UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA AUTOMATIZADO DE CALIBRACIÓN DE SENSORES DE DESPLAZAMIENTO, FUERZA Y ACELERACIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO ELÉCTRICO – ELECTRÓNICO
PRESENTA
IVÁN LÓPEZ PINEDA LÓPEZ

Dirigida por:
Dr. Luis A. Álvarez Icaza Longoria

CIUDAD UNIVERSITARIA, MÉXICO, D.F., 2006

Agradecimientos

A mis padres: El C. P. Francisco López Pineda y la C. P. Georgina López Toledo.

A mis hermanos: Luis, Juan Carlos, Francisco y Elia Guiexhuba.

A mi cuñada: Concepción.

A mi sobrina: Xunaxi Natasha.

A toda mi familia.

Al Dr. Luis A. Álvarez Icaza Longoria, al Mtro. Rolando A. Carrera Méndez, al Mtro. René E. Jiménez Fabián y a Laura Mayanin de la Subdirección de Electromecánica del Instituto de Ingeniería.

Al Mtro. Roberto Durán Hernández, al Ing. Miguel Ángel Mendoza García, al Mtro. Abraham Roberto Sánchez Ramírez, al Ing. Eddy Marcel Grandry Carreyn de la Subdirección de Estructuras y al Mtro. Victor Franco de la Subdirección de Hidráulica y Ambiental del Instituto de Ingeniería.

A mis sinodales: Ing. Rodolfo Peters Lammel, Mtro. Juan Manuel Gómez González, Ing. Norma Elva Chávez Rodríguez e Ing. José Salvador Zamora Alarcón.

A Anabel.

A mis amigos y compañeros: Diana, Victor, Erik, Mario Alberto, Fernando, Giovanni Andrés, Grisel, Eduardo, Julio César, Carlos Humberto, Graciela, Jazmín, Marco Antonio, Roberto Mario, Martín, María Ana y Adán.

A la Facultad de Ingeniería, al Instituto de Ingeniería y a la Universidad Nacional Autónoma de México, nuestra Máxima Casa de Estudios.

Gracias a todos ustedes que hicieron esto posible.

Índice general

Índice de figuras

Capítulo 1

Introducción

En el Instituto de Ingeniería de la UNAM se desarrollan algoritmos de control basados en mediciones de fuerza y aceleración para reducir las vibraciones sísmicas en estructuras civiles. Las pruebas de estos algoritmos se realizan simulando movimientos telúricos en la mesa vibradora del mismo Instituto con un modelo a escala de un edificio de cinco pisos (Figura 1.1) fijado por la base a la mesa. Dispuesto horizontalmente y fijado en un extremo a la base y en el otro al primer nivel del edificio se encuentra un amortiguador magneto-reológico. Dicho amortiguador tiene acoplada una celda de carga que mide la fuerza ejercida por éste. Seis acelerómetros atornillados al edificio cuantifican la aceleración de la base y de cada piso. De forma complementaria se registran los desplazamientos de cada uno de los pisos por medio de cinco LVDT's fijados a una estructura de soporte en la misma mesa a un lado del modelo a escala. Todos estos sensores se encuentran conectados mediante cajas de conexiones a un par de tarjetas de adquisición de datos instaladas en una computadora industrial, que corre un programa en tiempo real para determinar la señal de control aplicada al amortiguador. La fuerza que ejerce éste último varía en función de la corriente que circula a través del fluido contenido en el mismo, disipando energía de forma controlada.

Es necesario realizar una calibración periódica de los sensores para poder tener

Figura 1.1: Modelo a escala en la mesa vibradora

confianza en los resultados de las pruebas. Este proceso consiste en encontrar la relación existente, expresada como una constante de calibración, entre la entrada y la salida del sensor. Para ello se registra la magnitud física que se aplica al sensor en cantidades conocidas, dentro del rango y en las condiciones en las que trabajará en las pruebas, y la correspondiente salida en volts. Con una serie de estos datos se calcula la ecuación de la recta que correlaciona ambas variables. Si resulta un buen ajuste, la constante de calibración es un factor por el que hay que multiplicar la salida de tensión del sensor para encontrar el valor de la magnitud física aplicada al mismo.

El procedimiento hecho a mano lleva entre 15 y 20 minutos dependiendo del tipo de sensor. Se requiere que una persona observe la lectura de un multímetro conectado a la salida del sensor y que registre los pares de mediciones a mano en papel. Después se necesita utilizar una calculadora o una computadora para calcular la ecuación de la recta por el método de mínimos cuadrados y es indispensable comparar los valores correspondientes a la ordenada al origen y el coeficiente de correlación lineal para de-

terminar si la calibración puede ser expresada sólo por la pendiente de la recta y si el ajuste es adecuado.

En esta tesis se desarrolla el Sistema Automatizado de Calibración de Sensores (SACS) como una ayuda en el proceso de determinación de la constante de calibración de los sensores de fuerza, aceleración y desplazamiento. Se espera que con la infraestructura disponible se cuente con un programa fácil de usar, que se reduzca el tiempo necesario para calibrar un sensor, que se analicen los datos con criterios preestablecidos, que se eviten errores humanos en los cálculos y además que se provea un registro de calibraciones.

Para probarlo se montará el sensor en el arreglo adecuado para llevar a cabo una calibración estática y su salida se conectará a un canal de la tarjeta de adquisición de datos. Por medio de la interfaz gráfica programada en MATLAB, el usuario proporcionará los datos que identifican al sensor (tipo, modelo y número de serie). Después se suministrará al sensor una cantidad conocida de la magnitud física que mide y se indicará la misma al programa. El programa en ese momento leerá la salida del sensor y la registrará junto con la magnitud especificada por el usuario. El par de datos se graficará inmediatamente y cuando se cuenta con al menos dos puntos, el programa podrá graficar la recta calculada por mínimos cuadrados junto con los puntos que la determinan.

A continuación se presenta una breve descripción del contenido de cada capítulo:

- Capítulo 1. Esta introducción.
- Capítulo 2. Se detalla el principio de funcionamiento de cada uno de los sensores utilizados.
- Capítulo 3. Se describe el proceso mediante el cual se puede calibrar un sensor.

- Capítulo 4. Se explica el diseño y la implementación del SACS.
- Capítulo 5. Se refieren las pruebas realizadas con el sistema.
- Capítulo 6. Se presentan las conclusiones.
- Apéndice A. Se muestran las especificaciones del equipo utilizado.
- Apéndice B. Se lista el código del programa desarrollado.

Capítulo 2

Sensores

En este capítulo se describe el funcionamiento de los sensores LVDT, celda de carga y acelerómetro[1] .

2.1. LVDT

El LVDT (Linear Variable Differential Transformer, Transformador diferencial variable lineal) es uno de los dispositivos más útiles para convertir un desplazamiento mecánico en una señal eléctrica.

Consiste en un carrete hueco no magnético sobre el cual están dispuestos un arrollamiento central primario y dos arrollamientos exteriores secundarios conectados en

[1] Ver especificaciones de los sensores utilizados en el Apéndice A.

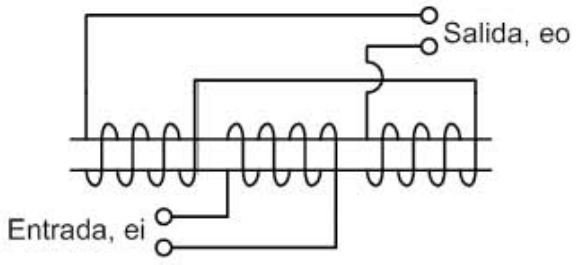

Figura 2.1: LVDT

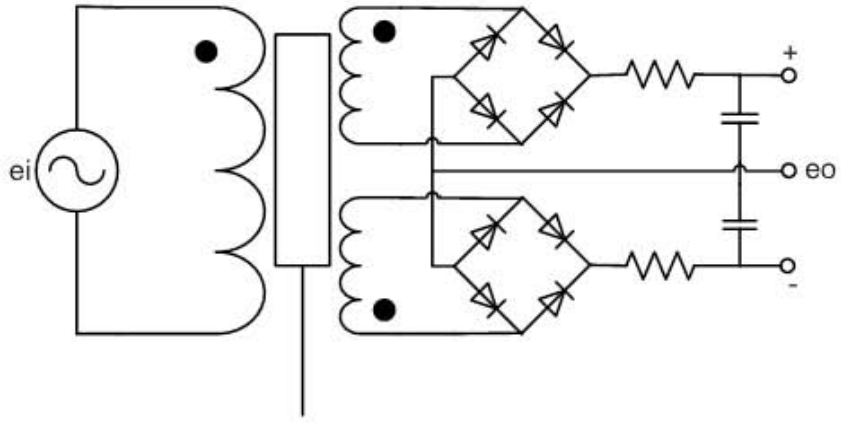

Figura 2.2: Circuito para LVDT

serie-oposición, dispuestos como se muestra en la Figura 2.1. El embobinado primario es energizado con una tensión sinusoidal de entre 3 y 15 [V] de amplitud RMS y una frecuencia de 60 a 20,000 [Hz]. De acuerdo a la posición del núcleo varía la amplitud de las tensiones inducidas en los secundarios.

El núcleo está formado por un cilindro de material magnético que puede desplazarse axialmente. Si dicho núcleo está perfectamente centrado no aparecerá tensión entre los extremos del secundario, ya que en ambas mitades del secundario se inducen tensiones iguales y de polaridades opuestas.

Si el núcleo se desplaza a partir del punto de equilibrio, aumentará el coeficiente de acoplamiento entre el primario y la mitad del secundario a la cual se acerca y disminuirá simultáneamente el mismo coeficiente respecto a la mitad de la que se aleja. De este modo, se inducirá más tensión en una de las mitades y menos tensión en la otra. Dependiendo del sentido de la desviación, esta tensión tendrá la misma fase que la señal aplicada al primario, o la fase opuesta [2].

Si las terminales de los arrollamientos secundarios son independientes, puede utilizarse el circuito demodulador sensible a la fase y red de filtrado RC de la Figura 2.2, para tener una tensión de CD en la salida proporcional al desplazamiento del núcleo y

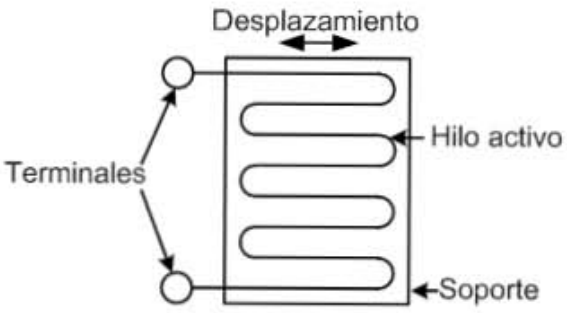

Figura 2.3: Galga extensométrica

con una polaridad que indica el sentido del mismo [1].

2.2. Celda de carga

La celda de carga funciona a partir de galgas extensométricas (*strain gages*) y se emplea en la medición de fuerzas.

Las galgas extensométricas detectan la deformación elástica de un elemento resistivo incorporado en una estructura mecánica de soporte como un cambio en el valor de su resistencia originado por las modificaciones geométricas producidas por dicha deformación.

Consisten en un hilo metálico de pequeño diámetro (del orden de 0.01 a 0.1 [mm]) embebido en una lámina muy delgada de un material de soporte (papel, plástico, etc.) dentro del cual sigue un trazado ondulante o en zig-zag.

De acuerdo con su disposición, el transductor extensométrico simple de la Figura 2.3 solo será sensible a esfuerzos dirigidos según la doble flecha.

Se encuentran captadores comerciales con dimensiones entre 1 x 1 [mm] y 100 x 50 [mm] con un espesor típico de 0.2 [mm]. La gama de valores de resistencias incluye valores de 100 a 1000 [Ω]. También se fabrican con técnicas de fotograbado de circuito impreso o con materiales semiconductores. Algunos pueden sensar dos o tres direcciones

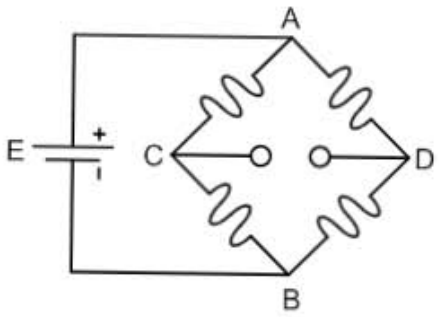

Figura 2.4: Puente de Wheatstone en celda de carga

de deformaciones [2].

En una celda de carga se encuentran adheridas cuatro galgas extensométricas en una columna cilíndrica o rectangular, de modo que cuando se aplica una carga, se deforman variando proporcionalmente su resistencia.

Se utiliza regularmente un puente de Wheatstone (Figura 2.4) totalmente compensado con los cuatro brazos activos, donde el diseño mecánico es tal que la aplicación de un esfuerzo hace crecer las resistencias de los brazos AD y CB y hace disminuir la resistencia de las galgas dispuestas entre AC y DB. Si las galgas son idénticas, variarán su resistencia en la misma medida por la variación de la temperatura. Por esto la variación de la resistencia de las galgas conectadas en puente de Wheatstone es independiente de la temperatura. La gama de medida está determinada por la construcción de la columna (desde varios kilogramos hasta algunas toneladas).

Las señales derivadas de los transductores extensométricos son en general muy débiles y requieren amplificación. Algunas celdas de carga comerciales tienen incluida una etapa de acondicionamiento de señal, de modo que pueden ser conectadas directamente a los sistemas de adquisición de datos. Los amplificadores utilizados (usualmente amplificadores operacionales) deberán ser de altas prestaciones en lo que se refiere a tensión de desviación (*offset*) y deriva.

