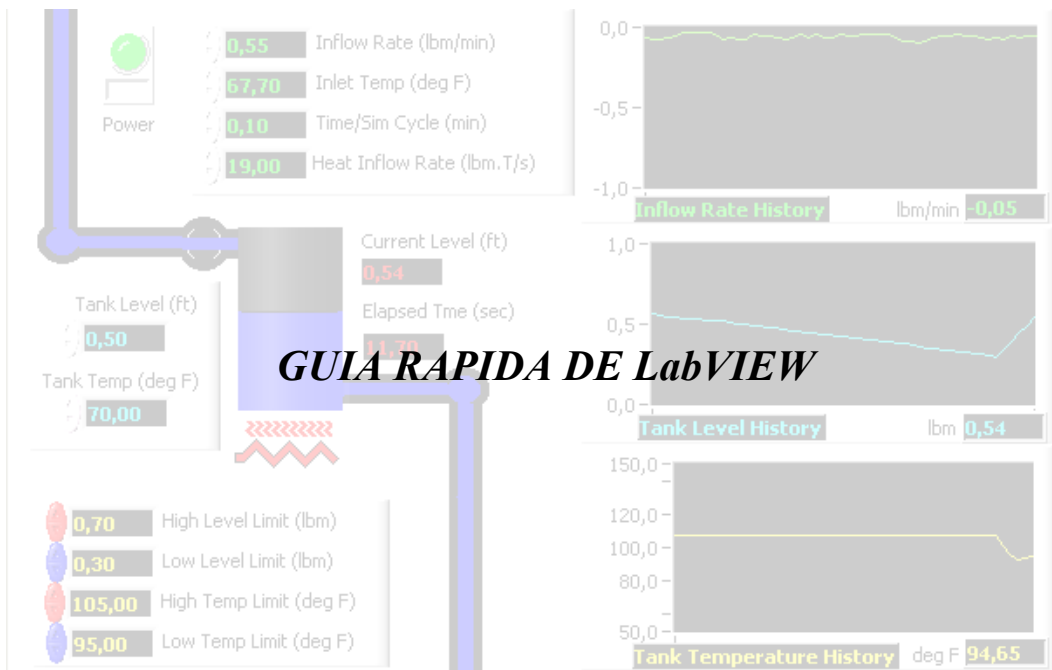




Departament
d'Enginyeria
Electrònica



Autores:

**Jordi Zaragoza
Lluís Ferrer**



Agradecimientos a Oliver Marcelo y Oriol Goula por su participación, ayuda y colaboración en la preparación de esta guía.

Índice:

1 INTRODUCCIÓN	4
2 ESTRUCTURA DE LABVIEW	5
3 ENTRONO DE TRABAJO	5
3.1 CREACIÓN DE UN VI	5
3.2 PANEL DE CONTROL (FRONT PANEL)	6
3.2.1 Paleta de controles (Controls palette)	7
3.3 DIAGRAMA DE BLOQUES	8
3.3.1 Paleta de funciones (functions palette)	9
3.3.2 Paleta de herramientas (Tools palette)	11
4 PROGRAMACIÓN EN LABVIEW	12
5 EJECUCIÓN DE UN VI	12
6 ESTRUCTURAS	14
6.1 CASE STRUCTURE	15
6.2 SEQUENCE STRUCTURE	15
6.3 FOR LOOP	16
6.4 WHILE LOOP	17
6.5 FORMULA NODE	18
7 EJEMPLO: CONSTRUCCIÓN DE UN VI	20
7.1 PANEL FRONTAL	20
7.2 DIAGRAMA DE BLOQUES	21
8 ADQUISICIÓN DE DATOS	23
8.1 TARJETA DE ADQUISICIÓN DE DATOS	23
8.2 ENTRADAS ANALÓGICAS	25
8.3 SALIDAS ANALÓGICAS	27

1 INTRODUCCIÓN

LabVIEW constituye un revolucionario sistema de programación gráfica para aplicaciones que involucren adquisición, control, análisis y presentación de datos. Las ventajas que proporciona el empleo de LabVIEW se resumen en los siguientes puntos:

1. Se reduce el tiempo de desarrollo de las aplicaciones, como mínimo de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
2. Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
3. Da la posibilidad a los usuarios de crear soluciones completas y complejas.
4. Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
5. El sistema está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
6. Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

LabVIEW es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el lenguaje C o BASIC. Sin embargo, LabVIEW se diferencia de dichos programas en el importante aspecto que los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o lenguaje “G” para crear programas basados en diagramas de bloques.

Para el empleo de LabVIEW no se requiere gran experiencia en programación, ya que se emplean iconos, términos e ideas familiares a científicos e ingenieros, y se apoya sobre símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que el resto de lenguajes de programación convencionales.

LabVIEW posee extensas librerías de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW incluye librerías específicas para la adquisición de datos, control de instrumentación VXI, GPIB, comunicación serie, análisis y guardado de datos.

LabVIEW también proporciona potentes herramientas que facilitan la depuración de los programas.

2 ESTRUCTURA DE LABVIEW

Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.

Todos los VIs tienen un panel frontal y un diagrama de bloques. Las paletas contienen las opciones que se emplean para crear y modificar los VIs. A continuación se procederá a realizar una somera descripción de estos conceptos.

3 ENTRONO DE TRABAJO

Esta sección describe los requerimientos del software para esta guía rápida, como se inicia el software en la versión PC, como se accede a las diferentes herramientas, ayudas y como se inicializa un nuevo diseño.

En esta sección se pretende que los usuarios sean capaces de:

1. Utilizar LabVIEW para crear aplicaciones.
2. Entender paneles frontales, diagramas de bloques, íconos y paneles de conexión.
3. Incorporación de funciones de LabVIEW.
4. Creación y guardado de programas en LabVIEW para su utilización como subrutinas.

Para abordar los diferentes objetivos de esta guía y para una mejor comprensión por parte del estudiante, se deberán ir completando los diferentes ejercicios de ejemplo.

3.1 Creación de un VI

Para la creación de un nuevo VI seleccionar:

New → *Blank VI*

Donde previamente ya se ha ejecutado la aplicación principal de *National Instruments LabVIEW 7*.

Nos deben aparecer dos ventanas (Fig. 3.1) nuevas, la de *Front Panel* y la de *Block Diagram*.

En la ventana **Front Panel** se definen las variables de entrada/salida del programa, en forma de controles, displays etc.

En la ventana **Block Diagram** se escribe el código fuente del programa.

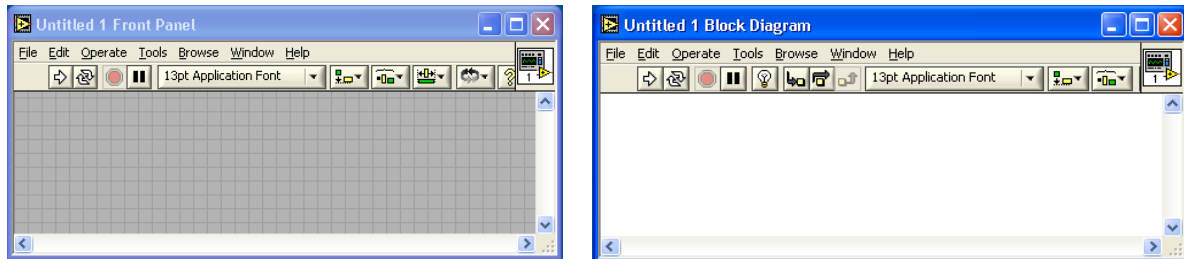


Figura 3.1 Front Panel y Block Diagram

3.2 Panel de control (Front Panel)

Se trata de la interfaz gráfica del VI con el usuario (Fig. 3.2). Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa (son las variables de entrada y salida del programa). Un panel frontal está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc. Cada uno de ellos puede estar definido como un control (a) o un indicador (b). Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.

