IMPLEMENTACIÓN DE LA "TOOLBOX" DE ADQUISICIÓN DE DATOS DE MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.

PRESENTADO POR:

CARLOS ENRIQUE RODRÍGUEZ SARMIENTO
ID: 000073003

WILLIAM EDUARDO RODRÍGUEZ SARMIENTO
ID: 000070457

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2010

IMPLEMENTACIÓN DE LA "TOOLBOX" DE ADQUISICIÓN DE DATOS DE MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.

PRESENTADO POR:

CARLOS ENRIQUE RODRÍGUEZ SARMIENTO
ID: 000073003

WILLIAM EDUARDO RODRÍGUEZ SARMIENTO
ID: 000070457

Trabajo de grado para optar por el título de Ingeniero Electrónico

DIRECTOR DE TESIS:

EDGAR BARRIOS URUEÑA
MSc. INGENIERÍA ELÉCTRICA

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2010

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del jurado

_____

Firma del jurado

Piedecuesta, _____ de 2010

DEDICATORIA

A mi Señor, Jesús, quien nos dio la fe, la fortaleza, la salud y la esperanza para terminar este trabajo.

A nuestras esposas, quienes nos brindan su amor, su cariño, su estímulo y su apoyo constante. Su cariño, comprensión y paciente espera para que pudiera terminar el grado son evidencia de su gran amor. ¡Gracias!

A nuestros hijos quienes nos prestaron el tiempo que les pertenecía para terminar y nos motivaron siempre con sus palabras, "No se rindan" y "Sean fuertes".

A nuestros amados padres, Esther y Jorge quienes nos enseñaron desde pequeños a luchar para alcanzar las metas. Nuestro triunfo es el de ustedes, ¡Dios los bendiga!

AGRADECIMIENTOS

Al Ingeniero Edgar Barrios Urueña, director de tesis, por su visión, colaboración y asesoría en el desarrollo de este trabajo.

Al ingeniero Alex Monclou, coordinador de la facultad de ingeniería electrónica por su colaboración para que esta tesis se pudiera realizar.

CONTENIDO

INTRODUCCIÓN…… 16

PARTE I

CONCEPTOS BASICOS PARA LA EJECUCION DE LA TOOLBOX DE ADQUISICION DE DATOS……..... 17

1. PROGRAMA MATLAB……. 18

2. INSTALACIÓN Y CONFIGURACIÓN DE LA TARJETA NI DAQ 6008/6009…… 21

2.1 VISIÓN GENERAL ……..…….. 21

2.2 HARDWARE NECESARIO…..…….…… 21

2.3 SOFTWARE NECESARIO…… 21

2.4 PROGRAMACIÓN…….. 21

3. SISTEMA DE ADQUISICIÓN DE DATOS....…… 27

3.1 DESCRIPCIÓN GENERAL…… 27

3.2 HARDWARE DE ADQUISICIÓN…….. 27

3.3 SENSORES Y ACTUADORES…… 29

3.4 ACONDICIONADOR DE SEÑAL……..…… 30

3.5 COMPUTADOR……..……. 31

3.6 SOFTWARE…….. 32

3.7 MANEJO DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB……... 34

3.7.1 Flujo y almacenamiento de datos para entrada análoga..…... 34

3.7.2 Transferencia de datos desde el hardware a la memoria del sistema ……. 35

3.7.2.1 Buffer FIFO……..…….. 36

3.7.2.2 DMA……. 36

3.7.3 Flujo de datos para salida análoga……..…… 37

3.7.4 Flujo de datos para salida digital…….. 38

3.8 EL OSCILOSCOPIO PARA LA ADQUISICION DE DATOS…..…….. 40

PARTE II

UTILIZACIÓN DE LOS CUATRO SUBSISTEMAS DEL HARDWARE DE ADQUISICIÓN DE DATOS…… 42

4. ENTRADA ANÁLOGA…….…… 43

4.1 EXPERIENCIA DE INTRODUCCIÓN A LA ENTRADA ANÁLOGA…. 43

4.2 EXPERIENCIA PARA EL ACCESO A CANALES DE ENTRADA…… 49

4.3 EXPERIENCIA DEL USO DEL TRIGGERS EN LA ENTRADA ANÁLOGA ..…….. 62

4.3.1 Definición de triggers ……. 63

4.3.2 Triggers inmediato…….. 63

4.3.3 Repeticion de un triggers……. 65

4.3.4 Triggers con retardo…….. 67

4.3.5 Software triggers…….. 70

5. SALIDA ANÁLOGA……..…….. 73

5.1 EXPERIENCIA DE INTRODUCCIÓN A LA SALIDA ANÁLOGA…… 73

5.2 EXPERIENCIA PARA EL ACCESO A CANALES EN LA SALIDA ANÁLOGA…….. 77

5.3 EXPERIENCIA PARA LA UTILIZACIÓN DEL TRIGGER EN LA SALIDA ANÁLOGA……. 88

6. ENTRADA Y SALIDA DIGITAL…… 93

6.1 EXPERIENCIA DE INTRODUCCIÓN A LA ENTRADA Y SALIDA DIGITAL……. 93

6.2 EXPERIENCIA PARA LA OBTENCIÓN DE ACCESO A LÍNEAS DIGITALES…… 97

7. USO DE LOS BLOQUES DE SIMULINK EN LA ADQUISICIÓN DE DATOS…… 107

7.1 EXPERIENCIA DEL BLOQUE DE ENTRADA ANÁLOGA…… 107

7.2 EXPERIENCIA DEL BLOQUE DE SALIDA ANÁLOGA…….. 114

PARTE III

INFORMACIÓN PARA EL PERFECCIONAMIENTO DEL USO DE LA TOOLBOX DE ADQUISICIÓN DE DATOS…… 116

8. EXPERIENCIA PARA EL ACCESO A LA TARJETA DE NATIONAL INSTRUMENT…… 117

8.1 DETERMINAR EL TIPO DE HARDWARE DE LA NATIONAL INSTRUMENT QUE SE ENCUENTRA INSTALADO……..... 118

8.2 CONFIGURACIÓN DE UNA ADQUISICIÓN DE DATOS……... 120

8.3 CONFIGURACIÓN DE LAS PROPIEDADES DE LA ADQUISICION121

8.4 ESTABLECER LA VELOCIDAD DE ADQUISICIÓN Y EL NUMERO DE MUESTRAS A ADQUIRIR……125

8.4.1 Obtener los datos…..…… 125

8.4.2 Inicie el objeto de salida análoga……... 126

8.4.3 Limpiar …… 127

9. EXPERIENCIA PARA LA SINCRONIZACIÓN DE LA ENTRADA Y LA SALIDA ANÁLOGA……..…… 128

9.1 SINCRONIZACIÓN DE LA ENTRADA Y SALIDA ANÁLOGA…… 128

9.2 CONECTAR Y CONFIGURAR EL HARDWARE…….. 128

9.3 SINCRONIZACIÓN DE LA SALIDA Y ENTRADA ANÁLOGA USANDO START …… 130

9.4 SINCRONIZACIÓN DE LA ENTRADA Y SALIDA ANÁLOGA USANDO EL TRIGGER MANUAL……. 133

9.5 CONFIGURACIÓN DE LA ENTRADA Y SALIDA ANÁLOGA USANDO EL GARDWARE RTS ……... 134

10. EXPERIENCIA DE COMANDOS DE AYUDA EN LA TOOLBOX DE ADQUISICIÓN DE DATOS ……..…….. 138

11. PROGRAMANDO CON MATLAB ……..…….. 149

11.1 GENERALIDADES…….. 149

11.2 ARCHIVOS-M: COMANDOS Y FUNCIONES ……. 149

11.3 ARCHIVOS DE COMANDOS …….. 150

11.4 ARCHIVOS DE FUNCIONES…….. 151

12. MODO DE EJECUCION DEL TEMPORIZADOR DE OBJETOS ……. 154

13. EJECUCION DE UNA DEVOLUCION DE LLAMADO CON LA FUNCION DE TIEMPOS MULTIPLES ……..…….. 155

14. EXPERIENCIA DE MEDICION DE RUIDO PERSONALIZANDO LAS MEDICIONES DEL TRIGGER EN LA ENTRADA ANALOGA ……... 157

14.1 CREE EL OBJETO DE ENTRADA ANALOGICA……..…… 158

14.2 DEFINIR LA CONDICION DE INFORMACION DEL TRIGGER ... 158

14.3 CONFIGURAR EL TIEMPO DE LLAMADO …….. 159

14.4 CONFIGURAR LAS PROPIEDADES DEL TRIGGER …… 160

14.5 CONFIGURAR LA CANTIDAD DE DATOS PARA ADQUIRIR … 160

14.6 COMIENZO DE LA ADQUISICION …… 161

14.7 LIMPIAR …….. 163

14.8 LA FUNCION DE DEVOLUCION DE LLAMADO …… 163

15. EXPERIENCIA PARA EL REGISTRO DE DATOS EN EL DISCO ……. 166

PARTE IV

PRACTICAS DE LABORATORIO

16. PRACTICAS DE LABORATORIO

16.1 PRÁCTICA # 1. SISTEMA DE ADQUISICIÓN DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB, ENTRADA ANÁLOGA.

16.2 PRÁCTICA # 2 SISTEMA DE ADQUISICIÓN DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB, SALIDA ANÁLOGA, ENTRADA DIGITAL Y SALIDA DIGITAL.

16.3 PRÁCTICA # 3. SISTEMA DE ADQUISICIÓN DE DATOS ANÁLOGOS Y DIGITALES CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DESDE SIMULINK Y APLICACIÓN PARA EL ESTUDIO DEL COMPORTAMIENTO DE LOS CONTROLADORES DISCRETOS P, PI Y PID, DESDE SIMULINK.

CONCLUSIONES

BIBLIOGRAFÍA

RESUMEN GENERAL DEL TRABAJO DE GRADO

TITULO: IMPLEMENTACIÓN DE LA “TOOLBOX” DE ADQUISICIÓN DE DATOS DE MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.

AUTOR: Carlos Enrique Rodriguez Sarmiento
William Eduardo Rodriguez Sarmiento.

FACULTAD: Facultad de Ingeniería Electrónica

DIRECTOR: Msc. Edgar Barrios Urueña.

RESUMEN

Esta tesis tiene como objetivo ser un manual de ayuda para cualquier interesado en alcanzar conocimientos y adquirir destrezas en el funcionamiento y uso de la Toolbox de Adquisición de Datos de MATLAB como software de aplicación en los sistemas de adquisición de datos.

Para un mayor entendimiento del estudiante se dividió el libro en tres partes:

La primera parte tiene conceptos básicos para la instalación y configuración del software de la National Instrumets. Con esto se asegura el correcto funcionamiento de la tarjeta NI USB 6008 y 6009.

La segunda parte es el estudio mediante ejemplos de los cuatro subsistemas del hardware. Los ejemplos utilizados se hacen con la tarjeta NI USB 6008 trabajando con los tres dispositivos objeto ai (entrada análoga), ao (salida análoga), dio (Salida entrada digital) en cada uno se explican los pasos para ejecutar sesiones de adquisición de datos. En el ultimo capitulo se hace adquisición de datos

mediante la herramienta SIMULINK. Cada subsistema de adquisición está representado por un bloque se explica cómo configurarlo y usarlo.

La tercera parte del libro se entregan herramientas para el manejo de necesidades específicas, como lo es la sincronización de la entrada y la salida análoga con diferentes métodos, la modificación y establecimiento de la velocidad de adquisición de datos, entre otras.

La cuarta parte es el culmen del libro. Corresponde a tres prácticas de laboratorio con un estilo didáctico y sensibilidad pedagógica. Las primeras dos llevarán al estudiante por el conocimiento y la práctica en los sistemas de adquisición de datos con la Toolbox de adquisición de datos de Matlab y la tercera es una aplicación para control digital

PALABRAS CLAVES: TOOLBOX

GENERAL SUMMARY OF WORK OF DEGREE

TITLE: I MPLEMENTATION OF THE "TOOLBOX" OF DATA OF MATLAB AND LABORATORY PRACTICES WITH THE USE OF THE CARDS NI-USB 6008/6009.

AUTHOR: Carlos Enrique Rodriguez Sarmiento
William Eduardo Rodriguez Sarmiento.

FACULTY: Faculty of Electronic Engineering

DIRECTOR: Msc. Edgar Barrios Urueña.

ABSTRACT

This thesis has as aim be a manual of help for any interested party in reaching knowledge and acquiring skills in the functioning and use of the Toolbox of Acquisition of Information of MATLAB as software of application in the systems of acquisition of information.

For a major understanding of the student the book was divided in three parts:

The first part has basic concepts for the installation and configuration of the software of the National Instrumets. With this insures himself neither the correct functioning of the card NOR USB 6008 and 6009.

The second part is the study by means of examples of four subsystems of the hardware. The used examples are done neither by the card NOR USB 6008 working with three devices I object ai (analogous entry), ao (analogous exit), it gave (Gone out digital entry) in each one the steps are explained to execute

meetings acquisitions of information. In the last chapter acquisition of information is done by means of the tool SIMULINK. Every subsystem of acquisition is represented by a block explains how to form it it and to use it.

The third part of the book tools are delivered for the managing of specific needs, since it it is the synchronization of the entry and the analogous exit with different methods, the modification and establishment of the speed of acquisition of information, between others.

The fourth part is the culmen of the book. It corresponds to three laborator practices with a didactic style and pedagogic sensibility. The first ones two will take the student for the knowledge and the practice in the systems of acquisition of information with the Toolbox of acquisition of information of Matlab and the third one is an application for digital control.

KEYWORDS: TOOLBOX

INTRODUCCION

MATLAB es utilizado hoy en día en un amplio rango de aplicaciones, incluyendo procesado de señal, sistemas de comunicación, diseño de control, procesado de imagen, sistemas de verificación y test, etc. Muchas de estas aplicaciones requieren la toma de datos reales obtenidos desde sensores y solucionar problemas de cálculo técnico, realizar análisis de datos y visualización. La adquisición de datos directamente en MATLAB puede ayudar a solucionar estos problemas con una reducción significativa del tiempo empleado. En éste trabajo se explica cómo analizar y visualizar datos adquiridos directamente en MATLAB, como conectarse a equipos externos desde MATLAB y como desarrollar e implementar aplicaciones basadas en interfaces graficas de usuario incorporando datos que se adquieren en tiempo real.

Contando con un sistema de adquisición de datos, una tarjeta DAQ (data adquisition) de National Instruments NI USB 6008/6009 y el software MATLAB versión 7.7 de Math Works con la Toolbox de adquisición de datos, se realizaran experiencias de lectura de datos analógicos y digitales, así como la generación de señales analógicas y digitales. Se explica paso a paso como realizar esas operaciones. MATLAB se ha constituido en un programa de alto nivel bastante utilizado en ingeniería particularmente en el área de control automático que permite utilizar las diferentes herramientas que ofrece para el análisis y diseño de sistemas de control

PARTE I

CONCEPTOS BASICOS PARA LA EJECUCION DE LA TOOLBOX DE ADQUISICION DE DATOS

En esta parte encontramos los conceptos básicos para el aseguramiento del uso de la toolbox de adquisición de datos con la instalación y configuración del software de la National Instrumet para la tarjeta NI DAQ 6008. El sistema de adquisición de datos se desglosa y se explica con el objeto de dar una mejor comprensión de los elementos que interactúan en una sesión de adquisición de datos. Finalizamos esta parte del informe con una explicación del uso y las características del osciloscopio que provee matlab para la toolbox de adquisición de datos.

1 EL PROGRAMA MATLAB

MATLAB significa "Matriz Laboratorio". Es un programa para cálculo técnico y científico; sus cálculos numéricos se hacen en forma matricial y dada su gran capacidad para realizar diferentes gráficos en dos y tres dimensiones y un lenguaje de programación propio se ha posicionado como una de las herramientas de mayor uso por parte de la comunidad técnica y científica.

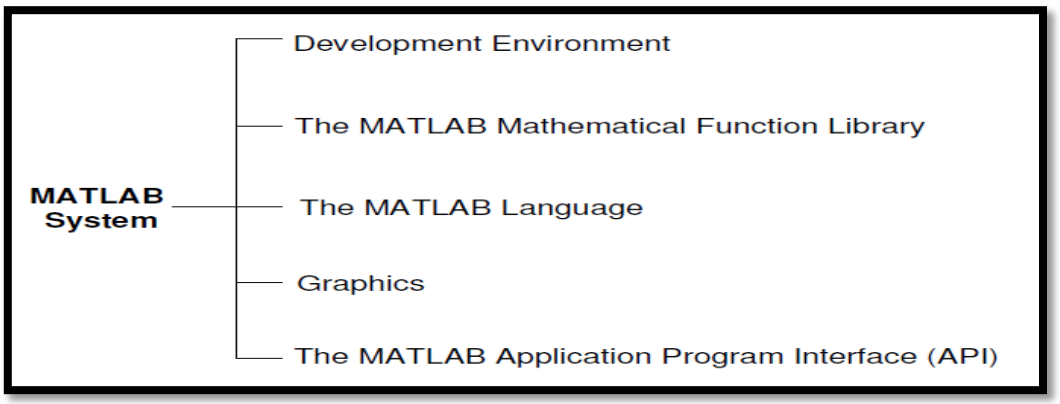
MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de programación tradicional, es por esto que MATLAB es una excelente herramienta de alto nivel para desarrollar aplicaciones técnicas, es fácil de utilizar y aumenta la productividad de los programadores respecto a otros entornos de desarrollo.

MATLAB en la actualidad dispone de una amplia cantidad de programas de apoyo especializados, denominados Toolboxes, estos extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes

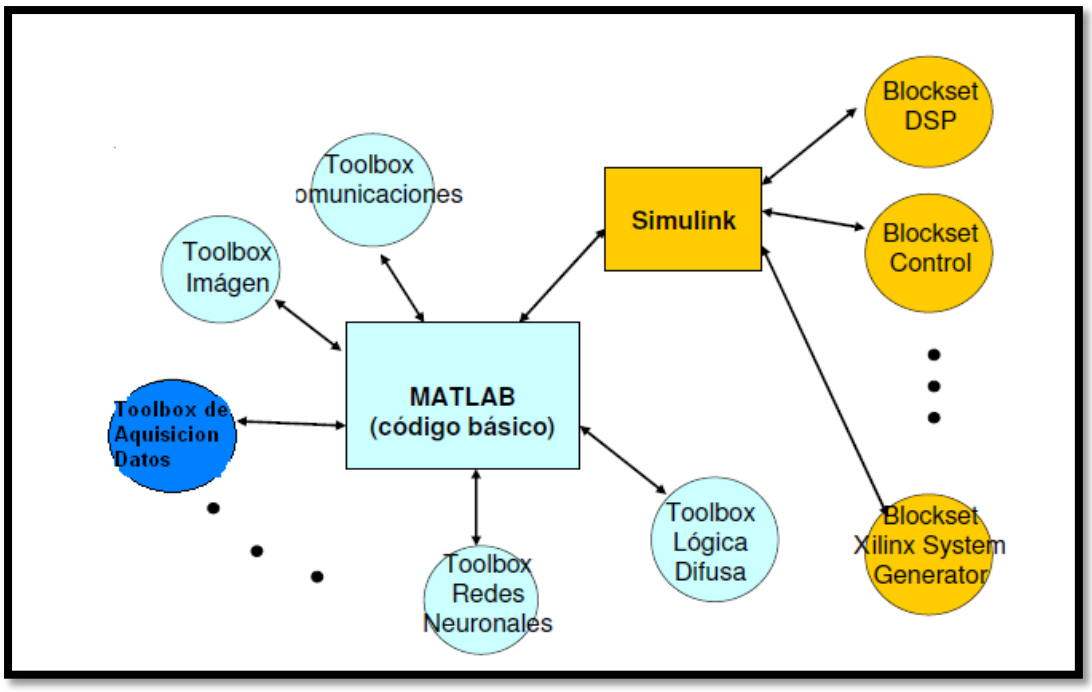
cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, por ejemplo están los 'toolbox' para proceso de imágenes, procesamiento digital de señales, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, entre otros.

Combinando MATLAB con Signal Processing Toolbox, Wavelet Toolbox y un conjunto de herramientas complementarias, se puede crear un ambiente de análisis personalizado de señales y desarrollo de algoritmos DSP. Para simulación y desarrollo de prototipos se puede agregar el Simulink y el DSP Blockset para modelar y simular sistemas DSP, y luego usar Real-Time Workshop para generar código C para su hardware designado.

MATLAB integra todos los requisitos claves de un sistema de computación técnico: cálculo numérico, gráficos, herramientas para aplicaciones específicas y capacidad de ejecución en múltiples plataformas como Windows 95/98/XP/NT, Macintosh, Unix y Linux.



Grafica 01. Cuadro sinóptico de la organización de Matlab (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB Pag 4)



Grafica 02. Estructura organizativa de Matlab (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB Pag 5)

2 INSTALACION Y CONFIGURACION DE LA TARJETA NI DAQ 6008/6009

2.1 VISION GENERAL

Para la realización de algunas experiencias que se encuentran en la parte II del libro, es necesario el uso de la tarjeta NI USB 6008/6009, por lo que se explica cómo se debe instalar y configurar la tarjeta para que funcione correctamente.

2.2 HARDWARE NECESARIO

Tarjeta DAQ USB 6009 ó 6008

Cable de conexión USB

Computadora con al menos un puerto USB

Conecte las terminales de tornillo a las terminales de la DAQ y pegue las etiquetas necesarias en la tarjeta. Encienda su computadora y permita que cargue su sistema operativo.

2.3 SOFTWARE NECESARIO

Toolbox de Adquisición de datos de MATLAB.

NI – DAQm

2.4 PROGRAMACION

Asegúrese de que tenga instalado la toolbox de adquisición de datos en la computadora. Si no es así, instálelo insertando los discos de instalación de Matlab

versión 7.7. Inserte los discos de NI DAQ mx que acompañan a la tarjeta DAQ USB 6008/9 en la computadora. Siga los pasos de instalación para los controladores de la tarjeta y del programa. Cuando haya finalizado, conecte la tarjeta DAQ al cable USB y este a algún puerto USB de la computadora. Aparecerá un globo de diálogo en la barra de tareas como el siguiente:

Grafica 03. Aviso de detección del hardware

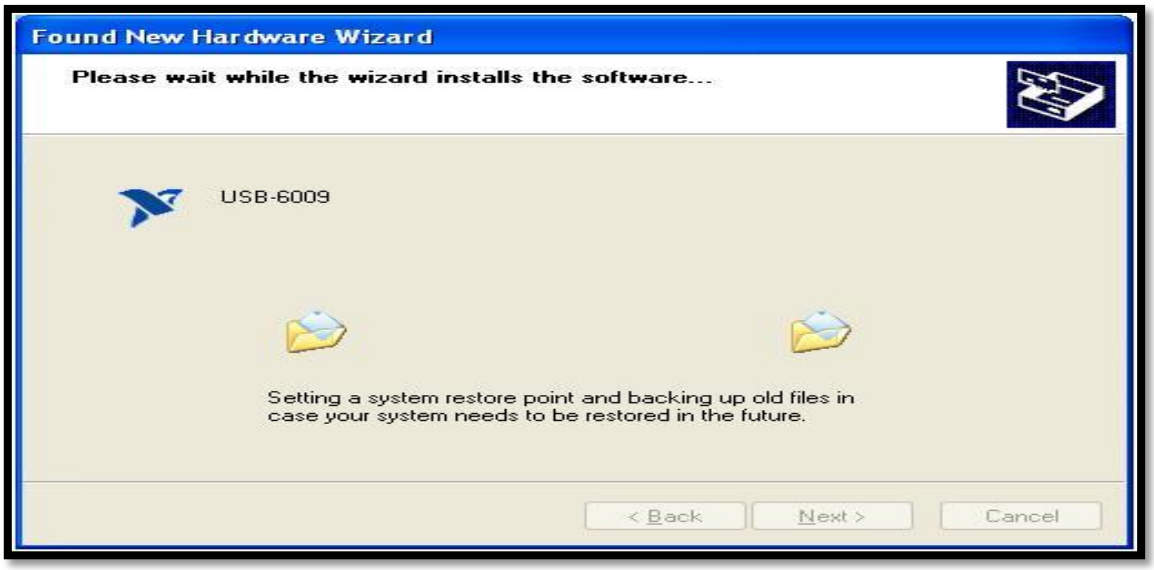
Posteriormente aparecerá el asistente de instalación de hardware nuevo. Seleccione la opción que evita que busque en Windows Update el controlador de la tarjeta y presione el botón de “siguiente”:

Grafica 04. Primera ventana que se presenta para la instalación del driver

Luego aparecerá la ventana de ubicación de driver. En este caso seleccione instalar el programa automáticamente y de clic en “Siguiente”:

Grafica 05. Segunda ventana que se presenta para la instalación del driver

Aparecerá una venta indicando que la instalación se está llevando a cabo y luego confirmará que la instalación ha sido exitosa:

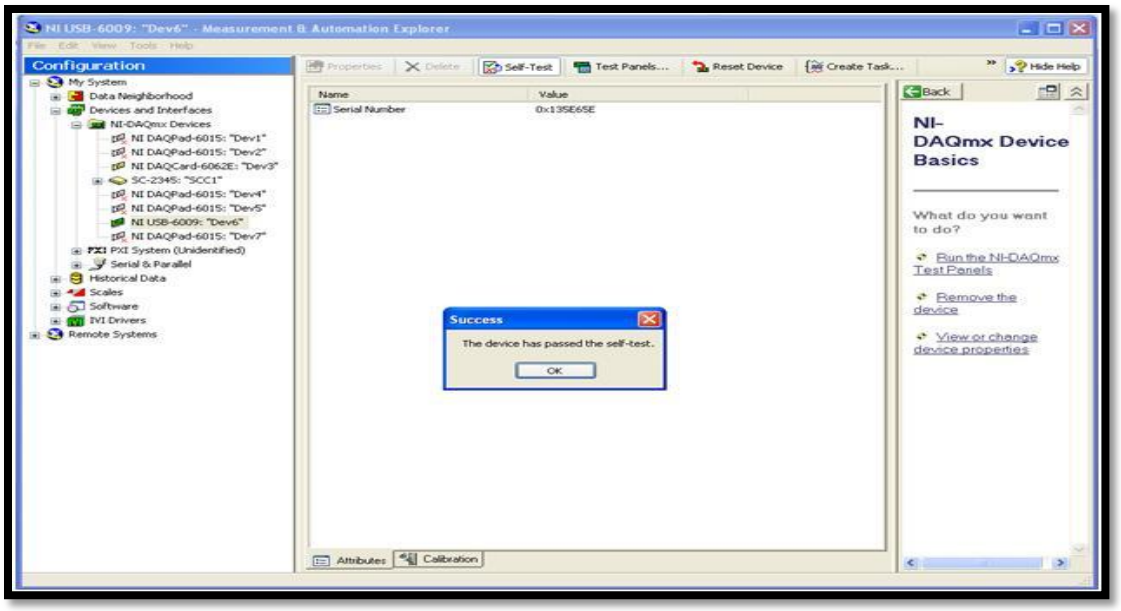


Grafica 06. Copiando los drivers desde el CD instalador al PC

Ejecute el programa Measurements and Automation Explorer (MAX) que se encuentra en Menú Inicio >> Todos los programas >> National Instruments. El ícono del programa es el siguiente:

Grafica 07. Icono para la ejecución del programa Measurements

Una vez abierto, revise en la sección de configuración y extienda la ramificación Devices and Interfaces >> NI DAQ-mx Devices. Si la instalación fue exitosa, aparecerá un ícono en verde de una tarjeta NI USB-6008/9 y MAX le habrá asignado un nombre a esta del tipo “Dev#”. Tome en cuenta este nombre porque será el identificador de la tarjeta en cualquier programa de National Instruments para acceder a ella. Presione el botón de “Self-Test” para revisar que la comunicación es efectiva si aparece la ventana “Success”



Grafica 08. Ventana inicial después de ejecutar el programa measurements de National

Como una breve introducción, la pestaña de Test Panels da acceso a las terminales del equipo de National Instruments conectadas a la computadora. Con esto se puede comprobar el buen funcionamiento de cada terminal.

Grafica 09. Icono para comprobar buen funcionamiento de cada terminal.

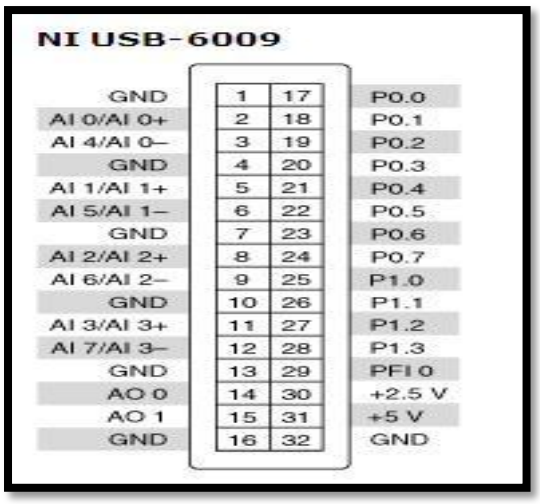
Reset Device es útil cuando la tarjeta deja de funcionar correctamente por varias razones.

Grafica 10. Icono Reset Device.

Por último, al presionar clic derecho sobre el ícono de la tarjeta, se abrirá un menú en donde se pueden encontrar los “Device Pinouts” que es la configuración física de los pines en la tarjeta DAQ y esto aplica para cualquier modelo de DAQ de National Instruments.

Grafica 11. Ventana en la que aparece Device pinouts.

Por ejemplo, para la DAQ USB 6008/9 los “Device Pinouts” se encuentran acomodados de tal manera que las entradas y salidas analógicas (AI/AO) se encuentran separadas de las entradas y salidas digitales (DI/DO) así:



Grafica 12. Configuración física de pines en la USB6008/6009.

Una vez que el aparato haya pasado el auto diagnóstico “Self Test” se procede a utilizar la adquisición de datos con la toolbox.

3 SISTEMA DE ADQUISICIÓN DE DATOS

3.1 DESCRIPCION GENERAL

La adquisición de datos o adquisición de señales consiste en la toma de muestras del mundo real para generar datos que puedan ser manipulados por un computador, 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. Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. El elemento que hace dicha transformación es el módulo de digitalización o tarjeta de Adquisición de Datos (DAQ).

El objetivo de cualquier sistema de adquisición de datos es proporcionar las herramientas y recursos necesarios para tomar señales físicas y convertirlas en datos que posteriormente se puedan procesar y mostrar.

Un sistema de adquisición de datos se podría tomar como un grupo de hardware y software que permiten interactuar con el mundo real. Un sistema de adquisición de datos típico consiste de estos componentes:

3.2 HARDWARE DE ADQUISICION

Es el corazón de cualquier sistema de adquisición de datos. La función principal es hacer la conversión de señales análogas a señales digitales y señales digitales a análogas. Conversión A/D y D/A

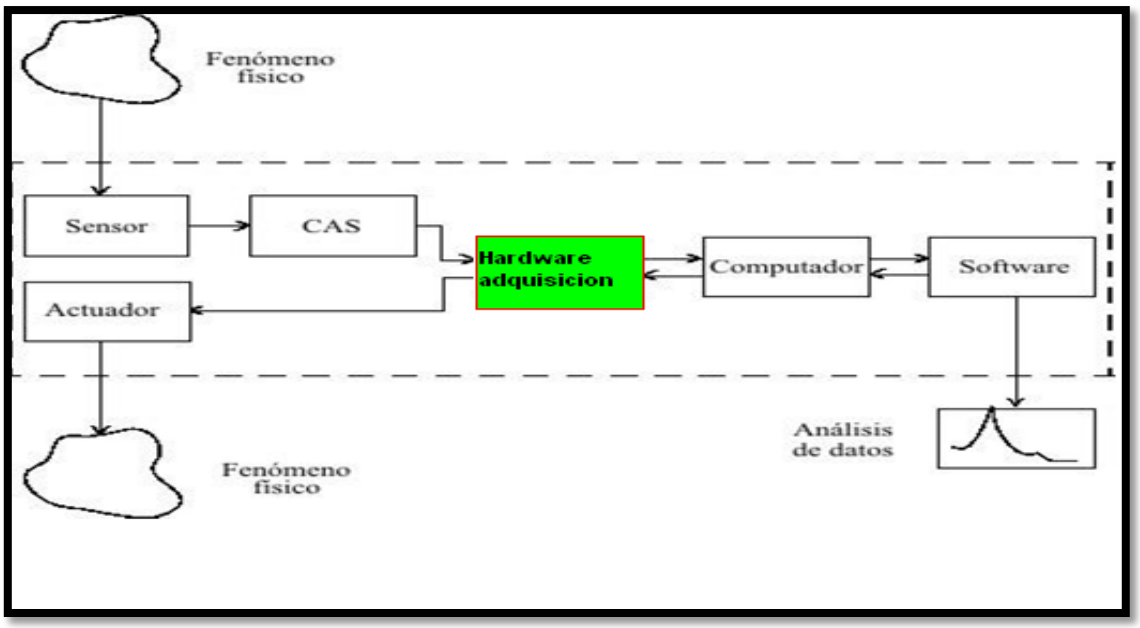


Grafico 13. Hardware de adquisición de datos componente de la toolbox de MATLAB. (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB)

El hardware de adquisición de datos puede instalarse de manera interna directamente en una ranura de expansión del computador o de manera externa que se conecta al computador a través de un cable al puerto USB.

Se caracteriza por los subsistemas que éste posee. Un subsistema es un componente del hardware que realiza una tarea específica. Los más comunes son:

- Entradas análogas
- Salidas análogas
- Entradas/salidas digitales
- Contador/Temporizador

El hardware utilizado para las prácticas de laboratorio es la tarjeta USB-6008 de la National Instrument, que tiene un bajo costo y múltiples funciones de adquisición de datos. Tiene 8 entradas analógicas (12 bits, 10 kS/s), 2 salidas analógicas (12 bits a 150 S/s), 12 E/S digitales; contador de 32 bits.

Grafica 14. Tarjeta NI USB-6008

3.3 SENSORES Y ACTUADORES

Dispositivo capaz de convertir un tipo de energía de entrada en otra, obtiene información de entornos físicos y la convierte en señales eléctricas o viceversa.

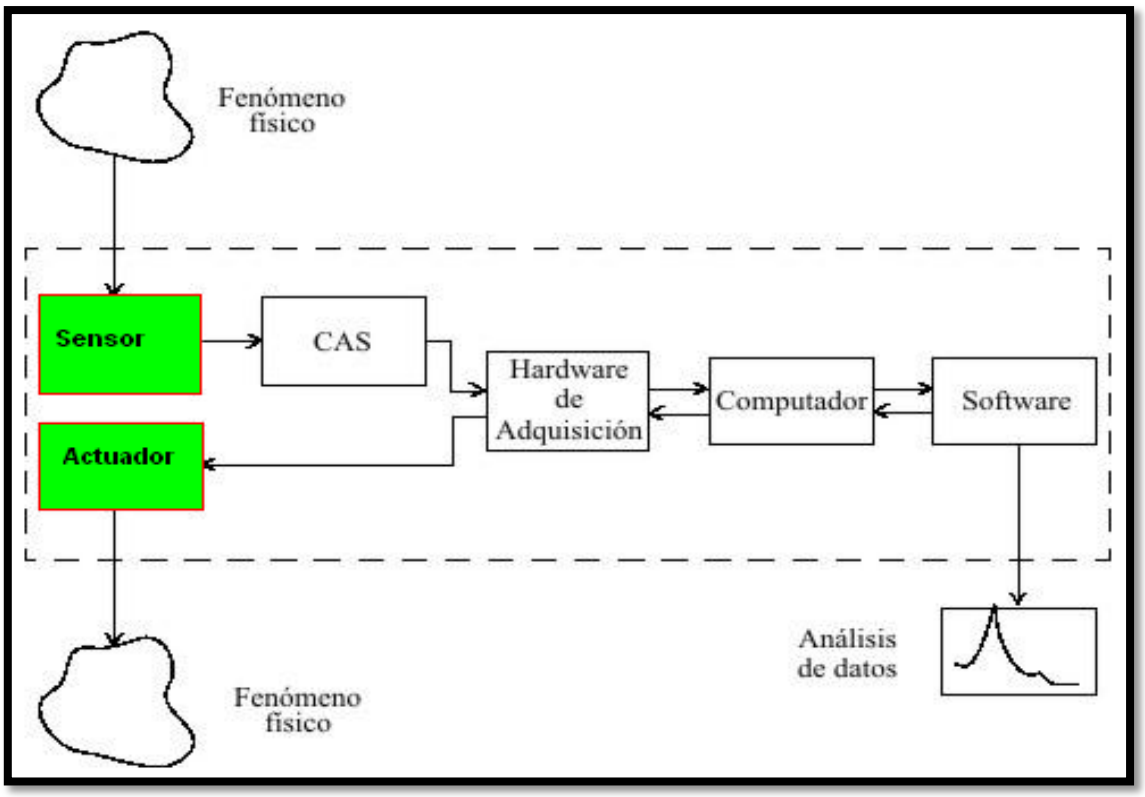


Grafico 15. Transductores, componente del sistema de adquisición de datos de MATLAB. (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB)

Hay dos tipos de sensores según la salida que produce: sensor digital y sensor análogo.

3.4 ACONDICIONADOR DE SEÑAL

Las señales de los sensores a menudo son incompatibles con el hardware de adquisición de datos y para superar esto las señales deben ser acondicionadas. Por ejemplo las señales podrían ser amplificadas o volverlas en señales sin componentes de frecuencias indeseadas. Las señales de salida también pueden ser acondicionadas.

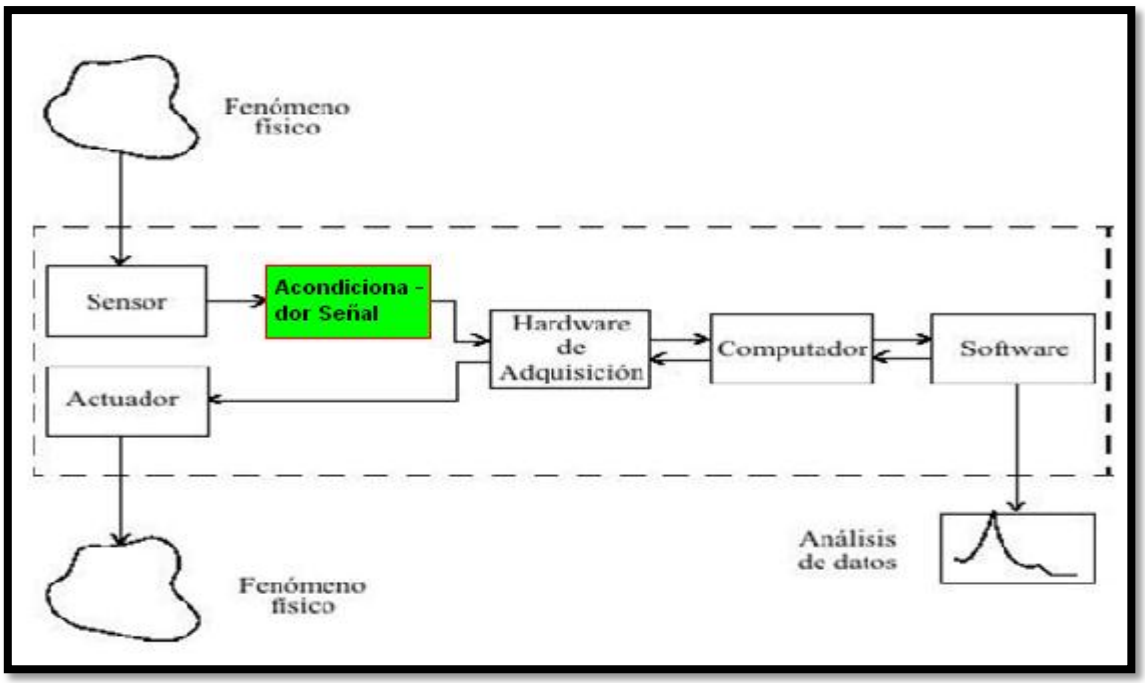


Grafico 16. Acondicionador de Señal, componente del sistema de adquisición de datos de MATLAB. (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB)

Los transductores generalmente son incompatibles con el "Hardware de adquisición", para resolver esta incompatibilidad, las señales adquiridas se deben acondicionar de alguna forma. El tipo de acondicionamiento depende del sensor que se utilice.

Las formas más comunes de acondicionar una señal son:

3.5 COMPUTADOR

Ofrece un procesador, un sistema de reloj, un bus para la transferencia de datos, memoria y espacio en el disco para almacenar datos.

El procesador controla la rapidez de datos aceptados por el convertidor. El sistema de reloj proporciona información acerca del tiempo de los datos adquiridos.

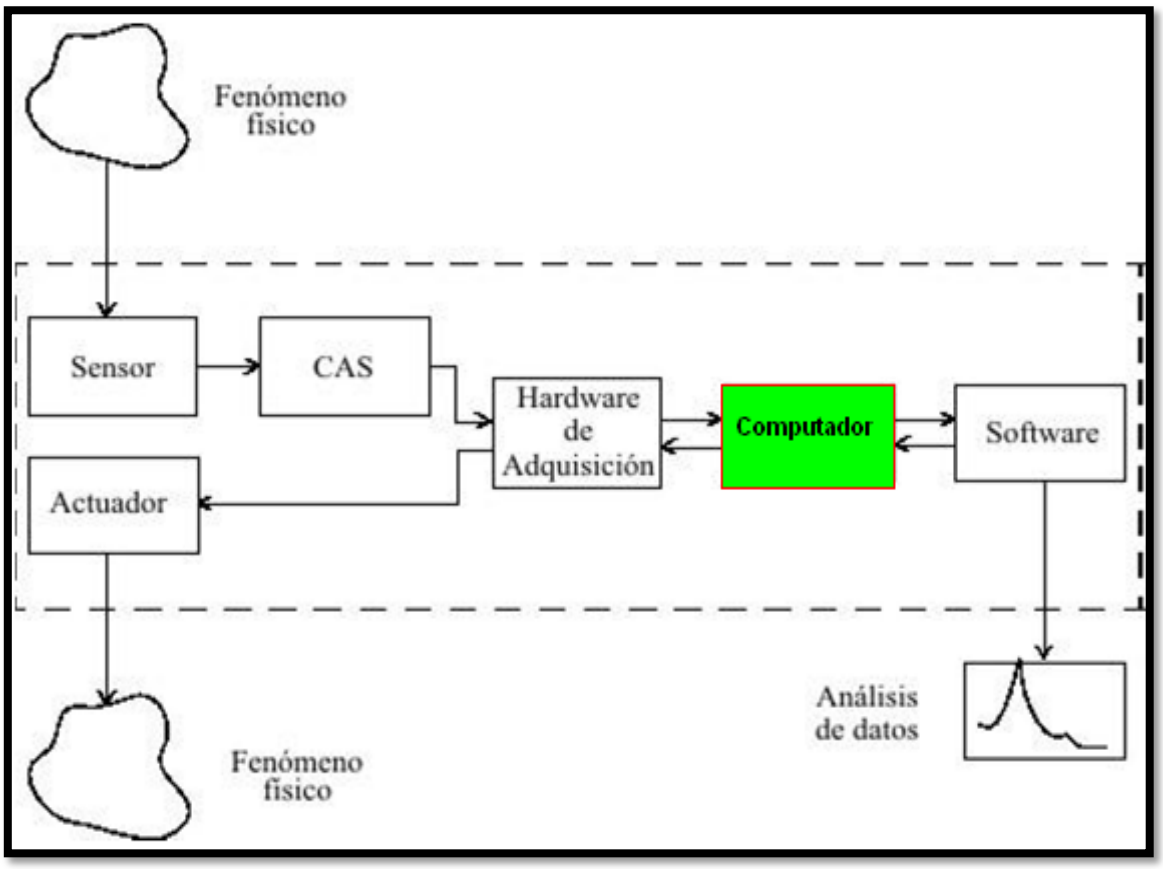


Grafico 17. Computador, componente del sistema de adquisición de datos de MATLAB. (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB)

Los datos son transferidos desde el hardware al sistema de memoria a través de la memoria dinámica de acceso (DMA) o interrupciones. La DMA es controlada por hardware y por lo tanto es extremadamente rápida. La tasa de adquisición de datos también está determinada por la arquitectura del sistema del bus.

