



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 GRÁFICA PARA EL PROCESAMIENTOS DE
DATOS USANDO EL GUI DEL MATLAB**

AUTOR: *William Quesada Pupo*

TUTORES: *Dr. C. Luis Delfín Rojas Purón.*
Ing. Daniel Mendiola Ellis.

Curso, 2010-2011
“Año 53 de la Revolución”



Declaración de autoridad

Yo: William Quesada pupo

Autor de este trabajo de Diploma tutorado por el Dr. C. Luis Delfín Rojas Purón, el Ing. Daniel Mendiola Ellis, 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.

William Quesada Pupo
(Autor)

Dr. C. Luis Delfín Rojas Purón
(Tutor)

Ing. Daniel Mendiola Ellis
(Tutor)



Pensamiento

“Años atrás la electricidad era fuerza rebelde, destructora y confusa. Hoy obedece al hombre como caballo domado”... ..“La disposición de los objetos anuncia ya el hermoso desenvolvimiento y futura amplitud de la Ciencia Eléctrica”

José Martí, “Exposición de Electricidad”. Marzo de 1883.



Dedicatoria

Dedico este trabajo:

A mis padres Nilsa Pupo y William Quesada, a mi hermano Humberto y a mi novia Yunisleydis García que sin su apoyo no hubiese sido posible desarrollar este trabajo.

A mis tutores, Dr. Luis Delfín Rojas Purón y al Ing. Daniel Mendiola Ellis, 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 mis padres, a mi hermano y a mi novia, por su sacrificio e incondicionalidad para conmigo, en los buenos y malos momentos por los que he transitado.

Agradezco a la Revolución Cubana por darme la oportunidad de convertirme en un profesional.

A mis tutores por su guía certera en la ejecución del trabajo y mi formación como Ingeniero.

Agradezco además al maravilloso colectivo del Departamento de Física, por darme la oportunidad de trabajar con ellos y de creer realmente en mí, en especial a la MSc. Dania Amat y Iván Casals.

Agradezco a mis verdaderos amigos el Ing. Gerardo Luis Pupo y a David San Miguel Hidalgo por la confianza depositada en mí.

Agradezco de forma general a mi familia y a mis amigos que de una forma u otro contribuyeron a la realización de este sueño.

A todos:

MUCHAS GRACIAS



Resumen

El trabajo consiste en implementar una interface gráfica para el procesamiento de datos utilizando las bondades del GUI del MATLAB, donde se elabora un software como herramienta de trabajo desde el PC y se tiene como caso de estudio un generador de funciones que permitirá la simulación y ajuste de los parámetros del circuito de mando del variador de velocidad Altivar. La interface resuelve tareas de comunicación, supervisión y control en los objetos de Simulink, tarjetas y puertos del PC. Todos los recursos son gestionados a partir de los drivers existente en el paquete de MATLAB. Esta herramienta virtual es utilizable en tareas académicas y de investigación.

Summary

The work is to implement a graphical interface for data processing using MATLAB GUI goodness, which produces software as a tool from the PC and case study is a function generator that will allow the simulation and adjust the parameters of the circuit controlling the Altivar. The interface resolves communication tasks, supervision and control in Simulink objects, cards and ports on the PC. All resources are managed from the existing drivers in the package of MATLAB. This online tool is usable in academic and research tasks.



Índice

Declaración de autoridad.....	I
Pensamiento	II
Dedicatoria	III
Agradecimientos.....	IV
Resumen	V
Summary	V
Índice.....	VI
Introducción general.....	- 1 -
Problema.....	- 1 -
Hipótesis.....	- 2 -
Objetivo.....	- 2 -
Resultados Esperados	- 2 -
<i>CAPÍTULO I: FUNDAMENTOS PARA EL PROCESAMIENTO DE DATOS</i>	
1.1 Introducción.....	- 3 -
1.2 Fundamentos del procesamiento de señales	- 3 -
1.3 Características básicas de un convertidor A/D	- 6 -
1.4 Tiempo de conversión:	- 8 -
1.5 Errores en los convertidores analógico/digital.	- 8 -
1.5.1 Error de offset	- 8 -
1.5.2 Error de cuantificación.....	- 9 -
1.5.3 Error de linealidad (linealidad integral)	- 9 -
1.5.4 Error de apertura	- 10 -
1.6 Etapa de acondicionamiento de la señal	- 10 -
1.7 Muestreo de la señal	- 12 -
1.7.1 El teorema de Nyquist o teorema de muestreo	- 13 -
1.8 Efectos de Aliasing:.....	- 14 -
Conclusiones Parciales	- 15 -



CAPÍTULO II: DESCRIPCIÓN DEL OBJETO DE ESTUDIO: VARIADOR DE VELOCIDAD ALTIVAR

2.1 Introducción.....	- 16 -
2.2 Descripción de un Variador	- 17 -
2.3 Variación de la velocidad para motores de inducción.	- 17 -
2.3.1 Cambio de frecuencia	- 20 -
2.3.2 Criterios generales de selección.....	- 23 -
2.4 Principales ventajas de la regulación de velocidad por variación de frecuencia.....	- 24 -
2.4.1 Control de velocidad más frecuente en los motores de inducción.....	- 24 -
2.5 Descripción del variador Altivar 71 utilizado en la estación experimental.....	- 26 -
2.5.1 Esquema de Potencia y Circuito de mando.....	- 27 -
2.5.2 Protocolos de Comunicación	- 28 -
2.5.3 Aplicaciones de Accionamientos de Velocidad Ajustable	- 36 -
Conclusiones Parciales.....	- 39 -

CAPÍTULO III: FUNDAMENTOS PARA EL PROCESAMIENTO DE DATOS DESDE MATLAB

3.1 Introducción.....	- 40 -
3.2 Entorno de trabajo MATLAB y herramientas para la adquisición de datos.	- 41 -
3.2.1 Interface del MATLAB para capturar y generar señales.	- 41 -
3.3 Instalación del Software REAL-TIME WINDOWS TARGET	- 42 -
3.3.1 Análisis la sección de Procesamiento de Datos en MATLAB	- 43 -
3.3.2 Procesamiento de Datos en Tiempo Real	- 45 -
3.3.3 Ejemplo de la sección de adquisición de datos en Matlab.....	- 47 -
3.3.4 Lectura de Datos Analógicos	- 50 -
3.4 Mapeo de Memoria en MATLAB.....	- 61 -
3.4.1 Mapa de Memoria para Señales y Parámetros.....	- 62 -
3.4.2 Mapeo de Memoria de Códigos y Datos.....	- 62 -
3.4.3 Comunicación Basada en Registros.....	- 63 -
3.5 Interfaz Gráfica de Usuario en MATLAB.....	- 67 -
Conclusiones Parciales:.....	- 86 -



CAPÍTULO IV: EVALUACIÓN ECONÓMICA DEL TRABAJO

4.1 Introducción.....	- 87 -
4.2 Cálculo de los Gastos de Explotación	- 87 -
4.3 Gastos de Inversiones	- 88 -
4.4 Cálculo del Tiempo de Amortización.....	- 89 -
<i>Conclusiones Generales</i>	- 90 -
<i>Recomendaciones</i>	- 91 -
<i>Bibliografía</i>	- 92 -



Introducción general

En la industria moderna, la utilización de la información es de suma importancia para determinar estrategias técnicas y administrativas sobre el proceso. La información se hace más relevante desde dos puntos de vista: su adquisición en tiempo real para revelar situaciones inmediatas y la historia del sistema para poder proyectarlo estadísticamente al futuro.

Los sistemas de procesamiento de datos se encaminan a utilizar configuraciones de modo que puedan implementar con la información obtenida acciones de control y monitoreo de los datos y cambios detectados. En las configuraciones actuales el mando del sistema, así como el almacenamiento de la información, se da en los computadores personales, con ventajas de versatilidad en aplicaciones de software para la utilización de la información.

Los componentes de los sistemas están diseñados para una compatibilidad con el computador personal, utilizando módulos para el tratamiento digital de las señales análogas.

El objetivo principal del trabajo, está basado en el procesamiento de datos ya recolectados en la PC, con el fin de elaborar un programa visual o gráfico, que permita el comportamiento de variables en el dominio del tiempo, o muestreadas de acuerdo al ancho de banda del hardware que tiene el sistema.

Problema

La necesidad de implementar una interface gráfica para el procesamiento de datos ya recolectados en la PC, de dispositivos generadores de señales, que intervienen en los circuitos de mando de variadores de velocidad.



Hipótesis

Si se tiene la base de datos ya recolectada en la PC y los recursos informáticos que ofrece el MATLAB, es posible entonces implementar una herramienta para procesar y gestionar los datos del sistema a través de una interface, resolviéndose así las tareas de comunicación, supervisión y control de los elementos informativos.

Objetivo

Elaborar una interface gráfica que permita el procesamiento de datos a través del GUI del MATLAB, para resolver gestiones de comunicación, supervisión y control de dispositivos u objetos contenidos sobre Simulink, tarjetas, y lectura y escritura de puertos en la PC.

Resultados Esperados

- ✚ Describir el sistema de acondicionamiento de señales para el procesamiento de datos.
- ✚ Caracterizar el variador de velocidad Altivar operando en condiciones de ensayo de laboratorio.
- ✚ Elaborar una interface informática para el procesamiento de datos desde el GUI del MATLAB.
- ✚ Obtener la comunicación y control de un dispositivo generador de señales montado sobre Simulink, mediante la interface gráfica elaborada por el GUI.
- ✚ Mostrar la supervisión de algunas variables generadas a partir de la interface elaborada desde el MATLAB.



Capítulo I

Fundamentos para el Procesamiento de Datos

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 estas, siendo así impulsoras 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 en que vivimos.

Es por ello que en el presente capítulo se describirá el proceso para el procesamiento de datos, así como las herramientas a utilizar para desarrollar dicho proceso, teniendo en cuenta las características de algunos de los elementos y dispositivos que permiten un correcto y seguro funcionamiento del sistema.

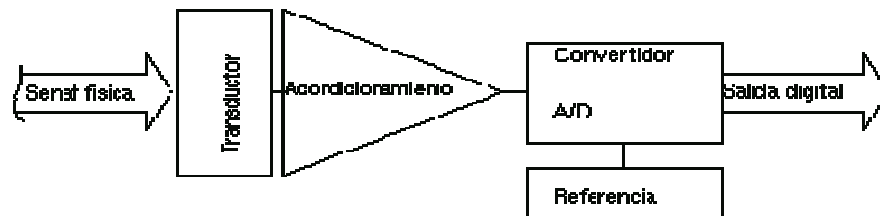
1.2 Fundamentos del procesamiento de señales

La adquisición o procesamiento de datos o señales, consiste en tomar muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otras electrónicas (sistemas digitales). Consiste en tomar un conjunto de señales físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan procesar en una computadora o PAC.¹

¹ *Rubio Mena, Juan Antonio. Nociones Básicas sobre adquisición de Señales.*

El propósito de la adquisición de datos, es medir un conjunto de variables eléctricas y físico como: voltaje, corriente, temperatura, presión o sonido. Para esto se requiere una etapa de acondicionamiento, que adecúa la señal a niveles compatibles con el elemento que se hace la transformación a señal digital. De modo que el elemento que realiza estos cambios es el módulo de digitalización o tarjeta de Adquisición de Datos (ADQ).

Estructura para un sistema de adquisición de datos o señales: En la figura que aparece a continuación podemos ver los bloques que componen nuestro sistema de adquisición de datos: ²



Esquema de bloques de un sistema de adquisición de datos.

Como podemos observar, los bloques principales son los siguientes:

- Transductor
- El acondicionamiento de señal
- El convertidor analógico-digital
- La etapa de salida (interface 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. ¹

¹ Rubio Mena, Juan Antonio. *Nociones Básicas sobre adquisición de Señales*.

² Pupo Bacallao, Gerardo Luis. *Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos*. Moa.2010.



Salvo que la señal de entrada sea eléctrica, podemos decir que el transductor es un elemento que convierte un tipo de energía 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).
- Adaptar la impedancia de salida de uno con la impedancia de entrada del otro.

La adaptación entre los rangos de salida y entrada del convertidor, tiene como objetivo, aprovechar el margen dinámico del convertidor, de modo que la máxima señal de entrada debe coincidir con la máxima del 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 1y 10k.

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 pueda presentar una serie determinada de valores. Por lo 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 SAD (Sistema de Adquisición de Señales) con el resto del equipo, el cual puede ser una serie de buffers digitales incluidos en el circuito del convertidor, hasta una interfaz RS-232, RS-485 o Ethernet para conectar a un ordenador o estación de trabajo, en el caso de sistemas adquisición de datos comerciales.

1.3 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 seguro y real. No las mencionaremos todas, sino las básicas, dejando un estudio en profundidad de los convertidores para otro documento. Las características que no debemos olvidar son éstas:²

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

² Pupo Bacallao, Gerardo Luis. *Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos. Moa.2010*



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, V_i , en el voltaje de entrada que se necesita para cambiar en un bit la salida digital. En resumen, tenemos que:²

$$V_i = \frac{V_{fe}}{(2^n - 1)}$$

donde n es el número de bits del convertidor, y V_{fe} 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:

$$\frac{2V}{2^8 - 1} = 7,84 \frac{mV}{\text{paso}}$$

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

² Pupo Bacallao, Gerardo Luis. *Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos*. Moa.2010.

$$\frac{5V}{2^8 - 1} = 19,6 \frac{mV}{\text{paso}}$$

1.4 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 muestra una cuota 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

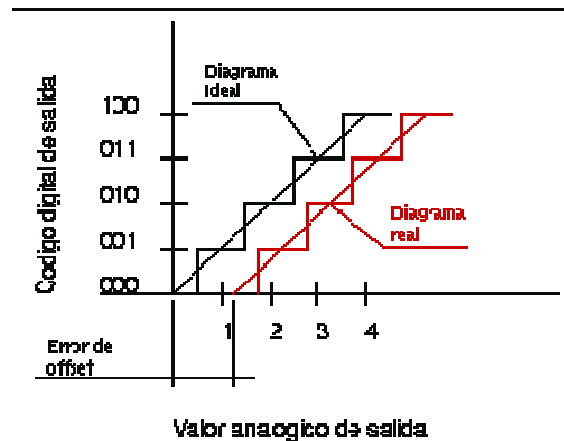
1.5 Errores en los convertidores analógico/digital.

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:

1.5.1 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.



Esquema del error de offset

1.5.2 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 $\pm \frac{1}{2}$ LSB (Bit menos significativo). Por tanto, cada código digital representa un valor que puede estar dentro del $\frac{1}{2}$ LSB a partir del punto medio entre valores digitales continuos.²

1.5.3 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.

² Pupo Bacallao, Gerardo Luis. *Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos*. Moa.2010.

1.5.4 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 $\frac{1}{2}$ LSB) la frecuencia máxima de muestreo deberá ser:

$$F_{\max} = \frac{1}{T_a \cdot \Pi \cdot 2^{n+1}}$$

En esta fórmula T_a 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 número 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.

1.6 Etapa de acondicionamiento de la señal

Con más detalles, en una etapa de acondicionamiento de señal 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 la señal de entrada debe de 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 esta 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 sean afectadas por diferencias en el potencial de masa o por tensiones de modo común.

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

Multiplexado: Es la conmutación de las entradas del convertidor, de modo que con un solo 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. Se aconseja que los multiplexores se utilicen antes del conversor y después del acondicionamiento de la señal, ya que de esta manera no molestará a los aislantes que podamos tener.

Filtrado: El fin de esta etapa es eliminar las señales no deseadas de la señal que estamos observando. En las señales cuasi-continuas, (como la temperatura) se usa un filtro de ruido de unos 4Hz, que eliminará interferencias, incluidos los 50/60Hz 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 si no se eliminan 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 extensométricas, 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.

1.7 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. ²

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.

²*Pupo Bacallao, Gerardo Luis. Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos. Moa.2010.*



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:

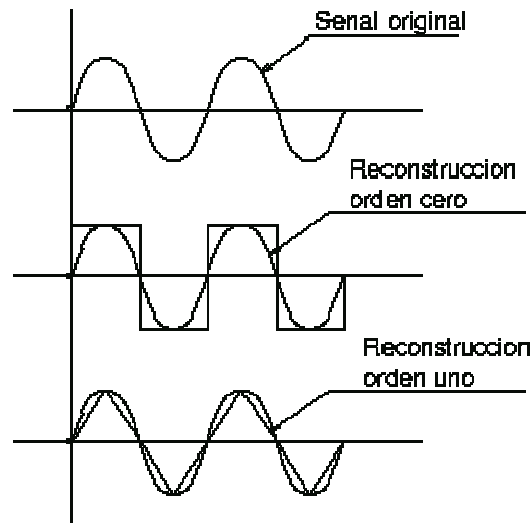
- Aumentar el número de muestras por ciclo
- Filtrado previo al multiplexado
- Filtrar la salida del convertidor digital / analógico

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.