En la Figura 2.5 se ilustran dos esquemas de amplificación. En la parte superior se

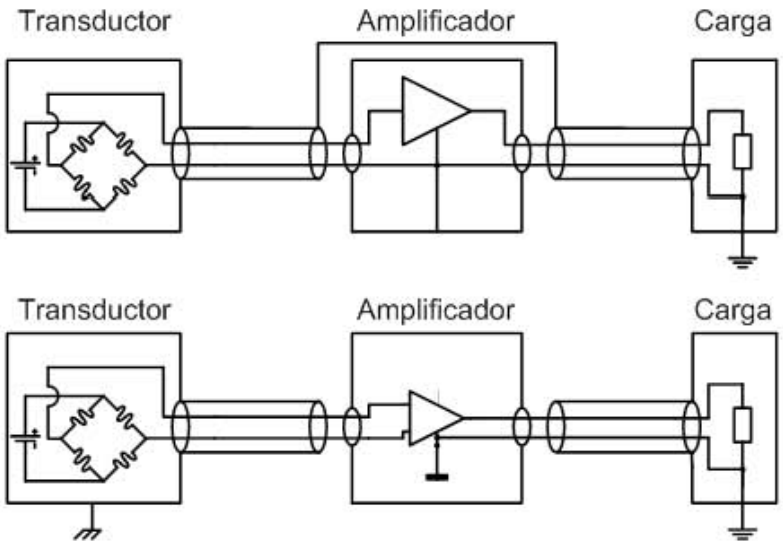

Figura 2.5: Amplificación para celda de carga

muestra un amplificador no diferencial, adecuado en los casos en que el transductor es flotante, es decir, no tiene conexión a tierra, mientras que en la parte inferior se utiliza un amplificador diferencial para los casos en que el transductor tiene, por construcción, una conexión a tierra [2].

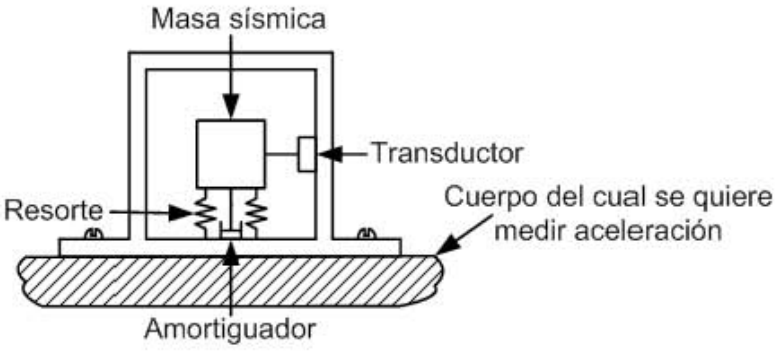

Figura 2.6: Acelerómetro de masa sísmica

2.3. Acelerómetro

Con un acelerómetro es posible tener una tensión proporcional a la aceleración absoluta aplicada al cuerpo al que está fijado.

En la Figura 2.6 se ilustra esquemáticamente un acelerómetro de masa sísmica. Consta de un transductor de desplazamiento, una masa de prueba apoyada en uno (o varios) resortes y acoplada a un amortiguador. Al someterse a una aceleración, la masa se desplaza en sentido contrario al de la aceleración y el(los) resorte(s) ejerce(n) una fuerza sobre ella, proporcional a la aceleración. Como la fuerza en un resorte lineal es proporcional al desplazamiento, basta con medir la posición relativa de la masa con el transductor para tener una señal porporcional a la aceleración absoluta.

Mediante la selección adecuada de la frecuencia natural y del amortiguamiento, es posible diseñar el instrumento sísmico de modo que el desplazamiento relativo entre la masa y la carcasa sea una función de la aceleración en una banda de frecuencias determinada [1].

La dependencia de la masa trae problemas porque una masa también experimenta fuerzas debido a campos gravitacionales. Entonces un acelerómetro no puede distinguir entre una fuerza debida a una aceleración y una fuerza debida a la gravedad [4].

Los acelerómetros de balance de fuerza o servo acelerómetros funcionan a partir de

un electroimán, un sensor de posición, un amplificador de error y un convertidor de corriente en tensión.

Cuando la masa comienza a moverse a partir de la posición central debido a una aceleración externa, el sensor de posición genera una señal de error que es amplificada para producir una corriente de polaridad correspondiente que se hace circular por la bobina del electroimán, el cual regresa la masa a su posición original. La corriente que se aplica al electroimán para restaurar a la masa es directamente proporcional a la aceleración que sufre la misma, dicha corriente se hace pasar por una resistencia en serie de donde se obtiene la señal de tensión de salida [3].

Capítulo 3

Calibración

En esta parte se describe la forma en que se realiza la calibración estática de los sensores.

La calibración estática consiste en variar solo una de las entradas del sensor en un rango de valores constantes, procurando que el resto de las entradas no cambie su valor. Este cambio en la entrada en estudio provoca que la salida varíe en un rango de valores. Si se desea un ajuste lineal, se registran los valores de entrada así como los correspondientes valores de salida para obtener la ecuación de la recta que mejor representa el comportamiento del sensor. Se utiliza el método de mínimos cuadrados y se acepta un coeficiente de correlación mayor o igual a 0.95. Si la ordenada al origen de la recta obtenida es menor al 1 % de la salida máxima, entonces puede ser despreciada y la pendiente de la recta representará la constante de calibración buscada.

Las relaciones entrada–salida desarrolladas de esta manera constituyen una calibración estática válida cuando las otras entradas permancecen constantes.

Es importante para la calibración contar con un instrumento estándar con una precisión al menos diez veces mayor que la del sensor a calibrar [4].

Para tener mediciones óptimas es recomendable hacer la calibración en el rango de valores en donde variará la entrada, con los cables con los que se operará el instrumento

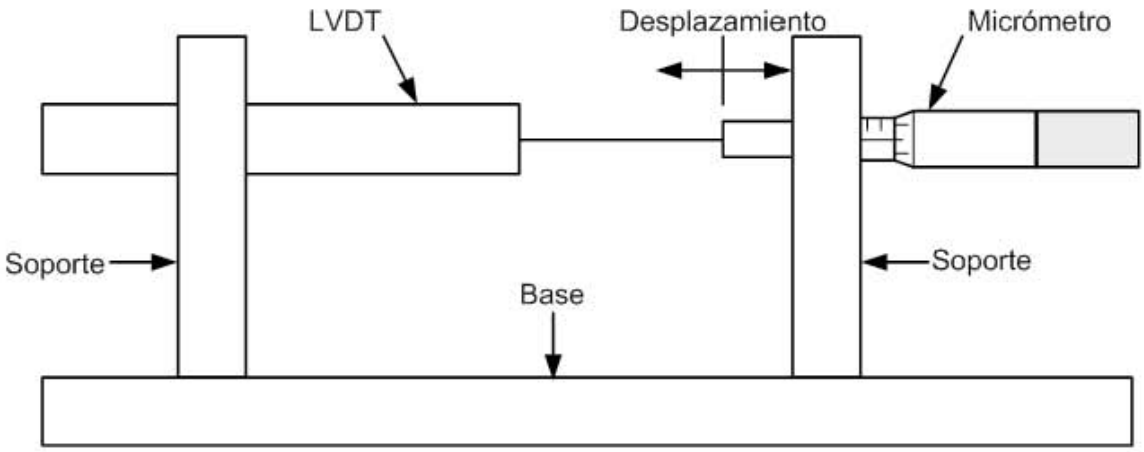

Figura 3.1: Dispositivo para calibrar LVDTs

y proveyendo un tiempo de precalentamiento.

3.1. LVDT

La calibración estática de los instrumentos de medición de desplazamiento se lleva a cabo satisfactoriamente utilizando micrómetros como estándar. Cuando son usados para medir directamente el desplazamiento del transductor, son adecuados para leer incrementos de hasta 0.01 [mm].

Se utiliza un dispositivo mecánico donde se asegura el LVDT a calibrar y un micrómetro, de modo que puedan acoplarse la varilla del núcleo del LVDT con el vástago móvil del micrómetro (Figura 3.1). El sensor es polarizado con una tensión, dentro de su rango de operación, suministrada por una fuente variable regulada. El valor de dicha tensión es ajustado de acuerdo a la lectura de un multímetro.

Una vez polarizado el LVDT, se conecta un multímetro a su salida y se coloca el micrómetro exactamente a la mitad de su escala. Después se busca la posición del sensor que da 0 [V] a la salida y se fija firmemente.

Empezando desde uno de los extremos se van registrando los valores de desplaza-

miento y de tensión para cada uno de los incrementos de la posición del micrómetro.

Con los datos obtenidos se calcula la ecuación de la recta por el método de mínimos cuadrados, tomando como variable independiente al desplazamiento y como variable dependiente a la tensión.

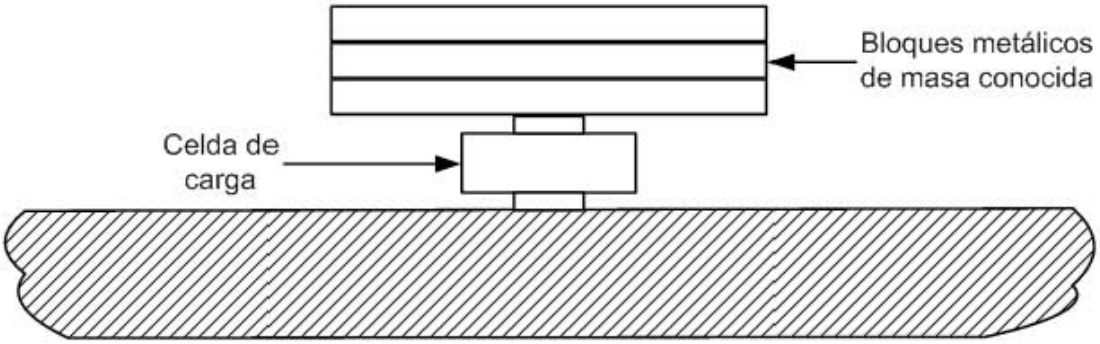

Figura 3.2: Arreglo para calibrar celdas de carga

3.2. Celda de carga

Como patrones de calibración se emplean bloques metálicos cuya masa ha sido previamente medida en una báscula apropiada y marcada sobre ellos.

La celda de carga es colocada sobre una superficie plana y firme y es polarizada con una fuente variable regulada, que se ajusta a una tensión adecuada usando un multímetro (Figura 3.2).

Los bloques metálicos son apilados uno por uno sobre el sensor. Se anota en cada paso el valor total de la masa colocada sobre él, así como su tensión de salida.

Otro método factible es usar un marco de carga que cuente con un sensor de referencia. Se acoplan ambos sensores a la misma máquina y se va aplicando una fuerza conocida paso a paso. Se va registrando la fuerza aplicada y la tensión de salida que corresponde.

Considerando a la fuerza como variable independiente y a la tensión de salida como variable dependiente se encuentra, utilizando el método de mínimos cuadrados, la ecuación de la recta característica de la celda de carga.

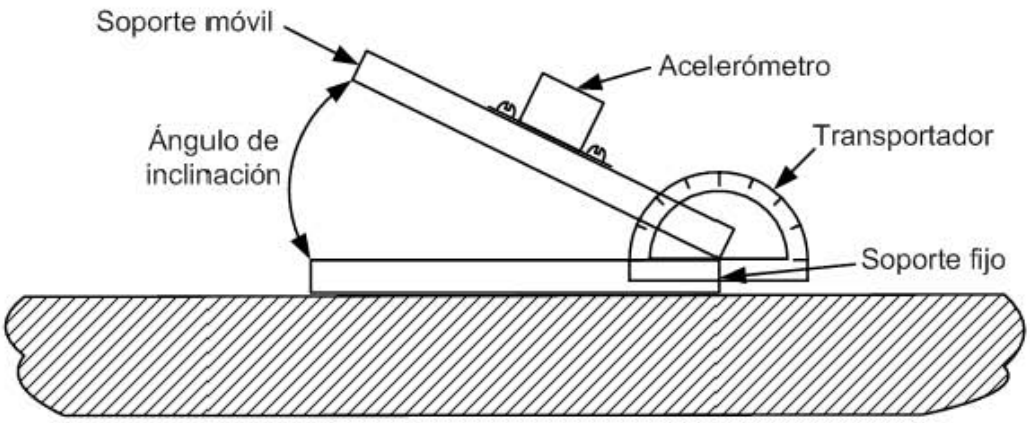

Figura 3.3: Dispositivo para calibrar acelerómetros

3.3. Acelerómetro

La calibración estática de acelerómetros en el rango de ±1 [g] se realiza apropiadamente fijando el acelerómetro a un soporte inclinable, cuyo ángulo con la vertical es medido con exactitud (Figura 3.3). Es indispensable conocer el valor de la aceleración local debida a la gravedad [17].

Para que el resultado sea una ecuación lineal es necesario obtener una tabla de ángulos (θ), a los cuales corresponde un incremento constante ($fracg$) de la componente de la aceleración debida a la gravedad que actúa sobre la masa sísmica del sensor, mediante la fórmula: $\theta = 360^{o}$ - $\frac{\arccos(fracg)\cdot 180}{\pi}$

Se fija el sensor al soporte inclinable y se polariza con la tensión especificada por el fabricante. El ángulo de inclinación de la superficie sobre la cual se coloca el sensor y la tensión de salida correspondiente son registrados para cada inclinación aplicada hasta completar la escala del transportador.

Con el valor local de la aceleración debida a la gravedad (g) se calcula, para cada ángulo registrado (θ), la aceleración (a) aplicada en el eje de sensado del acelerómetro mediante la ecuación: $a = g \cdot \cos(\theta)$.

Se determina entonces la ecuación de la recta que describe su comportamiento por

mínimos cuadrados, con la aceleración como variable independiente y la tensión como variable dependiente.

Capítulo 4

Desarrollo del sistema

En este capítulo se describe el diseño del SACS[1] y su implementación.

4.1. Diseño

Para diseñar una interfaz de interacción entre máquinas y humanos es necesario considerar una serie de aspectos fundamentales para conseguir que sea eficaz [8]. Es indispensable definir quién usará la interfaz para seleccionar el vocabulario que ha de emplearse. Los usuarios potenciales de este sistema serán estudiantes, ya sea de licenciatura o de posgrado, o investigadores con conocimientos básicos de instrumentación, calibración, adquisición de datos y MATLAB[2] . Necesitan conocerse los problemas que resolverá el uso del programa. Al utilizarlo el usuario, en contraste con una calibración manual, podrá visualizar progresivamente de forma gráfica los datos obtenidos, evitará hacer cálculos que pueden involucrar errores y que requieren de mucho tiempo y además contará con un registro detallado de la calibración. Se debe priorizar la simplicidad de las tareas más comunes que se pueden realizar con el programa, hay que

[1] **S**istema **A**utomatizado de **C**alibración de **S**ensores.

