bits.

• Cuatro contadores de 16-bit. Cada uno puede aceptar entradas de frecuencia hasta 10 MHz. Los contadores pueden conectar en cascada en dos contadores de 32 bits.

• Dos salidas de 16-bit de temporizador. Cada uno puede generar ondas cuadradas de 16 Hz a 1 MHz.

• Configuración a través de software. No hay interruptores o puentes en la DaqBoard/1000.

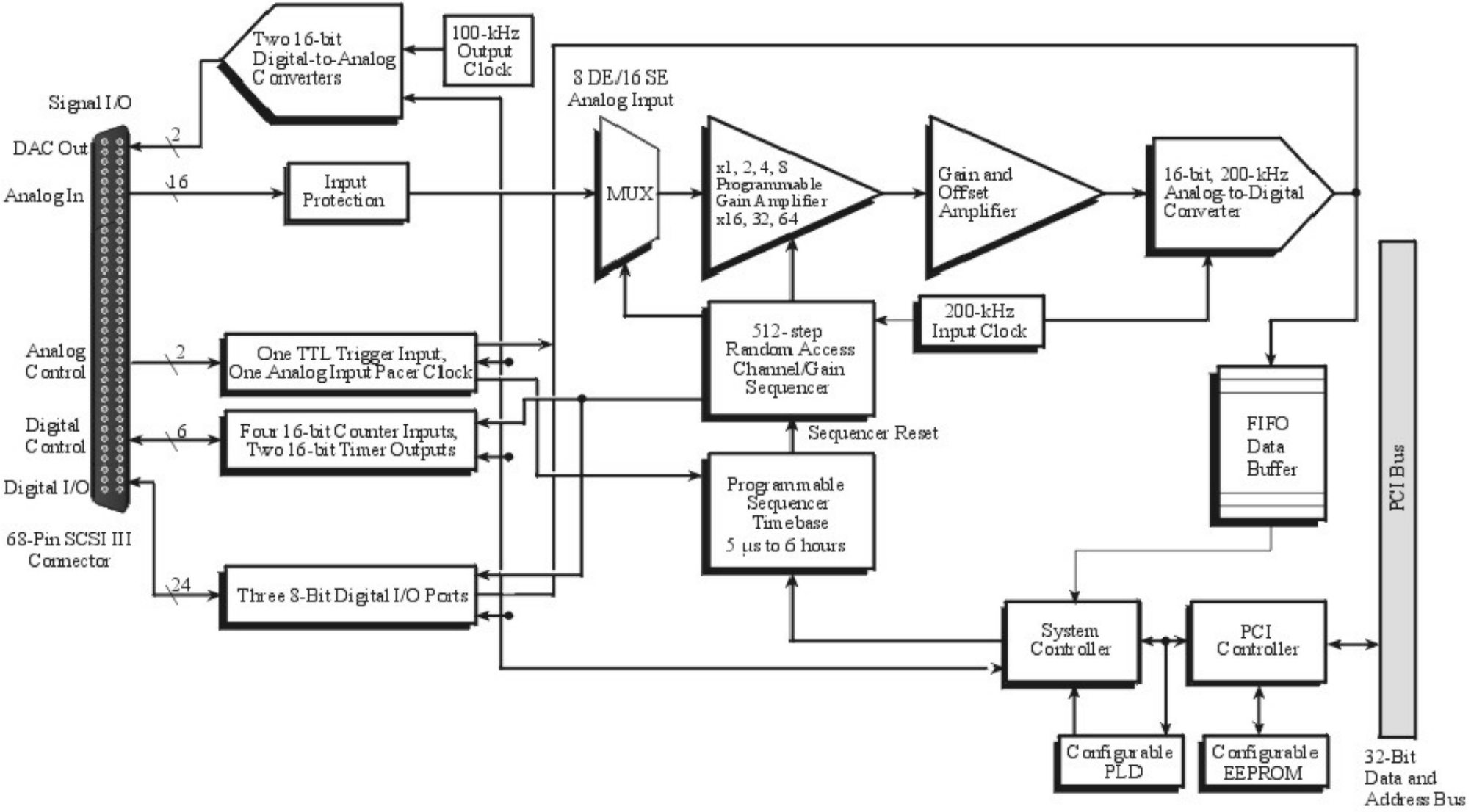


Figura 2.8. Esquema de conexión de la tarjeta de adquisición de datos al PCI bus

Conectores E / S.

Todas las señales de entrada y de salida están disponibles a 68 de la Junta de pines del conector SCSI III. Capítulo 2 se incluye un pinout. El siguiente cable y las opciones de la caja de bornes se puede utilizar para ofrecer convenientes conexiones de los terminales de tornillo para todas las señales líneas I / O.

Cable de empalme: El CA-G56 es un 68-conductor cable blindado. Se utiliza para conectar un tablero de la serie DaqBoard/1000 a un tablero de la TB-100 de terminación. La longitud del cable es de 3 pies.

TB-100: TB-100 es una placa de terminación opcional. Ofrece cómodas conexiones de terminal de tornillo para todas las señales líneas I / O de una junta de la serie DaqBoard/1000.

Especificaciones –Serie DaqBoard/1000

I/O Comparison Matrix for DaqBoard/1000 Series Boards					
DaqBoard Identity	**Analog Input Channels**	**Analog Output Channels**	**Digital I/O Channels**	**Counter Inputs**	**Timer Outputs**
1000	16	2	24	4	2

Tabla 2.1. Comparación de matriz de entrada y salida para tarjetas de serie DaqBoard/1000

Especificaciones Generales Aplicables a los DaqBoard/1000, / 1005

Warm-up: 1 hora a las especificaciones de puntuación

Rango de voltaje de la fuente: 4,75 a 5,25 VDC VDC (bus PCI)

Consumo de energía (por tarjeta): 3,5 W

Temperatura de funcionamiento: 0 a +60 ° C

Temperatura de almacenamiento: de -40 a +80 ° C

Humedad relativa: 0 a 95% sin condensación

Vibración: MIL STD 810E

Señales de E / S Conector: 68-pin SCSI de tipo III, lleva todas las E / S analógicas y digitales de E / S

Dimensiones: 165 mm W x 15 mm x 108 mm D H (6,5 "x 0,6" x 4,2 ")

Peso: 160 g (0,35 libras)

Aplicable a analógico DaqBoard/1000 Entradas, / 1005

Canales: 16 de una sola terminal o diferencial 8, programables en función de cada canal como una sola terminal o diferencial bipolar.

Ancho de banda: 500 kHz

Tiempo de establecimiento: 5 mS máximo a 1 LSB para el paso a gran escala

Máximo voltaje de entrada: 11 V respecto al analógico común

Protección de sobre-voltaje: 35V ±

Rangos: programable de software a través de un secuenciador por canal.

Voltage Range (Note 1)	**Accuracy (Note 2) One Year, 0 to 35°C ±(% reading + % range)**	**Input Noise (Note 3) (LSB rms) 10 Hz to 200 kHz**
	Absolute	**Typical**
-10 to +10 V	0.015+.005	1
-5 to +5 V	0.015+.005	1
-2.5 to +2.5 V	0.015+.005	1
-1.25 to +1.25 V	0.015+.005	2
-0.625 to +0.625 V	0.015+.008	2
-0.3125 to +0.3125 V	0.015+.008	3
-0.156 to +0.156 V	0.02+.008	3

Tabla 2.2. Datos específicos de la tarjeta de adquisición. Rango de voltaje, ruido de entrada etc.

Notas:
1) Las especificaciones asumen canales de entrada diferencial escaneado de una sola, 200 kHz velocidad de barrido, sin filtrar.
2) especificación de precisión es exclusiva de ruido. Las mediciones se realizaron en P1.
3) Entradas cortocircuito a tierra de la señal (SGND). 8192 muestras.

Especificaciones aplicables a DaqBoard/1000, / 1005

Tipo: Aproximaciones sucesivas de hasta 200 kHz tipo de conversión **Resolución**: 16 bits

Tiempo de conversión: 5 mS

Máxima velocidad de muestreo: 200 kHz

No linealidad diferencial: ± 2 LSB máxima Integral linealidad: ± 1 LSB máxima **Faltan códigos**: Ninguno, más la temperatura de funcionamiento de rango completo Fuentes del disparador: 4, seleccionables individualmente para iniciar y detener una adquisición. Adquisición de detención puede producirse en un canal diferente al de adquisición de inicio, la adquisición de stop pueden ser desencadenadas a través de los modos 2, 3 ó 4 se describe a continuación. Pre-disparador es compatible con los períodos pre-disparador fijo o variable. (IOtech, 2002)

1. De un canal digital de disparo: Una entrada digital por separado está previsto disparo digital. Latencia: 5 µs máx.
2. Disparo Digital Patrón: 8 o 16 bits patrón de activación en cualquier entrada digital. Programable para el disparador en la igualdad, arriba, abajo, o dentro o fuera de una ventana. bits individuales pueden ser enmascarados por "no importa" condición. Latencia: Un máximo de escaneado período.
3. Contador / Totalizador de Disparo: Contador / insumos totalizador puede desencadenar una adquisición. El usuario puede seleccionar para que activen en una frecuencia o en el recuento total que son iguales, encima, debajo o dentro / fuera de una ventana. Latencia: Un máximo de escaneado período.
4. Software de disparo: disparo se puede iniciar bajo control del programa.

Salidas analógicas Aplicable a DaqBoard/1000; no se aplica a DaqBoard/1005

Dos canales de salida analógica se actualizan de forma sincrónica en relación a los insumos escaneados, y marcó de un reloj de a bordo ya sea interna o una fuente de reloj externa. Salidas analógicas también se pueden actualizar de forma asincrónica, independiente de cualquier otra exploración en el sistema.

Bus mastering DMA.

Establece la CPU y el sistema de transferencias de datos independientes, garantizando resultados precisos que son con independencia de las actividades del sistema. Streaming desde el disco o la memoria es compatible, permitiendo longitud continua, casi infinita, los resultados de forma de onda (limitado sólo por los recursos disponibles del sistema PC).

Canales: 2 canales DAC (DAC0, DAC1)

Resolución: 16 bits

Variación de tensión de salida: ± 10 V

Digital Feedthru: 50 mV cuando se actualicen

Corriente de salida: ± 10 mA

Desplazamiento de error: ± 0,0045 V máximo

Ganancia de error: ± 0,01%

Velocidad de actualización: 100 kHz máximo, mínimo 1,5 Hz (sin mínimo con el reloj externo)

Tiempo de establecimiento: 10 µs máximo de 1 LSB para el paso a gran escala

Reloj Fuentes: 4 programables

1. A bordo de D / A de reloj, independiente de análisis de entrada de reloj
2. A bordo de escaneo de entrada de reloj
3. Exteriores D / A la entrada de impulsos, con independencia de reloj externo de entrada de escaneo.
4. Exteriores de escaneo de entrada de reloj E / S Aplicable a DaqBoard/1000, Digital / 1005

   Canales: 24, ampliable a 208 con opciones exteriores DBK
5. Entrada Modos de escaneo: programable 2 \
   1. Asíncrono, bajo control del programa en cualquier momento en relación con la entrada de escaneo.
   2. Síncrono con la entrada de escaneo

Puertos: 3 x 8-bits (82C55 emulación). Cada puerto se puede programar como entrada o salida.

Características de entrada: 100. serie, a 20 pF común.

Protección de entrada: ± 8 kV ESD abrazadera de diodos en paralelo

I / O niveles: TTL

Muestreo / Tasa de actualización: 200 kHz máximo.

Características de salida: Salida 12 mA por pin, 200 mA continuos total (por banco de 24 productos) Aplicable a los contadores de DaqBoard/1000, / 1005 entradas de contaje se pueden escanear de forma sincrónica con entradas analógicas y digitales escaneadas, basados en las temporizador interno programable, o una fuente de reloj externa. Bus mastering DMA establece la CPU y las transferencias de datos independientes del sistema, asegurando el rendimiento de adquisición de datos con independencia de las actividades del sistema. Los contadores se pueden configurar para desactivar cuando se lee, o al totalizar y claro bajo control del programa.

Canales: 4 x 16-bit, como en cascada de 2 x de 32-bit.

Frecuencia de mediciones: 10 MHz máxima.

