MANUAL BÁSICO DE PROGRAMACIÓN EN LABVIEW. MASTERHACKS

www.masterhacks.net

ÍNDICE

INTRODUCCIÓN

Este manual está dirigido a cualquier persona que esté interesada en aprender a programar en LabVIEW. Se verán los conceptos básicos y ejemplos para que el lector pueda asimilar todo de una manera más clara y rápida.

Es importante saber que para poder aprender a programar en LabVIEW se requieren conocimientos básicos de electrónica y programación en cualquier lenguaje de código escrito, ya que el entorno de programación en LabVIEW está diseñado preferentemente para adquisición de datos, diseño de sistemas, instrumentación y control instrumental.

En este documento se verá lo más básico para empezar a programar en este lenguaje, como el uso de arrays y clústeres, ciclos, nodos de propiedad, etc.

Para poder crear y ejecutar los programas en LabVIEW es necesario descargar el software desde la página oficial de National Instruments; http://www.ni.com/trylabview/esa/ .

En caso de tener alguna duda o comentario, favor de hacerlo llegar al correo contacto@masterhacks.net

¿QUÉ ES LABVIEW?

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) es un entorno de desarrollo y diseño de sistemas con un lenguaje visual gráfico. LabVIEW utiliza el lenguaje G (lenguaje gráfico) que acelera la productividad o desarrollo de programas para una mejor eficiencia en el desarrollo de sistemas.

Es un software creado por la empresa National Instruments en 1976 y sacado al mercado en 1986.

Al desarrollar un programa en LabVIEW, se crea un Instrumento Virtual o VI que contiene la interfaz gráfica del programa y el diagrama de bloques (código).

Cuando un programa está terminado, el usuario final hace uso del panel frontal, donde se encuentra todo lo necesario para controlar un sistema. El diagrama de bloques es donde se encuentra el código del programa, es donde el programador accede para modificar o mantener el programa.

Actualmente, el software de programación LabVIEW se puede utilizar en los sistemas operativos Microsoft Windows, Mac OS X, GNU/Linux.

Una de las principales características de LabVIEW es que es relativamente fácil de usar, no se requiere ser un experto en el área de programación para poder hacer un programa que se pudiera considerar como complejo o incluso imposible de hacer para algunos en otros lenguajes.

Aunque es necesario aclarar que para desarrollar programas que se apliquen a la automatización, control, adquisición y manejo de datos sí es necesario tener conocimientos más avanzados no solo de programación, sino de otras áreas específicas para cada aplicación del programa que se tenga planeado.

LabVIEW es principalmente utilizado por los ingenieros para el manejo de datos, la comunicación entre una computadora y un aparato o circuito externo es imprescindible para las aplicaciones que se le pueden dar al software, por lo que LabVIEW puede comunicarse con interfaces como:

- Puerto serial
- Puerto paralelo
- GPIB
- PXI
- VXI
- TCP/IP
- Irda
- Bluetooth
- USB
- OPC

Entre otros.

Esto es lo que hace del Laboratorio Virtual de Instrumentación una excelente opción para proyectos grandes donde se requieran conocimientos de programación, electrónica, mecánica, robótica, etc.
Este software es utilizado en empresas y agencias importantes como la NASA.

La programación representa un factor muy importante al hablar de proyectos de automatización y control, donde se especializan ingenieros mecatrónicos, robóticos y más. LabVIEW ofrece la herramienta para crear el software, el programa depende del programador y su funcionamiento, eficacia y utilidad dependerán de las habilidades de éste mismo.

“La potencia está en el software” Una frase muy célebre de LabVIEW, que hace referencia a la capacidad e importancia que puede tener un programa en un proyecto.

Conociendo LabVIEW

Al ejecutar el software LabVIEW, nos aparece una ventana como la siguiente:

Imagen #1: Pantalla de bienvenida de LabVIEW

Aquí podemos elegir del menú la opción de abrir un nuevo VI, un proyecto en blanco o un VI desde una plantilla. Para empezar elegiremos abrir un VI en blanco.

Nos aparecen dos ventanas, el Front Panel (Panel frontal) y Block Diagram (Diagrama de bloques). En el panel frontal, es donde podemos elegir los elementos que conformarán nuestro programa.

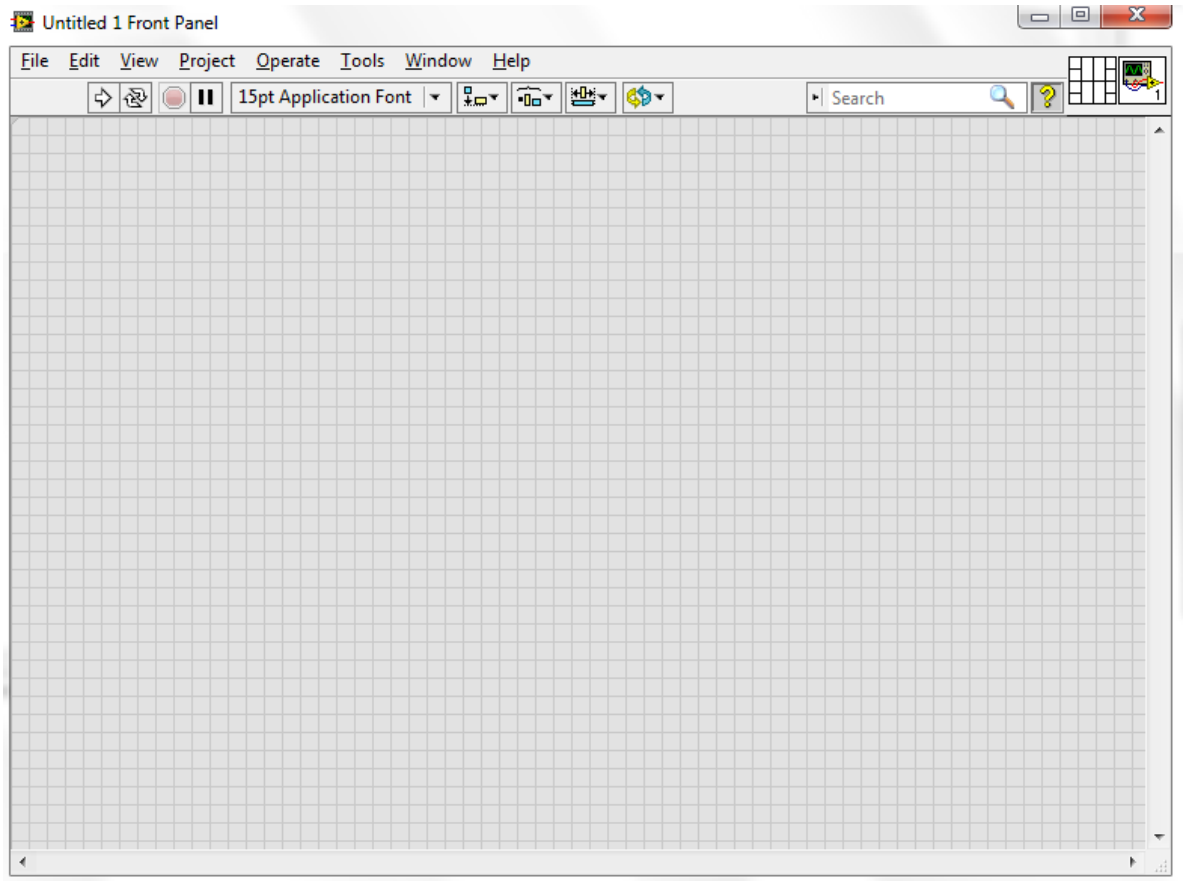


Imagen #2: Panel frontal del programa en LabVIEW

En la parte de arriba podemos encontrar el menú de ejecución, donde podemos ejecutar el programa, pausarlo o abortarlo;

Imagen #3: Menú de ejecución del panel frontal.

El primer botón (Run) sirve para ejecutar el programa, pero si el código no incluye un ciclo para que el programa se repita infinitamente o la cantidad de veces que el usuario elija, sólo parpadeará la pantalla y el programa no hará nada.
Si no se está trabajando con ciclos, es necesario hacer clic en el segundo botón (run continuously), que significa correr continuamente, de esta manera, el programa se ejecutará hasta que el usuario vuelva a dar clic en ese botón o realizar alguna acción en el programa que haga que éste se detenga.

El tercer botón sirve para abortar la ejecución. Este ícono se utiliza solamente cuando el programa se cicla, es decir; que no se puede detener por algún error en la programación. Esto sucede únicamente cuando se trabaja con ciclos y es recomendable no usarlo si no es absolutamente necesario, ya que el usarlo podría causar pérdida de datos.

El cuarto botón sirve para pausar la ejecución del programa.

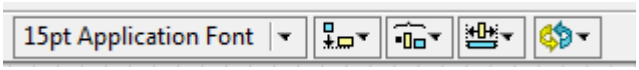


Imagen #4: Menú de estilos, alineación y tamaño de elementos.

El menú de la imagen 4 sirve para modificar el tamaño, fuente y color de las letras que se utilicen en la interfaz del programa. Los otros cuatro botones sirven para alinear y mover los elementos que se estén utilizando para dar un aspecto más estético.

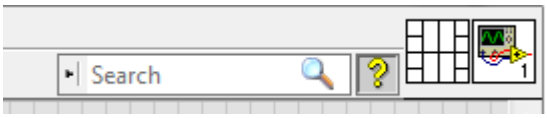


Imagen #5: Barra de búsqueda y propiedades del VI

En la esquina superior derecha se encuentra la barra de búsqueda y el botón de ayuda. También están dos íconos, el primero sirve para incluir sub VI´s en el programa, aquí se construye el panel conector para poder utilizar sub VI´s.

El otro ícono sirve para ver y modificar las propiedades del VI (clic derecho>VI properties).

Desde ahí se puede editar el tamaño de la ventana del programa, proteger el código con contraseña, el uso de CPU, etc.

También se puede editar el ícono dando clic derecho> Edit icon.

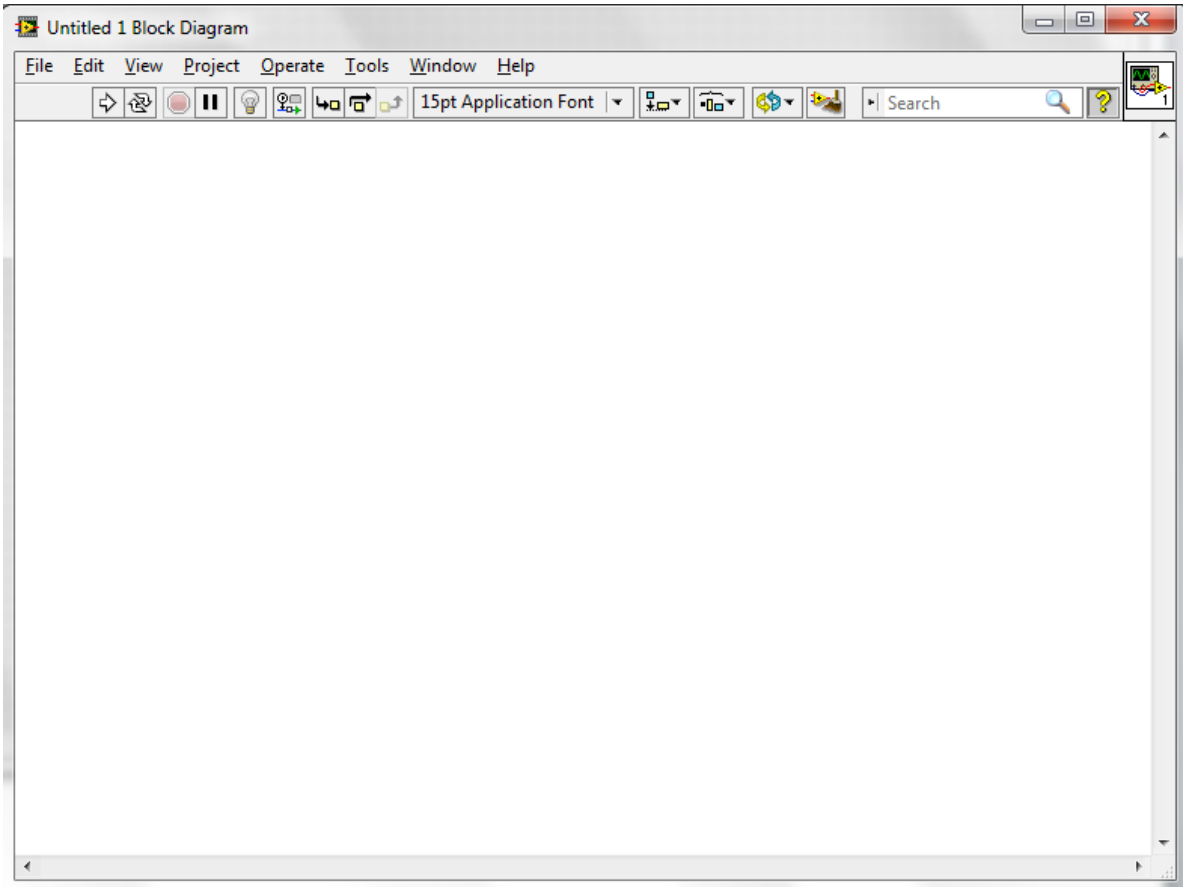


Imagen #6: Diagrama de bloques del programa.