[2] Se ha elegido MATLAB y sus complementos como la plataforma de desarrollo debido a que la mayoría de las computadoras del Instituto de Ingeniería cuenta con este conjunto de programas y para aprovechar la experiencia que adquirí durante mi servicio social.

conocer los pasos que involucra dicha tarea e identificar todos los elementos necesarios para llevarla a cabo. La calibración de sensores (obtención de la constante de calibración) es la actividad principal que se hará con ayuda del programa, aunque también deberá poder abrir archivos de registro. Los pasos a seguir para calibrar un sensor son:

- hacer las conexiones de señales y alimentación,
- en caso necesario, registrar la tensión de desviación que será restada a todas las mediciones,
- registrar uno a uno pares de valores de entrada y salida del sensor y
- calcular la constante de calibración, la tensión de desviación y el coeficiente de correlación.

El equipo[3] necesario para efectuar la calibración es:

- computadora personal corriendo el SACS,
- tarjeta de adquisición de datos,
- caja de conexiones,
- fuente de corriente directa para alimentación,
- dispositivo de calibración[4] adecuado,
- sensor a calibrar,
- cables y conectores.

[3] Ver especificaciones en el Apéndice A.
[4] Ver Capítulo 3.

Es de gran importancia enfocar el diseño en las funciones del programa, antes que en la apariencia; tomando en cuenta el conjunto de conceptos, datos, opciones y controles que manejará. Por lo tanto se clasifican los datos en tres categorías: datos del sensor, puntos de calibración y resultados, cada categoría será dotada de los controles apropiados. También debe ser en general entendible, fácil de aprender y consistente, de modo que su utilización no provoque más problemas que los que la tarea misma ya representa. Ésto es, no debe requerir que el usuario efectúe tareas no naturales, el despliegue de los controles debe estar de acuerdo con el paso del proceso que se está desarrollando, no deben desplegarse controles que no se utilicen y se deben manejar los mismos vocablos en todas y cada una de las partes del sistema. Es conveniente presentar la información como un resultado del análisis de los datos, en lugar de una serie de éstos sin explicaciones. El programa evaluará la bondad del ajuste, informando al usuario sólo si es o no satisfactoria. Por último, hay que brindarle una retroalimentación perceptual al usuario. De este modo sabrá que es necesario esperar mientras se adquieren datos o mientras se hacen cálculos.

La aplicación de todos estos principios debe ser evaluada mediante pruebas de utilidad[5] con usuarios potenciales que opinen libremente acerca del sistema mientras lo usan por primera vez.

4.1.1. Flujo de datos

La información necesaria para realizar y registrar la calibración es la siguiente:

- El tipo de sensor (proporcionado por el usuario),
- el modelo del sensor (proporcionado por el usuario),
- el número de serie del sensor (proporcionado por el usuario),

[5] Ver Capítulo 5

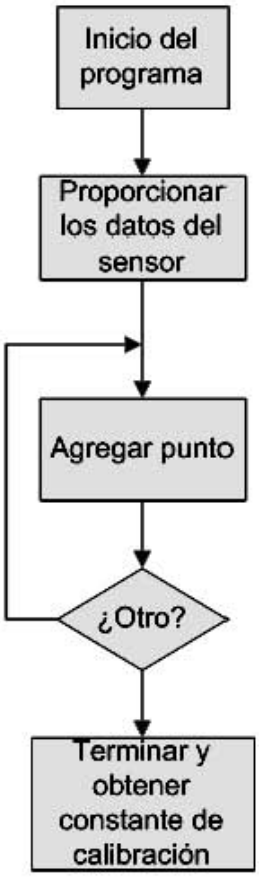

Figura 4.1: Diagrama de flujo

- una serie de valores de entrada del sensor (proporcionada por el usuario) con su correspondiente salida (adquirida directamente por la computadora a través de la tarjeta de adquisición de datos) y
- la fecha (obtenida de la computadora).

En la Figura 4.1 aparece un diagrama de flujo del proceso de calibración automatizada. Los datos del sensor deberán ser introducidos al principio, ya que las unidades de la magnitud física de entrada desplegadas en el programa dependen del tipo de sensor que se está calibrando.

Para que el usuario cuente con una retroalimentación visual inmediata de los puntos de la calibración en el plano coordenado, los valores de la magnitud física de entrada del sensor se ubicarán en el eje vertical y la tensión de salida del sensor en el horizontal. Gracias a esta convención se contará al finalizar con una constante de calibración, tal que al ser multiplicada por la salida del sensor en volts, se encontrará el valor de la magnitud física de entrada.

Cuando el usuario indique que ha terminado de introducir datos, se desplegará la constante de calibración resultante. También se informará si ésta es aceptable o no, con base en el valor del coeficiente de correlación obtenido. En ese momento puede generarse un archivo de registro en la ubicación indicada por el usuario.

4.2. Implementación

Para implementar el SACS se requirió del siguiente software:

- Sistema operativo Microsoft Windows XP Versión 5.1
  (Build 2600: Service Pack 2)
- MATLAB Versión 7.0.4.365 (R14) Service Pack 2
- GUIDE Versión 7
- Simulink Versión 6.2
- Curve Fitting Toolbox Versión 1.1.3
- Real-Time Windows Target Versión 2.5.2
- Real-Time Workshop Versión 6.2
- Lcc C Compiler Versión 2.4

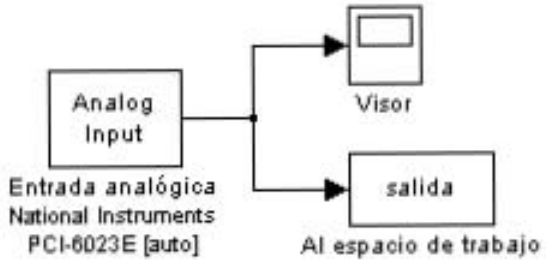

Figura 4.2: Modelo de Simulink para adquirir datos

El sistema se realizó con ayuda del ambiente de desarrollo de interfaces gráficas de MATLAB, llamado GUIDE (Graphical User Interface Development Environment) [10]. En GUIDE se colocan todos los elementos de la interfaz como se verán cuando se abra y con el editor de archivos M (M-File Editor) se especifican las subrutinas (*callbacks*) que se ejecutan al efectuar alguna acción sobre los controles. De esta forma es posible aprovechar la funcionalidad desde la línea de comandos de MATLAB [12], Curve Fitting Toolbox [11] y Simulink [15]. El programa consta al menos de dos archivos: un archivo con extensión *.fig*, con información sobre la disposición de los controles en la ventana de la interfaz y otro con extensión *.m*, con las instrucciones para realizar cada operación posible con ella. Se pueden requerir además imágenes y archivos de configuración. Para la adquisición de datos se utiliza un archivo *.mdl* y el código de tiempo real contenido en una sub-carpeta.

4.2.1. Adquisición de datos

La medición de la tensión de salida de los sensores se lleva a cabo por medio de una tarjeta de adquisición de datos National Instruments PCI-6023E[6] a través de un modelo de Simulink (Figura 4.2) que incluye el bloque de entrada analógica perteneciente a Real-Time Windows Target (RTWT) [13]. También contiene un bloque que genera una

[6] Ver especificaciones en el Apéndice A.

variable de salida en el espacio de trabajo para hacerla disponible a la interfaz y un visor para comprobar que funciona.

Real-Time Workshop (RTW) [14] es utilizado para generar el código que será compilado por el Lcc C Compiler para crear el programa que se ejecutará en tiempo real para la adquisición de datos.

Es necesario hacer las siguientes configuraciones para poder adquirir datos en tiempo real:

1. En la línea de comandos ejecutar "mex -setup". Responder "y"para localizar los compiladores instalados. Escribir el número que corresponda al "Lcc C Version 2.4" y confirmar con "y". Así queda seleccionado el compilador del código de tiempo real.

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc C version 2.4 in C:\MATLAB7\sys\lcc

[0] None

Compiler: 1

Please verify your choices:

Compiler: Lcc C 2.4
Location: C:\MATLAB7\sys\lcc

Are these correct?([y]/n): y
```

```
Try to update options file: C:\Doc...\MathWorks\MATLAB\R14\mexopts.bat
From template:              C:\MATLAB7\BIN\WIN32\mexopts\lccopts.bat
Done . . .
```

2. Ejecutar también en la línea de comandos "rtwintgt -setup" y confirmar con "y"para instalar (o reinstalar) el núcleo (*kernel*) de Real-Time Windows Target. Dicho núcleo es necesario para ejecutar el programa en tiempo real.