Rango de entrada de señal: -15 V a +15 V.

Características de entrada: 2,7 k. serie en paralelo con 20 pF a 10 k. común y a +5 V.

Protección de entrada: ± 8 kV ESD abrazadera de diodos en paralelo.

Nivel de disparo: TTL.

El ancho mínimo del impulso: 50 ns alta, baja 50 ns.

Frecuencia / pulso Aplicable a DaqBoard/1000 Generadores, / 1005.

Canales: 2 x 16-bits.

Onda de Salida: onda cuadrada.

Tasa de salida: 1 tipo de base MHz dividido por 1 a 65535 (programable).

Alto nivel de salida de voltaje: 2,0 V minimum@-3.75 mA, 3,0 V minimum@-2.5 mA.

Bajo nivel de tensión de salida: 0,4 V maximum@2.5 mA.

Accesorios y Cables Aplicable a DaqBoard/1000, / 1005

Terminación Board (Junta de TB-100) de terminación con terminales de tornillo para acceder a todas las E / S para una DaqBoard/1000 o DaqBoard/1005. La caja de bornes se conecta al conector de 68 pines del tablero de la serie de DaqBoard/1000 a través de un cable CA-G56.

Montaje en bastidor (Rack3) Este es un kit para el montaje de la placa TB-100 de terminación de un rack.

68-Conductor Cable (CA-G56)

68-conductores, 3 pies de cable blindado. Se utiliza para conectar un tablero TB-100 terminales a un DaqBoard/1000.

2.6 Conclusiones del capítulo II

- Una tecnología adecuada para el control de instrumentos de medición es usando las tarjetas con arquitectura GPIB, cuyo estándar de servicio es muy utilizado en el mercado.
- Aprovechando el flujo más accesible que ofrecen los fabricantes, es ventajoso utilizar adaptadores que permiten conectar instrumentos GPIB a (por ejemplo), una red Ethernet, operando de una forma más o menos transparente para el programador.
- Asimismo, el estándar VISA ha simplificado considerablemente el desarrollo de software, al hacer, hasta cierto punto, 'transparente' al bus físico de conexión.
- Debe tenerse en cuenta para futuras mejoras en esta temática del control de instrumentos, la incorporación de instrumentos de medida de gama alta que mantienen abiertas todas las puertas para la conexión. Es el caso por ejemplo de los osciloscopios de la serie Infiniuum de Agilent, que incorporan de serie dos puertos USB, dos puertos PS/2, GPIB y conexión a Ethernet.

CAPÍTULO III

TÉCNICAS INTELIGENTES PARA INTERFACES DE MATLAB EN ACCIONAMIENTOS ELÉCTRICOS

3.1 Introducción

Establecida desde hace varios años como una tecnología de vanguardia, la lógica difusa ó fuzzy logic está penetrando con fuerza en el mundo del control y promete convertirse en el método de mando universal de toda clase de dispositivos eléctricos y electrónicos. La Lógica Difusa es, desde un punto de vista práctico, un método de razonamiento estadístico que permite especificar los problemas de control del mundo real en términos probabilísticos, sin necesidad de recurrir a modelos matemáticos y con un nivel de abstracción mucho más elevado. En contraste con la lógica convencional, que utiliza conceptos absolutos para referirse a una realidad, la lógica difusa la define en grados variables de pertenencias a los mismos, siguiendo patrones de razonamientos similares a los del pensamiento humano.

Así por ejemplo, mientras dentro del marco rígido de la lógica 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 formal 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 aún así llegar a la solución correcta.

La lógica difusa es una ciencia relativamente reciente, aunque la idea de vaguedad que promulga ya había sido discutida desde el siglo XVIII por Berkeley, Hume, Kant, Bayes y otros pensadores. Incluso Aristóteles, creador de la lógica formal, admitía la existencia de diferentes grados de verdad y falsedad. Sin embargo, es Lofti Zadeth, profesor de computadores de la Universidad de Berkeley, quien en 1965 propone un método de razonamiento abstracto similar a patrón de pensamiento humano para representar los problemas de control del mundo real y crea la lógica difusa. A comienzos de los setenta 70s, el ingeniero Ebrahim Mandani, basado en la teoría de Zadeth, desarrolla el primer sistema de control fuzzy práctico, aplicado a una máquina de vapor.

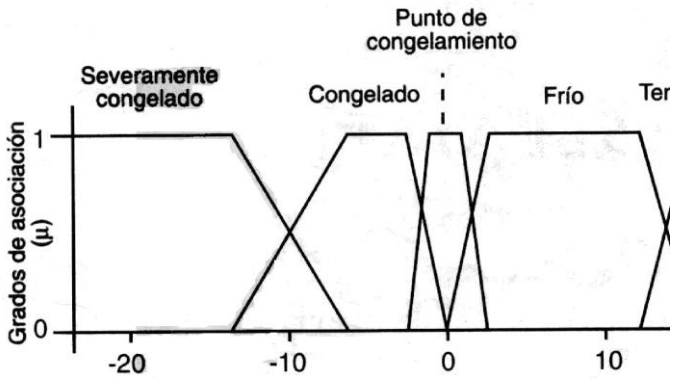


Figura 3.1. Valores reales y valores difusos.

El sistema de Mandani combinaba la experiencia de un operador humano con un conjunto de reglas lógicas para controlar automáticamente la cantidad de vapor o throuttle y la temperatura de la caldera de acuerdo a la presión de esta última y la velocidad de la máquina. Las dos entradas (velocidad y presión) eran procesadas de acuerdo a un algoritmo y producían dos salidas (vapor y temperatura) que actuaban sobre el proceso en la forma deseada. A finales de los 70s, los ingenieros daneses Lauritz Meter Holmblad y Jens Jurgen Ostergaard desarrollan el primer sistema de control fuzzy comercial, destinado a una planta de cemento.

A pesar de que han transcurrido muchas décadas desde su creación, hasta hace poco el mundo occidental está reconociendo el verdadero valor de la fuzzy logic. Además de factores culturales, dos razones explican esta actitud. En primer lugar la palabra fuzzy sugería algo confuso y sin forma. Esto alejaba sicológicamente a la comunidad técnica de la idea de utilizarla prácticamente. En segundo lugar, no había forma de probar analíticamente funcionaba correctamente debido a que la fuzzy logic, en contraste con la teoría de control convencional, no estaba basada en modelos matemáticos. La situación en Japón fue diferente.

Los japoneses aceptaron fonéticamente la palabra fuzzy, sin traducción y libre de las connotaciones negativas normalmente asociadas con el término, y adoptaron la teoría de Zadeth como propia. Esto le permitió evolucionar más tempranamente que los occidentales a la fase experimental. Así lograron comprobar que no eran necesarias las imposiciones para desarrollar y producir sistemas inteligentes.

Actualmente Japón es el líder mundial en la producción de aplicaciones basadas en fuzzy logic, con ventas estimadas para 1997 en 6.1 billones de dólares. En Japón funcionan también el más espectacular de todos los sistemas fuzzy creados por el hombre: el subterráneo de Sendai, inaugurado en 1987. Desde entonces, un controlador inteligente ha mantenido los trenes rodando velozmente a lo largo de la ruta, frenando y acelerando suavemente, deslizándose entre estaciones y deteniéndose exactamente, sin perder un segundo ni sacudir bruscamente un solo pasajero.

El interés actual en la lógica difusa surge también de la necesidad impuesta por nuevas tecnologías como la inteligencia artificial y las redes neuronales de disponer de sistemas expertos capaces de procesar información, tomar decisiones y responder a estímulos en forma similar al cerebro humano. Los investigadores están utilizando técnicas fuzzy para diseñar redes neuronales y estas, a su vez, para producir reglas de lógica difusa.

Una lavadora controlada por lógica fuzzy, por ejemplo, puede distinguir una prenda ligeramente sucia, lavando esta última con mayor vigor que la primera. Adicionalmente, puede calcular automáticamente el volumen de la carga de ropa, la velocidad, el nivel de agua y detergente y los tiempos óptimos de lavado, centrifugado, enjuague, agitación, etc. Si se agrega una red neuronal, esta le permitirá aprender nuevos hábitos mediante el desarrollo dinámico de nuevas reglas.

Actualmente, muchos productos de uso corriente (cámaras fotográficas y de video, electrodomésticos, etc); así como una gran variedad de controladores industriales, dispositivos médicos y otros sistemas relativamente complejos, están basados en lógica fuzzy. La tendencia continuará a medida que los diseñadores encuentren nuevas aplicaciones para esta tecnología. Otros usos de la lógica fuzzy incluyen:

- Modelos de control de trenes, aviones, botes y otras naves.
- Sistemas de seguridad para el hogar y la oficina.
- Sistemas de control y predicción climáticos.
- Controladores de velocidad de motores AC y DC.
- Servomecanismos y robots.
- Control de líneas de producción.

La facilidad de la fuzzy logic para adquirir y representar conocimientos ha estimulado también su aplicación en la solución de problemas sociológicos, sicológicos, políticos, administrativos, económicos, epidemiológicos y de otras disciplinas.

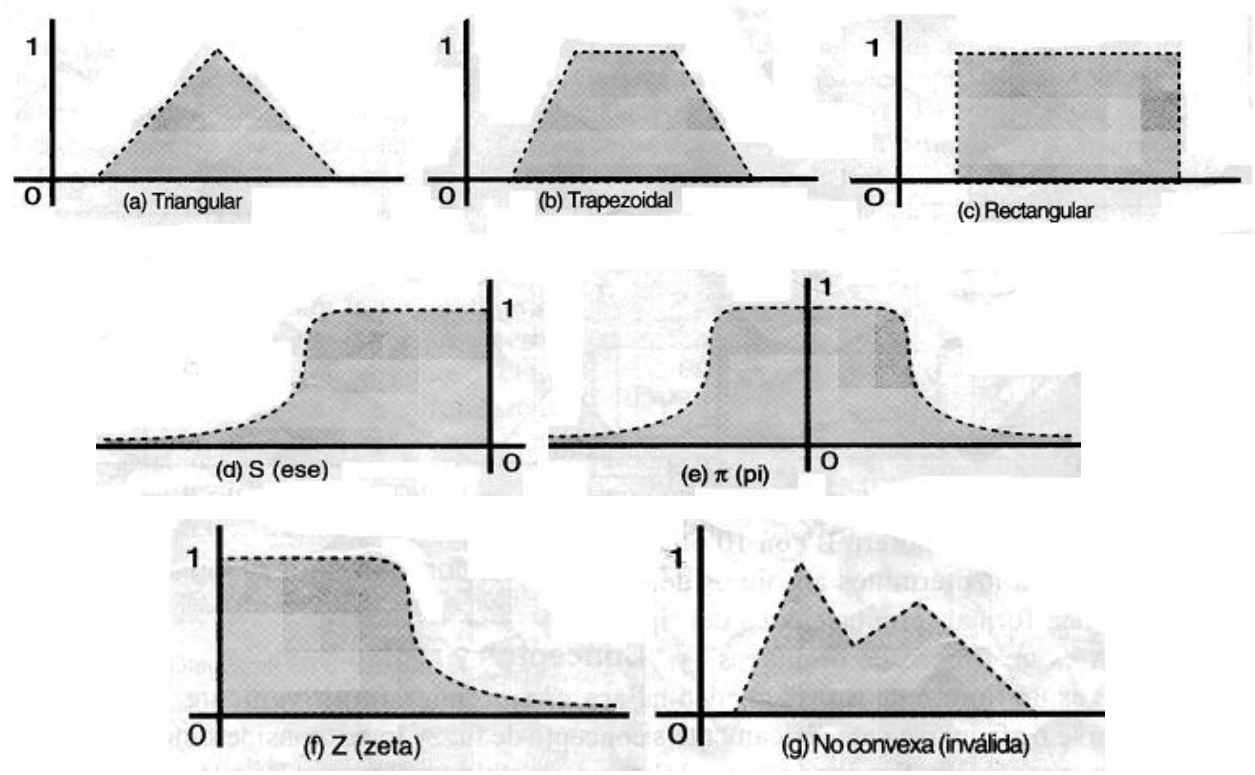


Figura 3.2. a), b), c),d),e),f),g). Funciones de pertenencias comunes.