En el diagrama de bloques los menús son muy parecidos, solo que aquí se agrega un botón al menú de ejecución que sirve para ver paso a paso lo que el diagrama de bloques está ejecutando, muy útil cuando se quiere ver el orden de ejecución de todo el código y ver su comportamiento.

Otro botón que se agrega es el que limpia todo el código, para poner todos los elementos de una forma más ordenada y legible.

Para empezar a colocar elementos en el panel frontal, podemos dar clic derecho en cualquier parte del panel y aparece el menú con todos los elementos, o se puede dar clic en View>Controls palette.

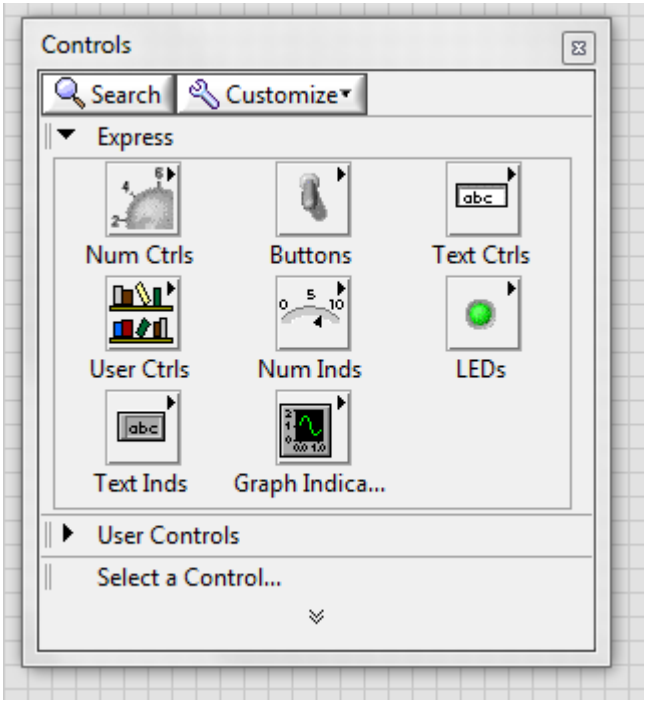


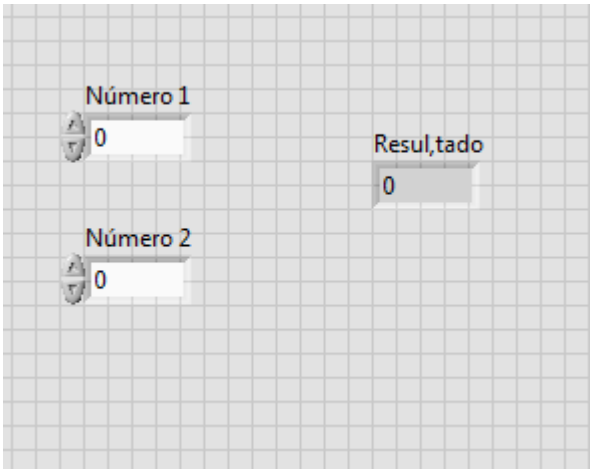
Imagen #7: Menú de elementos.

En ese menú podemos elegir cualquier control numérico, booleano, de texto, etc. Nos ofrece una gran cantidad de elementos que podemos usar en una infinidad de programas.

Para ver más elementos, solo se necesita dar clic en la flechita de abajo del menú para desplegar el menú completo.

EL PRIMER PROGRAMA EN LABVIEW

Como primer programa, podemos crear una sumadora sencilla de dos números. Para esto, necesitamos en el panel frontal dos controles numéricos y un indicador numérico.



Imagen# 8: Panel frontal de la sumadora.

Al crear los elementos en el panel frontal, estos aparecen automáticamente en el diagrama de bloques:

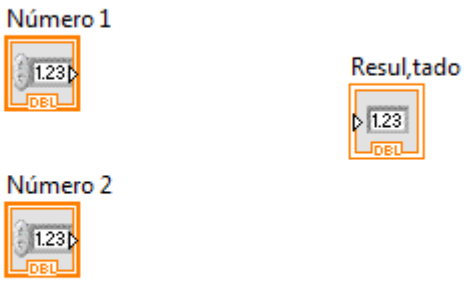


Imagen #9: Diagrama de bloques de la sumadora.

Aquí ya empieza el trabajo del programador, realizar las operaciones necesarias para que el programa funcione como se desea. Para seleccionar las funciones que se utilizarán, solo se tiene que dar clic derecho sobre el diagrama de bloques para que aparezca el menú, igual que en el panel frontal.

En este caso solo usaremos una simple suma:

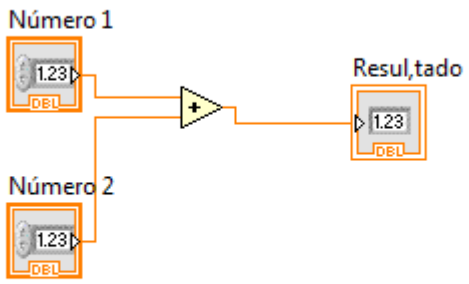


Imagen #10: Conexión de los elementos del diagrama de bloques.

Hasta aquí ya podemos ejecutar el programa, como no se está utilizando ciclo, le damos clic en correr continuamente.

Imagen #11: Programa en funcionamiento.

Podemos modificar los valores y el resultado se actualiza automáticamente mientras el programa se esté ejecutando continuamente. Para detenerlo solo hay que presionar nuevamente ese botón.

Si queremos cambiar la función de suma por otra como resta o multiplicación, solo damos clic derecho sobre el ícono y en replace elegimos la función que se requiera.

CONTROLES E INDICADORES

Los elementos de cada programa de LabVIEW se pueden definir como controles o indicadores. Como se vio en el programa anterior, se usaron dos controles para ingresar los números que se van a sumar. Con esto podemos deducir que un control es un elemento que puede ser controlado o modificado por el usuario, normalmente con perillas o flechas de incremento.

Por otro lado, los indicadores son los elementos que no pueden ser modificados por el usuario, y que solamente muestran datos o resultados que el programa arroja, por ejemplo en el programa anterior, el indicador muestra el resultado de la suma.

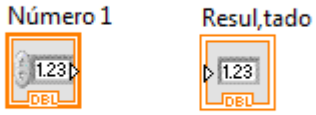


Imagen #12: Diferencias entre indicador y control.

En el diagrama de bloques es fácil distinguir un control de un indicador, el control tiene el color del borde más grueso y la flechita de lado derecho, indicando que el cable o datos van a salir del elemento, mientras que el indicador tiene el borde más delgado y la flechita del lado izquierdo, indicando que va a recibir un cable o datos.

Para cambiar un control a indicador o viceversa, solo se le da clic derecho al ícono y otro clic en change to control o change to indicator, según se requiera.

ELEMENTOS BOOLEANOS

Como se debe saber, los elementos booleanos funcionan con un cero o un uno, siendo falso o verdadero respectivamente, no tienen otro valor más que esos.

En LabVIEW los elementos booleanos se representan con el color verde y con las letras t o f. Estos son botones, leds, comparadores, entre otros.

Un ejemplo muy sencillo del uso de estos, es para encender un led con un switch, solo necesitamos un switch y un led, conectados entre sí, para poder encender el led con los valores verdadero o falso.

Imagen #13: Conexión de un switch y un led.

De esta forma al accionar el switch se enciende el led.

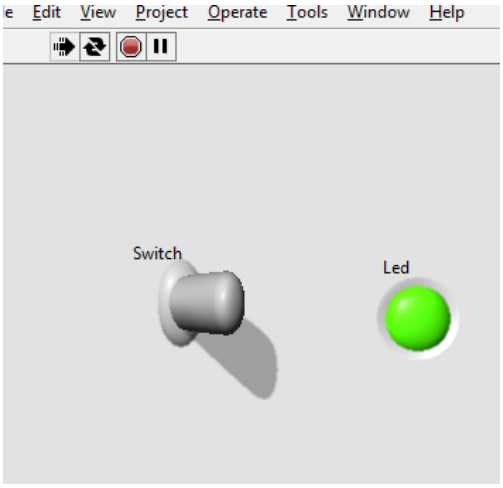


Imagen #14: Ejemplo del uso de elementos booleanos.

USO DE SELECT

En LabVIEW podemos encontrar dentro del menú comparisson (comparación), un ícono llamado select. Este funciona como la estructura condicional if.
Si la condición es verdadera, se devuelve determinado valor o función, si es falsa realiza la acción correspondiente.

Se puede apreciar un ejemplo muy sencillo del uso de select en el siguiente programa:

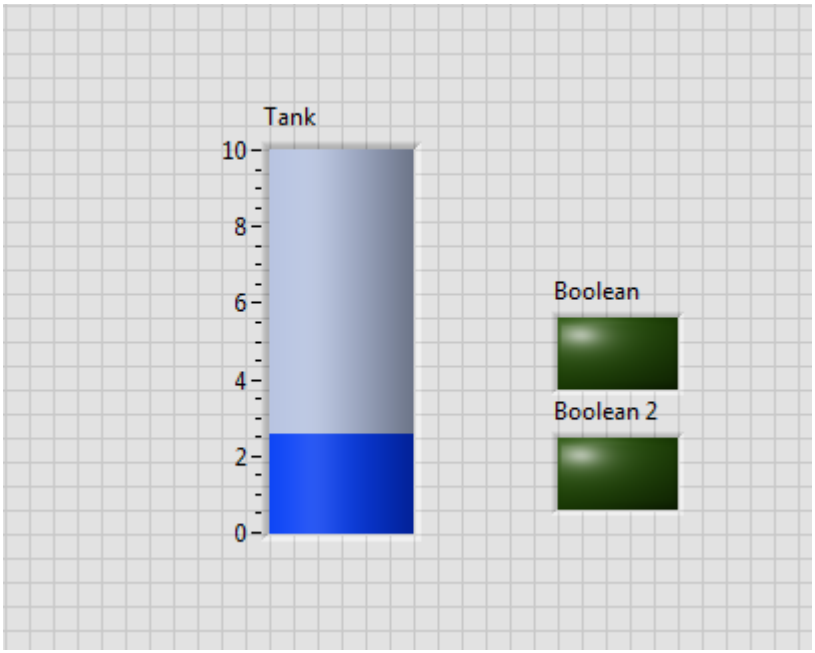


Imagen #15: Panel frontal para ejemplo del uso de select.

El objetivo del control numérico es que si es menor de 5, el led de abajo enciende, si es mayor de 5, se apaga el led de abajo y prende el de arriba.

Para esto utilizaremos dos select, uno para cada led.

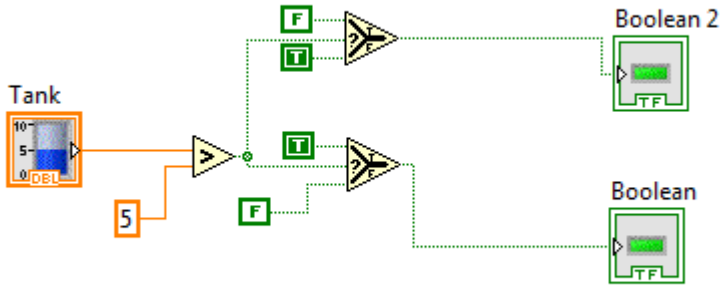


Imagen 16: Diagrama de bloques del ejemplo del uso de select.

Se utiliza un comparativo para el control numérico, en este caso mayor que. Si el dato ingresado por el usuario en el tanque es mayor a 5, prende el led de arriba, si es menor, prende el de abajo y se apaga el de arriba.
Nótese que se utilizaron constantes booleanas para los select, de esta manera se hace más efectivo su uso cuando no se utilizan ciclos.