```
>> rtwintgt -setup

You are going to install the Real-Time Windows Target kernel.
Do you want to proceed? [y] : y
The Real-Time Windows Target kernel has been successfully installed.
```

3. En el cuadro de diálogo de los parámetros del bloque de entrada analógica (Figura 4.3) se elige la tarjeta de adquisición de datos, un tiempo de muestreo de "0.01"[s], que equivale a 100 muestras por segundo, el canal de entrada[7] número "1", el rango de entrada de "-10 a 10"[V][8] y "Volts" como el tipo de señal de salida del bloque. El tiempo de muestreo es el adecuado por tratarse de señales de CD. En dicho canal se conectará la señal de salida del sensor, que tomará valores en el rango especificado. Los datos deben estar en Volts, puesto que la constante de calibración involucra esta unidad.

4. Para el bloque de salida "Al espacio de trabajo"(Figura 4.4) se indica "salida" como el nombre de la variable, "100" como la cantidad de puntos que se registrarán y "Array"(arreglo) como el formato en el que se guardará. Con esto se cuenta con

[7] Equivalente al canal de entrada número 0 (AI0 o ACH0) de la tarjeta.

[8] Para tener una mayor resolución se elige "0 a 5"[V] cuando se está seguro que la señal no excederá 5 [V] en ningún momento, como en el caso de los acelerómetros y las celdas de carga.

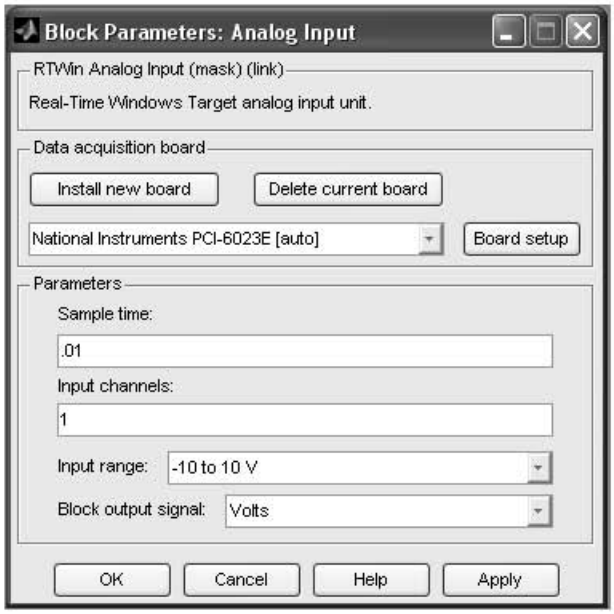

Figura 4.3: Parámetros del bloque de entrada analógica

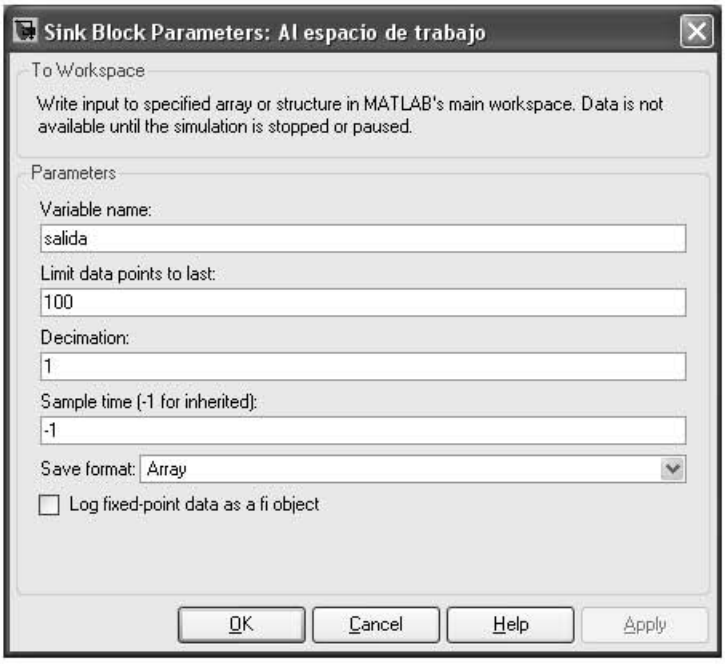

Figura 4.4: Parámetros del bloque Al espacio de trabajo

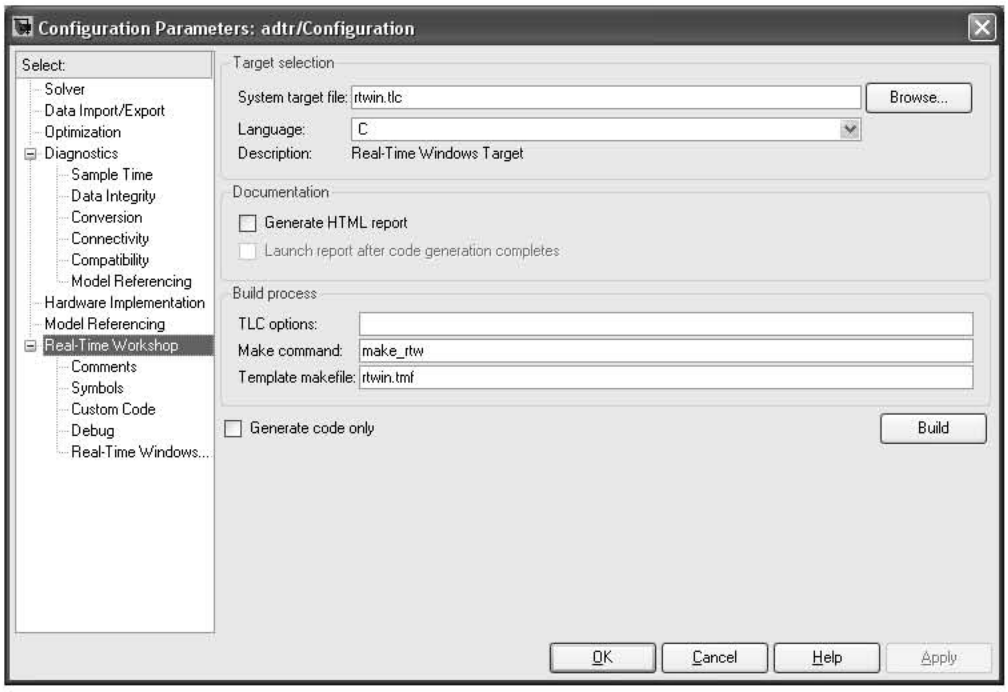

Figura 4.5: Opciones de Real-Time Workshop

el resultado de la digitalización en el espacio de trabajo para ser promediado por el programa de la interfaz.

5. La configuración de RTW (Menú Tools→Real-Time Workshop→Options...) debe quedar como se muestra en la Figura 4.5. Para esto hay que oprimir el botón "Browse..." y elegir "rtwin.tlc" como el archivo de destino (Figura 4.6). Así se le indica a RTW que los datos serán utilizados por RTWT.

6. Oprimir el botón "Signal & Triggering"del panel de control del modo externo (Figura 4.7) de Simulink (Menú Tools→External Mode Control Panel...) y escribir "100" en el campo de duración de la ventana "External Signal & Triggering" (Figura 4.8). Ésto define la cantidad de lecturas que se envían juntas a MATLAB desde el programa de tiempo real.

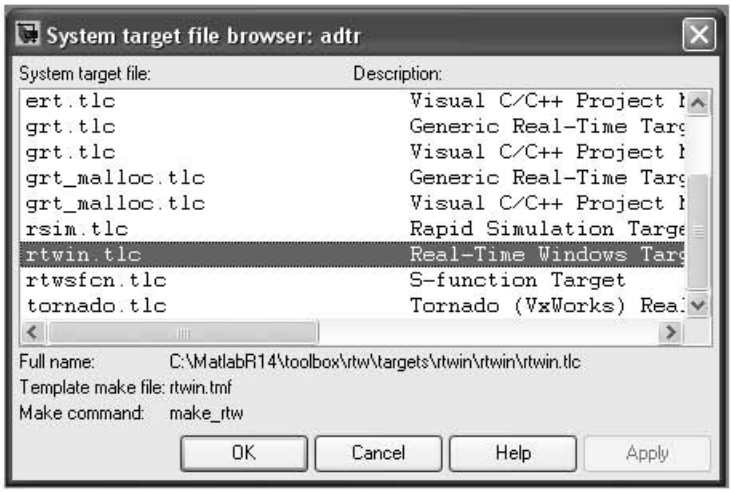

Figura 4.6: Selección del destino para Real-Time Workshop

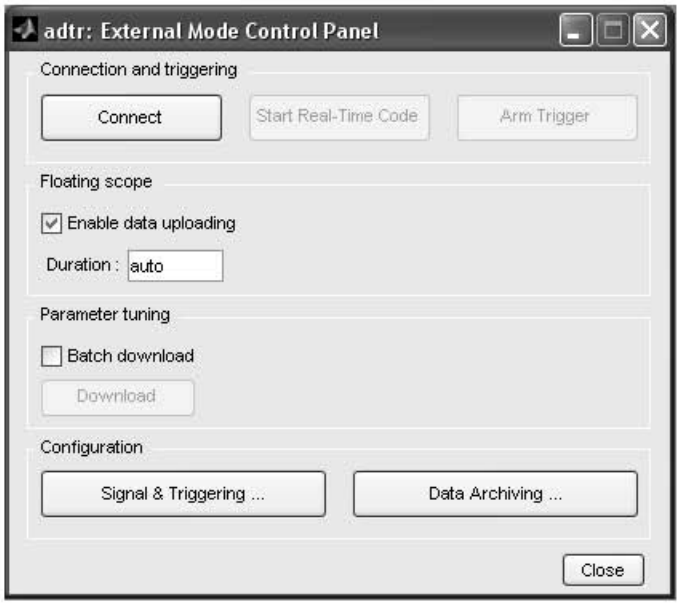

Figura 4.7: Panel de control del modo externo

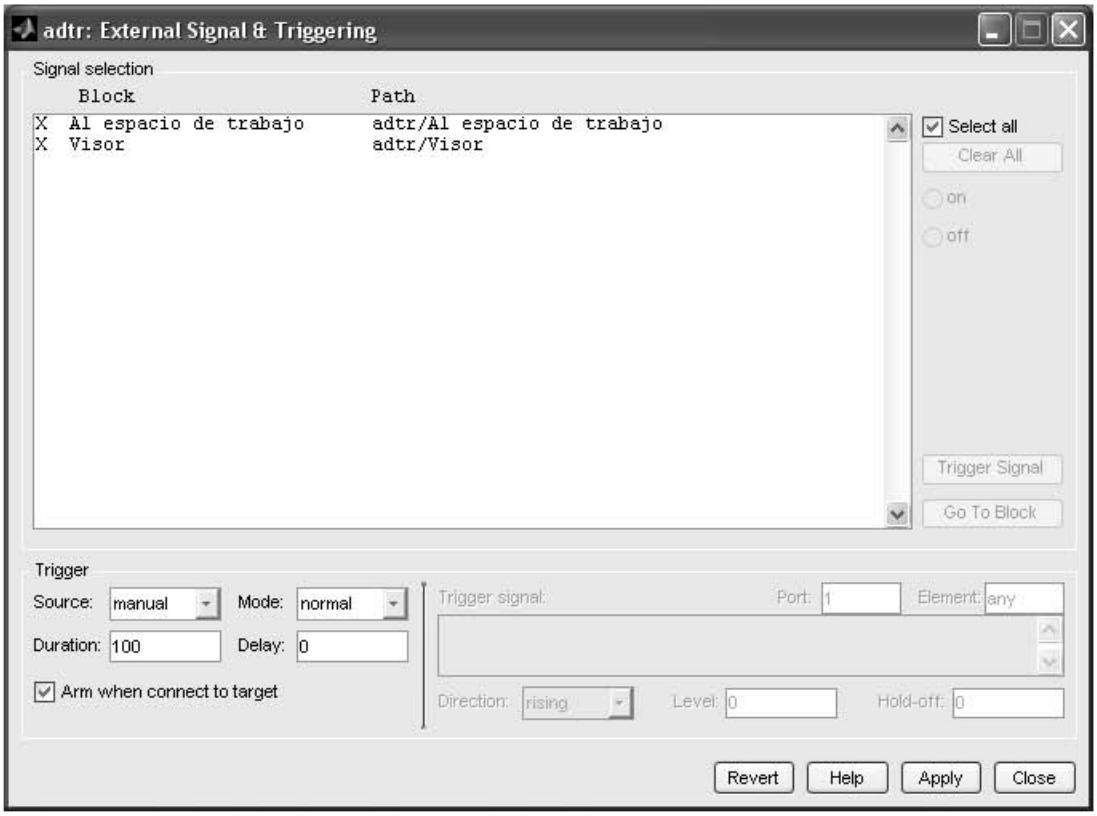

Figura 4.8: Configuración de señal y disparo del modo externo

7. Por último, el apartado "Solver" de los parámetros de configuración de Simulink (Menú Simulation→Configuration Parameters...) debe quedar como se muestra en la Figura 4.9. Se especifica que se adquirirán datos durante un segundo y que se usará el tiempo de muestreo de "0.01"segundos, igual al indicado en el bloque de entrada analógica.

Una vez hechas las configuraciones, se procede a crear el código y compilar el programa (Menú Tools→Real-Time Workshop→Build Model...). En la ventana de comandos de MATLAB aparece información del proceso. Si no hay errores termina el procedimiento con la leyenda: "*### Successful completion of Real-Time Workshop build procedure for model: ...*". Los archivos generados se encuentran en una sub-carpeta de la que contiene al modelo. Para comenzar la adquisición de datos en tiempo real se selecciona el modo externo, se oprime el botón para conectar al destino y se inicia la simulación.

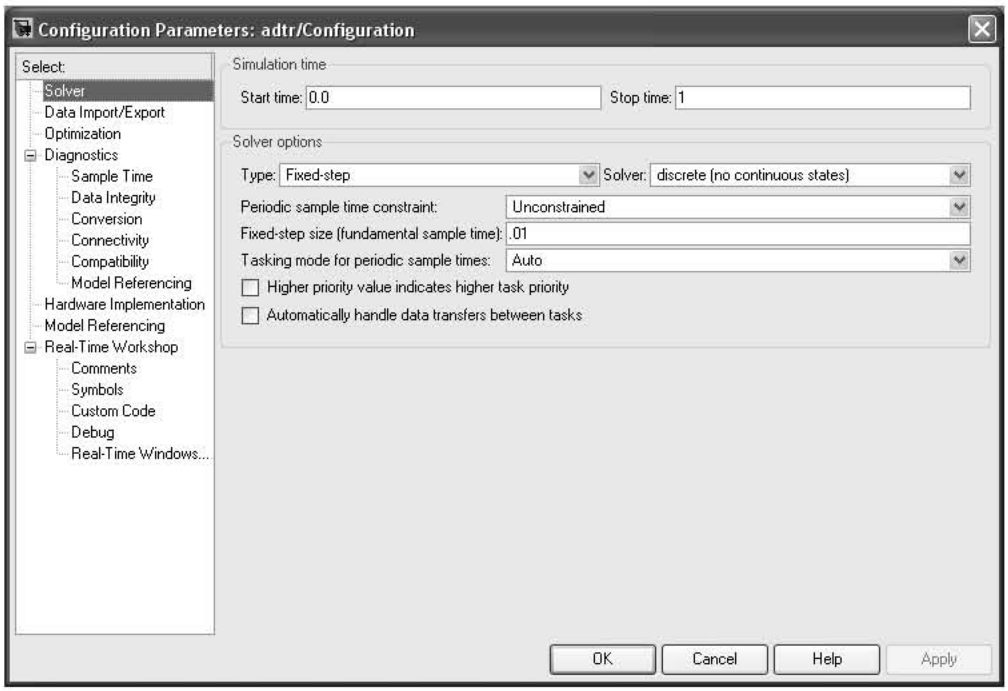

Figura 4.9: Parámetros de configuración de Simulink (Solver)

Figura 4.10: Ventana principal al iniciar el SACS

4.2.2. Interfaz gráfica

A continuación se describe cada parte de la interfaz gráfica.

Ventana principal

Se divide horizontalmente en dos partes. En la parte izquierda se muestra información en tres categorías: *Datos del sensor*, *Agregar un punto* y *Constante de calibración*. En la mitad derecha se muestran los puntos adquiridos y la recta resultante. Los controles son mostrados progresivamente para facilitar el uso y se ocultan o inhabilitan los que no se utilizan. En las Figuras 4.10 y 4.11 se muestra la ventana principal al inicio y con datos registrados, respectivamente.

Barra de menú El programa cuenta con dos menús: *Archivo* y *Acerca de...*. Dentro del menú *Archivo* (Figura 4.12) se encuentran los comandos *Nuevo*, *Abrir...*, *Guardar...* y *Cerrar*. El comando *Guardar...* es habilitado cuando se tienen dos o más puntos de calibración registrados. Al oprimirlo se termina la calibración y se abre el diálogo *Guardar archivo de registro*.

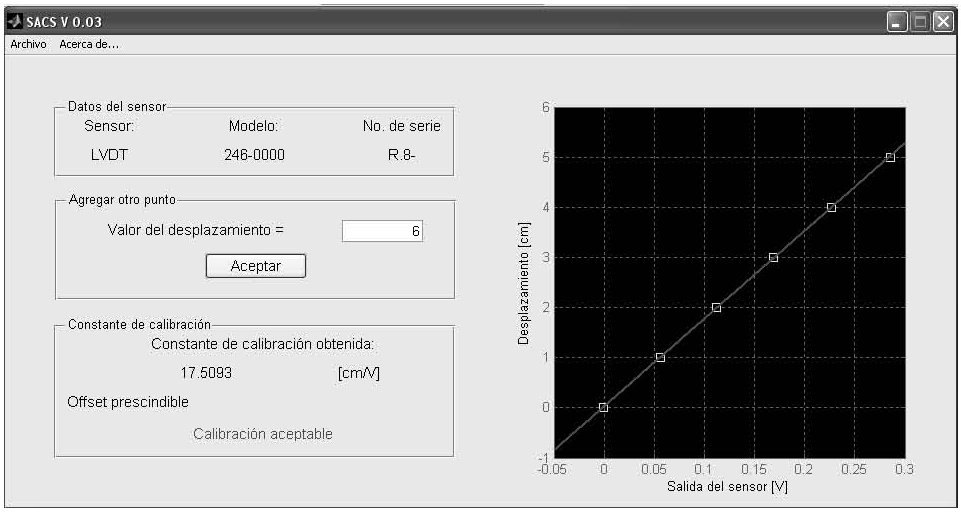

Figura 4.11: Ventana principal con algunos puntos registrados.

Figura 4.12: Menú Archivo

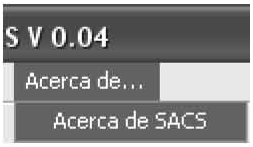

Figura 4.13: Menú Acerca de...

Al ejecutar el comando *Acerca de SACS* (Figura 4.13) se muestra en una ventana información acerca de la versión del programa.

Datos del sensor Al iniciar el programa se muestra en esta área el botón *Proporcionar datos* (Figura 4.10). En ese momento es el único control disponible. Después de introducir los datos del sensor en la ventana correspondiente (Figura 4.15), éstos son mostrados durante toda la calibración (Figura 4.11).