3.6 SOFTWARE

Permite la interacción entre la computadora y el hardware permitiendo que se pueda configurar la tasa de muestreo de la tarjeta, adquirir una predeterminada cantidad de datos y visualizar uso del toolbox de adquisición de datos

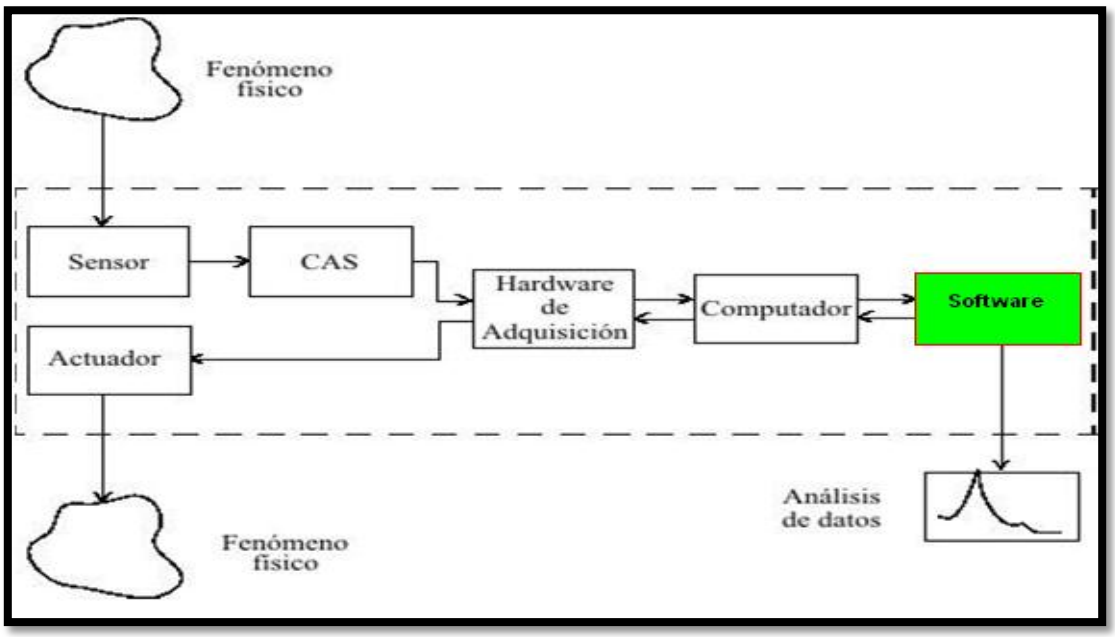


Grafico 18. Software, componente del sistema de adquisición de datos de MATLAB. (Juan Carlos Moctezuma Eugenio, ADQUISICIÓN DE AUDIO CON MATLAB)

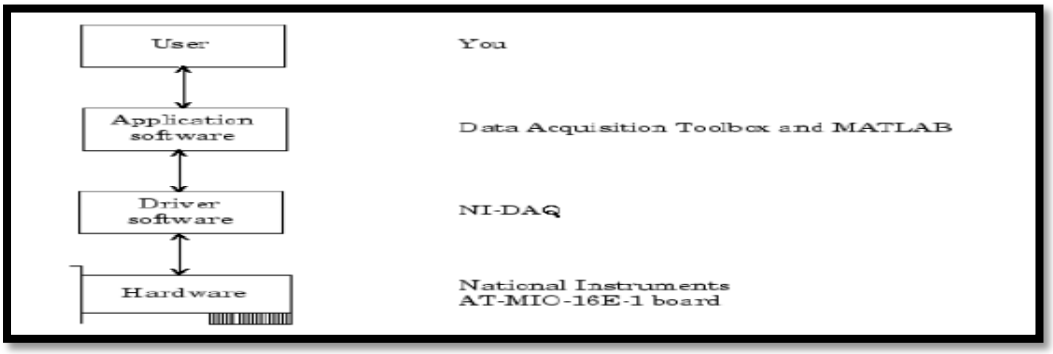


Grafico 18. Relación usuario, software y hardware. (Toolbox de adquisición de datos de matlab tarjeta ni-daq 6008/6009)

Como se muestra en la siguiente figura, estos componentes permiten pasar información entre Matlab y el hardware de adquisición de datos.

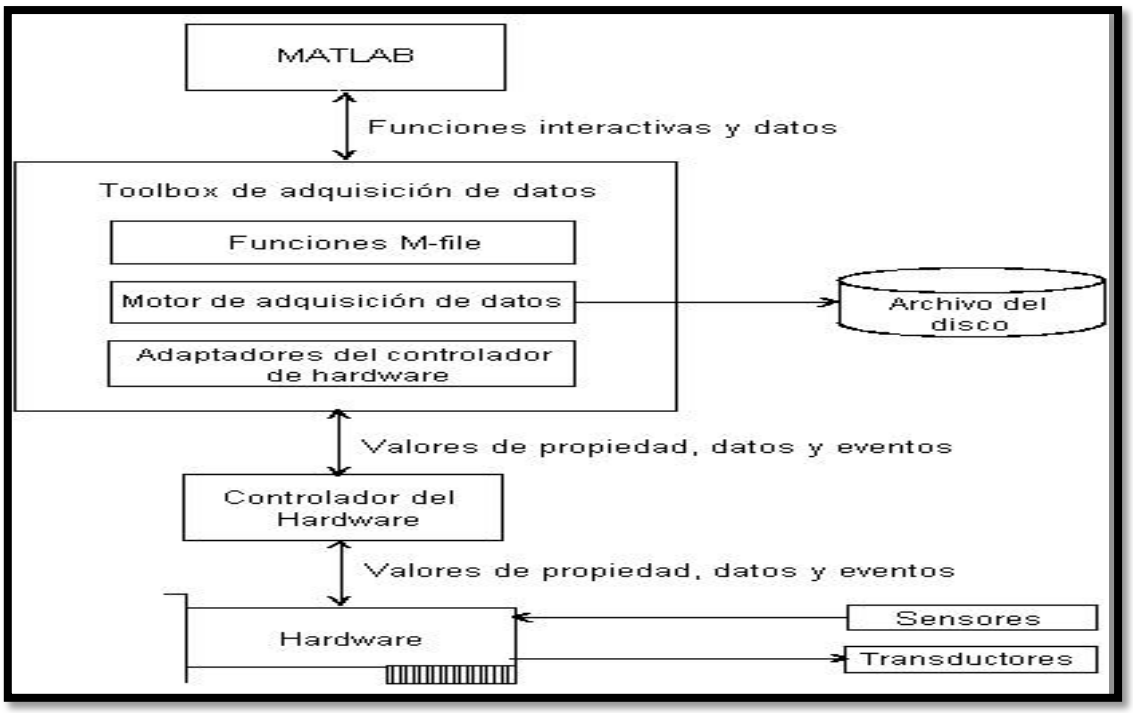


Grafico 19. Componentes básicos de la Toolbox de Adquisición de Datos.

(Toolbox de adquisición de datos de matlab tarjeta ni-daq 6008/6009)

3.7 MANEJO DE DATOS CON LA TOOLBOX DE ADQUISICION DE DATOS DE MATLAB

La manipulación de datos por parte del sistema de adquisición de datos se muestra a continuación.

3.7.1 Flujo y almacenamiento de datos para entrada análoga

En el flujo de datos, los datos son almacenados temporalmente en la memoria porque estos se van sobrescribiendo. La tasa con la cual los datos son sobrescritos depende de factores incluyendo la memoria disponible, la tasa con la que los datos son adquiridos y el número de canales del hardware.

Los datos almacenados no están disponibles automáticamente en el área de trabajo de MATLAB. Por ello, se tiene que extraer explícitamente del motor usando la función getdata.

El flujo de datos adquiridos consiste de dos pasos independientes:

1. Los datos adquiridos desde el hardware son almacenados en el motor.

2. Los datos son extraídos desde el motor y almacenados en el espacio de trabajo de MATLAB o sacados como archivo.

Estos pasos son ilustrados a continuación:

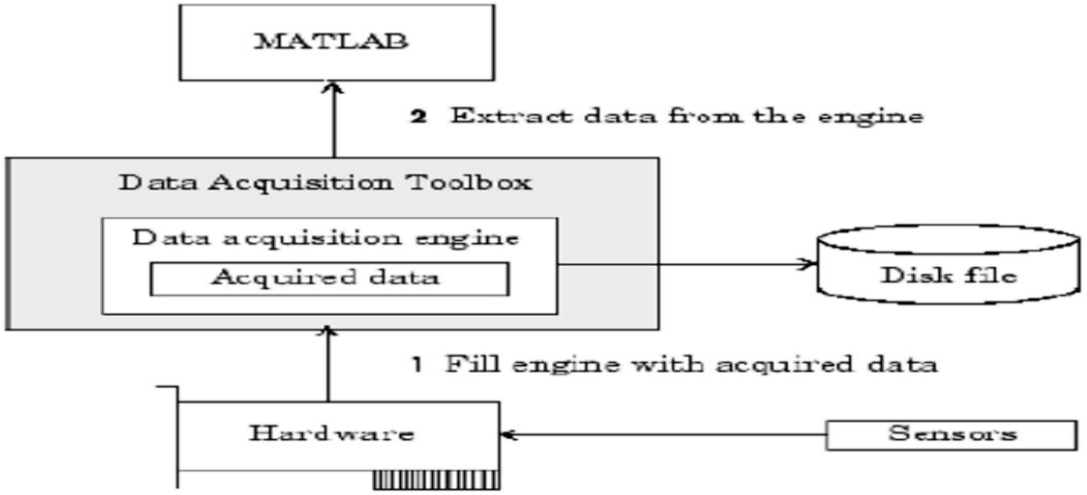


Grafico 20. Flip de datos adquiridos. (Toolbox de adquisición de datos de matlab tarjeta ni-daq 6008/6009)

3.7.2 Transferencia de datos desde el hardware a la memoria del sistema

La transferencia de datos adquiridos desde el hardware a la memoria del sistema sigue estos pasos:

1. Datos adquiridos se almacenan en el buffer de primera entrada - primera salida (FIFO) del hardware.

2. Los datos se transfieren desde el búfer FIFO a la memoria del sistema utilizando interrupciones o DMA.

Estos pasos ocurren de forma automática. Normalmente, todo lo que requiere es alguna configuración inicial del hardware cuando se instala.

3.7.2.1 Buffer FIFO

El buffer FIFO es usado para almacenar temporalmente los datos adquiridos. Los datos se almacenan temporalmente hasta que puedan ser transferidos a la memoria del sistema. El proceso de transferencia de datos de entrada y salida de un buffer FIFO de entrada analógica es el siguiente:

1. El buffer FIFO almacena las muestras adquiridas recientemente con una tasa de muestreo constante.

2. Antes de que el buffer FIFO se llena, el software comienza a remover las muestras.
   Por ejemplo, una interrupción se genera cuando la FIFO está medio llena, y las señales del software extraen las muestras tan pronto como sea posible.
3. Puesto que las interrupciones del servicio o la programación el controlador DMA puede tardar unos pocos milisegundos, datos adicionales son almacenados en el FIFO para futura recuperación.

4. Las muestras se transfieren a la memoria del sistema a través del bus del sistema.

Después de que las muestras se transfieren, el software es libre para realizar otras tareas hasta que ocurra la siguiente interrupción. Por ejemplo, los datos pueden ser procesados o guardados en un archivo del disco. Ya que la tasa promedio de almacenamiento y extracción de datos son iguales, los datos adquiridos no se perderán y su aplicación deberá funcionar sin problemas.

3.7.2.1 DMA

Memoria de acceso directo (DMA) es un sistema por el cual las muestras son automáticamente almacenadas en la memoria del sistema mientras que el

procesador hace otra cosa. El proceso de transferencia de datos a través de DMA es el siguiente:

1. Cuando los datos están listos para la transferencia, la tarjeta dirige el sistema de controlador DMA para ponerlos en la memoria del sistema tan pronto como sea posible.

2. Tan pronto como la CPU es capaz (que suele ser muy rápido), deja de interactuar con el hardware de adquisición de datos y el controlador DMA mueve los datos directamente dentro de la memoria.

3. El controlador DMA se prepara para la próxima muestra indicando la siguiente ubicación de memoria abierta.

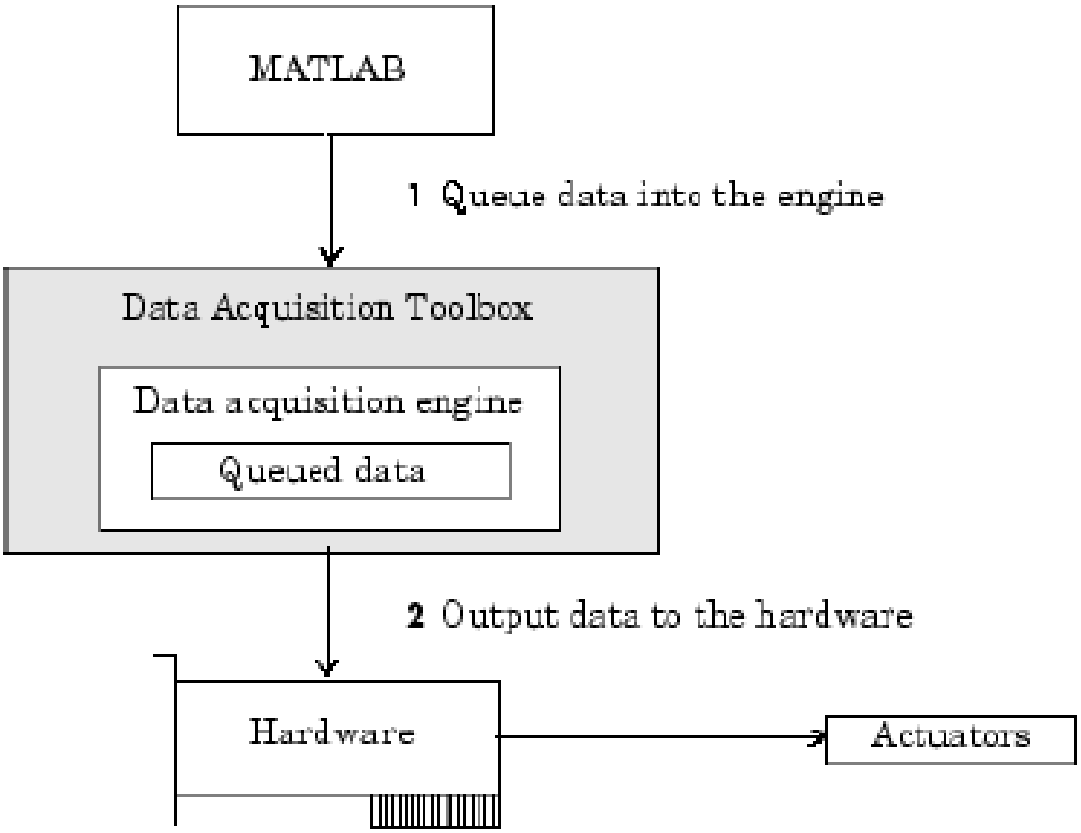
4. Los anteriores pasos son repetidos indefinidamente, con información que llega a cada una de las localidades de memorias abiertas en un buffer de circulación continua, ninguna interacción entre la CPU y la tarjeta es necesaria.

3.7.3 FLUJO DE DATOS PARA SALIDA ANALOGA

El flujo de salida de datos se refiere al flujo de datos desde el motor de adquisición de datos al hardware. Sin embargo, antes que los datos son sacados, se deben poner en cola en el motor con la función *putdata.* La cantidad de datos que se pueden poner en cola depende de factores incluyendo la memoria disponible, el número de canales del hardware y el tamaño de cada muestra de dato.

El flujo de salida de datos consiste de dos pasos independientes:

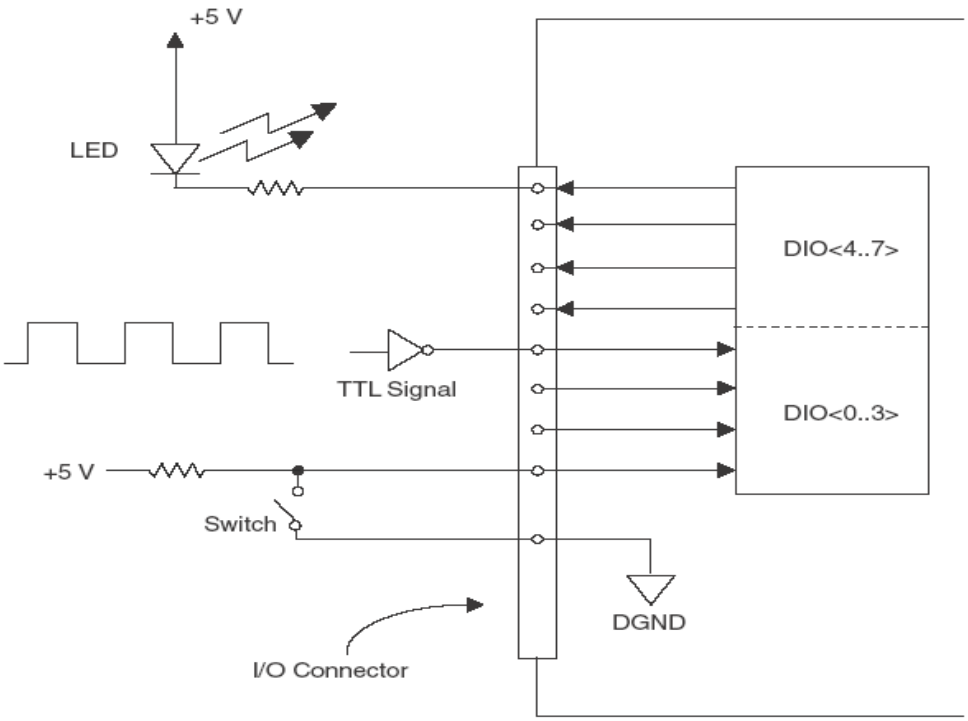
1. Los datos son puestos en cola en el motor desde el espacio de trabajo de MATLAB.
2. Los datos en cola en el motor son sacados al hardware.



Grafica 21. Flujo de datos en salida análoga. (Toolbox de adquisición de datos de matlab tarjeta ni-daq 6008/6009)

3.7.4 FLUJO DE DATOS PARA SALIDA Y ENTRADA DIGITAL

La tarjeta de adquisición de datos tiene ocho líneas de I/O digital (DIO<0..7>) para su utilización de propósito-general. DGND es la línea de referencia a tierra. Se puede configurar por programación cada una de las líneas como entrada o como salida. Al iniciar el sistema, los puertos digitales están en alta impedancia. En el grafico 22 se aprecia un ejemplo de utilización.



Grafica 22. Flujo de datos en salida análoga. (Toolbox de adquisición de datos de matlab tarjeta ni-daq 6008/6009)

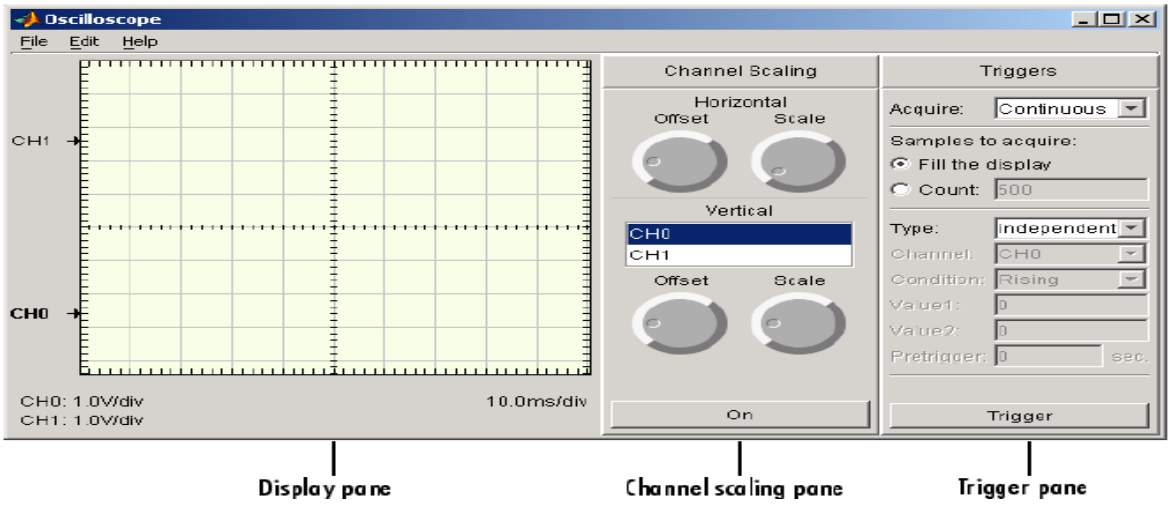
Dos contadores de propósito general integrados en la tarjeta están conectados a DIO6 y a DIO7. Por lo que ambas conexiones se pueden utilizar para controlar a dichos contadores.

Se puede programar cada puerto como entrada o como salida. Este circuito tiene tres modos propios de operación que le confiere gran versatilidad.

Es necesario no exceder el nivel de tensión de + 5V en las entradas digitales pues se corre el riesgo de dañar tanto a la tarjeta como al PC. Es importante anotar que los datos no se quedan almacenados en el motor como en la adquisición análoga, los datos se deben asignar a una variable que pase la información a alguna matriz n por m.

3.8 EL OSCILOSCOPIO

La Toolbox de adquisición de datos cuenta con un osciloscopio exclusivo para entrada análoga denominado Softscope, usa la interface (GUI) en una gráfica interactiva que muestra en pantalla los datos en tiempo de ejecución “tiempo real”.



- **Grafica 23. Pantallazo del osciloscopio.** (Help Data adquisition toolbox)

Como observamos en la figura 23, el Osciloscopio presenta un display, un recuadro para cada canal adicionado del hardware, perillas para offset y escala horizontal y vertical y un panel para el trigger.

- Para abrir el Osciloscopio a una entrada ai, se llama la función *Softscope(ai)*.
- La configuración del hardware se hace desde el banner Edit con la secuencia Edit>Hardware y para dar una nueva velocidad de muestreo Edit>Sample Rate.
- Para crear displays adicionales se sigue la secuencia Edit>Scope>Scope asignando el nombre del nuevo display en label y para asignar la información de un canal en particular al display se sigue Edit>Channel>Channel Display, luego de nombrar el nuevo display se da clic sobre Add, Apply y Ok. Con el clic derecho sobre el display se configura auto escala.

- Para ingresar a un display una señal de referencia que se ha debido definir en el workspace de Matlab se sigue Edit>Channel>Channel y para realizar operaciones matemáticas entre canales se sigue Edit>Channel>Channel
- Existen 3 tipos de trigger:

  One shot.- adquiere el número de muestras especificado, una vez

  Continuous.- continuamente adquiere el número de muestras especificadas

  Sequence.- continuamente adquiere el número de muestras especificadas y usa el trigger dependiente cada vez.
- Para realizar una medición predeterminada se da clic derecho sobre el Osciloscopio y para predefinir un nuevo tipo de medición se sigue Edit>Measurement>Measurement Type. Se mostrara un nuevo panel de mediciones. Con el clic izquierdo sobre un punto de alguna curva se observará su valor específico.

  Para exportar los datos al workspace se sigue File>Export>Channels

PARTE II
EXPERIENCIAS PARA EL APRENDIZAJE DE LA TOOLBOX DE ADQUISICION DE DATOS

La segunda parte del Informe se basa en experiencias de adquisición de datos con la toolbox de Matlab; se explica cómo ingresar o sacar datos ya sea de forma análoga o digital,; se presentan experiencias donde se les modifica alguna propiedad y se explica el concepto de estas propiedades como lo es el caso del uso del trigger. Se hace adquisición de datos con bloques de SIMULINK; se explica detalladamente alguno de los bloques y como utilizarlos para ejecutar una adquisición de datos.

4 ENTRADA ANALOGA

4.1 EXPERIENCIA DE INTRODUCCION A LA ENTRADA ANALOGA

En esta experiencia se pretende adquirir datos de la tarjeta NI DAQ 6008. Para empezar, primero debe verificar que el hardware este debidamente configurado (Verificar los pasos de instalación de la tarjeta NI DAQ 6008 que se encuentran en la parte 1 del informe capitulo 2).
Para adquirir los datos de su tarjeta NI DAQ 6008, necesita un origen de datos. La fuente de datos es la unidad para sistemas de control digital y análogo tarjeta utilizada en el laboratorio de control, la señal de entrada es una señal triangular.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la Toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera,

```
if (~isempty(daqfind))
   stop(daqfind)
end
```

El primer paso es crear un objeto de entrada análoga y especificar los canales para la adquisición de datos.

Los objetos son los componentes de la Toolbox con los que se acceden al dispositivo. Estos proveen una entrada a la funcionalidad del hardware y le permite el control del comportamiento de la funcionalidad de la aplicación de adquisición de datos. Cada objeto está asociado con un subsistema de hardware específico.
Muchos de los dispositivos de adquisición de datos contienen uno o más subsistemas que convierten (digitalizan) las señales del sensor en datos que el computador puede leer. Estos dispositivos se denominan subsistemas de entradas análogas (subsistemas AI, convertidores A/D, o ADCs). Después que la señal del sensor es digitalizada, se puede analizar, almacenar en la memoria del sistema, o almacenar en un disco de archivos.
Un objeto de entrada análoga se crea con la función analoginput. Esta función acepta el nombre del adaptador y el ID del dispositivo como argumentos de entrada. El ID del dispositivo se refiere al número asociado con la tarjeta cuando ésta es Instalada. (Cuando se una NI-DAQmx, esta es usualmente una línea como 'Dev1').

```
AI = analoginput('nidaq','Dev3');
addchannel(AI, 1);
```

Los Canales o las líneas son los elementos básicos del hardware con los cuales se adquiere o entrega datos. Después de crear un objeto de dispositivo, debe agregársele canales o líneas. Los canales son agregados a los objetos de entrada y salida análoga, mientras que las líneas son agregadas a los objetos de entrada/salida digitales.

Se podría pensar de un objeto del dispositivo como un contenedor de canales o líneas que refleja la funcionalidad de un dispositivo en particular. La funcionalidad de un dispositivo aplica a todos los canales o líneas que este contenga. Por ejemplo, la tasa de muestreo de un objeto de entrada análoga aplica a todas los canales contenidos por el objeto. Por el contrario, los canales y las líneas contenidas por el objeto del dispositivo reflejan la funcionalidad de un canal o una

línea en particular. Se agregan canales a un objeto de entrada o salida análoga con la función addchannel. Esta función requiere el objeto de dispositivo y al menos un ID de canal de hardware como argumentos de entrada. Se puede especificar opcionalmente los índices de MATLAB, nombres de canales y argumento de salida

Ahora, vamos a configurar el objeto de la entrada análoga de modo que adquiramos 40 segundos de muestras. Ya que se van a tomar 100 muestras por segundo en 4000 disparos.

```
AI.SampleRate = 100;
AI.SamplesPerTrigger = 4000;
AI.TriggerType = 'Immediate';)
```

Con la propiedad SampleRate se controla la tasa en la cual un subsistema de entrada análoga convierte los datos análogos a datos digitales. SampleRate debe ser especificado como muestras por segundo.

Cuando un trigger se ejecuta, un número predefinido de muestras es adquirido por cada miembro del grupo de canales y es almacenada en el motor o en un archivo en el disco. Se especifica el número de muestras a tomar por disparado con la propiedad SamplesPerTrigger.

Para objetos de entrada análoga, un trigger es definido como un evento que inicia el almacenamiento de datos a la memoria o a un archivo en el disco. Definir un trigger de entrada análoga requiere especificar el tipo de disparo con la propiedad TriggerType,

Con el objeto de la entrada análoga creado, vamos a empezar y luego obtenemos los datos.

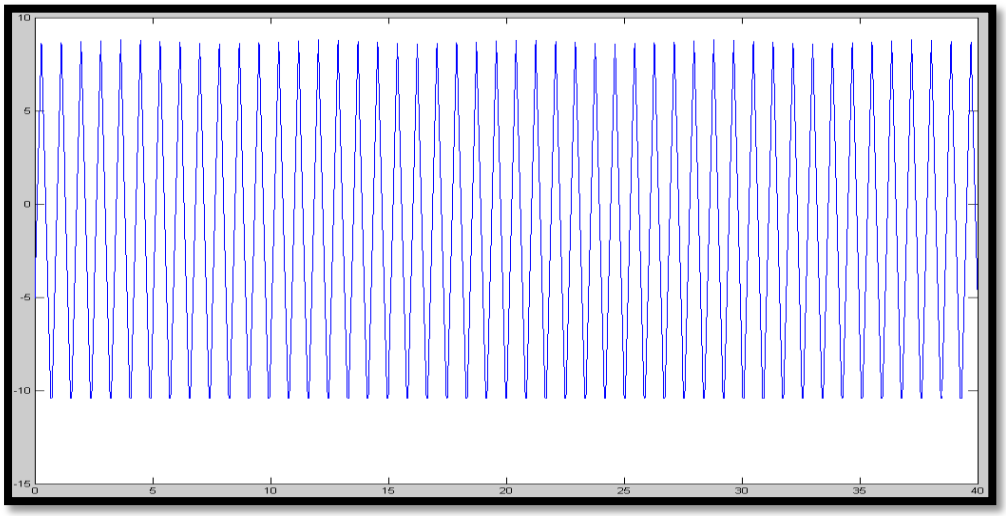
```
start(ai)
```

Se arranca un objeto de entrada análoga con la función start, Después de ejecutar start, la propiedad Running se establece automáticamente a ON, y tanto el objeto del dispositivo como el hardware se ejecutarán de acuerdo con los valores de las propiedades por defecto y configurados.

Mientras el objeto de entrada análoga está corriendo, se puede almacenar los datos adquiridos en el motor (memoria) o en un archivo en el disco. Sin embargo, antes que se puedan almacenar datos, un trigger tiene que ocurrir. Se configura un trigger de entrada análoga con la propiedad TriggerType.

Cuando un trigger ocurre, la propiedad Logging se establece automáticamente a ON y los datos adquiridos desde el hardware son almacenados en el motor en un archivo en el disco. Se pueden extraer datos almacenados en el motor con la función getdata. Getdata bloquea la ventana de comandos de MATLAB® hasta que los datos pedidos son retornados al área de trabajo. Se pueden extraer datos en cualquier momento después que un disparo ocurra. También se pueden retornar las muestras con los tiempos en los que fueron adquiridas con getdata.
Ahora que haya terminado la recolección de datos, grafique.

```
plot(t,d);
zoom on
```



Grafica 24. Entrada análoga con la tarjeta NI USB 6008.

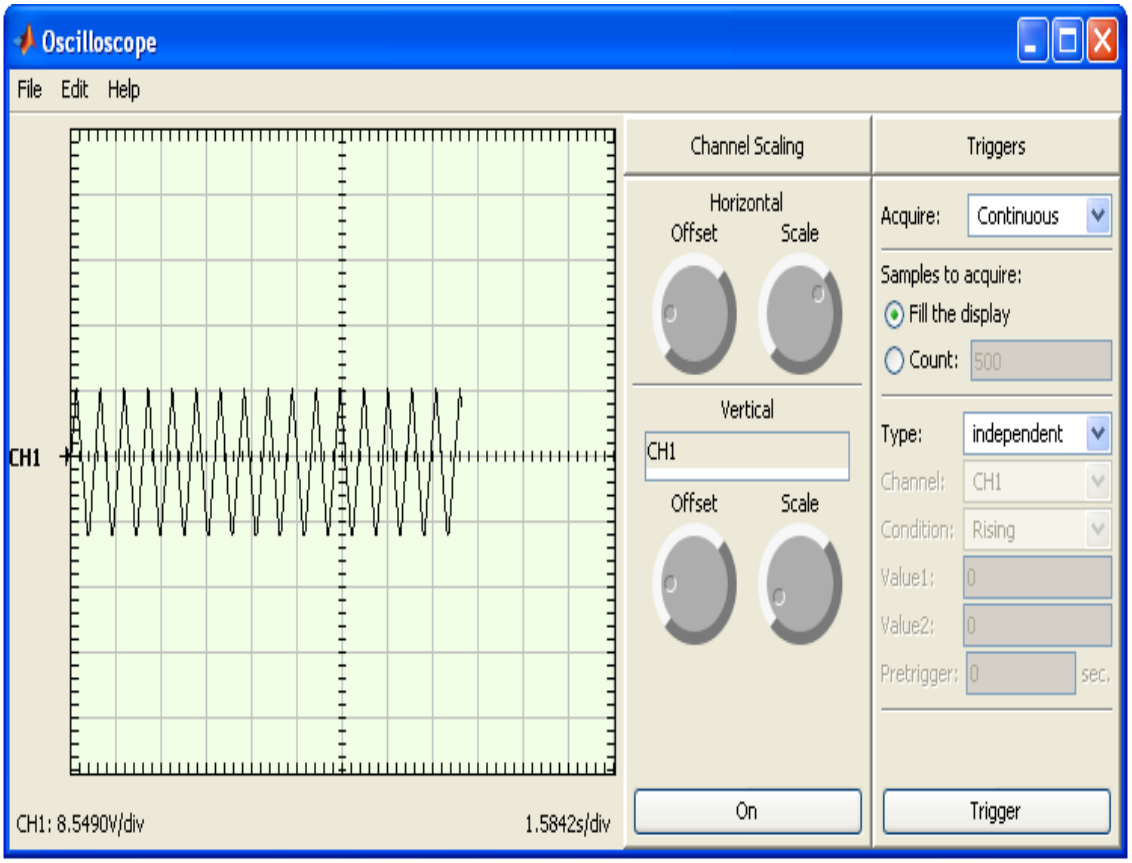
Anteriormente, se utilizo el GETDATA para recolectar 40 segundos de datos. El único problema es que GetData es una función “bloque”. Esto significa que va a esperar hasta los 40 segundos a que los datos hayan sido recogidos. Para muchas aplicaciones, esto no es lo que se requiere. La solución a esto es utilizar la función PEEKDATA. Para este ejemplo, vamos a configurar un número ilimitado de disparos y el uso PEEKDATA para ver lo que está sucediendo, Mientras que se adquieren datos con un objeto de entrada análoga, se pueden mirar los datos de antemano con la función peekdata. Esta función toma una “foto” de los datos más recientes pero no remueve los datos del motor.

```
AI.TriggerRepeat = inf;
start(AI);
pause(0.3);
pd1 = peekdata(AI,1000);
pause(0.3);
pd2 = peekdata(AI,1000);
pause(0.3);
pd3 = peekdata(AI,1000);
softscope(AI)
```

Observe que la función PEEKDATA no bloqueó y que sólo devuelve los datos que están disponibles. Pedimos 8000 muestras, pero no necesariamente se obtiene esa cantidad.

```
whos pd1 pd2 pd3
```

Name	Size	Bytes	Class	Attributes
pd1	1000x1	8000	double	
pd2	1000x1	8000	double	
pd3	1000x1	8000	double	



Grafica 25. Entrada análoga con la tarjeta NI USB 6008 uso del PEEKDATA.

Esto completa la introducción a los objetos de entrada analógica. Puesto que el objeto de entrada analógica ya no es necesario, usted debe:

Primero para parar el objeto de entrada analógica se ejecuta el comando STOP. Por último, con el comando DELETE para liberar memoria y otros recursos físicos.

stop(AI);
delete(AI);

4.2 EXPERIENCIA PARA EL ACCESO A CANALES DE ENTRADA ANALOGA

En esta experiencia se dan ejemplos sobre el uso de la notación get / set, la notación de puntos, y el nombramiento de índices de notación para obtener información sobre el canal y para configurar el canal para su adquisición. Se aprenderá acerca de la creación, el acceso y configuración de canales de entrada analógica.

En primer lugar, encuentre objetos que se estén ejecutándo dentro de la Toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera. Este código no suele ser necesario a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
   stop(daqfind)
end
```

Para empezar, se crea un objeto AI de entrada análoga para la tarjeta NI USB 6008.

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

Display Summary of Analog Input (AI) Object Using 'USB-6008'.

```
 Acquisition Parameters:  1000 samples per second on each channel.
                          1000 samples per trigger on each channel.
                          1 sec. of data to be logged upon START.
                          Log data to 'Memory' on trigger.
```

Trigger Parameters: 1 'Immediate' trigger(s) on START.

Engine status: Waiting for START.
0 samples acquired since starting.
0 samples available for GETDATA.

AI object contains no channels.

Usted puede agregar un canal a un objeto de entrada analógica con el comando ADDCHANNEL. ADDCHANNEL necesita al menos dos argumentos de entrada. El primer argumento especifica el objeto de entrada analógica del canal que se añade. El segundo argumento es el identificador del hardware de los canales que se están agregando al objeto de entrada analógica. Para la tarjeta de sonido, puede agregar hasta dos canales. Si un canal se añade, el sonido se graba y reproduce en mono. Si se agregan dos canales, el sonido se graba y reproduce en estéreo.

addchannel(AI, 1)

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'

Hay tres métodos para acceder a un canal. En el primer método, el canal se accede a través de la propiedad del objeto de entrada analógica del Canal. Por ejemplo, usando la notación punto, los canales de todos los objetos de entrada analógica serán asignados a la variable ch1.

ch1 = AI.Channel;

O, usando la notación GET

```
ch1 = get(AI, 'Channel');
ch1
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  1      ''          1          [-10 10]     [-10 10]      [-10 10]     'Volts'
```

En el segundo método, el canal se accede en tiempo de creación mediante la asignación de una variable de salida a ADDCHANNEL. El tercer argumento de entrada para ADDCHANNEL asigna el nombre de Chan2 al canal añadido.

```
addchannel(AI, 2, 'Chan2')
```

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  2      'Chan2'      2          [-10 10]     [-10 10]      [-10 10]     'Volts'
```

En el tercer método, el canal se accede a través de la propiedad ChannelName. Este método se conoce como “Referenciando un nombre”. El segundo canal añadido para el objeto de entrada análoga fue asignado el ChannelName 'Chan2. Por lo tanto, se puede acceder al segundo canal con el comando:

```
ch2 = AI.Chan2;
 ch2
```

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  2      'Chan2'      2          [-10 10]     [-10 10]      [-10 10]     'Volts'
```

Los canales que tienen el mismo objeto (o se asocian con el objeto de la misma entrada analógica) se pueden concatenar en cualquiera de los vectores fila o columna. Se producirá un error si intenta concatenar canales en una matriz.

```
ch = [ch1 ch2];

s = size(ch);

 s

s =

    1     2
```

Ó

```
ch = [ch1; ch2];
s1 = size(ch);
s1

s1 =

    2     1
```

Se puede acceder a canales específicos, especificando el índice del canal en el objeto de entrada análoga.

```
ch1 = AI.Channel(1);

ch2 = AI.Channel(2);

ch1
```

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'

ch2

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

2 'Chan2' 2 [-10 10] [-10 10] [-10 10] 'Volts'

O, si un canal matriz de 1-por-5 se crea, es posible acceder al primer, tercer y cuarto canal de la siguiente manera:

```
charray = [ch1 ch2 ch1 ch1 ch2];
temp = charray([1 3 4]);
charray
temp
```

```
charray = [ch1 ch2 ch1 ch1 ch2];
>> temp = charray([1 3 4]);
charray
temp
```

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'
2 'Chan2' 2 [-10 10] [-10 10] [-10 10] 'Volts'
1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'
1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'
2 'Chan2' 2 [-10 10] [-10 10] [-10 10] 'Volts'

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
    1       ''          1          [-10 10]     [-10 10]      [-10 10]     'Volts'
    1       ''          1          [-10 10]     [-10 10]      [-10 10]     'Volts'
    1       ''          1          [-10 10]     [-10 10]      [-10 10]     'Volts'
```

Puede obtener información sobre el canal con el comando GET. Al poner el comando GET acompañado con el canal se obtiene una lista de todas las propiedades y sus valores actuales.

Las propiedades comunes se muestran primero. Si existen propiedades específicas aparecerán en el segundo lugar.

```
get(ch1)
        ChannelName =
        HwChannel = 1
        Index = 1
        InputRange = [-10 10]
        NativeOffset = 0
        NativeScaling = 1
        Parent = [1x1 analoginput]
        SensorRange = [-10 10]
        Type = Channel
        Units = Volts
        UnitsRange = [-10 10]
```

Si una variable es suplida con el comando GET, la variable toma la estructura del canal y todas sus propiedades.

```
h = get(ch2);
```

h

h =

ChannelName: 'Chan2'

HwChannel: 2

Index: 2

InputRange: [-10 10]

NativeOffset: 0

NativeScaling: 1

Parent: [1x1 analoginput]

SensorRange: [-10 10]

Type: 'Channel'

Units: 'Volts'

UnitsRange: [-10 10]

Se puede obtener información relacionada de una propiedad específica, especificando el nombre de la propiedad como el segundo argumento en el comando GET (por ejemplo ch1).

Si la matriz de canal no es 1-por-1, una matriz de valores de la propiedad es retornada.

get(ch1, 'Units')

get(ch, 'Units')

ans =

Volts

ans =

'Volts'

'Volts'

Se puede obtener información relacionada con múltiples propiedades mediante la especificación de un conjunto de lista de nombres de propiedades como el argumento de la segunda entrada a GET (por ejemplo, ch2).

El valor de las propiedades de la matriz devuelta se refiere a cada objeto y nombre de la propiedad de la siguiente manera:

Tenga en cuenta que también puede utilizar la notación de puntos para obtener información acerca de cada canal.

ch1.Units
AI.Channel(2).ChannelName

ans =

Chan2

También puede obtener información de los canales a través de la propiedad CnajjelName.

AI.Channel(2).ChannelName

ans =

Chan2

>> AI.Chan2.InputRange

ans =

  -10   10

Puede asignar valores a las propiedades de diferentes canales con el comando SET. Si se ejecuta SET acompañado del nombre del canal, una lista de propiedades configurables y sus posibles valores son devueltos. Al igual que el GET, las propiedades comunes se muestran primero. Si cualquiera de las propiedades específicas del dispositivo de canal existe, aparecerán en segundo lugar.

set(ch1)

    ChannelName

    HwChannel

    InputRange

    SensorRange

    Units

    UnitsRange

Si se proporciona una variable de salida a compañada con SET, se devuelve una estructura donde los nombres son nombres de campo de la estructura del canal, propiedad y estructura de los valores de campo son los valores de las propiedades posibles para la propiedad del canal.

h = set(ch1);

h

h =

```
    ChannelName: {}
      HwChannel: {}
     InputRange: {}
    SensorRange: {}
          Units: {}
     UnitsRange: {}
```

Puede obtener información sobre posibles valores de la propiedad para una propiedad específica, especificando el nombre de la propiedad como el argumento de la segunda entrada a SET.

```
set(ch1, 'Units');
```

O bien, puede asignar el resultado a una variable de salida.

```
unit = set(ch1, 'Units');
```

Se puede asignar valores a una propiedad específica de la siguiente manera:

```
set(ch1, 'SensorRange', [-2 2]);
ch
```

Index:	ChannelName:	HwChannel:	InputRange:	SensorRange:	UnitsRange:	Units:
1	''	1	[-10 10]	[-2 2]	[-10 10]	'Volts'
2	'Chan2'	2	[-10 10]	[-10 10]	[-10 10]	'Volts'

Puede asignar la misma propiedad a múltiples canales con una sola llamada a SET.

```
set(ch, 'SensorRange', [-3 3]);
ch
```

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  1     ''          1         [-10 10]    [-3 3]      [-10 10]    'Volts'
  2     'Chan2'     2         [-10 10]    [-3 3]      [-10 10]    'Volts'
```

Puede asignar diferentes valores a una propiedad, a múltiples canales mediante la especificación de un conjunto de valores de propiedades matriz con el comando SET

```
set(ch, {'SensorRange'}, {[-2 2];[-1 1]});
ch
```

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  1     ''          1         [-10 10]    [-2 2]      [-10 10]    'Volts'
  2     'Chan2'     2         [-10 10]    [-1 1]      [-10 10]    'Volts'
```

Este concepto puede ser extendido para asignar múltiples propiedades de valores diferentes para múltiples canales. El conjunto de Matrices de valores deben ser m por n, donde m es el número de objetos y n es el número de propiedades. Cada fila del conjunto de valores de propiedades matriz, contiene el valor de las propiedades de un objeto único. Cada columna del conjunto de valores de propiedades contiene el valor de propiedad para una sola propiedad.

El valor de las propiedades asignadas se refiere a cada objeto y nombre de la propiedad como se muestra a continuación:

```
set(AI.Channel, {'Units'; 'UnitsRange'}, {'LeftUnits', [0 30]; 'RightUnits', [0 40]});
 AI.Channel

  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:
Units:
  1      ''           1          [-10 10]     [-2 2]        [0 30]       'LeftUnits'
  2      'Chan2'      2          [-10 10]     [-1 1]        [0 40]       'RightUnits'
```

Puede utilizar la notación de puntos para asignar valores a las propiedades del canal.

```
ch1.Units = 'OneUnits';
 ch2.SensorRange = [-4 4];
ai.Channel(1).ChannelName = 'Chan1';
 ai.Channel

ans =

    ChannelName: 'Chan1'
```

O bien, puede asignar valores a las propiedades del canal mientras hace referencia a su nombre.

```
ai.Chan2.Units = 'TwoUnits';
 ai.Channel

ans =

    ChannelName: 'Chan1'

ch1.Units = 'OneUnits';
ch2.SensorRange = [-4 4];
```

```
AI.Channel(1).ChannelName = 'Chan1';

AI.Chan2.Units = 'TwoUnits';
AI.Channel

  Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
    1      'Chan1'      1         [-10 10]    [-2 2]       [0 30]      'OneUnits'
    2      'Chan2'      2         [-10 10]    [-4 4]       [0 40]      'TwoUnits'
```

Usted puede convertir las señales de entrada analógica en valores que representan unidades específicas de ingeniería mediante la especificación de las siguientes propiedades de canal:

SensorRange - rango esperado de los datos del sensor.
InputRange - Gama de convertidor A / D.
UnitsRange - Gama de datos en MATLAB ® área de trabajo.
Unidades - Unidades de Ingeniería nombre.