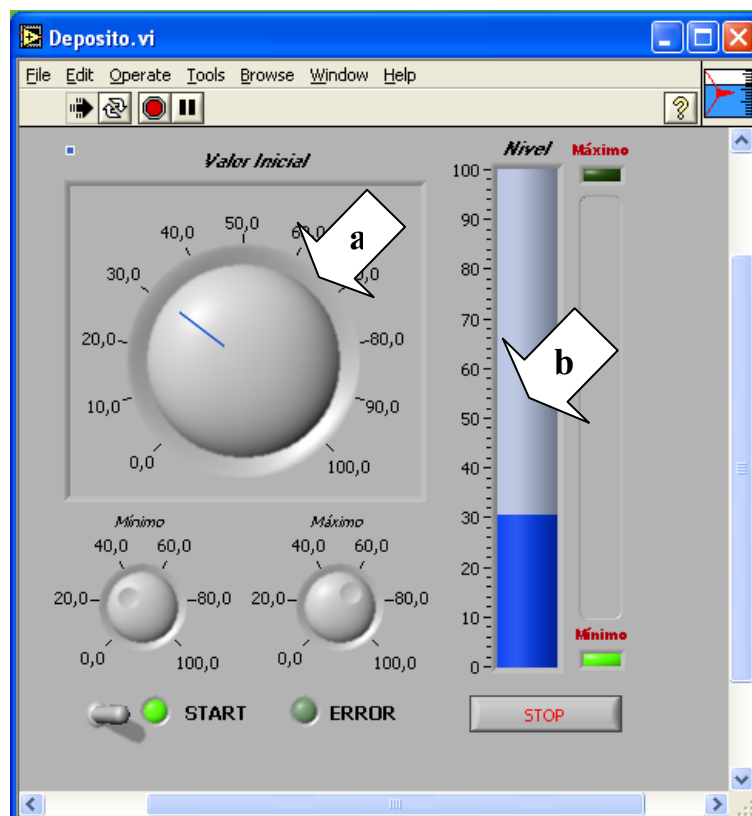


Figura 3.2. Panel Frontal.

Para poder visualizar las diferentes herramientas de visualización y control, pulsar sobre el **Panel Frontal** con el botón derecho del ratón o en el menú *Windows* → *Show Controls Palette*.

3.2.1 Paleta de controles (Controls palette)

Se utiliza únicamente en el panel frontal. Contiene todos los controles e indicadores que se emplearán para crear la interfaz del VI con el usuario.

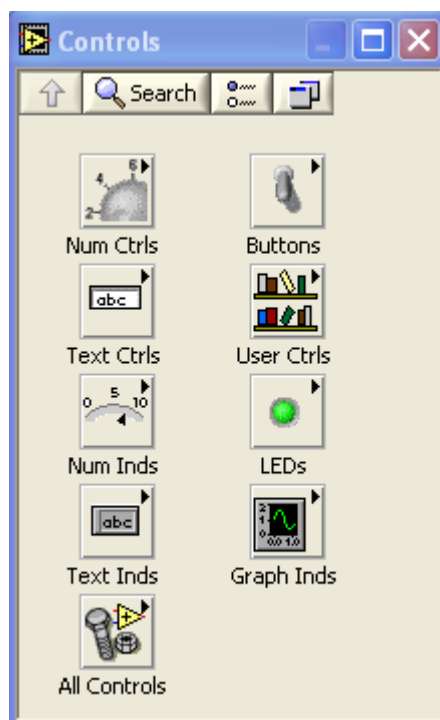
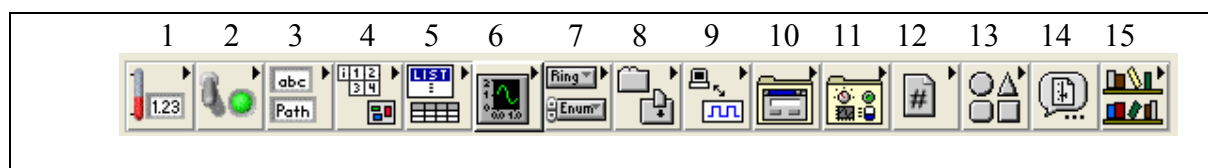


Figura 3.3 Paleta de Controles.

El menú *All Controls* de la ventana correspondiente (Fig. 3.3) al panel frontal contiene las siguientes opciones:



1. **Numeric.-** Para la introducción y visualización de cantidades numéricas.
2. **Boolean.-** Para la entrada y visualización de valores booleanos.
3. **String & Path.-** Para la entrada y visualización de texto.
4. **Array & Cluster.-** Para agrupar elementos.
5. **List & Table.-** Para visualizar y/o introducir datos.
6. **Graph.-** Para representar gráficamente los datos.
7. **Ring & Enum.-** seleccionar una lista opciones.
8. **Containers.-** Control de parámetros
9. **I/O.-** Datos digitales, VISA, IMAQ, DAQ.
10. **Dialog Controls.-** Controles variables.
11. **Classic Controls.-** Controles Clásicos.

12. **Refnum.**- Números de referencia para controles e indicadores
13. **Decorations.**- Elementos de demarcación
14. **User Controls.**- Para elegir un control creado por el propio usuario.
15. **Select a Control .**- Para seleccionar cualquier control

Al seleccionar objetos desde el menú *Controls* estos aparecen sobre el panel frontal, pueden colocarse donde convenga, y además tienen su propio menú desplegable que permite la configuración de algunos parámetros específicos de cada tipo de control.

Nota: situar diferentes controles en el panel frontal para poder visualizarlos y observar su aspecto en tanto en el panel frontal como en el diagrama de bloques

3.3 Diagrama de bloques

El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el panel frontal.

El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LabVIEW. En el lenguaje “G” las funciones y las estructuras son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales.

Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los terminales. A continuación se presenta un ejemplo de lo recién citado:

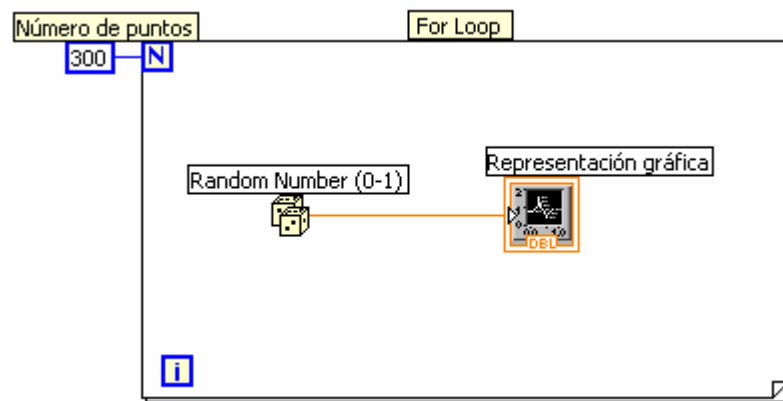


Figura 3.4 Diagrama de bloques

- Función (*Random number*).
- Terminales de control e indicadores (Número de puntos y representación gráfica).
- Estructura (*for loop*).

El diagrama de bloques se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos.

LabVIEW posee una extensa biblioteca de funciones, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc.

Las estructuras, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while, case,...).

Los cables son las trayectorias que siguen los datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal etc. Cada cable tiene un color o un estilo diferente, lo que diferencia unos tipos de datos de otros.

3.3.1 Paleta de funciones (*functions palette*)

Se emplea en el diseño del diagrama de bloques. La paleta de funciones contiene todos los objetos que se emplean en la implementación del programa del VI, ya sean funciones aritméticas, de entrada/salida de señales, entrada/salida de datos a fichero, adquisición de señales, temporización de la ejecución del programa,...

Para poder visualizar las diferentes herramientas de visualización y control, pulsar sobre el **Diagrama de Bloques** con el botón derecho del ratón o en el menú *Windows* → *Show Functions Palette*.