Agregar un punto El botón *Fijar cero* es mostrado en esta área después de introducir los datos del sensor. Al oprimir dicho botón se cuantifica la tensión de desviación presente en la señal de salida cuando la entrada es nula. Ése valor de tensión es sustraido de todas las siguientes lecturas. Una vez registrada la tensión de desviación, se despliega el nombre de la magnitud física que mide el sensor en cuestión y sus unidades, un cuadro que admite únicamente números y un botón para confirmar el dato introducido e iniciar la medición con la tarjeta de adquisición de datos (Figura 4.11). En esta sección se va proporcionando uno por uno los valores de la magnitud física que está siendo aplicada al sensor.

Constante de calibración A partir de que se cuenta con dos puntos se muestra en esta área el botón *Terminar y obtener constante*. Después de oprimirlo se muestra el resultado de la calibración (Figura 4.11) que consiste en la constante que relaciona la entrada al sensor con su salida, las unidades de dicha constante y si es o no aceptable la calibración de acuerdo al valor del coeficiente de correlación. También se muestra el botón *Guardar*, con el que se genera el archivo de registro.

Gráfica de los puntos y de la recta Muestra los puntos, representados por pequeños cuadros, conforme van siendo proporcionados al programa (Figura 4.11). Cuando

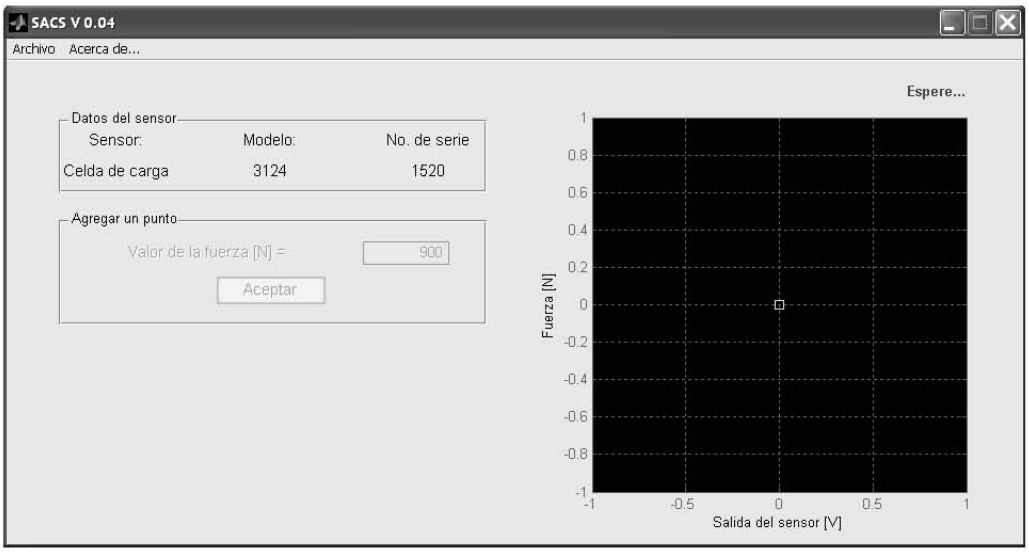

Figura 4.14: Indicador de estado

hay dos o más puntos registrados muestra la recta obtenida por el método de mínimos cuadrados. Cuenta con una cuadrícula y las etiquetas de sus ejes muestran el nombre de las magnitudes y las unidades empleadas.

Indicador de estado Aparece el texto *Espere...* en color rojo en la parte superior derecha de la ventana principal (Figura 4.14) cuando el programa se encuentra ocupado adquiriendo la señal o realizando cálculos y al mismo tiempo se deshabilitan todos los controles. El indicador desaparece y se vuelven a habilitar los controles una vez que termina la operación.

Ventana para introducir los datos del sensor

Esta ventana (Figura 4.15) aparece al oprimir el botón *Proporcionar datos* en la ventana principal. Se debe proporcionar un nombre o tipo de sensor, el modelo y el número de serie. Mediante un menú desplegable se puede elegir el sensor. En caso de que se requiera calibrar otro tipo de sensor se debe seleccionar el elemento *Otro*)

Figura 4.15: Ventana para introducir los datos del sensor

en la lista y se debe escribir su nombre en un campo que aparece abajo del menú. Dependiendo del sensor elegido, estará disponible una lista de modelos conocidos de esos sensores. Si el modelo a calibrar no aparece en la lista es posible indicarlo, en un campo adicional que aparece para tal efecto, cuando se selecciona *Otro* en el menú de modelos.

Por último, el número de serie se escribe en un cuadro de texto que admite tanto letras como números. Una ventana de error es mostrada cuando alguno de los datos no es especificado.

Figura 4.16: Diálogo para abrir archivo de registro

Diálogo para abrir archivo de registro

Es mostrado al ejecutar el comando *Abrir...* del menú *Archivo*. Cuenta con un título descriptivo y de forma predeterminada muestra solo los archivos de registro SACS que se encuentran en la carpeta actual. Tiene la apariencia de los diálogos estándar para abrir archivos en Windows. Si se trata de abrir un archivo que no tenga el formato adecuado, se mostrará una ventana de error.

Figura 4.17: Diálogo para guardar archivo de registro

Diálogo para guardar archivo de registro

Se muestra al oprimir el botón *Guardar...* (Mostrado después de oprimir el botón *Terminar y obtener constante*). o al ejecutar el comando *Guardar...* del menú *Archivo....* Es similar al diálogo anterior, pero es utilizado para elegir la ubicación y el nombre del archivo que se quiere guardar. Sugiere el nombre del archivo para evitar duplicidad. El nombre sugerido está compuesto por dos letras que identifican al sensor seguido del número de serie, enseguida un guión bajo y la fecha en la que se creó el archivo.

Formato del archivo de registro

En un principio se creó un formato de archivo arbitrario escribiéndolo directamente en un editor de archivos de texto plano y se utilizaron los comandos de alto nivel (como *csvread*) en MATLAB para poder abrirlo en el programa. Desgraciadamente no se encontró la forma de crear archivos con el mismo formato a partir de los correspondientes comandos de alto nivel (como *csvwrite*). Entonces se decidió usar los comandos de bajo nivel (como *fopen* y *fprintf*) para crear un archivo que tuviera un formato que los comandos de alto nivel (como *textread*) pudieran abrir.

El formato final en un archivo de texto ASCII con la extensión *.sacs*, cuenta con ocho líneas con información detallada de la calibración. La primera línea identifica al archivo e indica la versión del programa que lo generó. Las siguientes tres muestran la información del sensor: Nombre del sensor, modelo y número de serie. Después se muestra la fecha y la hora de creación del archivo de registro. En la sexta y séptima líneas aparecen los resultados de la calibración: Costante obtenida, offset y coeficiente de correlación. Y en la octava línea se encuentran separados por comas los nombres y las unidades de las magnitudes físicas relacionadas en la calibración. Enseguida pueden aparecer como mínimo dos líneas con los puntos de la calibración separados por comas. El archivo puede ser editado en cualquier editor de texto como el bloc de notas de Windows.

A continuación se muestra el contenido de un archivo de registro:

```
Archivo de registro de calibración SACS Versión 0.04
LVDT
246-0000
Ref8
07-Apr-2006 19:19:22
Constante de calibración obtenida: -1.1556 [cm/V]
```

```
Offset: 4.5772 [cm], Coeficiente de correlación: 1.0000
Tensión [V], Desplazamiento [cm]
1.9779, 2.2860
2.2031, 2.0320
2.4204, 1.7780
2.6453, 1.5240
2.8679, 1.2700
3.0857, 1.0160
3.3008, 0.7620
3.5179, 0.5080
3.7384, 0.2540
3.9610, 0.0000
```

Capítulo 5

Pruebas

En este capítulo se describen las pruebas efectuadas con el sistema.

5.1. Pruebas de utilidad

Figura 5.1: Sistema de adquisición de datos

Un total de seis usuarios potenciales probaron el SACS durante su desarrollo. Des-

pués de que algún usuario probaba el sistema, se modificaba el programa tratando de tomar en cuenta todos sus comentarios, sugerencias y los errores que no se habían detectado antes. Asímismo se probaba la nueva modificación en todos los casos posibles.

A cada uno de ellos se les platicó acerca del proceso de calibración de sensores a mano, después se les presentó el programa y se empezaron a registrar detalladamente sus acciones, preguntas y comentarios. Se procuró no decirles como utilizarlo porque se esperaba que intuitivamente encontraran la forma de hacerlo. Sin embargo, no fue posible cuando quedaban algunas dudas respecto al proceso manual y cuando, en versiones previas del programa, aparecían varios controles al mismo tiempo. El usuario no necesariamente oprimía el botón que se suponía debía oprimir. Es por eso que al inicio no aparece más que un botón.

Uno de los usuarios, que es un programador avanzado, sugirió que se hiciera una validación da cada dato introducido al programa. Ahora no se admiten letras ni espacios en los campos numéricos y tampoco se puede iniciar con espacios un campo de texto.

Otro usuario fue el que comentó que los ejes coordenados deberían especificar las unidades en las que se están graficando las magnitudes. Del mismo modo se refirió a la constante de calibración resultante.

Fue necesaria la deshabilitación de los controles cuando el sistema se encontraba ocupado, ya que el usuario se desesperaba aún cuando el indicador de estado era mostrado. También se ocultaron los controles para agregar datos después de guardar o al abrir un archivo de registro y se tuvieron que hacer más grandes todas las letras para mejorar la visualización.

La lista de modelos para cada sensor fue una sugerencia de un usuario, que argumentó que la mayoría de las veces se usa el mismo modelo.

Gracias a que en la plataforma en la que se desarrolló no se necesita compilar el código, el SACS ha sido modificado continuamente desde su creación, haciéndose cada

vez más robusto.

5.2. LVDT

Para la calibración del LVDT se utilizó un dispositivo como el esquematizado en la Figura 3.1. Está fabricado en acrílico y tiene montado un micrómetro calibrado en pulgadas (Figura 5.2) con una división mínima de 1/1000 [in] (254 [μm]) y escala máxima de 0.9 [in] (2.286 [cm]). El programa se ejecutó en una computadora de escritorio equipada con una tarjeta National Instruments PCI-6023E.

Se usó hule espuma para rellenar el hueco entre el sensor y la abertura en el dispositivo de modo que quedara sujetado firmemente.

Se utilizó una fuente de tensión regulada ajustada a 12 [V] y cable de tres hilos con blindaje conectado a tierra en el extremo de la fuente. Las puntas de los cables, que se conectan a la fuente y a la caja de conexiones de la tarjeta de adquisición de datos, se estañaron para evitar que se desordenaran los alambres. También se utilizaron conectores Canon macho y hembra de tres pines para conectar el sensor.

Se colocó todo el equipo en una superficie firme, se dejó calentar el equipo por 15 minutos y luego se obtuvieron constantes de calibración con diferentes cantidades de puntos (2, 3, 5, 10 y 20) en un rango de 0 a 2.286 [cm]. La constante de calibración promedio obtenida fue 1.1529 [$\frac{cm}{V}$] con una variación entre pruebas de 0.20 %. El *offset* promedio obtenido fue 0.0052 [cm] (que representa el 0.23 % de la salida máxima) con desviación estándar de 0.0033 [cm]. El coeficiente de correlación mínimo fue de 0.9982.

Se repitieron cinco veces las pruebas con 10 puntos de calibración y se obtuvo una constante promedio de 1.1836 con una variación entre mediciones de 0.06 %. Cada prueba se realiza en 8 minutos aproximadamente. La relación señal a ruido durante todas las mediciones fue de 60 [dB].

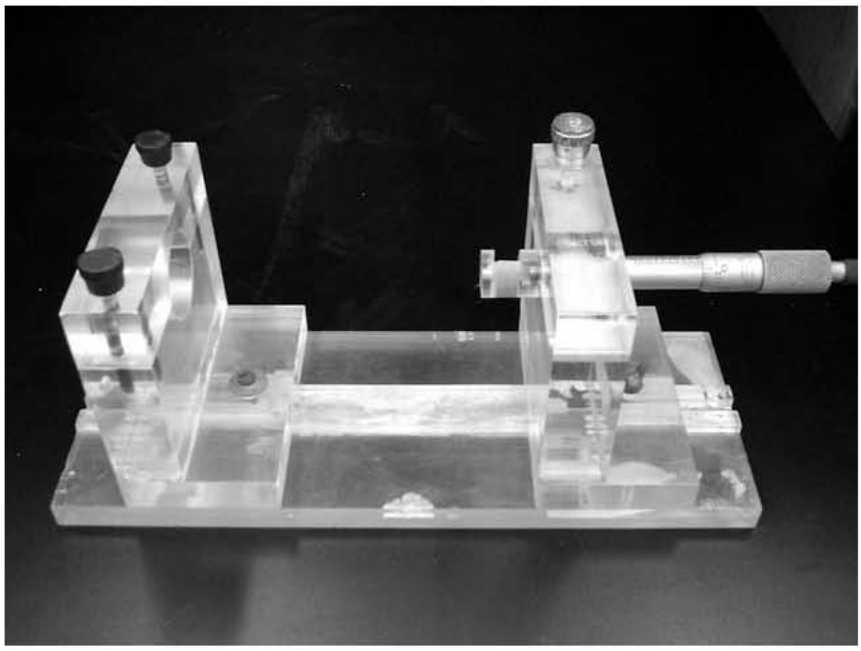

Figura 5.2: Dispositivo de acrílico para calibrar LVDT's

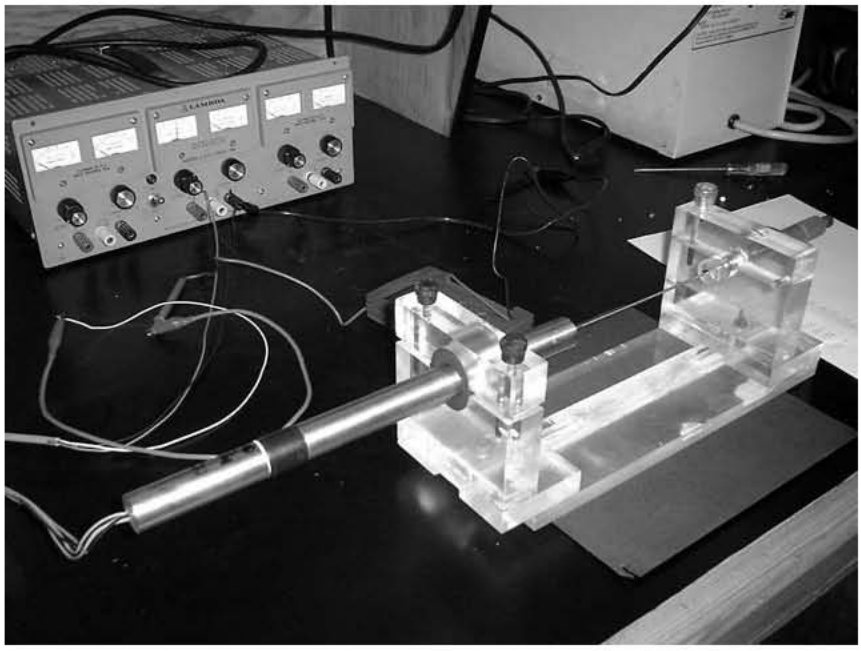

Figura 5.3: Calibración de LVDT

5.3. Celda de carga

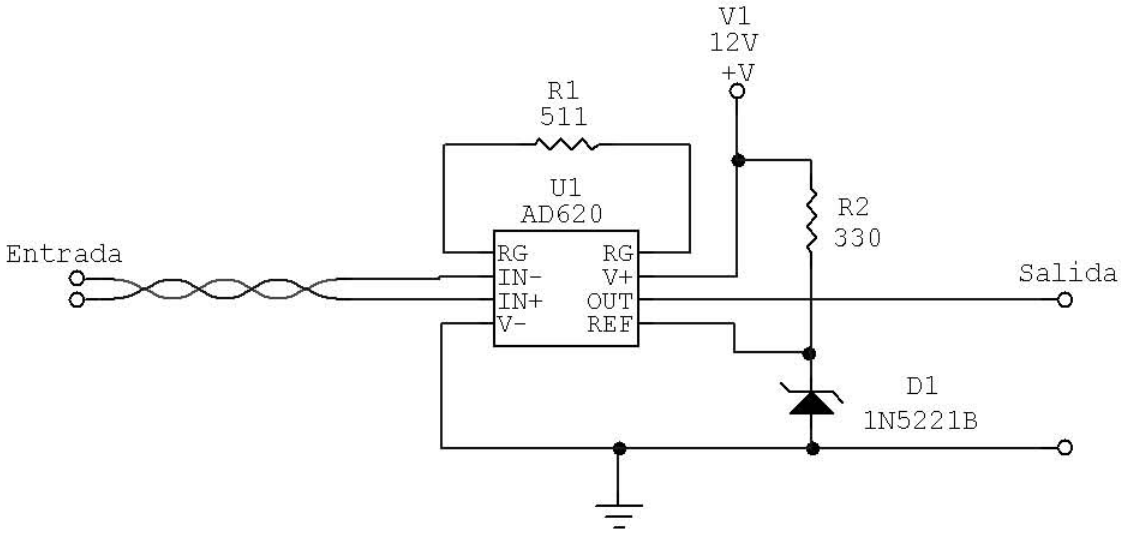

Figura 5.4: Diagrama del acondicionador de señal para celda de carga

Se empleó un marco de carga hidráulico MTS (Material Testing System) que se encuentra instalado en el Laboratorio de Estructuras del Instituto de Ingeniería (Figura 5.5). Dicha máquina cuenta con una celda de carga y un sistema de digitalización que se tomó como referencia en las calibraciones. La fuerza mínima que puede aplicarse es de 10 [N] y la máxima es de 500 [kN]. Se corrió el SACS en una computadora portátil con una tarjeta National Instruments PCMCIA DAQCard-AI-16E-4.

La señal de salida de la celda de carga (máx. 25 [mV]) era menor al tamaño que tenía el ruido inducido en el cable (alrededor de 30 [mV]), por lo que fue necesario implementar un acondicionador de señal basado en el amplificador de instrumentación integrado AD620 de Analog Devices con una ganancia de 97.6732 y una tensión de desviación de 2.5 [V], proporcionada por un diodo Zener, para tener una señal entre 0 y 5 [V]. Se armó en una tableta perforada procurando ocupar un espacio mínimo y se usaron conectores macho y hembra extra para tenerlo cerca de la celda de carga y evitar que el ruido que se indujera a lo largo del cable no fuera amplificado (Figura 5.6).

Se utilizó cable blindado de cuatro hilos con soldadura en las puntas de los cables. Se encontró que se tenía un nivel de ruido menor cuando el blindaje se conectaba al polo negativo de la fuente.

Se enroscó la celda de carga a la base superior del marco y se colocó un plato en el actuador. Se polarizó el circuito y la celda con la misma fuente de 12 [V] y se dejó calentar el arreglo por 15 minutos. Se calcularon las constantes de calibración para 2, 4, 6, 11 y 21 puntos en un rango de 0 a 90 [kN] y se obtuvo una constante promedio igual a 36.2564 $[\frac{kN}{V}]$ y entre los valores obtenidos hubo una variación de 0.31 %. Se calculó un *offset* igual a 107.7681 [N] (equivalente al 0.12 % de la salida máxima) y presentó una desviación estándar de 86.9553 [N]. El coeficiente de correlación mínimo fue 0.9999.

Cuando se repitió cinco veces la prueba con 11 puntos se encontró una constante de calibración promedio de 36.1902 $[\frac{kN}{V}]$ con variaciones de 0.05 %. La duración promedio de cada prueba es 4 minutos.

La relación señal a ruido en las mediciones con el amplificador fue de 53.98 [dB].

Figura 5.5: Calibración de celda de carga

Figura 5.6: Circuito acondicionador de señal

5.4. Acelerómetro

Se montó el acelerómetro sobre un teodolito óptico PENTAX que tiene una resolución de 1.00" (un segundo) para la medición de ángulos con respecto a la horizontal (Figuras 5.7 y 5.8). El teodolito puede medir una revolución completa, es decir, 360°. La medición de los ángulos se realiza por medio del visor pequeño, ubicado a un lado del principal, a través del cual se pueden ver tres ventanas. En la primera aparece el valor en grados, en la segunda las decenas de minutos y en la tercera las unidades de minutos, así como las decenas y unidades de segundos. Para estas pruebas también se utilizó una computadora portátil con la tarjeta PCMCIA DAQCard-AI-16E-4. Se tomó g = 977,927.7071 [mgal][1] $(1\ [gal] = 1\ [\frac{cm}{s^2}] = 0.01\ [\frac{m}{s^2}])$ como valor de referencia de aceleración debida a la gravedad.

El teodolito debe colocarse sobre una superficie lo más horizontal posible o en un tripié ajustable adecuado. Después debe nivelarse moviendo tres tornillos de ajuste de nivel para hacer que queden centrados dos niveles de burbuja circulares y uno cilíndrico.

Se empleó una fuente de alimentación regulada a 12 [V] y se hicieron las conexiones con cable blindado de tres hilos, con conectores Canon macho y hembra de tres pines del lado del sensor y cables estañados en el otro. El blindaje se conectó a la tierra física de la fuente y previo a las pruebas se dio un tiempo de calentamiento de 15 minutos. Se hicieron pruebas en un rango de 0 a 90° con 2, 3, 4, 5, 6, 11 y 21 puntos resultando una constante promedio de 922.0236 $[\frac{gal}{V}]$ y se calculó una variación de 0.20 %. El *offset* resultante fue 2465.7782 [mgal] (0.25 % de la salida máxima) y desviación estándar igual a 297.6956. El coeficiente de correlación mínimo fue 0.9999.

Repitiendo cinco veces las mediciones para la calibración con 11 puntos se obtuvo una constante promedio de 922.6805 $[\frac{gal}{V}]$ con desviaciones entre ellas de 0.10 %. Se

[1] Medida en el sótano del edificio del Instituto de Geofísica de la UNAM en Ciudad Universitaria por Manuel Mena Jara.

requieren en promedio 6 minutos para cada prueba.

La relación señal a ruido fue de 60 [dB].

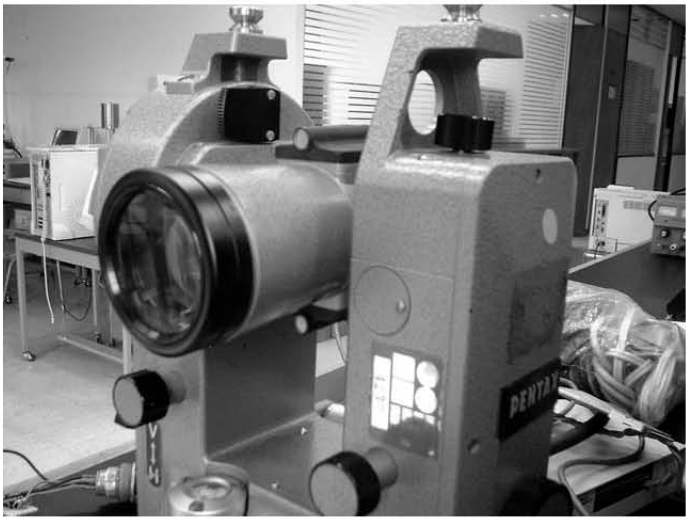

Figura 5.7: Teodolito óptico PENTAX

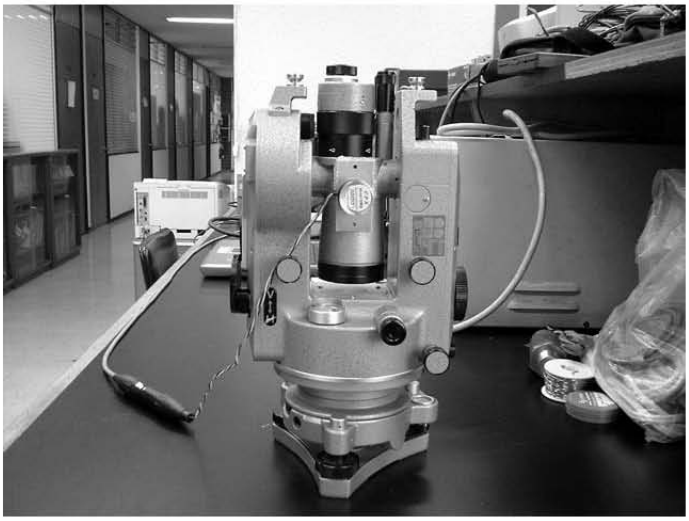

Figura 5.8: Calibración de acelerómetro

Capítulo 6

Conclusiones

Es importante hacer las calibraciones en las condiciones en las que será utilizado el sensor. Las variables ambientales, el tipo y el largo del cable utilizado, las conexiones, el tiempo de precalentamiento, así como el ruido en el lugar de trabajo afectan las mediciones.

La relación señal a ruido debe ser calculada antes de iniciar cualquier conjunto de pruebas experimentales con sensores. Hacer esto como un hábito evitará desperdiciar tiempo tratando de encontrar el motivo por el cual no se obtienen los resultados esperados.

Los circuitos de instrumentación integrados pueden ser implementados más rápidamente y dan mejores resultados que los armados a partir de amplificadores operacionales, debido a que el consumo de corriente y el espacio requerido es menor.

Conectar el blindaje del cable de señal al punto adecuado puede reducir hasta en un 40 % el ruido inducido en él. Es indispensable evitar el efecto de ciclo de tierra, que ocurre cuando se conecta el blindaje a tierra en dos puntos separados físicamente y circula una corriente a través de él, produciendo una diferencia de potencial que genera mediciones erróneas. También hay que considerar las conexiones a tierra de los equipos que se conectan a la línea, ya que en ocasiones el uso de un adaptador para omitir la

conexión a tierra de una clavija puede reducir drásticamente el ruido.

El desarrollo de software debe ser una tarea continua. El programa sufrió una gran cantidad de modificaciones desde sus primeras versiones, cuando no se probaba con la tarjeta de adquisición de datos, sino introduciendo todos los datos manualmente. Se guardaron un total de seis versiones con mejoras y depuración de errores en cada una de ellas. De no haberse realizado todas las pruebas, no se hubieran agregado todas las funciones ni se hubieran detectado errores presentes desde versiones anteriores.

Las variaciones entre constantes para diferentes cantidades de puntos ($<0.40\,\%$) y para diferentes pruebas con la misma cantidad de puntos ($<0.20\,\%$) son despreciables para todos los sensores.

La desviación (*offset*) es menor al $0.30\,\%$ de la salida máxima en todos los casos, por lo que se considera despreciable.

El coeficiente de correlación mínimo que resultó de todas las pruebas fue 0.9982, lo que implica que un modelo lineal describe muy bien el comportamiento de los tres sensores.

La relación señal a ruido de las pruebas con la celda de carga puede mejorarse aumentando la ganancia del amplificador de instrumentación y encerrando el circuito en una caja metálica conectada a tierra.

Se considera aceptable la relación señal a ruido de las pruebas con LVDT y con acelerómetro.

El sistema desarrollado en esta tesis cumplió con los objetivos ya que fue considerado como fácil de usar por las personas que tuvieron contacto con él, se redujo el tiempo necesario para obtener una constante de calibración entre 60 y $80\,\%$, los datos son analizados por el mismo programa sin la intervención del usuario y se mantiene un control de calibraciones mediante archivos de registro.

Apéndice A

Especificaciones del equipo utilizado

A.1. LVDT Transtek 0246-0000

En la Figura A.1 se muestra un LVDT de propósito general Transtek 0246-0000. Es un sistema integrado en una carcasa de acero inoxidable consistente en un oscilador de estado sólido, un transformador diferencial variable lineal y un demodulador sensible a la fase. Provee un aislamiento entre los circuitos de entrada y de salida, de modo que puede ser utilizado en sistemas flotantes o aterrizados. El núcleo está hecho de una aleación de alta permeabilidad magnética de hierro y níquel. Se puede ver un diagrama de bloques en la Figura A.2.

Figura A.1: LVDT Transtek 0246-0000

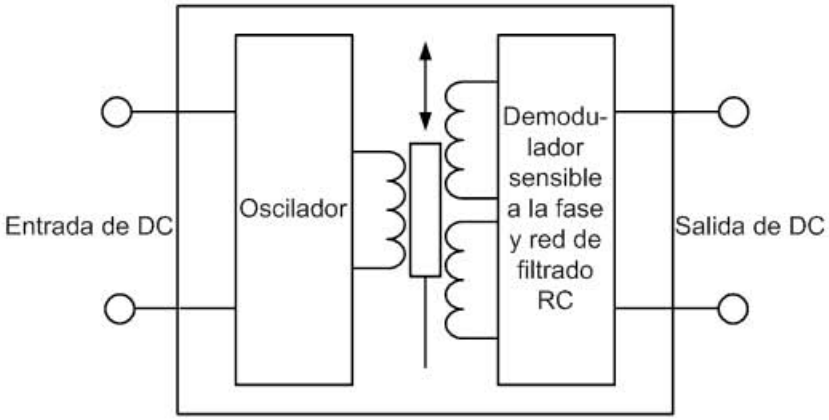

Figura A.2: Diagrama de bloques LVDT Transtek 0246-0000

Tiene un rango de ±76.2 [mm] y puede ser polarizado con una tensión de DC de 6 a 30 [V]. El consumo máximo de corriente cuando se alimenta con 30 [V] es de 52 [mA] y tiene una no-linealidad de ±0.5 % de la escala completa. La frecuencia de la portadora es de 1.4 [kHz] y su impedancia de salida de 5.6 [kΩ]. Puede medir vibraciones de hasta 75 [Hz] y tiene una resolución infinita.

También cuenta con una protección contra la polarización inversa. Para una operación correcta la impedancia de la carga debe ser de al menos 50 [kΩ]. La salida tendrá una polaridad negativa en un lado del punto neutro (cero) y una positiva del otro lado [5].

Figura A.3: Celda de carga Lebow 3124

A.2. Celda de carga Lebow 3124

La celda de carga de diámetro pequeño Lebow 3124 mostrada en la Figura A.3 está diseñada para adaptarse directamente a cilindros hidráulicos y para usarse en espacios reducidos.

Tiene una salida nominal de ±2 [mV] por cada volt de alimentación y una no linealidad de ±0.25 % de la salida máxima. Puede trabajar con una excitación máxima de 20 [V] de DC o AC RMS. Su histéresis y repetibilidad son de ±0.15 % y ±0.05 % de la salida máxima, respectivamente. La resistencia nominal del puente de Wheatstone que incorpora es de 350 [Ω] y la desviación de la salida por efecto de la temperatura es de ±0.0036 % de la lectura por cada [°C].

La capacidad nominal de la celda utilizada es de 100 [kN] y puede soportar una sobrecarga de 150 [kN]. Su frecuencia de resonancia es de 8.8 [kHz] [9].

La tensión de salida de este transductor es muy pequeña por lo que es necesario proveer de amplificación externa diseñada específicamente.

Figura A.4: Acelerómetro sísmico CFX Technologies USQ

A.3. Acelerómetro CFX Technologies USQ

El acelerómetro sísmico de balance de fuerza CFX Technologies USQ (Figura A.4) es una configuración de ultra bajo ruido (menos de 25 [μV]) y funciona con aceleraciones que presenten frecuencias de hasta 50 [Hz]. Posee un sistema servo de lazo cerrado que garantiza estabilidad, sensibilidad y precisión. Tiene un rango de escala completa de ± 2 [G]. Debe ser alimentado por una fuente de DC de 12 [V]. Su salida varía a partir de 2.525 [V] con una sensibilidad de 1.046 [$\frac{V}{G}$] [16].

A.4. Tarjeta de adquisición de datos National Instruments PCI-6023E

Cuenta con 16 canales de entrada analógica (8 en modo diferencial) con una resolución de 12 bits y cuatro ganancias seleccionables. Su razón de rechazo en modo común mínima es de 85 [dB]. Se sugiere un tiempo de precalentamiento de 15 minutos. La razón de muestreo máxima que puede manejar es 200 000 muestras por segundo y tiene una no linealidad máxima de $\pm$ 1 LSB [7].

A.5. Tarjeta de adquisición de datos National Instruments DAQCard-AI-16E-4

Tiene 16 canales de entrada analógica (8 si se selecciona el modo diferencial), resolución de 12 bits y puede adquirir hasta 250 000 muestras por segundo. Pueden seleccionarse ocho ganancias y tiene una no linealidad máxima de $\pm$ 1 LSB. Posee una razón de rechazo en modo común mínima de 85 [dB]. Se recomienda un tiempo de precalentamiento de 15 minutos [6].

Apéndice B

Código fuente del programa

A continuación se lista el código fuente de la ventana principal del SACS.