USO DE CADENAS DE TEXTO

Si queremos trabajar con texto en LabVIEW utilizaremos los controles e indicadores de texto. En el diagrama de bloques éstos tienen un color rosa, al igual que los elementos booleanos y numéricos, los strings tienen control, indicador y constante.

Para mostrar un texto determinado se utiliza una constante de texto, y para que ésta sea visible en el panel frontal, se utiliza un indicador de texto.

Usando el ejemplo anterior, podemos agregar un indicador de texto para visualizar los estados del programa:

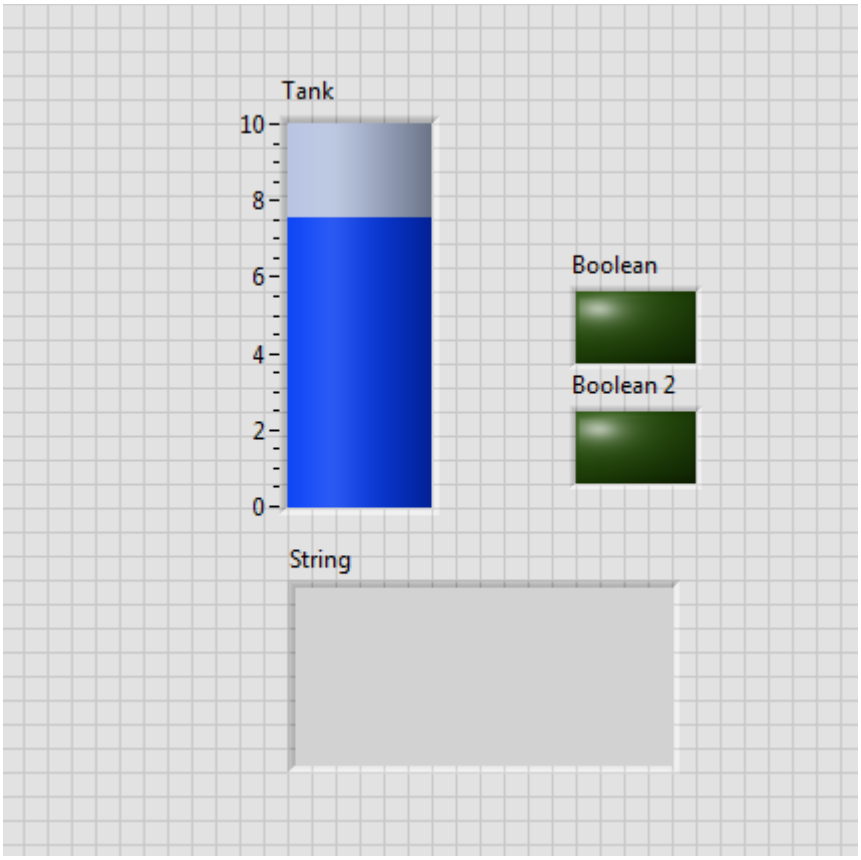


Imagen #17: Ejemplo del uso de cadenas de texto.

El objetivo del indicador textual es que si el nivel es mayor o menor de 5, se indique con un mensaje. Para esto podemos utilizar otro select conectado a cualquier de los dos leds, para evaluar si este está encendido o prendido, las constantes de texto estarán conectadas al select y la salida al indicador.

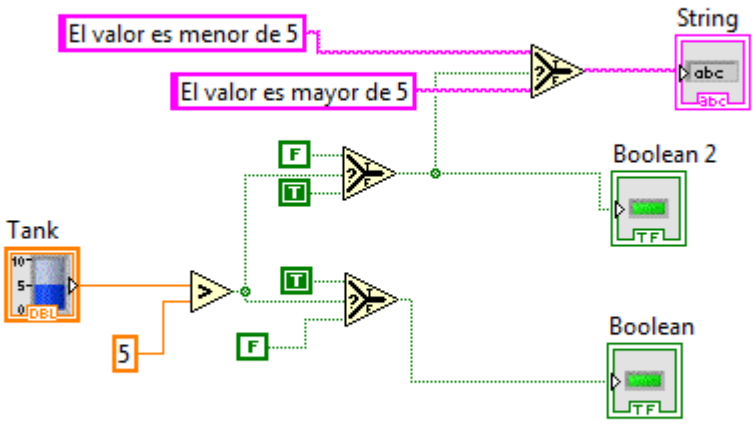


Imagen #18: Diagrama de bloques del ejemplo de uso de strings.

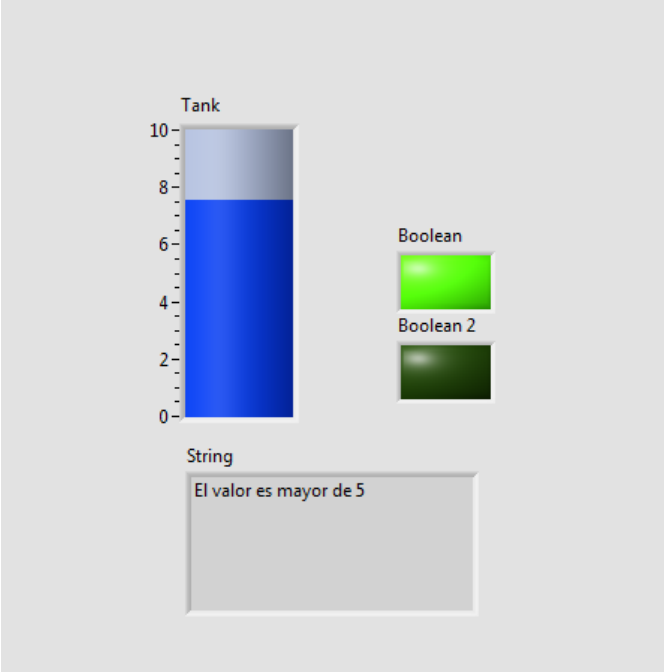


Imagen #19: Programa funcionando

USO DE LA ESTRUCTURA CASE

Asumiendo que ya se sabe que la estructura case sirve para ejecutar o albergar los posibles casos o acciones en función de una expresión.
En el lenguaje de programación C, la sintaxis es:

```
switch (funcion)
{
        case1:
                Sentencias;
                break;
                case 2:
                        Sentencias;
                        break;
}
```

Aquí, se ejecuta la estructura case al evaluar la función, en este caso tiene dos casos con diferentes sentencias cada uno.

En el lenguaje gráfico, las estructuras de control se representan mediante rectángulos. El case se conecta al elemento que requiera tener varias opciones y todas las operaciones se van guardando en cada caso.

Se puede ver el funcionamiento de la estructura case en el siguiente ejemplo:

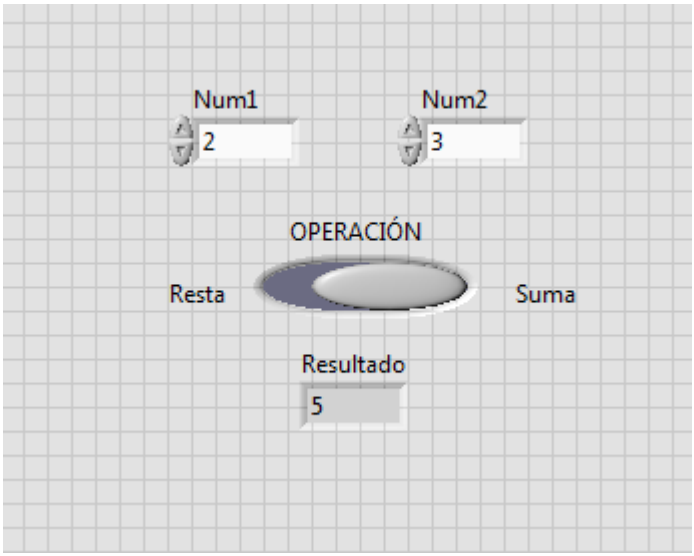


Imagen #20: Ejemplo del uso de la estructura case.

El programa es una calculadora de dos funciones, suma y resta, se piden dos números al usuario y con un botón elige si quiere que esos números se resten o se sumen.

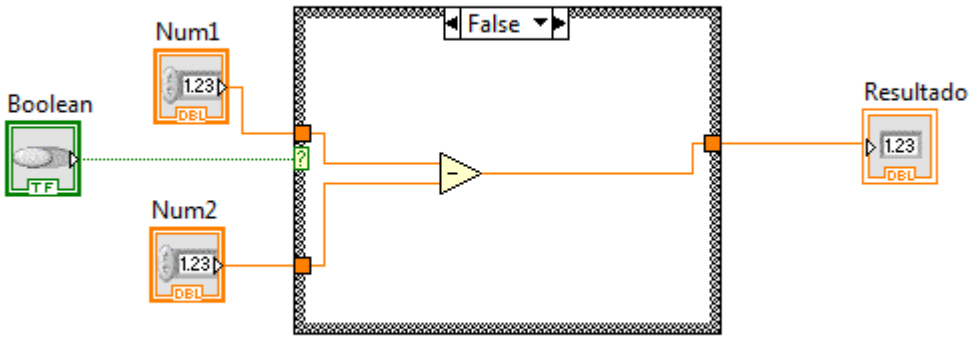


Imagen #21: Diagrama de bloques del ejemplo de uso de case.

Como se mencionó anteriormente, el botón se conecta al case, dando como resultado dos únicos casos, falso o verdadero. Al tratarse de un botón slide, el falso es cuando el botón está del lado izquierdo y el verdadero del lado derecho.

Al analizar la imagen 21 podemos notar que lo único que hay dentro del case es la operación que se va a realizar. Esto es porque si metemos cualquiera de los controles o indicadores, éstos solo servirán para un solo caso, y para que el programa funcione correctamente se tendrían que crear variables locales de cada elemento para colocarlos en el otro caso, pero eso es innecesario y haría que el código quede más grande.

En cambio, dejando los elementos afuera, al conectarlos al case, crean un cuadrito (túnel) en donde se pueden usar para los demás casos sin tener que crear variables.

En el case también se puede conectar un tab control, útil para dar un mejor aspecto al programa, almacenando cada operación o acción en un contenedor individual. El tab control se encuentra en el menú Containers (Contenedores).

Podemos ver un ejemplo con un programa para hacer los cálculos de la ley de Ohm:

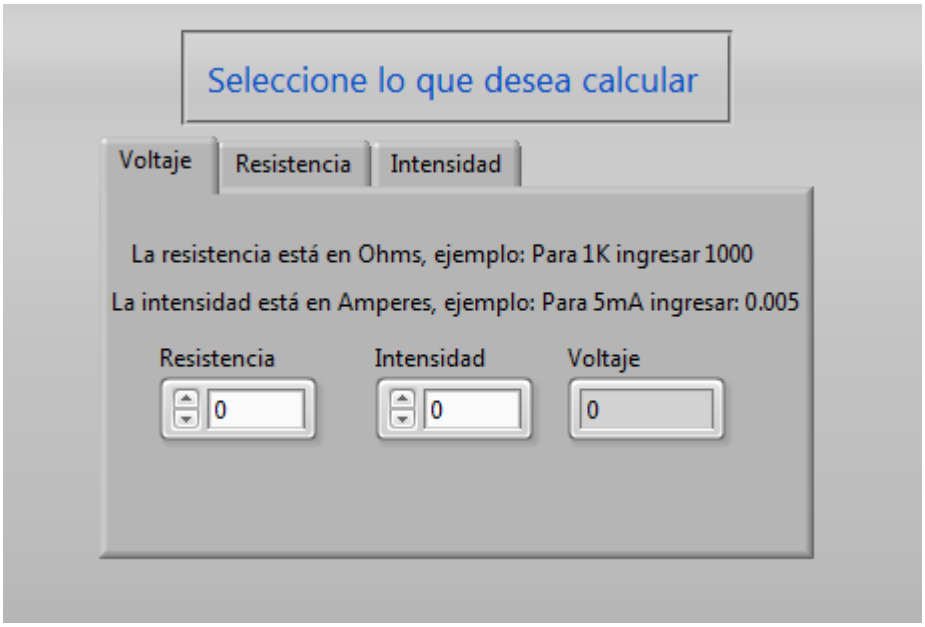


Imagen #22: Ejemplo del uso de tab control y case.

Aquí se usan tres pestañas para el tab control, (para añadir más pestañas se da clic derecho sobre cualquier pestaña y luego en add page after ó add page before). Cada pestaña representa el valor que se quiere obtener, ya sea voltaje, resistencia o intensidad.