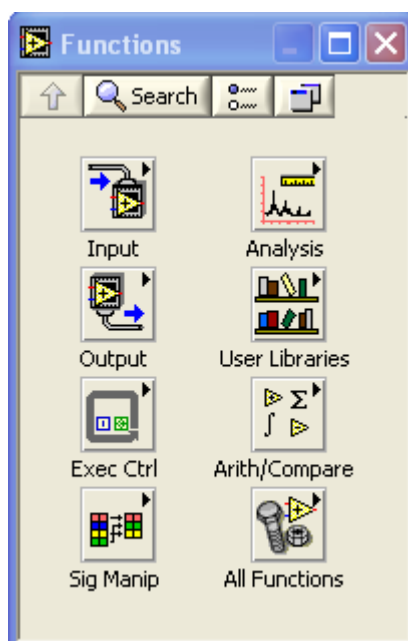
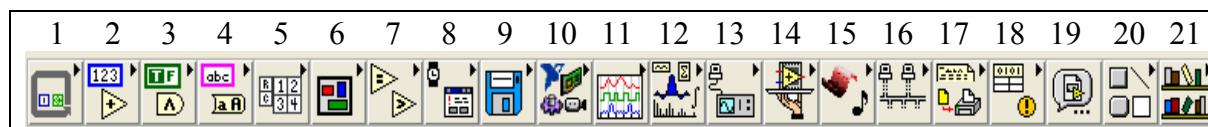


Figura 3.5 Paleta de Funciones.

El menú *All Fuctions* de la ventana correspondiente (Fig. 3.5) al panel frontal contiene las siguientes opciones:



1. **Structures.-** Muestra las estructuras de control del programa, junto con las variables locales y globales.
2. **Numeric.-** Muestra funciones aritméticas y constantes numéricas.
3. **Boolean.-** Muestra funciones y constantes lógicas.
4. **String.-** Muestra funciones para manipular cadenas de caracteres, así como constantes de caracteres.
5. **Array.-** Contiene funciones útiles para procesar datos en forma de vectores.
6. **Cluster.-** Un cluster es una agrupación de datos que incluso pueden ser de diferente tipo. Sería el equivalente en lenguaje C a un estructura.
7. **Comparison.-** Muestra funciones que sirven para comparar números, valores booleanos o cadenas de caracteres.
8. **Time & Dialog.-** Contiene funciones para trabajar con cuadros de diálogo, introducir contadores y retardos, etc.
9. **File I/O.-** Muestra funciones para operar con ficheros.
10. **NI Mesuraments.-** Propios.
11. **Waveform.-** Contiene generadores de señales.
12. **Analyze.-** Contiene un submenú en el que se puede elegir entre una amplia gama de funciones matemáticas de análisis.
13. **Instrument I/O.-** Muestra un submenú de VIs, que facilita la comunicación con instrumentos periféricos que siguen la norma ANSI/IEEE 488.2-1987, y el control del puerto serie.
14. **Aplication Control.-** Contiene varias funciones que regulan el funcionamiento de la propia aplicación en ejecución.
15. **Graphics & Sound.-** Contiene visualizadores gráficos y accesorios de audio.
16. **Communication.-** Muestra diversas funciones que sirven para comunicar varios ordenadores entre sí, o para permitir la comunicación entre distintos programas.
17. **Report Generation.-** Genera especificaciones.
18. **Advanced.-** Contiene diversos submenús que permiten el control de la ayuda, de los VIs, manipulación de datos, procesado de eventos, control de la memoria, empleo de programas ejecutables o incluidos en librerías DLL, etc.
19. **Select a VI.-** Permite seleccionar cualquier VI para emplearlo como subVI.
20. **Decoration.-** Contiene elementos indicadores.
21. **User Libraries.-** Muestra as librerías definidas por el usuario. En este caso, la librería mostrada contiene los drivers de la tarjeta de adquisición de datos de Advanced.

Nota: situar diferentes funciones en el diagrama de bloques para poder visualizarlos y observar su aspecto en tanto en el panel frontal como en el diagrama de bloques.

3.3.2 Paleta de herramientas (Tools palette)

Se emplea tanto en el panel frontal como en el diagrama de bloques. Contiene las herramientas necesarias para editar y depurar los objetos tanto del panel frontal como del diagrama de bloques.

Para poder visualizar las diferentes herramientas de la paleta de herramientas, pulsar sobre *Windows* → *Show Tools Palette*.

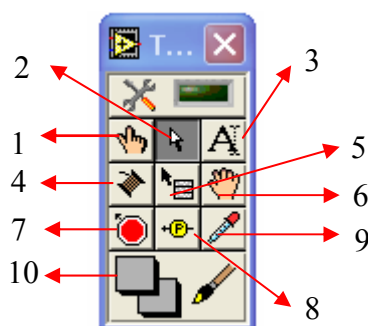


Figura 3.6 Paleta de herramientas

Las opciones que presenta esta paleta son las siguientes:

1. **Operate Value** .- Cambia el valor de los controles.
2. **Position Size Select**.- Desplaza, cambia de tamaño y selecciona los objetos.
3. **Edit Text** .- Edita texto y crea etiquetas.
4. **Connect Wire**.- Une los objetos en el diagrama de bloques.
5. **Object Shortcut Menu**.- Abre el menú desplegable de un objeto.
6. **Scroll windows**.- Desplaza la pantalla sin necesidad de emplear las barras desplazamiento.
7. **Set Clear Breakpoint**.- Fija puntos de interrupción de la ejecución del programa en VIs, funciones y estructuras.
8. **Probe data** .- Crea puntos de prueba en los cables, en los que se puede visualizar el valor del dato que fluya por dicho cable en cada instante.
9. **Get color** .- Copia el color para después establecerlo mediante la siguiente herramienta.
10. **Set Color** .- Establece el color de fondo y el de los objetos.

4 PROGRAMACIÓN EN LABVIEW

Con el entorno gráfico de programación de LabVIEW se comienza a programar a partir del panel frontal.

En primer lugar se definirán y seleccionarán de la paleta de controles todos los controles (entradas que dará el usuario) e indicadores (salidas que presentará en pantalla el VI), emplearán para introducir los datos por parte del usuario y presentar en pantalla los resultados.

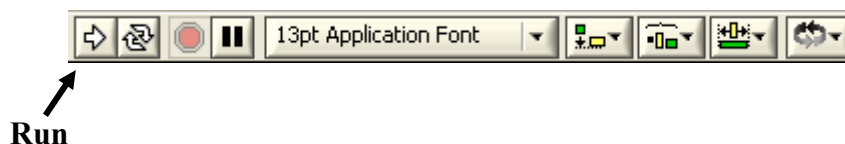
Una vez colocados en la ventana correspondiente al panel frontal todos los objetos necesarios, debe pasarse a la ventana *Diagram* menú **Windows** → **Show Diagram**, que es donde se realiza la programación propiamente dicha (diagrama de bloques). Al abrir esta ventana, en ella se encuentran los terminales correspondientes a los objetos situados en el panel frontal, dispuestos automáticamente por LabVIEW.

Se deben ir situando las funciones, estructuras, etc. que se requieran para el desarrollo del programa, las cuales se unen a los terminales mediante cables. Para facilitar la tarea de conexión de todos los terminales, en el menú “**Help**” puede elegirse la opción “**Show Help**”, con lo que al colocar el cursor del ratón sobre un elemento aparece una ventana con información relativa a éste (parámetros de entrada y salida). Además, si se tiene seleccionado el cursor de cableado, al situar éste sobre un elemento se muestran los terminales de forma intermitente.

5 EJECUCIÓN DE UN VI

Una vez se ha concluido la programación del VI se debe proceder a su ejecución. Para ello la ventana activa debe ser el panel frontal (si se está en la ventana del diagrama de bloques, se debe seleccionar la opción Window → **Show Panel del menú Window**).

Un a vez situados en el panel frontal, se pulsará el botón de Run, situado en la barra de herramientas.