```
function varargout = sacs04(varargin)
% SACS04 M-file for sacs04.fig
%    SACS04, by itself, creates a new SACS04 or raises the existing
%    singleton*.
%
%    H = SACS04 returns the handle to a new SACS04 or the handle to
%    the existing singleton*.
%
%    SACS04('CALLBACK',hObject,eventData,handles,...) calls the local
%    function named CALLBACK in SACS04.M with the given input arguments.
%
%    SACS04('Property','Value',...) creates a new SACS04 or raises the
%    existing singleton*.  Starting from the left, property value pairs are
%    applied to the GUI before sacs04_OpeningFunction gets called.  An
%    unrecognized property name or invalid value makes property application
%    stop.  All inputs are passed to sacs04_OpeningFcn via varargin.
%
%    *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%    instance to run (singleton)".
%
```

```
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help sacs04
% Last Modified by GUIDE v2.5 15-May-2006 15:44:01
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @sacs04_OpeningFcn, ...
                   'gui_OutputFcn',  @sacs04_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before sacs04 is made visible.
function sacs04_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to sacs04 (see VARARGIN)
% Choose default command line output for sacs04
handles.output = hObject;
hold off
```

```
colordef black;
handles.logo=imread('logo50aii.jpg');
image(handles.logo);
% --- Ocultar los controles
set(handles.ejes1,'Visible','off');
set(handles.terminar,'Visible','on');
set(handles.m,'Visible','off');
set(handles.b,'Visible','off');
set(handles.unidadv,'Visible','off');
set(handles.unidad,'Visible','off');
set(handles.estado,'String','');
set(handles.guardar,'Enable','off');
set(handles.uipanel2,'Visible','off');
set(handles.uipanel3,'Visible','off');
set(handles.text5,'Enable','on');
set(handles.calana,'Visible','off');
set(handles.pb_guardar,'Visible','off');
handles.f=0;
handles.tsen=0;
handles.vcero=0;
set(handles.y,'String','');
handles(1).dx='_';
handles(1).dy='_';
limpc(hObject,eventdata,handles);
mprodat(hObject,eventdata,handles);
limps(hObject,eventdata,handles);
uicontrol(handles.prodat);
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes sacs04 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = sacs04_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function y_CreateFcn(hObject, eventdata, handles)
% hObject    handle to y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function y_Callback(hObject, eventdata, handles)
% hObject    handle to y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
switch handles.tsen
```

```
    case 2
        an=str2double(get(handles.y,'String'));
        y=977727.7071*cos(((360-an)*pi)/180);
    case 4
        y=.254*str2double(get(handles.y,'String'));
    otherwise
        y=str2double(get(handles.y,'String'));
end
if handles(1).dy=='_'
    handles.dy=y;
else
    handles.dy=[handles.dy,y];
    handles.f=1;
end
if isnan(y)
    uicontrol(handles.y);
    set(handles.y,'String','');
    errordlg('Proporcione el valor de la magnitud física',...
        'Error de entrada de datos','modal');
    return;
end
set(handles.estado,'String','Espere...');
set(handles.text5,'Enable','off');
set(handles.aceptar,'Enable','off');
set(handles.y,'Enable','off');
set(handles.terminar,'Enable','off');
load_system('adtr');
set_param('adtr','SimulationMode','external');
set_param('adtr','SimulationCommand','connect');
set_param('adtr','SimulationCommand','start');
pause(1);
```

```
adx=evalin('base','mean(salida)')-handles.vcero;
if handles(1).dx=='_'
    handles.dx=adx;
else
    handles.dx=[handles.dx,adx];
end
graf(hObject,eventdata,handles);
if handles.f
    set(handles.uipanel3,'Visible','on');
    set(handles.guardar,'Enable','on');
    graf(hObject,eventdata,handles);
    ecua(hObject,eventdata,handles);
end
set(handles.y,'String','');
set(handles.estado,'String','');
set(handles.text5,'Enable','on');
set(handles.aceptar,'Enable','on');
set(handles.y,'Enable','on');
set(handles.terminar,'Enable','on');
set(handles.uipanel2,'Title','Agregar otro punto');
guidata(hObject, handles);
uicontrol(handles.y);

% --- Executes on button press in ecu.
function ecu_Callback(hObject, eventdata, handles)
% hObject    handle to ecu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.estado,'String','Espere...');
if get(hObject,'Value')
```

```
    graf(hObject,eventdata,handles);
    ecua(hObject,eventdata,handles);
else
    graf(hObject,eventdata,handles);
    limpc(hObject,eventdata,handles);
end
set(handles.estado,'String','');
guidata(hObject, handles);

% --- Obtiene la ecuación de la recta y la despliega
function ecua(hObject,eventdata,handles)
set(handles.estado,'String','Espere...');
[aj,bon]=fit(transpose(handles.dx),transpose(handles.dy),'poly1');
hold on;
colordef black;
h_=plot(aj);
legend off;
set(h_(1),'Color',[1 0 0],...
     'LineStyle','-', 'LineWidth',2,...
     'Marker','none', 'MarkerSize',6);
xlabel('Salida del sensor [V]','color',[0 0 0]);
metsen(hObject,eventdata,handles);
ylabel(sprintf('%s',get(handles.etigraf,'String')),'color',[0 0 0]);
set(handles.m,'String',num2str(aj.p1,'%0.4f'));
set(handles.b,'String',num2str(aj.p2,'%0.4f'));
set(handles.r2,'String',num2str(bon.rsquare,'%0.4f'));
set(handles.estado,'String','');
guidata(hObject, handles);

% --- Grafica los puntos que han sido agregados
function graf(hObject,eventdata,handles)
```

```
set(handles.estado,'String','Espere...');
colordef black;
hold off;
plot(handles.dx,handles.dy,'square');
xlabel('Salida del sensor [V]','color',[0 0 0]);
metsen(hObject,eventdata,handles);
ylabel(sprintf('%s',get(handles.etigraf,'String')),'color',[0 0 0]);
grid on;
set(handles.ejes1,'xcolor',[.4 .4 .4],'ycolor',[.4 .4 .4]);
set(handles.estado,'String','');
guidata(hObject, handles);

% --- Limpia los campos donde se despliegan los parámetros obtenidos
function limpc(hObject,eventdata,handles)
set(handles.m,'String','');
set(handles.b,'String','');
set(handles.r2,'String','');
set(handles.text1,'Visible','off');
set(handles.text2,'Visible','off');
set(handles.text3,'Visible','off');
guidata(hObject, handles);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if strcmp(questdlg('¿En verdad quiere cerrar esta ventana?',...
        'Confirme la operación','Sí','No','No'),'Sí')
    colordef white;
    evalin('base','clear datsal');
```

```
    evalin('base','clc');
    delete(handles.figure1);
end

% --------------------------------------------------------------------
function abrir_Callback(hObject, eventdata, handles)
% hObject    handle to abrir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[fname,pname] = uigetfile( ...
       {'*.sacs','Archivo de registro SACS (*.sacs)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Abrir archivo de registro');
if fname~=0
    info=textread(fullfile(pname,fname),'%s',7,'delimiter','\t');
    if ~strcmp(info{1},...
            'Archivo de registro de calibración SACS Versión 0.04')
        errordlg('Archivo de registro no válido',...
            'Error de archivo','modal');
        return;
    end
    set(handles.sen,'String',info{2});
    switch info{2}
        case 'Acelerómetro'
            handles.tsen=2;
            set(handles.unidadv,'String','[mgal/V]');
            set(handles.unidad,'String','[mgal]');
        case 'Celda de carga'
            handles.tsen=3;
            set(handles.unidadv,'String','[N/V]');
            set(handles.unidad,'String','[N]');
```

```
        case 'LVDT'
            handles.tsen=4;
            set(handles.unidadv,'String','[cm/V]');
            set(handles.unidad,'String','[cm]');
        otherwise
            handles.tsen=0;
    end
    metsen(hObject,eventdata,handles);
    set(handles.mod,'String',info{3});
    set(handles.nos,'String',info{4});
    set(handles.prodat,'Visible','off');
    set(handles.uipanel3,'Visible','on');
    mdatsen(hObject,eventdata,handles);
    [x,y]=textread(fullfile(pname,fname),'%f %f',...
        'delimiter',',','headerlines',8);
    handles.dx=transpose(x);
    handles.dy=transpose(y);
    graf(hObject,eventdata,handles);
    ecua(hObject,eventdata,handles);
    set(handles.uipanel2,'Visible','off');
    terminar_Callback(hObject,eventdata,handles);
    set(handles.pb_guardar,'Visible','off');
    set(handles.guardar,'Enable','off');
    guidata(hObject,handles);
else
    return;
end

% --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
terminar_Callback(hObject,eventdata,handles)
set(handles.pb_guardar,'Visible','off');
switch handles.tsen
    case 2
        ts='ac';
    case 3
        ts='cc';
    case 4
        ts='dt';
    otherwise
        ts='ot';
end
nombre=sprintf('%s%s_%s',ts,get(handles.nos,'String'),date);
[fname,pname] = uiputfile( ...
       {'*.sacs','Archivo de registro SACS (*.sacs)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Guardar archivo de registro',nombre);
if fname==0
    return;
end
ecua(hObject,eventdata,handles);
en1='Archivo de registro de calibración SACS Versión 0.04';
en2=get(handles.sen,'String');
en3=get(handles.mod,'String');
en4=get(handles.nos,'String');
en5=datestr(now);
en6=sprintf('Constante de calibración obtenida: %s %s',...
    get(handles.m,'String'),get(handles.unidadv,'String'));
en7=sprintf('Offset: %s %s, Coeficiente de correlación: %s',...
```

```
    get(handles.b,'String'),get(handles.unidad,'String'),...
    get(handles.r2,'String'));
en8=sprintf('Tensión [V], %s',get(handles.etigraf,'String'));
fid=fopen(fullfile(pname,fname),'wt');
if fid<0
    return;
end
fprintf(fid,'%s\n',en1);
fprintf(fid,'%s\n',en2);
fprintf(fid,'%s\n',en3);
fprintf(fid,'%s\n',en4);
fprintf(fid,'%s\n',en5);
fprintf(fid,'%s\n',en6);
fprintf(fid,'%s\n',en7);
fprintf(fid,'%s\n',en8);
tam=size(handles.dx);
for i=1:tam(2)
    fprintf(fid,'%0.4f, ',handles.dx(i));
    fprintf(fid,'%0.4f\n',handles.dy(i));
end
fclose(fid);
guidata(hObject,handles);

% --------------------------------------------------------------------
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure1_CloseRequestFcn(hObject,eventdata,handles);

% --------------------------------------------------------------------
```

```
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% --------------------------------------------------------------------
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function nuevo_Callback(hObject, eventdata, handles)
% hObject    handle to nuevo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sacs04_OpeningFcn(hObject,eventdata,handles);

% --------------------------------------------------------------------
function acercades_Callback(hObject, eventdata, handles)
% hObject    handle to acercades (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
warndlg('Sistema Automatizado de Calibración de Sensores (SACS) Version 0.04
Iván López Pineda López © 2006','Acerca de SACS','modal');

% --- Obtiene los datos del sensor
% --- Executes on button press in prodat.
function prodat_Callback(hObject, eventdata, handles)
% hObject    handle to prodat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
x=sacs_s;
if ~ischar(x)
    handles.tsen=x.vsen;
    set(handles.prodat,'Visible','off');
    metsen(hObject,eventdata,handles);
    set(handles.sen,'Visible','on','String',x.sensor);
    set(handles.mod,'Visible','on','String',x.modelo);
    set(handles.nos,'Visible','on','String',x.serie);
    mdatsen(hObject,eventdata,handles);
    set(handles.uipanel2,'Visible','on','Title','Fijar cero');
    set(handles.cero,'Visible','on','Enable','on');
    set(handles.text5,'Visible','off','Enable','on');
    set(handles.y,'Visible','off','Enable','on');
    set(handles.aceptar,'Visible','off','Enable','on');
end
guidata(hObject,handles);
uicontrol(handles.cero);