En cada pestaña hay dos controles numéricos y un indicador numérico, el usuario ingresa los dos valores y el programa hace los cálculos pertinentes.

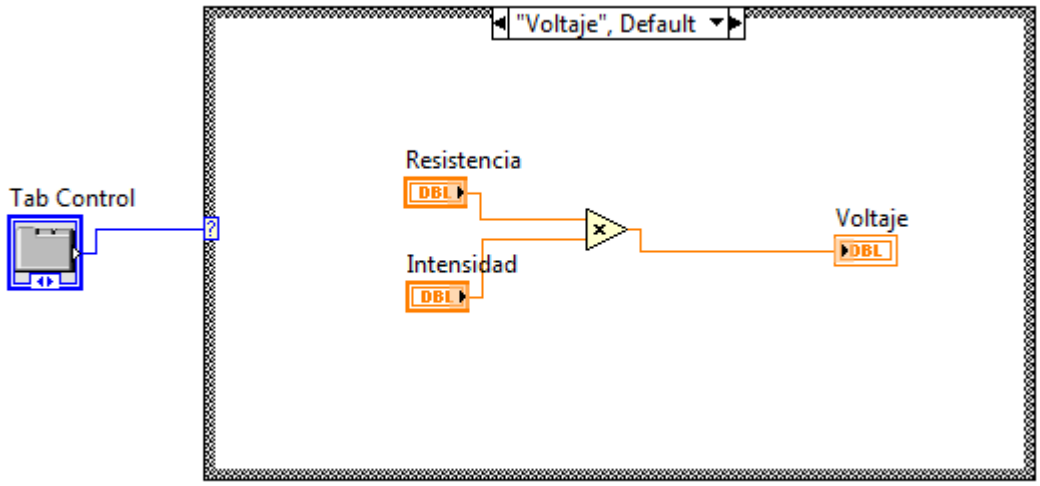


Imagen #23: Código del programa de ejemplo de uso de tab control.

Como se puede apreciar en la imagen 23, aquí si se meten todos los elementos a cada caso, esto porque se está utilizando un tab control y resulta mejor crear controles e indicadores para cada caso. Siguiendo la ley de Ohm, solo se hacen operaciones sencillas de división y multiplicación.

Nota: Al conectar el tab control al case, automáticamente aparecen los dos primeros casos, para agregar más casos a esta estructura, se da clic derecho al nombre del caso en la parte de arriba y luego en add case before o after según se requiera.

Si queremos utilizar cadenas de texto en un case que esté conectado a un botón, es muy simple. El case solo tendrá dos casos, y por ello solo utilizaremos dos mensajes.

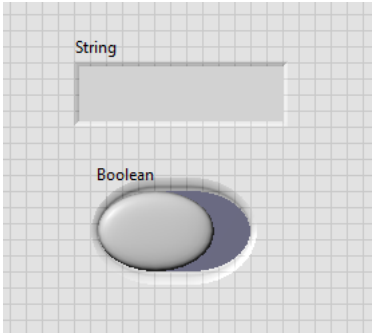


Imagen #23: Ejemplo del uso de strings con case.

El diagrama de bloques queda como se muestra:

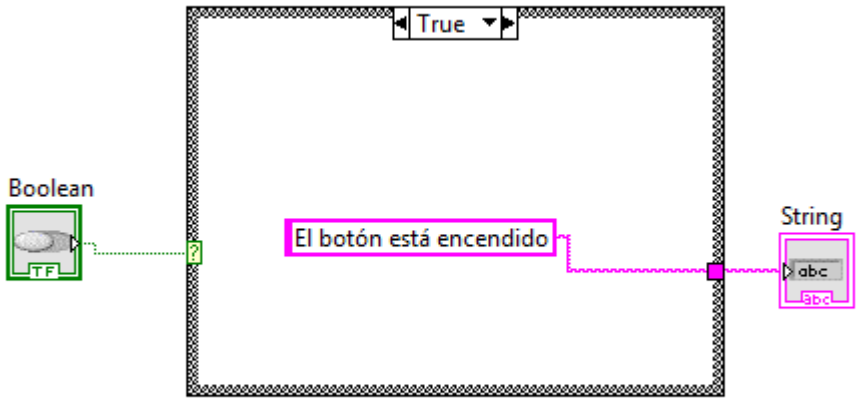


Imagen #24: Código del ejemplo de string en case.

En este caso, cuando el programa se corra continuamente, saldrá el mensaje del caso false ya que seguramente el botón estará en false cuando se corra el programa, al accionar el botón, se mostrará el mensaje del caso true.

USO DE ARRAY

El array (matriz o vector en español), se utiliza para almacenar varios elementos, siempre del mismo tipo. En un array no se pueden meter objetos que sean de distintos tipos, es decir; si se mete un led a un array, ese array sólo será de leds.

En LabVIEW, si se requiere usar cada elemento de un array por separado, se utilizan las herramientas del diagrama de bloques para poder hacerlo, tales como index array, initialize array, etc.

Al colocar un array en el diagrama de bloques, solo se le arrastra un objeto hacia dentro y el array toma automáticamente el tamaño del objeto. Para hacer más grande el array (que tenga más elementos del mismo tipo) solo se da clic en la flechita de tamaño ya sea abajo o arriba del array y se desplaza hacia el mismo lado.

Si se requiere que el array sea de dos dimensiones, se le da clic derecho y se da clic en add dimension.

Un ejemplo del uso de arrays puede ser el siguiente:

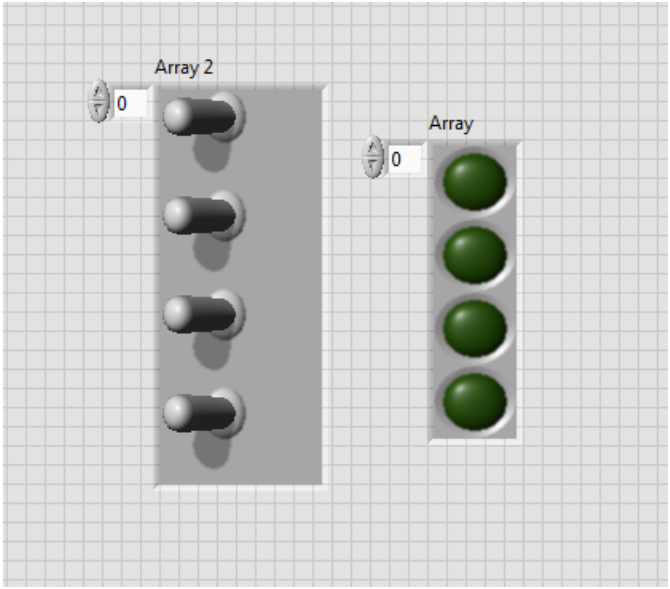


Imagen #25: Uso de Array en LabVIEW

En el panel frontal se tienen dos arrays, uno de leds y otro de switch.
En el diagrama de bloques sólo nos saldrán dos elementos, que son los arrays, si los conectamos, el programa funcionará perfectamente.

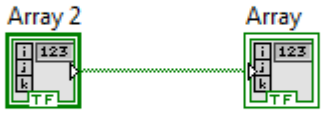


Imagen #26: Diagrama de bloques del ejemplo de Arrays.

Esto nos reduce de manera notable el código del programa, ya que si no usamos arrays en este programa, en el diagrama de bloques habría cuatro leds y cuatro switches, lo que haría muy grande el código.

Algo que se debe aclarar es que si se quiere controlar el array de cuatro leds con un solo switch fuera de un array, no se puede, ya que son elementos de distintos tipos.

Al conectar los dos arrays de esta forma, se respeta el orden de encendido, es decir; al activar el switch 1, se enciende el switch 1, y así sucesivamente. Si se requiere cambiar el orden de encendido, por ejemplo, que al activar el switch 1 se encienda el led 4, al activar el switch 2 se encienda el led 3, y así sucesivamente, se puede utilizar la herramienta Index array, para inicializar los elementos del array en el orden que se requiera.

Después, las salidas del array indexado se van conectan a la herramienta Build array, que construye un array para conectarlo al siguiente array.

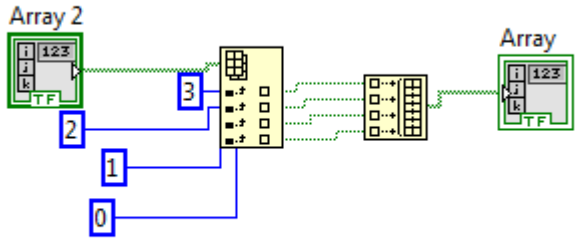


Imagen #27: Uso de Index array y Build array.

Como se ve en la figura 27, se le dio un orden ascendente a los elementos del array de switches, esas salidas se conectaron a un build array que a su vez se conectó al array de leds.
Al correr el programa debe funcionar correctamente.

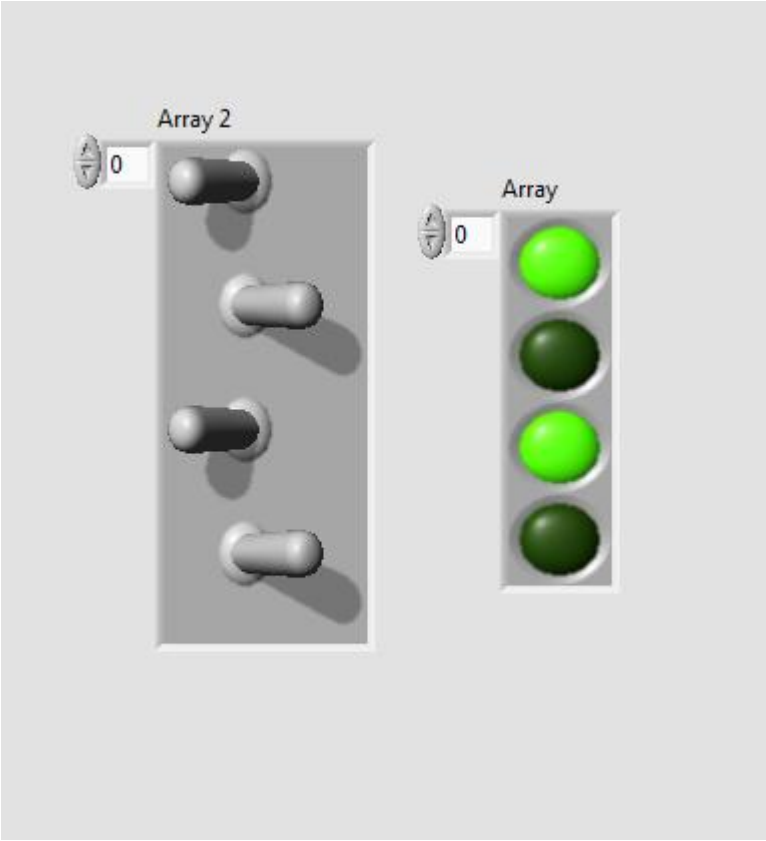


Imagen #28: Funcionamiento del programa con Index array.

USO DE ARRAYS CON STRINGS

Si queremos utilizar cadenas de texto para mostrar mensajes, en elementos que están los arrays es sencillo, podemos utilizar el ejemplo anterior para mostrar esto:

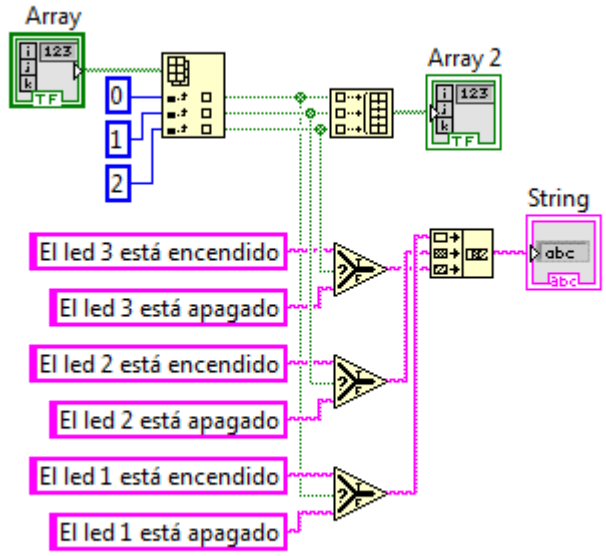


Imagen #29: Uso de string y array.

Como se puede apreciar en la imagen, el array de switches se indexa, para sacar los elementos y ordenarlos, esto con el fin de poder conectar cada elemento a un select que evalúa si el switch está activado o no.