El programa comenzará a ejecutarse. Mientras dura la ejecución del mismo, la apariencia del botón de Run es la que se muestra a continuación:



De este modo el programa se ejecutará una sola vez. Si se desea una ejecución continua, se pulsará el botón situado a la derecha del de Run (***Continuous Run***). Si durante el funcionamiento continuo del programa se vuelve a pulsar el citado botón, se finalizará la última ejecución del mismo, tras lo cual el programa se parará.



↖
Ejecución continua

Para finalizar la ejecución de un programa se puede realizar de dos formas. La primera y la más aconsejable, es emplear un botón en el panel frontal del VI, cuya pulsación produzca la interrupción del bucle de ejecución de la aplicación. La segunda forma de detener la ejecución del VI es pulsando el botón de pausa o el de stop. La diferencia entre ambos es que si se pulsa stop, la ejecución del programa finaliza inmediatamente, mientras que si se pulsa pausa, se produce una detención en el funcionamiento del programa, retomándose su ejecución una vez se vuelva a pulsar el mismo botón.



↖
Stop

↖
Pausa

6 ESTRUCTURAS

En la paleta de funciones la primera opción es la de las estructuras. Éstas controlan el flujo del programa, bien sea mediante la secuenciación de acciones, ejecución de bucles, etc.

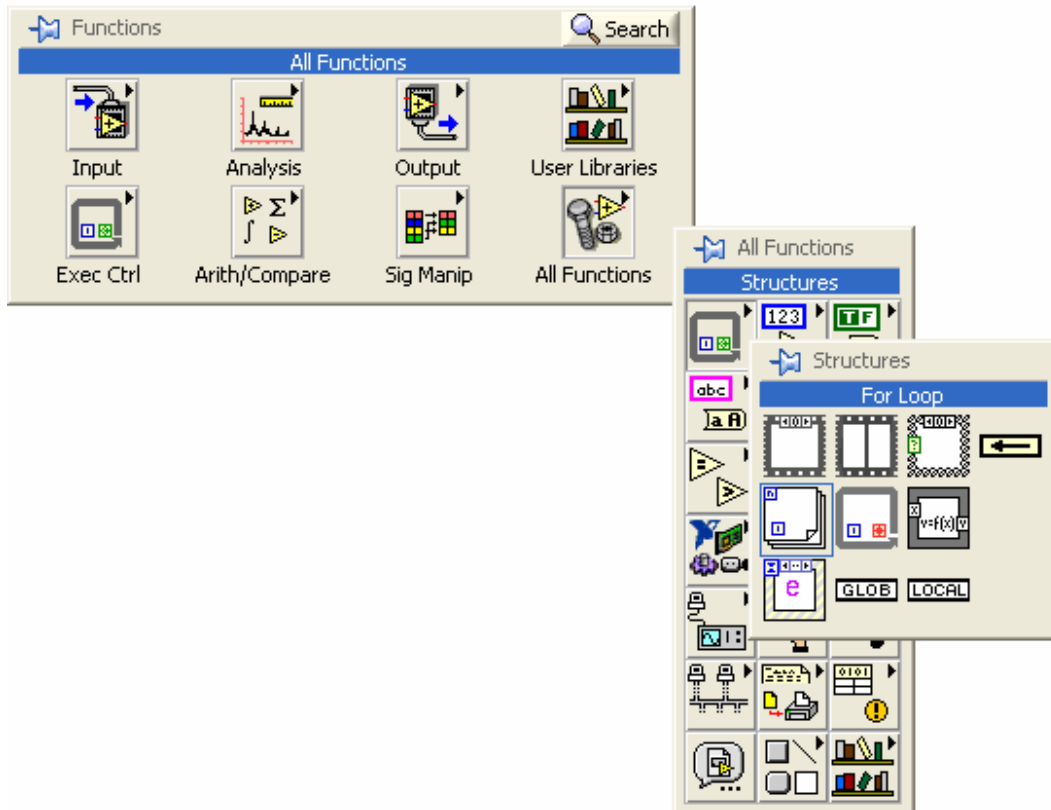


Figura 6.1 Estructuras

Las estructuras se comportan como cualquier otro nodo en el diagrama de bloques, ejecutando automáticamente lo que está programado en su interior una vez tiene disponibles los datos de entrada y una vez ejecutadas las instrucciones requeridas. Son suministrados los correspondientes valores a los cables unidos a sus salidas, sin embargo, cada estructura ejecuta su subdiagrama de acuerdo con las reglas específicas que rigen su comportamiento y que se especifican a continuación.

Un subdiagrama es una colección de nodos, cables y terminales situados en el interior del rectángulo que constituye la estructura. El For Loop y el While Loop únicamente tienen un subdiagrama. El Case Structure y el Sequence Structure, sin embargo, pueden tener múltiples subdiagramas superpuestos como si se tratara de cartas en una baraja, por lo que en el diagrama de bloques únicamente será posible visualizar en el mismo instante de tiempo uno de ellos. Los subdiagramas se construyen del mismo modo que el resto del programa. Las siguientes estructuras se hallan disponibles en el lenguaje G.

6.1 Case Structure

Al igual que otras estructuras posee varios subprogramas (diagramas), que se superponen como si de una baraja de cartas se tratara. En la parte superior del subprograma aparece el identificador del que se está representando en pantalla. En ambos lados de este identificador aparecen unas flechas que permiten pasar de un subprograma a otro.

En este caso el identificador es un valor que selecciona el subprograma que se debe ejecutar en cada momento.

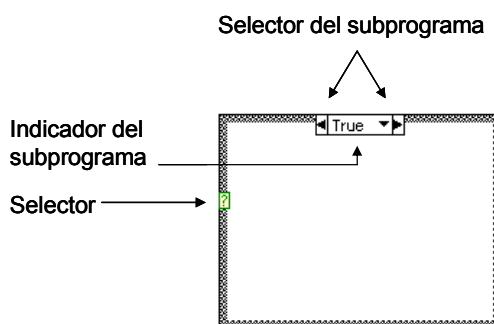


Figura 6.2 Case estructura

La estructura Case tiene al menos dos subprograma (*True and False*). Únicamente se ejecutará el contenido de uno de ellos, dependiendo del valor de lo que se conecte al selector.

6.2 Sequence Structure

De nuevo, este tipo de estructuras presenta varios subdiagramas, superpuestos como en una baraja de cartas, de modo que únicamente se puede visualizar una en pantalla.

También poseen un identificador del subdiagrama mostrado en su parte superior, con posibilidad de avanzar o retroceder a otros subdiagramas gracias a las flechas situadas a ambos lados del mismo.

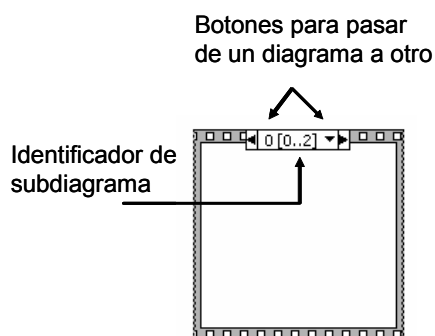
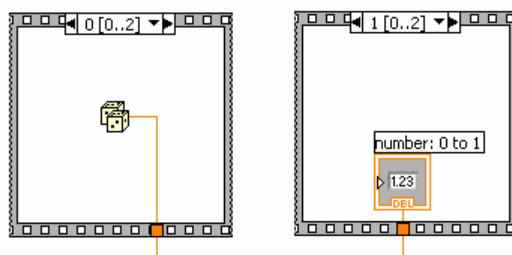


Figura 6.3 Sequence structure

Esta estructura secuencia la ejecución del programa. Primero ejecutará el subdiagrama de la hoja (*frame*) nº0, después el de la nº 1, y así sucesivamente. Para pasar datos de una hoja a otra se pulsará el botón derecho del ratón sobre el borde de la estructura, seleccionando la opción *Add sequence local*.