En general existen cinco tipos de situaciones en las cuales la aplicación de técnicas de control fuzzy resulta ventajosa o necesaria:

1. Sistemas complejos que son difíciles de modelar por métodos convencionales.
2. Sistemas controlados por expertos humanos.
3. Sistemas con entradas y salidas complejas y continuas.
4. Sistemas que utilizan la observación humana como entrada o como base de las reglas.
5. Sistemas que son confusos por naturaleza, como los encontrados en las ciencias sociales y del comportamiento.

La lógica difusa puede abarcar muchos aspectos de la vida moderna. Sin embargo, la más grande dificultad con la que se enfrenta su universalización es la polarización cultural del mundo occidental a favor de la lógica tradicional del ¨1 ¨ y el ¨ 0 ¨ y de los modelos matemáticos lineales. Estos últimos son adecuados para describir procesos simples. No obstante, la mayoría de los procesos del mundo real son no lineales y resultan demasiado complejos para ser modelados matemáticamente. Este es el tipo de problemas que deben resolverse por lógica difusa

3.2 Conceptos y reglas básicas

Para comprender intuitivamente el concepto de lógica difusa, consideremos como ejemplo un florero A con diez rosas rojas y un florero B con diez orquídeas. En los términos absolutos de la lógica formal, proposiciones del tipo A es un florero de orquídeas y B no es un florero de rosas pueden negarse o afirmarse categóricamente sin crear confusión. Suponga que en el florero A se cambian dos rosas por dos orquídeas. Evidentemente, ahora no podemos afirmar con la misma determinación que A es un florero de rosas porque también contiene orquídeas. En este caso, resulta más preciso decir que A es mayormente un florero de rosas o que A es particularmente un florero de orquídeas. Con este tipo de ambigüedades presentadas en la descripción de la realidad es que trabaja la lógica difusa.

En lógica difusa, los diferentes valores que pueden adoptar una variable del mundo real se subdividen o clasifican en grupos y cada valor dentro de un grupo se le asigna una cuota de pertenencia al mismo denominada grado de asociación o membresía. Esta última se designa generalmente como y puede adoptar valores entre 0 y 1. Los grupos se denominan μ conjuntos difusos (ajustes difusos ó fuzzy sets).

En la figura 1 se muestra un ejemplo de representación en forma de conjuntos difusos, de temperatura de un recinto. Los valores de temperatura medidos se grafican sobre el eje x y el grado de membresía de cada uno, asignado por experiencia práctica, sobre el eje y. El resultado es una serie de formas convexas, generalmente triángulos o trapecios, llamadas funciones de membresía o de pertenencias.

En este caso, la información de temperatura se divide en nueve conjuntos difusos, clasificados desde severamente congelados hasta muy caliente. El rango de valores de cada conjunto determina la base de la respectiva forma y los valores para los cuales la situación representada se cumple en un cien por ciento determinan el techo o vértice. En el caso de la figura 1, por ejemplo, todas las temperaturas entre 17 ºC y 24ºC tienen un determinado grado de pertenencia con respecto al conjunto –ambiente- , pero el recinto está verdaderamente a la temperatura ambiente (μ=1,0) entre 20ºC y 21 ºC.

Las formas triangulares y trapezoidales son las más utilizadas en aplicaciones de control difuso debido a que simplifican la manipulación aritmética y representan adecuadamente la experiencia humana. Sin embargo, son también posibles otras formas (figura 2), incluyendo la función rectangular utilizada por la lógica tradicional.

Todos los valores de una representación de lógica difusa deben poseer un μ definido. Aunque no es obligatorio, se recomienda permitir que las fronteras o flancos entre dos formas adyacentes se traslapen con un grado de pertenencia combinado del 100 % (μ=1), como sucede con los puntos 21 ºC y 24 ºC del ejemplo anterior, comunes a los conjuntos –ambiente- y –cálido- .Las formas no convexas son inválidas.

Modelo matemático del motor DC

El motor DC con excitación independiente es descrito mediante las ecuaciones siguientes:

$KF\omega p(t) = – Raia(t) – La[dia(t)/dt] + Vt(t)$ (1)

$KFia(t) = J[d\omega p(t)/dt] + B\omega p(t) + TL(t)$ (2)

donde,

$\omega p(t)$ – velocidad del rotor (rad/s)

$Vt(t)$ – Tensión terminal (V)

$ia(t)$ – Corriente de armadura (A)

$TL(t)$ – Torque de carga (Nm)

J – Inercia del rotor (Nm2)

KF - torque & back emf constante (NmA-1)

B – Coeficiente de fricción viscoso (Nms)

Ra – Resistencia de armadura (Ω)

La – Inductancia de armadura (H)

Partiendo de éstas ecuaciones podemos crear el modelo matemático del motor DC. El modelo es presentado en la figura 3.3.

donde,

Ta –Constante de tiempo del circuito y armadura del motor and Ta=La/Ra (s)

Tm – Constante de tiempo mecánica del motor Tm=J/B (s)

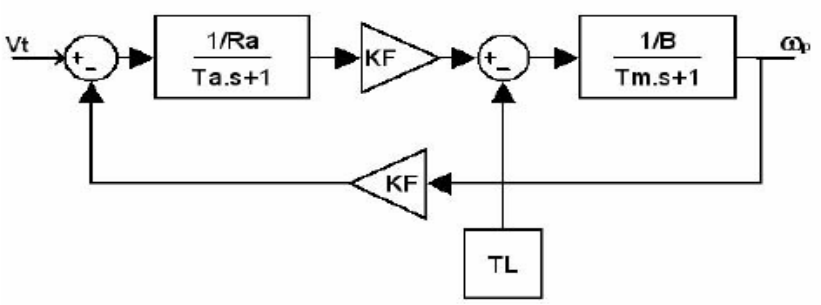


Figure 3.3. Modelo matemático de un motor CD independiente

3.3 Diseño del controlador difuso a través de la herramienta fuzzy de MATLAB

Se diseñará un controlador difuso a través de la herramienta Fuzzy de Matlab, para una planta que corresponderá a un motor DC de 5 Voltios. El sistema sobre el cual se basará el diseño del controlador difuso, consta del bloque correspondiente al controlador, el bloque de la planta, el bloque del sensor(realimentación entre la salida y el sumador a la entrada) y la referencia o entrada del sistema.

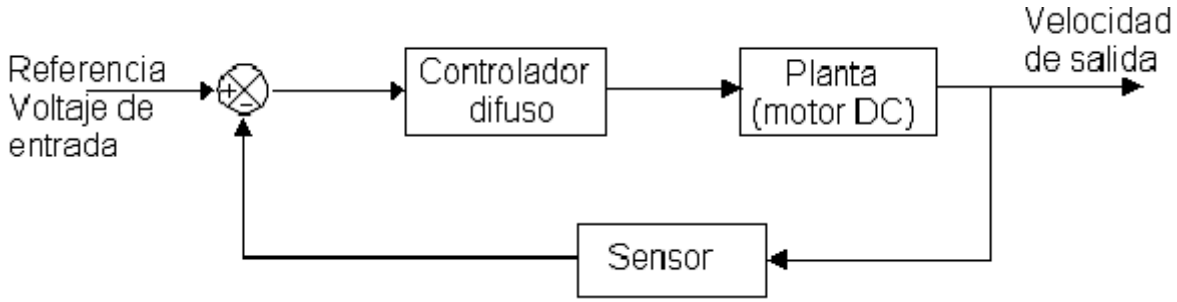


Figura 3.4. Diagrama de bloques del sistema completo

Designación de las entradas y salidas del controlador difuso en la ventana de FIS de MATLAB:

- Tamaño:
- Peso:
- Control del ángulo

Diseño del controlador difuso

Se diseñara el controlador difuso tomando como base tanto para las dos entradas como para la salida 6 conjuntos de ajustes difusos que representarán diferentes estados. Se obtendrá finalmente el controlador difuso y de determinará si es viable implementarlo de acuerdo a la simulación obtenida en Simulink.

Definición de los conjuntos difusos de las entradas (Vin y sensor) y la salida (Vout)

- Conjuntos de ajustes difusos para la variable tamaño:

º MG: recipiente muy grande

º G: recipiente grande

º N: recipiente normal

º P: recipiente pequeño

º MP: recipiente muy pequeño

- Conjuntos de ajustes difusos para la variable peso:

º P: recipiente pesado

º G: recipiente peso normal

º N: recipiente poco pesado

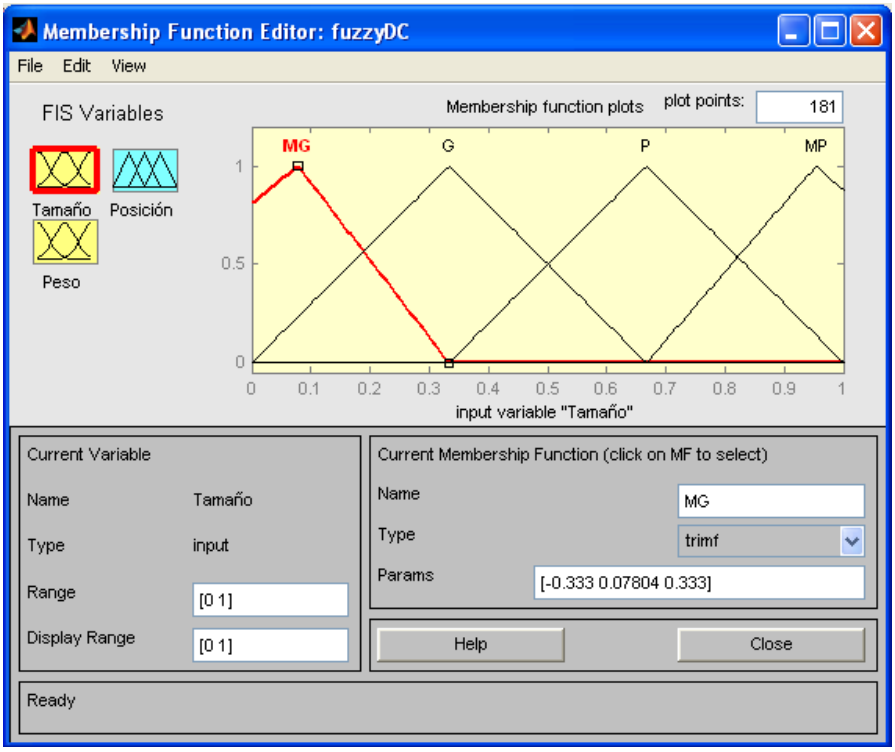


Figura 3.5. Conjunto de ajuste difuso para la variable tamaño

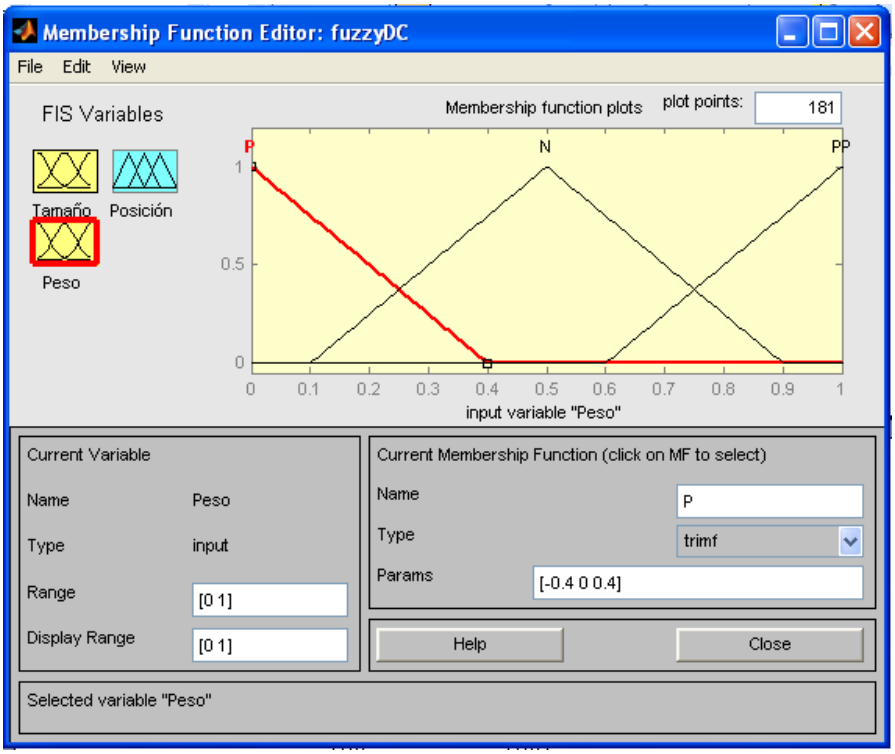


Figura 3.6. Conjunto de ajuste difuso para la variable peso

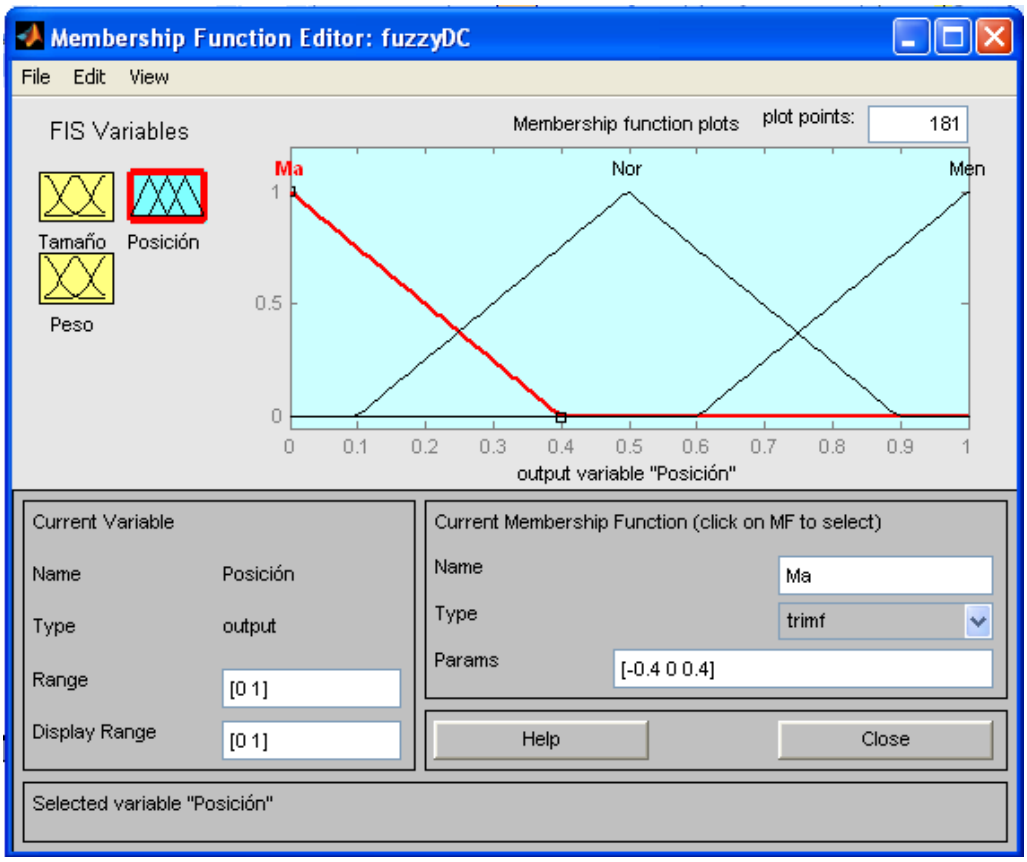


Figura 3.7. Conjunto de ajuste difuso para la salida posición del ángulo

- *Conjuntos difusos para la variable posición:*

º Ma: el motor debe moverse en un mayor ángulo

º Nor: el motor debe moverse en un ángulo normal

º Men: el motor debe moverse en un menor ángulo

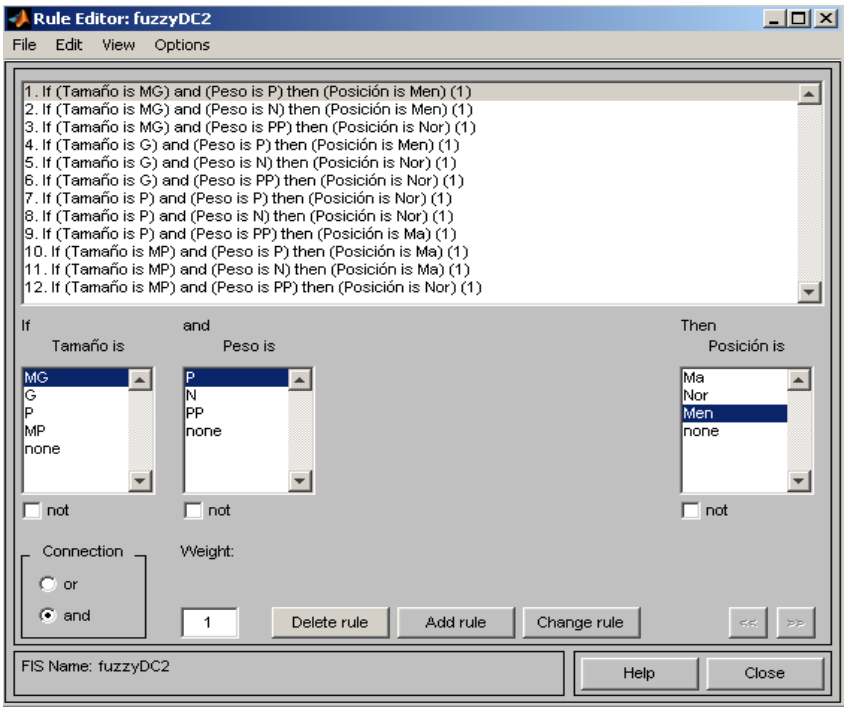


Figura 3.8. Visualización de las reglas representadas por medio de los conjuntos.

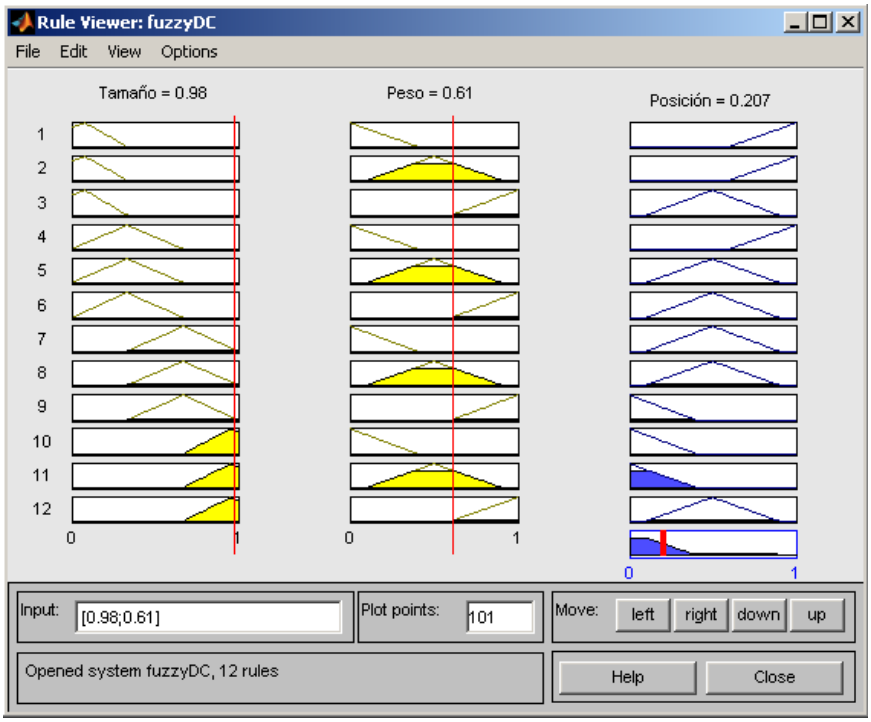


Figura 3.9. Visualización de las reglas representadas por medio de los conjuntos.

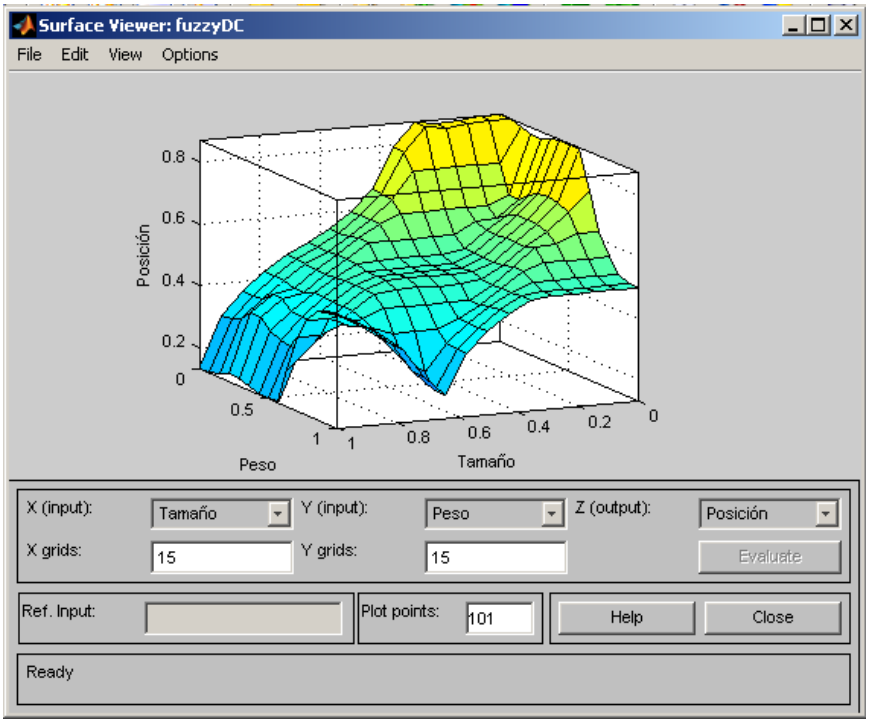


Figura 3.10. Superficie correspondiente al control difuso desarrollado.

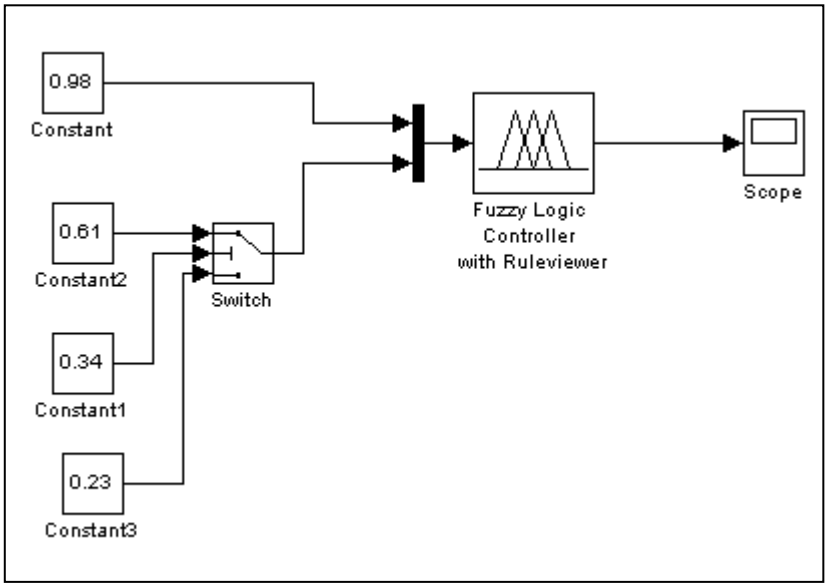


Figura 3.11 Simulación del controlador difuso en Simulink.