Si es falso, con una constante de texto se dice que el led está apagado, si es verdadero, dice que está encendido.

Lo interesante aquí es cómo mandar las tres salidas de texto al indicador. Simple, usamos la herramienta Concatenate string, que agrupa todas constantes de texto y saca una sola salida, como si se estuviera construyendo un array de texto y éste se conecta al indicador.

El resultado al correr el programa es el siguiente:

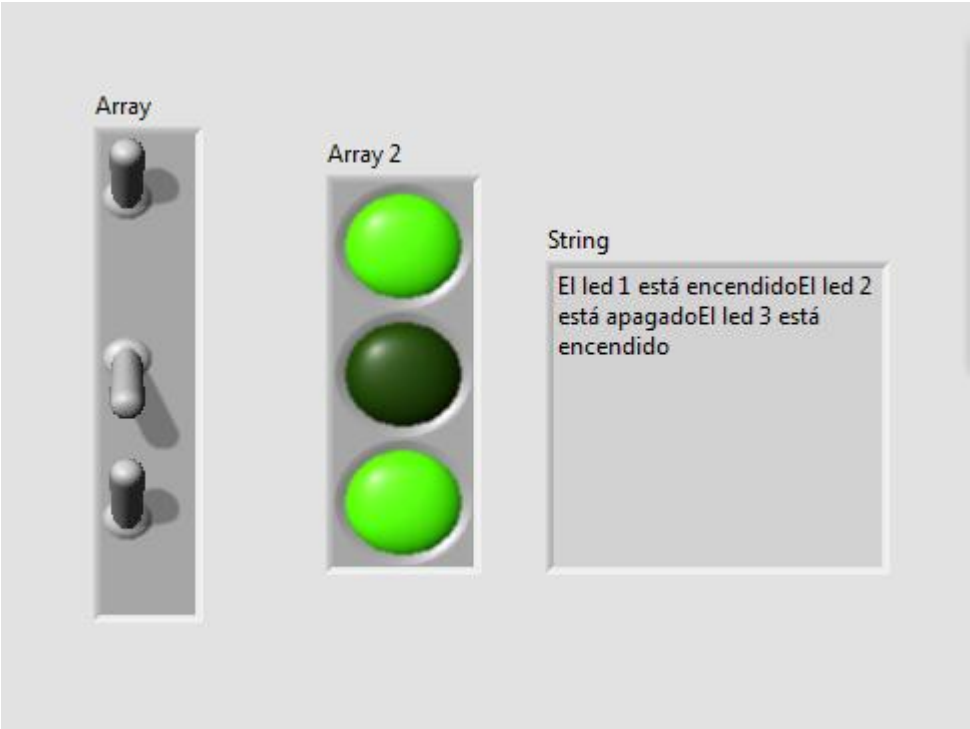


Imagen #30: Programa funcionando con Concatenate string.

Los textos mostrados en el indicador se ven muy amontonados y da un mal aspecto, esto se puede corregir utilizando un Carriage return constant, que da un salto de línea. Aplicándolo al diagrama de bloques, queda de la siguiente manera:

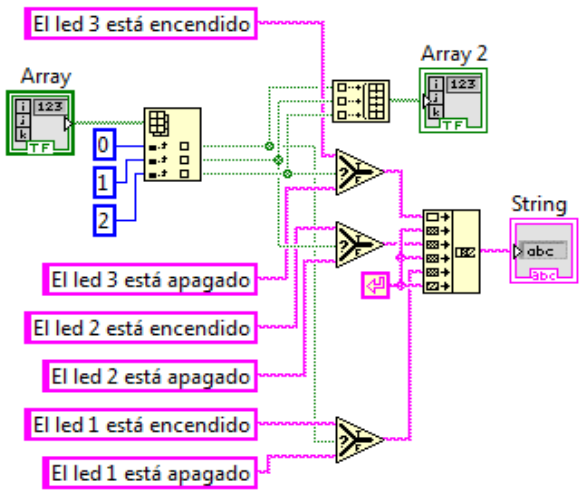


Imagen #31: Uso de Carriage return constant.

Esta constante de retorno se conecta cada vez que finaliza la frase (para este caso), no es necesario usar tres constantes de estas ya que una sola puede funcionar para todo el programa.

El resultado queda así:

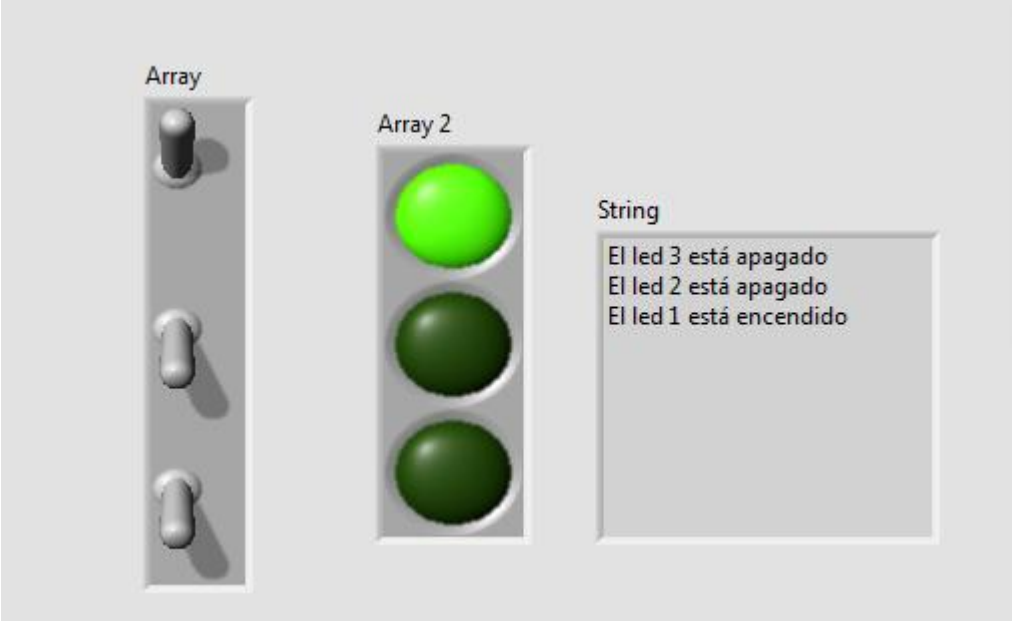


Imagen #32: Programa funcionando con Carriage return constant.

EL CICLO FOR

El ciclo for es una estructura de control utilizada en la gran mayoría de los lenguajes de programación, que permite establecer el número mínimo de iteraciones.

Sus elementos son la variable de control, inicialización de la variable de control, condición de control, incremento, y cuerpo.

La variable de control es la variable con la que el bucle va a trabajar, esta misma es la que se inicializa para establecer un valor predeterminado con el que va a iniciar la iteración.

Su uso se orienta a vectores, permite agregar, modificar o eliminar datos según el índice.

En LabVIEW, este ciclo también se representa con un rectángulo:

Imagen #33: Representación gráfica del ciclo for.

La terminal N representa el número de veces que se va a repetir el ciclo, la terminal i representa la iteración, que va desde 0 a n^{-1}.

Es importante aclarar que cuando se va a utilizar un ciclo, se debe agregar un Timing (Wait), que sirve para que LabVIEW no consuma toda la memoria de la PC y esto haga que se sature. El mínimo valor del Wait puede ser de 5 milisegudos.

Podemos tomar el siguiente programa como ejemplo del uso del ciclo for:

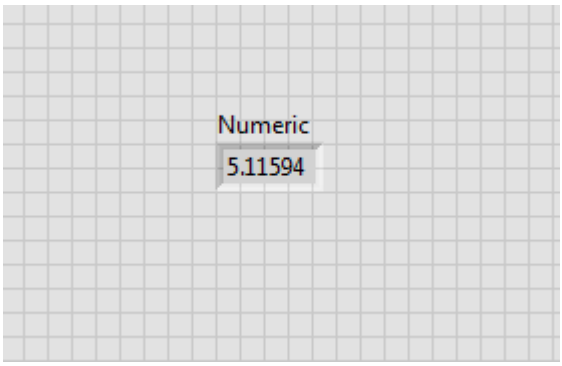


Imagen #34: Ejemplo del uso del ciclo for.

En el panel frontal solo tenemos un indicador numérico.

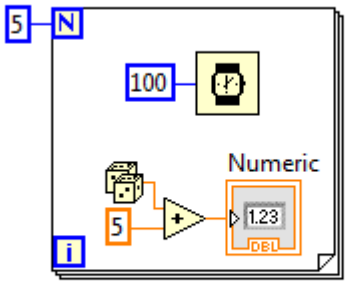


Imagen #35: Diagrama de bloques del ejemplo del ciclo for.

Se puede observar que el número de repeticiones del ciclo for será de 5, y la operación será la suma de un número aleatorio y cinco. Esta operación se hará cinco veces y esos cinco resultados se imprimirán en el indicador numérico.

Aquí el Timing cumple una función interesante, ya que si le ponemos un valor bajo como 5 o 10, no se aprecia el cambio en el indicador numérico y parecerá que sólo se hizo una operación, en cambio, si se le pone 100 o más, ya se puede ver cómo aparecen los 5 números, lógicamente, en el indicador sólo quedará el último resultado.

Si queremos que el número de repeticiones del ciclo quede al azar, entonces sólo ponemos el Random conectado a la terminal N. Con algunas operaciones básicas, se pueden establecer límites para el número aleatorio.

IMPORTANTE: Al utilizar ciclos, ya sea for o while, no se debe utilizar el botón correr continuamente, aquí sólo se usar el botón correr, para respetar las acciones del ciclo y evitar que éste se cicle.
Asimismo, el botón abortar sólo se puede usar cuando el programa esté ciclado.

FOR CONDICIONAL

Se dice que un ciclo for es condicional cuando éste tiene una terminal de paro, se trata de una terminal a donde se puede conectar un botón de stop o una condición para que el programa se detenga en determinado momento sin importar que no se hayan ejecutado el número de repeticiones establecidas.

Esta terminal de paro se puede agregar dando clic derecho sobre el ciclo for y posteriormente clic en conditional terminal.

Imagen #36: Representación del ciclo for condicional.

Nótese que en la terminal N aparece un circulito rojo, éste representa que el ciclo for tiene una terminal de paro.

SHIFT REGISTER

Los shift registers son elementos que guardan valores y mediante operaciones son modificados, éstos son muy útiles para muchos programas, en especial aquellos que utilicen contadores.

Estos elementos se usan en ciclos, para agregar uno, se le da clic derecho al ciclo y luego se da clic en add shift register.

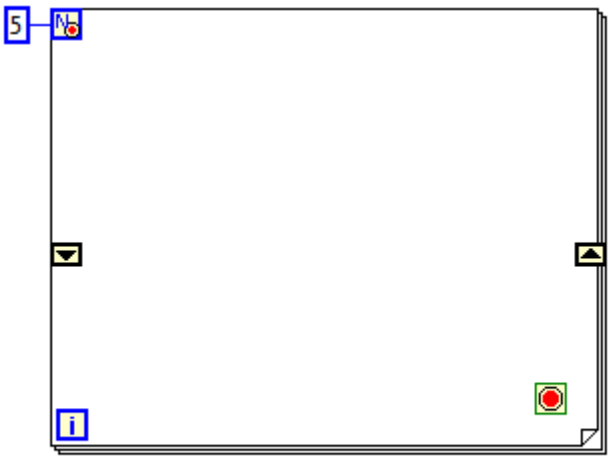


Imagen #37: Representación gráfica del shift register en ciclo for.

La terminal de la izquierda es el valor inicial que se guardará en el shift register, mientras que la terminal de la derecha, es donde se guardará el nuevo valor.

Un ejemplo del uso de shift register con array es el siguiente:

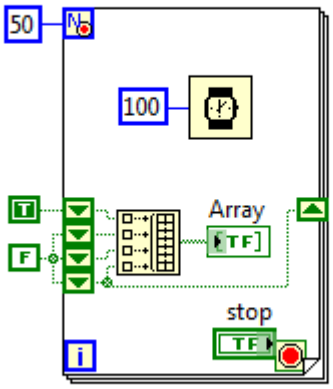


Imagen #37: ejemplo del uso de shift register y array.

Aquí, se tiene un array de 4 leds, se usa un shift register que está aumentado a 4 elementos que representan a cada led, esto se hace poniendo el cursor del mouse debajo del shift register y arrastrándolo hacia abajo.