Por ejemplo, supongamos que usted está usando un micrófono que puede medir los niveles sonoros de hasta 120 dB y este rango produce un voltaje de salida de -0,5 a 0,5 voltios. Su convertidor A / D debe tener un rango de voltajes validos que se pueden establecer. Lo mejor del rango del hardware A / D es el que abarca las expectativas del rango del sensor.

La configuración del canal se indica a continuación:

```
set(ai.Channel, 'SensorRange', [-0.5 0.5]);

set(ai.Channel, 'InputRange', [-1 1]);

set(ai.Channel, 'UnitsRange', [0 120]);

set(ai.Channel, 'Units', 'dB');
```

Las unidades solo pueden ser las usadas en ingeniería y son de tipo lineal. Puede realizar conversiones no lineales mediante la creación de la función adecuada M-file.

4.3 EXPERIENCIA DEL USO DEL TRIGGER EN LA ENTRADA ANALOGA.

En esta experiencia se explorara cada una de las funciones del trigger. Las propiedades del trigger incluyen las siguientes demostraciones: TriggerType, TriggerRepeat, TriggerDelay, TriggerDelayUnits, TriggerChannel, TriggerCondition y TriggerConditionValue.

Para hacer esta demostración, se debe usar la tarjeta NI USB 6008 y proporcionarle una señal de entrada.

En primer lugar, encuentre objetos que se estén ejecutándose dentro de la Toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

En esta experiencia, vamos a explorar las propiedades relacionadas con el trigger y adquirir 3000 muestras usando trigger manual, inmediato y software.

4.3.1 Definicion de Triggers

Un triggers se define como un evento que inicia el registro de datos en la memoria y / o un archivo de disco. El registro de estado es indicado por la propiedad de registro. Cuando se produce un disparo, la propiedad de registro se establece en On. El destino de los datos registrados es indicado por la propiedad LoggingMode. Se puede registrar los datos en la memoria o un archivo de disco de forma predeterminada, los datos se registran en la memoria.

Un objeto de entrada analógica puede registrar los datos usando un trigger inmediato, un trigger manual, o un trigger de software. El tipo de trigger se indica mediante la propiedad TriggerType. Un trigger inmediato es el tipo de disparo por defecto.

4.3.2 Triggers inmediato

Un trigger inmediato comienza el registro de datos inmediatamente después del comando START.

En primer lugar, vamos a crear el objeto de entrada análoga y añadir dos canales a la misma. El objeto de entrada analógica se ha configurado para adquirir 1000 muestras por segundo.

```
AI = analoginput('nidaq','Dev3');
addchannel(AI,1:2);

set(AI, 'SampleRate', 1000);
```

El número de muestras de datos que el trigger adquirirá está indicado por la propiedad SamplesPerTrigger. El objeto de entrada analógica se ha configurado para adquirir 3.000 muestras por trigger. El objeto también está configurado para adquirir datos de forma inmediata.

```
set(AI, 'SamplesPerTrigger', 3000);

set(AI, 'TriggerType', 'immediate');
```

Tan pronto como el objeto de entrada analógica se inicia, el disparo se producirá. Cuando el trigger se ejecuta, el número de muestras especificadas por la propiedad se adquieren “SamplesPerTrigger” para cada canal y se almacena en el motor de adquisición de datos.

```
start(ai);
```

Usted puede recuperar los datos desde el motor de adquisición de datos con la función GETDATA.

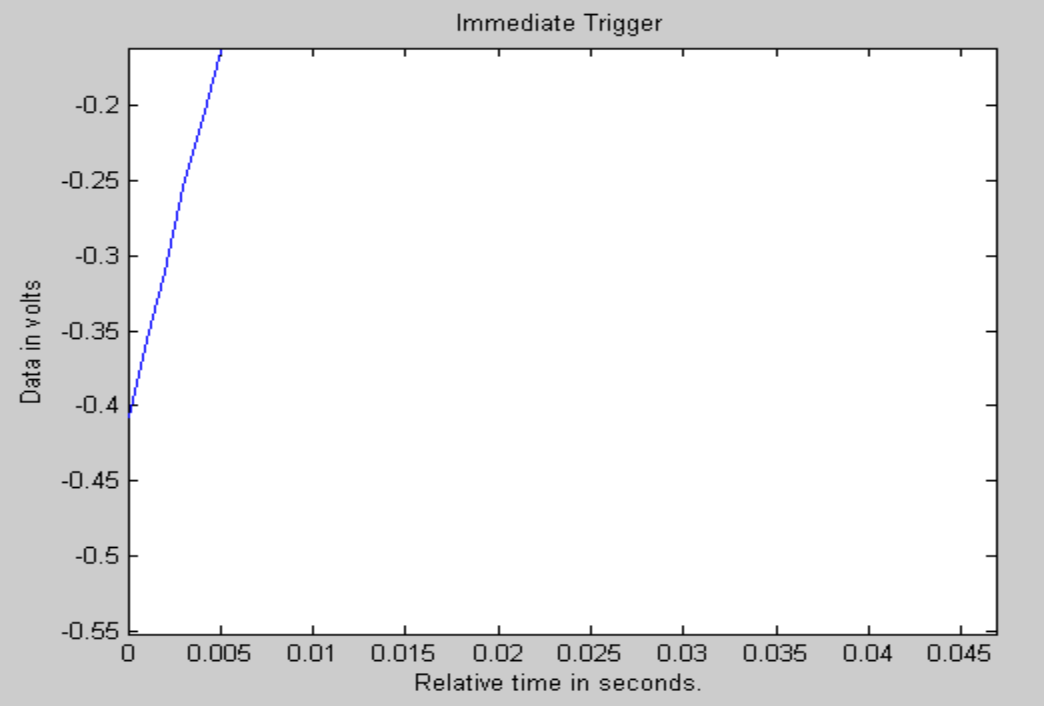
El tamaño de los datos será el número de muestras por disparo por el número de canales.

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

ans =

      3000           2
```

Los datos pueden ser visualizados en función del tiempo con el trigger empezando en el tiempo = 0. Como puede ver, tomó 0,375 segundos para obtener los datos. Esta vez se calcula tomando la relación SamplesPerTrigger / muestreo.

```
plot(time,data);
```

```
zoom on;
title('Immediate Trigger');
xlabel('Relative time in seconds.');
ylabel('Data in volts');
```



Grafica 26. Disparo con la propiedad trigger inmediato.

4.3.3 Repeticiones de un trigger

Puede configurar los triggers para que se produzcan repetidas veces. La repiticion del Trigger se controla por la propiedad TriggerRepeat. Cuando la propiedad TriggerRepeat se ajusta a su valor predeterminado de 0, el disparador se producirá una vez. Si la propiedad TriggerRepeat se establece en un entero positivo, entonces el trigger se repite el número especificado de tiempo en que la condición de disparo se cumple. Si se establece en TriggerRepeat Inf, a continuación, el trigger se repite continuamente cuando la condición de activación se cumple y la adquisición de datos se puede detener sólo con la intervención de

parada que es el comando STOP.
Con un trigger inmediato, cada disparo se producirá inmediatamente después del disparo anterior ha terminado de ejecutarse.

Vamos a ejecutar el objeto de entrada analógica

set(AI, 'TriggerType', 'immediate');

>> set(AI, 'SamplesPerTrigger', 1500);

>> set(AI, 'TriggerRepeat', 1);

>> start(AI);

Se puede recuperar los datos desde el motor de adquisición de datos con la función GETDATA.

El primer comando extrae las primeras 1500 muestras procedentes del motor de adquisición de datos.

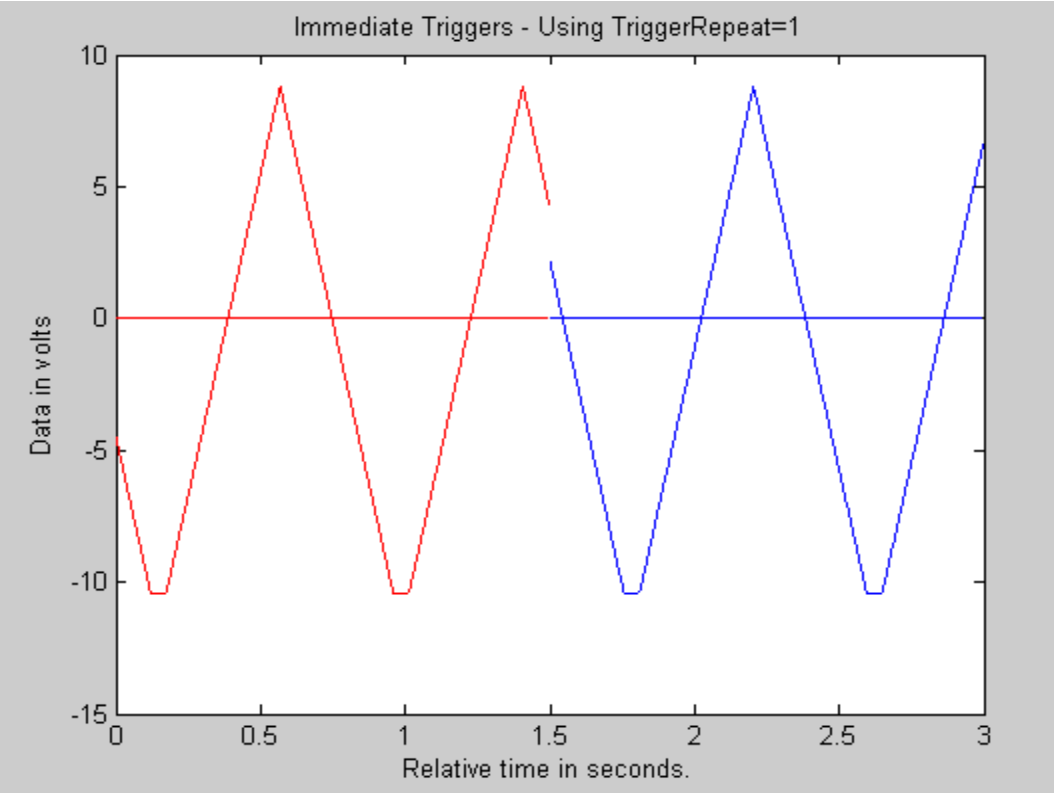
El segundo comando extrae las siguientes 1500 muestras procedentes del motor de adquisición de datos.

```
[data1,time1] = getdata(AI);
[data2,time2] = getdata(AI);
```

Puede visualizar los datos en función del tiempo desde el primer trigger que ocurre en el tiempo = 0.

```
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
```



Grafica 27. Disparo con la propiedad trigger Repeat=1

4.3.4 Triggers con retardo

El Trigger con retardo le permiten controlar con exactitud cuando los datos se registran después de que un disparo se ejecuta. Usted puede registrar los datos ya sea antes de un trigger ejecutado o después de un trigger que se ejecuta. El Trigger con retardo se especifica con la propiedad TriggerDelay.

Cuando el registro de los datos se produce antes de un trigger se llama pretriggering, mientras que el registro de datos después de un trigger se llama postriggering. Un pretriggering se especifica mediante un valor de propiedad

TriggerDelay negativo, mientras que un postriggering se especifica con un valor de la propiedad TriggerDelay positivo.

Se puede retrasar triggers ya sea en segundos o en muestras mediante la propiedad TriggerDelayUnits.

Configuremos el objeto de entrada análoga para que adquiera un total de 3000 muestras. 1000 muestras serán adquiridas antes del trigger manual y 2000 muestras serán adquiridos después del trigger manual. .

```
set(AI, 'TriggerType', 'manual');
set(AI, 'SamplesPerTrigger', 3000);
set(AI, 'TriggerDelay', -1000);
set(AI, 'TriggerDelayUnits', 'samples');
```

El objeto de entrada analógica se inicia y el motor de adquisición de datos se ejecutará.

```
start(AI);
```

```
status = get(AI, 'Running')
```

```
status =
```

On

La toolbox necesita una oportunidad para adquirir los datos del pretrigger. Vamos a una pausa por 1,5 segundos para permitir que los datos se acumulen en la memoria intermedia pretriggering

```
pause(1.5);
```

La entrada análoga se ejecuta después del trigger y luego se espera hasta dos segundos para que la adquisición se complete.

trigger(AI);

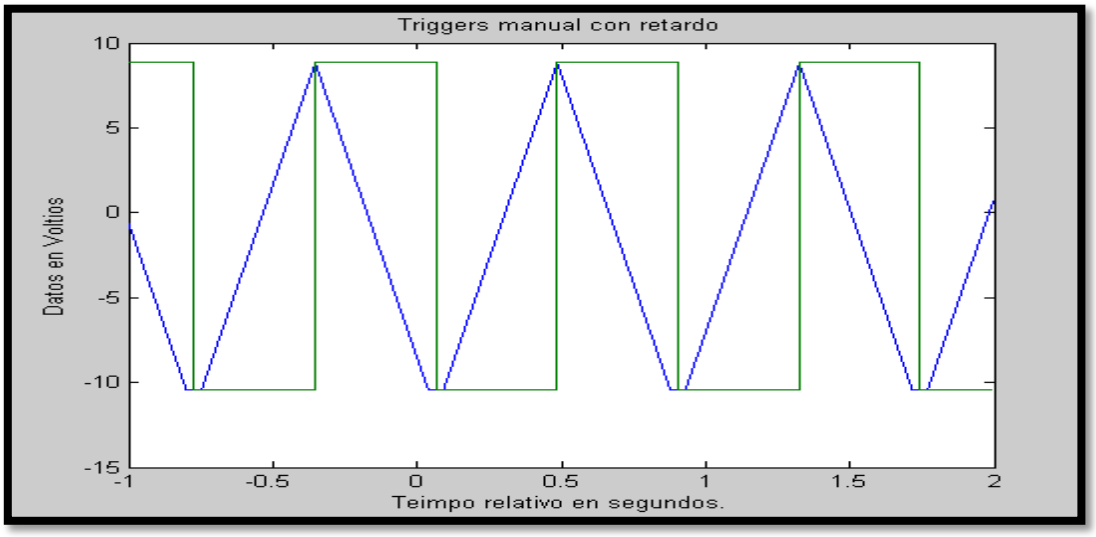
wait(AI,2)

Las 3000 muestras se encuentran almacenadas en el motor de adquisición de datos, para poderlas tomar se utiliza el comando GETDATA.

Los datos pueden ser visualizados en función del tiempo a partir del primer trigger que ocurren en el tiempo = 0. Por lo tanto, los datos pretriggered se trazarán con un valor negativo de tiempo y los datos adquiridos después del disparo se trazarán con un valor de hora positivo.

[data,time] = getdata(AI);

plot(time,data);
zoom on;
title('Triggers manual con retardo');
xlabel('Teimpo relativo en segundos.');
ylabel('Datos en Voltios');



Grafica 28. Triggers manual con retardo.

4.3.5 Software triggers

Un software trigger comienza el registro de datos cuando una señal que cumple una condición especificada se detecta en uno de los canales que se especifiquen.

El canal utilizado como fuente de trigger se define por la propiedad TriggerChannel. La condición que debe cumplirse para que se produzca un trigger es especificado por la propiedad TriggerCondition. Usted puede establecer esta propiedad a uno de los siguientes valores:

Rising - La señal debe estar por encima del valor especificado.
Falling - La señal debe estar por debajo del valor especificado.
Leaving La señal debe salir del rango especificado de los valores.
Entering - La señal debe estar entrando en el rango especificado de los valores.

El valor especificado o el rango que la condición de disparo debe cumplir es indicado por la propiedad TriggerConditionValue.

Vamos a configurar el objeto de entrada analógica, para adquirir muestras de 1000 a 3000 muestras por segundo con el software trigger. El trigger se producirá cuando una señal en el primer canal tiene un flanco de subida y pasa a través de 2 voltios..

```
set(AI, 'TriggerType', 'software');

set(AI, 'TriggerRepeat', 0);

 set(AI, 'TriggerCondition', 'rising');

 set(AI, 'TriggerConditionValue', 2);
```

set(AI, 'TriggerChannel', AI.channel(1));

Para capturar una segunda tanda de datos antes de que el trigger ocurra el motor de adquisición de datos va a esperar durante 2 segundos, esta es una condición que debe cumplir antes de detenerse.

También se ejecuta el objeto de entrada análoga.

set(AI, 'TriggerDelay', -1);

set(AI, 'TriggerDelayUnits', 'seconds');

set(AI, 'TimeOut', 2);

start(AI)

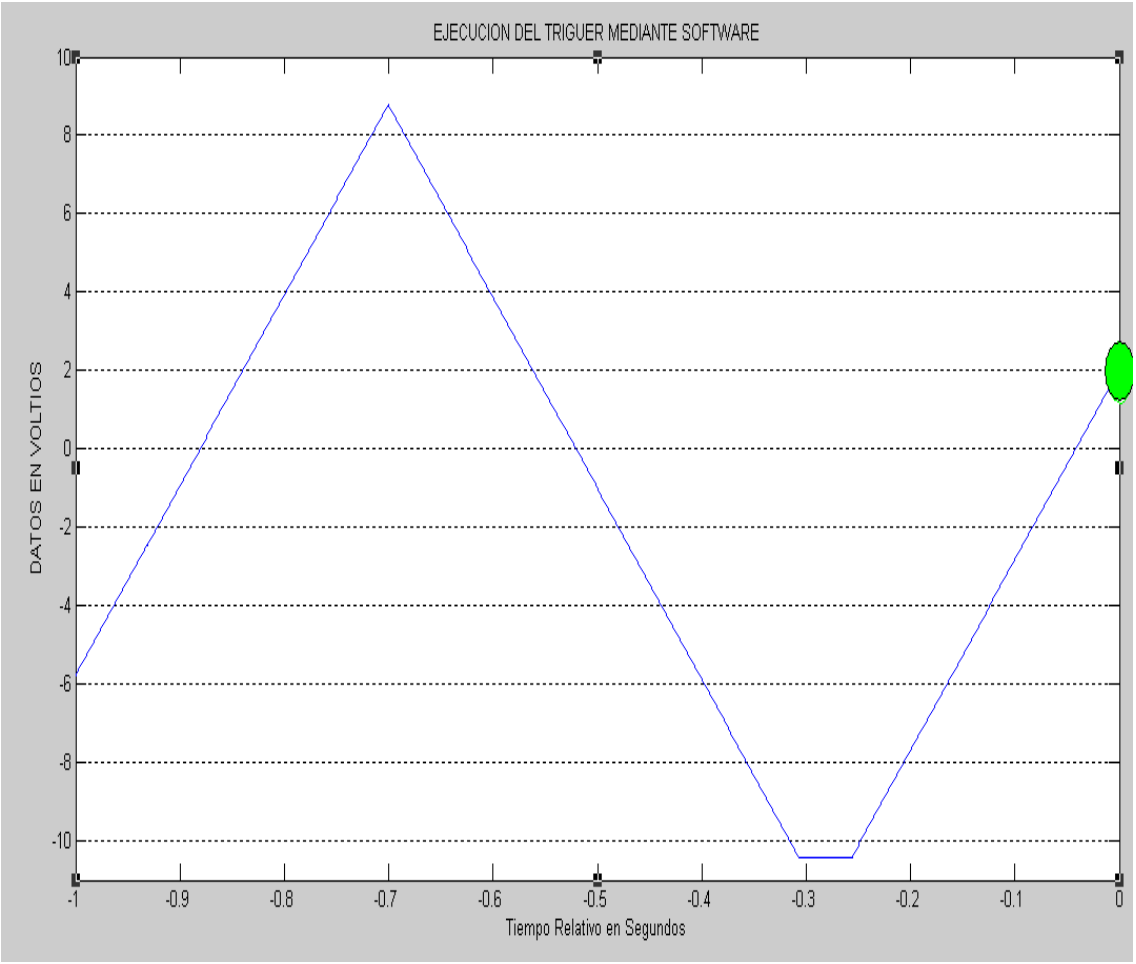
Las 2000 muestras se encuentran almacenadas en el motor de adquisición de datos, para poderlas tomar se utiliza el comando GETDATA.

Si la condición del trigger no se cumplió, GETDATA tendría un tiempo de espera después de dos segundos y ningún dato será tomado.

Los datos pueden ser visualizados en función del tiempo a partir del primer trigger que ocurren en el tiempo = 0.

```
disp. ('Se debe suministrar una señal mayor que 2 voltios para activar el 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');
```

Debe proporcionar una señal mayor de 2 voltios para activar el gatillo



Grafica 29. Disparo con la propiedad Software Trigger.

Ahora elimine el objeto de entrada analógica.

delete(ai)

clear all

clc

5 SALIDA ANALOGA

5.1 EXPERIENCIA DE INTRODUCCION A LA SALIDA ANALOGA

Esta experiencia ilustra una adquisición de datos básica usando un objeto de salida análoga. Esto se demuestra mediante la configuración de las propiedades de salida análoga, la adición de un canal con el objeto de salida analógica y la salida de datos a una tarjeta de sonido.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty (daqfind))
   stop (daqfind)
end
```

En este ejemplo, podrá sacar datos desde la tarjeta de sonido de su computador. Para empezar, primero debe verificar que su entorno está configurado de modo que usted puede enviar datos a la tarjeta de sonido.

En esta demostración, una onda sinusoidal con dos diferentes frecuencias se emitirá desde la tarjeta de sonido.

Para empezar, vamos a crear un objeto de salida análoga asociado con el dispositivo winsound. Dos canales serán añadidos al objeto de salida análoga, ao. Esto permitirá que el dispositivo winsound pueda ejecutarse en modo estéreo.

Para crear un objeto de salida análoga se utiliza la función *analogoutput.* Esta función acepta el nombre y el ID del adaptador de hardware como argumentos. El ID de dispositivo se refiere a el número asociado con la tarjeta cuando ésta es instalada. Algunos vendedores se refieren al ID del dispositivo como el número del dispositivo o al número de la tarjeta.
Cada objeto de salida análoga es asociado con una tarjeta y un subsistema de salida análoga.

```
ao = analogoutput ('winsound');
addchannel (ao, [1 2]);
```

Después de crear el objeto, se debe adherirle canales. La colección de canales contenida por el objeto es referida con un *grupo de canales.* Un grupo de canales consiste en un mapeo entre los Id de los canales del hardware y los índices de MATLAB.

Para agregarle canales a un objeto de salida análoga hay que tener en cuenta estas reglas:

- Los canales deben ser del mismo hardware. No se puede agregar canales de diferentes dispositivos o de diferentes subsistemas del mismo dispositivo.
- Los canales deben ser muestreados con la misma tasa de muestreo.

Los canales son agregados al objeto con la función *addchannel*. Esta función requiere el objeto y por lo menos un ID como argumentos de entrada.

Vamos a configurar el objeto de salida análoga para que saque datos a 8000Hz.

```
set(ao, 'SampleRate', 8000);
```

Ahora, vamos a crear los datos de salida a la tarjeta de sonido - una onda sinodal en dos frecuencias diferentes.

```
t = linspace(0,2*pi,16000);
```

y = sin(500*t);

y1 = sin(800*t);

OUTPUTTING DATA. Outputting data, involucra las funciones que administran la cola del motor de adquisición y funciones que ejecutan el objeto de salida análoga.

QUEUING DATA. Para los datos que van a enviarse, primero se deben poner en cola en el motor de adquisición de datos con el comando PUTDATA.

putdata(ao, data);

Los datos son una matiz de m por n, donde m es el número de muestras que van a enviarse y n es el número de canales de salida.

putdata(ao, [y' y']);
putdata(ao, [y1' y1']);

El objeto de la salida analógica y el motor de adquisición de datos se inicia con el comando START. Iniciar el objeto de salida significa que el dispositivo de hardware y el motor de adquisición de datos están en ejecución. Correr no significa necesariamente que los datos estén saliendo. Para que los datos se envíen un trigger debe ocurrir.

Para configurar el objeto de salida análoga para que se active inmediatamente después de un START se usa el trigger inmediata.

set(ao, 'TriggerType', 'Immediate');

Recuerde que siempre puede escribir el nombre del objeto para obtener un resumen de su configuración actual.

ao
Display Summary of Analog Output (AO) Object Using 'NVIDIA(R) nForce(TM) Audio'.

```
      Output Parameters:  8000 samples per second on each channel.
     Trigger Parameters:  1 'Immediate' trigger on START.
        Engine status:  Waiting for START.
                        4 total sec. of data currently queued for START.
                        32000 samples currently queued by PUTDATA.
                        0 samples sent to output device since START.
AO object contains channel(s):
  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:
  1       'Left'        1           [-1 1]        [-1 1]       'Volts'
  2       'Right'       2           [-1 1]        [-1 1]       'Volts'
```

El objeto de salida analógica se ha iniciado, y debe esperar 5 segundos para que la salida se complete.

```
start(ao);

wait(ao,5);
```

Se puede repetir varias veces los datos que se encuentran en cola utilizando la propiedad RepeatOutput. Si RepeatOutput es mayor que 0, entonces todos los datos en cola se repetirán el número especificado de veces según sea el comando de repeticiones.

Vamos a configurar el objeto de salida análoga para de enviar los datos a la tarjeta de sonido dos veces (o repite una vez).

```
set(ao, 'RepeatOutput', 1);
putdata(ao, [y' y']);
putdata(ao, [y1' y1']);
```

El objeto de salida analógica se ha iniciado, y esperan hasta 9 segundos para que se complete la salida.

```
start(ao);
```

wait(ao,9);

Esto completa la introducción a la salida análoga. Puesto que el objeto de salida análoga ya no es necesario, debe eliminarlo con el comando DELETE para liberar memoria y otros recursos físicos.

delete(ao);

5.2 EXPERIENCIA PARA EL ACCESO A CANALES EN LA SALIDA ANALOGA.

En esta experiencia se crea canales de salida analógica mediante la demostración de cómo agregar canales a un objeto de salida analógica y cómo configurar los canales para su adquisición. Se da ejemplos sobre el uso de get / set de notación, la notación de puntos, y la invocación de la notación de índice para obtener información sobre el canal y para configurar el canal para su adquisición.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

En este ejemplo, usted aprenderá acerca de la creación, el acceso y configuración de canales de salida analógica.

Para empezar, debemos crear el objeto de salida analógica, AO, asociado con el dispositivo winsound.

ao = analogoutput('winsound');

Se puede agregar un canal a un objeto de salida análoga con el comando ADDCHANNEL. ADDCHANNEL necesita al menos dos argumentos de entrada. El primer argumento especifica el nombre del objeto que se añade (AO). El segundo argumento es el identificador (ID) de los canales de hardware que se está agregando al objeto de salida analógica.

Para la tarjeta de sonido se puede agregar hasta dos canales. Si un canal se añade, el sonido se graba y se reproduce en mono, si se agregan dos canales, el sonido se graba y se reproduce en estéreo.

addchannel(ao, 1); % Agregando un canal

Hay tres métodos para acceder a un canal. En el primer método, el canal se accede a través de la propiedad del objeto de salida análoga del Canal. Por ejemplo, usando la notación de puntos, los canales de todos los objetos de salida analógica serán asignados a la variable ch1.

O, usando la notación siguiente:

ch1 = ao.Channel;

ch1 = get(ao, 'Channel');

ch1

Index:	ChannelName:	HwChannel:	OutputRange:	UnitsRange:	Units:
1	'Mono'	1	[-1 1]	[-1 1]	'Volts'

En el segundo método, el canal es accedido para crear tiempos para asignar una variable de salida para ADDCHANNEL.

El tercer argumento de entrada para ADDCHANNEL asigna el nombre Chan2 para el canal creado.

```
ch2 = addchannel(ao, 2, 'Chan2');
ch2
  Index: ChannelName: HwChannel: OutputRange: UnitsRange: Units:
  2      'Chan2'      2
```

En el tercer método, el canal se accede a través la propiedad ChannelName. Este método se conoce como “Nombre de Referencia”. El segundo canal añadido para el objeto de entrada análoga fue asignado el ChannelName 'Chan2. Por lo tanto, se puede acceder al segundo canal con el comando:

```
ch2 = ao.Chan2;

ch2
```

```
  Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

  2      'Chan2'      2          [-1 1]      [-1 1]       [-1 1]       'Volts'
```

Los canales que tienen el mismo objeto (o se asocian con el objeto de la misma salida analógica) se pueden concatenar en cualquiera de los vectores fila o columna. Se producirá un error si intenta concatenar canales en una matriz.

```
cha = [ch1 ch2];   % row concatenation

ch = [ch1; ch2];   % column concatenation
```

```
size(cha)
size(ch)
ans =
    1     2
ans =
    2     1
```

Se puede acceder a canales específicos, especificando el índice del canal en el objeto de entrada análoga.

```
ch1 = ao.Channel(1); % Para obtener el primer canal
ch2 = ao.Channel(2); % Para obtener el Segundo canal
ch1
ch2
```

```
  Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
  1      'Left'       1          [-1 1]      [-1 1]       [-1 1]       'Volts'
```

```
  Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
  2      'Chan2'      2          [-1 1]      [-1 1]       [-1 1]       'Volts'
```

O, si un canal matriz de 1-por-5 se crea, es posible acceder al primer, tercer y cuarto canal de la siguiente manera:

```
charray = [ch1 ch2 ch1 ch1 ch2];
temp = charray([1 3 4]);
charray
temp
```

```
  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:  Units:
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
    2     'Chan2'       2           [-1 1]       [-1 1]        [-1 1]       'Volts'
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
    2     'Chan2'       2           [-1 1]       [-1 1]        [-1 1]       'Volts'

  Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:  Units:
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
    1     'Left'        1           [-1 1]       [-1 1]        [-1 1]       'Volts'
```

Puede obtener información sobre el canal con el comando GET. Al poner el comando GET acompañado con el canal se obtiene una lista de todas las propiedades y sus valores actuales. Las propiedades comunes se muestran primero. Si existen propiedades específicas aparecerán en el segundo lugar.

```
get(ch1)
        ChannelName = Left
        HwChannel = 1
        Index = 1
        InputRange = [-1 1]
        NativeOffset = 1.5259e-005
        NativeScaling = 3.0518e-005
        Parent = [1x1 analoginput]
        SensorRange = [-1 1]
        Type = Channel
        Units = Volts
        UnitsRange = [-1 1]
```

Si a una variable es suplida con el comando GET, la variable toma la estructura del canal y todas sus propiedades.

```
h = get(ch2);
h
h =
      ChannelName: 'Chan2'
        HwChannel: 2
            Index: 2
       InputRange: [-1 1]
     NativeOffset: 1.52590218966964e-005
    NativeScaling: 3.05180437933928e-005
           Parent: [1x1 analoginput]
     SensorRange: [-1 1]
             Type: 'Channel'
            Units: 'Volts'
      UnitsRange: [-1 1]
```

Se puede obtener información relacionada con una propiedad específica, especificando el nombre de la propiedad como el argumento de la segunda entrada a GET (por ejemplo ch1).

```
get(ch1, 'Units')
ans =
Volts
```

Si la matriz de canal no es 1-por-1, una matriz de valores de la propiedad es retornada.

```
get(ch, 'Units')
ans =
    'Volts'
```

'Volts'

Usted puede obtener información de cualquiera de las propiedades especificando la propiedad, para esto se utiliza el comando GET acompañado del primer argumento que es el canal de salida y como segundo argumento la propiedad o propiedades de las que quiera tener información.

```
get(ch2, {'ChannelName','HwChannel','Units'})
ans =
    'Chan2'    [2]    'Volts'
```

Si son varios canales se puede obtener las propiedades de los dos canales con el comando matriz ch, utilizando el comando GET. El valor de las propiedades de la matriz devuelta se refieren a cada objeto y nombre de la propiedad de la siguiente manera:

```
get(ch, {'ChannelName','HwChannel','Units'})

ans =

    'Left'     [1]    'Volts'

    'Chan2'    [2]    'Volts'
```

También puede utilizar la notación de puntos para obtener información acerca de cada canal. Por ejemplo, ch1.Units, ó ao.Channel (2). ChannelName;

También puede obtener información de los canales a través de referencias con nombre. Por ejemplo: ao.Chan2.OutputRange.

Puede asignar valores a las propiedades de diferentes canales con el comando SET. Si ejecutamos el comando set y como el primer argumento el canal, una

lista de propiedades configurables y sus posibles valores son devueltos. Al igual que el GET, las propiedades comunes se muestran primero. Si cualquiera de las propiedades de dispositivos específicos del canal existen, se aparece en segundo lugar.

set(ch1)

ChannelName

DefaultChannelValue

HwChannel

OutputRange

Units

UnitsRange

Si usted proporciona una variable de salida a SET, devuelve una estructura donde los nombres son nombres de campo de la estructura del canal, propiedad y estructura de los valores de campo son los valores de las propiedades posibles para la propiedad del canal.

h = set(ch1);

h

h =

ChannelName: {}

DefaultChannelValue: {}

HwChannel: {}

OutputRange: {}

Units: {}

UnitsRange: {}

Puede obtener información sobre posibles valores de la propiedad para una propiedad específica, especificando el nombre de la propiedad como el argumento de la segunda entrada a SET. Por ejemplo, establezca (CH1, "Unidades") o unidad = set (CH1, "Unidades") para asignar el resultado a una variable de salida.

Puede asignar valores a una propiedad específica de la siguiente manera:

```
set(ch1, 'UnitsRange', [0 100]);

ch

  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:

  1       'Left'        1           [-1 1]        [0 100]      'Volts'

  2       'Chan2'       2           [-1 1]        [-1 1]       'Volts'
```

Puede asignar la misma propiedad para múltiples canales con una llamada a SET.

```
set(ch, 'UnitsRange', [-50 50]);

ch

  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:

  1       'Left'        1           [-1 1]        [-50 50]     'Volts'

  2       'Chan2'       2           [-1 1]        [-50 50]     'Volts'
```

Puede asignar diferentes valores de propiedad a múltiples canales mediante la especificación de una propiedad matriz y asignar a cada canal un valor diferente con el comando SET.

```
set(ch, {'UnitsRange'}, {[-20 20];[0 10]});
```

```
ch
```

```
  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:
  1       'Left'        1           [-1 1]        [-20 20]     'Volts'
  2       'Chan2'       2           [-1 1]        [0 10]       'Volts'
```

Esto puede hacerse extensiva a asignar propiedades de múltiples valores diferentes para los múltiples canales. Cada celda tiene un valor de la propiedad y debe ser m-por-n, donde m es el número de objetos y n es el número de unidades. Cada fila de la matriz de valores de propiedad celda contiene el valor de las propiedades de un objeto único. Cada columna de la matriz de valores de propiedad celda contiene el valor de las propiedades de una sola propiedad.

El valor de las propiedades asignadas matriz se refiere a cada objeto y nombre de la propiedad de la siguiente manera:

```
set(ao.Channel, {'Units'; 'UnitsRange'}, {'LeftUnits', [0 30]; 'RightUnits', [0 40]});
ao.Channel
```

```
  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:
  1       'Left'        1           [-1 1]        [0 30]       'LeftUnits'
  2       'Chan2'       2           [-1 1]        [0 40]       'RightUnits'
```

Puede utilizar la notación de puntos para asignar valores a las propiedades del canal.

```
ch1.Units = 'OneUnits';
ch2.UnitsRange = [0 50];
```

```
ao.Channel(1).ChannelName = 'Chan1';
ao.Channel
  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:
  1       'Chan1'       1           [-1 1]        [0 30]       'OneUnits'
  2       'Chan2'       2           [-1 1]        [0 50]       'RightUnits'
```

O bien, puede asignar valores a las propiedades a través de canal hace referencia a su nombre.

```
ao.Chan2.Units = 'TwoUnits';
ao.Channel
  Index:  ChannelName:  HwChannel:  OutputRange:  UnitsRange:  Units:
  1       'Chan1'       1           [-1 1]        [0 30]       'OneUnits'
  2       'Chan2'       2           [-1 1]        [0 50]       'TwoUnits'
```

El rango de datos de salida para su convertidor D/A y el rango valido pal convertir D/A puede ser controlado por las siguientes propiedades de ingeniería:

OutputRange - the D/A converter range.

UnitsRange - output data range.

Units - Engineering units name.

El rango de datos del hardware D/A, se seleccionará de manera que los datos de salida no se recorten, se debe utilizar los rangos posibles para la salida del hardware.

Por ejemplo, supongamos que se imprimen los datos que oscila entre -2 V y 2V, y el convertidor D / A tiene un alcance máximo de -1 a 1 voltio voltios. Si los datos no se escalan, se pisa un rango de datos pues se está excediendo el rango que puede salir por el hardware. Al establecer la propiedad UnitsRange a -2 a 2 voltios, los datos, se incrementarán de manera que un valor de datos de -2 se

asignarán a un valor de -1 a nivel de hardware y un valor de datos de 2 se asignarán a un valor de 1 a nivel de hardware y el corte no se producirá.

La configuración del canal se indica a continuación:

```
set(ao.Channel, 'OutputRange', [-1 1]);

 set(ao.Channel, 'UnitsRange', [-2 2]);
```

Sólo conversiones lineales son compatibles. Puede realizar conversiones no lineales mediante la creación de la función adecuada M-file.

5.3 EXPERIENCIA PARA LA UTILIZACION DEL TRIGGER EN LA SALIDA ANALOGA.

En esta experiencia se muestra cómo los datos pueden transmitirse mediante un trigger inmediato y manual. Las diversas propiedades del trigger se exploran en relación con cada tipo de trigger. Las propiedades de trigger que se mostraran incluyen: TriggerType y RepeatOutput.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

Un trigger para salida análoga se define como un evento que inicia la salida de datos.
Un objeto de salida análoga puede sacar datos utilizando un trigger inmediato o un trigger manual. El tipo de trigger se indica mediante la propiedad TriggerType. Un trigger inmediato es el tipo de disparo por defecto.

En esta demo, vamos a explorar las propiedades relacionadas con triggering, y los datos de salida utilizando un trigger inmediato y manual.

Un trigger inmediato se produce automáticamente justo después de ejecutar el comando START.

Empezamos creamos el objeto de salida análoga y se le asignan dos canales. Esto permitirá que el dispositivo winsound se ejecute en modo estéreo. El objeto de salida análoga se configura para que de una salida de 8000 muestras por segundo.

Nota: El objeto ao está configurado para la salida de datos de forma inmediata.

```
ao = analogoutput('winsound');

addchannel(ao, [1 2]);

set(ao, 'SampleRate', 8000);

set(ao, 'TriggerType', 'Immediate');
```

Ahora vamos a definir los datos a la salida, que no es más que una onda sinusoidal con una frecuencia de 500 Hz.

```
y = sin(500*(0:(2*pi/8000):2*pi))';
```

Outputting data implica que los datos van a quedar en cola en el motor de adquisición de datos. Con el comando PUTDATA se sacan de cola los datos del motor de adquisición de datos.

putdata(ao, data);

El dato es una matriz m-por-n donde m es el número de muestras que van a enviarse y n es el número de canales de salida.

putdata(ao, [y y]);

Tan pronto como el objeto de salida analógica se inicia, el trigger se ejecuta. Cuando se ejecuta el trigger, los datos en que están en cola en el motor de adquisición de datos saldrán a la tarjeta de sonido. 8000 muestras están siendo la producción a 8000 muestras por segundo.

Se debe tomar aproximadamente un segundo a la salida de los datos. La salida se produce en segundo plano, mientras continúa la ejecución de MATLAB ®. El comando WAIT se pone en pausa hasta que la salida de datos es completa o el tiempo de espera expira dos segundos (lo que ocurra primero). Si el tiempo de espera, un error se generará.

```
 start(ao)
tic
wait(ao,2);
toc
Elapsed time is 0.986394 seconds.
```

Usted puede configurar el objeto de salida análoga para que saque repetidamente los datos siempre con la misma cola con la propiedad RepeatOutput. Si RepeatOutput es mayor que 0, entonces todos los datos en cola antes de que el START se emita será n más uno las veces que se repita.

Vamos a configurar el objeto de salida analógica, ao, a la salida de los datos en cola tres veces.

```
set(ao, 'RepeatOutput', 2);

putdata(ao, [y y]);
```

El objeto de salida analógica se ha iniciado, luego esperamos hasta 4 segundos para que la salida se complete. Se debe tomar alrededor de tres segundos para enviar los datos.

```
start(ao)
tic
wait(ao,4);
toc
Elapsed time is 3.014145 seconds.
```

Un trigger manual inicia la salida de datos manualmente después de ejecutar el comando TRIGGER.

Vamos a configurar el objeto de salida analógica, AO, de datos de salida a 8000 muestras por segundo con un disparo manual.

```
set(ao, 'SampleRate', 8000);
set(ao, 'TriggerType', 'manual');
set(ao, 'RepeatOutput', 0);
putdata(ao, [y y]);
```

El objeto de salida analógica se inicia con el comando START.
El motor de adquisición de datos va a correr tan pronto se ejecute el comando de START.

Sin embargo, las muestras de datos no se emitirán a la tarjeta de sonido hasta que el comando TRIGGER se ejecute. Por lo tanto, el número de muestras de la salida del motor de adquisición de datos será cero.

```
start(ao);
get(ao, 'Running')
get(ao, 'SamplesOutput')
ans =
On
```

ans =

0

Para ejecutar el trigger manual y los datos se emitan a la tarjeta de sonido, se utilizaría:

```
trigger(ao);

wait(ao,2);
```

Por último, eliminar el objeto de salida analógica.

```
delete(ao)
```

6 ENTRADA Y SALIDA DIGITAL

6.1 EXPERIENCIA DE INTRODUCCION A LA ENTRADA Y SALIDA DIGITAL.

En esta experiencia se muestra un período de adquisición de datos básico utilizando un objeto de entrada y salida digital E/S. Esto se demuestra mediante la adición de líneas al objeto de E / S digital, y la escritura y lectura de datos de las líneas configuradas.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

Para empezar, cree un objeto de E / S digital que estén relacionados con la tarjeta de National Instruments.

Un objeto de entrada o salida digital (DIO) se crea con la función digitalio. Digitalio acepta el nombre del adaptador y el ID del dispositivo de hardware como argumentos de entrada. Para tarjetas de adquisición de datos, el ID del dispositivo se refiere al número asociado con la tarjeta cuando ésta es instalada. Con las tarjetas NI-DAQmx, usualmente es una cadena de caracteres como ‘Dev1’.

Cada objeto de entrada/salida digital es asociado con un puerto paralelo o un subsistema. Por ejemplo, para crear un objeto de entrada/salida digital asociado con una tarjeta de National Instruments:

Cuatro líneas de salida se agregan al objeto de E / S digital desde el puerto 0. Esta configuración le permite escribir valores en estas líneas.

```
dio = digitalio('nidaq', 'Dev1');
addline(dio, 0:3, 0, 'Out')
  Index:  LineName:  HwLine:  Port:  Direction:
  1       ''         0        0      'Out'
  2       ''         1        0      'Out'
  3       ''         2        0      'Out'
  4       ''         3        0      'Out'
```

Puede escribir valores para las líneas de E/S digital como un valor decimal o como un valor binvec. Un valor binvec es un vector binario, lo que está escrito con el bit menos significativo es el elemento vector de la izquierda y el bit más significativo es el elemento más a la derecha del vector. Por ejemplo, el valor decimal 23 en notación se escribe como binvec

[1 1 1 0 1]

El valor binvec se convierte en un valor decimal de la siguiente manera,

1*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 1*2^4

La toolbox le proporciona dos funciones convenientes para la conversión entre valores decimales y valores binvec.

Usted puede convertir un valor decimal en un valor binvec con la función DEC2BINVEC.

```
binvec = dec2binvec(196)

binvec =
    0     0     1     0     0     0     1     1
```

Usted puede convertir un valor binvec a un valor decimal con la función BINVEC2DEC

```
decimal = binvec2dec([1 0 1 0 1 1])
decimal =
   53
```

Se puede utilizar la función PUTVALUE para escribir valores a las líneas de los objetos de E/S digital. Solo se puede escribir en una línea configurada que sea para salida 'Out'. Se producirá un error si se escribe a una línea configurada que sea para entrada 'in'.

Por ejemplo, para escribir el valor binvec [0 1 0 0] al objeto dio de E / S digital

```
putvalue(dio, [0 1 0 0]);
```

Del mismo modo, puede escribir a un subconjunto de las líneas de la siguiente manera.

```
putvalue(dio.Line([ 1 3 ]), [ 1 1 ]);
```

Las líneas no escritas que permanecen en sus valores actuales (en este ejemplo, las líneas con un índice de 2 y 4 permanecen en el índice [1 0], respectivamente). Es importante señalar que si un valor decimal se escribe en un objeto de E/S digital, y el valor es demasiado grande para ser representado por el objeto, se

devuelve un error. De lo contrario, la palabra se rellena con ceros.