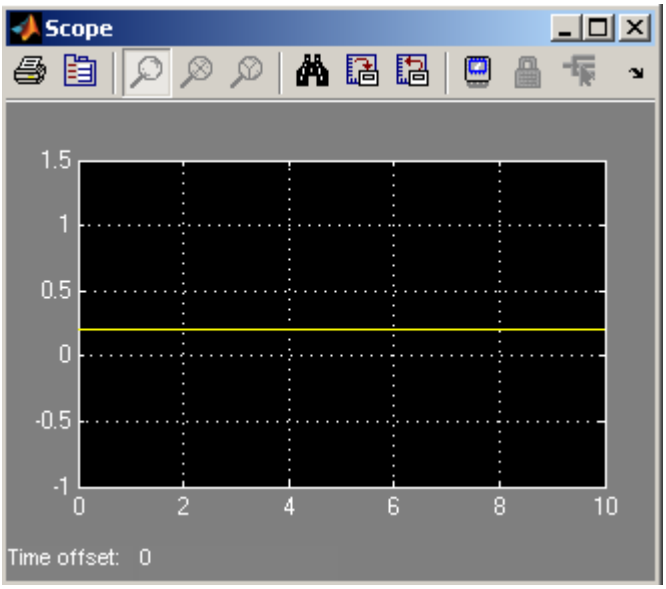


Figura 3.12 Respuesta ante el controlador difuso

3.4 Implementación de una red neuronal básica

Un controlador difuso como el propuesto, posee ventajas en el control de mando de una máquina (en este caso un motor paso a paso), la implementación de la inteligencia artificial utilizando lógica fuzzy es mucho más eficiente si aplicamos una red neuronal básica que logre aprender del propio sistema utilizando patrones de referencia.

En este caso particular vamos a determinar el tipo de fluido que se va ser procesado y envasado utilizando un espectro de colores, cada densidad del líquido posee un color característico, el cual va a ser utilizado a la hora de crear la red neuronal básica, esta identificará el tipo de fluido y definirá en cual envase debe ser envasado.

Se presenta en para este caso el siguiente software

```
% --- Explorando volúmenes de fluidos con planos ---

% -- Ejemplo de flujo de datos de fluidos --

% -- Dibuja un plano tipo Slice --

% Dibuja el plano rotado, se ajusta por la propiedad FaceColor el intercepto
% para colorear la figura usando el comando colormap, y ajusta la propiedad
% EdgeColor. Incrementa DiffuseStrength a .8 para hacer el plano más brillante
% después de adicionar la luz de fuente.

[x,y,z,v] = flow;

xmin = min(x(:));
ymin = min(y(:));
zmin = min(z(:));

xmax = max(x(:));
ymax = max(y(:));
zmax = max(z(:));

hslice = surf(linspace(xmin,xmax,100),...
    linspace(ymin,ymax,100),...
    zeros(100));

rotate(hslice,[-1,0,0],-45)
```

```
xd = get(hslice,'XData');
yd = get(hslice,'YData');
zd = get(hslice,'ZData');

h = slice(x,y,z,v,xd,yd,zd);
set(h,'FaceColor','interp',...
    'EdgeColor','none',...
    'DiffuseStrength',.8)

% ajusta y adiciona tres ortogonales con sus valores xmax, ymax, zmin
% ewn el contexto del primer plano, con ángulo.

hold on
hx = slice(x,y,z,v,xmax,[],[]);
set(hx,'FaceColor','interp','EdgeColor','none')

hy = slice(x,y,z,v,[],ymax,[]);
set(hy,'FaceColor','interp','EdgeColor','none')

hz = slice(x,y,z,v,[],[],zmin);
set(hz,'FaceColor','interp','EdgeColor','none')

pause

% Define la vista con el comando View

% Ajusta en proporciones correcta el volumen al display, ajusta los datos
% con radio [1,1,1] (daspect). Ajusta el eje alrededor del volumen (axis) y retorna
% el objeto a 3-D.

% Para poner el largo del volumen usa el comando (camzoom).
% Para poner la proyección del volumen usa el comando (camproj).

daspect([1,1,1])
axis tight
box on
view(-38.5,16)
camzoom(1.4)
camproj perspective
```

```
pause

% Para adicionarle luz y colores específicos al volumen

lightangle(-45,45)
colormap (jet(10))
set(gcf,'Renderer','zbuffer')

pause

% Para modificar el mapa de colores.

colormap (flipud(jet(10)))

% Para ajustar los límites de colores

caxis([-5,2.4832])

% Para adicionar una barra de colores y suministrarlo a través de una tecla

colorbar('horiz')

pause

% --- RED BASICA RADIAL PARA IDENTIFICAR COLOR Y DENSIDAD ---
% Se definen 21 valores de entradas P y asociados a la salida targets T.

P = -1:0.1:2;

% --- Para enconrar matriz del colormap ---
% T = colormap (flipud(jet(10)))
 T = [1.0 0.3333 0 ...
    1.0000 0.6667 0 ...
    1.0000 1.0000 0 ...
    0.6667 1.0000 0.3333 ...
    0.3333 1.0000 0.6667 ...
    0 1.0000 1.0000 ...
    0 0.6667 1.0000 ...
```

```
    0 0.3333 1.0000 ...
    0 0 1.0000 ...
    0 0 0.6667 0.55];
        plot(P,T,'r--'),grid
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');

% La función NEWRB crea rápidamente una red básica radial, la cual aproxima
% la función por P y T.

% Se ajustan y entrenan los targets, NEWRB toma dos argumentos,
% el 'sum-squared error goal' y la 'spread constant'. El spread de la
% neurona básica radial B se ajusta a un valor más grande.

eg = 0.02;                          % sum-squared error goal
sc = 127;                           % spread constant
net = newrb(P,T,eg,sc);

%NEWRB, neurons = 0, MSE = 0.176192

% La red básica radial no puede atender a todas las neuronas NEWRB,
% es por eso que no puede generar muchas respuestas al mismo tiempo.

Y = sim(net,P);
hold on;
plot(P,Y);
legend('P','Y',4);
hold off;
```

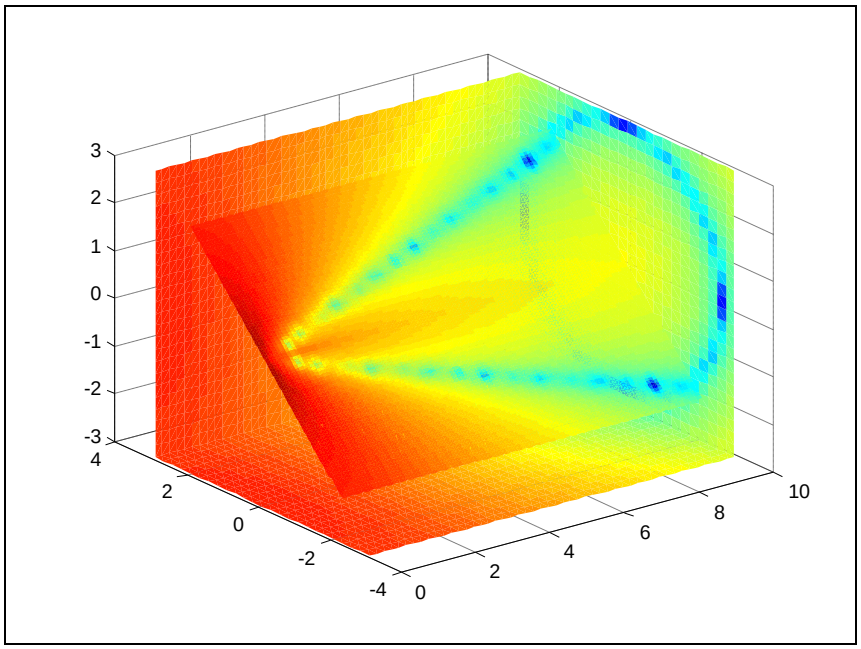


Figura 3.13 Flujo de datos de fluidos representado en un espectro de colores

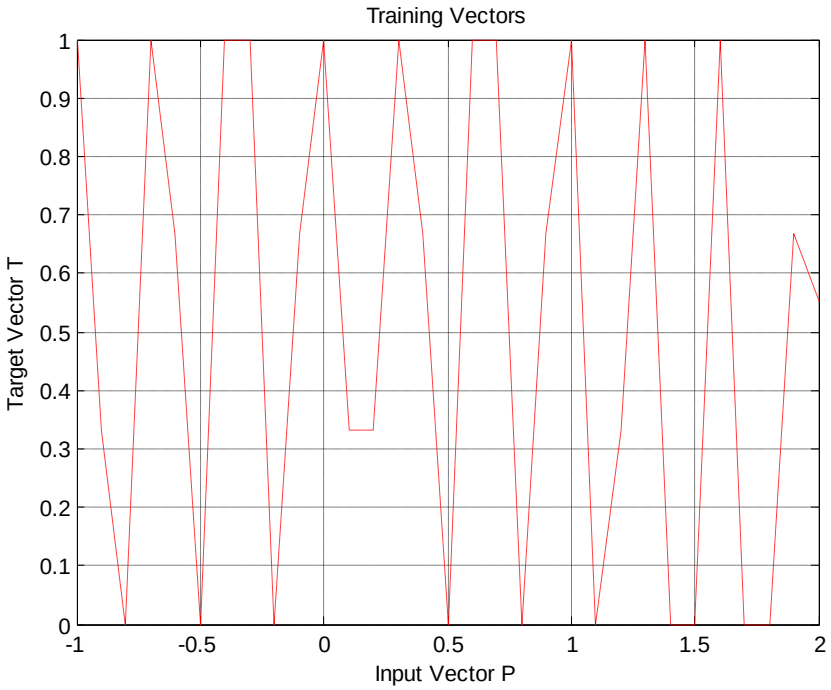


Figura 3.14. Entrenamientos de los vectores de entrada P y T,

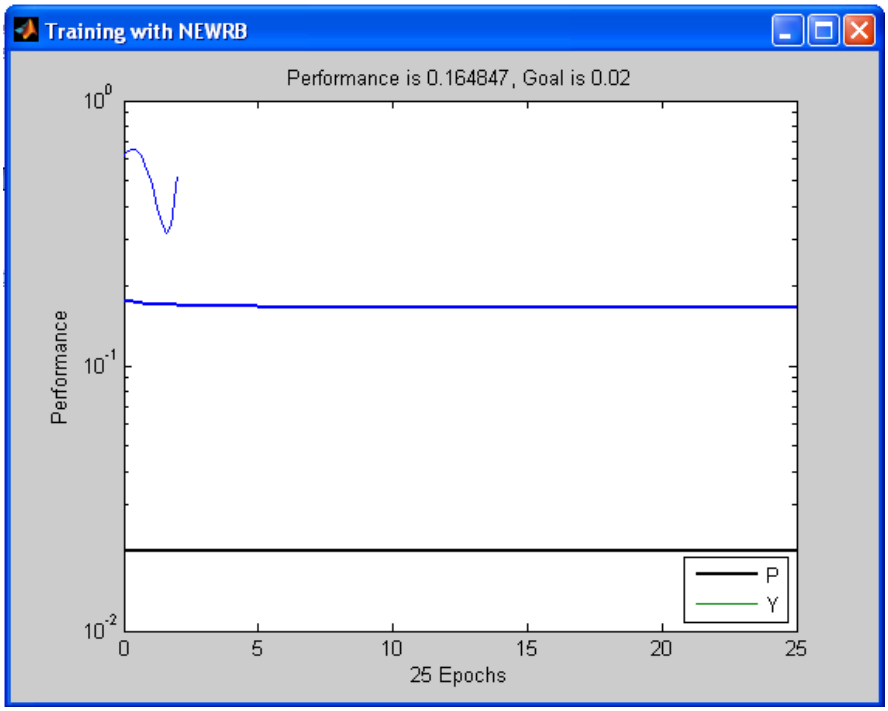


Figura 3.15. Entrenamiento por épocas

3.5 Conclusiones del capítulo III

- La lógica difusa es una herramienta que puede ser utilizada en el mundo del control en métodos de mando de toda clase de dispositivos eléctricos y electrónicos.

- La lógica difusa es un método de razonamiento estadístico basado en la teoría de conjunto, que permite especificar los problemas de control sin necesidad de recurrir siempre a modelos matemáticos y con un nivel de abstracción mucho más elevado.

- Ante la necesidad de controlar un sistema de llenado de recipientes con diferentes líquidos o fluidos, usando medios electrónicos existentes con la implementación de la técnica neuro-fuzzy, es posible diseñar un supervisor difuso que responda a necesidades específicas.