1.7.1 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 ¡cuidado!, 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 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).



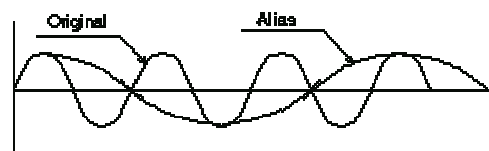
Muestreo de señales y su reconstrucción por métodos orden cero y orden uno.

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.

1.8 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:



Esquema del efecto Alias



Como se aprecia, al tomar varias muestras con un período 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 el ruido, y otras señales senoidales, que aparentemente no son ruido, pero que también afectan la señal en cuestión. Por tanto, cualquier frecuencia de muestreo excesivamente baja, nos da información falsa sobre la señal.

Conclusiones Parciales

- Se tiene adecuada información sobre el procesamiento de datos usando los medios informáticos virtuales en los laboratorios docentes de la carrera de Ingeniería Eléctrica.
- Es factible la elaboración de una interface en MATLAB para gestionar las propiedades de dispositivos procesadores de señales o datos.

Capítulo II

Descripción del Objeto de Estudio: Variador de Velocidad Altivar

2.1 Introducción

El procesamiento de datos en dispositivos de potencia es un aspecto complejo, debido a la cantidad de componentes electrónicos que hay que atender en las tareas de comunicación, supervisión y control.

El variador Altivar posee de diseño su propio sistema supervisor a nivel industrial, pero no monitorea todas las variables que tiene disponible el sistema desde su tarjeta de adquisición de datos. Es por esto, que es de interés ofrecer desde una interface gráfica, varias opciones de supervisión, comunicación y control con propósitos de investigación en nuestros laboratorios y ampliar las ventanas de visualización de sus principales variables de explotación.

El presente capítulo, tiene como objetivo, describir este dispositivo convertidor, que representa en sí un Inversor Sinusoidal de Tensión a PWM como objeto industrial de futura investigación en el campo de procesamiento estadístico de datos. Para esto se ofrecen elementos técnicos y de control para aplicar los elementos informativos tratados en este trabajo.

Una gran parte de los equipos utilizados en la industria moderna funcionan a velocidades variables, como por ejemplo los trenes laminadores, los mecanismos de elevación, las máquinas-herramientas, etc. En los mismos, se requiere un control preciso de la velocidad para lograr una adecuada productividad, una buena terminación del producto elaborado, o garantizar la seguridad de personas y bienes.



El estudio de este proceso, para cada caso particular, tiene una gran importancia práctica, ya que la elección correcta de las características de los motores y variadores a instalar para un servicio determinado, requieren el conocimiento de las particularidades de éste producto.

La regulación de velocidad puede realizarse por métodos mecánicos, como poleas o engranajes, o por medios eléctricos.

La máquina de inducción alimentada con corriente C.A., especialmente la que utiliza un rotor en jaula de ardilla, es el motor eléctrico más común en todo tipo de aplicaciones industriales y el que abarca un margen de potencias mayor. Pero no basta conectar un motor a la red para utilizarlo correctamente, sino que existen diversos elementos que contribuyen a garantizar un funcionamiento seguro.

2.2 Descripción de un Variador

Los variadores son convertidores de energía encargados de modular la energía que recibe el motor. Otra definición utilizada son los variadores de velocidad, que constituyen dispositivos que permiten variar la velocidad y la acopla de los motores asíncronos trifásicos, convirtiendo las magnitudes fijas de frecuencia y tensión de red en magnitudes variables.

2.3 Variación de la velocidad para motores de inducción.

Con el motor asíncrono de jaula, para obtener una auténtica variación de velocidad, es necesario variar la frecuencia de las tensiones de alimentación. Este proceso cuyo campo de aplicación va siendo cada vez mayor, utiliza un motor especialmente sencillo y robusto y requiere convertidores y un control más elaborado.³

³ ***Bonal. Jean. Accionamientos Eléctricos a Velocidad Variable, V1. Editorial Técnica y Documentación. Paris, Francia. 1999***

Las prestaciones de los convertidores de frecuencia dependen de las características de los semiconductores que utilizan. Los componentes electrónicos existentes en el mercado permiten responder a la casi totalidad de las necesidades industriales, tanto en el ámbito técnico como en el económico.

Con respecto a las demás soluciones, el método de variación de velocidad con motor asíncrono de jaula y convertidor de frecuencia presenta entre otras las siguientes ventajas:³

- § Mejora la explotación de los procesos flexibilizando el comportamiento de las máquinas.
- § Optimiza el consumo de energía.
- § Aumenta la seguridad y la disponibilidad de los accionamientos.

Los variadores de frecuencia (AFD, *Adjustable Frequency Drive*; o bien VFD *Variable Frequency Drive*) permiten controlar la velocidad tanto de motores de inducción (asíncronos de jaula de ardilla o de rotor devanado), como de los motores síncronos mediante el ajuste de la frecuencia de alimentación al motor.⁴

Para el caso de un motor asíncrono, la velocidad se determina mediante la siguiente expresión:

$$Nm = \frac{120 \cdot f \cdot (1 - s)}{P}$$

Donde:

N_s = velocidad síncrona (rpm)

N_m = velocidad mecánica (rpm)

f = frecuencia de alimentación (Hz)

³ Bonal. Jean. *Accionamientos Eléctricos a Velocidad Variable*, V1. Editorial Técnica y Documentación. París, Francia. 1999

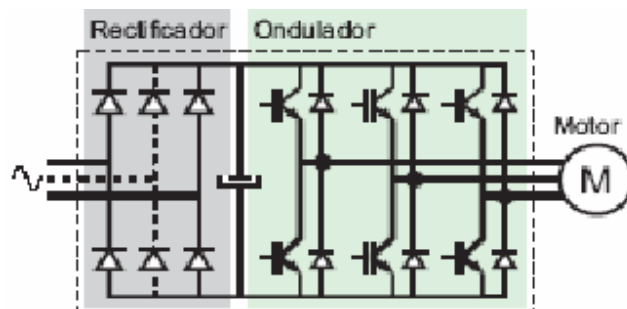
⁴ Fonseca. Alpajón. Dannys. *Interfaz para comunicación de sistema SCADA de accionamientos con variadores de velocidad*. Moa. 2009

s = deslizamiento (adimensional)

P = número de polos.

Como puede verse en la expresión anterior, la frecuencia y la velocidad son directamente proporcionales, de tal manera que al aumentar la frecuencia de alimentación al motor, se incrementará la velocidad de la flecha, y al reducir el valor de la frecuencia disminuirá la velocidad del eje. Por ello es que este tipo de variadores manipula la frecuencia de alimentación al motor a fin de obtener el control de la velocidad de la máquina.⁴

Estos variadores mantienen la razón Voltaje/ Frecuencia (V/Hz) constante entre los valores mínimo y máximos de la frecuencia de operación, con la finalidad de evitar la saturación magnética del núcleo del motor y además porque el hecho de operar el motor a un voltaje constante por encima de una frecuencia dada (reduciendo la relación V/Hz) disminuye el par del motor y la capacidad del mismo para proporcionar potencia constante de salida.



Esquema eléctrico de un Convertidor de frecuencia.

⁴ Fonseca. Alpajón. Dannys. *Interfaz para comunicación de sistema SCADA de accionamientos con variadores de velocidad*. Moa. 2009

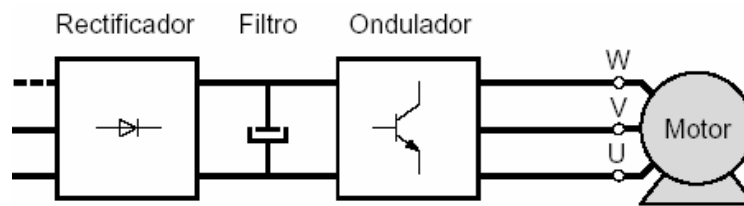


Diagrama de bloques de un Convertidor de frecuencia.

2.3.1 Cambio de frecuencia

La preferencia actual por la regulación a frecuencia variable, se debe a la posibilidad de utilizar el sencillo y robusto motor de jaula de ardilla; cuyo mantenimiento es mucho más fácil que el de un motor de contactos deslizantes, lo que resulta muy importante en máquinas que operan bajo condiciones ambientales difíciles. Además este tipo de motor eléctrico resulta más económico y compacto que los restantes.⁵

Asimismo, este método permite transformar fácilmente un equipo de velocidad fija en un accionamiento de velocidad variable, sin realizar grandes modificaciones.

Con este tipo de regulación se puede obtener un amplio control de velocidades, con el máximo par disponible en todas las frecuencias con un elevado rendimiento. Si se prolonga la característica al cuadrante generador se puede obtener un frenado muy eficiente por reducción de frecuencia, con una recuperación de energía hacia la red de alimentación.

Si bien pueden utilizarse distintos tipos de convertidores de frecuencia rotativos (semejantes al sistema Ward-Leonard), en la actualidad la modificación de la frecuencia se realiza fundamentalmente por medio de variadores estáticos electrónicos que

⁵ *Control y regulación de la velocidad en motores asíncronos trifásicos. SicaNews [newsletter@sicaelec.com].2002*

ofrecen una regulación suave, permitiendo un aumento en la vida útil de todas las partes involucradas y originando un ahorro en el mantenimiento por ausencia de partes en movimiento que sufran desgastes.

Los mismos se construyen generalmente con tiristores gobernados por un microprocesador que utiliza un algoritmo de control vectorial del flujo, y consisten básicamente en un convertidor estático alterna-alterna (cicloconvertidor) o alterna-continua-alterna (convertidor de enlace), que permiten la modificación progresiva de la frecuencia aplicada, con la consiguiente modificación de la corriente y el par motor. En algunos casos se agregan filtros de armónicas.

En el ciclo convertidor se sintetiza una onda de menor frecuencia a partir de una alimentación polifásica de mayor frecuencia, conectando sucesivamente los terminales del motor a las distintas fases de la alimentación. La onda sintetizada generada es rica en armónicos y en algunos casos el circuito puede generar subarmónicos que podrían llegar a producir problemas si excitasen alguna resonancia mecánica del sistema.

Por otro lado, el ciclo convertidor ofrece una transformación simple de energía de buen rendimiento, permite la inversión del flujo de potencia para la regeneración y la transmisión de la corriente reactiva; proporcionando una gama de frecuencias de trabajo que va desde valores cercanos a cero hasta casi la mitad de la frecuencia de alimentación, con fácil inversión de fase para invertir el sentido de rotación.

En ciertos casos este tipo de convertidor se emplea en motores asíncronos de rotor bobinado con alimentación doble, estando el estator conectado a la red y el rotor al convertidor.

En el convertidor de enlace la alimentación de la red de corriente alterna se rectifica en forma controlada y luego alternativamente se conmutan las fases del motor al positivo y al negativo de la onda rectificada, de manera de crear una onda de alterna de otra frecuencia.



La tensión y frecuencia de salida se controlan por la duración relativa de las conexiones con las distintas polaridades (modulación del ancho de pulso) de manera de conservar constante el cociente tensión / frecuencia para mantener el valor del flujo magnético en el motor.

Aunque la onda de tensión obtenida no es sinusoidal, la onda de corriente tiende a serlo por efecto de las inductancias presentes. Además, de este modo se obtiene una amplia gama de frecuencias por encima y por debajo de la correspondiente al suministro, pero exige dispositivos adicionales c.c./c.a. para asegurar el flujo de potencia recuperada.

Hay que considerar que las corrientes poliarmónicas generan un calentamiento adicional que disminuye el rendimiento y puede llegar a reducir el par (por ejemplo, el 5º armónico produce un campo giratorio inverso).

También cabe acotar que la vibración de los motores aumenta cuando se los alimentan con convertidores electrónicos de frecuencia y que la componente de alta frecuencia de la tensión, de modo común de los convertidores de frecuencia, puede causar un acoplamiento con la tierra a través de la capacidad que se forma en los rodamientos, donde las pistas actúan como armaduras y la capa de grasa como dieléctrico.

Asimismo digamos que los variadores de velocidad generalmente también sirven para arrancar o detener progresivamente el motor, evitando por ejemplo, los dañinos golpes de ariete que pueden aparecer en las cañerías durante la parada de las bombas.

Estos convertidores poseen protecciones contra asimetría, falla de tiristores, sobretensión y sobrecarga; además de vigilancia del tiempo de arranque con limitación de la corriente, control de servicio con inversión de marcha, optimización del factor de potencia a carga parcial, maximización del ahorro de energía durante el proceso.⁵

⁵ *Control y regulación de la velocidad en motores asíncronos trifásicos. SicaNews [newsletter@sicaelec.com].2002*

2.3.2 Criterios generales de selección

Las principales cuestiones que hay que considerar son:

- Verificar la tensión de suministro y las tensiones nominales del convertidor y del motor.
- Seleccionar el convertidor adecuado a la potencia nominal del motor.
- Comprobar también que la corriente nominal del convertidor sea igual o mayor que la del motor seleccionado.
- Verificar el tipo de característica de momento de la carga (constante, tipo centrífugo, etcétera).
- El momento real de la carga debe estar en todos los puntos por debajo del indicado por la curva de cargabilidad (hay que conocer qué tipo de convertidor va a ser usado). Si la operación no es continua en todos los puntos, se puede sobrepasar el momento indicado por la curva, pero este caso requiere de una evaluación especial.
- Prestarle mucha atención a los casos especiales en cuanto a altos requerimientos de momento de arranque o de momento máximo.
- El momento máximo del motor debe ser por lo menos 40% mayor que el momento a cualquier frecuencia.
- Cuando se emplee freno eléctrico, hay que realizar las comprobaciones adecuadas.
- Debe comprobarse el intervalo de velocidad requerido y el que puede proporcionar el convertidor.
- La velocidad máxima permisible del motor no se puede exceder (esto se debe chequear con las normas).
- Analizar si hay necesidades especiales en cuanto al medio ambiente.
- Debe comprobarse el sistema de tierra del motor, del equipo accionado y del tacómetro.

- Verificar técnica y económicamente si un sistema separado de enfriamiento reduce el tamaño del motor y, consecuentemente, el tamaño del convertidor.
- A altas velocidades debe prestarse especial atención a la construcción de los rodamientos, la lubricación, el ruido del ventilador, el balanceo, las velocidades críticas, los sellos de los ejes y el momento máximo del motor.
- A bajas velocidades debe evaluarse la lubricación de los rodamientos, la ventilación del motor y el ruido electromagnético.

2.4 Principales ventajas de la regulación de velocidad por variación de frecuencia

Desde el punto de vista técnico destacan:

La amplia gama de velocidades disponible que permite responder a todas las demandas del proceso sin necesidad de recurrir a medios mecánicos.

La simplicidad de los motores de corriente alterna, especialmente el asíncrono de rotor en jaula de ardilla, con lo que se reducen drásticamente los problemas de reparación y mantenimiento de equipos más complejos.

Desde el punto de vista económico, las ventajas principales radican en:

El ahorro de energía que se produce por el mejor rendimiento del motor. La menor inversión que representa el motor de corriente alterna frente a otros equipos capaces de actuar a velocidad variable.

La reducción de los costes de mantenimiento e instalación tanto en costes directos como en tiempo de detención del proceso productivo.

2.4.1 Control de velocidad más frecuente en los motores de inducción

De acuerdo a las formas de realizar el control en el motor se clasifican en:

- Control escalar
- Control vectorial



- Control directo del torque.

El control escalar se realiza de la forma siguiente:

Las propiedades que ofrece el control por frecuencia en los motores de inducción son:

- Controlabilidad de las variables de tensión y frecuencia.
- Simulación de variables sinusoidales de corriente alterna usando un modulador.
- Flujo magnético con radio U/f constante.
- Accionamiento a lazo abierto.
- Nivel del torque dictado por la carga.

Las referencias de las tensiones y frecuencias son alimentadas dentro del modulador las cuales simulan las ondas sinusoidales y alimentan los devanados del estator del motor eléctrico. Esta técnica es llamada Modulación por Ancho de Pulsos (PWM siglas en inglés) y utiliza el factor de la topología del Inversor que opera con un rectificador a diodo en la entrada, un circuito intermedio C.D. que mantiene la tensión constante y el circuito inversor a la salida con transistores tipo IGBT, que alimentan los devanados del motor.