En este ejemplo, el objeto de E / S digital consta de cuatro líneas. Por lo tanto, el mayor valor decimal permitido es de 15 o

(1*2^0 + 1*2^1 + 1*2^2 + 1*2^3).

Si el valor decimal 5 se escribe en el objeto de a E / S digital, sólo un vector de 3 elementos binvec es necesario para representar el valor decimal. Las líneas restantes tendrán un valor de 0 por defecto. Por lo tanto, los dos comandos siguientes son equivalentes.

```
putvalue(dio, 5);
putvalue(dio, [1 0 1 0]);
```

Se pueden ver los valores de las líneas digitales E/S con la función GetValue. GetValue siempre devuelve un valor binvec. Usted puede leer desde una línea configurada tanto para entrada o salida 'In' o 'Out'.

```
value = getvalue(dio)
value =
    1     0     1     0
```

Usted puede convertir el valor binvec a un valor decimal con la función BINVEC2DEC.

```
value = binvec2dec(getvalue(dio))
value =
    5
```

Del mismo modo, se puede leer un subconjunto de las líneas de la siguiente manera.

```
getvalue(dio.Line([ 1 3 ]))
ans =
    1     1
```

Cuando haya terminado, usted debería limpiar los objetos de datos innecesarios de la adquisición. Elimine el objeto de E / S digital y limpie el área de trabajo.

```
delete (dio);
clear dio;
```

6.2 EXPERIENCIA PARA LA OBTENCION DE ACCESO A LINEAS DIGITALES.

En esta experiencia se tratara sobre el uso de get / set de notación, la notación de puntos, y la notación de índice para la obtención de información sobre la línea y para configurar la línea para su adquisición. También se crea, accede, y configura líneas de E/S digital utilizando la tarjeta NI USB 6008.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
   stop(daqfind)
end
```

Para empezar, creamos el objeto de E / S digital dio, asociado con el hardware de National Instrument ID..

```
dio = digitalio('nidaq', 'Dev1');
```

Se agrega la línea al objeto de E/S digital con el comando ADDLINE. ADDLINE necesita por lo menos tres argumentos para la creación del canal de entrada. El primer argumento especifica el objeto dio de la E / S digital, el segundo argumento es el identificador de la línea de hardware que se añade al objeto de E / S digital, el tercer argumento de entrada especifica la dirección de la línea. La dirección puede ser 'In' en cuyo caso se lee la línea de origen, o 'Out', en cuyo caso la línea se escribe.

```
addline(dio, 0, 'In');
```

Hay tres métodos para acceder a una línea. En el primer método, la línea se accede a través de las propiedades de las líneas de E/S digital. Por ejemplo, usando la notación de puntos, las líneas del objeto de todas las entradas digitales se asignaran a la variable línea 1.

```
line1 = dio.Line
  Index:  LineName:  HwLine:  Port:  Direction:
  1       ''         0        0      'In'
```

O, usando la notación GET

```
line1 = get(dio, 'Line')
  Index:  LineName:  HwLine:  Port:  Direction:
  1       ''         0        0      'In'
```

En el segundo método, la línea se accede en tiempo de creación mediante la asignación de una variable de salida a ADDLINE.

El cuarto argumento de entrada para ADDLINE asigna el nombre Line2 para la línea creada.

```
line2 = addline(dio, 1, 'In', 'Line2')
  Index:  LineName:  HwLine:  Port:  Direction:
```

```
  2      'Line2'   1       0      'In'
```

En el tercer método, la línea se accede a través de su valor de la propiedad LineName. Este método se refiere a la referencia que se cita. La segunda línea añade al objeto de E / S digital el LineName "Línea 2". Por lo tanto, la segunda línea se puede acceder con el comando

```
line2 = dio.Line2
  Index:  LineName:  HwLine:  Port:  Direction:
  2      'Line2'   1       0      'In'
```

Las líneas que tienen el mismo padre (o están asociados con el mismo objeto de E/S digital) se pueden concatenar en cualquiera de los vectores fila o columna. Se producirá un error si intenta concatenar líneas en una matriz.

```
daqline = [line1 line2];
size(daqline)
ans =
    1     2
```

ó

```
daqline = [line1;line2];
size(daqline)
ans =
    2     1
```

Se puede acceder a una línea específica por especificación del índice de la línea en el objeto de E/S digital. Para la obtención de la primera línea:

```
line1 = dio.Line(1)
  Index:  LineName:  HwLine:  Port:  Direction:
  1      ''        0       0      'In'
```

Para obtener la segunda línea

```
line2 = dio.Line(2)
  Index:  LineName:  HwLine:  Port:  Direction:
  2       'Line2'    1        0      'In'
```

O, si la matriz de la línea se crea de 1-por-5, es posible acceder a la primera, tercera y cuarta líneas de la siguiente manera

```
linearray = [line1 line2 line1 line1 line2]
  Index:  LineName:  HwLine:  Port:  Direction:
  1       ''         0        0      'In'
  2       'Line2'    1        0      'In'
  1       ''         0        0      'In'
  1       ''         0        0      'In'
  2       'Line2'    1        0      'In'
```

```
temp = linearray([1 3 4])
  Index:  LineName:  HwLine:  Port:  Direction:
  1       ''         0        0      'In'
  1       ''         0        0      'In'
  1       ''         0        0      'In'
```

Se puede obtener información sobre la línea con el comando GET. Al invocar la línea con el comando GET. Donde una lista de todas las propiedades y sus valores actuales se muestran. Las propiedades comunes se muestran primero. Si cualquiera de las propiedades específicas del dispositivo de línea existe, aparecerán en segundo lugar.

```
get(line1)
        Direction = In
        HwLine = 0
        Index = 1
        LineName =
```

```
    Parent = [1x1 digitalio]
    Port = 0
    Type = Line
```

Se le puede asignar a una línea cualquier nombre con comando GET, la estructura de la línea retorna cuando llamamos a esa variable, los valores de la propiedad de la línea son los mismos.

```
h = get(line2)
h =
  Direction: 'In'
     HwLine: 1
      Index: 2
   LineName: 'Line2'
     Parent: [1x1 digitalio]
       Port: 0
       Type: 'Line'
```

Se puede obtener información relacionada con una propiedad específica, especificando el nombre de la propiedad como el segundo argumento del comando GET.

```
get(line1, 'Direction')
ans =
In
```

Si la matriz de la línea no es 1-por-1, un conjunto de valores de propiedades matriz retornan.

```
get(daqline, 'Direction')
ans =
  'In'
  'In'
```

Se puede obtener información relacionada con múltiples propiedades mediante la especificación de un conjunto de nombres de propiedades como el argumento de la segunda entrada a GET.

```
get(line2, {'LineName','HwLine','Direction'})
ans =
    'Line2'    [1]    'In'
```

Si son más de dos líneas matriz se puede visualiza las propiedades de las líneas con el comando GET.

```
get(daqline, {'LineName','HwLine','Direction'})
ans =
        ''    [0]    'In'
    'Line2'    [1]    'In'
```

El valor de las propiedades de la matriz devuelta se refiere a cada objeto y nombre de la propiedad de la siguiente manera

Puede utilizar la notación de puntos para obtener información de cada línea.

```
line1.Direction
ans =
In
dio.Line(2).LineName
ans =
Line2
```

También puede obtener información de la línea a través de referencias con nombres.

```
dio.Line2.Direction
ans =
In
```

Puede asignar valores a las propiedades de línea diferente con el comando SET. Si sólo la línea se coloca como primer argumento acompañada del comando SET,

una lista de propiedades configurables y sus posibles valores son devueltos. Al igual que el GET pantalla, las propiedades comunes se muestran primero. Si cualquiera de las propiedades específicas de la línea del dispositivo existe, aparecerá en segundo lugar.

```
set(line1)
        Direction: [ {In} | Out ]
        HwLine
        LineName
```

Si usted proporciona una variable de salida para SET, una estructura es retornada donde la estructura encuentra nombres de las propiedades de la línea y la estructura encurta valores que son los valores de propiedades posibles para las propiedades de la línea.

```
h=set(line1)
h =
    Direction: {2x1 cell}
       HwLine: {}
     LineName: {}
```

Puede obtener información sobre posibles valores de la propiedad para una propiedad específica, especificando el nombre de la propiedad como el argumento de la segunda entrada a SET.

```
set(line1, 'Direction')
[ {In} | Out ]
```

O bien, puede asignar el resultado a una variable de salida.

```
direction = set(line1, 'Direction')
direction =
    'In'
    'Out'
```

Puede asignar valores a una propiedad específica de la siguiente manera

```
set(line1, 'LineName', 'Line1');
daqline
  Index:  LineName:  HwLine:  Port:  Direction:
  1       'Line1'    0        0      'In'
  2       'Line2'    1        0      'In'
```

Puede asignar diferentes valores a la misma propiedad para múltiples líneas con una llamada del comando SET. Tenga en cuenta que el dispositivo es el puerto configurable. La línea existente se configura como 'In' y la línea que se añade es configurada como 'Out'. Como resultado, el cuadro de herramientas cambiará todas las líneas a la nueva dirección y emitirá una advertencia.

```
set(daqline, 'Direction', 'Out');
daqline
Advertencia: El puerto no es la línea configurable. Todas las direcciones de línea en el puerto se han establecido

Index:  LineName:  HwLine:  Port:  Direction:
  1       'Line1'    0        0      'Out'
  2       'Line2'    1        0      'Out'
```

Puede asignar diferentes valores de propiedad a varias líneas, indicando una serie de valores de propiedad en el comando SET.

```
set(daqline, {'Direction'}, {'Out'; 'Out'})
daqline
  Index:  LineName:  HwLine:  Port:  Direction:
  1       'Line1'    0        0      'Out'
  2       'Line2'    1        0      'Out'
```

Se puede hacer para múltiples propiedades de las líneas e incluso colocar diferente valor para cada propiedad y para cada línea. La matriz de valores de la

propiedad debe ser m-por-n, donde m es el número de objetos y n es el número de unidades. Cada fila de la matriz de valores de propiedad contiene el valor de la propiedad de un objeto único. Cada columna de la matriz de valores de propiedad contiene el valor de las propiedades de una sola propiedad.

```
set(dio.Line, {'Direction'; 'LineName'},...
  {'Out', 'DIOLine1'; 'Out', 'DIOLine2'});
dio.Line
  Index:  LineName:   HwLine:  Port:  Direction:
  1       'DIOLine1'  0        0      'Out'
  2       'DIOLine2'  1        0      'Out'
```

La asignación de valores a las propiedades matriz relata para cada objeto y nombre de propiedad como sigue:

```
            Direction       LineName
line1           'Out'         'DIOLine1'
line2           'Out'         'DIOLine2'
```

Puede utilizar la notación de puntos para asignar valores a propiedades de la línea.

```
line1.Direction = 'Out';
line2.LineName = 'MyLine2';
dio.Line(1).LineName = 'MyLine1';
dio.Line
  Index:  LineName:  HwLine:  Port:  Direction:
  1       'MyLine1'  0        0      'Out'
  2       'MyLine2'  1        0      'Out'
```

O bien, puede asignar valores a las propiedades de la línea a través de referencias con nombre.

```
dio.MyLine2.LineName = 'Digital';
```

dio.Line

Index:	LineName:	HwLine:	Port:	Direction:
1	'MyLine1'	0	0	'Out'
2	'Digital'	1	0	'Out'

Esto completa la introducción de líneas de E/S digital. Se debe eliminar el objeto de E / S digita, dio, con el comando DELETE, para liberar memoria y otros recursos físicos.

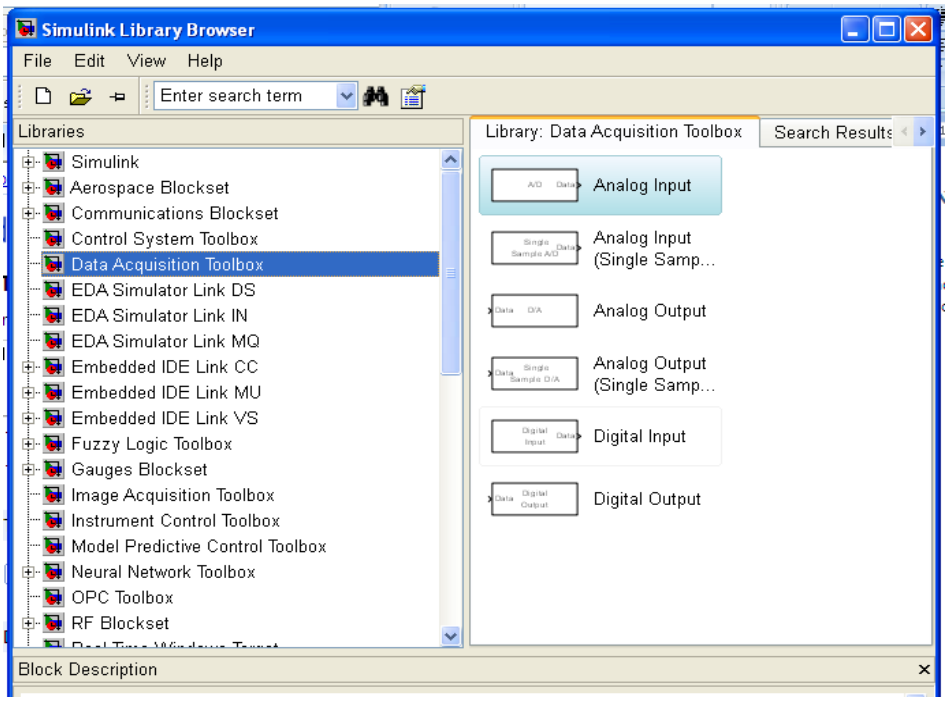
```
delete(dio);
```

7 USO DE LOS BLOQUES DE SIMULINK EN LA ADQUISICION DE DATOS

En la librería de simulink se encuentran una serie de bloques que sirven para la adquisición de datos. En este capítulo se harán experiencias explicando cómo utilizar algunos de estos bloques y sus características.

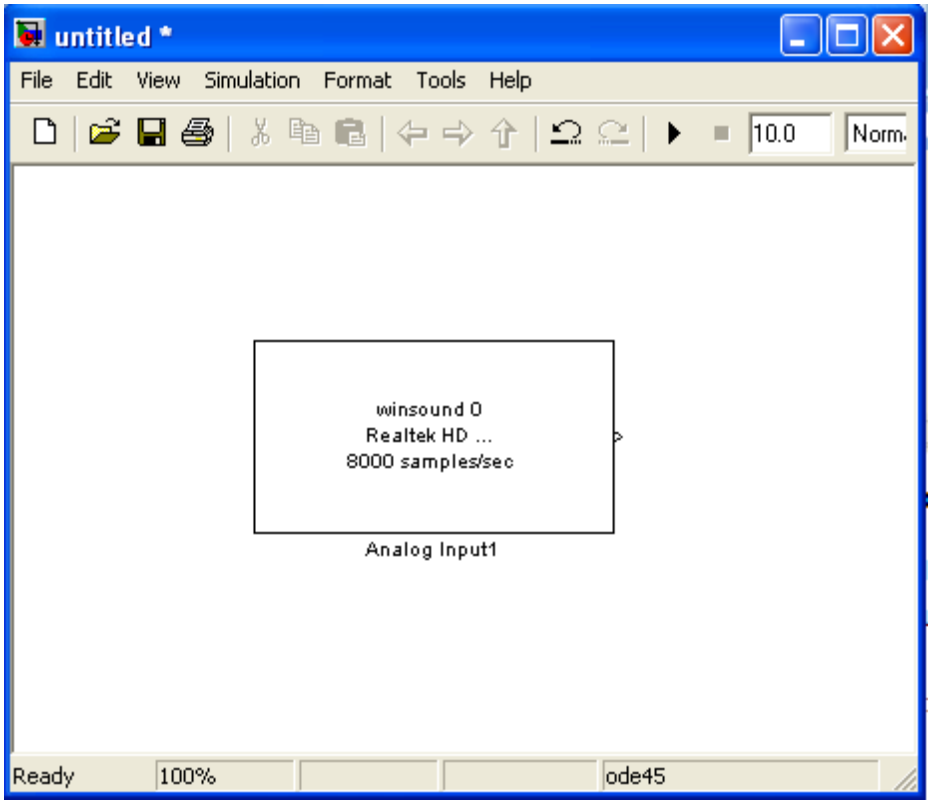
7.1 EXPERIENCIA DEL BLOQUE DE ENTRADA ANALOGA

Para usar el bloque de entrada análoga, se debe primero crear el nuevo modelo, luego dentro de las bibliotecas de simulink buscamos la librería de la Toolbox de adquisición de datos. En esta encontramos varios bloques para la adquisición de datos le damos clic en el bloque de Entrada Análoga.



Grafica 30. Biblioteca de Adquisición de Datos en Simulik

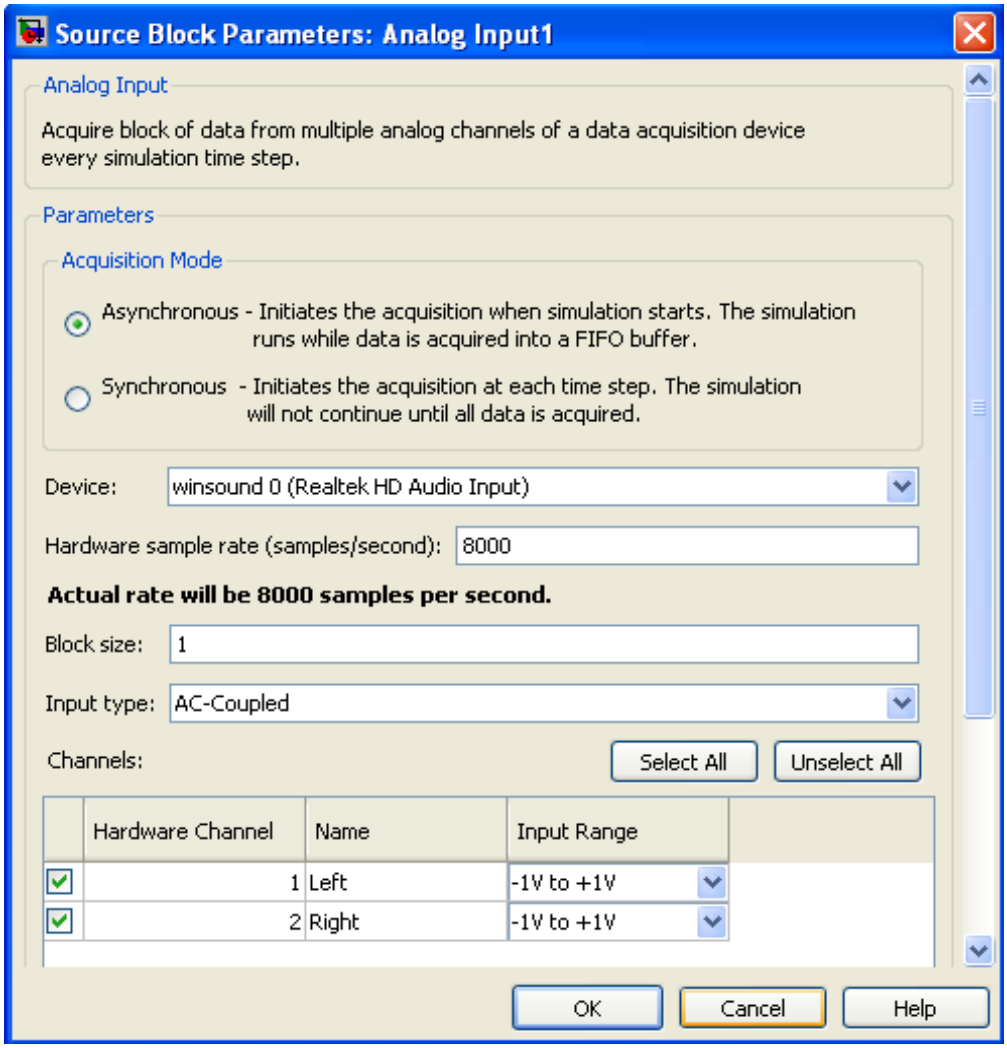
En el siguiente paso adicionamos el bloque de entrada análoga al nuevo modelo, nótese que el nombre del bloque cambia dependiendo el hardware que está instalado en su equipo, en este caso solo se tiene el dispositivo de sonido, ver grafica siguiente.



Grafica 31. Bloque de entrada análoga

Para esta experiencia necesitamos adicionar el Scope al modelo pues la intención es crear un modelo simple para adquirir datos desde un micrófono, desde una tarjeta de sonido (Dispositivo análogo). Visualizamos la intensidad de las ondas de sonido.

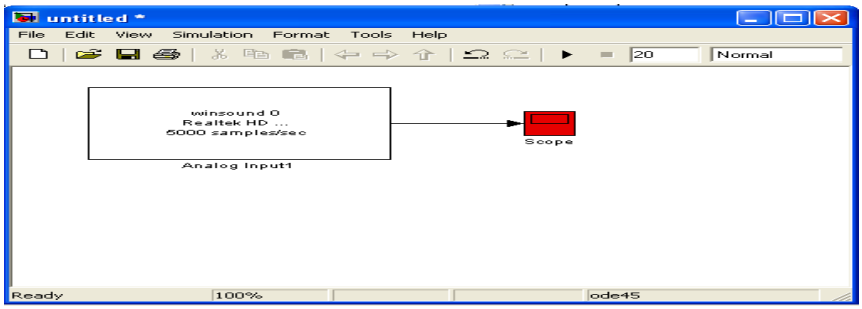
Para especificar los parámetros del bloque de entrada análoga, damos doble clic sobre este, aparecerá una ventana llamada “Source Block Parameters dialog box”, Utilice los distintos ámbitos para determinar los valores actuales de los parámetros del bloque de entrada analógica o para cambiar los valores.



Grafica 32. Parámetros del bloque de entrada análoga.

En este ejemplo, se mantiene la configuración que aparece por defecto, excepto en el tamaño del bloque. Cambiamos el tamaño del bloque a 5, lo que representa que cinco muestras serán aportadas por cada canal en el tiempo seleccionado. Como se puede ver en el cuadro de dialogo, la adquisición será asíncrona, utilizando los dos canales el de la derecha y el de la izquierda.

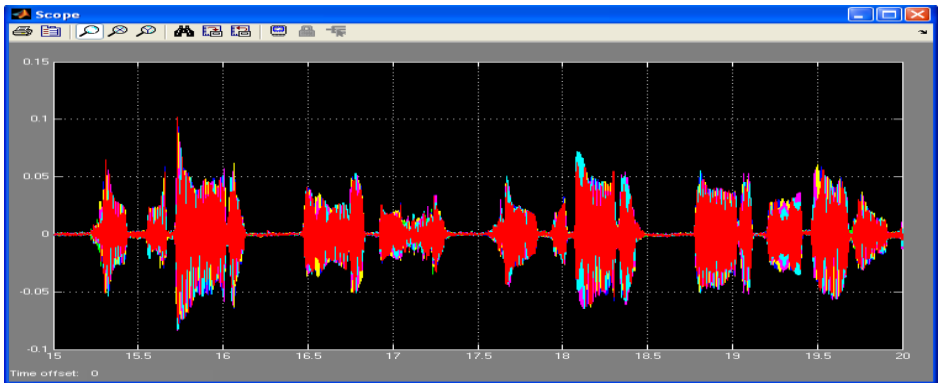
Corra la simulación, cambie el tiempo de ejecución a 20 segundos, abra el scope dándole doble clic sobre el icono, se pude ver en tiempo real las ondas sonoras cuando el modelo está corriendo.



Grafica 33. Ejecutando la simulación.

Mientras la simulación está corriendo, la barra de status del modelo indica el progreso de la simulación. En el momento de ejecución debe empezar a hablar por el micrófono para que se visualicen las ondas de sonido en el scope.

Cuando los 20 segundos terminan se para automáticamente la toma de muestras, las muestras tomadas deben de arrojarle una grafica parecida a esta:



Grafica 34. Toma de muestras de sonido.

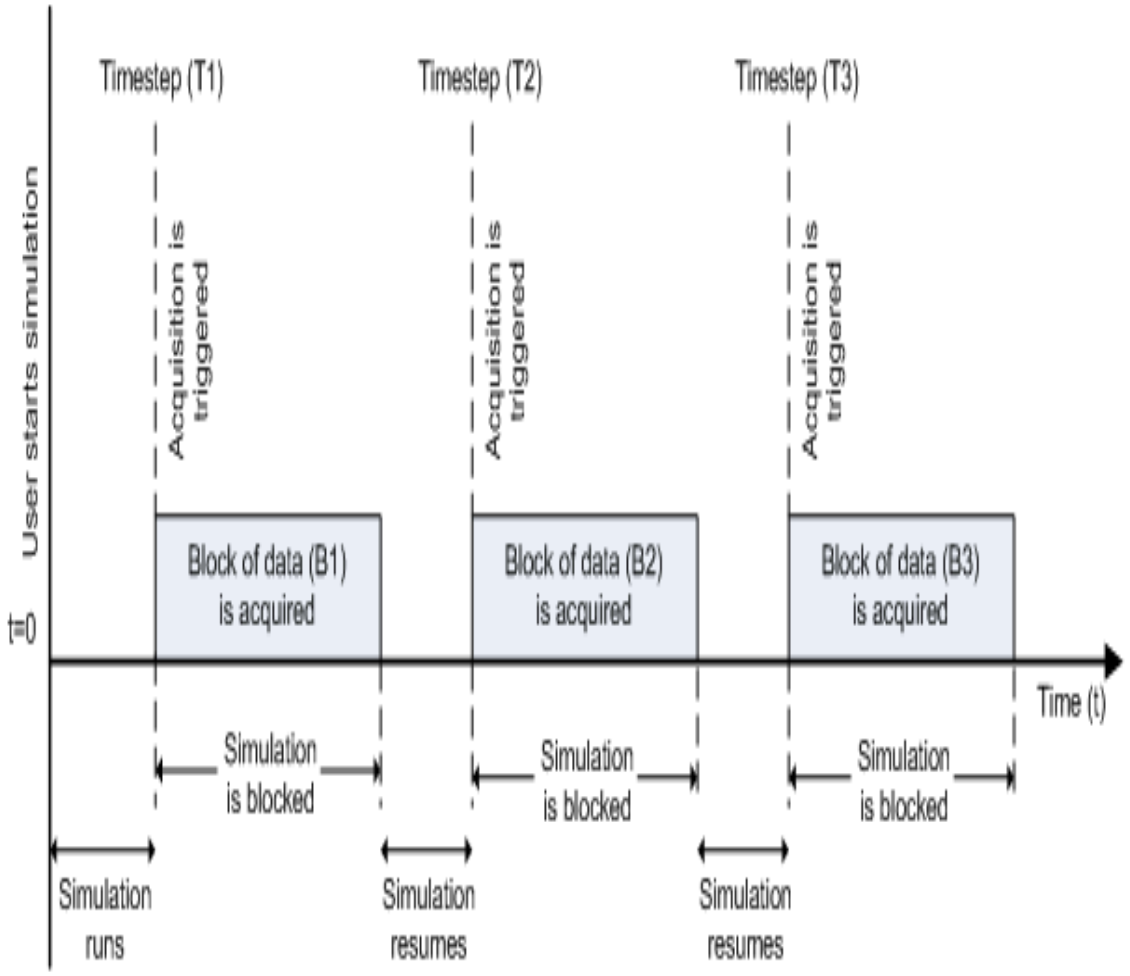
Los modos de adquisición de datos afectarían la experiencia, incluyo información de este tema para que quede más claro.

Se encuentran dos modos de adquisición de datos:

- Modo Asíncrono: Este modo inicia la adquisición cuando la simulación empieza. Cuando esta corre los datos pasan a una memoria tipo FIFO. La adquisición es continua y depende de la cantidad de muestras que se tomen.

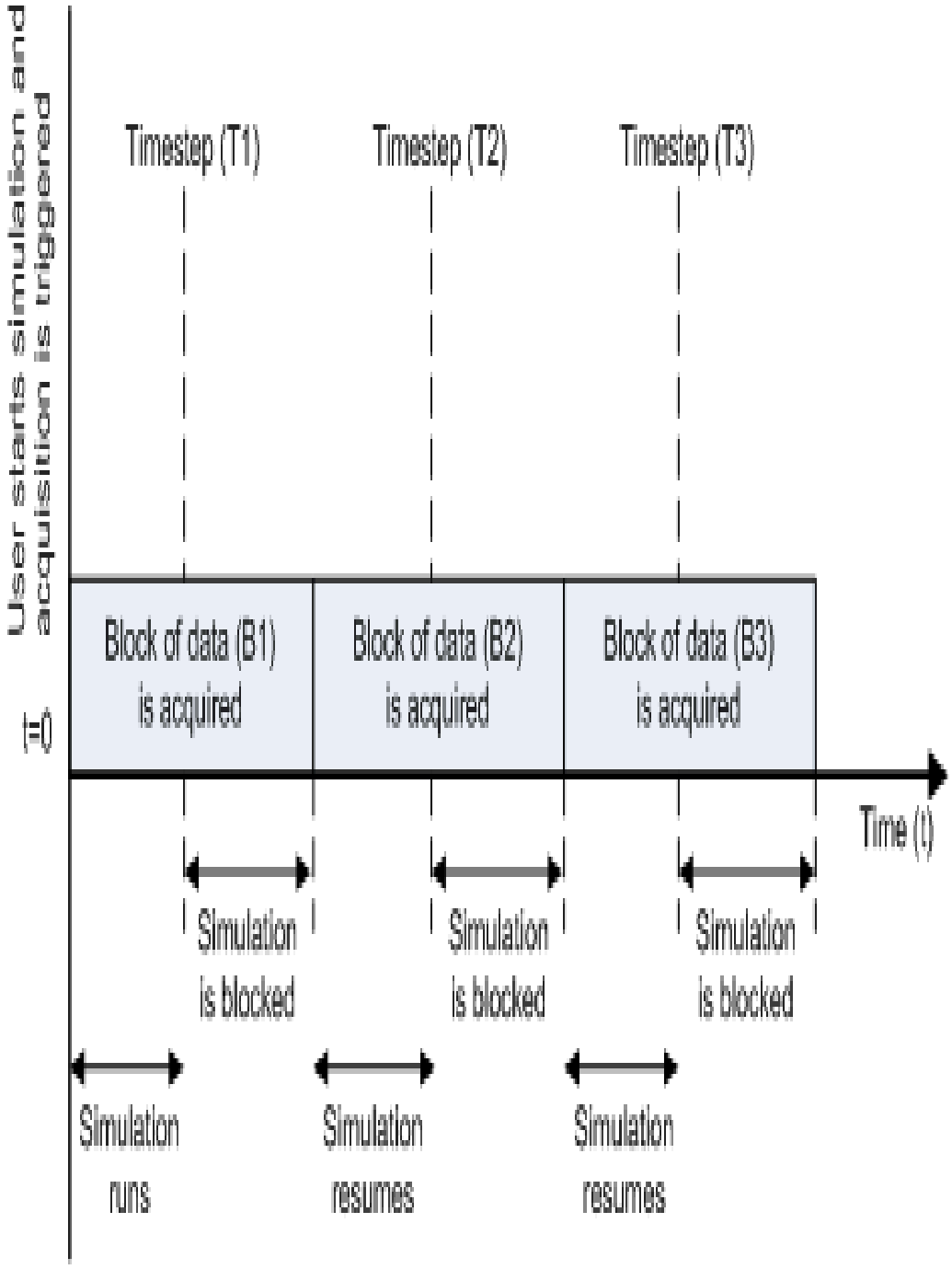
- Modo Síncrono: Inicia la adquisición para cada paso de tiempo. La simulación no continúa hasta que el bloque de datos solicitado se adquiere. Esto es sin buffer de entrada.

El siguiente diagrama muestra la diferencia en tres el modo síncrono y asíncrono para el bloque de entrada análoga.



Grafica 35. Modo síncrono para la adquisición de datos análoga.

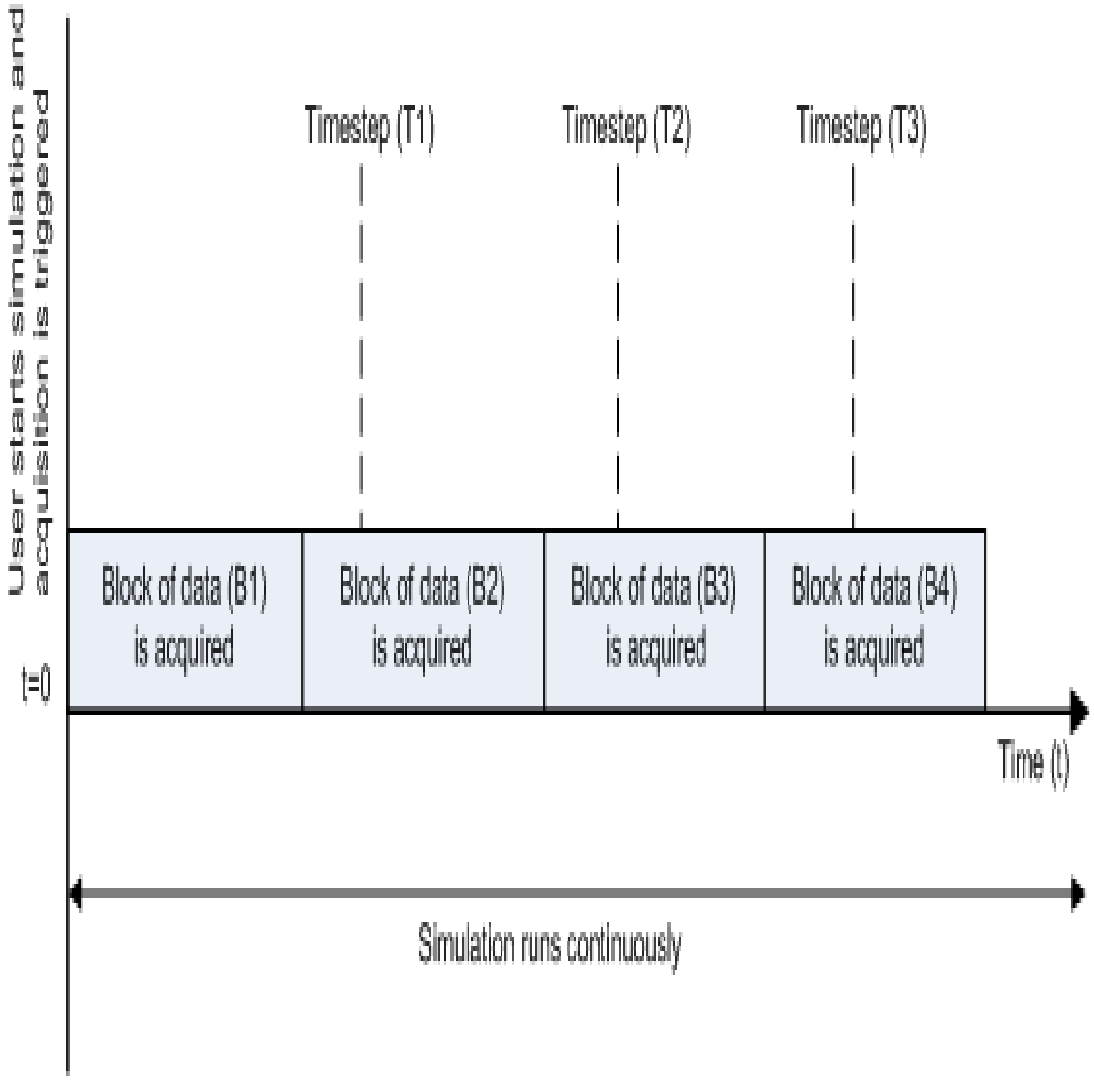
Para el primer periodo T1, la adquisición es iniciada por el bloque de datos requerido B1. La simulación no continua mientras B1 no es adquirida completamente.



Grafica 36. Modo asíncrono para la adquisición de datos análoga.

En esta grafica se observa el caso cuando la velocidad de simulación supera la velocidad de adquisición de datos. Para el primer periodo T1, el bloque requerido

de datos B1 empieza la adquisición. Por lo tanto, la simulación no continúa hasta que los datos del bloque B1 no se adquieren completamente.



Grafica 37. Modo asíncrono para la adquisición de datos análoga caso 2.

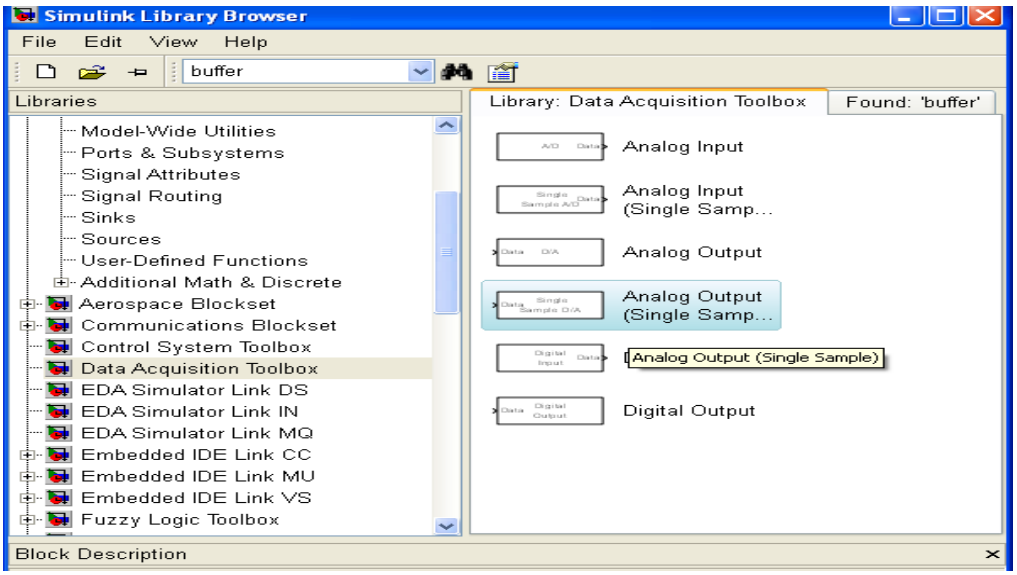
En la grafica se observa el caso cuando la velocidad de adquisición supera la velocidad de simulación. Para el primer periodo T1, los datos del bloque B1 se adquieren completamente. Por lo tanto, la simulación corre continuamente.

Varios factores, incluyendo el hardware del dispositivo y la complejidad del modelo, puede afectar a la velocidad de simulación, haciendo que tanto los escenarios 1 y 2 que se produzca en la misma simulación.

7.2 EXPERIENCIA DEL BLOQUE DE SALIDA ANALOGA

En esta experiencia se trabajara con la tarjeta de National instruments 6008 y con el bloque de salida análoga (señal Simple). Este bloque genera de a una sola muestra, sincrónicamente con el hardware, durante el tiempo de ejecución del modelo.

Para empezar se crea el nuevo modelo en simulink, luego vamos a las bibliotecas y escogemos la librería de la Toolbox de adquisición de datos. Para esta experiencia escogemos el bloque de Salida Análoga (Simple).



Grafica 38. Librería de la toolbox de adquisición de datos bloque de salida análoga.

Al bloque se le pueden agregar los puertos que se necesiten para la salida, dependiendo del tipo de hardware que se tenga, para esta experiencia trabajaremos con un solo puerto de salida.

Cabe resaltar que el bloque de salida análoga Simple soporta el uso del modo de acelerador de simulink. Para usar este modo es necesario tener los compiladores de C++.

El bloque de salida análoga simple soporta el uso de modelos de referencia. Esta característica le permite incluir dentro del modelo de simulink otro modelo de simulink como componente modular.

Completar el diseño del modelo con la señal a sacar, seguido de un retenedor de orden cero que determina el periodo de muestras y el scope.

Con doble clic sobre el bloque de salida análoga simple, se despliega un pantallazo con los parámetros del bloque, vamos a device y escogemos el hardware de la tarjeta de National Instrument 6008 (Nidaq Dev1 6008), seleccionamos el canal 0, marcando la casilla izquierda, y quitamos la marcación del canal 1. En número de puertos seleccionamos un solo canal para todo el hardware.

PARTE III
EXPERIENCIAS DE AYUDA DE LA TOOLBOX DE ADQUISICION DE DATOS

En esta tercera parte del libro, se encuentran herramientas que sirven para un manejo más avanzado del uso de la toolbox de adquisición de datos, como es la sincronización de la entrada y la salida análoga con diferentes métodos, la modificación y establecimiento de la velocidad de adquisición de datos, comando de ayuda que le permiten al estudiante conocer que hardware y adaptadores están instalados en la maquina además le muestra comandos que le permiten conocer más afondo algunas propiedades y funciones, medición de ruido interno y guardar el objeto en el disco duro del PC.

8 EXPERIENCIA PARA EL ACCESO A LA TARJETA DE NATIONAL INSTRUMENT.

La toolbox de adquisición de datos, soporta entradas y salidas análogas y entradas y salidas digitales usando las tarjetas de la National Instruments. En esta experiencia, nos podremos comunicar con dispositivos de la National Instrumets de la Serie M de adquisición de datos, así como otros dispositivos de la National Instrument que utilizan la interfaz del software NI-DAQmx.

Contenido

* Determinar el hardware de la National Instruments que está instalado
* Configurar una Adquisición
* Descubrir y configurar las propiedades de la Adquisición
* Establecer la tasa de adquisición y número de muestras a adquirir
* Obtener los datos
* Limpiar

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

8.1 DETERMINAR EL TIPO DE HARDWARE DE LA NATIONAL INSTRUMENT QUE SE ENCUENTRA INSTALADO.

En este ejemplo, se adquieren datos desde el hardware de la National Instruments instalado en su ordenador. Para ello, debe tener instalado NI-DAQmx versión 7.5 en el equipo, y verificar que el hardware funciona correctamente.

Visualice el hardware instalado accediendo a la interface NI-DAQ.

```
hw = daqhwinfo('nidaq')
hw =

          AdaptorDllName: [1x63 char]
       AdaptorDllVersion: '2.8 (R14SP3+)'
             AdaptorName: 'nidaq'
              BoardNames: {1x4 cell}
       InstalledBoardIds: {'Dev4'  'Dev5'  'Dev6'  '1'}
    ObjectConstructorName: {4x3 cell}
```

Daqhwinfo devuelve información acerca del hardware que está instalad y los Id que la Nationals tienen asignados para estos dispositivos. Para los dispositivos que utilizan la NI-DAQmx, estos Id son iguales a 'Dev6', que son los que normalmente aparecen en la interface NI-DAQ.

Lista de los Id que se instalan:

```
hw.InstalledBoardIds
hw.BoardNames
ans =

    'Dev4'    'Dev5'    'Dev6'    '1'
ans =

    'PCI-4472'    'PCI-6514'    'PCI-6221'    'PCI-4472'
```

Observe que la tarjeta 'PCI-4472' aparece en la lista dos veces. Como todos los de la serie E y los dispositivos de la serie S, este dispositivo está disponible tanto a través de la nueva interfaz NI-DAQmx y la interfaz tradicional de NI-DAQ.

8.2 CONFIGURACION DE UNA ADQUISICION DE DATOS.

Crear un objeto de entrada analógica asociado con uno de los dispositivos encontrados. Este objeto requiere la configuración del hardware. El ID que suple es el mismo que se muestra en la instalación de la tarjeta y el que se puede visualizar con el comando daqhwinfo.

```
% Crear un objeto de entrada analógica

ai = analoginput('nidaq','Dev6');
```

addchannel Se utiliza para especificar qué canales son los que se van a utilizar en una adquisición.

```
% Los datos se obtendrán de los canales de hardware 1 y 3
addchannel(ai, 1);
addchannel(ai, 3);
```

Escribiendo el nombre de la variable objeto se puede validar la configuración básica de la adquisición.

```
Ai
Display Summary of Analog Input (AI) Object Using 'PCI-6221'.


 Acquisition Parameters:  1000 samples per second on each channel.

                          1000 samples per trigger on each channel.
                          1 sec. of data to be logged upon START.
                          Log data to 'Memory' on trigger.

    Trigger Parameters:  1 'Immediate' trigger(s) on START.

       Engine status:  Waiting for START.

                          0 samples acquired since starting.
                          0 samples available for GETDATA.