Se inicializa con una constante true, esto indica que el primer led estará encendido al correr el programa. En un lapso de 100 milisegundos pasa a false y el siguiente a true, así hasta llegar al último, y esto se repite 50 veces o hasta que el usuario presione el botón stop.

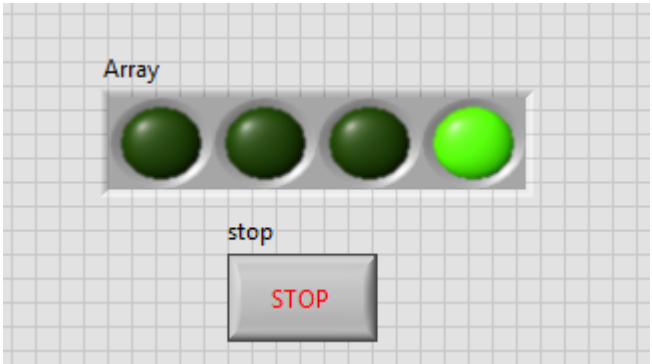


Imagen #38: Panel frontal del programa de ejemplo de shift register y array.

EL CICLO WHILE

A diferencia del For, el ciclo While ejecuta determinada acción hasta que el programa termine su función o hasta que el usuario decida detener el programa. Si el programa no tiene un fin determinado, el programa se ejecutará infinitamente o hasta que el usuario lo detenga.

El ciclo While no funciona si no se conecta un botón de stop o una condición de paro a la terminal de stop, si no ocurre esto, será imposible ejecutar el programa.

Imagen #39: Representación gráfica del ciclo while.

Aquí también es posible utilizar shift register.

Podemos ver un ejemplo muy sencillo de un programa para encontrar la hipotenusa de un triángulo usando el Teorema de Pitágoras, con el ciclo while.

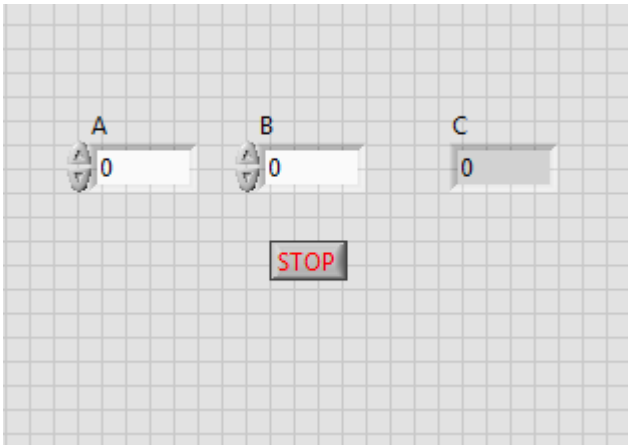


Imagen #40: Ejemplo del uso del ciclo while.

En el panel frontal se tienen dos controles numéricos y un indicador numérico, cada uno representa un lado del triángulo.

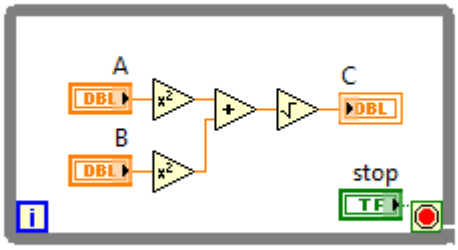


Imagen #41: Diagrama de bloques del ejemplo con while.

En el diagrama de bloques, se tiene un ciclo while, que contiene toda la operación. Según el Teorema de Pitágoras, se sabe que el cuadrado de la hipotenusa es igual a la suma de los catetos.

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$

Entonces una forma simple de resolverlo, es usando la función square, que eleva al cuadrado cualquier número que se le conecte, entonces, se usan dos, uno para cada lado del triángulo. Éstos se suman y se le saca raíz cuadrada al resultado, usando la función square root. Ambas funciones se encuentran en el menú Numeric.

El resultado se imprime en el indicador numérico.

Nótese que el ciclo while tiene un botón de stop conectado a la terminal de paro, sin esto, el programa no se ejecuta.

Hasta aquí todo va bien, pero hay un pequeño error, que en este programa puede no causar problemas, pero al trabajar con programas grandes, puede ocasionar muchos problemas. Es necesario colocar un Wait.

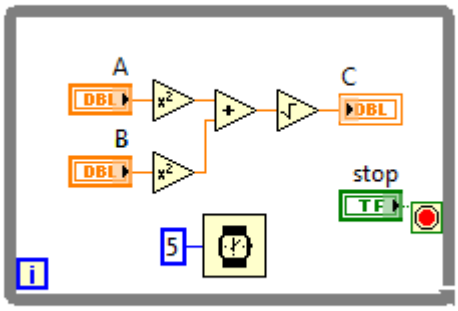


Imagen #42: Uso del ciclo while con Timing.

Un ejemplo para el uso de shift register con un ciclo while, podría ser un contador, para llenar un tanque.

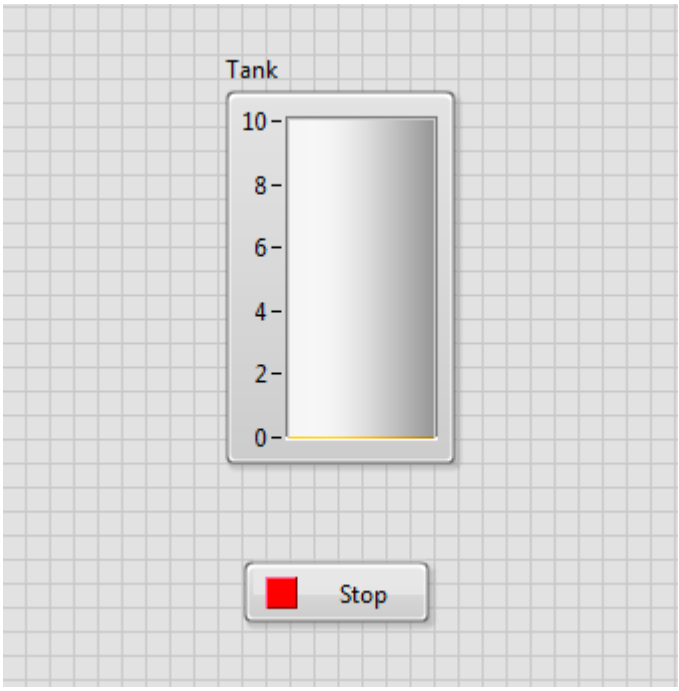


Imagen #43: Panel frontal para el programa con while y shift register.

Aquí, el propósito del programa podría ser que el tanque se vaya llenando, como si se tratara de una máquina automática de llenado de algún líquido, como un despachador de café o algo así.

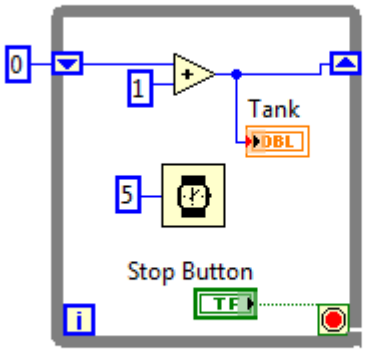


Imagen #44: Diagrama de bloques del ejemplo de un contador.

El shift register se inicializa en cero y se le suma uno, esto hace que en cada repetición se sume uno al valor final. Esto hace que el tanque se vaya llenando lentamente, dando un efecto como de máquina automatizada.

Aquí, el timing juega un papel muy importante, ya que si se deja en el valor que muestra la imagen 44, el llenado va a ser muy rápido y no se va a poder apreciar, entonces se tiene que cambiar el 5 por un 200 o más para que se pueda ver mejor cómo se va llenando el tanque.

También se puede cambiar el uno de la suma por otro número menor o mayor, para dar más precisión a la cantidad de líquido que se vierte en el tanque, si se utilizan decimales, es necesario cambiar la representación para poder usarlo, ya que en el ejemplo se están usando enteros.

Otra cosa muy útil sería que el programa se detuviera cuando el tanque llegue a 10. Para esto, usamos una función de comparación Equal (igual), conectada a una compuerta OR, esto para que el programa se detenga si el tanque llega a 10 o si el usuario presiona el botón de stop.

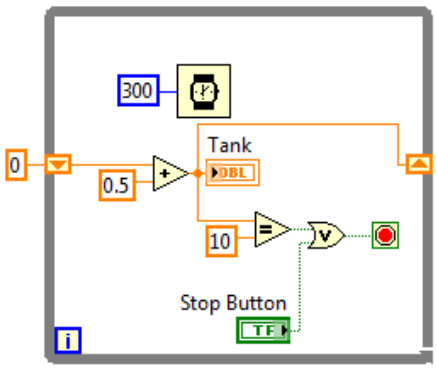


Imagen #45: Diagrama de bloques del ejemplo terminado.

Se puede ver un video del funcionamiento de este programa aquí:
http://www.youtube.com/watch?v=4-ONceG9hWQ

VARIABLES LOCALES

En lenguajes de programación como C se usan variables para todo, por ejemplo, se declara una variable llamada edad, esta variable se puede usar muchas veces en todo el código del programa, respetando su tipo de dato.

En LabVIEW se puede hacer lo mismo, se pueden crear variables locales de elementos para poder usarlos varias veces, aunque no es recomendable abusar de las variables locales, algunas veces resulta muy útil. De hecho, se deben usar solamente cuando es necesario.

Para crear una variable local, solo se da clic derecho sobre el elemento, luego en create>Local variable.

Imagen #46: Control numérico slide y su variable local.

La variable local parece una constante, si se le da un clic, se puede cambiar por otro elemento que se encuentre en el diagrama de bloques.

USO DE CLUSTER

En LabVIEW un cluster es una colección de elementos de diferentes tipos. Es parecido al array, la diferencia es que en el array sólo se pueden introducir un solo tipo de elementos, es decir; un led o un botón. En cambio, en el cluster se puede introducir cualquier elemento, ya sean leds, botones, medidores, controles numéricos y de texto, etc.

El uso del cluster puede reducir el tamaño del código y dependiendo del programa que se esté desarrollando, su uso puede ser imprescindible.

Podemos ver un panel frontal con un cluster que alberga algunos elementos:

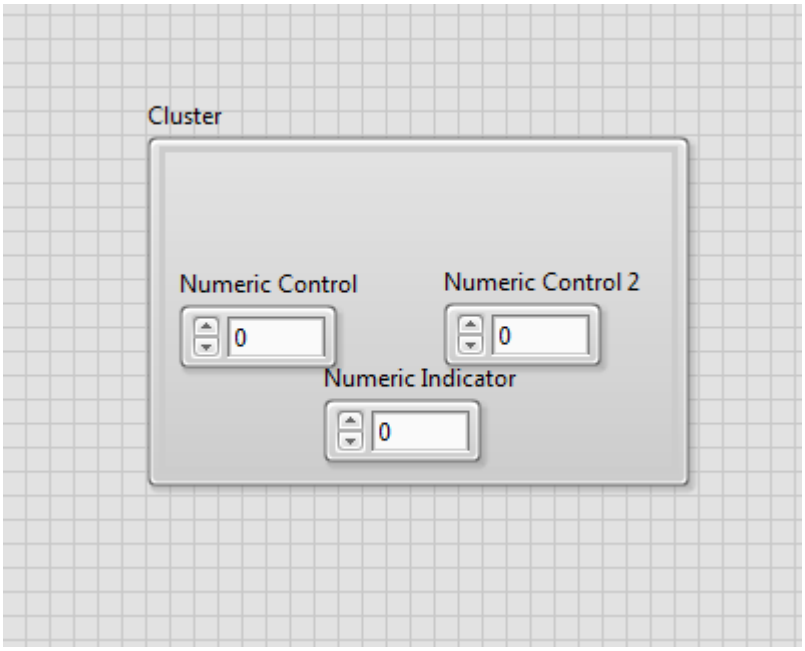


Imagen #47: Cluster con un led, un push button y un indicador de texto.

Como se puede ver, se tienen tres elementos diferentes. El tamaño del cluster, a diferencia del array, se puede modificar al tamaño que más le agrade al programador.

En el diagrama de bloques, lo anterior aparece con un solo elemento, al igual que el array. Para trabajar con cluster, se usan las funciones del menú cluster, class & variant.

Para sacar los elementos del cluster, podemos usar Unbundle y Unbundle by name. La diferencia entre estos es que el unbundle saca los elementos de forma individual, en orden ascendente. Y el unbundle by name saca los elementos mostrando sus nombres, en algunas ocasiones es más fácil utilizar esta función para saber con qué elementos se está trabajando.