Significativamente este método no usa un lazo de realimentación el cual toma mediciones de velocidad o posición desde el árbol del motor y se alimentan estos dentro del lazo de control. Este tipo de control se denomina control del accionamiento a lazo abierto.

Las ventajas de este tipo de control son:

- Bajo costo.
- No requiere de dispositivos de realimentación, es simple.

Es decir, por no necesitar dispositivos de realimentación, es de bajo costo y ofrece simple y económica solución. Puede ser utilizado en accionamientos de ventiladores, algunas bombas, etc, donde no se requiere de un control altamente exacto o preciso.

Tienen como desventajas:

- No usa el campo orientado.
- Ignora el estado del motor.
- El torque no es controlado.
- Usa un modulador demorado.

Esta técnica se conoce como control escalar de frecuencia. Instantáneamente la frecuencia y el voltaje son controlados como variables principales y son aplicados directamente en los devanados del estator. El estado del motor es ignorado. No resulta un control exacto del accionamiento, el modulador básicamente es lento en su comunicación entre las señales del voltaje y la frecuencia y las necesidades de respuesta del motor ante cambios de la carga u otra perturbación.

2.5 Descripción del variador Altivar 71 utilizado en la estación experimental

De forma particular y atendiendo a los criterios de selección, en nuestro trabajo utilizaremos fundamentalmente el variador Altivar 71 por presentar un gran número de ventajas y cualidades de factibilidad que presentamos a continuación:

Descripción de forma general del variador utilizado.

El variador de velocidad Altivar 71 es un inversor de frecuencia para motores asíncronos de jaula de ardilla de 0,37 a 500 kW.

Sus funciones evolucionadas y sus prestaciones en lazo abierto y lazo cerrado le permiten responder perfectamente a las exigencias de las máquinas complejas y de altas potencias. Fiable y robusto, el Altivar 71 asegura un servicio sin interrupción. Resiste las caídas de tensión hasta – 50 %, disminuye las perturbaciones de las instalaciones con sus filtros RFI y soporta temperaturas de hasta 50°C. Posee una

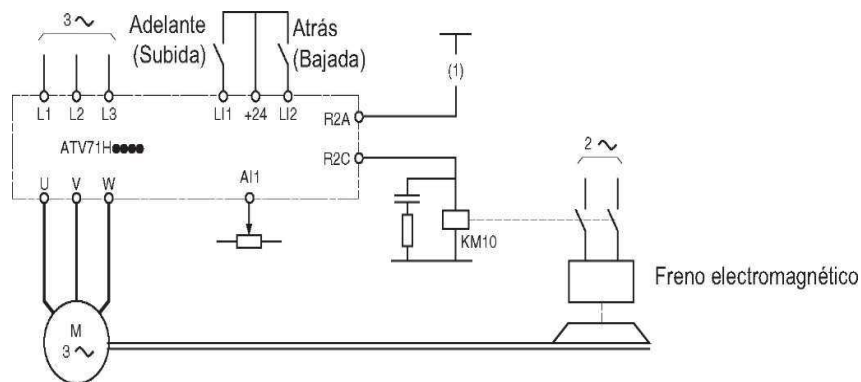


amplia gama de certificaciones de calidad de conformidad con los estándares: baja tensión en 50178 y IEC 61800-5-1. Emisores conductivos y radiados CEM: entorno 2 (sector industrial)- IEC 61800-3, EN 55011 - EN 55022. En cuanto al grado de protección: IP21 hasta 75 kW en 380/480 V y hasta 45kW en 220/ 240 V. Para valores más altos que estos, se cumple con el grado de protección agregando el kit de conformidad IP21. En cuanto a la Temperatura ambiental de operación: de -10 a +50°C sin reducir la capacidad normal. Operación hasta 60°C reduciendo la capacidad normal y usando un kit de control de ventilación acorde con el rango de tensión. Para una altitud máxima de operación: 1000 m sin reducir la capacidad normal, para valores más altos se reduce la capacidad de la corriente 1% por cada 100 m adicionales. Limitado a 2000 m del sistemas de distribución eléctrica referidos a tierra (TT o TN-S). En cuanto a la humedad relativa: 95 %, sin condensación, ni goteo de agua. Conforme con IEC 60068-2-3. Las condiciones ambientales: IEC 721-3-3, clase 3C1 y 3S2. Conforme con clase 3C2 para la versión reforzada del Altivar 71 (y el sufijo S337 para ordenar la referencia).En cuanto a la alimentación: 200V -15% 240V +10%, trifásica para ATV71H...M3 y ATV71H..M3X. Variadores ATV71HU40M3 hasta ATV71HU75M3 pueden ser usados para operación monofásica reduciendo la capacidad nominal del variador por un punto y agregando un inductor.

Posee un terminal gráfico extraíble El "joystick" de navegación permite un acceso rápido y sencillo a los menús desplegables. La pantalla gráfica muestra de forma clara los textos en 8 líneas de 24 caracteres. Puede conectarse en enlace multipunto a diversos variadores. Se suministra con 6 idiomas instalados de base (español, alemán, inglés, chino, francés e italiano). Se pueden cargar otros idiomas mediante flasheado. Hasta 15 kW, el variador Altivar 71 puede pedirse con un terminal de 7 segmentos integrado.

2.5.1 Esquema de Potencia y Circuito de mando

El menú [ARRANQUE RÁPIDO] (SIM-) debe configurarse sólo o antes que otros menús de configuración del variador. Si previamente se ha realizado una modificación en uno de éstos, en particular en [CONTROL MOTOR] (drC-), algunos parámetros de [ARRANQUE RÁPIDO] (SIM-) pueden modificarse como, por ejemplo, los parámetros del motor si se ha elegido un motor síncrono. Regresar al menú [ARRANQUE RÁPIDO] (SIM-) después de haber modificado otro menú de configuración del variador no tiene ningún efecto pero tampoco supone un riesgo. Para no complicar inútilmente la lectura de este capítulo, no se describirán los cambios que siguen a una modificación de otro menú de configuración.



Esquema de conexión del variador con el motor para frenado.

Menú “Arranque rápido”

El menú “Arranque rápido” permite asegurar en pocos pasos el funcionamiento de la aplicación, optimizar el funcionamiento y asegurar su protección.

La arquitectura, la jerarquización de los parámetros y las funciones de acceso directo ofrecen una programación simplificada y rápida, incluso para funciones complejas.

2.5.2 Protocolos de Comunicación

Cada variador de velocidad está provisto de targetas (Dispositivos E/S) para poder lograr la comunicación (interface) con programas de supervisión y control. Con el objetivo de

implementar la interface entre el programa y el accionamiento, el variador está provisto de un mapa de memoria, que determina el reconocimiento y adquisición de los parámetros eléctricos del accionamiento en el programa. Cada uno de los variadores, sin exceptuar el ALTV 71 constan de un mapa de memoria como se muestra en las tablas 1 y 2. Utilizando para la interface ese mapeo de memoria en el protocolo de comunicación

Address	Description	Default value	Address	Description	Default value
40 001	Value of output variable 1	Value of the control word (CMd)	40 017	Value of read register 1	Value of the status word (EtA)
40 002	Value of output variable 2	Value of the speed setpoint (LFrd)	40 018	Value of read register 2	Value of the output speed (rFrd)
40 003	Value of output variable 3		40 019	Value of input variable 3	
40 004	Value of output variable 4		40 020	Value of input variable 4	
40 005	Value of output variable 5		40 021	Value of input variable 5	
40 006	Value of output variable 6		40 022	Value of input variable 6	
40 007	Value of output variable 7		40 023	Value of input variable 7	
40 008	Value of output variable 8		40 024	Value of input variable 8	
40 009	Value of output variable 9		40 025	Value of input variable 9	
40 010	Value of output variable 10		40 026	Value of input variable 10	

Tabla 2.2: Mapas de memorias y descripción de los parámetros a supervisar. (ALTIVAR 71)

Address	Description	Default value	Address	Description	Default value
50 001	Logic address of output variable 1	Address of the control word (CMd) = 8501	50 017	Logic address of input variable 1	Address of the status word (EtA) = 3201
50 002	Logic address of output variable 2	Address of the speed setpoint (LFrd) = 8602	50 018	Logic address of input variable 2	Address of the output speed (rFrd) = 8604
50 003	Logic address of output variable 3	0	50 019	Logic address of input variable 3	0
50 004	Logic address of output variable 4	0	50 020	Logic address of input variable 4	0
50 005	Logic address of output variable 5	0	50 021	Logic address of input variable 5	0
50 006	Logic address of output variable 6	0	50 022	Logic address of input variable 6	0
50 007	Logic address of output variable 7	0	50 023	Logic address of input variable 7	0
50 008	Logic address of output variable 8	0	50 024	Logic address of input variable 8	0
50 009	Logic address of output variable 9	0	50 025	Logic address of input variable 9	0
50 010	Logic address of output variable 10	0	50 026	Logic address of input variable 10	0

Tablas 2.3: Mapas de memorias y descripción de los parámetros a supervisar. (ALTIVAR 71)

Herramientas de diálogo.



El variador Altivar 71 se suministra con un terminal gráfico extraíble:

- El “joystick” de navegación permite un acceso rápido y sencillo a los menús desplegables.
- La pantalla gráfica muestra de forma clara los textos en 8 líneas de 24 caracteres.
- Las funciones avanzadas de la visualización permiten acceder fácilmente a las funciones más complejas.
- Las pantallas de visualización, los menús y los parámetros pueden personalizarse para el cliente o la máquina.
- Ofrece pantallas de ayuda en línea.
- Se pueden memorizar y descargar las configuraciones, son memorizables cuatro ficheros de configuraciones.
- Puede conectarse en enlace multipunto a diversos variadores.
- Puede instalarse a la puerta de armario con un montaje de grado de protección IP54 o IP65.
- Se suministra con 6 idiomas instalados de base (español, alemán, inglés, chino, francés e italiano). Se pueden cargar otros idiomas mediante flasheado. Hasta 15 kW, el variador Altivar 71 puede pedirse con un terminal de 7 segmentos integrado.

El software de programación PowerSuite, permite la configuración, el ajuste y la puesta a punto del variador Altivar 71, así como del conjunto de los demás variadores de velocidad y arrancadores de Telemecanique. Puede utilizarse en conexión directa, a través de Ethernet, por medio de un módem o con una conexión inalámbrica Bluetooth.

Opciones

El variador Altivar 71 puede integrar hasta tres tarjetas opcionales simultáneamente:

- 2 entre las tarjetas siguientes(1):
 - Tarjetas de extensión de entradas/salidas, entradas lógicas, salidas de colector abierto, relés, entrada de sonda PTC, entradas analógicas, salidas analógicas, entrada de control por pulsos de frecuencia.



- Tarjetas de comunicación industrial (Ethernet TCP/IP, Modbus/Uni-Telway, Fipio, Modbus Plus, Profibus DP, DeviceNet, INTERBUS,...). Para una conexión a las principales redes del mercado.
 - Tarjeta programable “Controller Inside”. Permite adaptar el variador a las aplicaciones específicas de forma rápida y evolutiva, mediante la descentralización de las funciones de automatismo (programación en lenguajes conforme a la norma IEC 61131-3).
- 1 entre las tarjetas de interface de codificador de tipo incremental o absoluto: RS422, de colector abierto, Push Pull RS422 con emulación de codificador Resolver SinCos, SinCosHiperface, EnDat, SSI



Para:

- Obtener el par nominal incluso a una velocidad 0.



- Aumentar la precisión de velocidad y de par.
- Aumentar el rendimiento dinámico.

Pueden asociarse opciones externas al variador Altivar 71:

- Módulos y resistencias de frenados (estándar o dedicados a la elevación).
- Unidades de frenado a la red.
- Inductancias de línea, inductancias CC y filtros pasivos, para la reducción de las corrientes armónicas.
- Inductancias de motor y filtros senoidales para las grandes longitudes de cables para suprimir los blindajes.
- Filtros CEM adicionales de entrada.

Canales de control y de consigna

Las órdenes de control (marcha adelante, marcha atrás, parada, etc.) y las consignas pueden proceder de los siguientes canales:

Control	Consigna
• Borneros: entradas lógicas LI • Terminal gráfico • Modbus integrado • CANopen integrado • Tarjeta de comunicación • Tarjeta Controller Inside	• Borneros: entradas analógicas AI, entrada de pulsos, codificador • Terminal gráfico • Modbus integrado • CANopen integrado • Tarjeta de comunicación • Tarjeta Controller Inside • Más/menos velocidad a través del bornero • Más/menos velocidad a través del



	terminal gráfico
--	------------------

[Perfil E/S] (IO): El control y la consigna pueden provenir de canales distintos. Este perfil permite una utilización simple y ampliada a de la comunicación. Los controles se pueden llevar a cabo a través de las entradas lógicas en el bornero o a través del bus de comunicación. Cuando los controles se llevan a cabo a través de un bus, éstos están disponibles en una palabra y funcionan como un bornero que contiene únicamente entradas lógicas. Las funciones de aplicación se pueden asignar a los bits de esta palabra. Un mismo bit puede tener varias asignaciones.

Consignas

Fr1, SA2, SA3, dA2, dA3, MA2, MA3:

- borneros, terminal gráfico, Modbus integrado, CANopen integrado, tarjeta de comunicación, tarjeta Controller Inside.

Fr1b, para SEP e IO:

- borneros, terminal gráfico, Modbus integrado, CANopen integrado, tarjeta de comunicación, tarjeta Controller Inside

Fr1b, para SIM:

- borneros, accesible sólo si Fr1 = borneros

Fr2:

- borneros, terminal gráfico, Modbus integrado, CANopen integrado, tarjeta de comunicación, tarjeta Controller Inside y más/menos velocidad



Canal de control en el perfil [Perfil E/S] (IO)

Selección de un canal de control.

Es posible asignar un control o una acción, a un canal fijo seleccionando una entrada LI o un bit Cxxx:

- Si se selecciona, por ejemplo, LI3, esta acción siempre se iniciará a través de LI3 independientemente del canal de control conmutado.
- Si se selecciona, por ejemplo, C214, esta acción siempre se iniciará a través de CANopen integrado con el bit 14.

Independientemente del canal de control conmutado.

A un canal conmutable seleccionando un bit CDxx:

- Si se selecciona, por ejemplo CD11, esta acción se iniciará a través de
 - LI12 si el canal de borneros está activo,
 - C111 si el canal Modbus integrado está activo,
 - C211 si el canal CANopen integrado está activo,
 - C311 si el canal de tarjeta de comunicación está activo,
 - C411 si el canal de tarjeta Controller Inside está activo.
- Si el canal activo es el terminal gráfico, las funciones y controles asignados a los bits internos conmutables CDxx están inactivos.

Nota:

No es posible utilizar CD14 y CD15 en una conmutación entre 2 redes, ya que no corresponden a ninguna entrada lógica.

Servicios



- El variador Altivar 71 integra numerosas funciones de mantenimiento, supervisión y de diagnóstico.
- Funciones de test de variadores integradas con pantalla de diagnóstico en el Terminal gráfico extraíble.
- Imagen de las entradas/salidas.
- Imagen de la comunicación en los diversos puertos.
- Función de osciloscopio visualizable con el software de programación Power-Suite.
- Gestión del parque del variador gracias a los microprocesadores flasheables.
- Uso de estas funciones a distancia mediante la conexión del variador a un módem a través de la toma Modbus.
- Identificación de los elementos que constituyen el variador así como de las versiones de software.
- Históricos de los fallos con el valor de 16 variables cuando aparece el fallo.
- Flasheado de los idiomas del terminal.
- Se puede memorizar un mensaje de 5 líneas de 24 caracteres en el variador.

Terminal gráfico

Códigos de estado del variador:

- ACC: Rampa aceleración
- CLI: Limitación de intensidad
- CTL: Parada controlada tras pérdida de fase de red
- DCB: Frenado por inyección de corriente continua en curso
- DEC: Rampa deceleración
- FLU: Magnetización del motor en curso
- FST: Parada rápida
- NLP: Potencia no alimentada (sin red en L1, L2, L3)
- NST: Parada en rueda libre
- OBR: Deceleración autoadaptada



- PRA: Función Power removal activada (variador bloqueado)
- RDY: Variador listo
- RUN: Variador en marcha
- SOC: Corte aguas abajo controlado en curso
- TUN: Autoajuste en curso
- USA: Alarma de subtensión

Macro configuración

El variador Altivar 71 ofrece una programación rápida y sencilla por macro configuración que corresponde a aplicaciones o usuarios diferentes: marcha/paro, manu- tención, elevación, uso general, conexión a redes de comunicación, regulador PID, maestro/esclavo. Cada una de las configuraciones sigue siendo totalmente modificable.