CAPÍTULO IV

SOFTWARE ASISTENTE PARA INTERFACE DE CONTROL DE INSTRUMENTOS

4.1 Introducción

El control automático ha desempeñado un papel vital en el avance de la ingeniería y la ciencia. Además de su gran importancia en los sistemas que exigen la precisión de espacios y de posición como los accionamientos de servomotores destinados a la apertura y cierre de válvulas, mecanismos de manufacturas, bandas transportadoras, de vehículos espaciales, de guiado de misiles, robóticos y análogos, el control automático se ha convertido en una parte importante e integral de los procesos modernos industriales y de fabricación. El control automático tiene muchas aplicaciones en el quehacer de la humanidad y está estrechamente relacionado con el progreso de nuestra sociedad.

Como los avances en la teoría y la práctica del control automático proporcionan los medios para conseguir un comportamiento óptimo de los sistemas dinámicos, mejorar la productividad, simplificar el trabajo de muchas operaciones manuales repetitivas y rutinarias, así como de otras actividades, la mayoría de los ingenieros y científicos deben tener un buen conocimiento de este campo.

4.2 Motores paso a paso (step- step).

El motor de pasos es una forma de motor de CD que está diseñado para girar un determinado número de grados por cada pulso eléctrico que se aplique a su unidad de control. Los tamaños de los pasos pueden ir desde menos de un grado hasta 15° o más. Una ventaja notable del motor de pasos es su compatibilidad con los sistemas electrónicos digitales. Esos sistemas son cada vez más comunes en una amplia gama de aplicaciones y al mismo tiempo se están fabricando con mayor potencia y menor costo. En muchas aplicaciones se puede obtener información sobre la posición tan solo con mantener una cuenta de los pulsos que se mandan al rotor, y no se necesitan sensores de posición ni control por retroalimentación. Los motores de pasos no tienen escobillas ni conmutador mecánico. En su lugar, la acción de conmutación necesaria para la función del motor de CD es lograda por transistores externos. Es más, el rotor no tiene devanado de armadura, simplemente es una colección de imanes permanentes salientes. Existen dos divisiones fundamentales:

- Motores paso a paso de rotor de disco o rotor IP
- Motores paso a paso de reluctancia variable

También es digno de nombrarse el motor de pasos hibrido que combina características de los motores de reluctancia variable y de imán permanente.

4.3 Adquisición de datos por el puerto paralelo de la PC

El puerto paralelo, usado comúnmente para comunicar el PC con la impresora, lee y escribe todos sus pines simultáneamente o *en "paralelo"*. De ahí su nombre.

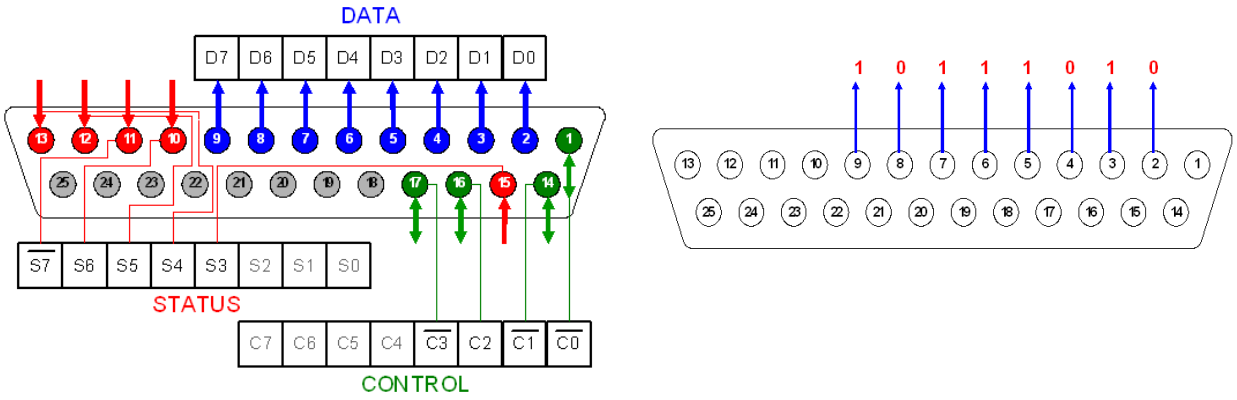


Figura 4.1. a),b). Esquema y distribución de los pines de un puerto paralelo

Buses de información del puerto paralelo

- 8 Pines de ***salida*** en el Bus de datos DATA (Llamado puerto *0* en Matlab)
- 5 Pines de ***entrada*** en el Bus de Estado STATUS (Llamado puerto *1* en Matlab)
- 4 Pines de ***entrada ó salida*** en el Bus de CONTROL (Llamado puerto *2* en Matlab)
- Los 8 pines restantes son Tierra " – " *(18 al 25)*

Características operacionales del puerto paralelo

El *Puerto Paralelo* está compuesto por 17 líneas (pines) de señal y ocho líneas de tierra Opera en los ***niveles digitales*** convencionales, donde 0 - 0,5 voltios representan **0** (OFF), 3.8 - 5 voltios representan **1** (ON). Diferente de la comunicación serial donde ON es representado por voltajes entre -3 y -25 voltios, y OFF por señales entre 3 y 25 voltios. *Entonces...* no se puede hacer conexiones *serial-paralelo* sin un circuito electrónico de acople.

Debido a que el Puerto Paralelo trabaja en los niveles digitales convencionales, es posible (y fácil) conectarlo a chips lógicos TTL.

Para que un circuito externo se acople perfectamente al puerto paralelo, sólo hace falta unir la tierra del circuito con la tierra del puerto (pines 17 al 25).
Para mayor seguridad, es necesario aislar los pines del puerto paralelo, por medio de compuertas inversoras, buffer u opto aisladores.

4.4 Manejo de salidas digitales (BUS de datos "DATA")

Las salidas del *puerto paralelo* nos permiten *enviar información* en forma de señales digitales, al entorno *fuera del computador*. Por lo tanto, podemos convertir variables propias de un programa (software) en *acciones físicas tangibles en el mundo real por medio de un circuito conectado al puerto* y de esta manera lograr verdadera *interacción entre un proceso físico y un programa de computador*.

- El bus de datos o **DATA** puede enviar un byte (8 bits) al tiempo (transmisión paralela).
- Las ocho líneas de datos están representadas físicamente por los pines 2 al 9, llamados D0,D1..D7. Donde D0 es el bit menos significativo = 2^0 y $D^7 = 2^7$.

Para enviar información a través del puerto paralelo, antes que nada debemos **definir una variable** u objeto (dependiendo del lenguaje de programación) en el cual se almacene la dirección dicho puerto, para que el programa conozca desde el principio la ubicación exacta de donde debe leer y escribir información.
Para nuestro caso que trabajaremos en ***MATLAB***, lo primero es definir objeto llamado ***"dio"*** (por digital input/output) en el cual almacenaremos la dirección del puerto paralelo.
Para este fin, nos apoyaremos en la función "***digitalio***", que es la encargada de localizar automáticamente los dispositivos de entrada y salida digital, definir su dirección y características. Digitemos en la ventana "*command Window*" de ***MATLAB***, la instrucción:

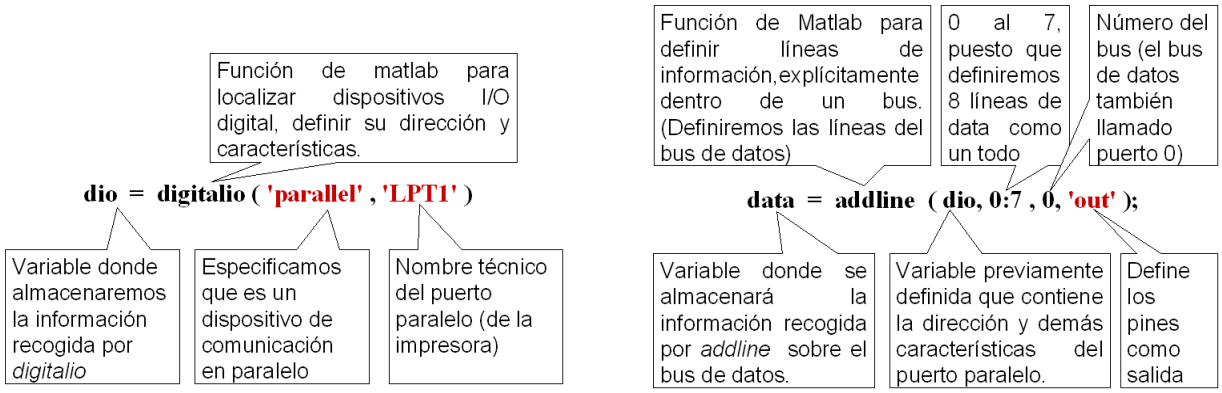
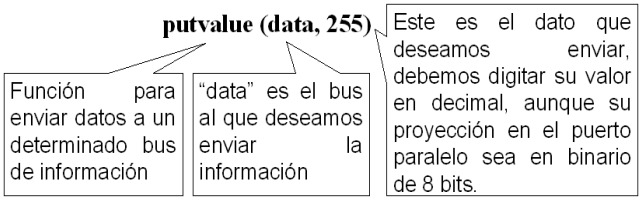


Figura 4.2. a),b). Instrucciones para el envio de información por el PP

Ya conocida la dirección del Puerto Paralelo, el siguiente paso es definir por separado la dirección de cada uno de los buses de comunicación. Para ello se usa la función *addline* y se almacena cada dirección en una variable del mismo nombre del bus ya sea Data, Status y Control. Definido el bus de datos, lo usaremos para *enviar información en forma digital*. Para ello, se usa la función:



4.4 Elaboración de la interface para el control de un motor paso a paso.

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. La característica de estos motores es poder moverlos un paso a la vez por cada pulso que se le aplique.

MATLAB posee comandos sencillos para el control del puerto paralelo. Basta con crear la entrada digital del puerto y asignar que pines son de entrada y cuáles de salida.

```
ent= digitalio('parallel','LPT1');
dato= addline(ent,0:4,'out');
putvalue(dato,2);
```

El código anterior crea la entrada digital del puerto paralelo, asigna los pines 2 a 5 como salidas y escribe el valor decimal 2 (MATLAB realiza automáticamente la conversión a binario) en el puerto.

Las condiciones iniciales del programa colocan a cero los pines de puerto paralelo:

```
allcero= digitalio('parallel','LPT1');
dato= addline(allcero,0:3,'out');
putvalue(dato,0);
```

Como se puede ver en la **Figura 4.1**, lo único que se debe programar en la interfaz gráfica es la captura del retardo, la dirección de giro y el encendido-apagado del motor.

Figura 4.1. Interface para el control del motor paso a paso

La mayor parte del código se programa en el toggle button ON-OFF, cuyo campo Tag es state. Sin embargo, es necesario un código adicional para el texto del botón interruptor de la dirección.

```
f=get(handles.direction,'Value');
if f==1
set(handles.direction,'String','DIRECTION "L"');
else
set(handles.direction,'String','DIRECTION "R"');
end
```

La programación del botón de encendido sería la siguiente:

```
d=get(hObject,'Value');
if d==1
    set(handles.state,'String','ON');
    diego=digitalio('parallel','LPT1');
    dato=addline(diego,0:3,'out');
    g=1;
    while g
        e=get(handles.direction,'Value');
        if e==0
            mov=[3 6 12 9];
        else
            mov=[9 12 6 3];
        end
        delay=str2double(get(handles.speed,'String'))*1e-3;
        if delay<0 ||isnan(delay)
            errordlg('Time out of range','ERROR');
            delay=500;
            set(handles.speed,'String',500);
            set(handles.state,'String','OFF');
            set(handles.state,'Value',0);
            break;
        end
```

```
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(1));
        pause(delay);
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(2));
        pause(delay);
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(3));
        pause(delay);
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(4));
        pause(delay);
    end
else
    set(handles.state,'String','OFF');  end
```