AI object contains channel(s):
```

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'
1 '' 3 [-10 10] [-10 10] [-10 10] 'Volts'

8.3 CONFIGURACION DE LAS PROPIEDADES DE LA ADQUISICION.

Para determinar la lista de propiedades configurables, utilice la función SET. Consulte la documentación de la toolbox de adquisición de datos y así obtener la información completa acerca de las diferentes propiedades configurables.

set(ai)

BufferingConfig

BufferingMode: [{Auto} | Manual]

Channel

ChannelSkew

ChannelSkewMode: [{Minimum} | Equisample | Manual]

ClockSource: [{Internal} | ExternalScanCtrl | ExternalSampleCtrl | ExternalSampleAndScanCtrl]

DataMissedFcn: string -or- function handle -or- cell array

InputOverRangeFcn: string -or- function handle -or- cell array

InputType: [NonReferencedSingleEnded | SingleEnded | {Differential}]

LogFileName

LoggingMode: [Disk | {Memory} | Disk&Memory]

LogToDiskMode: [{Overwrite} | Index]

ManualTriggerHwOn: [{Start} | Trigger]

Name

RuntimeErrorFcn: string -or- function handle -or- cell array

SampleRate

SamplesAcquiredFcn: string -or- function handle -or- cell array

SamplesAcquiredFcnCount

SamplesPerTrigger

StartFcn: string -or- function handle -or- cell array

StopFcn: string -or- function handle -or- cell array

Tag

Timeout

TimerFcn: string -or- function handle -or- cell array

TimerPeriod

TriggerChannel

TriggerCondition: [{None}]

TriggerConditionValue

TriggerDelay

TriggerDelayUnits: [{Seconds} | Samples]

TriggerFcn: string -or- function handle -or- cell array

TriggerRepeat

TriggerType: [Manual | {Immediate} | Software | HwDigital]

UserData

NIDAQMX specific properties:

DriveAISenseToGround: [{Off} | On]

ExternalSampleClockSource: [PFI0 | PFI1 | {PFI2} | PFI3 | PFI4 | PFI5 | PFI6 | PFI7 | PFI8 | PFI9 | RTSI0 | RTSI1 | RTSI2 | RTSI3 | RTSI4 | RTSI5 | RTSI6]

ExternalScanClockSource: [PFI0 | PFI1 | PFI2 | PFI3 | PFI4 | PFI5 | PFI6 | {PFI7} | PFI8 | PFI9 | RTSI0 | RTSI1 | RTSI2 | RTSI3 | RTSI4 | RTSI5 | RTSI6]

HwDigitalTriggerSource: [{PFI0} | PFI1 | PFI2 | PFI3 | PFI4 | PFI5 | PFI6 | PFI7 | PFI8 | PFI9 | RTSI0 | RTSI1 | RTSI2 | RTSI3 | RTSI4 | RTSI5 | RTSI6]

NumMuxBoards

TransferMode: [{DualDMA} | SingleDMA | Interrupts]

Para obtener una lista completa de todas las propiedades de los objetos, con su configuración actual, utilice la función get.

```
get(ai)
        BufferingConfig = [64 30]
        BufferingMode = Auto
        Channel = [2x1 aichannel]
        ChannelSkew = 4e-006
        ChannelSkewMode = Minimum
        ClockSource = Internal
        DataMissedFcn = @daqcallback
        EventLog = [1x0 struct]
        InitialTriggerTime = [0 0 0 0 0 0]
        InputOverRangeFcn = []
        InputType = Differential
        LogFileName = logfile.daq
        Logging = Off
        LoggingMode = Memory
        LogToDiskMode = Overwrite
        ManualTriggerHwOn = Start
        Name = nidaqmxDev6-AI
        Running = Off
        RuntimeErrorFcn = @daqcallback
        SampleRate = 1000
        SamplesAcquired = 0
        SamplesAcquiredFcn = []
        SamplesAcquiredFcnCount = 1024
        SamplesAvailable = 0
        SamplesPerTrigger = 1000
        StartFcn = []
        StopFcn = []
        Tag =
```

Timeout = 1
TimerFcn = []
TimerPeriod = 0.1
TriggerChannel = [1x0 aichannel]
TriggerCondition = None
TriggerConditionValue = 0
TriggerDelay = 0
TriggerDelayUnits = Seconds
TriggerFcn = []
TriggerRepeat = 0
TriggersExecuted = 0
TriggerType = Immediate
Type = Analog Input
UserData = []

NIDAQMX specific properties:
DriveAISenseToGround = Off
ExternalSampleClockSource = PFI2
ExternalScanClockSource = PFI7
HwDigitalTriggerSource = PFI0
NumMuxBoards = 0
TransferMode = DualDMA

Los dispositivos de la National Instrument permiten la configuración de las líneas de la entrada análoga como simple o diferencial. Para obtener más información sobre tipos de entrada, consulte la documentación del hardware. Para descubrir las opciones disponibles para una propiedad específica, utilice la función SET con el nombre de la propiedad:

set(ai,'InputType')
[NonReferencedSingleEnded | SingleEnded | {Differential}]

Configurar% la entrada analógica de modo diferencial
set(ai,'InputType','Differential');

Para obtener más información sobre una propiedad específica, utilice daqhelp

8.4 ESTABLECER LA VELOCIDAD DE ADQUISICION Y EL NÚMERO DE MUESTRAS A ADQUIRIR.

Configure el objeto de entrada analógica para adquirir más muestras a una rata mayor que el valor predeterminado.

* Los datos serán adquiridos a un ritmo de 10.000 muestras por segundo
* 10.000 muestras serán adquiridos

ai.SampleRate = 10000;
ai.SamplesPerTrigger = 10000;

8.4.1 Obtener los datos

Revise la configuración básica para la adquisición.

ai

Display Summary of Analog Input (AI) Object Using 'PCI-6221'.

Acquisition Parameters: 10000 samples per second on each channel.
10000 samples per trigger on each channel.

1 sec. of data to be logged upon START.
Log data to 'Memory' on trigger.

Trigger Parameters: 1 'Immediate' trigger(s) on START.

Engine status: Waiting for START.
0 samples acquired since starting.
0 samples available for GETDATA.

AI object contains channel(s):

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'
3 '' 3 [-10 10] [-10 10] [-10 10] 'Volts'

8.4.2 Inicie el objeto de entrada analógica.

start(ai);

La toolbox adquiere datos en segundo plano mientras continúa la ejecución del código M en MATLAB. El código M es libre para ejecutar otras funciones mientras que la adquisición se lleva a cabo. Los datos se almacenan en la toolbox hasta que lo pongas en MATLAB.

Con el fin de pausar MATLAB, se utiliza la función wait, con esta función la adquisición se completa antes de intentar recuperar los datos. Si la adquisición se completa antes de que expire el tiempo de espera, MATLAB continuará.

wait(ai,2); % Esperar hasta 2 segundos para que la adquisición sea completa.

Cuando la adquisición se completa, para la transferencia de los datos de la toolbox hacia MATLAB se utiliza la función GetData, guardando la información en el workspace como una variable de datos. Para graficar esta información se utiliza la función plot.

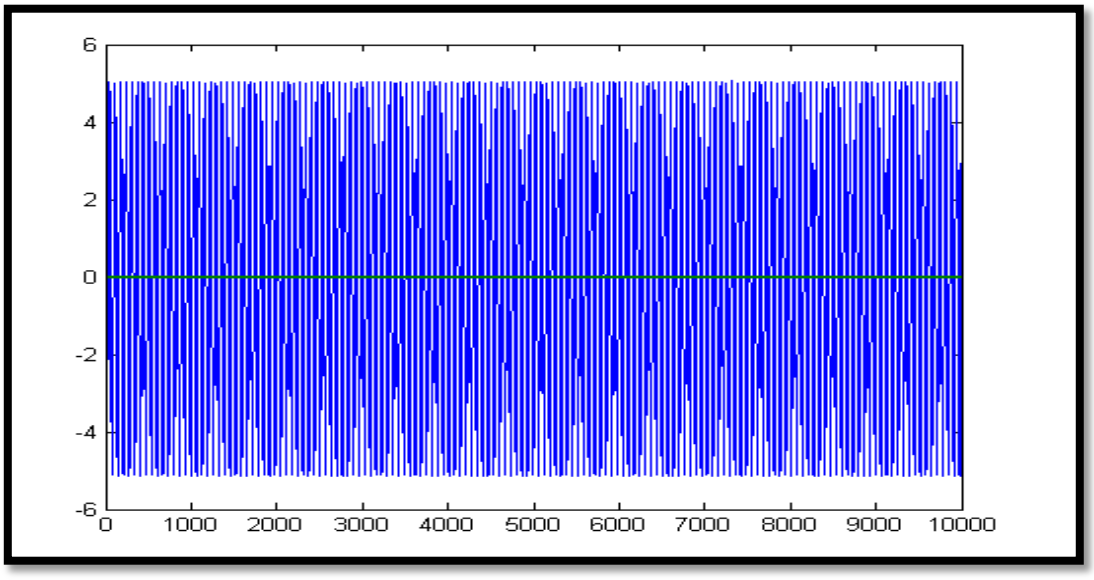
```
data = getdata(ai);
plot(data);
```

8.4.3 Limpiar

Esto completa la introducción a la entrada analógica. Dado que el objeto de entrada analógica ya no es necesario, usted debe:

* Detenga el objeto de entrada analógica de ejecución utilizando la función stop.
* Eliminar con la función delete para liberar memoria y otros recursos del sistema.

```
stop(ai);
delete(ai);
```



Grafica 39. Adquisición de diez mil muestras uso del Getdata

9 EXPERIENCIA PARA LA SINCORIZACION DE LA ENTRADA Y LA SALIDA ANALOGA.

Esta experiencia es un soporte para conozca la sincronización de los dispositivos usando funciones programables de interface (PFI) terminales y Real-Time Sistema de Integración (RTSI). Ambas son características del hardware de adquisición de datos de la National Instruments.

9.1 SINCRONIZACION DE LA ENTRADA Y SALIDA ANALOGA

A menudo, puede ser necesario para sincronizar el inicio de su salida analógica y las operaciones de entrada analógica. Esto se hace comúnmente para el estímulo / respuesta de las pruebas. Esta demostración podrá discutirse y evaluarse en algunos mecanismos disponibles en la toolbox.

9.2 CONECTAR Y CONFIGURAR EL HARDWARE.

Para esta experiencia, estamos usando un dispositivo PCI de National Instrument NI USB 6008. Canal 0 del subsistema de entrada analógica, Se conecta al canal 0 del subsistema de salida analógica.

Configure el subsistema de adquisición de datos de entrada analógica. Configure el subsistema de adquisición para obtener 1.000 muestras por segundo, 10 veces más rápido que el subsistema de salida analógica. Por lo tanto, será capaz de ver con claridad las actualizaciones de salida analógica.

```
ai = analoginput('nidaq','Dev1');
addchannel(ai,0);
ai.SampleRate = 1000;
ai.SamplesPerTrigger = 1000;
```

ai

Veamos el resumen de la entrada análoga (AI) Usando 'USB-6008'.

Adquisición de Parámetros: 1000 muestras por Segundo por cada canal.
1000 muestras por disparo sobre cada canal.
1 sec. de datos registrados en START.
Registro de datos para 'Memoria de disparo'.

Trigger Parameters: 1 'Immediate' trigger(s) on START.

Estado del motor: Esperando para START.
0 muestras adquiridas después de empezar.
0 muestras disponibles para GETDATA.

El objeto AI contiene los Canales:

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:
1 '' 0 [-10 10] [-10 10] [-10 10] 'Volts'

Configurar el subsistema de salida analógica

```
ao = analogoutput('nidaq','Dev1');
addchannel(ao,0);
ao.SampleRate = 1000;
ao
```

Mostrar resumen de salida análoga (AO) Object usando 'NI USB -6008'.

Parámetros de salida: 100 muestras por Segundo por cada canal.

Parámetros de disparo: 1 'Immediate' trigger on START.

Estado del motor: Esperando para START.
0 total sec. de datos en cola parar START.
0 muestras que se encuentran programadas para PUTDATA.
0 muestras enviadas a un dispositivo de salida START.

AO object contains channel(s):

Index:	ChannelName:	HwChannel:	OutputRange:	UnitsRange:	Units:
1	''	0	[-10 10]	[-10 10]	'Volts'

9.3 *SINCRONIZACION DE LA SALIDA Y ENTRADA ANALOGA USANDO START.*

Sin ningún tipo de coordinación de hardware, habrá una demora entre el comienzo de la entrada analógica y salida analógica. Esto representa el tiempo para preparar las operaciones.

Asegúrese de que la salida analógica está en cero. Esto asegura que no tengamos inconvenientes. Ver cuándo empieza la salida analógica.

```
putsample(ao,0)
```

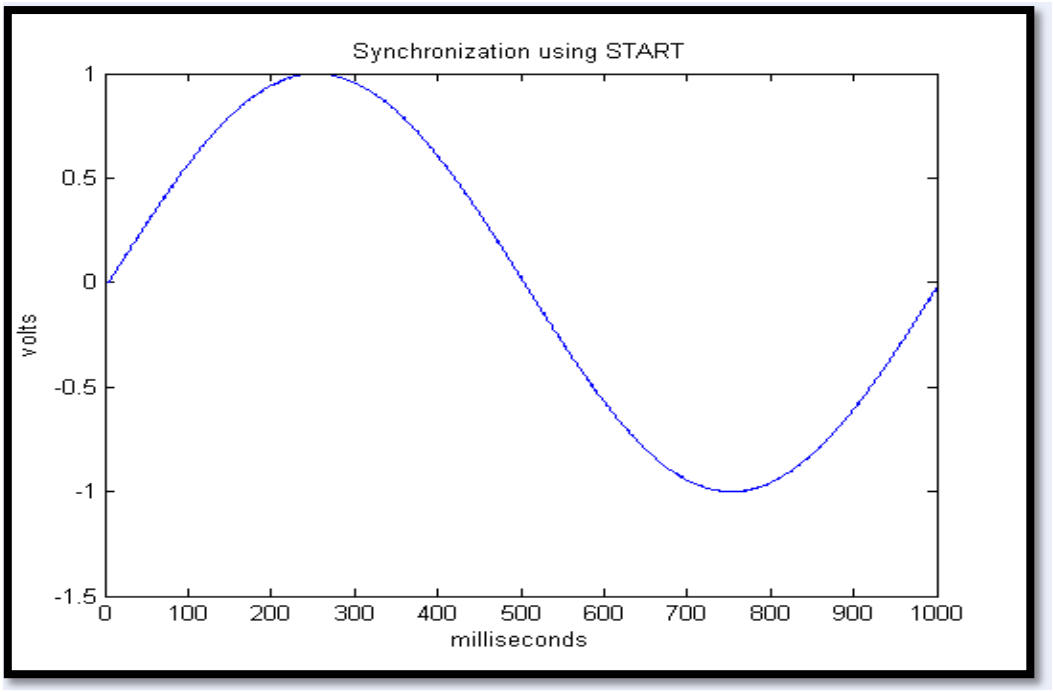
% Generar un señal de prueba de salida (1Hz onda senoidal) y la señal de %carga de prueba en el búfer de salida analógica.

```
outputSignal = sin(linspace(0,pi*2,ao.SampleRate)');
putdata(ao,outputSignal)
```

```
% Inicie la adquisición y generación. Estas dos operaciones no  están
%coordinadas por el hardware. Debido a que su comienzo no es en orden
%simplemente la entrada análoga comienza entes de la salida análoga.

start([ai,ao])

% Espere hasta dos segundos para que se puedan completar las operaciones.,
% Y recuperar los resultados de la Toolbox de adquisición de datos.

wait([ai,ao],2)
[data,time] = getdata(ai);
```

Grafique el resultado. Observe que en el extremo izquierdo, hay un retraso entre el inicio de la entrada analógica y la salida analógica.

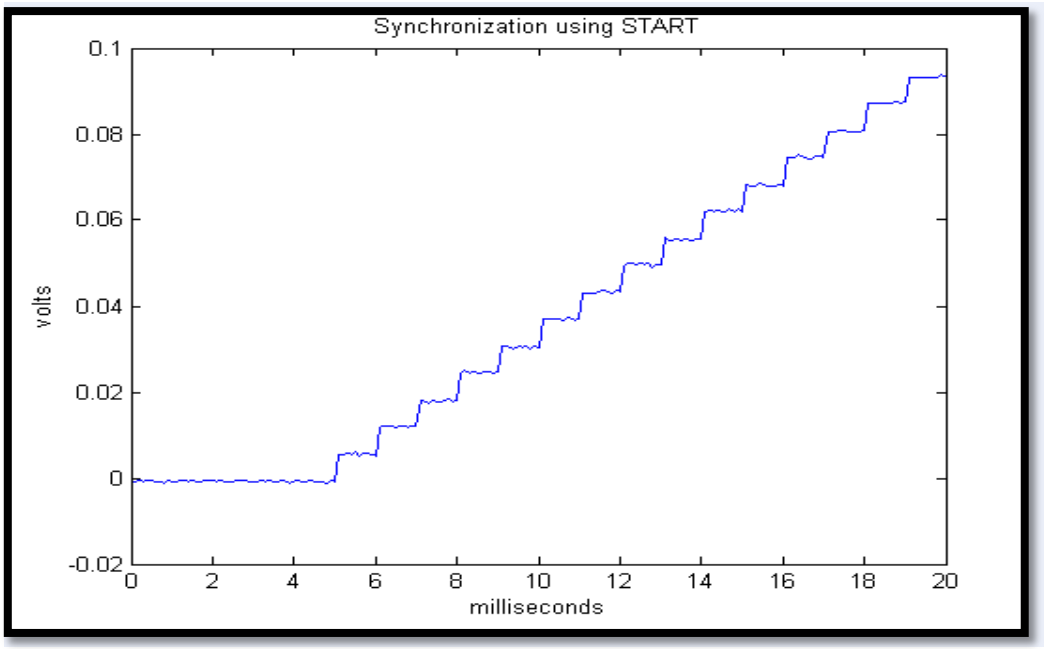
```
% Multiplique el l tiempo por  1000 para convertir segundos a milisegundos.

plot(time * 1000,data)
title('Synchronization using START')
xlabel('milliseconds')
ylabel('volts')
```



Grafica 40. Sincronización usando el START

Con un zoom in, se puede ver el retraso de 4-5 milisegundos.

xlim([0 20])



Grafica 41. Retraso de la entrada análoga con respecto a la salida.

9.4 SINCRONIZACION DE LA ENTRADA Y SALIDA ANALOGA USANDO EL TRIGGER MANUAL.

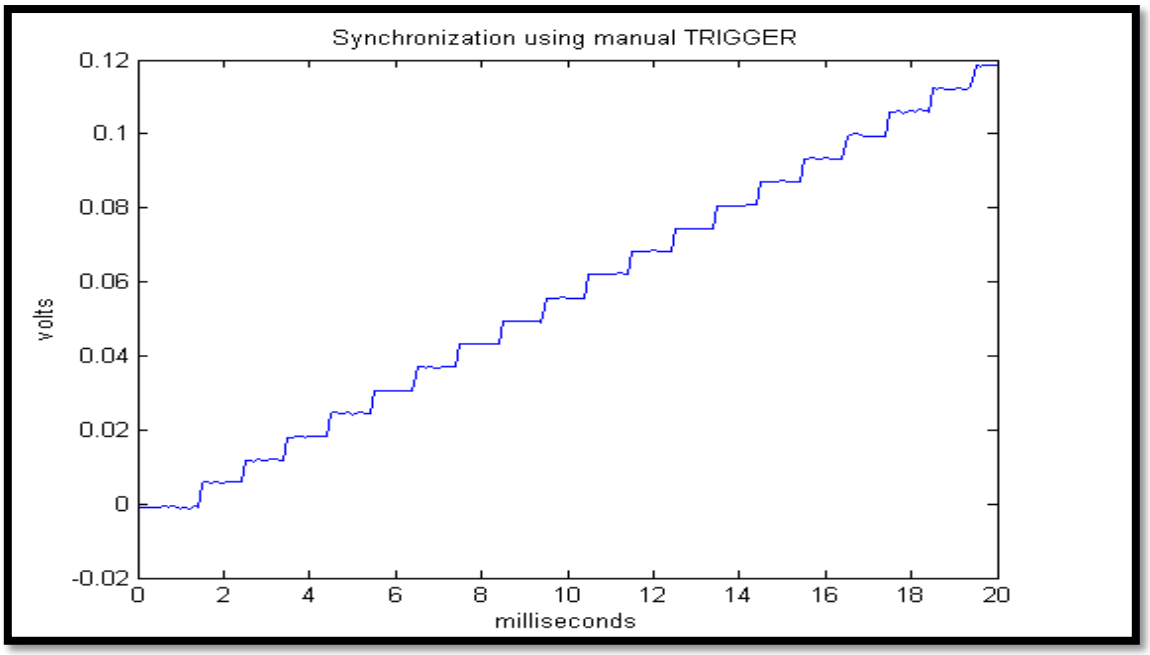
Podemos minimizar el retraso en el software con el comando TRIGGER. Esto establece la adquisición y generación antes de tiempo. Cuando el comando TRIGGER es usado, se utiliza una cantidad mínima de tiempo para iniciar las operaciones.

```
%Configure los subsistemas para el modo manual de trigger.

ai.TriggerType = 'Manual';
ao.TriggerType = 'Manual';
% Asegúrese de que la salida analógica está en cero.
putsample(ao,0)
%Cargue la  señal de prueba en el búfer de salida
analógica.putdata(ao,outputSignal)

% Establezca  la adquisición y generación. Con TriggerType establecido en
% modo 'Manual', START configura todo lo posible, salvo en realidad
% llamando el hardware para comenzar.

start([ai,ao])
%Iniciar los dos subsistemas. Estas dos operaciones no son coordinadas por
%hardware.

trigger([ai,ao])
wait([ai,ao],2)
[data,time] = getdata(ai);
```

Grafique el resultado. Tenga en cuenta que el sistema de entrada analógica muestra un retraso más pequeño al principio.

```
plot(time * 1000,data)
```

```
xlim([0 20])
title('Synchronization using manual TRIGGER')
xlabel('milliseconds')
ylabel('volts')
```



Grafica 42. Sincronización usando el modo manual del TRIGGER.

9.5 SINCRONIZACION DE LA ENTRADA Y SALIDA ANALOGA USANDO EL HARDWARE RTSI.

Para coordinar los subsistemas de entrada y salida análoga, hay un bus interno disponible en los dispositivos llamado RTSI.

RTSI se puede utilizar para sincronizar los dos subsistemas en la misma tarjeta sin cableado adicional. Además, puede utilizar RTSI para sincronizar múltiples subsistemas en múltiples tarjetas con un cable.
Se puede eliminar el retraso de la coordinación de los subsistemas usando el bus RTSI. Configure el comienzo de la adquisición para desencadenar el inicio de generación de hardware.

Configurar los subsistemas al utilizar RTSI. Usted configura el subsistema de entrada analógica como "el maestro". Cuando se inicie, se envía un pulso en el bus RTSI. El subsistema de salida analógica (el "esclavo") detecta el pulso en el bus RTSI, y empezar en cuestión de nanosegundos.

```
% Establecer el subsistema de entrada analógica para iniciar cuando el comando
START se publicará en breve.
ai.TriggerType = 'Immediate';

% Establecer el sistema de entrada analógica a la señal on line RTSI 0 cuando se
inicia.
ai.ExternalTriggerDriveLine = 'RTSI0';
ao.TriggerType = 'HwDigital';
ao.HwDigitalTriggerSource = 'RTSI0';
% Asegúrese de que la salida analógica está en cero.
putsample(ao,0)

% Cargar la señal de prueba en el búfer de salida analógica.
putdata(ao,outputSignal)
```

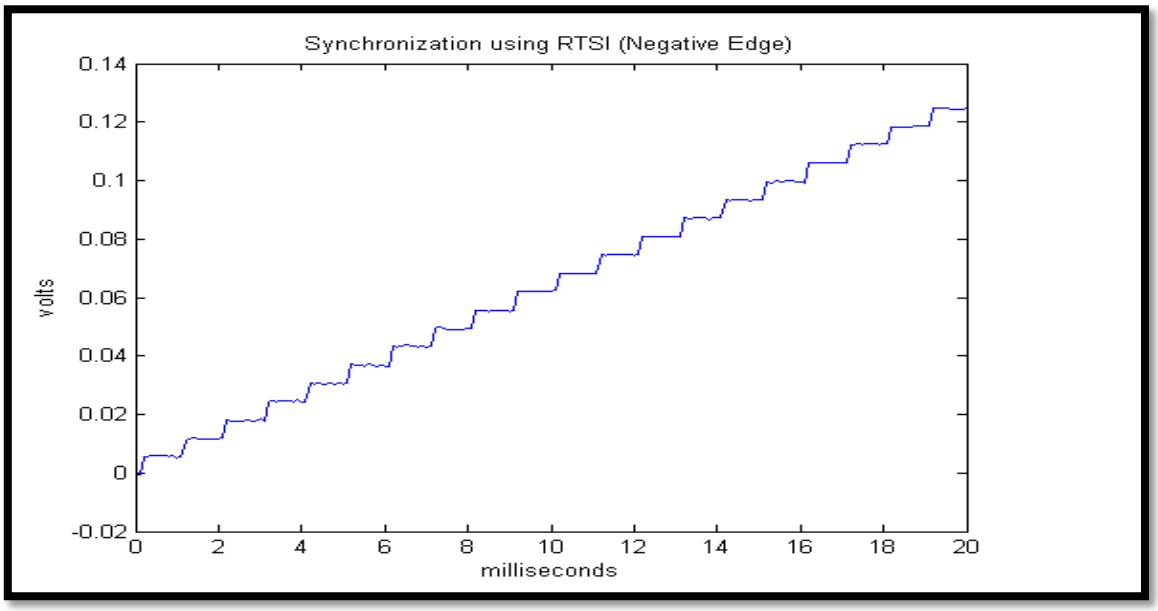
Inicie la adquisición y generación. Tenga en cuenta que debe iniciar la salida analógica primero, ya que debe esperar a la entrada analógica para empezar.

```
start(ao)
start(ai)

wait([ai,ao],2)
[data,time] = getdata(ai);
```

Grafique el resultado. Observe que la salida analógica se inició antes de la entrada analógica. La primera lectura debería haber sido cero. De forma predeterminada, el valor predeterminado para TriggerCondition salida analógica es 'NegativeEdge',

pero las líneas de alta actividad RTSI operan. El efecto es que el gatillo se produjo al final del pulso de disparo, en lugar de al principio.

```
plot(time * 1000,data)
xlim([0 20])
title('Synchronization using RTSI (Negative Edge)')
xlabel('milliseconds')
ylabel('volts')
```



Grafica 43. Sincronización usando el RTSI.

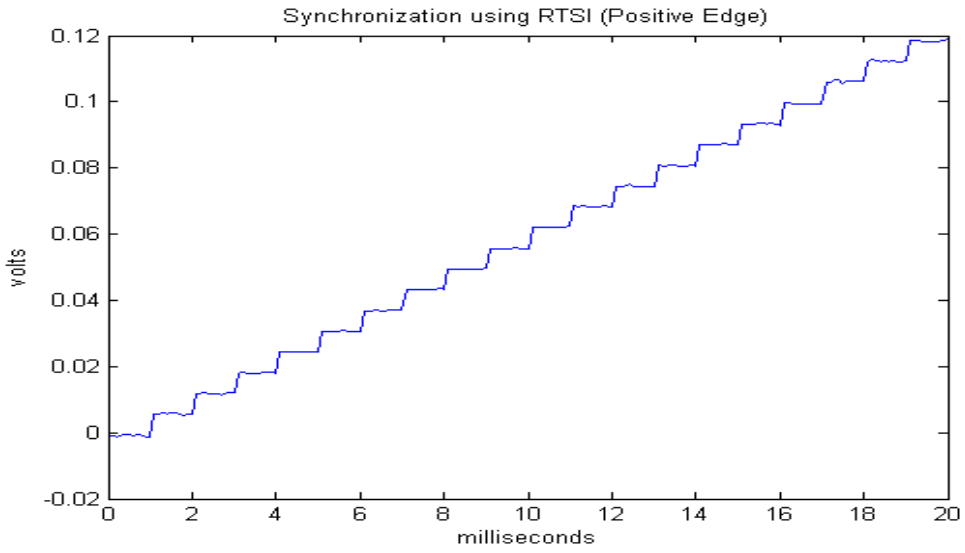
Repita la prueba, con la TriggerCondition AO en 'PositiveEdge "y repita la operación.

```
ao.TriggerCondition = 'PositiveEdge';


% Asegúrese de que la salida analógica está en cero.
putsample (ao, 0)

%Cargar  la señal de prueba en el búfer de salida analógica.

putdata(ao,outputSignal)
% Inicie la adquisición y generación de nuevo.
```

```
start(ao)
start(ai)

wait([ai,ao],2)
[data,time] = getdata(ai);
```

Grafique el resultado. Ahora la salida analógica y la entrada iniciaran simultáneamente.

```
plot(time * 1000,data)
xlim([0 20])
title('Synchronization using RTSI (Positive Edge)')
xlabel('milliseconds')
ylabel('volts')
```



Grafica 44. Sincronización usando el RTSI inicio simultaneo de entrada y salida análoga.

Cuando haya terminado con los objetos de entrada analógica, debe eliminarlos para garantizar que los recursos de hardware son liberados.

```
delete(ao)
delete(ai)
```

10 EXPERIENCIA DE COMANDOS DE AYUDA EN LA TOOLBOX DE ADQUISICION DE DATOS.

Esta experiencia le muestra cómo determinar qué hardware y adaptadores están instalados en su máquina, le muestra cómo determinar qué objetos de la Toolbox de adquisición de datos se pueden crear para cada adaptador instalado y qué comandos son necesarios para crear esos objetos, también le muestra cómo obtener ayuda en la adquisición de datos, propiedades y funciones.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

En esta parte de la experiencia se determinará:

* Qué hardware y adaptadores están instalados en su máquina
* Qué objetos en la Toolbox de Adquisición de Datos se pueden crear.
* Cómo puede encontrar información y ayuda en un objeto de adquisición de datos.

Un dispositivo de hardware es la interfaz que le permite pasar la información entre el motor de adquisición de datos y el controlador del hardware. La Toolbox de adquisición de datos le proporciona información acerca de los dispositivos:

* Advantech ® ('Advantech')
* Medición ™ Informática ("MCC")
* National Instruments ® ('nidaq')

* Paralelamente los puertos ('paralelo)
* Tarjetas de sonido ("winsound ')

Se puede visualizar la información relacionada con el hardware y MATLAB con el comando DAQHWINFO.

```
daqhardwareinfo = daqhwinfo
daqhardwareinfo =
        ToolboxName: 'Data Acquisition Toolbox'
     ToolboxVersion: '2.12 (R2008a)'
      MATLABVersion: '7.6 (R2008a)'
  InstalledAdaptors: {4x1 cell}
```

La información devuelta incluye una lista de adaptadores de hardware instalados en su máquina.

```
daqinstalledadaptors =
  'mcc'
  'nidaq'
  'parallel'
  'winsound'
```

Si se necesita instalar un adaptador o mover un adaptador existente, entonces debe ser registrado manualmente. Un adaptador de hardware está registrado manualmente con el comando:

```
daqregister('adaptorname')
```

Donde 'adaptorname' es uno de los adaptadores de hardware soportado como 'winsound', 'nidaq "," MCC ", etc

```
daqregister('winsound')
ans =
```

'mwwinsound.dll' successfully registered.

Ahora que sabemos lo que son adaptadores instalados en el equipo, puede mostrar información específica relacionada con un adaptador al pasar el nombre del adaptador de DAQHWINFO. Esta información incluye:

* El nombre del archivo DLL adaptador de hardware y su versión
* La descripción del controlador del hardware y su versión.
* Comandos de MATLAB para crear un objeto de adquisición de datos utilizando el nombre del adaptador especificado.

Los comandos para crear un objeto de adquisición de datos winsound se enumeran a continuación.

```
daqwinsoundinfo = daqhwinfo('winsound')
daqwinsoundinfo =
          AdaptorDllName: 'C:\Program
Files\MATLAB\R2008a\toolbox\daq\daq\private\mwwinsound.dll'
       AdaptorDllVersion: '2.12 (R2008a)'
             AdaptorName: 'winsound'
              BoardNames: {'NVIDIA(R) nForce(TM) Audio'}
       InstalledBoardIds: {'0'}
   ObjectConstructorName: {'analoginput('winsound',0)'
'analogoutput('winsound',0)'  ''}
```

Puede crear un objeto de entrada analógica y un objeto de salida analógica para el adaptador de winsound. El tercer valor devuelto está vacío porque no admiten tarjetas de sonido digital I / O.

```
daqwinsoundinfo.ObjectConstructorName{:}
ans =
analoginput('winsound',0)
ans =
```

```
analogoutput('winsound',0)
ans =
    ''
```

Ahora que sabemos que el comando para crear un objeto de entrada analógica para el adaptador de winsound, lo creamos:

```
ai = analoginput('winsound', 0);
ai
Display Summary of Analog Input (AI) Object Using 'NVIDIA(R) nForce(TM) Audio'.
  Acquisition Parameters:  8000 samples per second on each channel.
                           8000 samples per trigger on each channel.
                           1 sec. of data to be logged upon START.
                           Log data to 'Memory' on trigger.
     Trigger Parameters:  1 'Immediate' trigger(s) on START.
         Engine status:  Waiting for START.
                           0 samples acquired since starting.
                           0 samples available for GETDATA.
AI object contains no channels.


ao = analogoutput('winsound');
ao


Display Summary of Analog Output (AO) Object Using 'NVIDIA(R) nForce(TM)
Audio'.
       Output Parameters:  8000 samples per second on each channel.
      Trigger Parameters:  1 'Immediate' trigger on START.
          Engine status:  Waiting for START.
                           0 total sec. of data currently queued for START.
                           0 samples currently queued by PUTDATA.
                           0 samples sent to output device since START.
AO object contains no channels.
```

Usted puede encontrar información concreta respecto al objeto de entrada analógica o un objeto creado para salida analógica al pasar el objeto a DAQHWINFO. Esta información incluye:

* El número de canales que se pueden agregar al objeto
* El rango de frecuencias de muestreo
* El tipo de datos nativos
* La polaridad

```
daqhwinfo(ai)
ans =
              AdaptorName: 'winsound'
                     Bits: 16
                 Coupling: {'AC Coupled'}
               DeviceName: 'NVIDIA(R) nForce(TM) Audio'
          DifferentialIDs: []
                    Gains: []
                       ID: '0'
              InputRanges: [-1 1]
            MaxSampleRate: 44100
            MinSampleRate: 8000
           NativeDataType: 'int16'
                 Polarity: {'Bipolar'}
               SampleType: 'SimultaneousSample'
           SingleEndedIDs: [1 2]
            SubsystemType: 'AnalogInput'
            TotalChannels: 2
    VendorDriverDescription: 'Windows Multimedia Driver'
       VendorDriverVersion: '5.10'
```

Y para el objeto de salida analógica:

```
daqhwinfo(ao)
```

```
ans =
              AdaptorName: 'winsound'
                     Bits: 16
               ChannelIDs: [1 2]
                 Coupling: {'AC Coupled'}
               DeviceName: 'NVIDIA(R) nForce(TM) Audio'
                       ID: '0'
            MaxSampleRate: 44100
            MinSampleRate: 8000
           NativeDataType: 'int16'
             OutputRanges: [-1 1]
                 Polarity: {'Bipolar'}
               SampleType: 'SimultaneousSample'
            SubsystemType: 'AnalogOutput'
            TotalChannels: 2
   VendorDriverDescription: 'Windows Multimedia Driver'
      VendorDriverVersion: '5.10'
```

Mediante la configuración de los valores de las propiedades para sus objetos de adquisición de datos, se puede controlar el comportamiento de su aplicación de adquisición de datos.

Los objetos de la Toolbox de adquisición de datos (objetos de entrada analógica, salida analógica y digital I / O objetos) contienen dos tipos de propiedades:

1) Propiedades de base
2) Propiedades específicas del dispositivo

Las propiedades de base aplica a los subsistemas de hardware, todo ello apoyado de un tipo determinado (entrada analógica, salida analógica, digital I / O).

Por ejemplo, todos los objetos de entrada analógica tienen una propiedad de muestreo, independientemente del proveedor de hardware. Además de las propiedades de base, puede haber un conjunto de propiedades específicas del dispositivo que está determinado por el hardware que esté utilizando.

Cuando PROPINFO se llama con un objeto como argumento de entrada, una estructura se devuelve. Los nombres de campo de la estructura son los nombres de las propiedades. Los valores de campo son una estructura que contiene los datos de la propiedad. Esta información incluye:

- El tipo de propiedad de datos
- Limitaciones en el valor de la propiedad
- El valor predeterminado de la propiedad
- Para modificar las propiedades solo puede cuando el objeto no está corriendo.
- Una indicación de si la propiedad es específica del dispositivo

Por ejemplo, puede obtener información sobre la propiedad de muestreo con el comando que se muestra a continuación. La información devuelta indica la propiedad de muestreo:

* Debe ser un doble entre los valores 8000 y 44100
* Tiene un valor predeterminado de 8000
* Puede ser modificado únicamente cuando el objeto no se está ejecutando
* Es una propiedad de la base a disposición de todos los objetos de entrada analógica

```
aiInfo = propinfo(ai);
aiInfo.SampleRate
ans =
```

```
            Type: 'double'
      Constraint: 'bounded'
ConstraintValue: [8000 44100]
   DefaultValue: 8000
        ReadOnly: 'whileRunning'
 DeviceSpecific: 0
```

Alternativamente, usted puede conocer un nombre de propiedad con el comando PROPINFO. En este caso, la información para la propiedad especificada es devuelta.

```
aoOut = propinfo(ao, 'TriggerType')
aoOut =
            Type: 'string'
      Constraint: 'enum'
ConstraintValue: {'Manual'  'Immediate'}
   DefaultValue: 'Immediate'
        ReadOnly: 'whileRunning'
 DeviceSpecific: 0
```

La información devuelta indica la propiedad TriggerType:

* Tiene un valor predeterminado de "inmediato".
* Puede ser modificado únicamente cuando el objeto no se está ejecutando
* Es una propiedad de la base a disposición de todos los objetos de salida analógica.
* Está obligado a ser una enumeración con los únicos valores aceptables están 'Manual' e 'inmediata', según lo indicado por

aoOut.ConstraintValue

ans =

'Manual' 'Immediate'

Una descripción más detallada de propiedad se da con el comando DAQHELP. Las propiedades relacionadas se muestran en medio de comillas, letras minúsculas y mayúsculas.

daqhelp SampleRate
SAMPLERATE doublé

SampleRate Especifica la velocidad de muestreo por canal que los datos son digitalizados.

SampleRate Especifica la velocidad de muestreo por canal (en muestras por segundo) que de la entrada analógica (AI) o la salida analógica (AO) del subsistema digitaliza los datos.

AI y subsistemas AO tienen un número finito (aunque a menudo grande) número de validez velocidades de muestreo. Si se especifica una frecuencia de muestreo que no coincide con uno de los valores válidos, el motor de adquisición de datos selecciona automáticamente la frecuencia de muestreo más cercano disponible.

Desde el muestreo se puede establecer en un valor que difiere de la especificada, se debe devolver la frecuencia de muestreo real que utiliza la función get o el dispositivo resumen objeto de visualización. Alternativamente, puede utilizar la función SETVERIFY que establece el valor de muestreo y devuelve el valor real que se establece.

Para conocer la gama de velocidades de muestreo con el apoyo de su junta directiva, utilice la PROPINFO función.

El valor de muestreo no puede ser modificado mientras el objeto se está ejecutando.

daqhelp StartFcn
STARTFCN string, function_handle

startFcn especifica la función M-file para ejecutar antes de que el dispositivo objeto y el dispositivo de hardware comenzar a ejecutar.

Un evento de inicio se genera inmediatamente después de que el comando START se ejecute.

Este evento se ejecuta el M-archivo especificado para StartFcn.
Cuando el M-StartFcn archivo ha terminado de ejecutar, ejecutar de forma automática en On, el objeto de dispositivo y el dispositivo de hardware comenzar a ejecutarse.

DAQHELP también se puede utilizar también para mostrar las funciones de ayuda de la Toolbox de Adquisición de Datos. Esto puede ser útil en el acceso a la ayuda de una función desde MATLAB.
Mediante el uso de DAQHELP, usted no tiene que saber el nombre de clase de un objeto para obtener la ayuda correcta

daqhelp delete

DELETE Elimina del motor los objetos de la toolbox de adquisición de datos.

DELETE (OBJ) elimina objeto, OBJ, desde el motor de adquisición de datos. Si los canales o líneas están asociados con OBJ, también son eliminados. OBJ puede ser un conjunto de dispositivos o una entrada de tipo canal ó línea.

Si OBJ es el último objeto identificado por el hardware de acceso, la asociación de controlador de hardware y el adaptador se cierran (Van a off) y no se carga el objeto.

Usando el CLEAR se puede remover el objetos desde el workspace

DELETE debe ser utilizado al término de una sesión de adquisición de datos.

Si se está ejecutando un OBJ, DELETE (OBJ) eliminará OBJ y una advertencia será publicada. Si se está ejecutando OBJ y se ejecuta DELETE (OBJ) el objeto no se elimina y se genera un error.

Si se encuentran múltiples referencias de un objeto de adquisición de datos en el área de trabajo, sí eliminamos una de esas referencias invalidara automáticamente el restante de las referencias.

Ejemplo:

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

Cuando haya terminado, se borra el objeto y limpiamos la variable del espacio de trabajo para liberar recursos del sistema.

```
delete(ai);
delete(ao);
```

11 PROGRAMANDO CON MATLAB

11.1 GENERALIDADES

Programar en MatLab es usar una serie de comandos que permitan realizar una tarea o función específica. Estos pueden ser escritos uno por uno a través de la línea de comandos:

```
>>A=[1 2 3;4 5 6;7 8 9]
A =
1 2 3
4 5 6
7 8 9
>>A'
ans =
1 4 7
2 5 8
3 6 9
```

El primer comando A=[1 2 3;4 5 6;7 8 9] define la matriz A y el siguiente comando A' calcula y presenta en pantalla la transpuesta de A.

11.2 ARCHIVOS-M: COMANDOS Y FUNCIONES

Los archivos de disco que contienen instrucciones de MATLAB se llaman archivos-M. Esto es así porque siempre tienen una extención de ".m" como la última parte de su nombre de archivo.

Un archivo-M consiste de una secuencia de instrucciones normales de MATLAB, que probablemente incluyen referencias a otros archivos-M. Un archivo-M se puede llamar a sí mismo recursivamente. Puedes crear archivos-M utilizando un editor de texto ó procesador de palabras.

Hay dos tipos de archivos-M: los de comandos y las funciones. Los archivos de comandos, automatizan secuencias largas de comandos. Los archivos de funciones, permiten añadir a MATLAB funciones adicionales expandiendo asi la capacidad de este programa. Ambos, comandos y funciones, son archivosordinarios de texto ASCII.

11.3 ARCHIVOS DE COMANDOS

Cuando un archivo de comandos es invocado, MATLAB simplemente ejecuta los comandos encontrados en dicho archivo. Las instrucciones en un archivo de comando operan globalmente en los datos en el espacio de trabajo. Los comandos son utilizados para hacer análisis, resolver problemas, ó diseñar secuencias largas de comandos que se conviertan en interactivas. Por ejemplo, suponga que el archivo fibo.m contiene los siguientes comandos de MATLAB:

```
% Un archivo-M para calcular los elementos de la serie de Fibonacci
f = [1 1]; i = 1;
while f(i) + f(i+1) < 1000
f(i+2) = f(i) + f(i+1);
i = i + 1;
end
plot(f)
```

Si escribimos fibo en una ventana de MATLAB seguido de "enter" vemos que MATLAB calcula los primeros 16 números de Fibonacci, y luego grafica estos. Luego que la ejecución del archivo es completada, las variables f y i permanecen en el espacio de trabajo.

Los programas de demostraciones incluidos en MATLAB son ejemplos de como usar comandos para hacer tareas más complicadas. Para utilizar estos escribas demos en el "prompt" de MATLAB.

11.4 ARCHIVOS DE FUNCIONES

Un archivo-M que contiene la palabra function al principio de la primera línea, es un archivo de función. En una función, a diferencia de un comando, se deben de pasar los argumentos. Las variables definidas y manipuladas dentro de la función son locales a esta y no operan globalmente en el espacio de trabajo. Los archivos de funciones se utilizan para extender a MATLAB, i.e., crear nuevas funciones para MATLAB utilizando el lenguaje propio de MATLAB.

El archivo mean.m contiene las instrucciones:

```
function y = mean(x)
% Valor medio.
% Para vectores, mean(x) retorna el valor medio de los elementos del vector x.
% Para matrices, mean(x) es un vector fila conteniendo el valor medio de cada columna.
[m, n] = size(x);
if m == 1
m = n;
end
y = sum(x)/m;
```

(Las lineas que comienzan con "%" son interpretadas como comentarios por MATLAB). La existencia de este archivo en el disco duro define una nueva función en MATLAB llamada mean. Si z es un vector de los enteros desde 1 a 99, por ejemplo,

```
z = 1:99;
```

entonces, el valor promedio es encontrado escribiendo

```
mean(z)
```

que resultaría

```
ans =
50
```

Veamos algunos detalles de mean.m:

La primera línea declara el nombre de la función, los argumentos de entrada, y los argumentos de salida. Sin esta línea sería un archivo de comando.
% indica que el resto de la línea es un comentario.
Las primeras líneas documentan el archivo-M y aparecen en la pantalla cuando escribimos help mean.
Las variables m, n, e y son locales a mean y no existen en el espacio de trabajo. (O si existen, permanecen sin cambios.)
No es necesario asignar los enteros de 1 al 99 en la variable x. Utilizamos mean con una variable llamada z.
Este vector que contenía los enteros de 1 a 99 fue pasado ó copiado a mean donde se convirtió en una variable local llamada x.

Ejemplo

```
% Ejemplo de un archivo-m
% Creación del vector x usando el comando for
n=5;
for i=1:n
x(i)=i^2;
end
x
% Fin del archivo-m
```

Este ejemplo es un archivo-m tipo comando. Para ejecutarlo, en la línea de comandos se debe escribir el nombre del archivo:

```
>>ejemplo
x =
1 4 9 16 25
```

Ejemplo

```
% Calcula el promedio de los elementos de un vector y dibuja dicho vector
% Sintaxis: promedio(x) donde x es el vector a promediar
function p = promedio(x)
```

```
n=length(x);
p=0;
for i=1:n
p=p+x(i);
end
p=p/n;
plot(x);
```

Para ejecutar la función, se hace la llamada en la línea de comandos incluyendo el parámetro. La función promedio usa por parámetro un vector. Este vector debe ser definido previamente.