Se distinguen siete macro configuraciones:

- marcha/paro (configuración de fábrica)
- manutención
- uso general
- elevación
- regulador PID
- bus de comunicación
- maestro/esclavo

2.5.3 Aplicaciones de Accionamientos de Velocidad Ajustable

Aplicaciones

La gama de variadores de velocidad Altivar 71, permite satisfacer las mayores exigencias gracias a los diferentes tipos de control motor y las numerosas funcionalidades integradas. Está adaptada a los accionamientos más exigentes:



- Par y precisión de velocidad a velocidad muy baja, dinámica elevada con control vectorial de flujo con o sin captador.
- Gama de frecuencia ampliada para los motores de alta velocidad.
- Puesta en paralelo de motores y accionamientos especiales gracias a la ley de tensión/frecuencia.
- Precisión de velocidad estática y ahorro energético para los motores síncronos de lazo abierto.
- Flexibilidad sin sacudidas para las máquinas excéntricas con el ENA System (Energy Adaptation System).

Las funciones del Altivar 71 aumentan el rendimiento y la flexibilidad de uso de las máquinas para múltiples aplicaciones

Oferta completa

La gama de variadores de velocidad Altivar 71 cubre las potencias de motor comprendidas entre 0,37 kW y 500 kW con tres tipos de alimentación:

- 200 . . . 240 V monofásica, de 0,37 kW a 5,5 kW.
- 200 . . . 240 V trifásica, de 0,37 kW a 75 kW.
- 380 . . . 480 V trifásica, de 0,75 kW a 500 kW.

Integra de forma estándar los protocolos Modbus y CANopen así como numerosas funciones. Estas funciones pueden ampliarse por medio de tarjetas opcionales de comunicación, entradas/salidas e interface de codificador.

Toda la gama cumple con las normas internacionales IEC-EN 61800-5-1, IEC-EN 61800-2, IEC-EN 61800-3, está certificada conforme a CE, UL, CSA, DNV, C-Tick, NOM 117, GOST y ha sido desarrollada para responder a las directivas sobre la protección del entorno (RoHS, WEEE, etc.).

El variador Altivar 71 se inserta en la cadena de seguridad de las instalaciones. Integra la función de seguridad “Power Removal” que prohíbe el arranque intempestivo del motor.

Esta función cumple con la norma sobre máquinas EN 954-1 categoría 3, con la norma sobre instalaciones eléctricas IEC-EN 61508 SIL2 y con el proyecto de norma de accionamiento de potencia IEC-EN 61800-5-2.

Compatibilidad electromagnética CEM

La incorporación de filtros CEM en los variadores ATV 71H...M3 y ATV 71H...N4 y la consideración de CEM facilitan la instalación y la conformidad de los equipos para el mercado CЄ, de forma muy económica.

Los variadores ATV 71H...M3X están disponibles sin filtros CEM. Puede instalar filtros opcionales para reducir el nivel de emisiones.

Otras opciones externas como resistencias de frenado, filtros y módulos regenerativos completan esta oferta.

Instalación

El variador Altivar 71 se ha desarrollado para optimizar el dimensionamiento de las envolventes (armarios, cofres, etc.):

- La parte de potencia, de grado de protección IP54, puede montarse fácilmente en el exterior de la envolvente con la ayuda del kit para montaje empotrado con envolvente estanco VW3 A9 5...

Este montaje permite limitar la emisión de calor en la envolvente o reducir su tamaño.

- Temperatura ambiente en la envolvente:
 - 50 °C sin desclasificación.
 - Hasta 60 °C utilizando el kit de ventilación de control VW3 A9 4... en función de los calibres y eventualmente desclasificando la corriente de salida.
 - Montaje yuxtapuesto.
- También puede instalarse en la pared respetando la conformidad con NEMA tipo



1 con el kit VW3 A9 2.., IP21 o IP31 con el kit VW3 A9 1...

Conclusiones Parciales

- La descripción del variador de velocidad tipo ALTIVAR, ofrece una caracterización adecuada para su explotación con fines académicos y propósitos investigativos teniendo en cuenta los aspectos de su explotación industrial.
- Se tienen los elementos fundamentales para realizar el procesamiento de datos a partir del mapeo del hardware del variador de velocidad Altivar.



Capítulo III

Fundamentos para el Procesamiento de Datos desde MATLAB.

3.1 Introducción

La interconexión a nivel de computadoras, nos permiten realizar experimentos y desarrollar proyectos en diferentes áreas (procesamiento de señales, control automático, etc.). Mostramos los pasos necesarios para realizar el procesamiento de datos, utilizando como interface de programación el MATLAB.

El MATLAB se ha convertido en un poderoso software de ingeniería, a través del cual podemos implementar diferentes técnicas de procesamiento de datos, aumentando este valor por la ventaja de poder acceder a procesos mediante la toma de datos de estos, procesarlos y modificar el comportamiento del proceso mediante la salida de datos. Se indica cómo se puede acceder a señales analógicas de entrada de un proceso, con las facilidades que ofrece el MATLAB.⁶

Es por ello que en este capítulo hacemos una descripción de modo general del entorno de trabajo de MATLAB como una importante herramienta para el procesamiento de las señales, nos daremos a la tarea de realizar una correcta fundamentación de todos los aspectos necesarios que contiene dicho programa para poder poner en práctica el tema fundamental de nuestro trabajo que es la creación de una interface gráfica para el procesamiento de datos en accionamientos eléctricos usando MATLAB.

⁶ *Cilento Augusto, Sistemas de control de motores en Tiempo Real mediante MATLAB, 2007.*

3.2 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, procesamiento de señal, todo ello en un entorno fácil para el usuario.

MATLAB presenta una aplicación para hacer simulaciones en tiempo real, la *toolbox* **Real Time Windows Target**. 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 inducción trifásico con rotor del tipo jaula de ardilla.⁶

3.2.1 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 ADAC, que es la que permite operar con señales de entrada y/o salidas analógicas y digitales.²

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:

⁶ *Cilento Augusto, Sistemas de control de motores en Tiempo Real mediante MATLAB, 2007.*

² *Pupo Bacallao, Gerardo Luis. Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos. Moa.2010.*



- 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 entradas 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.

3.3 Instalación del Software REAL-TIME WINDOWS TARGET

Para poder realizar adquisición de datos con MATLAB, se requiere instalar el software Real-Time Windows Target que forma parte del mismo MATLAB. Este software permitirá que las aplicaciones accedan al Kernel del sistema donde los ejecutables deben ser cargados y ejecutados. El driver del modo Kernel intercepta las interrupciones de tiempo del reloj del computador, manteniéndolas para operaciones del sistema Windows y asegura la ejecución de las aplicaciones en tiempo real. La instalación del Real-Time Windows Target, se realiza en el área de trabajo del MATLAB, escribiendo:⁷

```
rtwintgt –install
```

⁷ Vargas Tamani, Bruno. *Adquisición de Datos desde Matlab. Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú*



Para verificar las características resultado de la instalación del Kernel, puede escribir:

Rtwho

El Real-Time Windows Target requiere de un compilador C que no está incluido en el software de MATLAB. De no tener instalado algún compilador C en su computadora, es necesario instalar uno. Para elegir el compilador C, de entre los instalados debe escribir en el área de trabajo:

mex –setup

Ud. debe indicar que le muestre la lista numerada de compiladores instalados y luego debe escoger el que desee, escribiendo el número correspondiente. Normalmente se recomienda elegir si está en la lista el Microsoft Visual Studio C/C++.⁷

3.3.1 Análisis la sección de Procesamiento de Datos en MATLAB

Estos elementos permiten gobernar desde Matlab los diferentes subsistemas de entrada salida presentes en el dispositivo físico de adquisición de datos instalado en el PC. Matlab dispone de tres tipos de dispositivos objeto: *ai* (entrada analógica), *ao* (salida analógica) y *dio* (entrada-salida digital).

De esta forma, para iniciar una sesión de adquisición de datos en Matlab se deben seguir los siguientes pasos:

- *Crear un dispositivo objeto*, usando para ello las funciones de Matlab *analoginput*, *analogoutput* o *digitalio* según sea el caso de una entrada analógica, salida analógica o entrada-salida digital, respectivamente. El formato de uso de estas tres funciones es idéntico, debiéndose indicar la denominación del “driver” dispuesto por Matlab que es adecuado para el dispositivo físico de adquisición (en adelante, tarjeta de adquisición) de que se dispone.

⁷ **Vargas Tamani, Bruno. Adquisición de Datos desde Matlab. Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú**

También hay que indicar el identificador de dispositivo que se asocia a la tarjeta de P-2B.2 adquisición cuando se instala. Un ejemplo de definición de un dispositivo objeto de entrada analógica con una tarjeta de National Instruments (como la que se dispone en el laboratorio), con un identificador de dispositivo igual a 1, sería:

```
ai = analoginput('nidaq',1);
```

- *Configurar los canales de E/S*, mediante la función *addchannel*. En ella se hace referencia al dispositivo objeto creado anteriormente. Por ejemplo, para crear un canal de entrada analógico denominado “an01” usando el canal hardware 1 de la tarjeta en modo diferencial (patillas 5 y 6 del bloque conector en el laboratorio, según la tabla de especificaciones de la tarjeta), se escribiría:

```
an01 = addchannel (ai,1);
```

Si lo que se desea es añadir un grupo de canales hardware, por ejemplo del 0 al 3, se escribiría:

```
an0_3 = addchannel (ai,0:3);
```

- *Definir las propiedades* para ajustar la adquisición de datos a nuestras necesidades. Para conocer las propiedades del objeto que se ha definido, se puede hacer uso de la función *get(<nombre-objeto>)*, o bien mediante *<nombre-objeto>.[nombre-propiedad]*. En principio, las dos propiedades básicas que interesa ajustar son la frecuencia de muestreo (*SampleRate*) y el número de muestras por disparo (*SamplesperTrigger*). Se emplea la función *set*, donde se



hace referencia al dispositivo objeto empleado, la propiedad que se quiere definir y el valor que se le asigna a la misma:

```
set = (ai, 'SampleRate', Fs);  
set (ai,'SamplesPerTrigger',Samples);
```

- También es importante conocer cuál es el verdadero valor de la frecuencia de muestreo actual, lo hacemos con *ActualRate = get(ai,'SampleRate');*
- *Iniciar la adquisición o salida de datos* usando la función *start*. Tras este punto suele ser interesante hacer uso de la función *getdata* y *plot* o cualquier otra que nos permita la visualización o procesamiento de los datos adquiridos y almacenados con la función *getdata*.

```
start(ai)
```

```
getdata(ai);
```

- *Finalización del proceso* haciendo uso de las funciones *delete* y *clear*:

```
delete(ai)
```

```
clear ai
```

3.3.2 Procesamiento de Datos en Tiempo Real

Está funcionando en un sistema operativo del consumidor, la caja de herramientas de la adquisición de datos no puede asegurar respuesta a un acontecimiento dentro de un límite de tiempo máximo especificado. Para asegurar un alto rendimiento de procesamiento de la adquisición, la caja de herramientas maneja los datos adquiridos en bloques, que aumenta el estado latente asociado a cualquier punto de referencias



adquirido dado. Además, debe compartir recursos de sistema con otros usos y conductores en el sistema.

Si usted desea crear un lazo de control con el menos estado latente, y no requiere un tiempo de reacción determinista, usted puede realizar operaciones del punto usando el `getsample` y el `putsample`. En este caso, se adquieren y se procesan los datos como sigue:

1. Los datos se adquieren a través del software del vendedor de hardware.
2. Los datos entonces se dan apagado al motor de la caja de herramientas de la adquisición de datos.
3. La caja de herramientas hace los datos disponibles en MATLAB o Simulink.
4. Los datos se funcionan con el algoritmo del control que usted desarrolla en MATLAB o Simulink.
5. Los datos entonces se encaminan de nuevo al motor, a través del software del vendedor de hardware, y sobre el tablero.

Esto todavía no garantiza el tiempo de reacción de un lazo de control. Un hilo de rosca de una prioridad más alta puede tomar precedencia sobre el lazo de control.

La mayoría de las tarjetas basadas en PC de la adquisición de datos proporcionan un reloj interno, apto de la exactitud que se utiliza para establecer el paso de la adquisición de datos. Las tarjetas almacenan los datos que recogen en una memoria local, y después se transfieren las muestras a la memoria principal de la computadora (con interrupciones o el acceso directo de memoria). La sincronización de las muestras adquiridas de esta manera es extremadamente exacta, y estas tarjetas pueden garantizar que los datos adquiridos fueron obtenidos en la tarifa solicitada de la muestra, y que no se cayera ningunas muestras. La tarifa máxima del muestreo es gobernada por la tarjeta de la adquisición de datos, no la PC.



3.3.3 Ejemplo de la sección de adquisición de datos en Matlab

Este ejemplo ilustra los pasos básicos que usted toma durante una sección de adquisición de datos usando un objeto en la entrada análoga. Usted puede funcionar este ejemplo por daqdoc3_1 que mecanografía en la ventana del comando de MATLAB.

Crear el objeto del dispositivo: crean el objeto AI de la entrada análoga para una tarjeta de ADAC. Los adaptadores y las identificaciones instalados del hardware se encuentran con daqhwinfinfo.

```
g = gpib('iotech,0,2');
```

```
obj=icdevice('ADAC/5501MF-V');
```

```
AI = analoginput('winsound');
```

```
%AI = analoginput('nidaq','Dev1');
```

```
%AI = analoginput('mcc',1);
```

Agregar los canales: agregan dos canales al AI.

```
addchannel(AI,1:2);
```

```
%addchannel(AI,0:1); % For NI and MCC
```

Ver información del driver

```
Instrhwinfinfo(obj)
```

Conectar el instrumento

```
connect(obj)
```

Verificar los comandos de comunicación

```
get(obj,'InstrumentModel')
```



devicereset(obj)

geterror(obj)

Desconectar el instrumento desde el objeto y borrarlo

disconnect(obj)

delete(obj,g)

Desde el editor:

Se crea la function:

```
function init(obj)
```

```
% Toma el objeto interface
```

```
g=get(obj,'interface');
```

```
fclose
```

```
% Configurar la interface
```

```
set(g,'InputBufferSize',5000);
```

```
set(g,'OutputBufferSize',5000);
```

Verificando el código creado

```
g=gpib('iotech',0,2)
```

```
get(g,'InputBufferSize','OutputBufferSize');
```

Configuración de los valores característicos: configuran la tarifa del muestreo a 11.025 kilociclos y definen un segundo adquisición.

```
set(AI,'SampleRate',11025)
```

```
set(AI,'SamplesPerTrigger',22050)
```

Comenzar la adquisición: con esta acción, se puede comenzar con el proceso de adquisición de datos creado para este caso.



```
start(AI)
```

Mientras ocurre la adquisición: la pausa de MATLAB hasta que la adquisición termina o hayan transcurrido 3 segundos. Si transcurren 3 segundos, y un error ocurre entonces se procede con:

```
wait(AI,3);
```

Extraer los datos adquiridos del motor y trazar los resultados

```
data = getdata(AI);
```

Trazar los datos y etiquetar las figuras hechas.

```
plot(data),
```

Para ponerle además los nombres o las variables correspondientes a cada eje de coordenadas de la figura se utilizan los siguientes comandos:

```
xlabel('Samples')
```

```
ylabel('Signal (Volts)')
```

Limpiar: cuando ya no necesite el AI, se debe quitar de memoria y del espacio de trabajo de MATLAB:

```
delete(AI)
```

```
clear AI
```

Entrada activa de datos

La señal de entrada es una señal de tensión o corriente en tiempo real procedente de un accionamiento eléctrico. En esta versión parcial de programa, el bloque de la entrada análoga se configura para recibir datos en modo asincrónico tales que todos los datos

están protegidos y no se pierdan ningunos de ellos durante la simulación. El bloque se puede también configurar para funcionar en modo síncrono en situaciones donde solamente sean importantes los datos más recientes.

3.3.4 Lectura de Datos Analógicos

Luego de instalado el Kernel mediante el Real-Time Windows Target y elegido el compilador para crear los ejecutables usados en tiempo real, ya podemos hacer uso de las librerías de MATLAB para adquisición de datos. Para leer datos analógicos, se conectan esas señales analógicas a los puertos de entrada de la ADAC. Luego cree un modelo nuevo en Simulink, como se muestra en ejemplo en la figura 1.