Paso de un dato de la frame 0 a la 1

Figura 6.4: Local sequence

6.3 For Loop

Es el equivalente al bucle *for* en los lenguajes de programación convencionales. Ejecuta el código dispuesto en su interior un número determinado de veces.

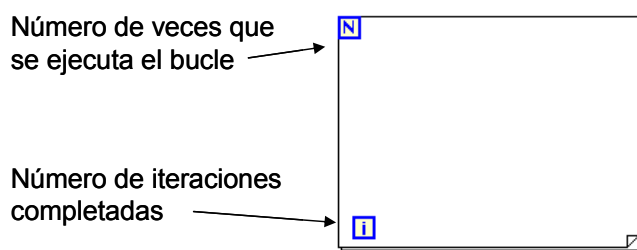


Figura 6.5 For Loop

Ejecutar el bucle *for* sería equivalente al siguiente fragmento de código:

```
For i = 0 to N-1
  Ejecutar el subdiagrama del interior del Bucle
```

Para pasar valores de una iteración a otra se emplean los llamados *shift registers*. Para crear uno, se pulsará el botón derecho del ratón mientras éste se halla situado sobre el borde del bucle, seleccionando la opción **Add Shift Register**. El shift register consta de dos terminales, situados en los bordes laterales del bloque. El terminal izquierdo almacena el valor obtenido en la iteración anterior. El terminal derecho guardará el dato correspondiente a la iteración en ejecución. dicho dato aparecerá , por tanto, en el terminal izquierdo durante la iteración posterior.

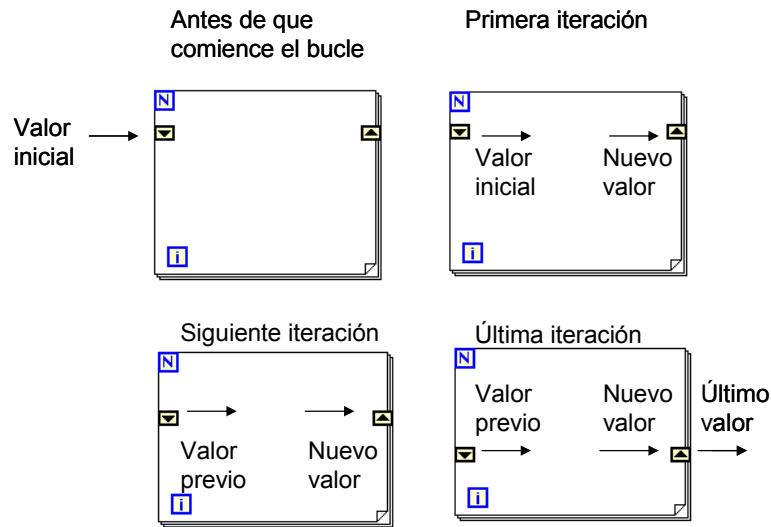


Figura 6.6: Shift registers

Se puede configurar un *shift register* para memorizar valores de varias iteraciones previas. Para ello, con el ratón situado sobre el terminal izquierdo del *shift register*, se pulsará el botón derecho, seleccionando a continuación la opción **Add Element**.

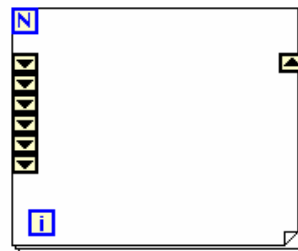


Figura 6.7 Shift register varias iteraciones

6.4 While Loop

Es el equivalente al bucle while empleado en los lenguajes convencionales de programación. Su funcionamiento es similar al del bucle for.

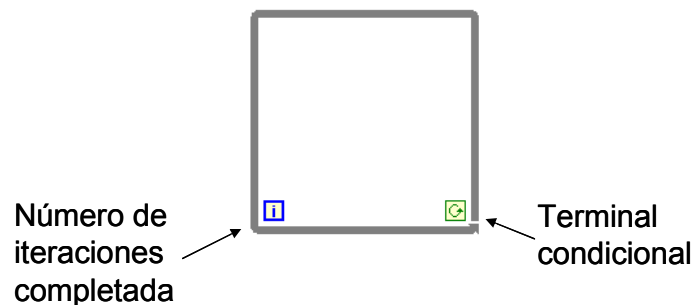


Figura 6.8 While Loop

El bucle while es equivalente al código siguiente:

Do
 Se ejecuta lo que hay en el interior del bloque
while terminal condicional is true

El programa comprueba el valor de lo que se halle conectado al terminal condicional al finalizar el bucle. Por lo tanto, el bucle siempre se ejecuta al menos una vez.

Con esta estructura también se pueden emplear los *shift registers* para tener disponibles los datos obtenidos en iteraciones anteriores (es decir, para memorizar valores obtenidos). Su empleo es análogo al de los bucles for, por lo que omitiré su explicación.

6.5 Formula Node

La estructura denominada “Formula Node” se emplea para introducir en el diagrama de bloques fórmulas de un modo directo. Resulta de gran utilidad cuando la ecuación tiene muchas variables o es relativamente compleja. Por ejemplo, se desea implementar la ecuación:

$$y = x \cdot x + x + 1$$

Empleando bloques pertenecientes al lenguaje G quedaría:

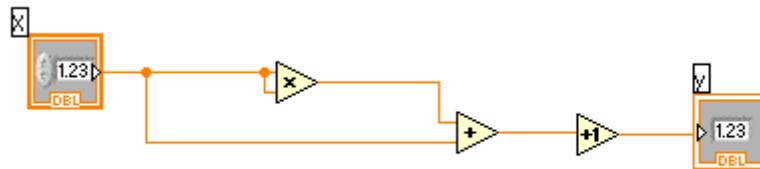


Figura 6.9 Formulas con lenguaje G

Si se utiliza la “formula node”, se obtiene:



Figura 6.10: Formulas con Formula Node

Para definir una fórmula mediante esta estructura, se actuará del siguiente modo:

1. En primer lugar, se deben definir las variables de entrada y las de salida. Para ello, se pulsa con el botón derecho del ratón sobre el borde de la fórmula **node**. A continuación se seleccionará *Add Input o Add Output*, según se trate de una entrada o una salida, respectivamente. Aparecerá un rectángulo, en el que se debe escribir el nombre de la variable (se distingue entre mayúsculas y minúsculas). Todas las variables que se empleen deben estar declaradas como entradas o salidas. Las que se empleen como variables intermedias se declararán como salidas, aunque posteriormente no se unan a ningún bloque posterior.
2. Una vez definidas las variables a emplear, se escribirán la o las fórmulas en el interior del recuadro (para ello se emplea el *edit text*). Cada fórmula debe finalizar con un “;”.
3. Los operadores y funciones que se pueden emplear se explican en la ayuda de LabVIEW, y son los que se muestran a continuación:

Operadores:

Asignación	=
Condicional	?:
OR lógico	
AND lógico	&&
Relacionales	== != > < >= <=
Aritméticos	+ - * / ^

Funciones:

Abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, rand, rem, sec, sgn, sin, sinc, sinh, sqrt, tan, tanh,

La sintaxis de una expresión incondicional es la siguiente:

```

<expresión condicional> ? <texpresión> : <fexpresión>

```

Si el valor lógico de la expresión condicional es *true* se ejecutará la *texpresión*. Si, por el contrario, fuese *false*, lo que se aplicará será *fexpresión*

Como ejemplo considérese el siguiente fragmento de código:

```

if (x >= 0) then
y = sqrt (x)
else
y = -99
end if

```

Se puede implementar este fragmento de código empleando un formula node, tal y como se muestra en la siguiente figura:

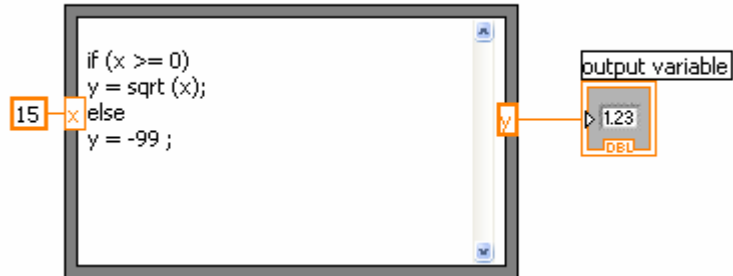


Figura 6.11 Formula Node

7 EJEMPLO: CONSTRUCCIÓN DE UN VI

En este apartado se mostrará cómo construir una aplicación mediante el empleo del entorno de programación que proporciona LabVIEW.