```
>>A=[1 2 4 3 7 5 6 1 2 0 8 5];
>>promedio(A)
ans =
3.6667
```

MatLab presenta las imágenes en una ventana de figuras. Al observar el contenido de dicha ventana luego de ejecutar la función promedio, se tiene:

Esta imagen es el resultado del comando plot(x) al ejecutar la función promedio. MatLab posee un conjunto de archivos-m incorporados (built-in). Puede agregársele archivos-m definidos por el usuario almacenando los mismos en el directorio principal de MatLab. Los comentarios incluidos en estos scripts y funciones se visualizan al usar el comando help seguido del nombre del archivo.

```
>>help promedio
Calcula el promedio de los elementos de un vector y dibuja dicho vector
Sintaxis: promedio(x) donde x es el vector a promediar
```

Para ver el contenido de un archivo-m se usa el comando type seguido del nombre del archivo.

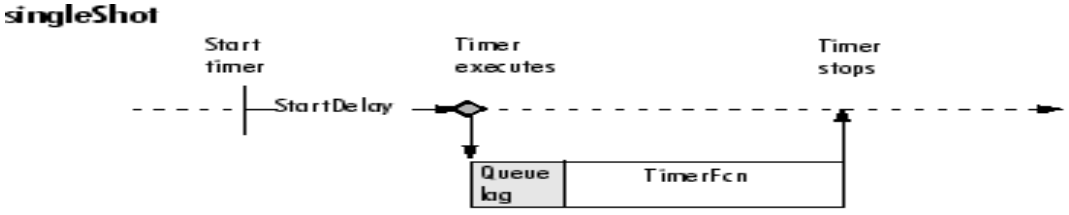
12 MODO DE EJECUCION DEL TEMPORIZADOR DE OBJETOS

Ejecución de un temporizador con la función de respaldo una sola vez

El temporizador de objetos soporta varios modos de ejecución que determinan cómo está programado el tiempo de devolución de llamado ejecutado con la función (TimerFcn). Se especifica el modo de ejecución, estableciendo el valor en la propiedad Execution Mode.

Para ejecutar una sola vez una función de devolución de llamada del temporizador, se llama a la propiedad ExecutionMode a 'singleShot. Este es el modo de ejecución por defecto. En este modo, el temporizador de objetos inicia la temporización y después del período de tiempo especificado con la propiedad StartDelay, se adiciona la función de devolución de llamada del temporizador (TimerFcn) a la cola de ejecución de MATLAB. Cuando finalice la función de temporizador de devolución de llamada, el cronómetro se detiene.

La siguiente figura ilustra gráficamente las partes de la ejecución del temporizador de devolución de llamada para un modo de ejecución singleShot. El área sombreada en la figura, denominada Queue lag (Retardo de la cola), representa la cantidad de tiempo entre el retardo de inicio llamado StartDelay y desde el instante cuando la función se comienza a ejecutar. La duración de este retraso depende de los procesos que se estén ejecutando en MATLAB en ese momento.



Grafica 45. Temporizador de devolución de llamada (singleShot)

13 EJECUCIÓN DE UNA DEVOLUCIÓN DE LLAMADO CON LA FUNCIÓN DE TIEMPOS MÚLTIPLES

El temporizador de objetos admite tres modos múltiples de ejecución:

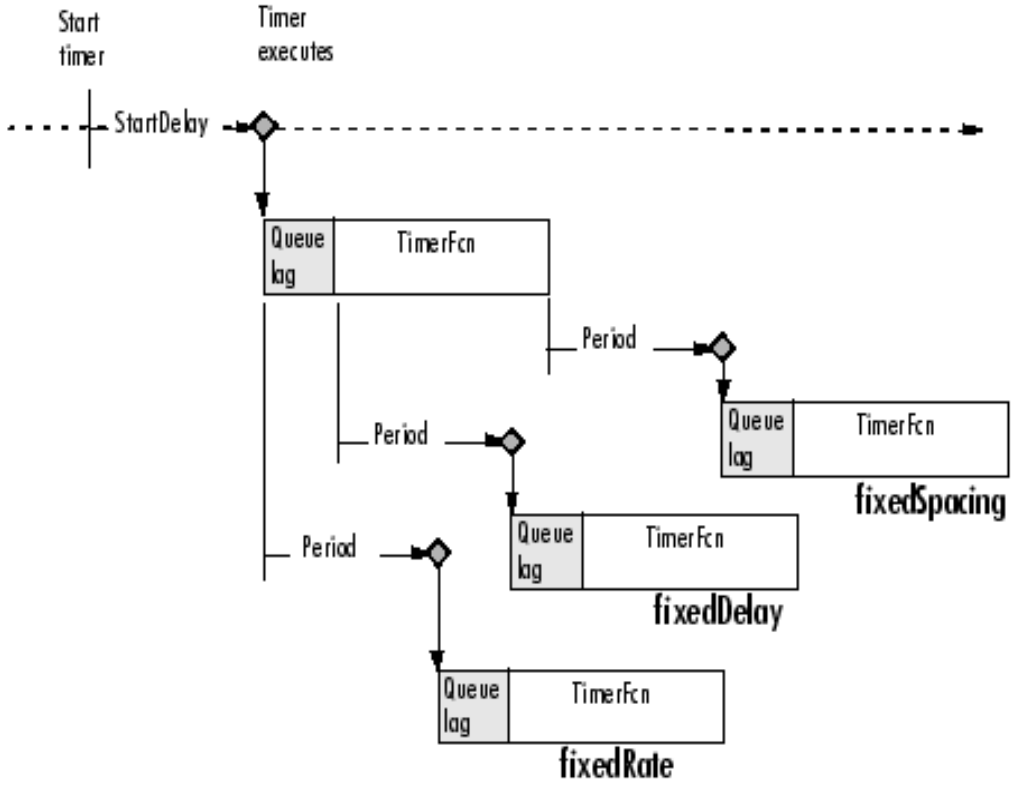
- 'fixedRate'
- 'fixedDelay'
- 'fixedSpacing'

En muchos sentidos, estos modos de ejecución funcionan al tiempo:
* La propiedad TasksToExecute especifica el número de veces que desea que el temporizador ejecuta la función de devolución de llamada (TimerFcn).
* La propiedad Período (period), especifica la cantidad de tiempo entre ejecuciones de la función del temporizador de devolución de llamada.
* La propiedad BusyMode especifica cómo el temporizador de objetos maneja colas de la función del temporizador de devolución de llamada que no han finalizado.

Los modos de ejecución difieren solo en donde ellos empiezan a medir el período de tiempo entre ejecuciones. En la tabla siguiente se describen estas diferencias.

Modo de Ejecución	Descripción
'fixedRate'	Periodo de tiempo entre ejecuciones, comienza inmediatamente después de la función de devolución de llamada del temporizador.
'fixedDelay'	Periodo de tiempo entre ejecuciones comienza cuando la función del temporizador de devolución de llamada realmente empieza la ejecución.
'fixedSpacing'	Periodo de tiempo entre ejecuciones comienza cuando la función de devolución de llamada del temporizador termina de ejecutarse.

La siguiente figura ilustra la diferencia entre estos modos. Tenga en cuenta que la cantidad de tiempo entre ejecuciones (especificado por la propiedad Periodo) sigue siendo el mismo. Sólo el punto en que comienza la ejecución es diferente.



Grafica 46. Diferencias entre los modos de ejecución

14 EXPERIENCIA DE MEDICION DE RUIDO PERSONALIZANDO LAS MEDICIONES DEL TRIGGER EN LA ENTRADA ANALOGA.

En esta experiencia se muestra cómo encontrar y adquirir un período de al menos 2 segundos de relativa calma que se produce entre dos señales. Esto sería útil para aplicaciones que necesitan medir el ruido de fondo y calcular una relación señal-ruido.

Para este ejemplo mostraremos cómo iniciar un software trigger utilizando la personalización de condiciones, que es mas complejo que los valores estándar proporcionados por la propiedad de los datos de objetos TriggerCondition.

Contenido

* Crear el objeto de entrada analógica
* Definir la información Trigger Condition
* Configurar el temporizador de devolución de llamada
* Configurar las propiedades del trigger
* Configurar la cantidad de datos para adquirir
* Iniciar la adquisición.
* Limpiar
* La función de llamada del temporizador

En primer lugar, encuentre objetos que se estén ejecutándose dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
```

```
    stop(daqfind)
end
```

14.1 CREE EL OBJETO DE ENTRADA ANALÓGICA

Antes de la adquisición de los datos, tenemos que crear un objeto de entrada analógica y especificar los canales para la adquisición de datos.

Un objeto de entrada análoga se crea con la función analoginput. Esta función acepta el nombre del adaptador y el ID del dispositivo como argumentos de entrada. El ID del dispositivo se refiere al número asociado con la tarjeta cuando ésta es Instalada. (Cuando se una NI-DAQmx, esta es usualmente una línea como 'Dev1').

```
% Crear el objeto de entrada analógica y especifique
% el canal por donde los datos deben ser recogidos.
AI = analoginput('nidaq','Dev3');
addchannel(AI,1);
```

14.2 DEFINIR LA CONDICIÓN DE INFORMACIÓN DEL TRIGGER

Debemos inicializar una estructura de MATLAB ® con la información necesaria para determinar cuando la condición de activación personalizada se ha cumplido. Buscamos

* Una señal
* Por lo menos 2 segundos de silencio

```
% La señal definida es mayor de 0,15 voltios.
extraData.signalVoltage = 0.15;
extraData.signalFound = false;
% Para pasar a quiet se define una señal menor de 0,04 voltios.

extraData.quietVoltage = 0.04;
extraData.secondsOfQuietNeeded = 2;
extraData.secondsOfQuietFound = 0;
extraData.quietFound = false;
```

Se guarda esta información en la propiedad UserData de modo que pueda acceder y modificar por una llamada de temporizador.

```
set(ai, 'UserData', extraData);
```

14.3 CONFIGURAR EL TIEMPO DE LLAMADO

Usamos las propiedades TimerFcn y TimerPeriod para llamar a una función M-file a intervalos regulares mientras que la adquisición se ejecuta. Usamos esta función de devolución de llamada del temporizador para examinar los datos entrantes para ver si la condición de activación se ha cumplido.

NOTA: La función de devolución de llamada del temporizador aparece en la parte final de esta demostración.

```
% Hacer período del temporizador 1 / 10 de nuestro período quiet.
timerPeriod = extraData.secondsOfQuietNeeded/10;
```

```
set(AI, 'TimerFcn', @TimerCallback);
set(AI, 'TimerPeriod', timerPeriod);
```

14.4 *CONFIGURAR LAS PROPIEDADES DEL TRIGGER.*

Usamos un trigger manual y configuramos el objeto para adquirir los datos durante el período de quiet.

Al establecer la propiedad TriggerType a 'manual', el registro de datos no comenzará hasta que la función del disparador se ejecute. La devolución de llamada del temporizador se ejecutará el trigger cuando las condiciones adecuadas se han cumplido.

Además, se establece la propiedad TriggerDelay a un valor negativo para indicar que queremos registrar los datos en el período de quiet que se produjeron antes de que el trigger se halle ejecutado.

```
set(AI, 'TriggerType', 'manual');
set(AI, 'TriggerDelay', -extraData.secondsOfQuietNeeded);
set(AI, 'TriggerDelayUnits', 'seconds');
```

14.5 *CONFIGURAR LA CANTIDAD DE DATOS PARA ADQUIRIR*

Hemos establecido la propiedad SamplesPerTrigger para indicar la cantidad máxima de datos que la adquisición debe tomar. Sin embargo, la devolución de llamada del temporizador se detendrá antes de la adquisición, en caso de detectar una segunda señal después del período de quiet.

```
sampleRate = get(AI, 'SampleRate');
set(AI, 'SamplesPerTrigger', 10 * extraData.secondsOfQuietNeeded * sampleRate);
```

14.6 *COMIENZO DE LA ADQUISICIÓN*

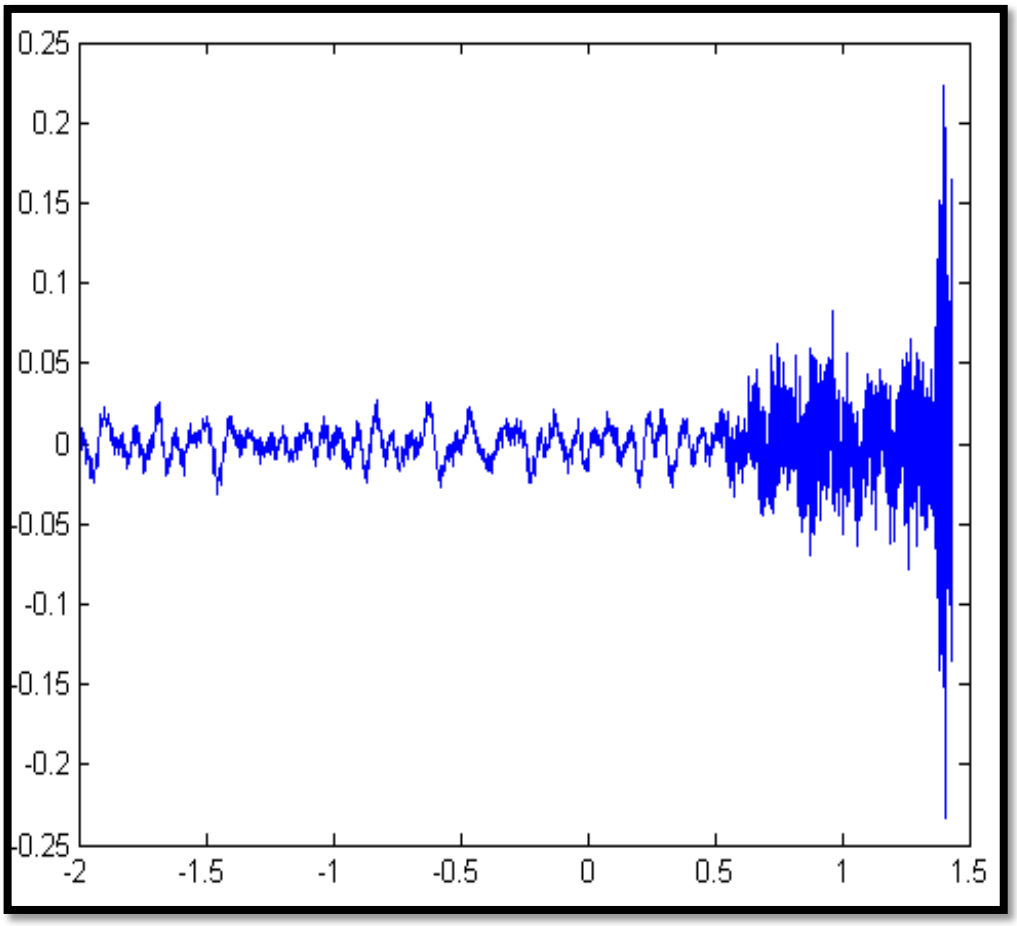
Ahora, comenzamos la adquisición. Después de llamar a la función de inicio, el objeto va a correr, pero no hay registro de datos se producirá hasta que la función del trigger se ejecute. A medida que el temporizador de devolución de llamada se ejecuta y evalúa los datos, los mensajes se mostrarán indicando lo que se ha detectado.

```
start(AI);
   % Wait for trigger to occur and the acquisition to end.
   % Set the timeout to a sufficiently large value.

   wait(AI, 60 * extraData.secondsOfQuietNeeded);

   % If the acquisition ended successfully, plot the available data.

   samplesAvailable = getAIi,'SamplesAvailable');
   [data, time] = getdata(AI, samplesAvailable);
   plot(time,data);


catch
   % If the wait timed out, we end up here.
   % Display a message and stop the acquisition.
   disp(sprintf('Could not find quiet period of %d seconds.', ...
```

```
            extraData.secondsOfQuietNeeded));
    stop(ai);

end
```

Advertencia: El número de muestras solicitada no está disponible. El número de muestras devueltas serán reducidas.

La primera señal encontrada. Esperando para un periodo quiet de 2 segundos. Período de quiet encontrado. Inicio de sesión iniciado.



Grafica 47. Medición de ruido.

14.7 LIMPIAR

Cuando haya terminado, se borra el objeto y limpiamos la variable del espacio de trabajo para liberar recursos del sistema.

```
delete(AI);
clear AI;
```

14.8 La función de devolución de llamado.

Usamos la función de devolución de llamado del temporizador para examinar los datos entrantes, para ver si nuestra condición de trigger se ha cumplido. Cuando nuestra condición de trigger se ha cumplido, la función del disparador está llamado a iniciar el registro.

```
type TimerCallback;
%%
% La adquisicion de datos de la toolbox llama al tiempo con la función callback
% con el objeto y evento..
function TimerCallback(ai,event)

% Primero, se deve tener la informacion necesaria para evaluar la condición %del trigger.
extraData = get(ai, 'UserData');
timerPeriod = get(ai,'TimerPeriod');
sampleRate = get(ai,'SampleRate');

% Entonces, podemos observer los datos recividos durante el period. sampleData = peekdata(ai, timerPeriod * sampleRate);
maxValue = max(sampleData);
```

```
minValue = min(sampleData);

% Luego, analizamos los datos recividos ypara asi determiner la condicion de % trigger.
% Si no se encuentra la primera señal. Observemos que es por esto

if ( ~extraData.signalFound )

   if ( maxValue > extraData.signalVoltage || minValue < -extraData.signalVoltage )

      % La primera señal esta siendo encontrada.
      extraData.signalFound = true;
      fprintf('First signal found. Waiting for quiet period of %d seconds.\n', ...
            extraData.secondsOfQuietNeeded);

   end

% Si una señal esta siendo encontrada, la observamos porque ocrre un periodo de quietud.

elseif ( ~extraData.quietFound )
   if ( minValue > -extraData.quietVoltage && maxValue < extraData.quietVoltage )

      % Incremento de los segundos de quietud que hemos visto

      extraData.secondsOfQuietFound = extraData.secondsOfQuietFound + timerPeriod;

      if extraData.secondsOfQuietFound >= extraData.secondsOfQuietNeeded

         % Cuando se encutra el deseado period, ejecutamos el trigger
```

```
.
            trigger(ai);
            extraData.quietFound = true;
            disp('Quiet period found. Logging started.');

        end

    %.Borramos los segundos de quietud que hemos visto.

    else
        extraData.secondsOfQuietFound = 0;
    end

% Si el period de quietud esta siendo encontrado, observamos la segunda señal.
 else
    if ( maxValue > extraData.signalVoltage || minValue < -extraData.signalVoltage )

        %Cuando la segunda señal es encontrada se para la adquisición.

        stop(ai);
        disp('Second signal found. Acquisition stopped.');
    end
end

%%
% Por último, se guarda la información actualizada a la propiedad UserData de
modo que % estará disponible para la siguiente ejecución  de la devolución de
llamada del %temporizador

set(ai, 'UserData', extraData);
```

15 EXPERIENCIA PARA EL REGISTRO DE DATOS EN EL DISCO.

En esta experiencia se ilustra cómo configurar un objeto de entrada analógica para registro de datos y se muestra un ejemplo para el registro de múltiples triggers en el disco.

Se dan ejemplos de cómo DAQREAD se puede utilizar para leer todos los datos registrados en el disco, una parte de los datos registrados en el disco, la información del evento que tiene lugar cuando el objeto se está registrando en el disco y el estado del objeto de adquisición de datos al iniciar la sesión y cuando se haya terminado.

En primer lugar, encuentre objetos que se estén ejecutando dentro de la toolbox y deténgalos. Así se evita que algún objeto que se esté ejecutando interfiera con esta demo. Este código no suele ser necesario fuera de esta demo a menos que haya varios objetos de adquisición de datos en ejecución.

```
if (~isempty(daqfind))
    stop(daqfind)
end
```

Para empezar, se crea el objeto de analógica que contiene un canal.

```
AI = analoginput('nidaq','Dev3');

 addchannel(AI,1);
```

Tenga en cuenta que hay cuatro propiedades que controlan la cantidad de datos que se registra en el disco:

```
|Logging| -- type |daqhelp Logging| for more info
   |LogFileName| -- type |daqhelp LogFileName| for more info
   |LoggingMode| -- type |daqhelp LoggingMode| for more info
```

|LogToDiskMode| -- type |daqhelp LogToDiskMode| for more info

Con los siguientes parámetros, los datos se registran en un archivo temporal y la memoria. Si se produce otro start, los nuevos datos se sobreponen a los datos antiguos en el archivo.

```
AI.LogFileName = tempname();
AI.LogToDiskMode = 'overwrite';
AI.LoggingMode = 'Disk&Memory';
```

Vamos a configurar el objeto de entrada análoga para que se ejecuten cinco disparos consecutivamente. Cada disparo recogerá 1000 muestras de datos utilizando una frecuencia de muestreo de 5.000 muestras por segundo.

```
AI.TriggerRepeat = 4;

AIi.SamplesPerTrigger = 1000;

AI.SamplesPerTrigger = 1000;

AI.SampleRate = 5000
```

Display Summary of Analog Input (AI) Object Using 'USB-6008'.

```
Acquisition Parameters:  5000 samples per second on each channel.
                         1000 samples per trigger on each channel.
                         1 sec. of data to be logged upon START.
                         Log data to 'Disk&Memory' on trigger.
                         'Overwrite' data to file on trigger.
```

Trigger Parameters: 5 'Immediate' trigger(s) on START.

Engine status: Waiting for START.

0 samples acquired since starting.

0 samples available for GETDATA.

AI object contains channel(s):

Index: ChannelName: HwChannel: InputRange: SensorRange: UnitsRange: Units:

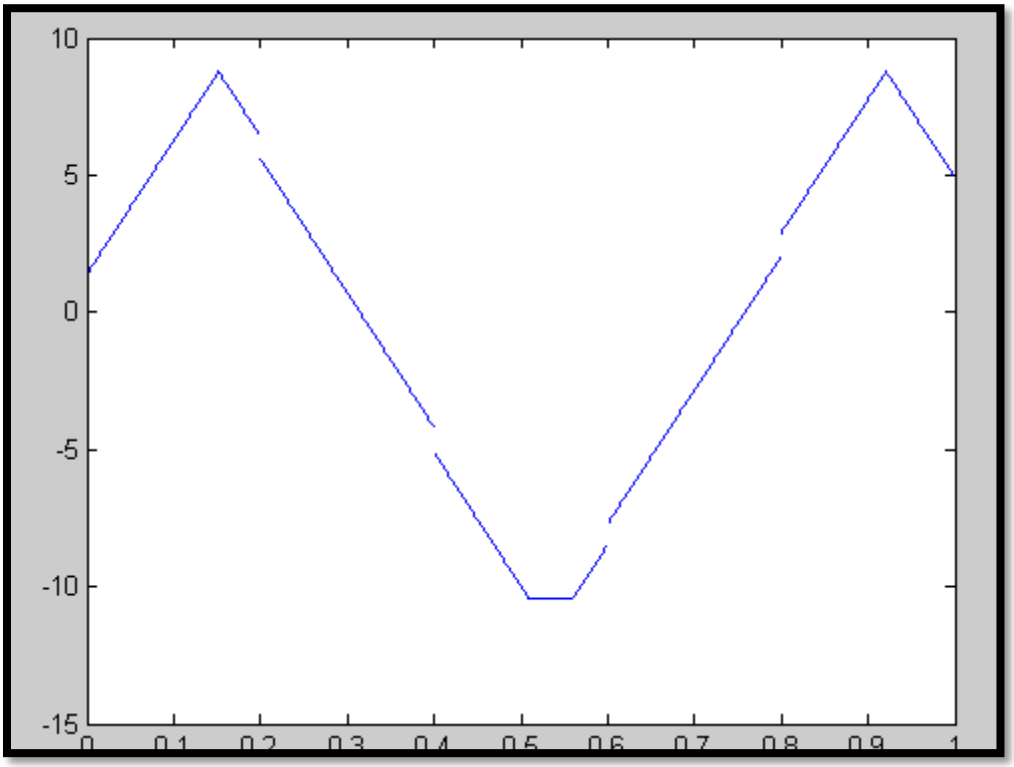
1 '' 1 [-10 10] [-10 10] [-10 10] 'Volts'

Se utilizo la propiedad TriggerType 'inmediate', la propiedad hace que se ejecute inmediatamente y que se ponga en 'On' después de que el objeto de entrada analógica se inicia con el comando START. Por lo tanto, los datos serán enviados de inmediato el archivo de registro al ejecutar el comando START. Luego, se espera hasta por 2 segundos para que la adquisición sea completa.

```
start(AI);
AI.logging
wait(AI,2);

ans =
On
```

El comando DAQREAD se utiliza para leer el archivo daq file. Con la siguiente sintaxis, todas las muestras se registran en el disco se lee en los datos variables. Los datos serán una matriz m-por-n donde m es el número de muestras registradas y n es el número de canales. Sin embargo, si se ha identificado múltiples factores desencadenantes, los datos entre los factores desencadenantes están separados por n por m. En este caso, m incrementará el número de factores desencadenantes. El vector tiempo contendrá los tiempos relativos que cada muestra de datos se registra con respecto a la primera activación.

```
[data, time] = daqread(AI.LogFileName);
% Ahora, los datos se pueden mostrar:

plot(time,data);
```



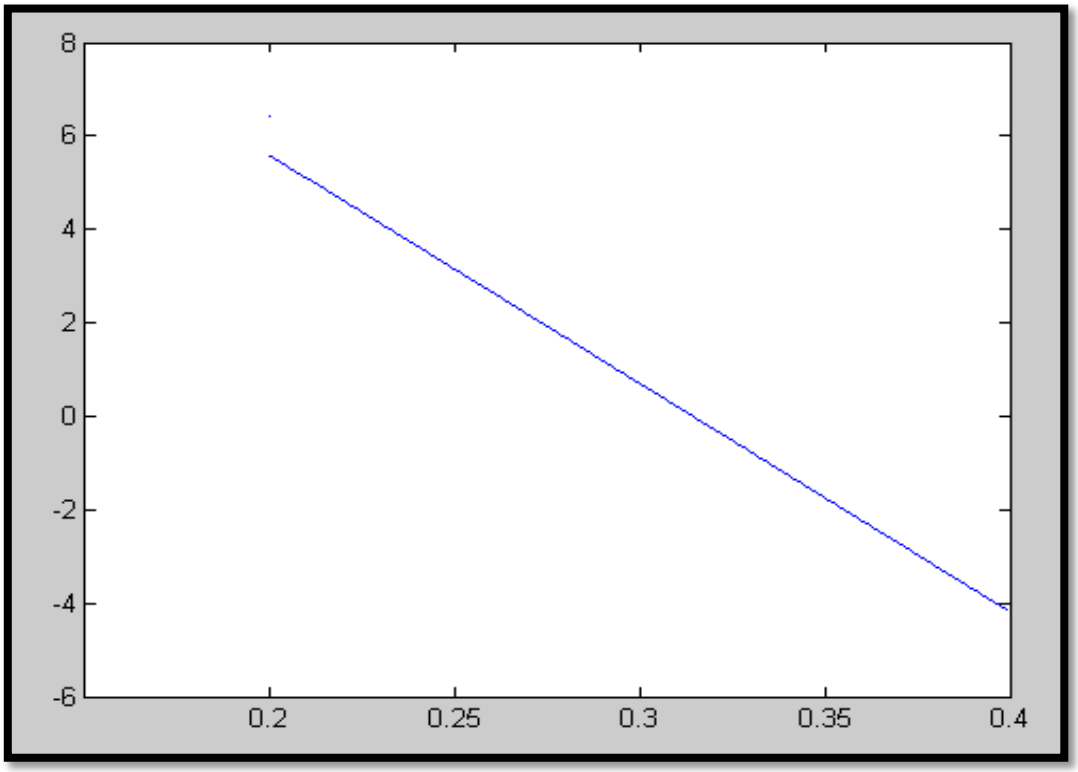
Grafica 48. Toma de datos utilizando DAQREAD.

Esto es posible para limitar la cantidad de datos leídos por DAQREAD utilizando una de las siguientes propiedades:

Samples	Especifica el rango de las muestras
Triggers	Especifica el rango de los triggers
Time	Especifica el rango de tiempo en segundos
Channels	Especifica el indice del canal.

Por ejemplo, el siguiente comando leerá todas las muestras entre la muestra de datos 1000 y la muestra de datos 2000.

```
[data,time] = daqread(AI.LogFileName, 'Samples', [1000 2000]);

% El subconjunto de datos se muestran a continuación.
plot(time,data);
```

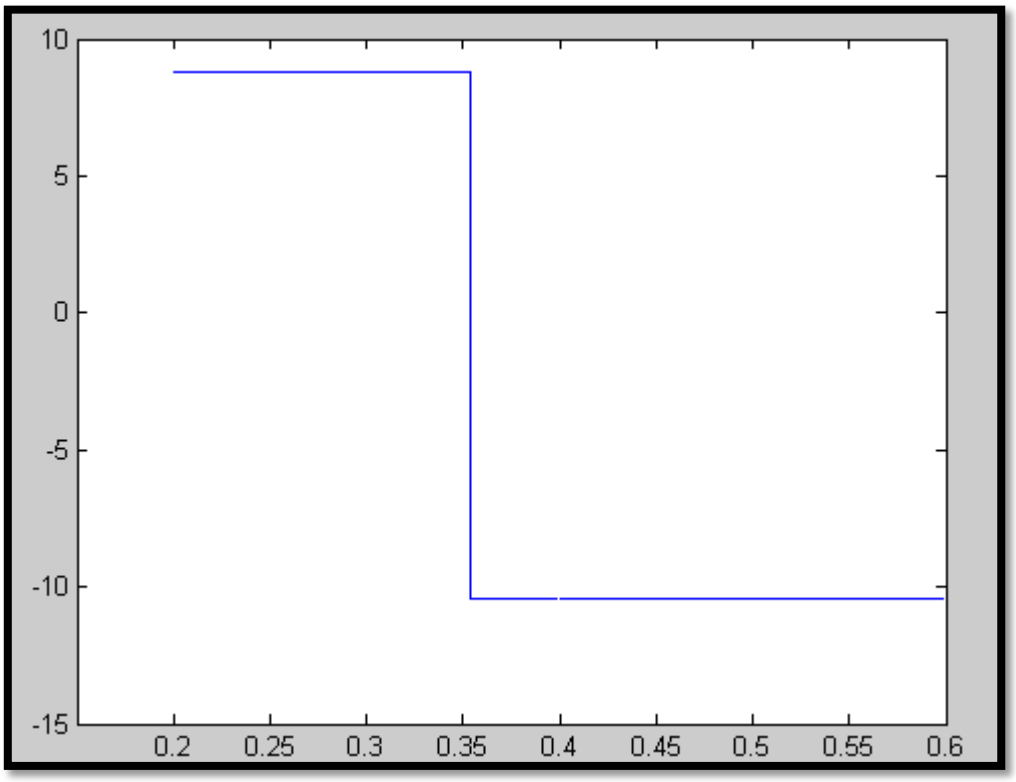


Grafica 49. Toma de datos utilizando DAQREAD entre las muestras 1000 y 2000.

Del mismo modo, usted puede reducir el número de canales de lectura mediante la especificación de la propiedad 'Channels'. Por ejemplo, para leer las muestras

500-1000 del segundo canal, debería asignar los siguientes valores para las muestras y propiedades de Canales.

```
[data,time] = daqread(AI.LogFileName, 'Samples', [500 1000], 'Channels', 2);

% El subconjunto de datos se muestran a continuación.
plot(time,data);
```



Grafica 50. Toma de datos de un solo canal

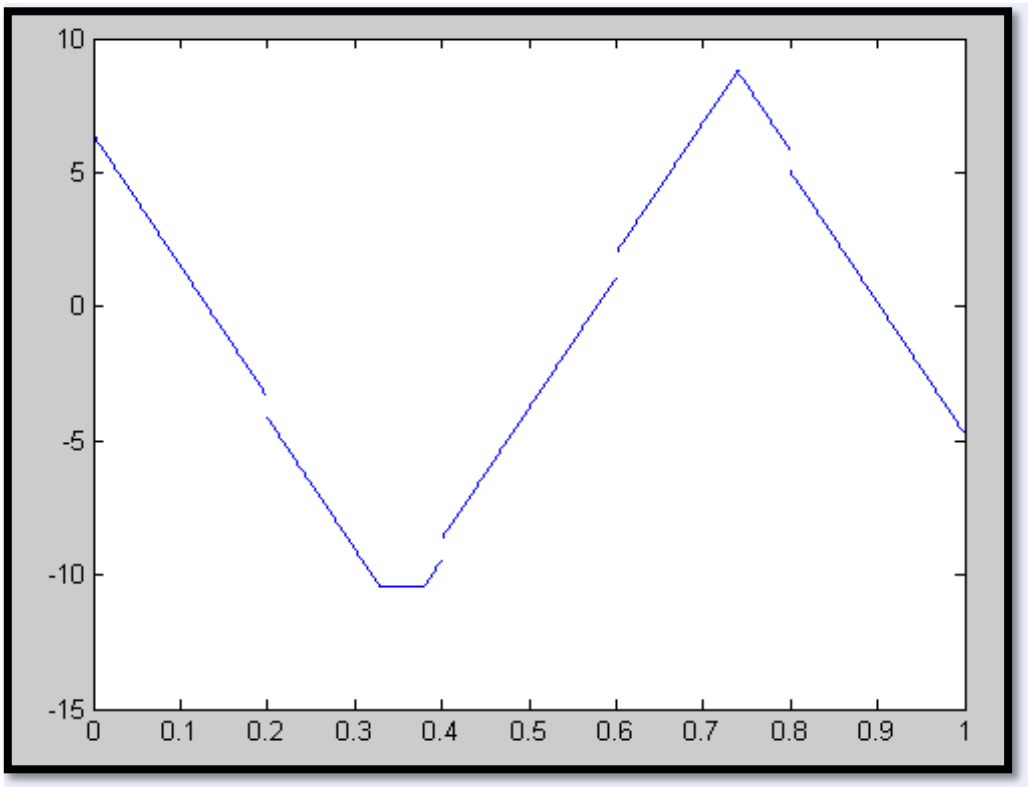
Se puede seleccionar el formato de los datos y variables de tiempo de salida con las siguientes propiedades en donde se le da el valor por defecto en llaves ().

DataFormat {double} | native

TimeFormat {vector} | matrix

Por ejemplo, para leer todos los datos en formato nativo para el canal 1 solamente, podría asignar el siguiente valor para las propiedades 'Channels' y ' DataFormat'.

```
[data,time] = daqread(AI.LogFileName, 'Channels', 1, 'DataFormat', 'native');

% El subconjunto de datos se muestran a continuación.

plot(time,data);
```



Grafica 51. Grafico en formato nativo del canal 1.

También puede utilizar el comando DAQREAD para leer la información relacionada del objeto utilizado para registrar los datos y la información del hardware.

```
daqinfo = daqread(AI.LogFileName, 'info');
```

Para obtener la información de los objetos:

daqinfo.ObjInfo

ans =

BufferingConfig: [256 20]

BufferingMode: 'Auto'

Channel: [1x2 struct]

ChannelSkew: 2.0833e-005

ChannelSkewMode: 'Minimum'

ClockSource: 'Internal'

EventLog: [1x7 struct]

ExternalSampleClockDriveLine: 'None'

ExternalSampleClockSource: 'PFI2'

ExternalScanClockDriveLine: 'None'

ExternalScanClockSource: 'PFI7'

ExternalTriggerDriveLine: 'None'

HwDigitalTriggerSource: 'PFI0'

InitialTriggerTime: [1x6 double]

InputType: 'Differential'

```
        LogFileName: [1x73 char]
            Logging: 'Off'
        LoggingMode: 'Disk&Memory'
      LogToDiskMode: 'Overwrite'
   ManualTriggerHwOn: 'Start'
               Name: 'nidaqmxDev3-AI'
            Running: 'Off'
         SampleRate: 5000
    SamplesAcquired: 5000
   SamplesAvailable: 5000
   SamplesPerTrigger: 1000
                Tag: ''
            Timeout: 1
        TimerPeriod: 0.1000
     TriggerChannel: []
    TriggerCondition: 'None'
TriggerConditionValue: 0
       TriggerDelay: 0
  TriggerDelayUnits: 'Seconds'
      TriggerRepeat: 4
    TriggersExecuted: 5
```

TriggerType: 'Immediate'

Type: 'Analog Input'

Para obtener información del canal:

daqinfo.ObjInfo.Channel

ans =

1x2 struct array with fields:

ChannelName

HwChannel

Index

InputRange

NativeOffset

NativeScaling

SensorRange

Type

Units

UnitsRange

Para obtener información sobre el evento puede utilizar el comando SHOWDAQEVENTS para mostrar la información del evento.

aqinfo.ObjInfo.EventLog;

```
showdaqevents(daqinfo.ObjInfo.EventLog)

  1 Start              ( 15:30:52, 0 )
  2 Trigger#1          ( 15:30:52, 0 )        Channel: N/A
  3 Trigger#2          ( 15:30:52, 1000 )     Channel: N/A
  4 Trigger#3          ( 15:30:53, 2000 )     Channel: N/A
  5 Trigger#4          ( 15:30:53, 3000 )     Channel: N/A
  6 Trigger#5          ( 15:30:53, 4000 )     Channel: N/A
  7 Stop               ( 15:30:53, 5000 )
```

Para obtener la información de hardware:

```
daqinfo.HwInfo

ans =

              AdaptorName: 'nidaqmx'
                     Bits: 12
                 Coupling: {'DC'}
               DeviceName: 'USB-6008'
          DifferentialIDs: [0 1 2 3]
```

```
                Gains: [1 2 4 5 8 10 16 20]
                   ID: 'Dev3'
           InputRanges: [8x2 double]
         MaxSampleRate: 10000
         MinSampleRate: 0.6000
        NativeDataType: 'double'
              Polarity: {'Bipolar'}
            SampleType: 'Scanning'
        SingleEndedIDs: [0 1 2 3 4 5 6 7]
         SubsystemType: 'AnalogInput'
         TotalChannels: 8
VendorDriverDescription: [1x35 char]
   VendorDriverVersion: '8.7'
```

Por último, limpiar el objeto de la adquisición.

```
delete(AI.LogFileName)
delete(AI);
```

PARTE IV
PRACTICAS DE LABORATORIO

16 PRACTICAS DE LABORATORIO

16.1 PRÁCTICA # 1. SISTEMA DE ADQUISICIÓN DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB, ENTRADA ANÁLOGA.

- **MARCO TEÓRICO 1ª PARTE. SISTEMA DE ADQUISICIÓN DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB.**

Descripción general.

El objetivo de cualquier sistema de adquisición de datos es proporcionar las herramientas y recursos necesarios para tomar señales físicas y convertirlas en datos que posteriormente se puedan procesar y mostrar. Un sistema de adquisición de datos típico consta de estos componentes:

- **Hardware de Adquisición:** Es el corazón de cualquier sistema de adquisición de datos. La función principal es hacer la conversión de señales análogas a señales digitales y señales digitales a análogas. Conversión A/D y D/A
- **Sensores y Actuadores (Transductores):** Un transductor es un dispositivo que convierte un tipo de energía de entrada en una energía de salida de otra forma.
- **Acondicionador de señal:** Las señales de los sensores a menudo son incompatibles con el hardware de adquisición de datos, y para superar esto las señales deben ser acondicionadas. Por ejemplo, las señales podrían ser amplificadas o volverlas en señales sin componentes de frecuencias indeseadas. Las señales de salida también pueden ser acondicionadas.
- **Computador:** Proporciona un procesador, un sistema de reloj, un bus de datos, memoria y espacio en el disco para almacenar datos.

- **Software:** Permite el intercambio de información entre el computador y el hardware.

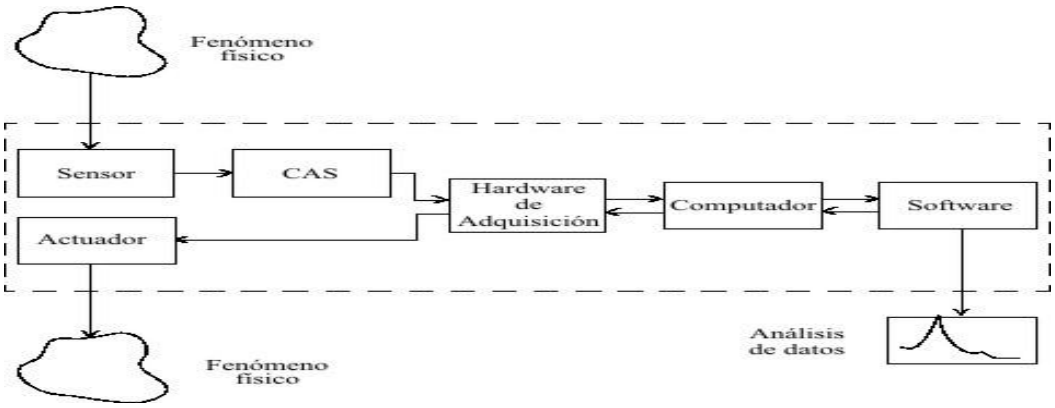


Figura 1.

Relación entre los componentes de un sistema de adquisición de datos.

1. Hardware de adquisición de datos.

Se caracteriza por los subsistemas que este posee. Un subsistema es un componente del hardware que realiza una tarea específica. Los más comunes son:

a) Entradas análogas

b) Salidas análogas

c) Entradas/salidas digitales

d) Contador/Temporizador

1.1 Subsistemas del hardware.

Un Hardware consta de múltiples subsistemas como se muestra a continuación, y son llamados *tarjetas multifuncionales.*

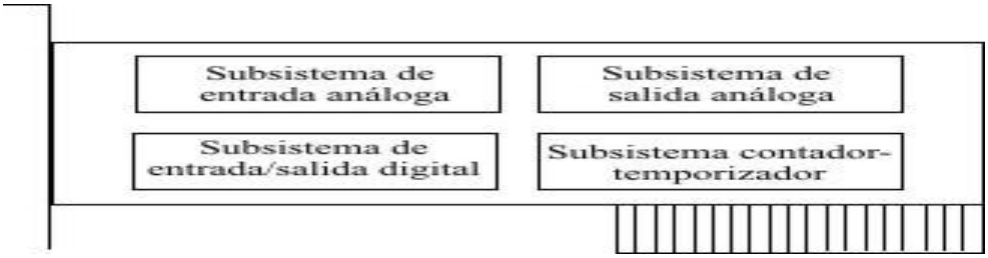


Figura 2.

Hardware de Adquisición de Datos.

1.1.1 Subsistemas de entrada análoga.

Convierte las señales análogas reales desde el sensor en señales lógicas (bits) que pueden ser leídas por el PC. Tal vez es el más importante de los subsistemas disponibles, comúnmente son dispositivos multicanales con una resolución de 12 o 16 bits. El **rango de entrada** puede ser cambiado por la selección de un valor de ganancia diferente y se debe configurar para que coincida con la **polaridad**. La **tasa de muestreo** puede ser igual o dos veces más grande que el mayor componente de frecuencia de la señal análoga. Una frecuencia de la mitad de la tasa de muestreo es llamada como frecuencia de Nyquist. La **configuración de canal** puede darse de Modo Diferencial o de Modo de Simple nodo. Para hacer mediciones de alta calidad se debe seguir estas reglas:

- Maximizar la precisión y exactitud.
- Minimizar el ruido.
- Comparar el rango del sensor de acuerdo al rango del A / D.

Nota: Con la Toolbox ™ de adquisición de datos no se tiene que especificar el rango y la ganancia. En lugar de esto, simplemente se especifica el rango de entrada deseado.

1.1.2 Subsistemas de salida análoga.

Convierte datos digitales almacenados en el computador en una señal análoga real. Normalmente los hardwares de adquisición de datos ofrecen dos canales de salida análoga. Estos subsistemas son llamados también como subsistemas AO, convertidores D/A o DAC's.