De la librería Real-Time Windows Target seleccione y arrastre el bloque Analog Input, el cual nos permitirá la lectura de datos Analógicos. Para poder visualizar los datos adquiridos agregue un elemento Scope. Opcionalmente puede agregar un elemento To Workspace, para crear una variable que almacene los datos adquiridos en el área de trabajo del Matlab. Se muestra también la creación de una variable para las bases de tiempos.⁷

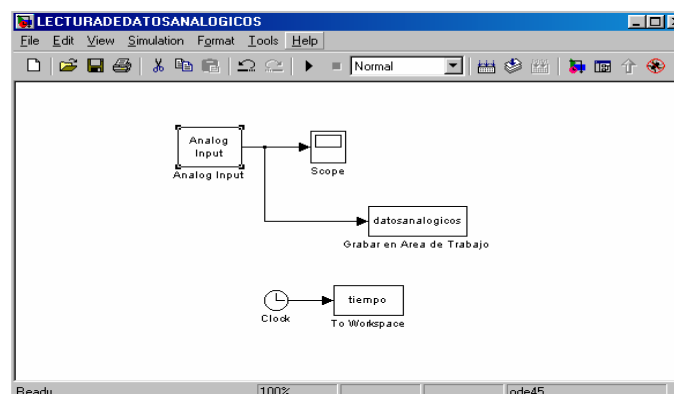


Fig.1: Creación de modelo de lectura de datos.

⁷ Vargas Tamani, Bruno. *Adquisición de Datos desde Matlab. Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú*

A continuación mostramos los pasos para realizar el proceso de adquisición de datos mediante MATLAB.

1. Debemos configurar el bloque Analog Input, seleccionémoslo y hagamos doble clic en él y aparecerá la pantalla de la figura 2.

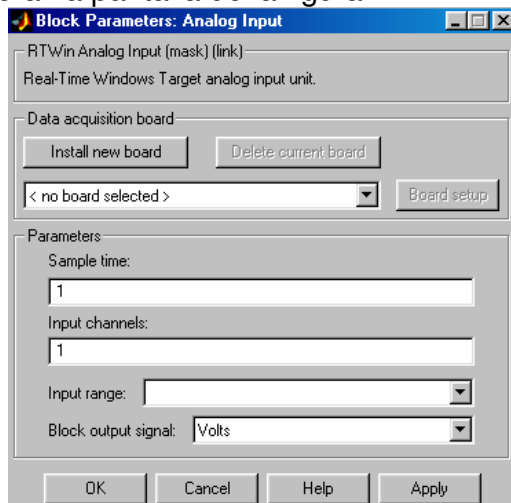


Fig. 2: Pantalla para configurar bloque Analog Input.

2. Primero debe seleccionar de las tarjetas de adquisición de datos instaladas en su computador, la que está usando. Observe la figura 3.

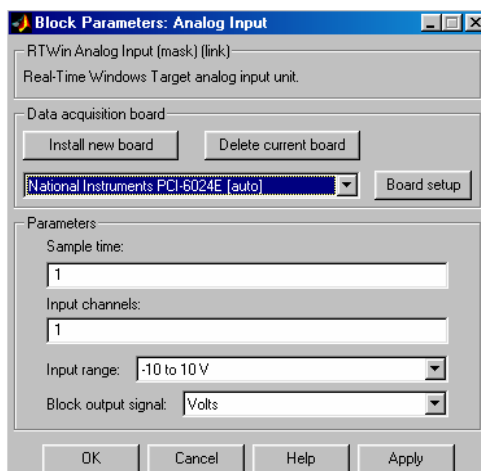


Fig. 3. Selección de ADAC/5501MF-V

3. Aparecerá en el modelo del proyecto el bloque Analog Input indicando la tarjeta seleccionada figura 4.

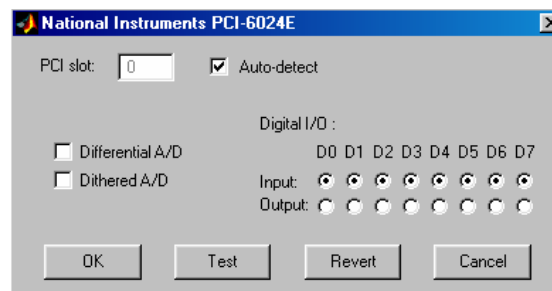


Fig. 4. Pantalla para test de tarjeta.

4. Pulse Test para verificar el estado de la tarjeta; en caso de estar en perfecto estado envía un mensaje con la pantalla mostrada en la figura 5.

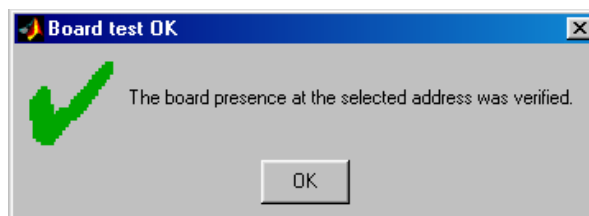


Fig. 5. Pantalla de verificación de DAQ.

5. En esa pantalla pulsamos OK y regresamos a pantalla de configuración (figura 3). Ahora configuramos el período de muestreo a utilizar y los canales a utilizar en la lectura de datos analógicos, como muestra la figura 6.

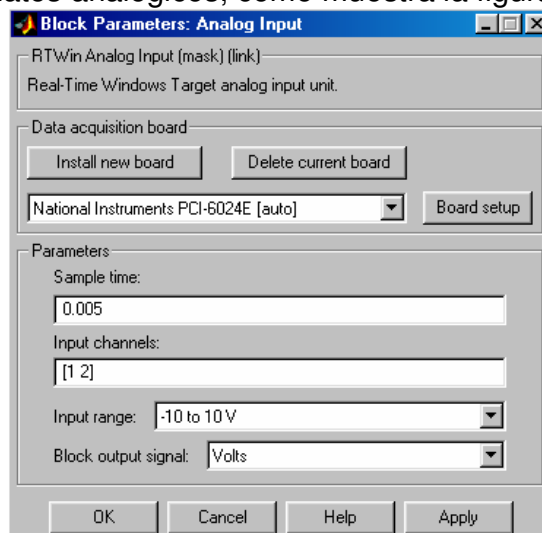


Fig. 6. Configura período de muestreo y canales de lectura.

Finalizamos pulsando OK.

6. Continúa la configuración de los parámetros de la presentación de la simulación. En la barra de menús Tools del modelo, seleccionar External mode control panel. Aparece la pantalla mostrada en la figura 7.

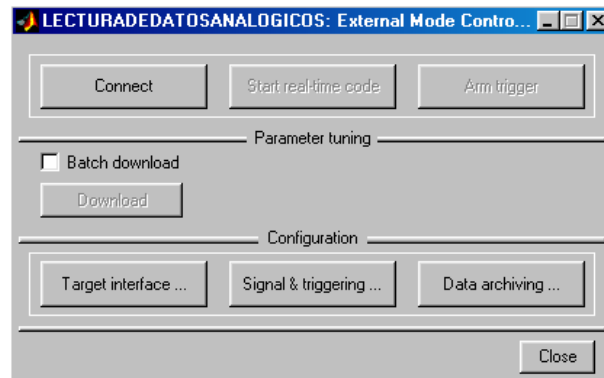


Fig. 7. Pantalla External Mode Control Panel.

7. Seleccionar la opción Target interface, escribir en el MEX-file: rtwinext. Observe la figura 8.

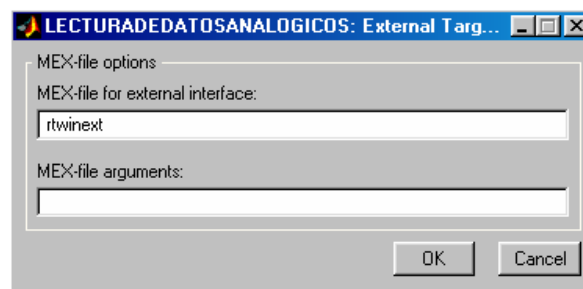


Fig. 8. Configuración del MEX-file.

Con OK, regresamos a pantalla External mode control panel, (figura 8.) seleccionar Signal & triggering.

8. Configurar el número de datos adquiridos que presentarán los bloques Scope y almacenarán los bloques To Workspace. Observe figura 9.

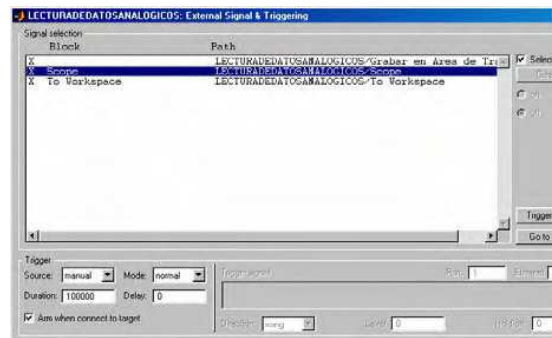


Fig. 9. Configurar número de puntos que presentarán los Scope.

9. Luego pulsamos Apply y Close para terminar esta configuración. Continúa ahora la configuración de los parámetros de simulación. En la barra de menús Simulation del modelo, seleccionar Solver y aparece pantalla de la figura 10.

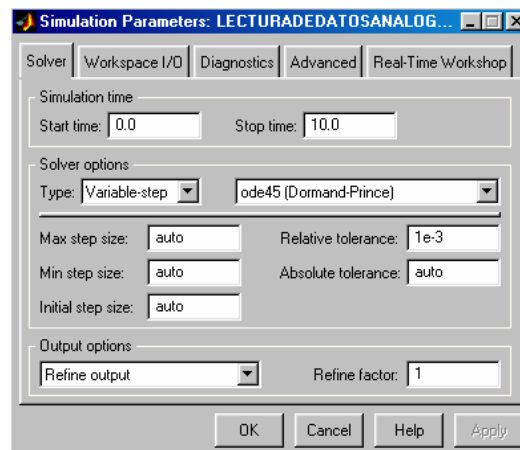


Fig. 10. Configurar solución de simulación

10. Seleccionamos tipo de solución Fixed-step y Ode5. Observe figura 11.

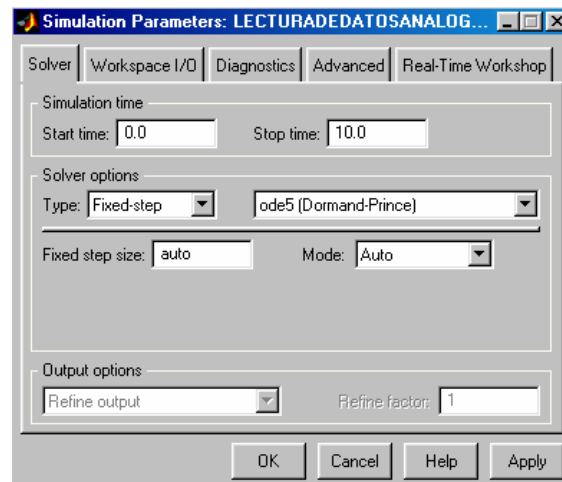


Fig. 11. Selección del modo de solución.

11. En la misma pantalla seleccionar Real-Time Workshop. Aquí configuraremos las opciones de creación y ejecución en tiempo real. Aparece inicialmente la pantalla de la figura 12.

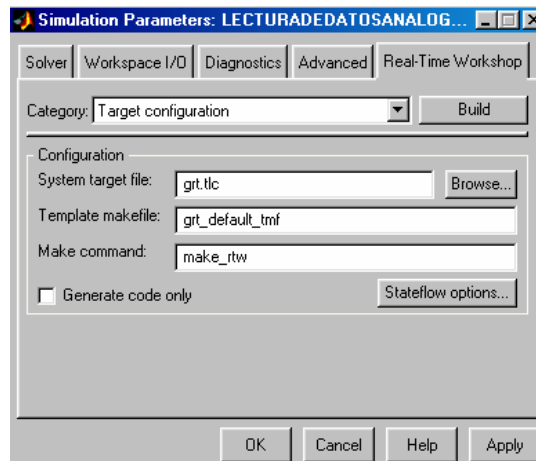


Fig. 12. Pantalla de configuración de opciones de creación y ejecución en tiempo real.

12. Al pulsar Browse, aparecen los diferentes archivos para el manejo de creación y ejecución en tiempo real. Seleccionar: rtwin.tlc. Observe figura 13.

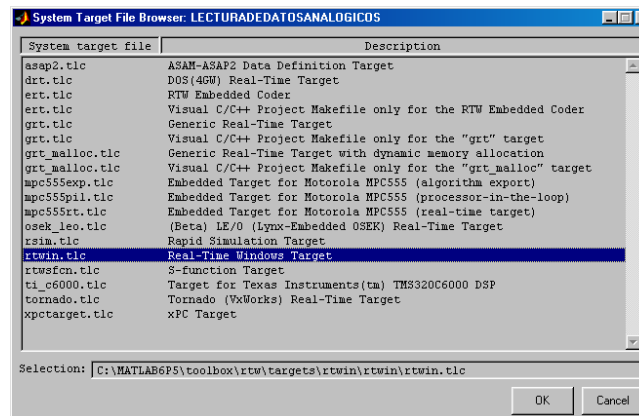


Fig. 13. Selección de manejador rtwin.tlc.

13. Pulse OK y aparece inicialmente la pantalla de la figura 14, que muestra la opción seleccionada.

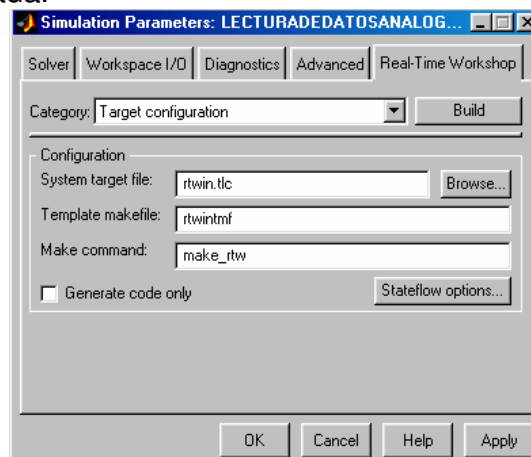


Fig. 14. Manejador rtwin.tlc para la DAQ.

14. Luego de pulsar OK, hemos terminado la configuración de la lectura de datos analógicos. Para realizar la adquisición de datos de lectura analógica; en la barra de menús Simulation del modelo, seleccionar External, como muestra la figura 15.

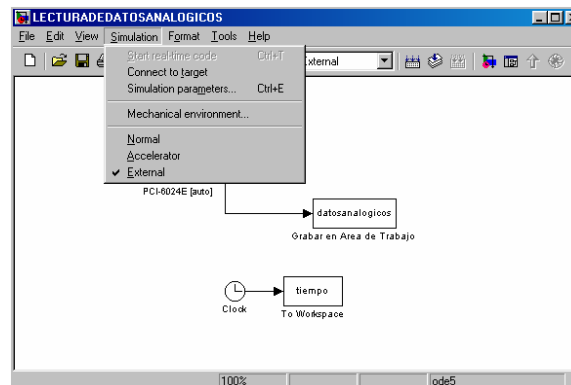


Fig. 15. Selección del modo External.

15. Luego debemos ordenar que se construya el modelo diseñado, para ello en la barra de menús Simulation del modelo, seleccionar Simulation Parameters y Build, como muestra la figura 16.

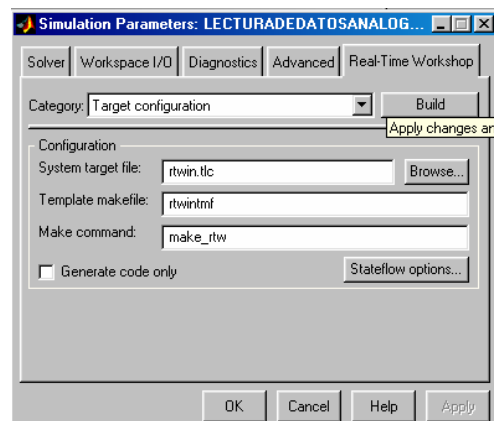


Fig. 16. Orden de construcción del modelo.