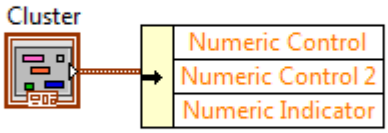


Imagen #48: Uso de unbundle by name.

Al conectar el cluster al unbundle by name, este sólo muestra un elemento, para visualizar los demás se tiene que colocar el cursor en la parte de debajo de la función, dar clic y arrastrar hacia abajo sin soltar para que vayan mostrándose los siguientes elementos.

Se puede observar el siguiente ejemplo donde se usa un cluster y variables locales:

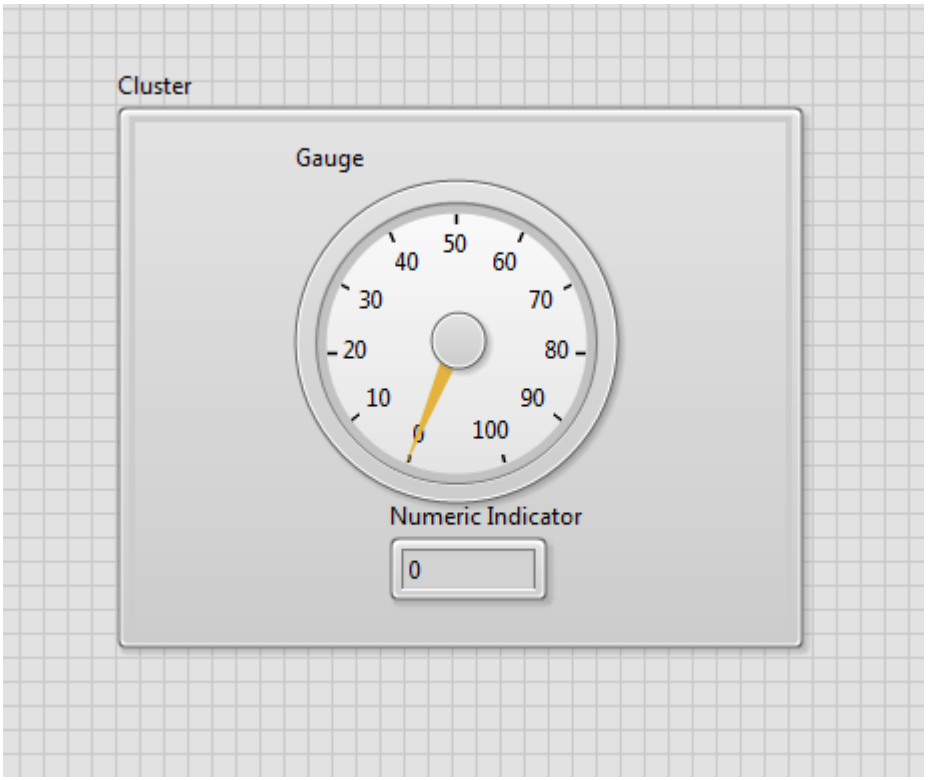


Imagen #49: Ejemplo del uso de cluster.

En el panel frontal se tiene un cluster con un medidor y un indicador numérico. El objetivo es que el usuario mueva la aguja del medidor y lo que marque se pueda observar en el indicador numérico.

El primer paso sería sacar los elementos del clúster, utilizando un unbundle by name, y se selecciona solo el medidor que es el que se va a utilizar como control. Luego utilizamos bundle by name para conectar el medidor al indicador numérico, aquí prácticamente estaríamos creando un cluster. Para finalizar, creamos una variable local del cluster y conectamos el bundle by name a éste.

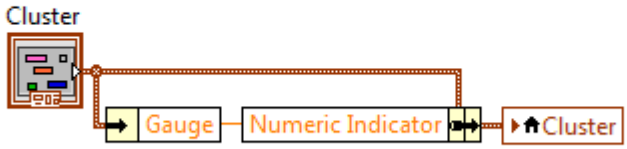


Imagen #50: Uso de cluster con variable local.

De esta manera, al correr el programa y mover la aguja del medidor, el indicador numérico nos mostrará el número real que la aguja está marcando.

APLICANDO LOS CONOCIMIENTOS

Máquina despachadora de refrescos.

Aplicando todo lo que se ha visto hasta ahorita, podemos crear un programa que simule una máquina expendedora de algún líquido, en este caso, refrescos.

El funcionamiento es simple, el programa muestra tres botones, éstos representan sabores, cola, manzana y limón.
El programa muestra un mensaje de bienvenida mientras no se presione algún botón.

Se muestra un control numérico que representa la ranura donde se introduce el dinero, y un indicador que imprime el cambio.

También hay un indicador numérico tipo tanque, que se irá llenando al presionar un sabor, siempre y cuando se haya introducido una cantidad de dinero suficiente.

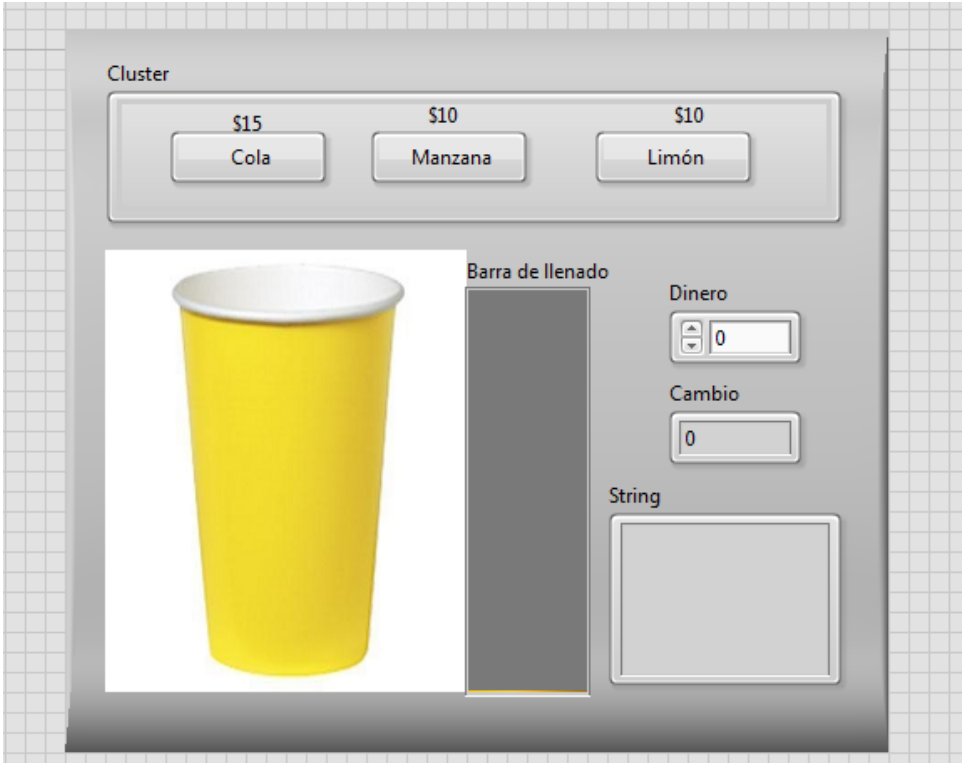


Imagen #51: Panel frontal del surtidor de refrescos.

Los botones que indican los sabores están dentro de un cluster, lo demás son controles e indicadores.

Entonces, al correr el programa, se mostrará un mensaje de bienvenida y el usuario podrá ingresar dinero y presionar el botón que elija.

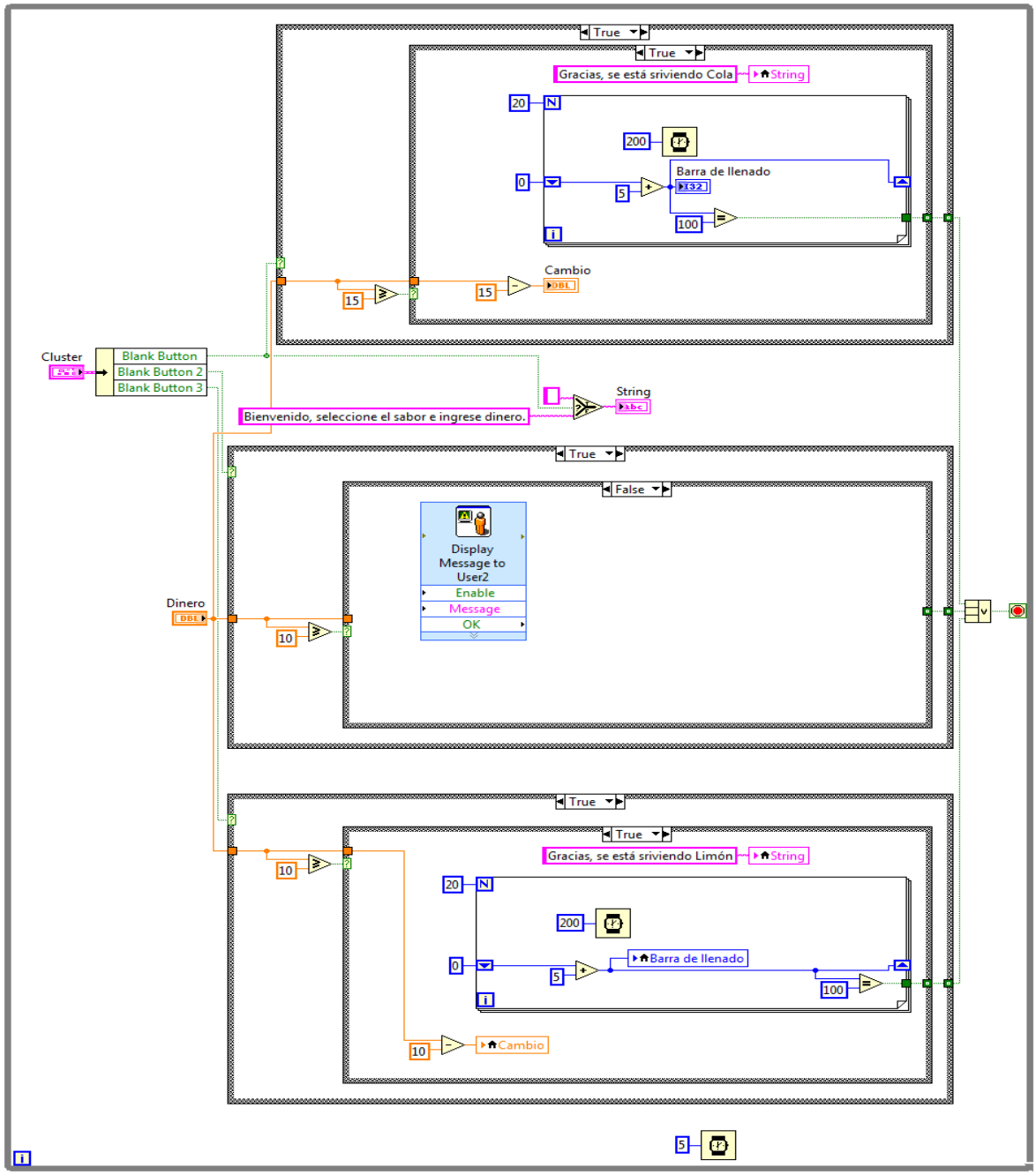


Imagen #52: Diagrama de bloques de la máquina de refrescos.

Como se puede ver en la imagen 52, se usa un ciclo while que anida 3 estructuras case, que a su vez, anidan otro case cada una, y éste, un ciclo for.

Analizaremos cada parte del código.

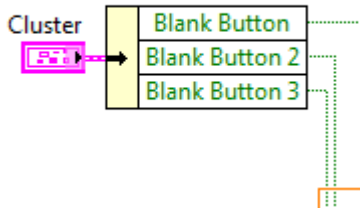


Imagen #53: Se sacaron los elementos del cluster con Unbundle by name.

Dentro del while, se utilizó unbundle by name para poder usar los botones del cluster individualmente. Estos tres botones se conectaron a una estructura case cada uno.

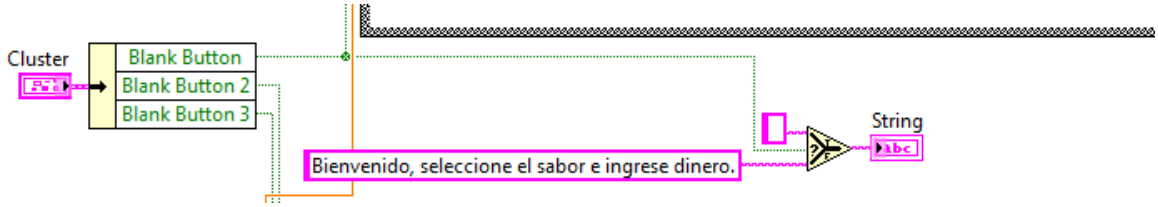


Imagen #54: Mensaje de bienvenida del programa.