1.1.3 Subsistemas de entradas/salidas digitales.

Están diseñados para valores de entradas y salidas digitales (valores lógicos), estos valores suelen ser manejados ya sea como bits simples, líneas de bits o como un puerto que normalmente consta de ocho líneas de bits. Si bien la mayoría de las tarjetas incluyen algunas capacidades de entradas y salidas digitales, están usualmente limitadas a operaciones simples y muchas veces es necesario un hardware especial para realizar operaciones avanzadas de entradas y salidas digitales.

1.1.4 Subsistemas de contador/temporizador.

Son usados para eventos de conteo, medición de frecuencia, periodo y para la generación de tren de pulsos. Siempre se debe usar el reloj de la tarjeta cuando la tasa de muestreo es alta y cuando se necesite un intervalo de tiempo fijo entre muestras.

1.2 La tarjeta NI USB - 6008/6009.

- 8 canales de entradas analógicas (AI)
- 2 canales de salidas analógicas (AO)
- 12 canales de entrada/salida (DIO)
- Un contador de 32-bit con interface USB de total velocidad.
- USB 6008 10000 muestras/s.
- USB 6009 48000 muestras/s.

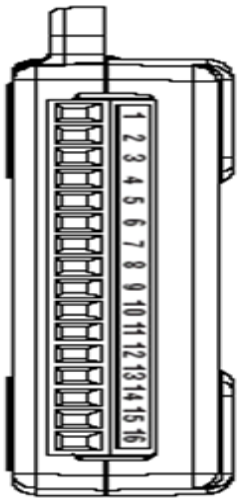
Module	Terminal	Signal, Single-Ended Mode	Signal, Differential Mode
	1	GND	GND
	2	AI 0	AI 0+
	3	AI 4	AI 0–
	4	GND	GND
	5	AI 1	AI 1+
	6	AI 5	AI 1–
	7	GND	GND
	8	AI 2	AI 2+
	9	AI 6	AI 2–
	10	GND	GND
	11	AI 3	AI 3+
	12	AI 7	AI 3–
	13	GND	GND
	14	AO 0	AO 0
	15	AO 1	AO 1
	16	GND	GND

Figura 4.

Puerto análogo.

Características: Resolución de 12 bits, convertidor de aproximación sucesiva, rango de salidas de 0V a +5 V, impedancia de salida de 50 Ω y corriente en corto circuito de 50 mA. El máximo voltaje sobre cualquier pin es ±10 V respecto a GND.

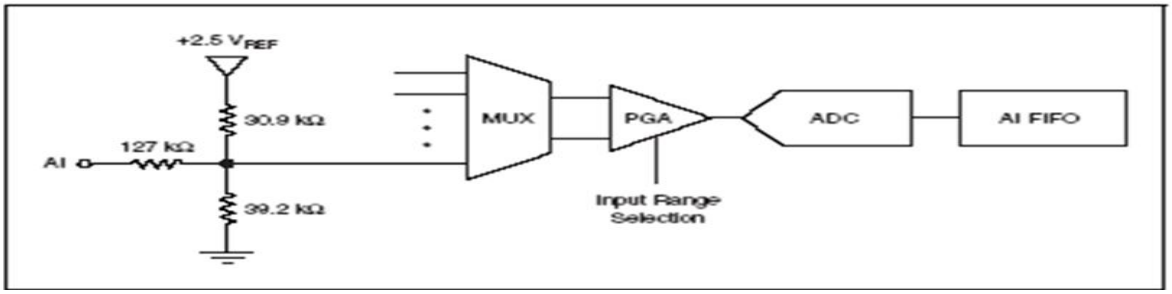


Figura 5.

Circuito de entrada análoga.

MUX: El multiplexor en ruta un canal AI por vez a la PGA.

PGA: Provee ganancias de entrada de 1, 2, 4, 5, 8, 10, 16, o 20 cuando se configuran para medición diferencial y ganancia 1 cuando se configuran para

entrada sencilla. La ganancia del PGA es automáticamente calculada según el rango de voltaje seleccionado en la aplicación

A/D convertidor: El convertidor (ADC) digitaliza la señal AI para convertir el voltaje análogo en un código digital

AI FIFO: Puede hacerse conversiones sencillas y múltiples A/D de número finito o continuo de muestras. Un búfer (FIFO) mantiene los datos durante la adquisición para que no se pierdan.

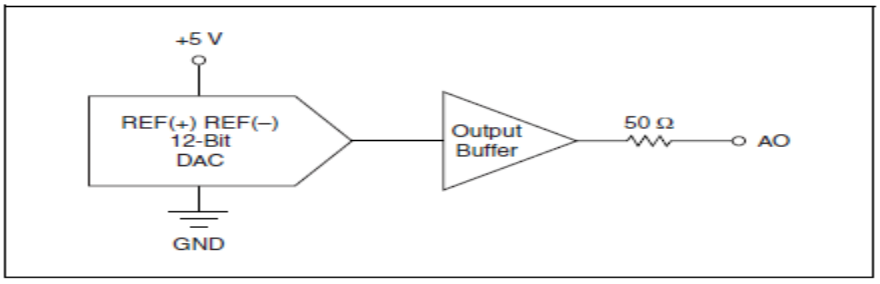


Figura 6.

Circuito de salida análoga.

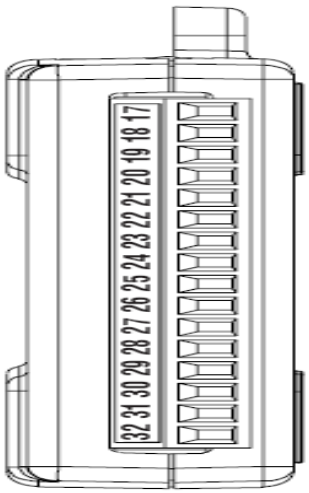
Module	Terminal	Signal
	17	P0.0
	18	P0.1
	19	P0.2
	20	P0.3
	21	P0.4
	22	P0.5
	23	P0 6
	24	P0.7
	25	P1.0
	26	P1.1
	27	P1.2
	28	P1.3
	29	PFI 0
	30	+2.5 V
	31	+5 V
	32	GND

Figura 7.

Puerto digital.

Características: 12 líneas digitales P0 a 07 y P1 a 03 (puerto DIO), GND es la referencia de tierra para la señal del puerto DIO, Puede programarse individualmente todas las líneas como entradas o salidas, control direccional de entrada o salida, compatibilidad TTL, LVTTL, CMOS, rango máximo de voltaje -0.5V a 5.8 V con respecto a GND y resistencia de 47 kΩ a5 V.

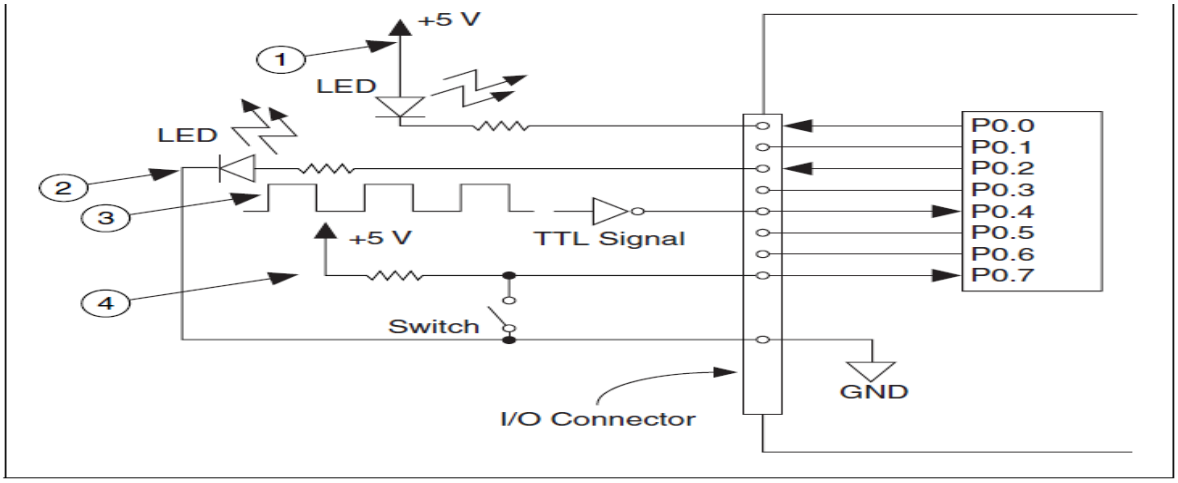


Figura 8.

Ejemplo de conexión a puerto digital.

Conexion de circuito I/O digital: Se muestra P0<0,7> configurado con entradas y salidas digitales, se puede configurar P1<03> similarmente.

Contador: Resolución de 32 bits, conteo de pulsos (pulso de bajada), resistencia de 47 KΩ a 5 V, máxima frecuencia de entrada de 5 MHz, mínimo ancho de pulso positivo de 100 ns, mínimo ancho de pulso negativo de 100 ns, alto voltaje de entrada de 20 V y bajo voltaje de entrada de 0,8 V.

Fuente de energía: +5 V**,** 200 mA, esta fuente puede usarse para energizar componentes externos.

Referencia externa: 2.5V, tiene una fuente de voltaje de referencia de alta pureza para el ADC usando un circuito regulador multiestado, amplificador y circuito filtro.

2. Tipos de software.

a) Controladores.

b) Software de aplicación.

Al usar la Toolbox de Adquisición de Datos con la tarjeta USB6008/6009 de National Instruments está asociada con el controlador NI-DAQmx y el software de aplicación Toolbox de Adquisición de Datos y Matlab. La relación entre el usuario, el software controlador, el software de aplicación y el hardware es así:

Figura 9.

Relación usuario, software y hardware.

2.1 Controladores.

El controlador NI-DAQmx y los controladores en general permiten el acceso y el control de las capacidades del hardware. Entre otras cosas, los controladores básicos permiten:

- Traer datos y obtener datos desde fuera de la tarjeta.
- Control de la tasa a la cual los datos son adquiridos.
- Integrar el hardware de adquisición de datos con los recursos del computador.
- Integrar el hardware de adquisición de datos con el hardware de acondicionamiento de señales.
- Acceso múltiple a subsistemas de la tarjeta
- Acceso múltiple a módulos o tarjetas de adquisición de datos

2.2 Software de aplicación.

Proporciona una interfaz al controlador. El software de aplicación básico permite:

- Reporte pertinente sobre la información tal como el número de muestras adquiridas.
- Generar eventos
- Administrar los datos almacenados en la memoria del computador
- Acondicionar una señal
- Graficar los datos adquiridos

Toolbox de Adquisición de Datos.

MATLAB _ **Mat**rix **Lab**oratory es un conjunto de comandos o funciones que realizan tareas específicas, además dispone de un código básico y más de 30 librerías especializadas llamadas "Toolboxes" las cuales trabajan en áreas especiales como: adquisición de datos, control, comunicaciones, procesamiento de señales, etc. Ver figura 10.

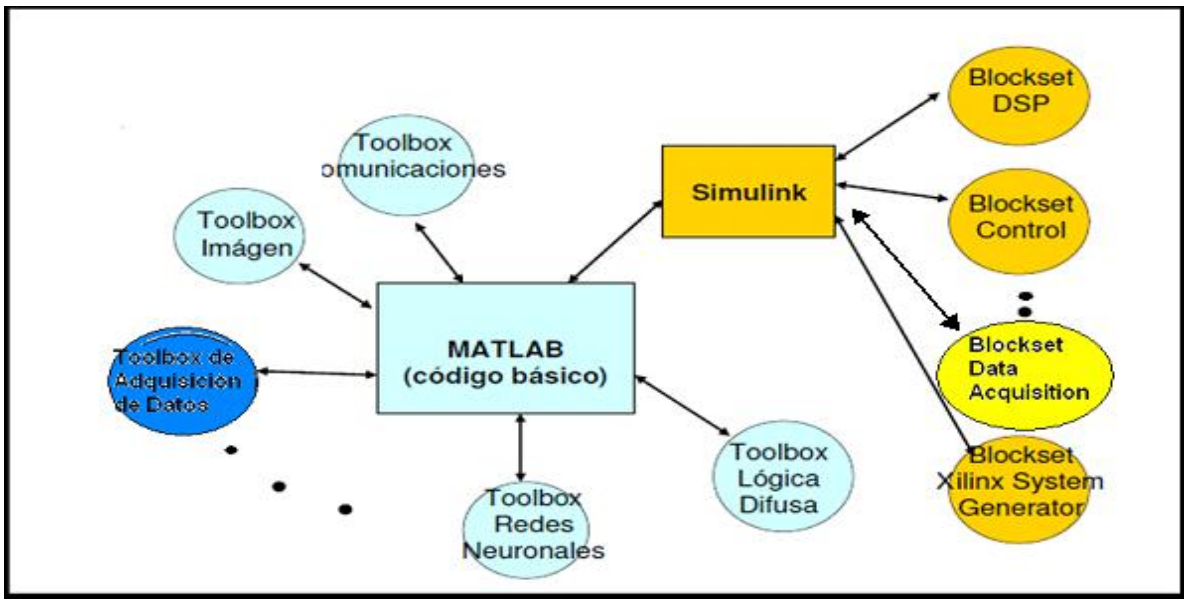


Figura 10.

Estructura organizativa de Matlab

2.2.1 Características.

La Toolbox de adquisición de datos tiene las siguientes características:

- Un marco para traer datos reales tomados al espacio de trabajo de MATLAB usando un hardware de adquisición de datos compatible con el PC.
- Soporte para entradas análogas (AI), salidas análogas (AO) y entradas y salidas digitales (DIO). Incluyendo conversiones simultáneas para entradas y salidas análogas.
- Soporte para estos tipos de hardware más populares:

  - Advantech®. Tarjetas que usen el administrador de dispositivos de Advantech.
  - Módulos VXI E1432A/33A/34A de Agilent Technologies®.
  - Keithley®. Tarjetas que usen DriverLINX.
  - Tarjetas de Measurement Computing™ Corporation.
  - National Instruments®. Tarjetas que usan el software tradicional NI-DAQ o NI-DAQmx.
  - Puertos paralelos LPT1 y LPT3.
  - Microsoft® Windows®. Tarjetas de sonido.

2.2.2 Componentes.

La Toolbox de adquisición de datos de MATLAB se divide en tres componentes principales:

· Las funciones M establecidas

· El motor de adquisición de datos

· Los manejadores (drivers) de la tarjeta de adquisición de datos.

Como se muestra en la figura 11, estos componentes permiten intercambiar información entre Matlab y el hardware de adquisición de datos.

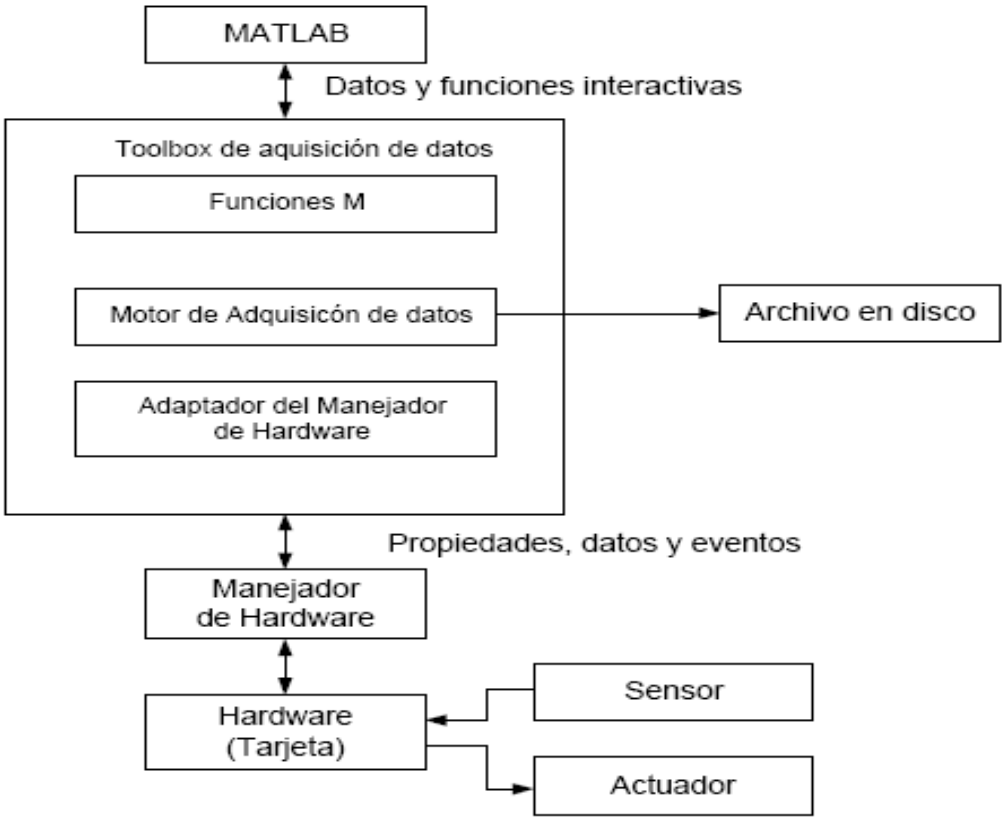


Figura 11.

Componentes de la Toolbox de adquisición de datos.

2.2.2.1 Propiedades.

Con el manejo de las propiedades es posible controlar el comportamiento de la aplicación. Las propiedades contienen información sobre la configuración del hardware.

2.2.2.2 Datos.

Pueden ser datos provenientes de un sensor conectado a un subsistema de entrada analógica para ser almacenados en Matlab o también pueden ser datos de salida de Matlab a un actuador conectado a un subsistema de salida analógica.

2.2.2.3 Eventos.

Un evento ocurre en un instante particular después de que una condición es cumplida y puede resultar en uno o más llamados como esté especificado. Los eventos pueden ser generados solo después de que se configuren las

propiedades asociadas. Algunas de las maneras en las que se puede usar los eventos incluye la iniciación del análisis después de que una predeterminada cantidad de datos sea adquirida, o mostrando un mensaje al espacio de trabajo de MATLAB después de que un error ocurra.

2.2.2.4 Funciones M.

Para ejecutar cualquier tarea con la aplicación de adquisición de datos, deben llamarse algunas funciones M. Entre otras cosas, estas funciones permiten:

· Crear dispositivos de objetos que proporcionan un camino de Matlab al hardware y permite controlar el comportamiento de la aplicación

· Capturar datos o sacar datos

· Configurar las propiedades

· Evaluar el estado y los recursos del hardware de adquisición

2.2.2.5 El motor de adquisición de datos.

Es una librería de enlace dinámico (.dll) en forma de archivo MEX que:

· Guarda los dispositivos de objetos y sus valores asociados de configuración que controlan la aplicación de adquisición de datos.

· Controla la sincronización de eventos.

· Controla el almacenaje de datos capturados o en espera de ser sacados. Mientras el motor ejecuta estas tareas, puede usarse Matlab para ejecutar otras tareas como el análisis de los datos adquiridos. En otras palabras, el motor y Matlab son asíncronos.

2.2.2.6 Adaptadores de Hardware.

El adaptador es la interfaz entre el motor de adquisición de datos y el controlador del hardware. El propósito principal del adaptador es pasar información entre MATLAB y el hardware por medio de sus controladores.

Para adquirir datos usando una tarjeta de National Instruments, la versión apropiada del driver NI-DAQ debe ser instalada en su plataforma.

Fabricante	Nombre del adaptador
National Instruments	nidaq
ComputerBoards	cbi
Agilent Technologies	hpe1432
Tarjetas de sonido para Windows	winsound

Tabla 1.

Adaptadores para algunos fabricantes.

2.2.3 Algunas funciones de ayuda e información.

- Para ver una lista de funciones disponibles en el equipo se usa:
  help daq

- Se puede ver el código para alguna función del equipo usando:
  type nombre_de_la_función

- Se puede ver la ayuda para alguna función con:
  daqhelp nombre_de_la_función

- Se puede ver la ayuda para alguna propiedad con:
  daqhelp(objeto, 'nombre_de_la_propiedad').
- Mediante la función “daqhwinfo” podemos obtener:
  Versión del Toolbox y de Matlab, nombre del adaptador, nombre de la tarjeta, presenta información acerca de un objeto relacionado a un dispositivo específico listando los subsistemas soportados por la tarjeta y la sintaxis para crear objetos asociados con un subsistema dado. Para indagar sobre la tarjeta USB 6008/6009 hw=daqhwinfo('nidaq').

2.2.4 Osciloscopio.

La Toolbox de adquisición de datos cuenta con un osciloscopio exclusivo para entrada análoga denominado Oscilloscope.

Este, también denominado Softscope usa la interface (GUI) en una gráfica interactiva que muestra en pantalla los datos en tiempo de ejecución “tiempo real”.

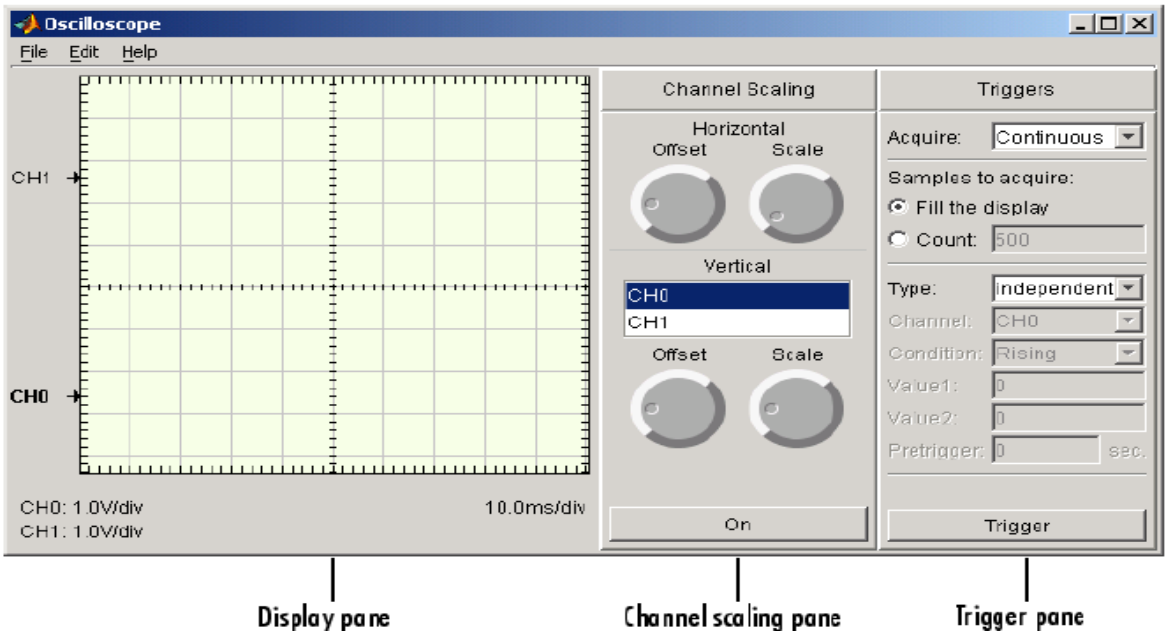


Figura 12.

Oscilloscope.

Como observamos en la figura 12, el Oscilloscope presenta un display, un recuadro para cada canal adicionado del hardware, perillas para offset y escala horizontal y vertical y un panel para el trigger.

- Para abrir el Oscilloscope a una entrada ai, se llama la función *Softscope(ai)*.

- La configuración del hardware se hace desde el banner Edit con la secuencia Edit>Hardware y para dar una nueva velocidad de muestreo Edit>Sample Rate.

- Para crear displays adicionales se sigue la secuencia Edit>Scope>Scope asignando el nombre al nuevo display en label y para asignar la información

de un canal en particular al display se sigue Edit>Channel>Channel Display luego de nombrar el nuevo display se da clic sobre Add, Apply y Ok. El clic derecho sobre el display configura la auto escala.

- Para ingresar a un display una señal de referencia que se ha debido definir en el workspace de Matlab se sigue Edit>Channel>Channel y para realizar operaciones matemáticas entre canales se sigue Edit>Channel>Channel.

- Existen 3 tipos de trigger:
  One shot.- adquiere el número de muestras especificado, una vez.
  Continuous.- continuamente adquiere el número de muestras especificadas.
  Sequence.- continuamente adquiere el número de muestras especificadas y usa el trigger dependiente cada vez.

- Para realizar una medición predeterminada se da clic derecho sobre el Oscilloscope y para predefinir un nuevo tipo de medición se sigue Edit>Measurement>Measurement Type. Se mostrara un nuevo panel de mediciones. Con el clic izquierdo sobre un punto de alguna curva se observará su valor específico.

- Para exportar los datos al workspace se sigue File>Export>Channels.

❖ MARCO TEÓRICO 2ª PARTE. ENTRADA ANÁLOGA.

➢ Transferencia de datos desde el hardware a la memoria del sistema.

La transferencia de datos adquiridos desde el hardware a la memoria del sistema o memoria del motor de adquisición de datos, sigue estos pasos:

1. Los datos adquiridos se almacenan en el buffer de primera entrada - primera salida (FIFO) del hardware.

2. Los datos se transfieren desde el búfer FIFO a la memoria del sistema utilizando interrupciones o DMA.

Estos pasos ocurren de forma automática. Normalmente, todo lo que requiere es alguna configuración inicial del hardware cuando se instala.

- **Buffer FIFO.**

El buffer FIFO es usado para almacenar temporalmente los datos adquiridos. Los datos se almacenan temporalmente hasta que puedan ser transferidos a la memoria del sistema. El proceso de transferencia de datos de entrada y salida de un buffer FIFO de entrada analógica es el siguiente:

5. El buffer FIFO almacena las muestras adquiridas recientemente con una tasa de muestreo constante.
6. Antes de que el buffer FIFO se llene, el software comienza a remover las muestras. Por ejemplo, una interrupción se genera cuando el FIFO está medio lleno, indicando al software extraer las muestras tan pronto como sea posible.
7. Las muestras se transfieren a la memoria del sistema a través del bus del sistema. Después de que las muestras se transfieren, el software es libre para realizar otras tareas hasta que ocurra la siguiente interrupción. Por ejemplo, los datos pueden ser procesados o guardados en un archivo del disco. Ya que la tasa promedio de almacenamiento y extracción de datos son iguales, los datos adquiridos no se perderán y su aplicación deberá funcionar sin problemas.

- **DMA.**

La memoria de acceso directo (DMA) es un sistema por el cual las muestras son automáticamente almacenadas en la memoria del sistema mientras que el

procesador hace otra cosa. El proceso de transferencia de datos a través de DMA es el siguiente:

5. Cuando los datos están listos para la transferencia, la tarjeta dirige el sistema del controlador de la DMA para ponerlos en la memoria del sistema tan pronto como sea posible.
6. Tan pronto como la CPU esté disponible (que suele ser muy rápido), deja de interactuar con el hardware de adquisición de datos y el controlador de la DMA mueve los datos directamente dentro de la memoria.
7. El controlador de la DMA se prepara para la próxima muestra indicando la siguiente ubicación de memoria abierta.
8. Los anteriores pasos son repetidos indefinidamente, con información que llega a cada una de las localidades de memoria abiertas desde los buffers con circulación continua, ninguna interacción entre la CPU y la tarjeta es necesaria.

- **Flujo de datos adquiridos.**

La tasa con la cual los datos son llevados a la memoria del sistema depende de factores incluyendo: la memoria disponible, la tasa con la que los datos son adquiridos y el número de canales del hardware.
La máxima tasa de muestreo, corresponde a la máxima tasa de muestreo de la tarjeta dividido por el número de canales.
El flujo de datos adquiridos consiste de dos pasos independientes:

1. Los datos adquiridos desde el hardware son almacenados en el motor.
2. Los datos son extraídos desde el motor y almacenados en el espacio de trabajo de MATLAB o sacados como archivo.

Estos pasos son ilustrados a continuación:

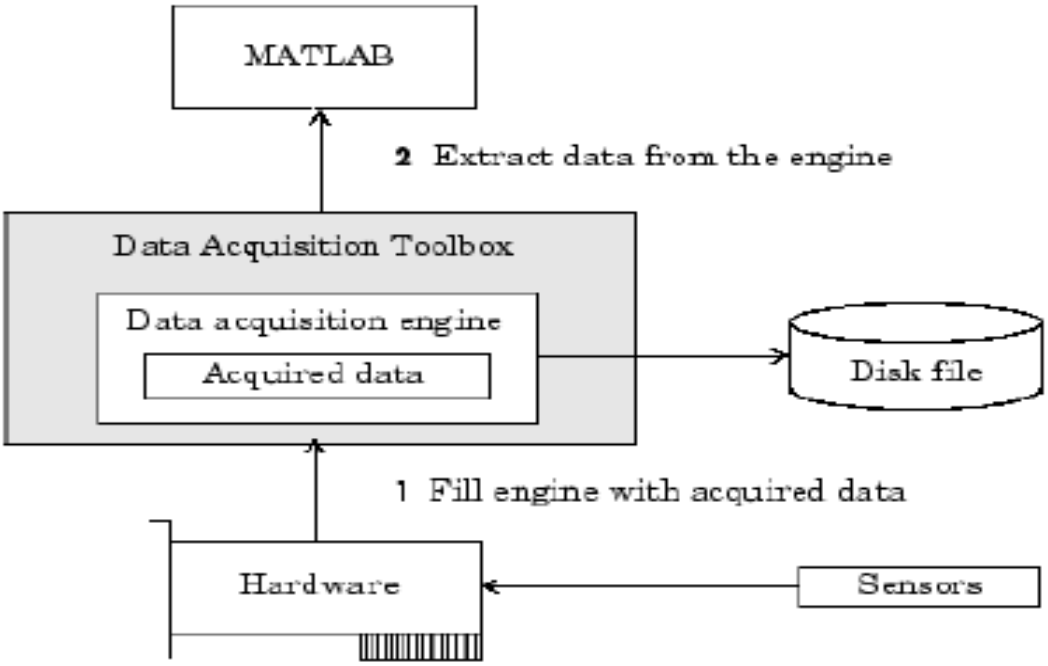


Figura 1.

Flujo de datos, en la adquisición de datos análogos.

- **Sesión de adquisición de datos.**

Una sesión de adquisición de datos se compone de los siguientes pasos:

1. Crear un objeto de dispositivo.
2. Agregar canales.
3. Configurar propiedades.
4. Inicializar la adquisición de datos
5. Esperar a que la adquisición de datos se complete
6. Extraer datos
7. Limpiar

1 **Crear un objeto de dispositivo:** Un objeto se crea usando la función de creación *analoginput()*. Estos objetos son los elementos básicos de la Toolbox que se utilizan para acceder al hardware.

Ejemplo: Para crear el objeto AI se da la siguiente instrucción:

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

Nombre dado al adaptador de las tarjetas de National I.	ID Cuando se utilizan tarjetas con NI-DAQmx se utiliza Dev#.

2 **Agregar canales:** Después que el objeto es creado, se le debe agregar canales con la función *addchannel*(). Los canales son los elementos básicos del hardware con los que se adquiere datos.

Ejemplo: Para adicionar los canales 1 y 2 al objeto AI, se da la instrucción siguiente:

addchannel(AI,1:2);

La relación entre un objeto de entrada análoga y los canales contenidos se muestra a continuación:

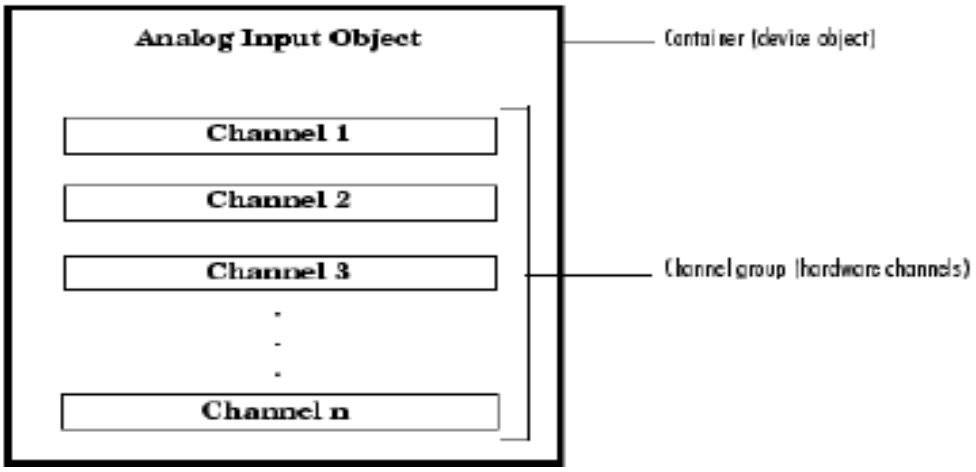


Figura 2.

Relación entre un objeto de entrada análoga y los canales contenidos.

3 **Configurar propiedades:** Para establecer el comportamiento del objeto de entrada análoga, se asignan valores a las propiedades usando la función *set*() o la notación punto. Por ejemplo para configurar la tasa de muestreo se utiliza: *set*(AI,'SampleRate',75); con la función *set()* o AI.SampleRate = 75; con la notación punto. Se está ordenando que la adquisición de la señal tenga una tasa de 75Hz o lo que es lo mismo a 75 muestras por segundo.

Las propiedades pueden ser configuradas en cualquier momento. Sin embargo, algunas propiedades son configurables solo cuando el objeto no está corriendo. Dependiendo de la configuración del hardware y de los requerimientos de la aplicación, es posible utilizar los valores por defecto de las propiedades y saltarse este paso.

La Toolbox de Adquisición de Datos soporta dos tipos de propiedades para objetos de entrada análoga: propiedades comunes y propiedades de canal. Las propiedades comunes aplican para todos los canales contenidos por el objeto, mientras que las propiedades de canal aplican para los canales de forma individual. Las propiedades para objetos de entrada análoga son:

a) Propiedades comunes. Entre otras:

- Propiedades básicas de configuración.
  - SampleRate: Especifica la tasa de muestreo por canal.
  - TriggerType: Especifica el tipo de Trigger a ejecutar. Aunque el objeto de entrada análoga pudiera ejecutarse al ser inicializado, la adquisición no es iniciada porque el dato no está siendo enviado a la memoria del sistema. Esto requiere que un evento ocurra y depende del valor de propiedad tipo de disparo. Después de ejecutarse el *trigger()*, la propiedad Logging se establece automáticamente a ON, y tanto el objeto del dispositivo como el hardware se ejecutarán.
    - Inmediate: se ejecuta justo después de *start().*
    - Manual: se ejecuta después de invocar la función *trigger().*
    - Software: se ejecuta cuando se detecta que la señal de un canal específico cumple una condición determinada. El canal utilizado como fuente de trigger se define con la propiedad TriggerChannel. El valor o el rango especificado que la condición de disparo debe cumplir es indicado por la propiedad

TriggerConditionValue que pertenece a las propiedades comunes dentro de la categoría propiedades del trigger. La condición que debe cumplirse para que se produzca un trigger es especificado por la propiedad TriggerCondition. Usted puede establecer esta propiedad a uno de los siguientes valores:

- Rising: En la señal se debe detectar en un flanco de subida el valor especificado.
- Falling: En la señal se debe detectar en un flanco de bajada el valor especificado.
- Leaving: La señal debe estar saliendo del rango especificado de valores.
- Entering: La señal debe estar entrando en el rango especificado de valores.

  - SamplesPerTrigger: Especifica el número de muestras a adquirir por trigger. Se establece en infinito "inf" para hacer ingreso de datos de manera indefinida.

- Propiedades de configuración del hardware. Entre otras:
  - InputType: Especifica la configuración de los canales del hardware que pueden ser SingleEnded (simple nodo) o Differential (diferencial).
    - Differential: Hay dos canales asociados con cada señal de entrada, uno para la señal de entrada y uno para la señal de referencia (retorno). La adquisición es la diferencia de tensión entre los dos canales, lo que ayuda a reducir el ruido y cualquier voltaje que es común a ambos canales. National Instruments recomienda que se usen entradas diferenciales bajo cualquiera de estas condiciones:
      - La señal de entrada es de bajo nivel (menos de 1 voltio).
      - Las puntas que conectan a las señales son mayores a 10pies.
      - La señal de entrada requiere un punto de referencia a tierra separado o señal de retorno.

- Las puntas de la señal se mueven a través de un medio ruidoso.

  - SingleEnded: Un canal referenciado a tierra.

- Propiedades de status. Podemos hacer una evaluación del estado de la entrada análoga usando las propiedades de status.
  - Logging: Indica si el dato está siendo enviado a la memoria.
  - Running: Indica si el objeto esta inicializado.
  - SamplesAcquired: Indica el número de muestras adquiridas por canal.
  - SamplesAvailable: Indica el número de muestras disponibles por canal en el motor de adquisición de datos.

b) Propiedades de canal. Para observar las propiedades de canal, se debe examinar la configuración básica de la adquisición digitando el nombre del objeto. Dentro de las propiedades de canal se encuentran entre otras:

- ChannelName: Posibilita establecer un descriptivo nombre para el canal. Para asignar el nombre Chan1 al canal 1 se sigue con la función *set()* así: *set*(ai.Channel(1),'ChannelName','Chan1')
- InputRange: Especifica el rango de entrada de datos.
- UnitsRange: Especifica el rango escalando los datos.

4 **Inicializar la adquisición de datos:** Cuando la función *start*() se ejecuta, el objeto empieza a adquirir datos. Mientras la adquisición ocurre se pueden ejecutar comandos en Matlab y luego esperar a que la adquisición de datos se complete. Después de ejecutar *start(),* la propiedad Running se establece automáticamente a ON, y tanto el objeto del dispositivo como el hardware estarán listos para ejecutarse de acuerdo con el valor de las propiedades por defecto y configurados.
Ejemplo: Para inicializar el objeto AI se da la siguiente instrucción:
start(AI)

5 **Esperar a que la adquisición de datos se complete:** Es permitido continuar trabajando en el espacio de trabajo de MATLAB mientras la Toolbox está

adquiriendo datos. Sin embargo, en muchos casos, simplemente se quiere esperar a que la adquisición de datos se complete antes de continuar. La función *wait()* se usa para pausar Matlab hasta que la adquisición termine.
wait(AI,duracion + 1)

6 **Extraer o adquirir datos:** Después de que los datos son adquiridos, deben ser extraídos desde el motor al workspace explícitamente con la función *getdata*(). Si la adquisición se hace usando el oscilloscope, los datos se envían al workspace exportándolos del Oscilloscope.

7 **Limpiar:** Cuando no se necesita por más tiempo el objeto, este puede removerse de la memoria usando la función *delete*() y usando el comando clear es removido del área de trabajo de Matlab.
delete(AI)
clear AI

- **Adquisición de un dato.**

La función *getsample()*, adquiere inmediatamente una muestra análoga sin guardar muestras en, o tener que extraer muestras desde, el motor de adquisición de datos. Se podrá ejecutar ésta función en cualquier momento luego de que los canales hayan sido adicionados al objeto de entrada análoga. *getsample()* no es soportada por las tarjetas de sonido.

Ésta función al usar la variable ¨muestra¨ como ejemplo, se configura:
muestra = getsample(objeto)
Inmediatamente retorna un vector fila, incluyendo una muestra por cada canal contenido por el objeto.

Nota: Para profundizar en los temas relacionados, dirigirse al libro: **IMPLEMENTACIÓN DE LA "TOOLBOX" DE ADQUISICIÓN DE DATOS DE**

MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.

- **PROCEDIMIENTO 1.**

1. **Explorando la Toolbox.**

1.1 Verificar para la tarjeta de sonido: Versión del Toolbox y de Matlab, nombre y versión del adaptador, nombre de la tarjeta. hw=daqhwinfo('winsound').

1.2 En la parte inferior encontrará una serie de comandos secuenciados que se utilizan para adquisición de voz con la Toolbox de Matlab, se debe poner al frente de cada instrucción la letra correspondiente a la explicación.

a. Las tarjetas de sonido tienen dos canales, si solo se selecciona uno la tarjeta estará en modo mono y seleccionando los dos estará en modo estéreo. Con esta instrucción se crean dos canales.
b. Se configura la tasa de muestreo a 8kHz.
c. Se especifica el número de muestras por disparo.
d. Se inicia la adquisición.
e. Se llama al Oscilloscope.
f. Se determina el tiempo de adquisición.
g. Se crea un objeto (ai) de entrada análoga, en el caso de la tarjeta de sonido, winsound no requiere un identificador (ID).
h. A una variable se le asigna la actual tasa de muestreo.

```
>> ai=analoginput('winsound');                    ---------
>> addchannel(ai,1:2);                            ---------
>> duracion=10;                                   ---------
>> set(ai,'SampleRate',8000);                     ---------
>> ActualRate=get(ai,'SampleRate')                ---------
```

ActualRate =

8000

>> set(ai,'SamplesPerTrigger',duracion*ActualRate)---------
>> softscope(ai) ---------
>> start(ai) ---------

2. **Adquisición de voz.** Para adquirir los datos con su tarjeta de sonido, necesita un origen de datos, los datos del sensor y un receptor de datos. En esta demostración: La fuente de datos es la entrada de sonido a la tarjeta de sonido. Este sonido puede provenir de una variedad de fuentes, incluyendo la voz, las manos aplaudiendo, o un CD en su ordenador. El sensor es un micrófono en su computadora y el receptor es un canal asociado a un objeto de entrada analógica.

2.1 Digitar el anterior código en Matlab y emitir el sonido a adquirir, utilizando un micrófono justamente después de ejecutar el comando *Start()* o asegurándose que se está ejecutando alguna entrada de sonido. El código se encuentra organizado.

2.2 Agregar segundo display para el canal 2.

2.3 Ingresar la señal de referencia ref, al display 1.
t = 0:0.0001:2; w = 200*2*pi; ref = 2*sin(w*t);

2.4 Configurar la escala y el offset en cada display.

2.5 Ingresar la señal matemática m1, al display 2.
abs(CH1)+abs(CH2).

2.6 Realizar las siguientes mediciones:

a) Frecuencia de ref.
b) Voltaje medio de CH1.
c) Máximo valor de CH2.
d) Mínimo valor de m1.

❖ **PROCEDIMIENTO 2.**

1. **Adquisición con el oscilloscope, uso del trigger inmediato y canales en nodo simple.**

a. Para adquirir datos analógicos, se conectan esas señales a los puertos de entrada de la DAQ, que corresponderán a los canales. En este caso dos señales en nodo simple por AI 1 y AI 2 (ver práctica 1).

b. Muestre esas señales en el osciloscopio FLUKE. Para la posterior comparación.

c. Comprobar la correcta instalación de la tarjeta USB 6008/6009 y su ID con la función hw=daqhwinfo('nidaq').

d. Digitar el siguiente código en Matlab. Al cambiarse una línea se deberán re digitar las líneas subsiguientes, siempre que no se incluyan las líneas del séptimo ítem.

1 Crear un objeto de dispositivo:

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

2 Agregar canales:

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

2.1 Examinar la configuración básica de la adquisición:

```
AI
```

3 Examinar la lista de propiedades configurables:

```
set(AI)
```

3.1 Configurar propiedades: Configurar la tasa de muestreo a 1000 muestras/s y definir el tiempo de adquisición en 10 s. Con el tipo de trigger inmediato.

```
duracion=10;
set(AI,'SampleRate',1000)
ActualRate=get(AI,'SampleRate')
set(AI,'SamplesPerTrigger',duracion*ActualRate)
set(AI, 'TriggerType', 'Immediate');
set(AI, 'InputType', 'SingleEnded');
```

3.2 Examinar las propiedades y su configuración:

```
get(AI)
```

4 Llamar al oscilloscope:

```
softscope(AI)
```

5 Correr la adquisición de datos:

```
start(AI)
```

5.1 Obtener el número de muestras adquiridas por canal:

```
get(AI, 'SamplesAcquired')
```

6 Exportar los datos al Workspace:

Siga la secuencia: File>Export>Channels.

7 Limpiar:

```
delete(AI)
clear AI
```

e. Compárense las señales tomadas en el osciloscopio Fluke con las mostradas en el oscilloscope, en su forma, amplitud y frecuencia.

f. Cambie la tasa de muestreo a 10, 500 y 5000 muestras/s.

g. Cambie el tiempo de duración a infinito y manipule la frecuencia y la amplitud de la señal original.

2. Adquisición en el ambiente de Matlab, uso del trigger por software y canales en modo diferencial.

a. Para adquirir datos analógicos, se conectan esas señales a los puertos de entrada de la DAQ, que corresponderán a los canales. En este caso una señal en modo diferencial por AI 1 señal de entrada y la señal de referencia sigue siendo GND (ver práctica 1).

b. Muestre la señal en el osciloscopio FLUKE. Para la posterior comparación.

c. Comprobar la correcta instalación de la tarjeta USB 6008/6009 y su ID con la función hw=daqhwinfo('nidaq').

d. Digitar el siguiente código en Matlab. Al cambiarse una línea se deberán re digitar las líneas subsiguientes, siempre que no se incluyan las líneas del sexto ítem.