16. El proceso de construcción se puede observar en la pantalla del Matlab, mediante mensajes como se aprecia en la figuras 17.



```
MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB65\work\ADQUISICION DE DATOS\11AG...

-opts:ref
-incremental:no
-fullbuild
-nofaultlib
-version:2.00
-osversion:4.00
-subsystem:native,4.00
-export:Header
-export:Inquiry
-export:Disable
-export:Enable
-export:GetBoards
Creating library ..\LECTURADEDATOSANALOGICOS.lib and object ..\LECTURADEDATOSANALOGICOS.
### Created Real-Time Windows Target module LECTURADEDATOSANALOGICOS.rwd.
C:\MATLAB65\work\ADQUISICION DE DATOS\11AGOSTO2006\LECTURADEDATOSANALOGICOS_rtwin>
### Successful completion of Real-Time Workshop build procedure for model: LECTURADEDATOSAN.
>>
```

Fig. 17. Mensajes del proceso de construcción.

17. Durante el proceso de construcción se crean un grupo de archivos, como los mostrados en las figuras 18 y 19.

Nombre	Tamaño	Tipo
LECTURADEDATOSANALOGICOS_rtwin		Carpeta de archivos
LECTURADEDATOSANALOGICOS	9 KB	Rose Model
LECTURADEDATOSANALOGICOS.rwd	18 KB	Archivo RWD

Fig. 18. Carpetas generadas en construcción del modelo.

Nombre	Tamaño	Tipo
lex_svr.obj	13 KB	Archivo intermedio
LECTURADEDATOSANALOGICOS	1 KB	Archivo por lotes MS-DOS
LECTURADEDATOSANALOGICOS.c	7 KB	Archivo de código fuente de C
LECTURADEDATOSANALOGICOS.h	4 KB	Archivo de encabezado de C
LECTURADEDATOSANALOGICOS.mk	15 KB	Archivo MK
LECTURADEDATOSANALOGICOS.obj	7 KB	Archivo intermedio
LECTURADEDATOSANALOGICOS_data.c	1 KB	Archivo de código fuente de C
LECTURADEDATOSANALOGICOS_data.obj	1 KB	Archivo intermedio
LECTURADEDATOSANALOGICOS_dt.h	2 KB	Archivo de encabezado de C
LECTURADEDATOSANALOGICOS_private.h	3 KB	Archivo de encabezado de C
LECTURADEDATOSANALOGICOS_targ_data_map	3 KB	MATLAB M-file
LECTURADEDATOSANALOGICOS_types.h	1 KB	Archivo de encabezado de C
modelsources	1 KB	Documento de texto
rt_nonfinite.obj	2 KB	Archivo intermedio
rt_sim.obj	4 KB	Archivo intermedio
rt_stub.obj	4 KB	Archivo intermedio
rtw_proj.tmw	1 KB	Archivo TMW
updown.obj	12 KB	Archivo intermedio

Fig. 19. Archivos generados en construcción del modelo.

18. Antes de ejecutar el modelo de lectura de datos analógicos, debemos ordenar la conexión con la tarjeta de adquisición de datos. Debajo de la barra de menús del modelo, seleccionar el icono de conexión con la tarjeta DAQ, como muestra la figura 20.

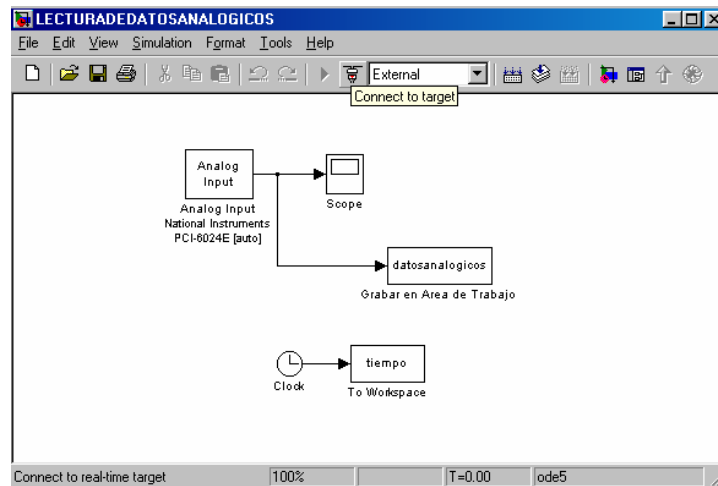


Fig. 20. Orden de conexión con la tarjeta de adquisición de datos.

19. La figura 21, muestra el proceso de conexión con la tarjeta de adquisición de datos.

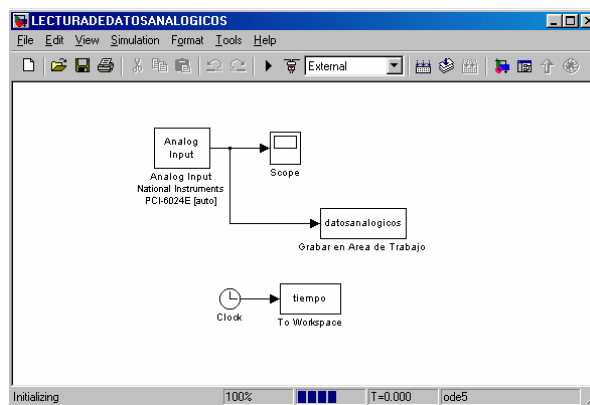


Fig. 21. Conexión con la tarjeta de adquisición de datos.

20. Para ejecutar el modelo debajo de la barra de menús del modelo, seleccionar el ícono de ejecución del modelo (Start simulation), como muestra la figura 22.

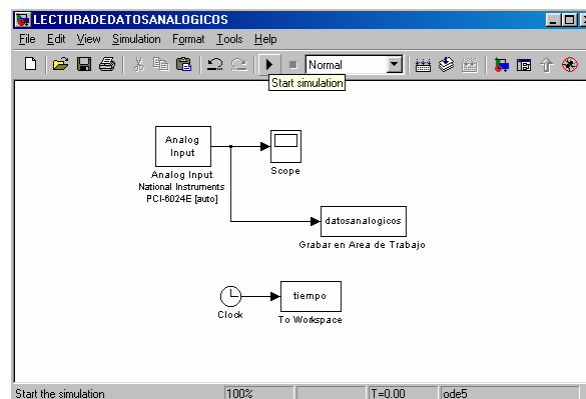


Fig. 22. Orden de ejecución del modelo.

21. Conforme se ejecuta el modelo, Ud. puede observar en el Scope del modelo las señales adquiridas, como muestra la figura 23.

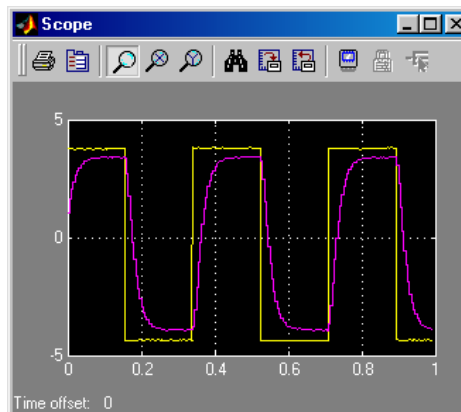


Fig. 23. Datos adquiridos que muestra el Scope.

22. Otra forma de obtener los gráficos de las señales adquiridas, es utilizar las variables almacenadas en el área de trabajo. La figura 24, muestra el gráfico de las señales adquiridas, planteadas a partir de las variables almacenadas en el área de trabajo.

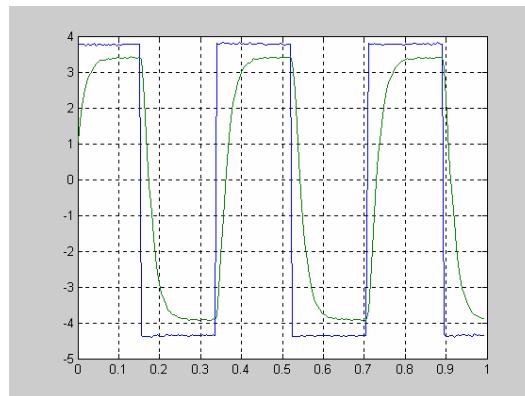


Fig. 24. Ploteo de las señales adquiridas, a partir de las variables almacenadas en el área de trabajo.

23. Podemos comparar las señales adquiridas por nuestro modelo de lectura de datos, comparando con las mismas señales medidas por medio de un osciloscopio, como muestra la figura 25.

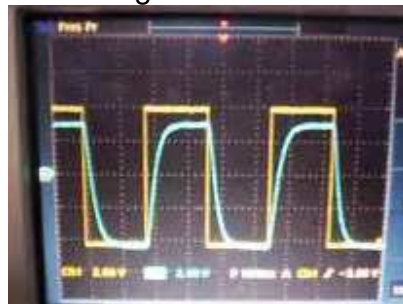


Fig. 25. Señales medidas por osciloscopio.

3.4 Mapeo de Memoria en MATLAB.

El MATLAB muestra la información del uso de la memoria, en la forma y el contenido que se utiliza corrientemente. Para esto es importante conocer su capacidad máxima cuando se salvan los datos en forma de arreglos.

Es necesario conocer durante el proceso de procesamiento de datos, la eficiencia en que se realizan, para esto es importante conocer los arreglos habilitados en la memoria en la cual interviene directamente la memoria física total (RAM).



La memoria habilitada para todos los arreglos está sujeta a los datos, el MATLAB, deriva estos números desde pequeños valores en la cual interviene la dirección de espacio virtual en MATLAB, y la capacidad total de memoria habilitada para el sistema. Toda la información es almacenada en forma de cadena y la memoria lo que hace es direccionar los registros que son escritos o leídos desde los comandos ejecutados por MATLAB.

Es decir que la dirección de espacio virtual está contenida en la memoria virtual total que utiliza MATLAB en su proceso.

En ocasiones se utilizan direcciones reservadas con el empleo de máquinas virtuales por ejemplo: *Java Virtual Machine*.

3.4.1 Mapa de Memoria para Señales y Parámetros

Tiene interés en este caso la estadística del uso de la memoria para el sistema Windows a 32 bits con las siguientes características:

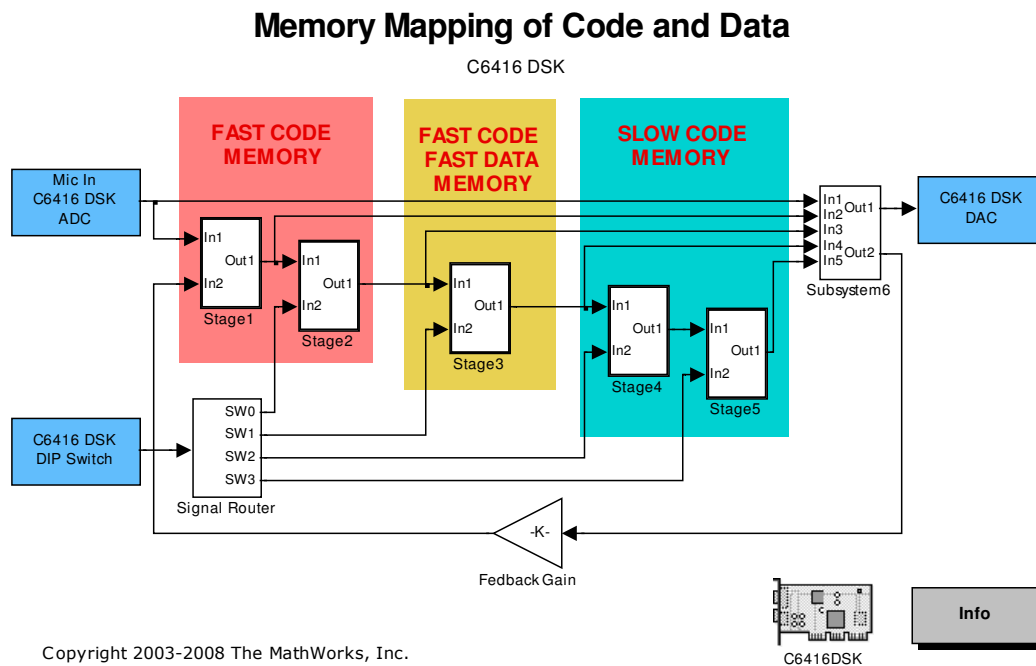
- Arreglo posible máximo.....677 Mb.
- Capacidad de memoria para todos los arreglos.....1 601 Mb.
- Memoria usada por MATLAB.....446 Mb.
- Memoria física.....3 327 Mb.

En nuestro trabajo no vamos a hacer uso de una máquina virtual, se propone utilizar únicamente los recursos propios del procesamiento de datos en MATLAB.

3.4.2 Mapeo de Memoria de Códigos y Datos.

Existen modelos sobre Simulink que demuestran cómo se posicionan los códigos generados en la memoria, necesitando para esto una licencia del modelo Real-Time-Workshop (RTW), o Proceso en Tiempo Real.

Este es el modelo de cómo hacer el mapeo de códigos y datos en la memoria usando la tarjeta C6416DSK



Modelo en Simulink para el mapeo de códigos y datos en memoria desde el MATLAB.

Como no se cuenta con la tarjeta, no será posible realizarlo desde el punto de vista físico.

Este tipo de proceso se utiliza en un demos usando el board de un DSP, es decir desde una tarjeta procesadora digital de señales.

3.4.3 Comunicación Basada en Registros.

Este tipo de proceso es usado dentro del toolbox de control de instrumentos.



La idea es leer desde los registros de memoria en una determinada dirección, previamente asignada a un código de registro en cada instrumento, por ejemplo A16 localizado en el espacio de memoria el cual contiene un instrumento HP E1432A que contiene la siguiente información:

Offset	Información
0	Registro ID
2	Registro Tipo de Instrumento
4	Estatus del Registro
6	Offset del Registro

Tabla de la Información contenida dentro del instrumento HP E1432A

En la memoria se manejan funciones y propiedades las cuales están basadas en operaciones de lectura y escritura.

Por ejemplo:

Se puede construir un objeto de fabricación de HP (Helen Poke) para tecnología VISA.

```
v = visa('agilent', 'VXI0::8::INSTR');
```

Es decir se crea el objeto, se conecta el objeto a tu instrumento para eso se procede de la siguiente forma:

```
fopen(v)
get(v, 'Status')
```

Se toma la característica del registro

```
get(v, 'MemorySpace')
ans =
A16/A24
```



Se ven entonces las características y propiedades del registro, la base y la dirección de la memoria:

```
get(v, {'MemoryBase', 'MemorySize'})  
ans =  
    '200000H'    [262144]
```

Lectura y escritura con la memoria de alto nivel

```
data = memread(v, 0, 'uint16', 'A16')  
data =  
    53247  
dec2hex(data)  
ans =  
CFFF
```

Lectura de un bloque de datos

```
data = memread(v, 0, 'uint16', 'A16', 2)  
data =  
    53247  
    20993
```

Escritura de la memoria de alto nivel

El comando MEMWRITE permite escribir en determinados registros o valores simples especificados en direcciones de la memoria utilizando los siguientes comandos:

```
original_Value = memread(v, 6, 'uint16', 'A16');  
memwrite(v, 40960, 6, 'uint16', 'A16');  
memread(v, 6, 'uint16', 'A16')
```

Estas operaciones de lectura y escritura para memorias de bajo nivel se realizan más rápido pero requieren del mapa de la memoria para ser leídas y escritas. Para esto se procede de la forma siguiente:

```
get(v, {'MappedMemoryBase', 'MappedMemorySize'})
```



La lectura de la información del ID del registro en el espacio A16 se programa de la siguiente forma:

```
mempeek(v, 0, 'uint16')
```

Para leer la información del dispositivo del instrumento se programa como sigue:

```
mempeek(v, 2, 'uint16')
```

El comando MEMPOKE permite escribir datos en el espacio de memoria, por ejemplo:

```
original_Value = mempeek(v, 6, 'uint16');  
mempoke(v, 45056, 6, 'uint16');  
mempeek(v, 6, 'uint16')
```

Cuando usted desea finalizar la operación de lectura y escritura en los registros puede utilizar los siguientes comandos de programación:

```
memunmap(v)  
get(v, {'MappedMemoryBase', 'MappedMemorySize'})
```

Si se finaliza la comunicación con el objeto, se debe desconectar y removerlo desde la memoria utilizando el espacio de trabajo asignado mediante los siguientes comandos:

```
fclose(v);  
delete(v);  
clear v
```

Es decir se cierra, se borra y se limpia el objeto.



3.5 Interfaz Gráfica de Usuario en MATLAB

Guide es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.⁸

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 GUI, (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.²

El proceso a seguir para el desarrollo de un programa mediante GUIDE, es que una vez que 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 para 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 interface.