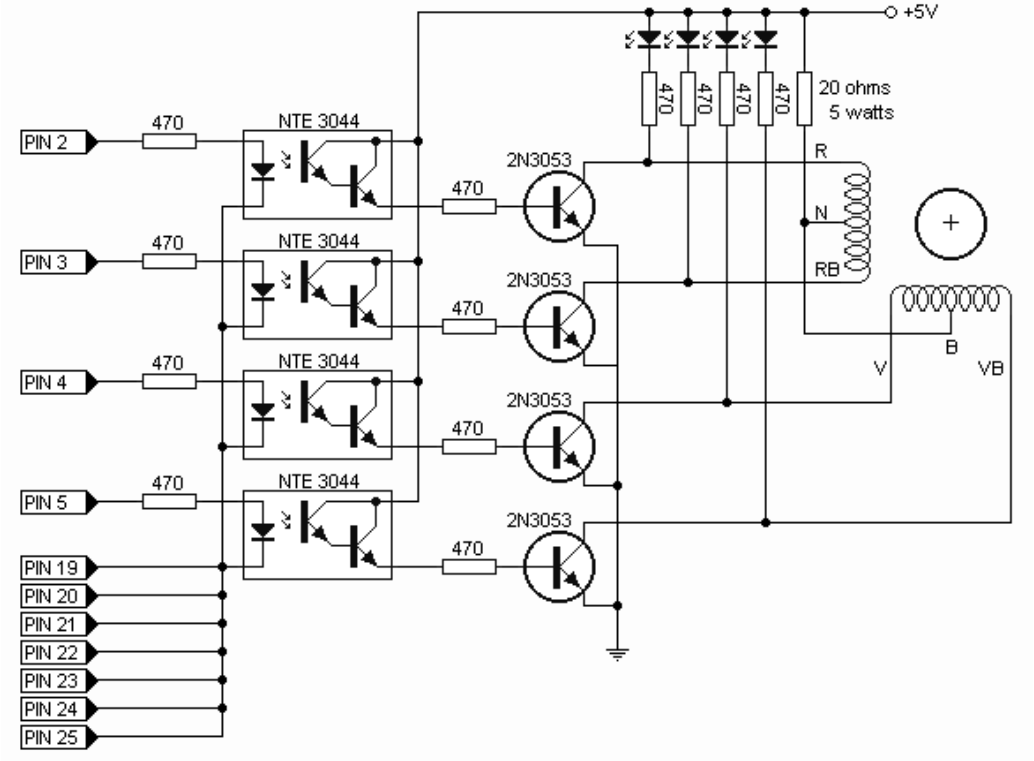


Figura 4.2. Esquema de conexión del motor

Asimismo esta función controla el número de pasos, la velocidad y dirección de giro:

```
function motor(step,delay,direction)

warning off

if nargin==1

    delay=1;direction=1;

elseif nargin==2

    direction=1;

end

if isnan(step)||step<0

    error('Step value must be positive integer');

elseif direction~=1 && direction~=2
```

```
    error('Direction options: 1 or 2')
elseif isnan(delay)||delay<0
    error('Delay value must be positive integer')
end
step=ceil(step);
inout=digitalio('parallel','LPT1');
dato=addline(inout,0:3,'out');
if direction ==1
    sent=[3 6 12 9];
else
    sent=[9 12 6 3];
end
m=1;
for n=1:step
    putvalue(dato,sent(m));
    pause(delay);
    m=m+1;
    if m>4
        m=1;
    end
end
```

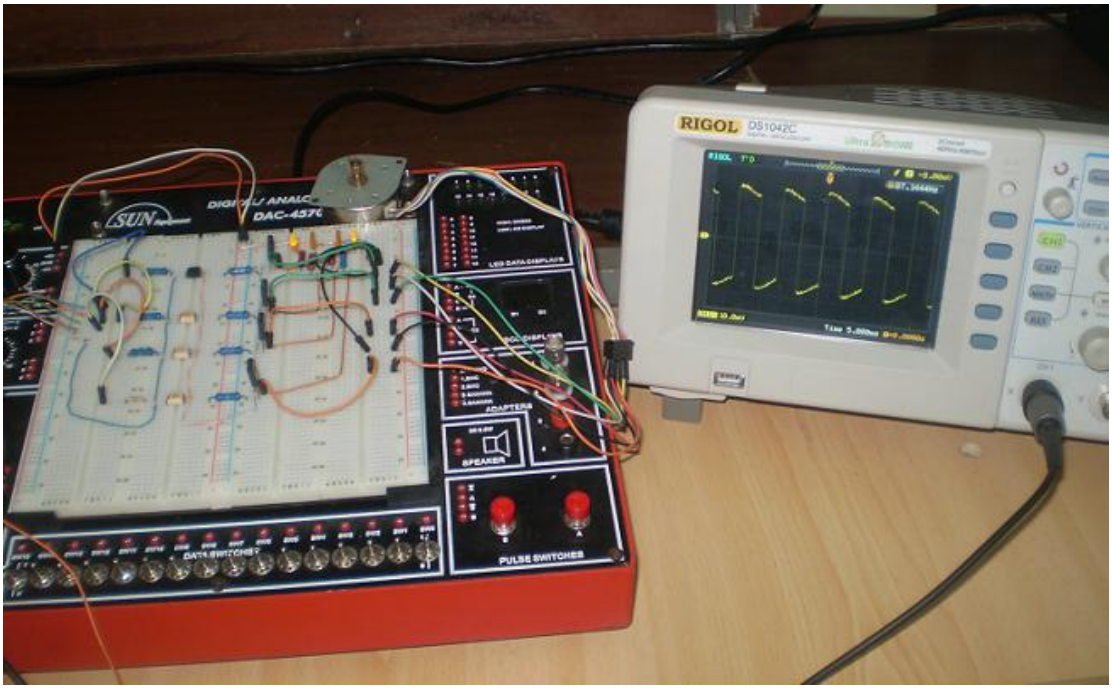


Figura 4.3 Montaje experimental para el control del motor paso a paso

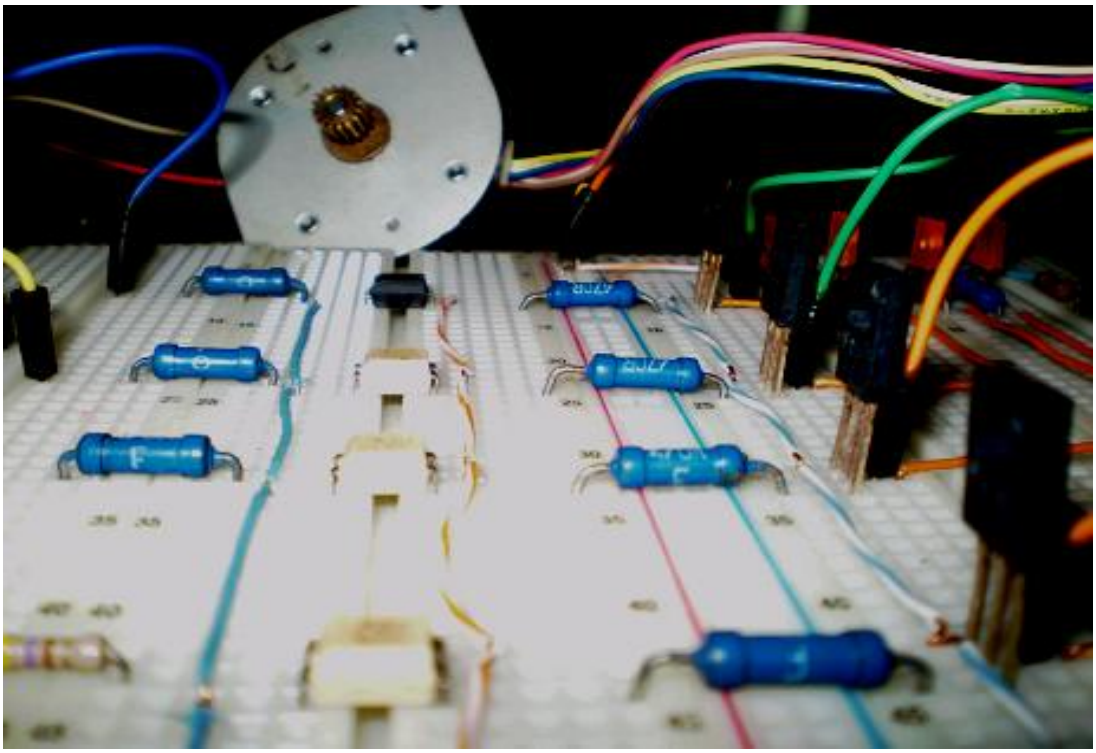


Figura 4.4. Circuito de control para el mando del motor paso a paso montado en un protoboard

4.5 Propuesta de aplicación para un sistema de control inteligente utilizando un motor paso a paso.

En la actualidad, se ha despertado un gran interés por la creación de aplicaciones que permitan tener un nivel conversacional con la máquina y conocimientos de expertos en un área determinada. La creación de esta aplicación está enmarcada dentro del área de la Inteligencia Artificial y tiene como tarea fundamental aumentar la calidad y la eficiencia de una planta implementando mandos de control automático e inteligentes.

En este caso se ha simulado llenado de envases de diferente tamaño, y con diferentes contenidos a llenar, el sistema está compuesto de la siguiente manera:

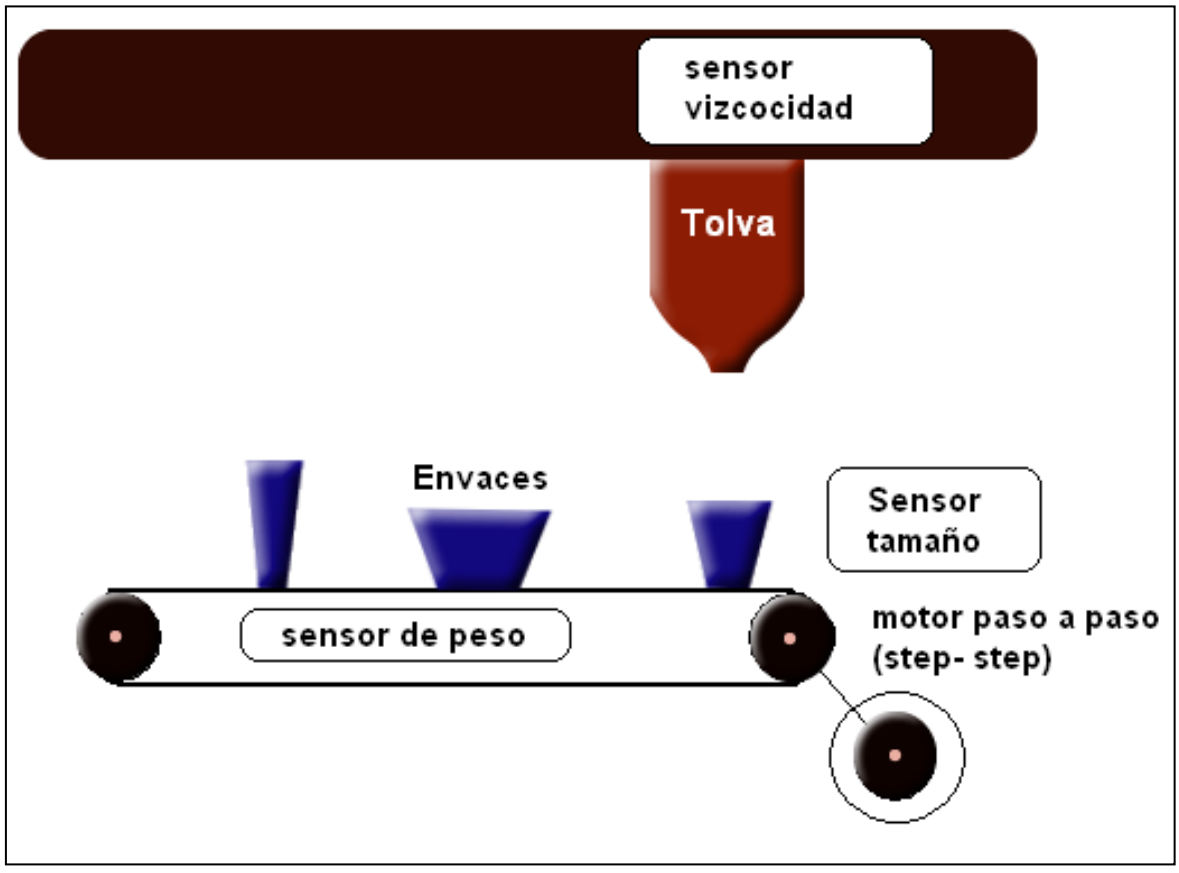


Figura 4.3. Diagrama del sistema a diseñar para el control automático

4.6 Conclusiones del capítulo IV

- Se implementa el hardware de control de posición para un motor paso a paso. Lo asiste una interface desde el MATLAB, que garantiza hasta 360 grados.