1 Crear un objeto de dispositivo:

```
ai = analoginput('nidaq','Dev1');
```

2 Agregar canales:

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

2.1 Examinar la configuración básica de la adquisición:

ai

3 Examinar la lista de propiedades configurables:

set(ai)

3.1 Configurar propiedades: Entrada en modo diferencial. Configurar la tasa de muestreo a 5000 muestras/s y definir el tiempo de adquisición en 15s. Con el tipo de trigger software, que producirá el evento de inicio cuando la señal en el canal AI 1 tenga un flanco de subida pasando por 1 voltio. Multiplicar cada dato por 10 haciendo un acondicionamiento con el rango de unidades.

```
set(ai, 'InputType', 'Differential');
set(ai,'SampleRate',5000)
set(ai,'SamplesPerTrigger',75000)
set(ai, 'TriggerType', 'software');

set(ai, 'TriggerCondition', 'rising');

set(ai, 'TriggerConditionValue', 1);

set(ai, 'TriggerChannel', ai.channel(1));

set(ai.Channel(1),'UnitsRange',[-100 100])
```

3.2 Examinar la configuración básica de la adquisición:

ai

3.3 Examinar las propiedades y su configuración:

get(ai)

4 Inicializar la adquisición de datos:

start(ai)

4.1 Comprobar si el dato está siendo enviado a la memoria:

status = get(ai, 'Logging') Continuar cuando este en Off.

4.2 Obtener el número de muestras adquiridas por canal:

get(ai, 'SamplesAcquired') Continuar cuando el número de muestras sea completo.

5 Exportar los datos al Workspace:

[data,time] = getdata(ai);

5.1 Graficar:

plot(time,data);
zoom on;
title('Software Trigger');
xlabel('Tiempo relativo en segundos.');
ylabel('Datos en voltios');

6 Limpiar:

delete(ai)
clear ai

e. Compare las señales tomadas en el osciloscopio Fluke con las mostradas en el oscilloscope, en su forma, amplitud y frecuencia.

f. Observe el valor de voltaje para el tiempo relativo de 0s.

g. Cambie la tasa de muestreo 50 muestras/s.

16.2 PRÁCTICA # 2 SISTEMA DE ADQUISICIÓN DE DATOS CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DE MATLAB, SALIDA ANÁLOGA, ENTRADA DIGITAL Y SALIDA DIGITAL.

❖ MARCO TEÓRICO 1ª PARTE. SALIDA ANÁLOGA.

- **Descripción general.**

Un subsistema de salida análoga convierte los datos digitales almacenados en el computador a una señal análoga real. Las tarjetas NI USB-6008/6009 ofrecen dos canales con 12 bits de resolución. La Toolbox de Adquisición de Datos provee acceso a los subsistemas de salida análoga a través de un objeto de salida análoga.

- **Flujo de salida de datos.**

El flujo de salida de datos se refiere al flujo de datos desde el motor de adquisición de datos al hardware. Sin embargo, antes de que los datos sean sacados, se deben poner en cola en el motor. La cantidad de datos que se pueden poner en cola depende de factores incluyendo la memoria disponible, el número de canales del hardware y el tamaño de cada muestra de dato.

El flujo de salida de datos consiste de dos pasos independientes:

3. Los datos son puestos en cola en el motor desde el espacio de trabajo de MATLAB.
4. Los datos en cola en el motor son sacados al hardware.

Estos pasos son ilustrados a continuación:

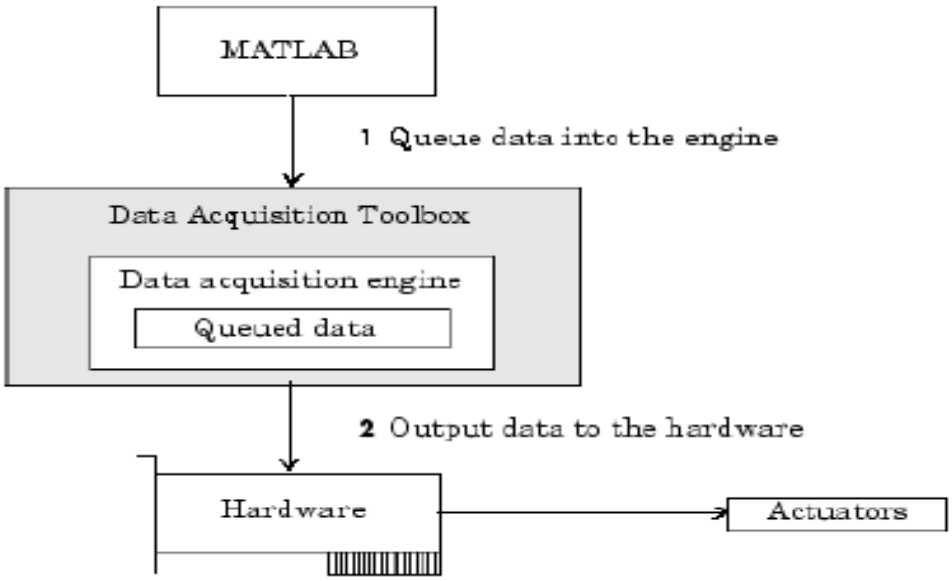


Figura 1.

Pasos del flujo de datos, en la salida de datos.

- **Sesión de adquisición para sacar datos.**

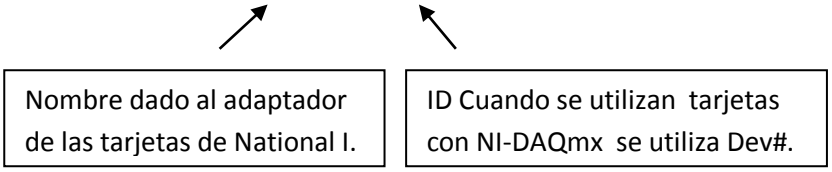
Una sesión de adquisición de datos se compone de los siguientes pasos para sacar datos:

8. Crear un objeto de dispositivo.
9. Agregar canales.
10. Configurar propiedades.
11. Enfilar los datos
12. Inicializar la entrega de datos
13. Detener el objeto de salida análoga
14. Limpiar

1 **Crear un objeto de dispositivo:** Para crear un objeto de salida análoga se utiliza la función *analogoutput().* Ésta función la compone el nombre del adaptador y el ID del hardware como se ve a continuación.

Ejemplo: Para crear el objeto ao se da la siguiente instrucción:

ao = analogoutput('nidaq','Dev1');



2 **Agregar canales:** Los canales son agregados al objeto con la función *addchannel()*. Ésta función requiere el objeto como argumento de entrada. Los canales son los elementos básicos del hardware con el cual se entrega datos.

Ejemplo: Para adicionar los canales 1 y 2 al objeto ao, se da la instrucción siguiente:

addchannel(ao,1:2);

3 **Configurar propiedades:** Para establecer el comportamiento del objeto de salida análoga, se asignan valores a las propiedades usando la función *set*() o la notación punto. La Toolbox de Adquisición de Datos soporta dos tipos de propiedades para objetos de salida análoga: propiedades comunes y propiedades de canal. Las propiedades comunes aplican para todos los canales contenidos por el objeto, mientras que las propiedades de canal aplican para los canales de forma individual. Las propiedades para objetos de salida análoga son:

c) Propiedades comunes. Entre otras:

- Propiedades básicas de configuración.
    - SampleRate: Especifica la tasa de muestreo por canal.
    - TriggerType: Un trigger de salida análoga es definido como un evento que inicia la salida de datos. Se tienen dos tipos de trigger, inmediato y manual. Cuando se produce un trigger, la propiedad Sending es establecida automáticamente en ON y los datos en cola son sacados desde el motor al subsistema del hardware. Para que el trigger de una salida análoga ocurra, se debe seguir estos pasos:

- Poner en cola los datos en el motor.
- Configurar las propiedades del trigger
- Ejecutar la función *start()*.
- Ejecutar la función *trigger()* si el valor de la propiedad TriggerType es manual.

- Propiedades de status. Podemos hacer una evaluación del estado de la salida análoga usando las propiedades de status.
  - Sending: Indica si el dato está siendo enviado al hardware.
  - Running: Indica si el objeto esta inicializado.
  - SamplesOutput: Indica el número de muestras sacadas por canal desde el motor de adquisición de datos.
  - SamplesAvailable: Indica el número de muestras disponibles por canal en el motor de adquisición de datos.

d) Propiedades de canal. Para observar las propiedades de canal, se debe examinar la configuración básica del objeto de salida análoga digitando el nombre del objeto. Dentro de las propiedades de canal se encuentran entre otras:

- Units: Especifica la etiqueta de las unidades.
- OutputRange: Especifica el rango de salida de datos.
- UnitsRange: Especifica el rango escalando los datos.

4 **Enfilar los datos:** Antes de sacar datos análogos, los datos deben se enfilados o puestos en cola en el motor de adquisición de datos con la función *putdata()*.

Ejemplo: Para enfilar ¨data¨ para dos canales del objeto ao, se dá la instrucción siguiente:

putdata(ao,[data data])

5 **Inicializar la entrega de datos:** Para inicializar el objeto de salida análoga se usa la función *start()*.Después de que start es ejecutado, la propiedad Running es establecida automáticamente en ON, y tanto el dispositivo como el hardware se

ejecutan según lo configurado y los valores por defecto de propiedades. Mientras el objeto esté corriendo, se puede continuar enfilando datos. Sin embargo, un trigger se debe ejecutar para que la propiedad sending pase a ON, de lo contrario el motor de adquisición no ejecuta la salida de datos.

Ejemplo: Para inicializar el objeto ao se da la siguiente instrucción:
start(ao)

6 **Detener el objeto de salida análoga:** Un objeto de salida análoga puede detenerse bajo una de estas condiciones:
- La función *stop()* es llamada.
- Los datos enfilados son sacados.
- El tiempo de ejecución se acaba.

Cuando alguna de estas condiciones se ejecutan sending pasa automáticamente a OFF.

7 **Limpiar:** Cuando no se necesita por más tiempo el objeto, este puede removerse de la memoria usando la función *delete*() y usando el comando clear es removido del área de trabajo de Matlab.
delete(ao)
clear ao

- **Salida de un dato.**

La función *putsample()*, saca inmediatamente una muestra análoga sin poner en cola las muestras en el motor de adquisición de datos. Se podrá ejecutar ésta función en cualquier momento luego de que los canales hayan sido adicionados al objeto de salida análoga. *putsample()* no es soportada por las tarjetas de sonido.

Ésta función se configura así:
putsample(objeto,dato)

Inmediatamente saca un vector fila dato, el cual incluye una muestra por cada canal contenido por el objeto.

❖ MARCO TEÓRICO 2ª PARTE. ENTRADA DIGITAL Y SALIDA DIGITAL.

- **Descripción general.**

Los subsistemas de E/S digital están diseñados para la transferencia de valores digitales desde y hacia el hardware. Estos valores se manejan ya sea en forma de bits individuales o líneas, o como puerto. La Toolbox de adquisición de datos es un software que provee acceso a un subsistema de entrada y salida digital a través de un objeto de entrada y salida digital. Una muy importante ventaja con respecto de otros software de aplicación es el manejo de los datos mediante un vector binario.

- **Flujo de datos para salida y entrada digital.**

A diferencia de los objetos de entrada y salida análoga, no se puede controlar el comportamiento de los objetos de entrada/salida digital por medio de la configuración de propiedades. Esto se debe a que un buffer DIO no es soportado y los datos no son almacenados en el motor. En cambio, se pueden escribir o leer valores directamente de las líneas de hardware. Al iniciar el sistema, los puertos digitales están en alta impedancia. Es necesario no exceder el nivel de tensión de + 5V en las entradas digitales pues se corre el riesgo de dañar tanto a la tarjeta como al PC.

➢ Sesión de entrada y salida de datos digitales.

Una sesión de adquisición de datos digitales se compone de los siguientes pasos:

1. Crear un objeto I/O digital.
2. Agregar líneas a un objeto de I/O digital.
3. Leer Valores digitales.
4. Escribir valores digitales.
5. Limpiar

1 **Crear un objeto I/O digital:** Un objeto de entrada y salida digital (DIO) se crea con la función *digitalio()*; esta acepta el nombre del adaptador y el ID del dispositivo de hardware como argumentos de entrada. Para tarjetas de adquisición de datos, el ID del dispositivo se refiere al número asociado con la tarjeta cuando ésta es instalada. Con las tarjetas NI-DAQmx, usualmente es una cadena de caracteres como 'Dev1'. La función *daqhwinfo(),* determina los adaptadores disponibles y los IDs del dispositivo.

2 **Agregar líneas a un objeto de I/O digital:** Después de crear el objeto de entrada/salida digital, se le deben agregar líneas. Para agregar líneas se utiliza la función *addline()*. Esta función requiere el objeto del dispositivo, al menos un ID de línea de hardware y la dirección (entrada o salida) como argumentos de cada línea agregada. Opcionalmente se puede especificar los nombres de líneas.
Se pueden retornar las características de la línea y del puerto con la función *daqhwinfo()*. Por ejemplo, la tarjeta National Instruments USB 6008/6009 tiene dos puertos con ocho y cuatro líneas. Para retornar las características de entrada/salida digital para esta tarjeta:

```
hwinfo = daqhwinfo(dio);
```

Muestra las características de la línea por cada puerto.

```
hwinfo.Port(1)
ans =
                ID: 0
 LineIDs: [0 1 2 3 4 5 6 7]
Direction: 'in/out'
Config: 'port'
```

```
hwinfo.Port(2)
ans =
    ID: 1
 LineIDs: [0 1 2 3]
 Direction: 'in/out'
 Config: 'port'
```

Esta información nos dice, que se pueden configurar las 12 líneas tanto para entrada como para salida y que los dos puertos son de puerto configurable.
Que la tarjeta USB 6008/6009 sea un dispositivo con puertos de puerto configurable nos indica, que no se pueden direccionar de manera individual las líneas asociadas con el puerto. Por lo tanto, se deben configurar todas las líneas o para entrada o para salida. Se puede agregar cualquier número deseado de líneas de puerto configurable a un objeto de entrada/salida digital. Sin embargo, el motor direccionará todas las líneas de manera oculta. Los puertos de la tarjeta USB 6008/6009 es de puerto y no de línea configurable, si se intenta combinar las dos configuraciones, un error es retornado.

3 **Leer Valores digitales:** Se pueden leer valores de una o más líneas con la función *getvalue()*. Esta función requiere el objeto de entrada/salida como un argumento de entrada.
La lectura de valores de líneas digitales, sigue estas reglas:

- Si el objeto de entrada/salida digital contiene líneas de un dispositivo de puerto configurable, entonces todas las líneas son leídas incluso si no son contenidas por el objeto del dispositivo. Sin embargo, solo los valores de las líneas contenidas en el objeto son retornadas.
- Siempre se puede leer en una línea configurada para entrada.
- Para hardware de National Instruments usando la interfaz tradicional NI-DAQ, las líneas configuradas para entrada retornan un valor 1 por defecto.

➢ *getvalue() siempre retorna un* binvec*.* Para convertir el binvec a un valor decimal, se usa la función *binvec2dec().*

4 **Escribir valores digitales:** Se escriben valores a las líneas digitales con la función *putvalue()*. Esta función requiere el objeto de entrada/salida digital y los valores que se van a escribir como argumentos de entrada.

Escribir valores digitales, cumple las siguientes reglas:

- Si el objeto de entrada/salida digital contiene líneas de un dispositivo de puerto configurable, entonces el motor de adquisición de datos escribe en todas las líneas asociadas con el puerto incluso si no están contenidas por el objeto del dispositivo.
- Cuando se escriben valores decimales,
  - Si el valor es muy grande para ser representado por las líneas contenidas por el objeto del dispositivo, entonces un error es retornado.
  - Se pueden escribir un máximo de 32 líneas. Para escribir más de 32 líneas, se debe usar un valor binvec.
- Cuando se escriben valores binvec,
  - Se pueden escribir cualquier número de líneas.
  - Debe haber un elemento en el binvec para cada línea a la que se escriba
- Siempre se puede escribir en una línea configurada para salida.
- Un error es retornado si se escribe un valor negativo o si se escribe a una línea configurada para entrada.

5 **Limpiar:** Cuando no se necesita por mucho tiempo el objeto, este puede removerse usando la función *delete()* y usando el comando clear son removidos del área de trabajo de MATLAB.

Nota: Para profundizar en los temas relacionados, dirigirse al libro: **IMPLEMENTACIÓN DE LA "TOOLBOX" DE ADQUISICIÓN DE DATOS DE MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.**

❖ **PROCEDIMIENTO 1.**

1. **Emitir un sonido (pito) con el uso de la tarjeta de sonido y uso del trigger manual.**

h. Comprobar la correcta instalación de la tarjeta de sonido, del controlador y su ID con la función hw=daqhwinfo('winsound').

i. Digitar el siguiente código en Matlab y oír. Al cambiarse una línea se deberán re digitar las líneas subsiguientes, siempre que no se incluyan las líneas del séptimo ítem.

1 Crear el objeto:

```
ao = analogoutput('winsound');
```

2 Agregar canales:

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

2.1 Examinar la configuración básica de la adquisición:

```
ao
```

3 Examinar la lista de propiedades configurables:

```
set(ao)
```

3.1 Configurar propiedades y crear señal:

```
set(ao,'SampleRate',44100)
set(ao, 'TriggerType', 'manual');
data = sin(500*(0:(2*pi/8000):2*pi))';
```

3.2 Examinar las propiedades y su configuración:

```
get(ao)
```

4 Enfilar los datos:

```
putdata(ao,[data data])
```

5 Inicializar el objeto:

start(ao)

5.1 Comprobar si el objeto esta inicializado y si los datos se han puesto en cola en el motor de adquisición de datos:

status = get(ao, 'Running')

5.2 Obtener el número de muestras disponibles por canal en el motor de adquisición de datos:

get(ao, 'SamplesAvailable')

6. Generar el evento de envío de datos al hardware:

trigger(ao);

6.1 Obtener el número de muestras sacadas por canal desde el motor de adquisición de datos:

get(ao, 'SamplesOutput')

7 Limpiar:

delete(ao)

clear ao

3. **Uso de la función *putsample()*.** Se sacarán datos de dos vectores con la tarjeta USB 6008/6009. Debido a que ésta tarjeta no posee reloj interno en el subsistema de salida, en éste procedimiento se utiliza la función *putsample*().

a. Comprobar la correcta instalación de la tarjeta USB 6008/6009 y su ID con la función hw=daqhwinfo('nidaq').

b. Conectar el osciloscopio FLUKE a los canales de la tarjeta.

c. Digitar el siguiente código en Matlab y ver en el osciloscopio.

```
a=1;
% crear los vectores a sacar.
vector1=[1.6 2 3 4 5 4 2.3 2.5];
vector2=[4 5 3.2 2 4.7 3.1 2 1];
% crear el objeto de salida análoga.
AO=analogoutput('nidaq','Dev3');
% adicionar canales.
Canales = addchannel(AO,0:1);
% ciclo incremental de la variable contadora.
for a=1:8
% poner inmediatamente la muestra a del vector1 en el canal 0 y la muestra a del
% vector2 en el canal 1.
putsample(AO,[vector1(a) vector2(a)])
% condición para detener el sitema.
if a==length(vector1)
end
% definir el tiempo de salida de cada dato.
pause(T)
end
```

d. Observar el tiempo de salida de cada dato y el nivel de voltaje.

❖ **PROCEDIMIENTO 2.**

1. **Escribir y leer datos digitales.**

1.1 Seguir las instrucciones.

1 Crear un objeto del dispositivo – Crear el objeto de entrada/salida digital dio para una tarjeta National Instruments. El ID del hardware se encuentra con hw=daqhwinfo('nidaq').

dio = digitalio('nidaq','Dev1');

1.1 Observar las características de la línea por cada puerto.

hwinfo = daqhwinfo(dio);

hwinfo.Port(1)

hwinfo.Port(2)

2 Agregar líneas para escribir – Agregar cuatro líneas de salida en el puerto 0.

addline(dio, 0:3, 0, 'Out')

3 Escribir valores – Escribir un valor de trece en las cuatro líneas en el puerto 0 como un número decimal o un binvec. Los pasos 2 y 3 deben ser consecutivos.

putvalue(dio, 13); Para decimal.

3.1 Salvar la palabra del objeto a una variable y convertirlo a decimal.

valus= getvalue(dio)

valusd = binvec2dec(valus)

4 Agregar líneas para leer – Agregar cuatro líneas de entrada en el puerto 1.

addline(dio, 0:3, 1, 'in')

5 Leer valores – Leer el valor de las cuatro líneas del puerto 0, a las cuatro líneas del puerto 1 y convertirlos a decimal. Los pasos 4 y 5 deben ser consecutivos.

valut = getvalue(dio)

value = valut(5:8)

valued = binvec2dec(value)

6 Escribir un valor de quince a las otras cuatro líneas del puerto 0 como un número decimal o un binvec, y leer de vuelta los valores en el puerto 1.

7 Limpiar – Cuando no se necesite mas el objeto dio, se debe remover de la memoria y del área de trabajo de MATLAB.
delete(dio)
clear dio

8 Con el motor detenido, leer la palabra que indica la posición del motor en el puerto 0.

16.3 PRÁCTICA # 3. SISTEMA DE ADQUISICIÓN DE DATOS ANÁLOGOS Y DIGITALES CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DESDE SIMULINK Y APLICACIÓN PARA EL ESTUDIO DEL COMPORTAMIENTO DE LOS CONTROLADORES DISCRETOS P, PI Y PID, DESDE SIMULINK.

- **MARCO TEÓRICO 1ª PARTE. SISTEMA DE ADQUISICIÓN DE DATOS ANÁLOGOS Y DIGITALES CON LA TOOLBOX DE ADQUISICIÓN DE DATOS DESDE SIMULINK.**

- **Descripción general.**

El programa de MathWorks para simulación (modelación y análisis) de sistemas dinámicos no lineales SIMULINK, es una utilidad gráfica para la simulación de sistemas, tanto analógicos como discretos. Tiene un tratamiento similar a los otros Toolboxes de MATLAB, en el sentido que se instala de forma separada, pero

sigue siendo la mejor herramienta para aprovechar toda la potencia de MATLAB y de los otros "Toolboxes" lo que hace que sea adecuado para aplicaciones de adquisición de datos, procesado de señales, comunicaciones, control, etc. Los diagramas de bloques permiten una descripción en alto nivel del sistema además de ser fácilmente modificables con la finalidad de conseguir el comportamiento deseado, es decir, proporcionan una estructura jerárquica para evaluar el comportamiento de algoritmos alternativos bajo diferentes condiciones de funcionamiento. Cada bloque puede representar un solo elemento del proceso o bien un subsistema, además de ser fácilmente modificable para reflejar un cambio en el algoritmo o el enfoque del diseño. Simulink posee la librería de bloques para la Toolbox de adquisición de datos que corresponde a los siguientes bloques:

- **Analog Input** — Adquiere datos desde múltiples canales análogos.
- **Analog Input (una muestra)** — Adquiere una muestra de datos desde múltiples canales análogos.
- **Analog Output** — Exporta datos a múltiples canales análogos.
- **Analog Output (una muestra)** — Exporta una muestra de datos a múltiples canales análogos.
- **Digital Input** — Adquiere el más reciente set de datos desde múltiples líneas digitales.
- **Digital Output** — Exporta datos a múltiples líneas digitales.

Estos bloques se podrán usar para adquirir datos análogos o digitales a un modelo en Simulink®, o para exportar datos análogos o digitales desde el modelo a un hardware, también se podrán interconectar con bloques de otras librerías. Para abrir la librería de bloques de la Toolbox de adquisición de datos, desde la ventana de librerías de Simulink se selecciona la librería Data Acquisition Toolbox.

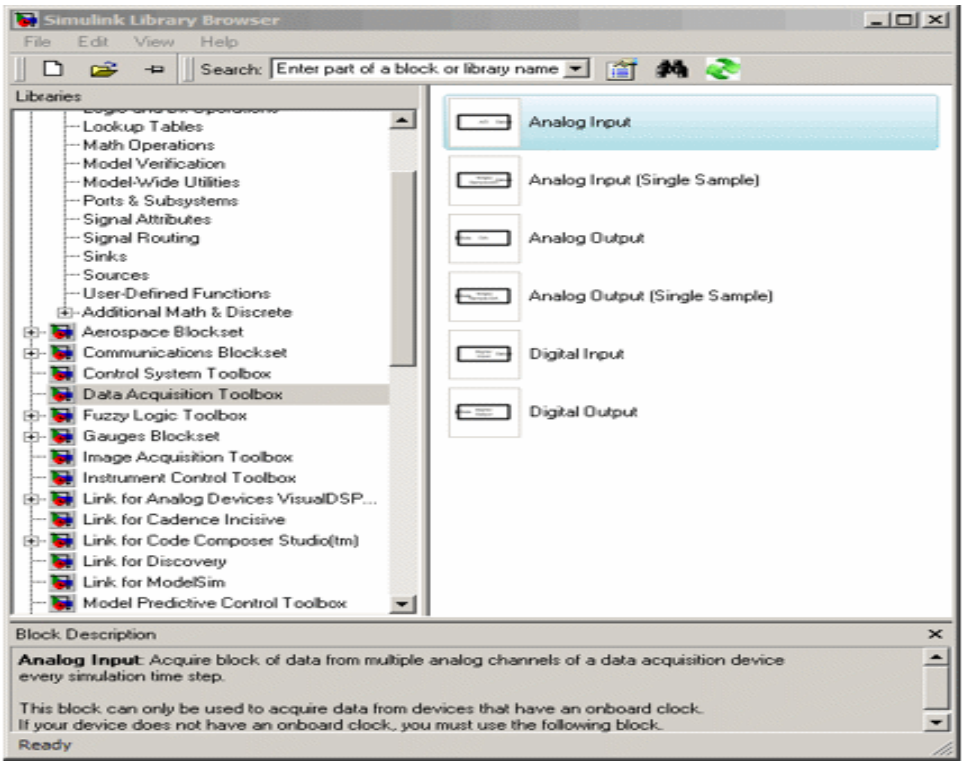


Figura 1.

Librería de bloques de la Toolbox de adquisición de datos.

La librería de bloques de la Toolbox de adquisición de datos soporta el uso del acelerador de Simulink (Simulink® Accelerator™), que mejora el tiempo de ejecución. Para su uso se requiere la instalación del compilador C^{++}.

- **MARCO TEÓRICO 2ª PARTE. APLICACIÓN PARA EL ESTUDIO DEL COMPORTAMIENTO DE LOS CONTROLADORES DISCRETOS P, PI Y PID, DESDE SIMULINK.**

- **Control de posición.**

El objetivo del control de posición es conseguir que la salida del sistema siga la entrada.

En la experiencia se usa un sistema realimentado. El sistema de adquisición de datos se emplea para ingresar la posición del motor y para sacar la alimentación al motor. La diferencia entre la señal de referencia y la medición de la posición del motor es la señal de error. Al multiplicar el error con el bloque controlador obtenemos la alimentación del motor. Sintonizando el controlador, se puede cumplir requerimientos como niveles de error y porcentaje de overshoot o disminuir el riesgo de inestabilidad.

La medición de la posición del motor se hace mediante un potenciómetro lineal del que se obtiene un voltaje entre -10V y 10V en un giro completo. Esta posición se toma para el caso de la tarjeta análoga FEEDBACK en el pin (Θo) o en la tarjeta www.octoplusaz.com en (SALIDA – OUTPUT). La señal de referencia será el escalón unitario.

- **Algoritmo del P I D paralelo discretizado.**

9.5.2.1.1.1.1 El literal T representa el periodo de muestreo.

$$Pc(z) = K_p + K_i \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}} + K_d \frac{1-z^{-1}}{T}$$

- **Algoritmo del P I D serie discretizado.**

9.5.2.1.1.1.2 El literal t representa el periodo de muestreo.

$0.05<\alpha<0.1$ $ki=kp/ti$ $kd=kp.td$

$$Sc(z) = \frac{k3 + k1.z^{-1} + k2.z^{-2}}{k6 - k4.z^{-1} + k5.z^{-2}}$$

$k1=(t^2-2.ti.t-4.ti.td)kp$

$k2=(2.ti.td-t.td)kp$

$k3=(t.td+2.ti.td+2.ti.t)kp$

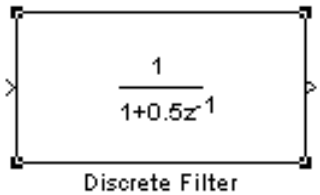
$k4=4.ti.td.\alpha+2.ti.t$

$k5=2.ti.td.\alpha$

$k6=2.ti.t+2.ti.td.\alpha$

- **Bloque del filtro discreto.**

Este bloque nos brinda todas las posibilidades en la implementación del PID discreto en Simulink. Pertenece a la librería Discrete, se usa en el modelado de sistemas discretos para las funciones con variables en el dominio de la transformada Z.



9.5.2.2

Figura 1.

Bloque del filtro discreto.

Nota: Para profundizar en los temas relacionados, dirigirse al libro: **IMPLEMENTACIÓN DE LA "TOOLBOX" DE ADQUISICIÓN DE DATOS DE MATLAB Y PRÁCTICAS DE LABORATORIO CON EL USO DE LAS TARJETAS NI-USB 6008/6009.**

- **PROCEDIMIENTO 1.**

  - **Adquisición de datos análogos.**

1. Conectar la tarjeta USB 6008/6009.

2. Entrar a la librería de bloques de la Toolbox de adquisición de datos y adicionar el bloque Analog Input al modelo.

3. Configurar el bloque de parámetros del bloque Analog Input (doble clic derecho):
   - Escoger el adaptador, el ID y la tarjeta con ¨Device¨.
   - Especificar la velocidad de muestreo en ¨Hardware sample rate (samples/second)¨.
   - Establecer el tamaño de bloque con ¨Block size¨ en 2.
     - Se indica la cantidad de muestras a capturar por bloque para la interpolación.
     - El tiempo en cada bloque depende de la tasa de muestreo.
     - En tanto mayor sea el tamaño de bloque será menor la cantidad de estos. Por ejemplo, con una tasa de muestreo de 10 muestras/s y un tamaño de bloque de 2, se obtendrán 5 bloques de dos muestras en 1s, que se reflejará en el resultado como 5 mediciones. Para un tamaño de bloque de 10, se obtendrá 1 bloque de 10

muestras en 1s que se reflejará en el resultado como solo 1 medición.

- Configurar la entrada al hardware en diferencial con ¨Input type¨.
- Determinar los canales y el nombre de los mismos en ¨Channels¨.
- Establecer el número de puertos (1 per hardware channel) en ¨Number of ports¨.

4. Adicionar el Scope, el multiplexor (si se quiere ingresar más de una señal) y conectar los bloques (ver Figura 2.)

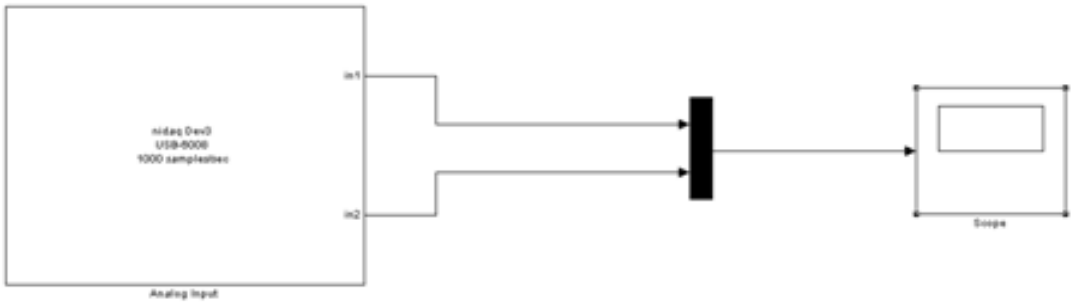


Figura 2.

Diseño de ejemplo a simular.

5. Dar doble clic sobre el Scope, hacer clic sobre el icono de parámetros y seleccionando la pestaña ¨Data history¨ deshabilitar ¨Limit data points to last¨.

6. Preparar el envío de datos al workspace. Dar doble clic sobre el Scope, hacer clic sobre el icono de parámetros y seleccionando la pestaña ¨Data history¨ habilitar ¨Save data to workspace¨, establecer un nombre para la variable con ¨Variable name¨ y seleccionar como formato (Structure with time) en ¨Format¨.

7. Conectar los canales de la tarjeta y el osciloscopio FLUKE a las señales a adquirir.
8. En el recuadro ¨Simulation stop time¨ del modelo definir el tiempo de adquisición.

9. Iniciar la adquisición con el botón ¨Start simulation¨ y compare la gráfica del Scope con el osciloscopio en su amplitud, forma y periodo.

10. Confirmar la inclusión de la variable en el workspace.

11. Para llevar los datos al workspace y verlos, digitar el siguiente código (para un nombre de variable A):

    A
    t=A.time
    A.signals
    y=A.signals.values

- **Salida de una muestra.**

1. A un nuevo modelo, adicionar el bloque Analog Output (Single Sample).

2. Configurar el bloque de parámetros del bloque Analog Output (Single Sample) (doble clic derecho):
    - Escoger el adaptador, el ID y la tarjeta con ¨Device¨.
    - Determinar los canales y el nombre de los mismos en ¨Channels¨.
    - Establecer el número de puertos (1 per hardware channel) en ¨Number of ports¨.

3. Completar el diseño con las señales a sacar (una muestra de cada una), seguido de un retenedor de orden cero (zero-older Hold), (ver Figura 3).

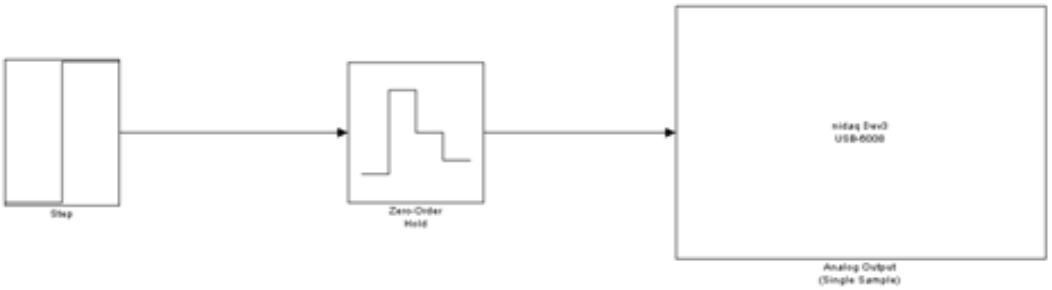


Figura 3.

Diseño de ejemplo a simular.

4. Determinar el tiempo de duración de la muestra con el retenedor de orden cero (doble clic derecho).

5. Conectar el osciloscopio FLUKE a los canales.

6. Iniciar la salida con el botón ¨Start simulation¨ y observar la grafica del osciloscopio.

❖ **PROCEDIMIENTO 2.**

o **PID discreto.**

1. Conectar la tarjeta de adquisición fijando el canal 1 de entrada análoga para ingresar la posición del motor y el canal 0 de salida análoga para sacar la señal de alimentación para el motor, tomándola desde la salida acondicionada con rango de -10v a 10v.

- Control proporcional P.

2. Desarrollar el modelo en Simulink de la figura 2. Especificar el período de muestreo en (Block sample time) para el bloque de entrada análoga

de una muestra y para el bloque del filtro discreto se establece en (Sample time).

3. Ingrese cuatro valores para Kp según las indicaciones del profesor, desarrollar la tabla 1 y la gráfica 1.

- Control proporcional-integral PI paralelo.

4. Desarrollar el modelo en Simulink de la figura 3. Especificar el período de muestreo.

5. Fije Kp e ingrese cuatro valores para Ki según las indicaciones del profesor, desarrollar la tabla 2 y la gráfica 2.

- Control proporcional-integral-derivativo PID serie.

6. Desarrollar el modelo en Simulink de la figura 4. Especificar el período de muestreo.

7. Ingrese cuatro valores para cada kp, ki y kd según las indicaciones del profesor, desarrollar la tabla 3 y la gráfica 3.

8. Repetir la simulación para el PID de mejor comportamiento variando el nivel de voltaje final del bloque step en el transcurso de la simulación y observar

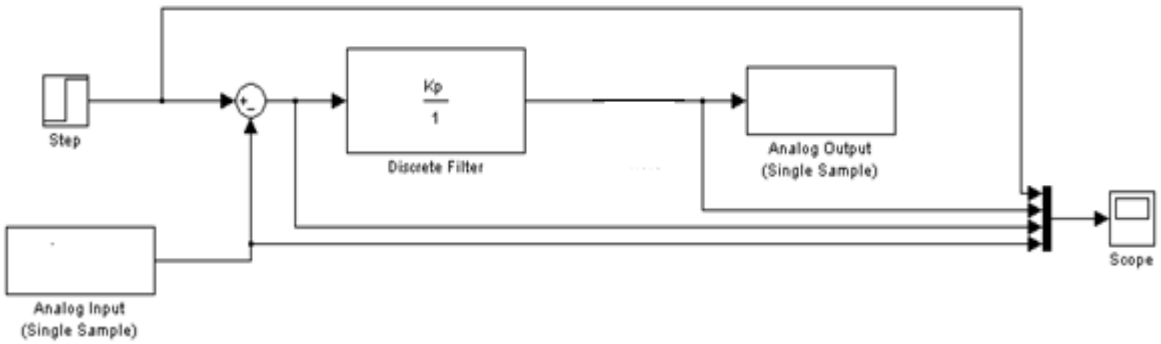


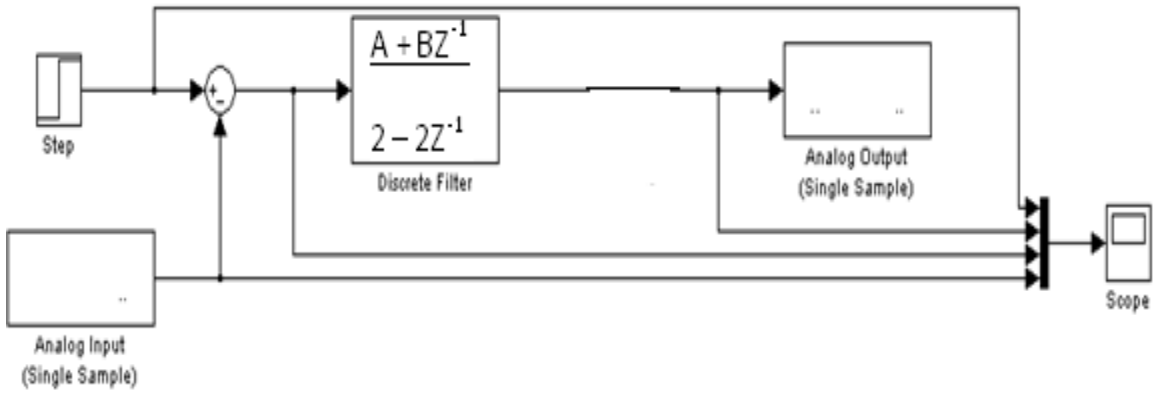
Figura 2.

Kp	1	2	3	4
Error de posición				
Overshoot (%)				
Tiempo de asentamiento				
Voltaje de alimentación				
Posición del motor				

Tabla 1.

Ep, PO y t_S				
Kp	1	2	3	4

Gráfica 1.



A=2Kp+TKi; **B=-2Kp+TKi;**

Figura 3.

Ki	1	2	3	4
Error de posición				
Overshoot (%)				
Tiempo de asentamiento				
Voltaje de alimentación				
Posición del motor				

Tabla 2.

Ep, PO y t_S				
Ki	1	2	3	4

Gráfica 2.

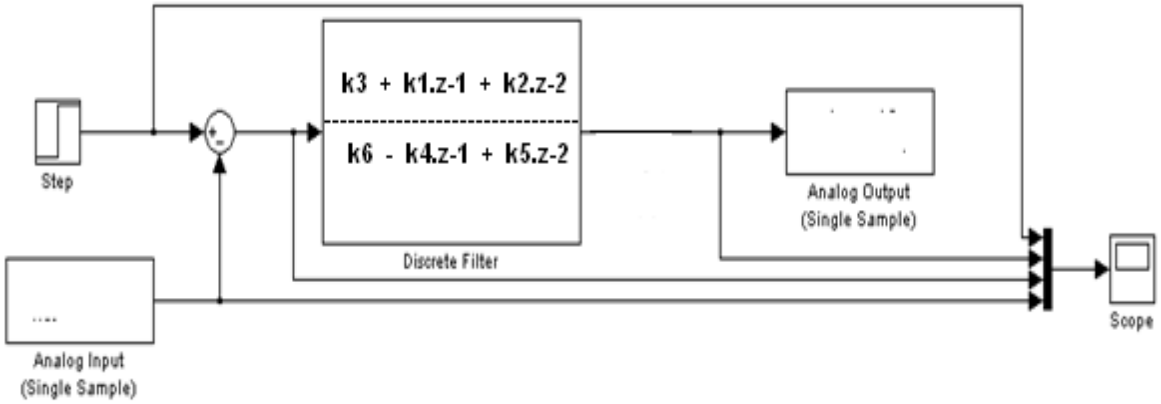


Figura 4.

Kp-ki-kd	1	2	3	4
Error de posición				
Overshoot (%)				
Tiempo de asentamiento				
Voltaje de alimentación				
Posición del motor				

Tabla 3.

Ep, PO y t_S				
Kp-ki-kd	1	2	3	4

Gráfica 3.

CONCLUSIONES

- La instalación y configuración de las tarjetas NI USB 6008/6009 son un procedimiento que garantiza el correcto funcionamiento de la toolbox de adquisición de datos.
- El objetivo de cualquier sistema de adquisición de datos es proporcionar las herramientas y recursos necesarios para tomar señales físicas y convertirlas en datos que posteriormente se puedan procesar y mostrar.
- El método de adquisición de datos con la toolbox de adquisición de datos de matlab es efectivo y de sencilla aplicación para la entrada y salida de datos digitales y análogos.
- En la adquisición de datos digitales matlab crea un vector binario que se convierte en una ventaja con respecto a otros software de aplicación para la manipulación de la información.
- Simulink es la herramienta más versátil para aprovechar la toolbox de adquisición de datos, los diagramas de bloques permiten una descripción de alto nivel del sistema de adquisición.
- El osciloscopio (Oscilloscope) es una herramienta de la toolbox de adquisición que facilita la visualización de las señales ingresadas para su medición, comparación y análisis.
- Los datos que se visualizan en una adquisición no son en tiempo real dado que factores como la memoria del computador y la velocidad de procesador afectan el tiempo de respuesta, pero es una muy buena aproximación al tiempo real.
- La tarjeta de adquisición puede es configurada de acuerdo a los requerimientos de la aplicación, en cuanto a velocidad de muestreo, tipo de canal, rangos, tiempos de disparo, acondicionadores de señal.
- La reconocida robustez y aplicabilidad de matlab y la práctica en el uso que de este software se adquiere en el transcurso de la vida universitaria sugiere optar por el uso de la toolbox de adquisición de datos de matlab como software de aplicación para los sistemas de adquisición de datos.

BIBLIOGRAFIA

- Real-Time Windows Target 2.6.1. Run Simulink models on a PC in real time.

Disponible en :
http://www.mathworks.com/products/rtwt .
Acceso : Enero 2010.

- Real-Time Windows Target Release Notes.

Disponible en :
http://www.mathworks.fr/access/helpdesk/help/pdf_doc/rtwin/rn.pdf .
Acceso : Febrero 2010.

- Data Acquisition Toolbox Quick Reference Guide

Disponible en :
http://www.mathworks.fr/access/helpdesk/help/pdf_doc/daq/daqquickref.pdf
Acceso: Noviembre 2009

- Data Acquisition Toolbox Release Notes

Disponible en :
http://www.mathworks.fr/access/helpdesk/help/pdf_doc/daq/rn.pdf
Acceso: Noviembre 2009

- Data Acquisition Toolbox Adaptor Kit 2

Disponible en :
http://www.mathworks.fr/access/helpdesk/help/pdf_doc/daq/adaptorkit.pdf
Acceso: Septiembre 2009

- Demo Matlab & Simulink

Disponible en :

http://www.mathworks.fr/products/daq/demos.html
Acceso: Septiembre 2009

- Ejemplos de la toolbox de adquisición de datos

Disponible en :
http://www.mathworks.fr/access/helpdesk/help/toolbox/daq/exampleindex.html
Acceso: Agosto 2009

- Adquisición de Datos usando Matlab

Bruno Vargas Tamani
Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú.

- Tutorial de MatLab Simulik

Hender Molina - Lisbeth Román
Disponible en :
http://www.angelfire.com/la/hmolina/matlab7.html
Acceso: Febrero 2010

- Data adquisition toolbox

Matlab R2008b
Acceso: Help – Navigator – Contents – Data Adquisition Toolbox

- Data Acquisition Toolbox DEMOS

Matlab R2008b
Acceso: Help – Navigator – Demos – Toolboxes – Data Adquisition

- Adquisición de Audio usando MATLAB

Juan Carlos Moctezuma Eugenio
jcmoctezuma@ccc.inaoep.mx
Sep – 2007
Corrección parte 1 del libro

- User guide and specifications USB-6008/6009

Disponible en :

http://www.tau.ac.il/~electro/pdf_files/computer/ni_6008_ADC_manual.pdf

Acceso: Agosto 2009