7.1 Panel frontal

En primer lugar, se debe construir el panel frontal deseado, que en este ejemplo debe tener el siguiente aspecto:

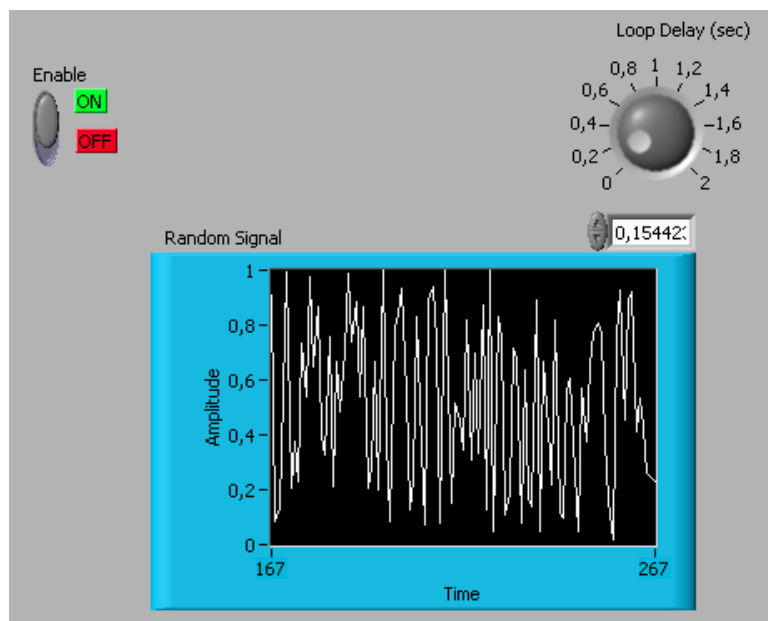


Figura 7.1 Construcción Panel Frontal

Proceso a seguir:

1. Abrir un panel frontal nuevo.

2. Colocar un "**vertical switch**" (paleta Boolean), cuyo nombre será *Enable*. Su finalidad será finalizar la adquisición.



3. Emplear el **Edit Text** para crear una etiqueta libre para ON y OFF. Utilizar el **Set Color** para hacer que el borde de dicha etiqueta tenga volumen. La T en el borde inferior izquierdo de la paleta de colores hace transparente un objeto.

4. Colocar el gráfico **waveform chart**, situado en la paleta **Graph**. Su nombre será *Random Signal*. El gráfico representará valores aleatorios en tiempo real.

5. El gráfico tiene un display digital que muestra el último dato. Pulsar el botón derecho del ratón situado sobre el gráfico, y seleccionar **Digital Display** del submenú **Visible Items**. Asimismo se deberá deseleccionar **Legend** y **Palette** del mismo submenú .



6. Empleando el **Edit Text**, pulsar dos veces con el botón izquierdo del ratón sobre el 10.0 en el eje Y del gráfico, introducir 1 y pulsar fuera del gráfico. Repetir para el -10. Así se habrá cambiado el fondo de escala.

7. Colocar un **knob (paleta Numeric)**, cuyo nombre será **Loop Delay (sec)**. Este control determinará la velocidad de ejecución del bucle. Pulsar sobre él con el botón derecho del ratón y deseleccionar **Digital Display** del submenú Show.



8. Empleando el **Edit Text**, pulsar dos veces con el botón izquierdo del ratón sobre el 10.0 de la escala, introducir 2 y pulsar fuera del control para introducir el nuevo valor.

7.2 Diagrama de bloques

El siguiente es el aspecto que presentará el diagrama de bloques una vez finalizada su construcción:

1. Abrir el diagrama de bloques (**menú Window, opción Show Diagram**).

2. Colocar el **While Loop** (subpaleta **Structures** de la paleta de funciones). Dicha estructura, como todas las demás es de tamaño ajustable.

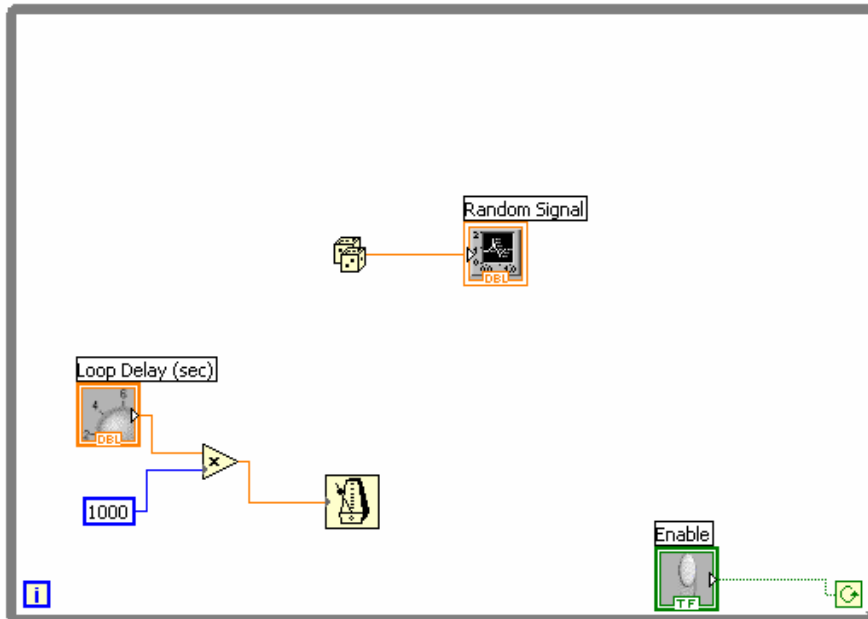







Figura 7.2 Construcción diagrama Bloques

 3. Seleccionar la función **Random Number (0-1)** de la subpaleta **Numeric** del menú de funciones.

 4. Seleccionar la función **Wait until Next ms Multiple** de la subpaleta **Time & Dialog** del menú de funciones.

 5. Seleccionar la función de **multiply** de la subpaleta **Numeric**, del menú de funciones, así como una constante numérica, introduciendo el valor 1000 en lugar de 0, que es el que aparece por defecto.

 6. Colocar los cables tal y como se muestra en la figura anterior, empleando para ello la **Connect Tool**.

 7. Volver al panel frontal. Con la **Operate Value** poner el interruptor en su posición ON. Ejecutar el programa pulsando el botón run. . La frecuencia de ejecución de las iteraciones del bucle **While** es la indicada en el panel frontal con el control **Loop Delay (sec)**. Es decir, se generará y representará un valor aleatorio cada periodo de tiempo (en segundos) seleccionado.

8. Para finalizar la ejecución del bucle, colocar el interruptor en la posición de OFF. De ese modo la condición de ejecución del bucle **While** será falsa, por lo que se detendrá a la siguiente iteración.

8 ADQUISICIÓN DE DATOS

En este apartado el estudiante debe materializarse con la tarjeta de adquisición de datos *National Instruments PCI 6014*. El fin es conseguir el dominio, manejo y configuraciones de las I/O analógicas y digitales, como también del software de programación visual, el cuál incorpora diversas funciones.

8.1 Tarjeta de adquisición de datos

Para la realización de las prácticas será utilizada la tarjeta genérica de bajo coste (PCI 6014) de la casa *National Instruments*.