- Se verificó la comunicación entre el PC, y la tarjeta de control, utilizando las bondades del GUI del MATLAB.

- Para la comunicación se utilizó el puerto paralelo del PC con un protocolo de tipo sincrónico, con 4 líneas de transferencia.

CAPÍTULO V

EVALUACIÓN TÉCNICO- ECONÓMICO DEL TRABAJO

5.1 Introducción

La aplicación y el empleo de técnicas inteligentes en los accionamientos de instalaciones industriales contribuyen a mejorar la eficiencia y el entorno de trabajo en estos sistemas.

Por otra parte, el software para la adquisición de datos y monitoreo de procesos industriales, trae consigo une serie de ventajas con respecto a los sistemas de control en los que se utiliza mucho la mano del hombre, desde este punto de vista, la aplicación de software SCADA reduce mucho los costos por mano de obra, brinda ventajas a la hora de evaluar el estado del proceso, gracias a la utilidad de históricos que se presenta, y a su vez se puede asignar tareas de mantenimiento a el equipo, y así mismo se puede tener una idea sobre la vida útil de cada uno de los accesorios, como también la pérdida de equipos por mal funcionamiento.

Desde el punto de vista económico la implementación de estos sistemas inteligentes asegura desde un inicio el desempeño óptimo del proceso, pero no se puede dejar de lado, que el montaje de una estación con todos los equipos vinculados con el software, de principio resultará una inversión fuerte, por la necesidad de adquisición de transductores, PCs, cables de comunicación; prácticamente se necesita una inversión, que en nuestro caso se ha simplificado por el uso de un esquema sencillo de control de posición a través de motor paso a paso, pero operable, para el desarrollo de un sistema de llenado de recipientes con diferentes líquidos o fluidos, usando medios electrónicos existentes con la implementación de la técnica neuro-fuzzy.

De todas formas existe un gasto inicial, que resulta insignificante en este caso, porque la mayor parte de los medios existen en el laboratorio.

5.2 Costos en la explotación del accionamiento industrial

Estos están asociados con los gastos de energía y el efecto de la implementación de un sistema de precisión que mejora la eficiencia del llenado de los recipientes y minimiza el tiempo de operación de la instalación. Consecuentemente, el esquema electrónico propicia el aprendizaje del tipo de fluido y ajusta el desplazamiento de los recipientes de acuerdo a las dimensiones de los mismos y el tipo de fluido. Vamos a valorar la

diferencia entre los gastos de energía cuando no existe el sistema supervisor y de control y con éste.

Por otra parte va ser significativo también el efecto de humanización de las tareas con el uso de la informática. Este trabajo brinda las bases para el uso de sistemas experto en otros accionamientos con características análogas.

5.3 Gastos en inversiones

Estos incluyen los gastos de la instrumentación de campo, en accesorios del PC y el software.

$$CTI = \Sigma CIC + \Sigma CV$$

Donde:

ΣCIC = Costo total de la instrumentación de campo

ΣCV = Costo total de equipos y medios de explotación.

Costo total general sería:

$$CTG = CTI + EPA$$

donde:

EPA – Estimado de la parte de la parte automática (15 al 20% del costo de la instrumentación *CTI*)

5.4 Impacto económico de la interface para la supervisión

En la tabla vamos a reflejar un resumen de las principales incidencias con el uso del esquema de la interface del supervisor en las tareas y explotación del transportador para llenado de recipientes con diversos fluidos.

CONTRIBUCIÓN DE LOS LABORATORIOS DEL ISMM			
Contribuciones	**Cantidad**	**Valor unitario**	**Valor Total (USD)**
Bibliografía	15	-	500.00
Autómata programable Modicon	1	285.00	285.00
Instalación de prueba	1	120.00	120.00
Software	varios	1200.00	1 200.00
Hubs	1	70.00	70.00
Motor paso paso (step-step)	1	-	58.00
Motor de inducción 10 kW, 220 V.	1	-	350.00
Banda transportadora de 5 metros	1	-	115.00
Sensores de peso	4	750.00	3000.00
Recipientes de fluidos	12	-	120.00
Tolva de descarga	1	-	45.00
Cronómetro	2	-	7.00
Gasto de energía eléctrica	kW.h		48 horas mensuales($5.76)
Total			**5875.76**

Tabla 5.1. Contribución de los laboratorios del ISMM

CONTRIBUCION DEL ÁREA DE INTELIGENCIA (En USD)			
Contribuciones	**Cantidad**	**Valor unitario**	Valor Total (USD)
Servidor de red (PC)	1	1 500.00	1 500.00
Driver de instrumentos electrónicos	varios	450.00-	450.00
Manómetro fluxómetro	varios	-	550.00
Medidor de Viscosidad	varios	-	700.00
Tarjeta Procesadora de Señales DSP	1	550.00	550.00
Osciloscopio digital portable	1	1 083.23/25	43.32
Sensores ópticos medidores de flujo magnético, velocidad, temperatura, posición y vibraciones.	**varios**	**-**	**1 250.00**
Chip neuro-fuzzy	**1**	**13 750.00**	**13 750.00**
Software más capacitación de aplicaciones más diseño de sistema	varios	9 800.00	9 800.00
Total			**28593.32**

Tabla 5.2. Contribución del área de inteligencia

Se realizó un análisis contable de los equipos existentes reales en el laboratorio de circuito y accionamientos eléctricos, y asimismo se contabilizó el precio de los equipos que había que comprar. La diferencia de ambos nos da el aporte en USD que ofrece este humilde trabajo con la implementación del supervisor neuro-fuzzy.

5.5 Conclusiones del capítulo V

Podemos concluir que para un monto de 7240.76 USD, el cual corresponde a los equipos que se deben comprar, y otro de 27228.32 USD que corresponde a los equipos que no se deben comprar pues existen en nuestro laboratorio, con el uso del supervisor se tiene un ahorro de 19987.62 USD, con la correspondiente mejorías de las condiciones de trabajo de los accionamientos eléctricos y el ahorro de energía.

Conclusiones Generales

- Con el empleo de los recursos del MATLAB se construyó una interface que permite la comunicación entre el PC y una tarjeta de adquisición y supervisión de datos destinada para el control de posición de un accionamiento.
- Se elaboró un software que atiende la interface PC tarjeta supervisora para el control de posición usando un motor paso a paso, que resuelve la automatización del proceso de llenado de recipientes en una banda transportadora.
- Se desarrolló un esquema supervisor y de control que mejora las condiciones de trabajo de un accionamiento para el llenado de recipientes utilizando sensores y la técnica inteligente de redes neuronales y lógica fuzzy combinada.
- Este estudio corrobora la efectividad de los sistemas inteligentes en la explotación de accionamientos de procesos.

Recomendaciones

- Continuar el estudio de estos sistemas de accionamientos eléctricos controlados usando los recursos de sistemas inteligentes pero implementando la comunicación inalámbrica.
- Utilizar el esquema de supervisor y control presentados para la automatización de procesos análogos en la industria.

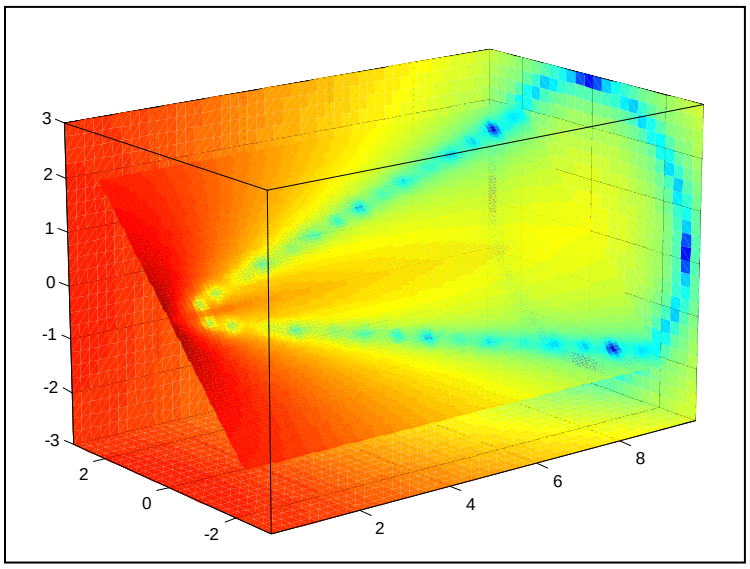
Bibliografía

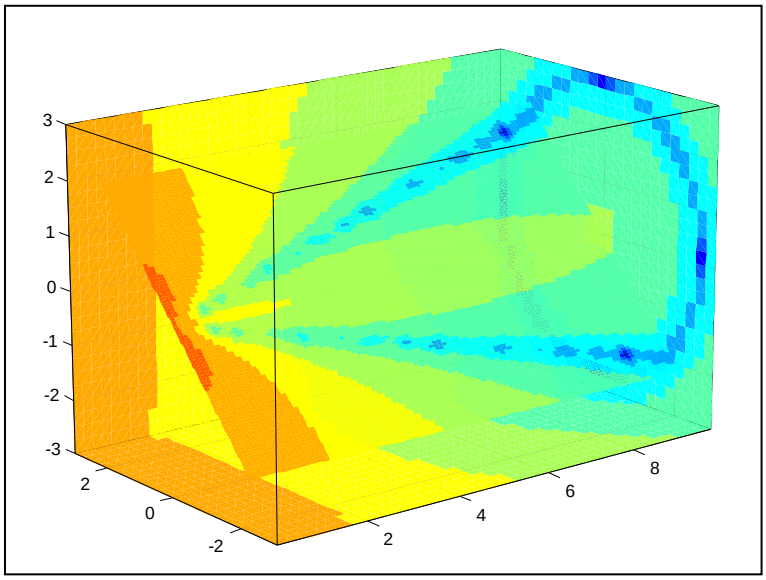
1. Cilento Augusto, Sistemas de control de motores en Tiempo Real mediante MATLAB, 2007.
2. Chacon, Dijort, J.Castrillo, Supervisión y control de procesos,2001-02.
3. Dzung Phan Quoc, ANN Control System DC Motor, HCMC University of Technology. Vietnam.
4. Florescu, Adriana. Fuzzy Design for DC motro speed control. "Politehnica" University of Bucharest.
5. Gimenez, Ing Gustavo. Placas de adquisión de datos y control para PC, 2002
6. Isasi, P. y Galván Inés. "*Redes de Neuronas Artificiales Un enfoque práctico*". Editorial Pearson Prentice Hall (2004).
7. Moleykutty, G. Speed control of separately Excited DC Motor. Faculty of engineering and Technology, Multimedia University, Melaka, Malaysia.
8. M. Rashid, Power Electronic, Prentice Hall, New York 2004.
9. N. Mohan, T. Undeland, Power Electronic Applications, converters and design.
10. Rojas Purón L., Morera Hernández M. Supervisor gráfico de accionamiento eléctrico asistido por MATLAB. Taller Nacional de NTIC aplicadas a la Ingeniería Eléctrica. ISPJAE. Cuidad de la Habana. Junio del 2003.
11. Rugeles Vargas, Ing Andrés. Diseño de un controlador difuso a través de la herramienta fuzzy de MATLAB, 2007

12. Sanchez, M. Realización de cnotroladores lógicos difusos para el control de procesos en tiempo real: especificación y diseño. Instituto de Investigaciones eléctricas. Cuernavaca, Morelos, México.
13. Seco Granja, Fernando. Conexión de instrumentos de medida con GPIB, 2004.
14. Sotelo Neyra, MSc. Victor. Control de posicionamiento de alta precisión para servomotores robóticos por lógica difusa y seguidor de posición por deslizamiento. Universidad nacional de ingeniería.
15. http://www.matpic.com
16. http://www.monografias.com/trabajos12/redneur/redneur.shtml
17. http://espanol.geocities.com/melvinosman/iartificial.doc
18. http://www.monografias.com
19. http://www.secyt.frba.utn.edu.ar/gia/RNA.pdf
20. https://upcommons.upc.edu/pfc/bitstream/2099.1/2746/1/37766-1.pdf

Anexos

Espectros de colores que identifica un tipo de fluido





Mapa de colores con el espectro