% --- Limpia los datos del sensor
function limps(hObject,eventdata,handles)
set(handles.sen,'String','');
set(handles.mod,'String','');
set(handles.nos,'String','');

% --- Muestra el botón Proporcionar datos
function mprodat(hObject,eventdata,handles)
set(handles.prodat,'Visible','on');
set(handles.text14,'Visible','off');
set(handles.text15,'Visible','off');
set(handles.text16,'Visible','off');
```

```
set(handles.sen,'Visible','off');
set(handles.mod,'Visible','off');
set(handles.nos,'Visible','off');

% --- Muestra los datos del sensor
function mdatsen(hObject,eventdata,handles)
set(handles.text14,'Visible','on');
set(handles.text15,'Visible','on');
set(handles.text16,'Visible','on');
set(handles.sen,'Visible','on');
set(handles.mod,'Visible','on');
set(handles.nos,'Visible','on');

% --- Muestra etiquetas correspondientes a cada sensor
function metsen(hObject,eventdata,handles)
switch handles.tsen
  case 2
    set(handles.text5,'String','Valor del ángulo de inclinación [°] =');
    set(handles.etigraf,'String','Aceleración [mgal]');
    set(handles.unidadv,'String','[mgal/V]');
    set(handles.unidad,'String','[mgal]');
  case 3
    set(handles.text5,'String','Valor de la fuerza [N] =');
    set(handles.etigraf,'String','Fuerza [N]');
    set(handles.unidadv,'String','[N/V]');
    set(handles.unidad,'String','[N]');
  case 4
    set(handles.text5,'String','Valor del desplazamiento [in/10] =');
    set(handles.etigraf,'String','Desplazamiento [cm]');
    set(handles.unidadv,'String','[cm/V]');
    set(handles.unidad,'String','[cm]');
```

```
  otherwise
    set(handles.text5,'String','Valor de la magnitud física =');
    set(handles.etigraf,'String','Valor de la magnitud física');
end
guidata(hObject,handles);

% --- Executes on key press over aceptar with no controls selected.
function aceptar_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isequal(get(hObject,'CurrentKey'),'return')
aceptar_Callback(hObject,eventdata,handles);
end

% --- Executes on key press over prodat with no controls selected.
function prodat_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to prodat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in terminar.
function terminar_Callback(hObject, eventdata, handles)
% hObject    handle to terminar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.terminar,'Visible','off');
set(handles.text1,'Visible','on');
set(handles.m,'Visible','on');
set(handles.unidadv,'Visible','on');
set(handles.unidad,'Visible','on');
```

```
if str2double(get(handles.b,'String'))<=.1
    set(handles.text2,'String','Offset prescindible','Visible','on');
    set(handles.b,'Visible','off');
else
    set(handles.text2,'String','Offset:','Visible','on');
    set(handles.b,'Visible','on');
end
if str2double(get(handles.r2,'String'))>=.9
    set(handles.calana,'String','Calibración aceptable',...
        'ForegroundColor',[0 .502 0]);
else
    set(handles.calana,'String','Calibración no aceptable',...
        'ForegroundColor',[1 0 0]);
end
set(handles.calana,'Visible','on');
set(handles.uipanel2,'Visible','off');
set(handles.pb_guardar,'Visible','on');

% --- Executes on button press in pb_guardar.
function pb_guardar_Callback(hObject, eventdata, handles)
% hObject    handle to pb_guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pb_guardar,'Visible','off');
guardar_Callback(hObject,eventdata,handles);
guidata(hObject,handles);

% --- Executes on button press in cero.
function cero_Callback(hObject, eventdata, handles)
% hObject    handle to cero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set(handles.estado,'String','Espere...');
set(handles.cero,'Enable','off');
load_system('adtr');
set_param('adtr','SimulationMode','external');
set_param('adtr','SimulationCommand','connect');
set_param('adtr','SimulationCommand','start');
pause(1);
handles.vcero=evalin('base','mean(salida)');
handles.dx=0;
handles.dy=0;
graf(hObject,eventdata,handles);
set(handles.cero,'Enable','on','Visible','off');
set(handles.estado,'String','');
set(handles.uipanel2,'Title','Agregar un punto');
set(handles.text5,'Visible','on');
set(handles.y,'Visible','on');
set(handles.aceptar,'Visible','on');
guidata(hObject,handles);
uicontrol(handles.y);
```

A continuación se lista el código fuente de la ventana para introducir los datos del sensor del SACS.

```
function varargout = sacs_s(varargin)
% SACS_S M-file for sacs_s.fig
%    SACS_S by itself, creates a new SACS_S or raises the
%    existing singleton*.
%
%    H = SACS_S returns the handle to a new SACS_S or the handle to
%    the existing singleton*.
%
%    SACS_S('CALLBACK',hObject,eventData,handles,...) calls the local
%    function named CALLBACK in SACS_S.M with the given input arguments.
%
%    SACS_S('Property','Value',...) creates a new SACS_S or raises the
%    existing singleton*.  Starting from the left, property value pairs are
%    applied to the GUI before sacs_s_OpeningFunction gets called.  An
%    unrecognized property name or invalid value makes property application
%    stop.  All inputs are passed to sacs_s_OpeningFcn via varargin.
%
%    *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%    instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help sacs_s
% Last Modified by GUIDE v2.5 01-May-2006 20:36:24
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @sacs_s_OpeningFcn, ...
                   'gui_OutputFcn',  @sacs_s_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before sacs_s is made visible.
function sacs_s_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to sacs_s (see VARARGIN)
set(handles.osen,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.omod,'Visible','off');
set(handles.text6,'Visible','off');
% Choose default command line output for sacs_s
handles.output=struct;
handles.output.vsen=0;
handles.output.sensor='';
handles.output.modelo='';
handles.output.serie='';
% Update handles structure
```

```
guidata(hObject, handles);
% Insert custom Title and Text if specified by the user
if(nargin > 3)
    for index = 1:2:(nargin-3),
        if nargin-3==index, break, end
        switch lower(varargin{index})
         case 'title'
          set(hObject, 'Name', varargin{index+1});
         case 'string'
          set(handles.text1, 'String', varargin{index+1});
        end
    end
end

% Determine the position of the dialog - centered on the callback figure
% if available, else, centered on the screen
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
    FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
else
```

```
    GCBFOldUnits = get(gcbf,'Units');
    set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position');
    set(gcbf,'Units',GCBFOldUnits);
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);

% Show a question icon from dialogicons.mat - variables questIconData
% and questIconMap
load dialogicons.mat

IconData=questIconData;
questIconMap(256,:) = get(handles.figure1, 'Color');
IconCMap=questIconMap;

Img=image(IconData, 'Parent', handles.axes1);
set(handles.figure1, 'Colormap', IconCMap);

set(handles.axes1, ...
    'Visible', 'off', ...
    'YDir'   , 'reverse'       , ...
    'XLim'   , get(Img,'XData'), ...
    'YLim'   , get(Img,'YData')  ...
    );

% Make the GUI modal
set(handles.figure1,'WindowStyle','modal')
```

```
% UIWAIT makes sacs_s wait for user response (see UIRESUME)
uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = sacs_s_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% The figure can be deleted now
delete(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=size(handles.output(1).sensor);
if handles.output.vsen~=5
    if s(2)>14 | s(2)==0
    errordlg('Seleccione un sensor','Error de entrada de datos','modal');
    pause(10);
    return
    end
else
    if s(1)==0 | s(2)==0
    errordlg('Escriba el nombre del sensor',...
'Error de entrada de datos','modal');
    return
```

```
    end
end
esigual=strcmp(handles.output.modelo,'Seleccione un modelo...');
s=size(handles.output.modelo);
if esigual
    errordlg('Seleccione un modelo de sensor',...
        'Error de entrada de datos','modal');
    return
end
if s(1)==0 | s(2)==0
    errordlg('Escriba el modelo del sensor',...
        'Error de entrada de datos','modal');
    return
end
s=size(handles.output.serie);
if s(1)==0 | s(2)==0
    errordlg('Escriba el número de serie del sensor',...
        'Error de entrada de datos','modal');
    return
end
if isspace(handles.output.sensor(1))
    errordlg('Escriba nuevamente el nombre del sensor',...
        'Error de entrada de datos','modal');
    set(handles.osen,'String','');
    return
end
if isspace(handles.output.modelo(1))
    errordlg('Escriba nuevamente el modelo del sensor',...
        'Error de entrada de datos','modal');
    set(handles.omod,'String','');
    return
```

```
end
if isspace(handles.output.serie(1))
    errordlg('Escriba nuevamente el número de serie del sensor',...
        'Error de entrada de datos','modal');
    set(handles.nos,'String','');
    return
end
% Update handles structure
guidata(hObject, handles);
% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = get(hObject,'String');
% Update handles structure
guidata(hObject, handles);
% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output='Cancelar';
```

```
guidata(hObject,handles);
if isequal(get(handles.figure1, 'waitstatus'), 'waiting')
    % The GUI is still in UIWAIT, us UIRESUME
    uiresume(handles.figure1);
else
    % The GUI is no longer waiting, just close it
    delete(handles.figure1);
end

% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
    % User said no by hitting escape
    handles.output = 'Cancelar';
    % Update handles structure
    guidata(hObject, handles);
    uiresume(handles.figure1);
end
if isequal(get(hObject,'CurrentKey'),'return')
    pushbutton1_Callback(hObject,eventdata,handles)
end

% --- Executes on selection change in sen.
function sen_Callback(hObject, eventdata, handles)
% hObject    handle to sen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
contents = get(hObject,'String');
handles.output.vsen=get(hObject,'Value');
switch handles.output.vsen
    case 1
        set(handles.mod,'Enable','off');
        set(handles.osen,'Visible','off','String','');
        set(handles.text5,'Visible','off');
        set(handles.omod,'Visible','off');
        set(handles.text6,'Visible','off');
        handles.output.sensor=contents{get(hObject,'Value')};
    case 2
        set(handles.mod,'Enable','on','String',...
            {'Seleccione un modelo...';'USQ';'Otro'});
        mod_Callback(hObject,eventdata,handles);
        set(handles.osen,'Visible','off','String','');
        set(handles.text5,'Visible','off');
        set(handles.omod,'Visible','off');
        set(handles.text6,'Visible','off');
        handles.output.sensor=contents{get(hObject,'Value')};
    case 3
        set(handles.mod,'Enable','on','String',...
            {'Seleccione un modelo...';'3124';'3132';'Otro'});
        mod_Callback(hObject,eventdata,handles);
        set(handles.osen,'Visible','off','String','');
        set(handles.text5,'Visible','off');
        set(handles.omod,'Visible','off');
        set(handles.text6,'Visible','off');
        handles.output.sensor=contents{get(hObject,'Value')};
    case 4
        set(handles.mod,'Enable','on','String',...
            {'Seleccione un modelo...';'246-00000';'Otro'});
```

```
        mod_Callback(hObject,eventdata,handles);
        set(handles.osen,'Visible','off','String','');
        set(handles.text5,'Visible','off');
        set(handles.omod,'Visible','off');
        set(handles.text6,'Visible','off');
        handles.output.sensor=contents{get(hObject,'Value')};
    case 5
        set(handles.mod,'Enable','off');
        set(handles.osen,'Visible','on');
        set(handles.text5,'Visible','on');
        set(handles.omod,'Visible','on');
        set(handles.text6,'Visible','on');
        handles.output.sensor='';
        osen_Callback(hObject,eventdata,handles);
        handles.output.modelo='';
        omod_Callback(hObject,eventdata,handles);
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function sen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function nos_Callback(hObject, eventdata, handles)
% hObject    handle to nos (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output.serie=get(hObject,'String');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function nos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function mod_Callback(hObject, eventdata, handles)
% hObject    handle to mod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
contents = get(hObject,'String');
handles.output.modelo=contents{get(hObject,'Value')};
esi=strcmp(handles.output.modelo,'Otro');
switch esi
    case 1
    set(handles.omod,'Visible','on');
    set(handles.text6,'Visible','on');
    handles.output.modelo='';
    omod_Callback(hObject,eventdata,handles);
    case 0
    set(handles.omod,'Visible','off');
    set(handles.text6,'Visible','off');
```

```
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function mod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function osen_Callback(hObject, eventdata, handles)
% hObject    handle to osen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output.sensor=get(hObject,'String');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function osen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to osen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function omod_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to omod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output.modelo=get(hObject,'String');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function omod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to omod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',...
    get(0,'defaultUicontrolBackgroundColor'));
end
```

Bibliografía

[1] T. G. Beckwith. *Mechanical Measurements*. Addison-Wesley, Inglaterra, 1961.

[2] José María Ferrero Corral. *Instrumentación electrónica. Sensores*. Universidad Politécnica de Valencia, España, 1994.

[3] Machine design. *Accelerometers*. http://www.machinedesign.com/bde/electrical/bdeee6/bdeee6_6.html.

[4] Ernest O. Doebelin. *Measurement Systems: Application and Design*. McGraw-Hill, EUA, 1966.

[5] Transtek Inc. *Series 240. General Purpose DC LVDTs Datasheet*.

[6] National Instruments. *DAQCard E Series User Manual*. National Instruments, EUA, 1999.

[7] National Instruments. *NI 6023E/6024E/6025E Family Specifications*. National Instruments, EUA, 2005.

[8] Jeff Johnson. *GUI Bloopers. Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann Publishers, EUA, 2000.

[9] Lebow. *Model 3124. Low Profile Load Cell Datasheet*.

[10] The Mathworks. *Creating Graphical User Interfaces. Version 7.* The Mathworks, EUA, 2005.

[11] The Mathworks. *Curve Fitting Toolbox. For use with MATLAB. User's Guide. Version 1.* The Mathworks, EUA, 2005.

[12] The Mathworks. *Getting Started with MATLAB. Version 7.* The Mathworks, EUA, 2005.

[13] The Mathworks. *Real-Time Windows Target. For use with Real-Time Workshop. User's Guide. Version 2.* The Mathworks, EUA, 2005.

[14] The Mathworks. *Real-Time Workshop. For use with Simulink. Getting Started. Version 6.* The Mathworks, EUA, 2005.

[15] The Mathworks. *Using Simulink. Version 6.* The Mathworks, EUA, 2005.

[16] CFX Technologies. *Model USQ. Force Balance Seismic Accelerometer Datasheet.*

[17] Erhard Wielandt. Seismic sensors and their calibration. 2002.