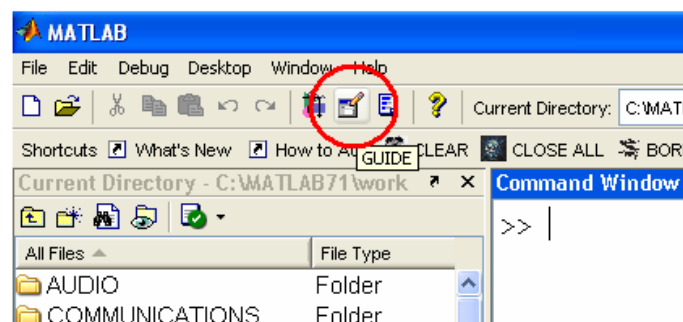
⁸ Barragán Guerrero, Diego Orlando. *Manual de Interfaz Gráfica de usuario en MATLAB*.

² Pupo Bacallao, Gerardo Luis. *Interface para el Control Inteligente de Instrumentos en Sistemas SCADA de Accionamientos Eléctricos*. Moa.2010.



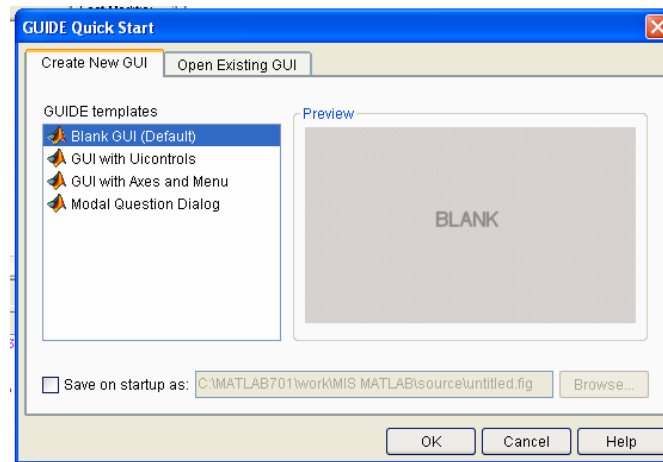
- Un archivo .M que es el que contiene el código con las correspondencias de los botones de control de la interfaz. Cada vez que se introduzca un elemento gráfico en él. FIG generará unas líneas de programa automáticamente asociadas a ese tipo de control. Estas líneas de programas son vacías, es decir, 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.

Iniciamos nuestro trabajo, con la primera acción que sería la ejecución del ícono de GUIDE como se muestra en la siguiente figura:



Ícono GUIDE.

De modo que al ser ejecutado este ícono la ventana de inicio resultante se presenta como sigue:



Ventana de inicio de GUI.

Para darle comienzo a nuestro programa, vamos a programar una pantalla donde podemos colocar el tema de nuestro diseño, nombre del programador.

Para esto, tenemos el siguiente código (copiar y pegar en un m-file):

```
function presentacion
%Autor: William Quesada Pupo
%Estudiante de Ing. Eléctrica
%*****
% Interface gráfica para el procesamiento de datos
%*****
```

Programa para GUIDE- Simulink.

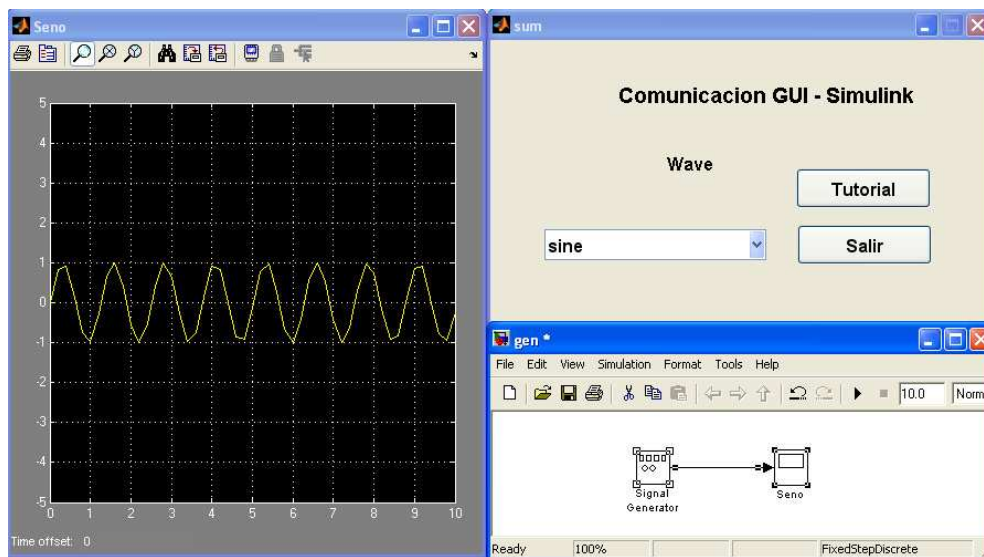
Al comenzar con la ejecución del programa, veremos que la primera ventana en abrirse será la que nos muestra la supervisión y el control de instrumentos desde el GUIDE, estableciendo así la comunicación y el control de algunas acciones que se visualizan en la interface.



Ventana de inicio de la interface en GUI

Con este programa se pretende mostrar la interacción entre una Interfaz Gráfica y Simulink.

Este sencillo programa controla un parámetro del bloque *Signal_Generator*: la forma de onda. En la parte de inicialización del programa (`guide_simulink_OpeningFcn`) se escribe lo siguiente:



Entorno programa GUI_SIMULINK para la comunicación.

```
find_system('Name','gen');  
open_system('gen');  
set_param('gen/Signal Generator','Wave','sine');  
set_param(gcs,'SimulationCommand','Start');
```

Estos comandos son los primeros que se ejecutan al abrir la interfaz gráfica. *Find_system* y *open_system* son para comprobar si existe el programa en simulink y para abrirlo. La sentencia para escribir en los bloques de simulink es *set_param*, que se usa para establecer en el parámetro *Wave form* del bloque *Signal_Generator* la onda *sine*. La sintaxis de *set_param* es:

```
set_param('nombre_del_programa/nombre_del_bloque','parámetro','valor')
```

El comando `set_param(gcs,'SimulationCommand','Start')` es para iniciar la ejecución del programa en simulink.

La programación del *Pop-up menú* (etiquetada el en campo *Tag* como *wave*) es como sigue:

```
onda = get(hObject,'Value');  
if onda==1  
    set_param('gen/Signal Generator','Wave','sine');
```



```
set_param(gcs,'SimulationCommand','Start');

elseif onda==2
    set_param('gen/Signal Generator','Wave','square');
    set_param(gcs,'SimulationCommand','Start');
elseif onda==3
    set_param('gen/Signal Generator','Wave','sawtooth');
    set_param(gcs,'SimulationCommand','Start');
else
    set_param('gen/Signal Generator','Wave','random');
    set_param(gcs,'SimulationCommand','Start');
end.
```

El programa de la supervisión lo realizamos de la siguiente forma.

COMUNICAR en pushbutton1.

```
function pushbutton1_Callback(hObject, eventdata, handles)
comunicar=get(handles.popupmenu1,'value');
switch comunicar
    case 1
        figsimulink;
    case 2
        dispositivos;
    case 3
        tutoria;

end
```

Se ejecuta la supervisión presionando el botón.

```
function supervision_Callback(hObject, eventdata, handles)
% hObject      manipula al pushbutton3 (see GCBO)
% eventdata    reservado para ser definido en una versión futura de MATLAB
% handles      manipulador con la estructura y los datos del usuario (see GUIDATA)

find_system('Name','gen');
open_system('gen');
set_param('gen/Signal Generator','Wave','sine');
set_param(gcs,'SimulationCommand','Start');
```

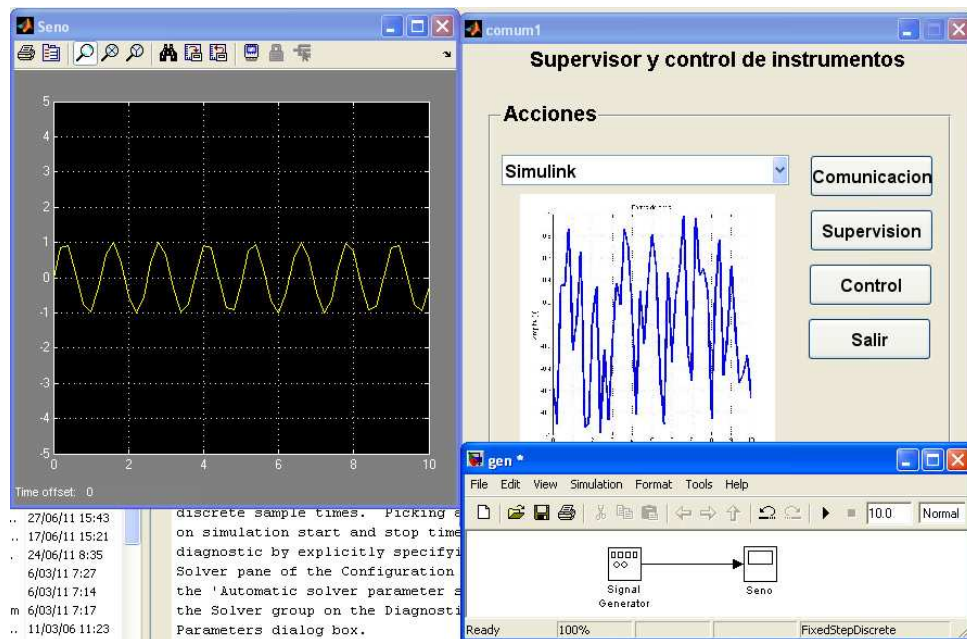
Supervisar el comportamiento variable desde Simulink
axes(handles.axes1);

```
a=imread('seno1.jpg');  
imshow(a)
```

SALIR en pushbutton 3.

```
function pushbutton3_Callback(hObject, eventdata, handles)  
close;
```

Obteniendo como resultado de esta programación la ventana GUI siguiente:



Entorno programa GUI_SIMULINK para la Supervisión de la señal obtenida del generador.

Este es otro programa basado también en GUI para la establecer la comunicación con otros protocolos, dispositivos y tutoriales que se encuentran la interface.



Entorno programa GUI para la comunicación con otros objetos.

Para comenzar con la inicialización del código el cuál por defecto sugiere no corregir comienza del modo siguiente:

```
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @protocolos_OpeningFcn, ...
    'gui_OutputFcn', @protocolos_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

Finalizando el Código de inicialización - NO CORREGIR

```
function varargout = comum1_OutputFcn(hObject, eventdata, handles)
```

Para conseguir la línea de comando que trae por defecto salida de la estructura del manipulador:

```
varargout{1} = handles.output;
```



```
axes(handles.axes1);  
a=imread('comunicación.jpg');  
imshow(a)
```

Al ACEPTAR en el pushbutton1 y establecer las demás opciones, debemos realizar la siguiente programación:

```
function pushbutton1_Callback(hObject, eventdata, handles)  
acceptar=get(handles.popupmenu1,'value');  
switch aceptar  
    case 1  
        protocolos;  
    case 2  
        dispositivos;  
    case 3  
        tutoria;  
  
end
```

Al efectuar el pushbutton 3 correspondiente a la opción Salir, con la idea de salir o de cerrar la ventana se programa como sigue:

```
function pushbutton3_Callback(hObject, eventdata, handles)  
% hObject      manipula al pushbutton3 (see GCBO)  
% eventdata    reservado para ser definido en una versión futura de MATLAB  
% handles      manipulador con la estructura y los datos del usuario (see GUIDATA)  
  
close;
```

Cuando es Aceptada, en la ventana o figura anterior la opción de protocolos se ejecuta automáticamente otro ventana mostrada a continuación, en la cual se establece una comunicación con la tarjeta de sonido integrada en la placa madre de la PC, y que es utilizada por MATLAB con el propósito de visualizar la posibilidad de adquirir o procesar esta señal. De modo que la figura que sigue nos permite leer y escribir sobre la tarjeta.



Entorno del programa GUI para la comunicación con otros objetos.

La programación para la misma es la siguiente:

Para LEER se procede in pushbutton1 .

```
function pushbutton1_Callback(hObject, eventdata, handles)
leer=get(handles.popupmenu1,'value');
switch leer
    case 1
```

VISUALIZAR SEÑAL DESDE UNA ENTRADA ANALOGA DE LA TARJETA DE SONIDO

El comando siguiente crea un objeto del dispositivo, el objeto de entrada análoga para la tarjeta de sonido (ai):

```
ai = analoginput('winsound');
```

Se adicionan los canales para ai.

```
addchannel(ai,1:2);
```

Configura valores de las propiedades, configurando el muestreo a 44.1 kHz y recolectando un segundo de datos (44,100 muestras) para un canal.

```
set(ai,'SampleRate',44100)
```



```
set(ai,'SamplesPerTrigger',44100)
```

Inicia la adquisición, cuando todos los datos son adquiridos automáticamente se detiene la ejecución de la captura.

```
start(ai)
wait(ai,2)
data = getdata(ai);
axes(handles.axes1);
plot(data)
```

Limpia la memoria, cuando la misma es muy grande, el ai necesita remover los datos desde la memoria y lo pone en un workspace.

```
delete(ai)
clear ai

case 2
    %
    axes(handles.axes1);
a=imread('esquema.jpg');
imshow(a)

case 3
    %ethernet;
case 4
    %modbus;
case 5
    %usb;
case 6
    %gpib;
    g=gpib('ni',0,2);
    fopen(g);
    fprintf(g,'Curve?')
    readasync(g,512);
    flushinput(g);
    fclose(g);

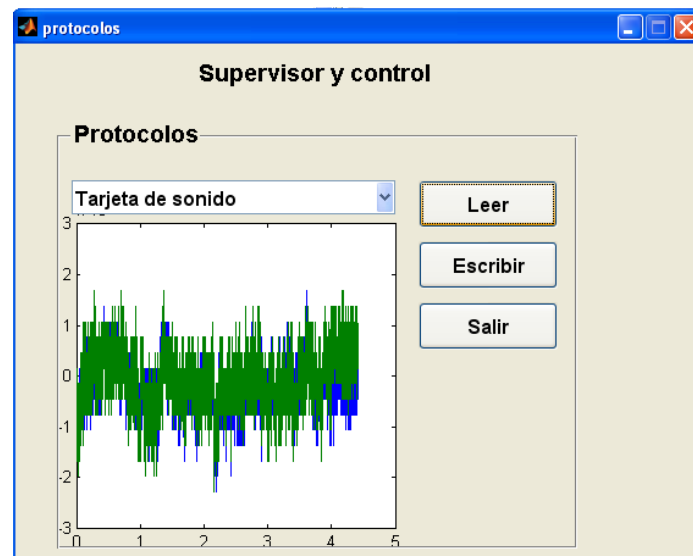
case 7
    %i-eeer-s485;

end
```

Usando puerto serie

```
s=serial('COM1','InputBufferSize',2500);  
fopen(s)  
fprintf(s,'Curve?');  
readasync(s);  
data=fread(s,2500);  
fclose(s);
```

Obteniendo como resultado de esta operación la ventana siguiente



La ventana muestra la señal de lectura obtenida de la tarjeta.

Para escribir:

ESCRIBIR in pushbutton2.

```
function pushbutton2_Callback(hObject, eventdata, handles)  
escribir=get(handles.popupmenu1,'value');  
switch escribir  
case 1
```

LA ENTRADA ANALOGA DE LA TARJETA DE SONIDO ESCRIBE UNA SEÑAL A 8000 MUESTRAS POR SEGUNDO.

```
ai = analoginput('winsound');  
addchannel(ai, [1 2]);  
set(ai, 'SampleRate', 8000);
```

El objeto (ai) también se configura para adquirir datos inmediatamente.



```
set(ai, 'SamplesPerTrigger', 3000);  
set(ai, 'TriggerType', 'immediate');
```

```
start(ai);
```

El tamaño de los datos será el número de muestras por disparo por el número de canales.

```
[data,time] = getdata(ai);  
size(data)
```

Los datos se pueden plotear contra el tiempo con el disparador actuando en un tiempo = 0. Como se puede ver, tomó 0.375 segundo para adquirir los datos. Esto se calcula tomando el ratio $\text{SamplesPerTrigger}/\text{SampleRate}$.

```
plot(time,data);  
zoom on;  
title('Immediate Trigger');  
xlabel('Relative time in seconds.');
```

```
ylabel('Data in volts');
```

Se configuremos el objeto de la entrada análoga (ai), para adquirir 3000 muestras con dos disparadores inmediatos.

Para comenzar con el objeto de la entrada análoga.