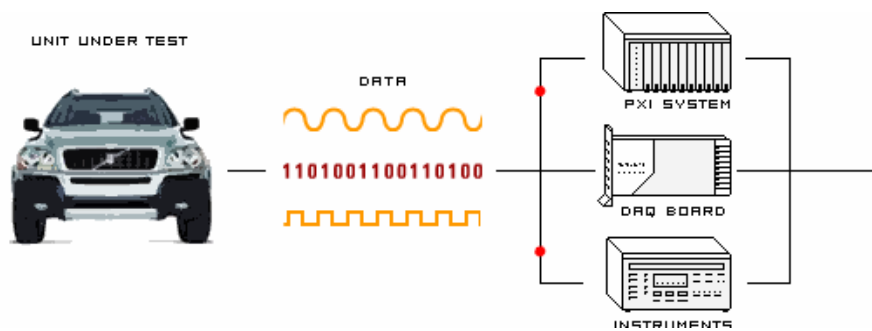


Figura 8.1. Test de diferentes sensores

NI PCI-6014



Figura 8.2. Tarjeta de adquisición de datos.

Las características más importantes son:

Entradas analógicas:

- 16 o 8 diferenciales, máximo 200 kS/s con 16 bits de resolución.
- Rango de entradas seleccionable entre ± 10 .
- Controlador DMA para la transferencia de datos de las entradas analógicas a la memoria del PC.
- Scan secuencial de la entrada 0 a la N.
- La conversión A/D puede inicializarse por software, un generador de pulsos interno, o una entrada externa.

Salidas analógicas:

- Dos canales de salida con un máximo de 10kS/s y 16 bits de resolución.
- Rango de entradas seleccionable entre ± 10 .
- La salida analógica puede inicializarse por software, un generador de pulsos interno, o una entrada externa.

Entradas y salidas digitales:

- 8 I/O digitales (TTL/CMOS);
- Dos contadores/ timers de 24 bits.

Para la conexión de las entradas y salidas de la tarjeta, se dispone de de una tarjeta de I/O de conexionado tal como se muestra (Fig. 8.3), donde la relación de los pines se muestra en (Fig. 8.4).



Figura 8.3. Tarjeta de conexionado de las entradas y salidas

ACH8	34	68	ACH0
ACH1	33	67	AIGND
AIGND	32	66	ACH9
ACH10	31	65	ACH2
ACH3	30	64	AIGND
AIGND	29	63	ACH11
ACH4	28	62	AISENSE
AIGND	27	61	ACH12
ACH13	26	60	ACH5
ACH6	25	59	AIGND
AIGND	24	58	ACH14
ACH15	23	57	ACH7
DAC0OUT	22	56	AIGND
DAC1OUT	21	55	AO GND
EXTREF	20	54	AO GND
DIO4	19	53	DGND
DGND	18	52	DIO0
DIO1	17	51	DIO5
DIO6	16	50	DGND
DGND	15	49	DIO2
5 V	14	48	DIO7
DGND	13	47	DIO3
DGND	12	46	SCANCLK
PFI0/TRIG1	11	45	EXTSTROBE*
PFI1/TRIG2	10	44	DGND
DGND	9	43	PFI2/CONVERT*
5 V	8	42	PFI3/GPCTR1_SOURCE
DGND	7	41	PFI4/GPCTR1_GATE
PFI5/UPDATE*	6	40	GPCTR1_OUT
PFI6/WFTRIG	5	39	DGND
DGND	4	38	PFI7/STARTSCAN
PFI9/GPCTR0_GATE	3	37	PFI8/GPCTR0_SOURCE
GPCTR0_OUT	2	36	DGND
FREQ_OUT	1	35	DGND

AI 8	34	68	AI 0
AI 1	33	67	AI GND
AI GND	32	66	AI 9
AI10	31	65	AI 2
AI 3	30	64	AI GND
AI GND	29	63	AI 11
AI 4	28	62	AI SENSE
AI GND	27	61	AI 12
AI 13	26	60	AI 5
AI 6	25	59	AI GND
AI GND	24	58	AI 14
AI 15	23	57	AI 7
DAC 0 OUT1	22	56	AI GND
DAC 1 OUT1	21	55	AO GND
RESERVED	20	54	AO GND
P0.4	19	53	D GND
D GND	18	52	P0.0
P0.1	17	51	P0.5
P0.6	16	50	D GND
D GND	15	49	P0.2
+5 V	14	48	P0.7
D GND	13	47	P0.3
D GND	12	46	AI HOLD
PFI 0/AI START	11	45	EXT STROBE
PFI 1/REF TRIG	10	44	D GND
D GND	9	43	PFI 2/AI CONV
+5 V	8	42	PFI 3/CTR 1 SEC
D GND	7	41	PFI 4/CTR 1 GATE
PFI 5/AO SAMP	6	40	CTR 1 OUT
PFI 6/AO START	5	39	D GND
D GND	4	38	PFI 7/AI SAMP
PFI 8/CTR0 GATE	3	37	PFI 8/CTR 0 SRC
CTR 0 OUT	2	36	D GND
FREQ OUT	1	35	D GND

Figura 8.4 Relación de pines entre la tarjeta de I/O y el conector de la tarjeta de adquisición de datos.

8.2 Entradas Analógicas

Para configurar los parámetros de adquisición de datos de una entrada hay diferentes formas, la tradicional DAQ, la DAQmx y con el DAQ *assistant*.

El primer método no se recomienda ya que sus características de productividad, rendimiento y precisión son inferiores a los otros dos métodos.

El método descrito a continuación corresponde a la configuración de una entrada analógica mediante el DAQ *assistant*.

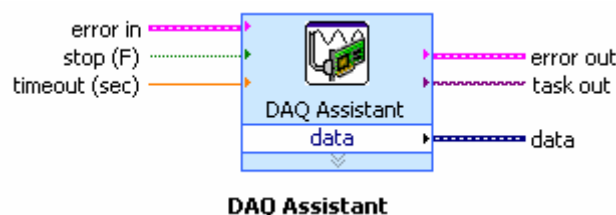


Figura 8.5. DAQ assitant

Encontraremos el DAQ *assistant* en la subcarpeta de **Input**. La colocaremos en el diagrama de bloques y si hacemos doble clic podremos configurar la tarea de adquisición de datos.