```
set(ai, 'TriggerType', 'immediate');  
set(ai, 'SamplesPerTrigger', 1500);  
set(ai, 'TriggerRepeat', 1);
```

```
start(ai);
```

El segundo comando extrae las segundas 1500 muestras del motor de la adquisición de datos.

```
[data1,time1] = getdata(ai);
```

```
[data2,time2] = getdata(ai);
```

Podemos plotear los datos contra el tiempo con el primer disparo que ocurre en un tiempo = 0, para eso escribimos:

```
plot(time1,data1, 'Color', 'red');  
hold on
```



```
plot(time2,data2, 'Color', 'blue');  
zoom on  
title('Immediate Triggers - Using TriggerRepeat=1');  
xlabel('Relative time in seconds.');
```

```
ylabel('Data in volts');  
hold off
```

Le damos entonces comienzo el objeto de la entrada análoga, con el comando para el COMIENZO.

```
set(ai, 'SamplesPerTrigger', 3000);  
set(ai, 'SampleRate', 8000);  
set(ai, 'TriggerType', 'manual');  
set(ai, 'TriggerRepeat', 0);
```

```
start(ai);
```

El motor de la adquisición de datos, funcionará tan pronto como se haga público el comando COMENZAR. Sin embargo, las muestras de los datos no serán almacenadas en el motor de la adquisición de datos hasta que se publique el comando del DISPARO. Por lo tanto, el número de las muestras disponibles del motor de la adquisición de datos será cero.

```
get(ai, 'Running')  
get(ai, 'SamplesAvailable')
```

Ahora daremos comienzo el disparo manual.

```
trigger(ai)  
[data,time] = getdata(ai);
```

```
plot(time,data);  
zoom on;  
title('Manual Trigger');  
xlabel('Relative time in seconds.');
```

```
ylabel('Data in volts');
```

Se configuremos el objeto de la entrada análoga para adquirir un total de 3000 muestras. 1000 muestras serán adquiridas antes de que ocurra el disparo manual y las otras 2000 serán adquiridas después de que ocurra el disparo.

```
set(ai, 'TriggerType', 'manual');  
set(ai, 'SamplesPerTrigger', 3000);  
set(ai, 'TriggerDelay', -1000);
```



```
set(ai, 'TriggerDelayUnits', 'samples');
```

Se comienza el objeto de la entrada análoga y el motor de la adquisición de datos comenzará a funcionar.

```
start(ai);  
status = get(ai, 'Running')  
pause(1.5);
```

Ahora accionamos el objeto de la entrada análoga, y después esperamos hasta dos segundos para que la adquisición termine.

```
trigger(ai);  
wait(ai,2)
```

Los datos se pueden plotear contra el tiempo con el disparador actuando en el tiempo = 0. Por lo tanto, pretriggered datos serán ploteados con un valor negativo del tiempo y los datos adquiridos después de que el disparador sean ploteados con valor positivo del tiempo.

```
[data,time] = getdata(ai);  
  
plot(time,data);  
zoom on;  
title('Immediate Triggers');  
xlabel('Relative time in seconds.');
```

En el orden para que este ejemplo tenga éxito, el micrófono de la adquisición de datos se debe sujetar de la señal mayor de .013 V para entonces activar el disparador. Si no el ejemplo fallará, volviendo un tiempo fuera y un gráfico en blanco.

```
set(ai, 'TriggerType', 'software');  
set(ai, 'TriggerRepeat', 0);  
set(ai, 'TriggerCondition', 'rising');  
set(ai, 'TriggerConditionValue', 0.013);  
set(ai, 'TriggerChannel', ai.channel(1));
```

Capturar un segundo valor de datos antes de que ocurra el disparo del motor de la adquisición de datos el cual esperará 2 segundos para que esta condición sea devuelta antes de parar.

Para comenzar también con el objeto de la entrada análoga.



```
set(ai, 'TriggerDelay', -1);  
set(ai, 'TriggerDelayUnits', 'seconds');  
set(ai, 'TimeOut', 2);
```

```
start(ai)
```

El software espera que el micrófono dispare para entonces adquirir los datos.

```
disp('You must supply a signal greater than 0.013 volts to activate the trigger');  
wait(ai,10);  
[data,time] = getdata(ai);
```

```
plot(time,data);  
zoom on;  
title('Software Trigger');  
xlabel('Relative time in seconds.');
```

```
ylabel('Data in volts');
```

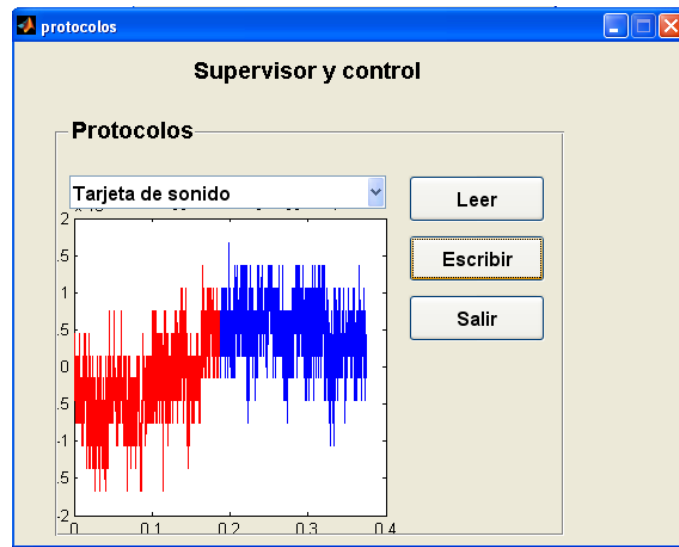
Ahora borrar el objeto de la entrada análoga.

```
delete(ai)  
case 2  
case 3  
case 4  
case 5  
case 6  
case 7  
end
```

Se ejecuta presionando sobre el boton pushbutton3.

```
function pushbutton3_Callback(hObject, eventdata, handles)  
close;
```

Obteniendo como resultado de esta operación la ventana siguiente:



La ventana muestra la señal de Escritura obtenida de la tarjeta.

Con este sencillo programa se mostrará la forma de leer y escribir valores en el puerto paralelo (PP) de nuestro PC a través de una interfaz gráfica.

Antes de empezar con el diseño de la interfaz, es necesario conocer como MATLAB trabaja con el puerto paralelo.

En el *command window* ejecutamos:

```
out = daqhwinfo;  
out.InstalledAdaptors
```

Lo que dará como resultado en el caso de mi PC:

```
ans =  
'parallel'  
'winsound'
```

Una vez que Matlab ha reconocido el PP, ejecutamos:

```
daqhwinfo('parallel')  
ans =  
AdaptorDllName: [1x50 char]  
AdaptorDllVersion: '2.7 (R14SP3)'  
AdaptorName: 'parallel'  
BoardNames: {'PC Parallel Port Hardware'}  
InstalledBoardIds: {'LPT1'}
```

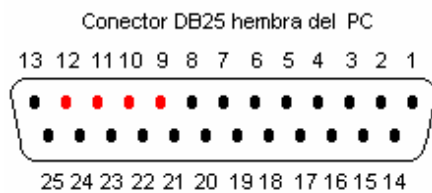


ObjectConstructorName: {' ' 'digitalio('parallel','LPT1')'}

La última línea indica el nombre de la entrada digital:

digitalio('parallel','LPT1').

Como sabemos, el PP puede ser de entrada y salida. La siguiente figura muestra la distribución de pines del PP:



Numeración de pines del PP.

En este conector:

- 8 pines son para salida de datos (bits de DATOS), y van desde el pin 2 (bit menos significativo) hasta el pin 9 (bit más significativo).
- 5 pines son de entrada de datos (bits de ESTADO). Estos pines son: 15, 13, 12, 10 y 11, del menos al más significativo.
- 4 pines son de control (bits de CONTROL). Tienen la característica de ser bidireccionales es decir que los puedes utilizar tanto de entrada como de salida.

Estos pines son: 1, 14, 16 y 17, del menos al más significativo.

Sin embargo, configurando en la BIOS del PC (accesible en unos PCs con la tecla *F12*, en otros con la tecla *Del* o *Supr*, y a veces con *F10*) el puerto paralelo (LPT1) como de entrada y salida, es posible usar los pines del 2 al 9 como de entrada.

Para determinar cuántos pines podemos usar en el PP, ejecutamos:

```
parport = digitalio('parallel','LPT1');  
hwinfo = daqhwinfo(parport)  
hwinfo =  
AdaptorName: 'parallel'  
DeviceName: 'PC Parallel Port Hardware'  
ID: 'LPT1'  
Port: [1x3 struct]  
SubsystemType: 'DigitalIO'
```



TotalLines: 17
VendorDriverDescription: 'Win I/O'
VendorDriverVersion: '1.3'

Como se puede ver en la información, 17 de los 25 pines del PP se los puede utilizar como I/O. Los restantes pines son *tierra*.

Una vez creada la entrada digital del PP, lo que sigue es asignar que pines serán para entrada y cuales para salida. Usamos la función *addline*, cuya sintaxis es:

```
dato2= addline(parport,0:7,'in'); %Para valores de entrada  
dato = addline(parport,0:7,'out'); %Para valores de salida
```

Y se obtiene el dato de entrada con `dato3=getvalue(dato2)`. Se asigna el dato al puerto con: `putvalue(dato,255)`.

Con esta información se puede realizar la interfaz gráfica.

Como se puede notar, se inicializa el manejador *handles.ada* en 0. Este manejador almacena el valor ingresado por el usuario para ser escrito en el PP. Se inicializa a 0 porque es el valor que presenta la interfaz en el *edit text* de salida al empezar a ejecutarse el programa.

```
function envio_Callback(hObject, eventdata, handles)  
if handles.ada>255 || handles.ada<0 || isnan(handles.ada)  
errordlg('Número fuera de rango','ERROR');  
set(handles.dato_out,'String','0');  
handles.ada=0;  
end  
diego=digitalio('parallel','LPT1');  
dato=addline(diego,0:7,'out');  
putvalue(dato,handles.ada);  
guidata(hObject, handles);
```

La sentencia *if* se encarga de corregir errores de ingreso de datos. Los valores deben estar entre 0 y 255 y deben ser valores numéricos. En caso de ingresar un dato fuera de rango (como -9 o a) se presenta un mensaje de error y se escribe en el puerto el valor de 0.

Las líneas de código siguiente inicializan la entrada digital, asignan los pines de salida y escriben el valor en el PP.



```
function recibir_Callback(hObject, eventdata, handles)
judas=digitalio('parallel','LPT1');
dato2=addline(judas,8:12,'in');
dato3=getvalue(dato2);
dato4=binvec2dec(dato3);
set(handles.dato_in,'String',dato4);
guidata(hObject, handles);
```

De modo que poniendo en práctica toda esta información podemos hacer una interface gráfica para la lectura y escritura de puerto paralelo.

Conclusiones Parciales:

- La interface desarrollada, permite las acciones de comunicación MATLAB - Simulink, y lectura y escritura de puertos (serie, paralelo) y tarjetas (sonido del PC).
- Las comunicaciones con los objetos a través del MATLAB necesitan de sus respectivos drivers.
- Las acciones de control que ejerce la interface sobre los elementos del modelo montado en Simulink deben ser lo más simplificado posible.
- Los elementos implementados en esta interface, permiten la gestión de procesamientos de datos, que tiene como base un generador de señales, sirve como herramienta para el manejo de circuitos de mando de inversores sinusoidales de tensión modulados, porque se elaboran y comunican señales tipo seno, onda cuadrada, diente de sierra y aleatorias.

Capítulo IV

Evaluación Económica del Trabajo

4.1 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 una 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 la interface gráfica, reduce mucho los costos por mano de obra, brinda ventajas a la hora de evaluar el estado del proceso.

Desde el punto de vista económico la implementación de estos sistemas inteligentes asegura desde un inicio el desempeño óptimo del proceso, el montaje de una estación con todos los equipos vinculados con el software, por la cual resultará una inversión suave ya que contamos con los equipos para poner en marcha esta interface, como por ejemplo el variador de velocidad Altivar 71 y se cuenta además con el programa.

4.2 Cálculo de los Gastos de Explotación

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

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

$$CTI = 705 + 800 = 1505.00 \text{ \$USD}$$

Donde:

$$\Sigma CIC = \text{Costo total de la instrumentación de campo}$$



ΣCV = Costo total de equipos y medios de explotación.

Costo total general sería:

$$CTG = CTI + EPA$$

$$CTG = 1505 + 225.75$$

De modo que el CTG es de 1730.75 \$USD

donde:

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

4.3 Gastos de Inversiones

A continuación el costo del instrumento de campo que utilizamos para la puesta en marcha de la interface.

El variador de velocidad ALTIVAR 71 HU40N4 5 hp, cuesta \$705.00 USD



El costo de la PC es de unos \$800.00 USD



Teniendo en cuenta que un programa similar cuesta de 6 000.00 a 10 000.00 \$USD. El monto total de la inversión sería de 7730.75 \$USD, que representarían unos 6 735.55 CUC.

4.4 Cálculo del Tiempo de Amortización

La amortización es la reducción parcial de los montos de una deuda en un plazo determinado de tiempo.

Se tiene que
$$T_{amortización} = \frac{Inversión}{EnergíaAhorrada}$$

Donde:

$T_{amortización}$: es el tiempo de recuperación de la inversión.

Inversión: es lo que se invierte en la compra del producto.

Energía ahorrada: es la energía que se ahorra con el producto que se propone comprar en (CUC).

Conclusiones Parciales

- El costo de elaboración de la interface gráfica está en el orden 6 735.55 \$ CUC de modo que nuestra inversión se amortiza en un tiempo de 1,5 años.
- Teniendo en cuenta que contamos con todos los equipos necesarios para realizar este programa, estaríamos ahorrándole al país el valor de esta inversión.



Conclusiones Generales

- La interface gráfica elaborada desde el GUI del MATLAB permite el procesamiento de datos a través de gestiones de comunicación, supervisión y control de dispositivos u objetos contenidos sobre Simulink, tarjetas y lectura y escritura de puertos del PC.
- Las comunicaciones con los objetos a través del MATLAB necesitan de sus respectivos drivers.
- Las acciones de control que ejerce la interface sobre los elementos del modelo montado en Simulink deben ser lo más simplificado posible.
- Los elementos implementados en esta interface permiten la gestión de procesamientos de datos, que tiene como base un generador de señales, sirve como herramienta para el manejo de circuitos de mando de inversores sinusoidales de tensión modulados, porque se elaboran y comunican señales tipo seno, onda cuadrada, diente de sierra y aleatorias.



Recomendaciones

- Utilizar el presente trabajo como guía técnica y tutorial.
- Continuar elaborando con más funciones la interface gráfica a partir de la adquisición de un hardware más completo con vista al control de otros instrumentos.



Bibliografía

BOSE, B. K., *Power Electronics and AC Drives*. Prentice Hall. 1986.

BOSE, B. K. *Power Electronics and Variable Frequency Drives*. Prentice Hall. 1997.

BODSON J. *Digital control improves variable speed drives*. EPE Journal. Vol. 2. No 4. December 1992.

CRUZ BLANCO, YUNIER. *Propuesta de Variadores de Velocidad en el Primer Impulso del Combustible de la CTE de Felton*. Moa. 2009

CHACON, DIJORT, J.CASTRILLO, *Supervisión y control de procesos*, 2001-02.

COSTA MONTIEL ANGEL. Capítulo III: *convertidores estáticos de frecuencia*.

COLECTIVO DE AUTORES. *Instrumentación y control Automático*, Universidad Nacional de Cuyo Facultad de Ingeniería.

GIMENEZ, ING GUSTAVO. *Placas de adquisición de datos y control para PC*, 2002

INGENIERÍA EN ORGANIZACIÓN INDUSTRIAL. *Rectificadores controlados*. Universidad del país Vasco.

KUO, BENJAMÍN C. *Automatic Control Systems*. Seventh Edition. Prentice Hall. 1995.

KARRIS, S.T. *Introduction to Simulink with Engineering Applications*. [s.l]: Orchard Publications, 2006.

ROJAS PURÓN L., MORERA HERNÁNDEZ M. *Supervisor gráfico de accionamiento eléctrico asistido por MATLAB. Taller Nacional de NTIC aplicadas a la Ingeniería Eléctrica*. ISPJAE. Ciudad de la Habana. Junio del 2003.



SÁNCHEZ PARRA, M.; BAHAMACA FERNÁNDEZ, L.J. *Realización de controladores lógicos difusos para el control de procesos en tiempo real: Especificación y Diseño*. México: IIE – UCI, [s.a].