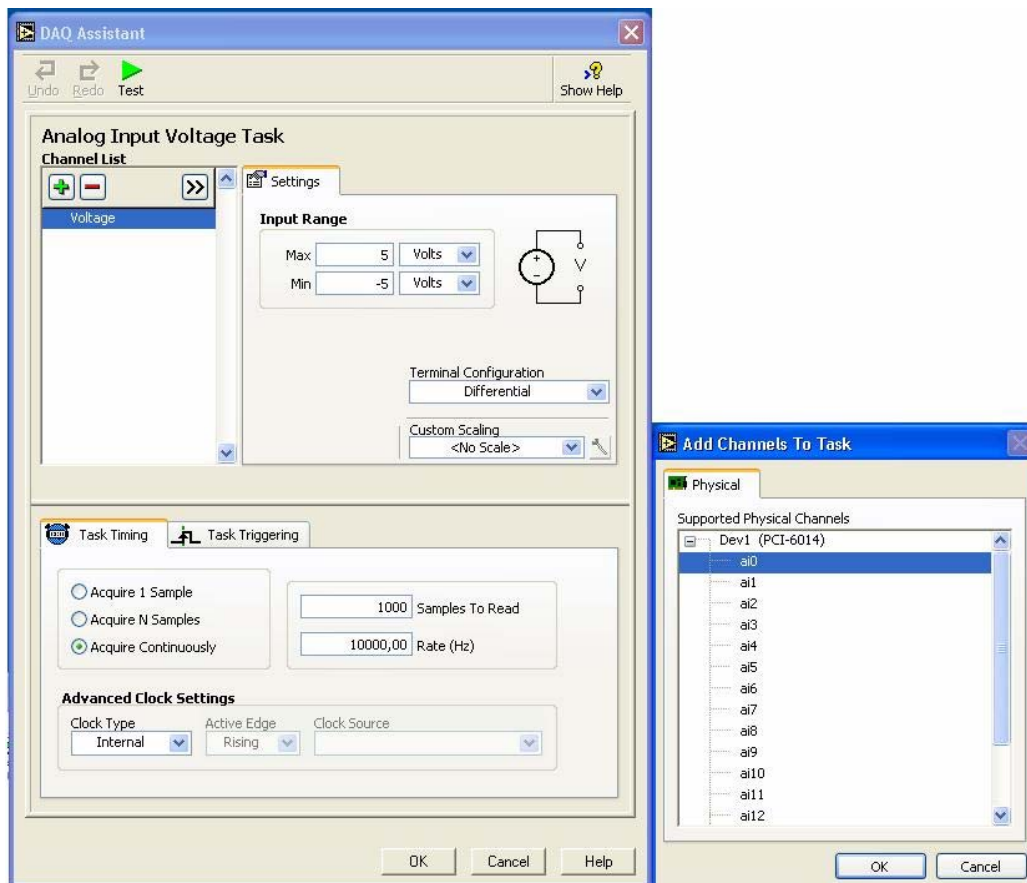


Figura 8.6. Configuración del DAQ assistant

Se pueden ver en (Fig. 8.6) los parámetros a configurar que son los siguientes:

- El rango de tensión de la entrada (**de 5V a -5V**)
- El canal de adquisición (**ai0**),
- El modo de adquisición:
 - *adquiere 1 muestra*
 - *adquiere N muestras*
 - *adquiere continuously*
- Frecuencia de adquisición, por ejemplo **10000 Hz**,
- Numero de muestras a leer (*Samples to read* de **1000**) .
- Terminal configuration. Se define el modo de entrar la señal. Hay dos opciones:
 - Referenciado a masa.
 - De modo diferencial. En este modo la señal a medir se conecta a dos entradas. Por ejemplo el canal 0 y 8, o el 1 y 9, el 2 y 10, ... Por consiguiente en modo diferencial solo disponemos de 8 entradas.

NOTA: realizar el siguiente ejemplo, en el que se tendrá que configurar la tarjeta con los parámetros que se han descrito anteriormente.

Una vez configurado el DAQ *assistant*, para que la adquisición de datos sea continua, lo pondremos dentro de un bucle *while* con un *Waveform Graph* en la salida de datos para poder visualizar nuestra entrada.

Nuestra entrada a adquirir será un seno de amplitud 2Vpp y frecuencia 50Hz que generaremos con el generador de funciones del laboratorio.

Teniendo en cuenta que hemos seleccionado una entrada diferencial (lo que nos implica una entrada de menor ruido) y por el canal **ai0**, la conexión de los pines de la placa de I/O corresponde al **pin 68** para **entrada positiva** y **pin 34** para la **entrada negativa**

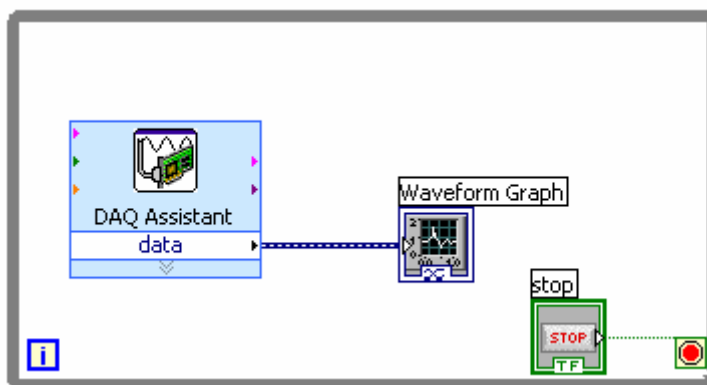


Figura 8.7. Vista final del programa

Nota: Realizar este ejemplo y verificar su funcionamiento.

8.3 Salidas Analógicas

En el caso de las salidas analógicas seguiremos los mismos pasos que en el caso anterior, con la diferencia de que configuraremos el DAQ *assistant* como salida por el canal **ao0**. Se generará un seno con el bloque *Simulate signal* (especificaremos los datos de la figura 8.8) y se verificará su buen funcionamiento con la lectura de la salida con el osciloscopio del laboratorio.

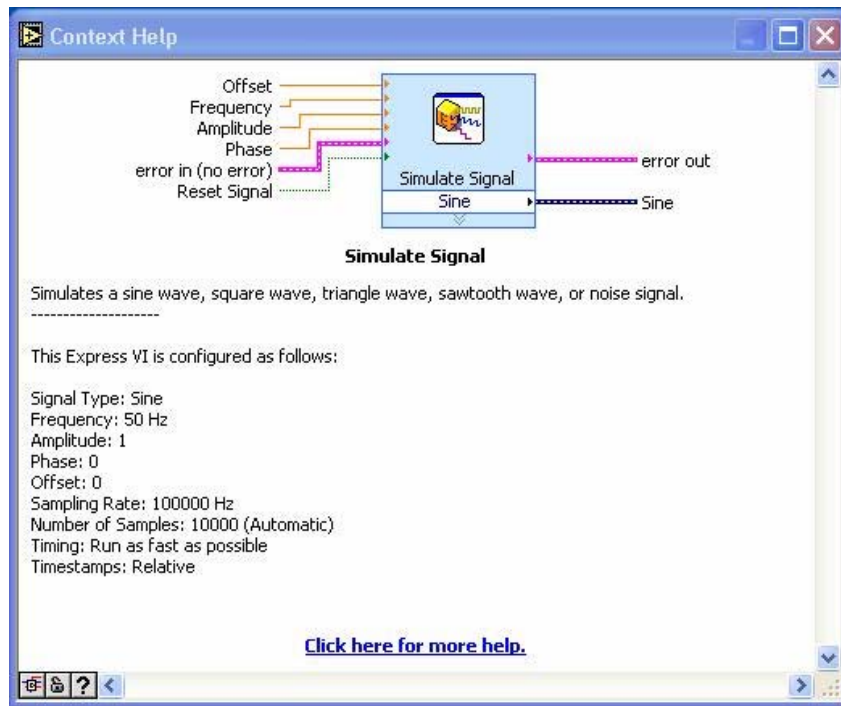


Figura 8.8 Generador de funciones

Teniendo en cuenta que la salida en este caso no es diferencial y por el canal **ao0**, la conexión a los pines de la placa de I/O corresponde al **pin 22** para **salida positiva** y **pin 56** para **GND**.

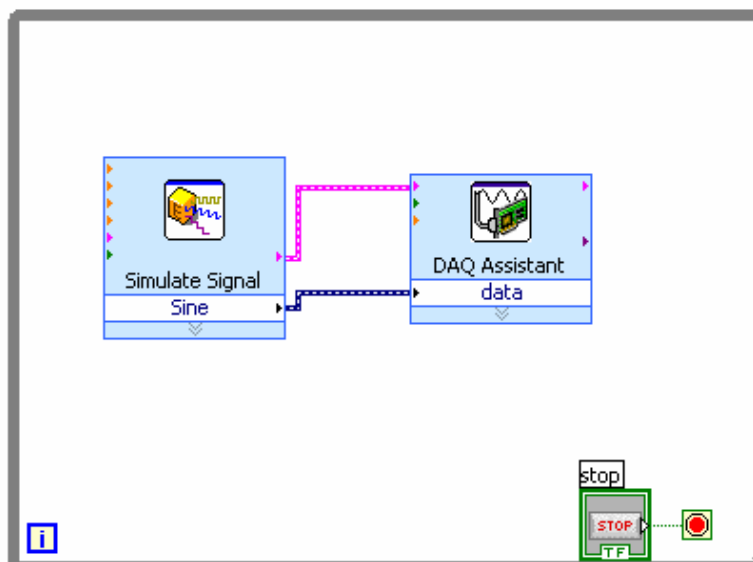


Figura 8.9. Vista final del programa

Nota: Realizar este ejemplo y verificar su funcionamiento.