El mensaje de bienvenida no debe estar dentro de ningún case, ya que este se debe mostrar al momento de ejecutar el programa. Se usa un Select conectado a cualquiera de los tres botones, en este caso, el primero, si el botón se activa, no se muestra el mensaje, si no se activa, se muestra el mensaje de bienvenida. Esto es algo lógico, ya que se sabe que no se puede presionar ningún botón antes de correr el programa, entonces, obviamente se mostrará el mensaje antes que nada.

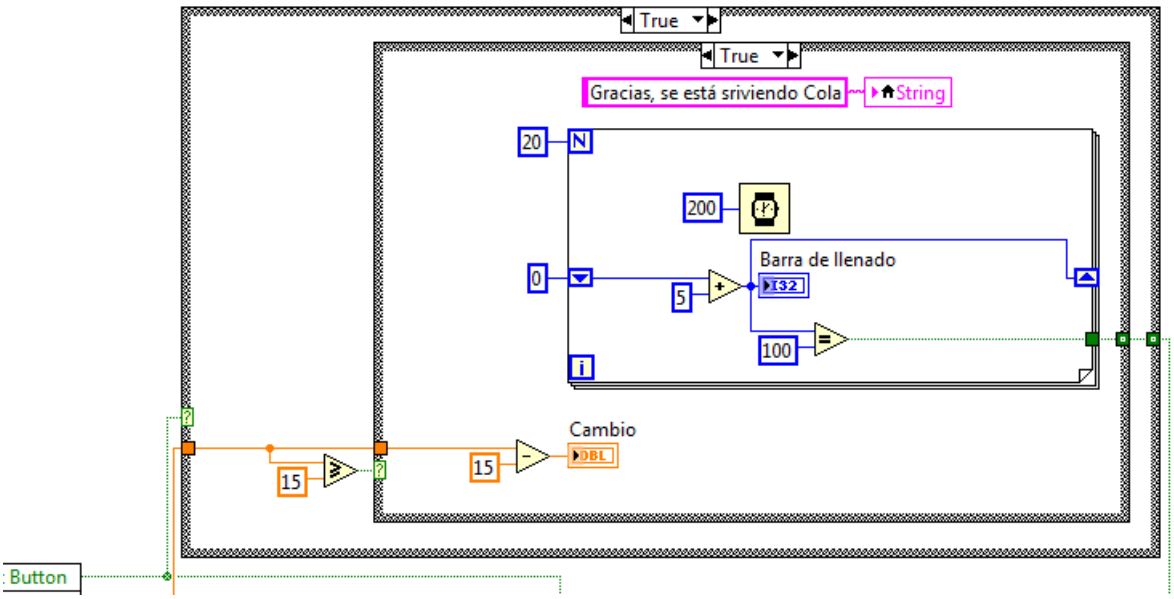


Imagen #55: Estructura Case del primer botón.

Todos los Case de los tres botones tienen lo mismo, solo cambia el costo de la bebida. En el primer case, estando en true, se evalúa si el dinero ingresado es igual o mayor que 15 (costo de la bebida). Si esto es verdadero, en un case anidado se hace la operación para mostrar el cambio.

También se muestra un mensaje que dice que ya se está preparando su bebida y se usa una variable local del indicador de texto.

Con un ciclo for se hace la simulación del llenado del vaso. Como se vio anteriormente, se usa un shift register como contador para ir llenando el vaso, empezando de cero y llenándose de 5 en 5 cada 200 milisegundos.
Después, si el valor del tanque es igual a 100, se manda la instrucción de detener el programa.

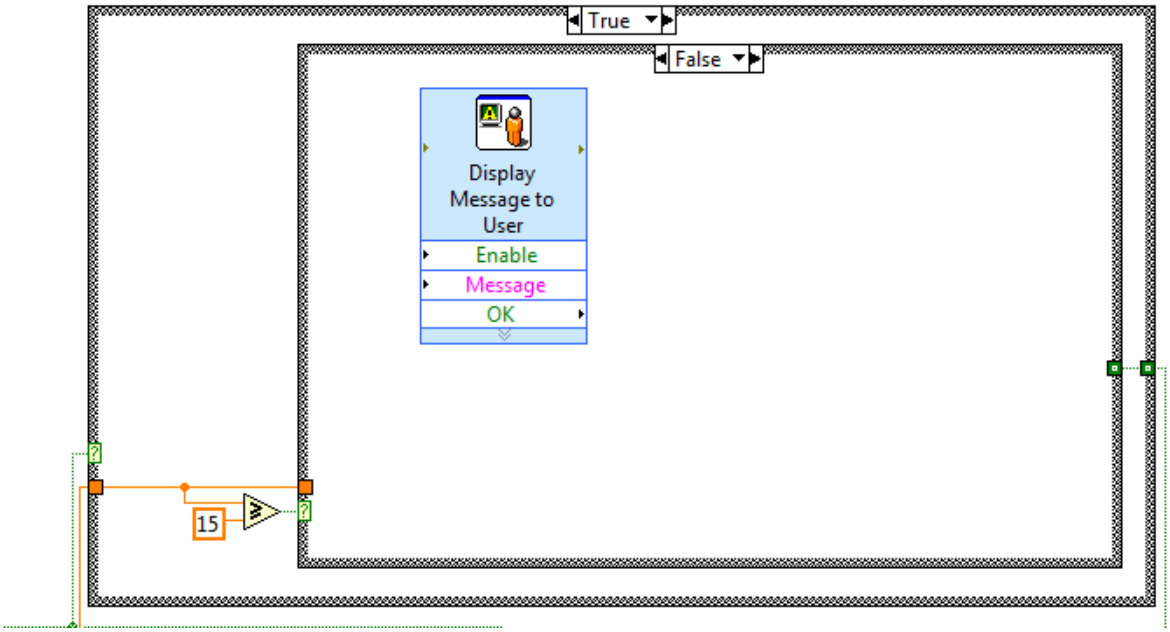


Imagen #56: Valor false del segundo case.

Si el valor del dinero ingresado es menor que 15, entonces el programa muestra un mensaje emergente que le indica al usuario que no le alcanza para esa compra.

Los siguientes dos case indican lo mismo, sólo cambia el valor del dinero y que se usan variable locales de los controles e indicadores.

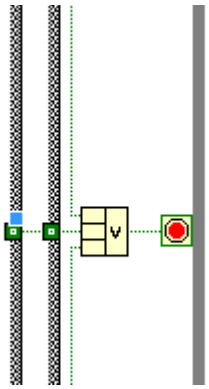


Imagen #57: Uso de compuerta OR para detener el programa.

Se utiliza un compound OR de tres casillas para detener el programa cuando el tanque de cualquiera de los tres botones llegue a 100.
Se puede ver un video del programa funcionando aquí:
http://www.youtube.com/watch?v=3q3HnQQ6Ix0

INSERCIÓN DE IMÁGENES Y DECORACIONES

Pueden surgir algunas dudas con el programa anterior, como por ejemplo, cómo insertar la imagen del vaso en el panel frontal.
Insertar imágenes es muy sencillo. Basta con tener una imagen guardada en el disco duro y arrastrarla desde su ubicación hasta el panel frontal, de esta manera se hace una copia y no importará si se borra la imagen del disco duro, ésta se quedará en el panel frontal, donde se podrá modificar su tamaño.

Para encontrar las decoraciones, nos vamos al menú Modern > Decorations. Ahí podemos encontrar el siguiente menú:

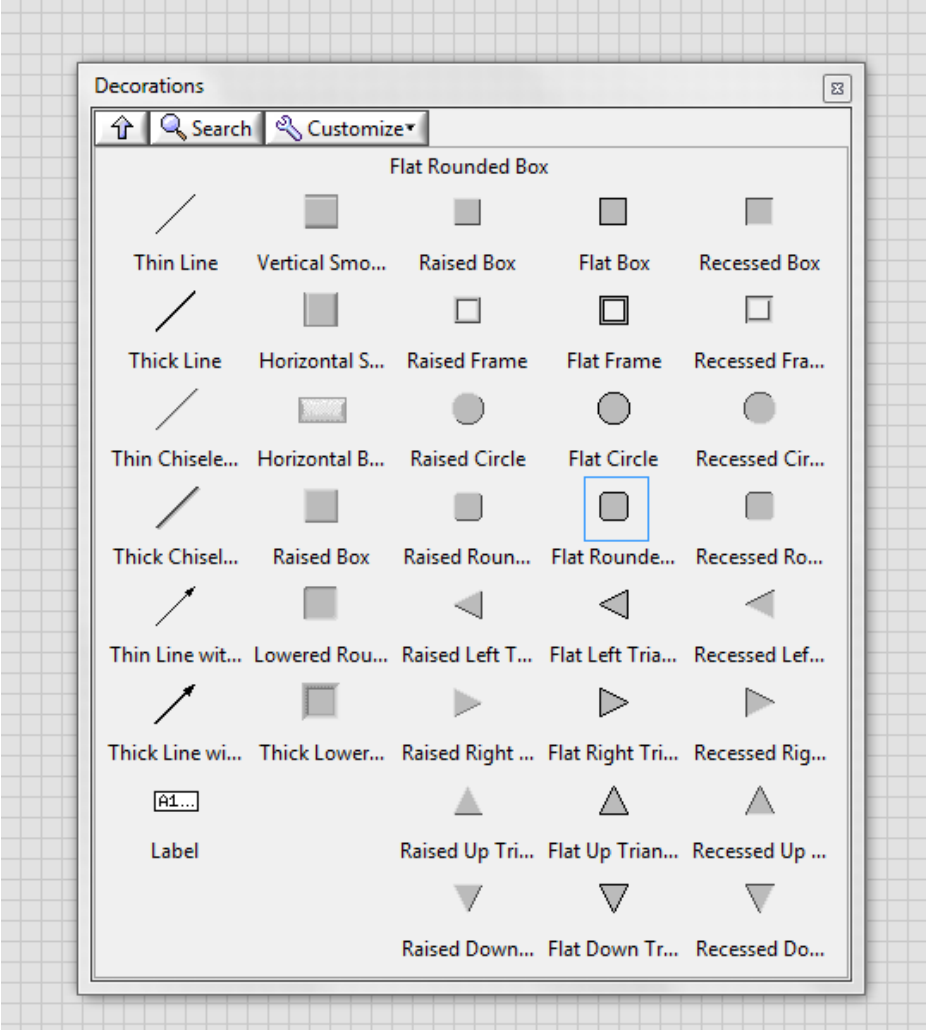


Imagen #58: Menú de decoraciones de LabVIEW.

Ahí se puede elegir cualquier tipo de decoración para que el programa que se esté desarrollando se vea más presentable.

Para que la decoración que hayamos elegido no tape todo, nos vamos al menú y damos clic en el ícono que tiene dos flechas circulares:

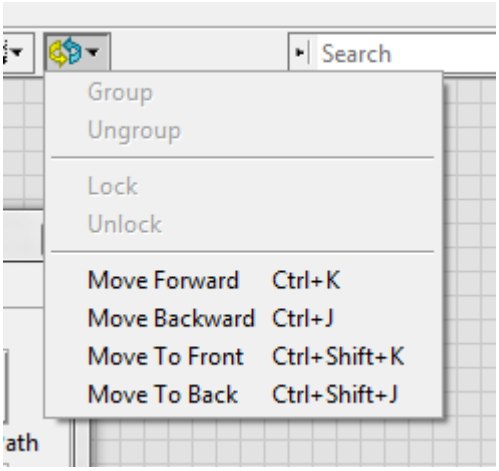


Imagen #59: Menú para mover elementos.

Aquí elegimos Move to back para que la decoración se vaya hasta atrás de todo lo que tengamos en el panel.

Para insertar imágenes al panel frontal es muy sencillo, solo basta con dar clic en Edit o Editar, y en Import picture to clipboard.
De esta manera se puede elegir la imagen que se requiera.

Otra forma es arrastrar la imagen desde su ubicación al panel frontal. Cualquiera de estas dos opciones es buena, y luego se puede modificar el tamaño de la imagen en el panel frontal.

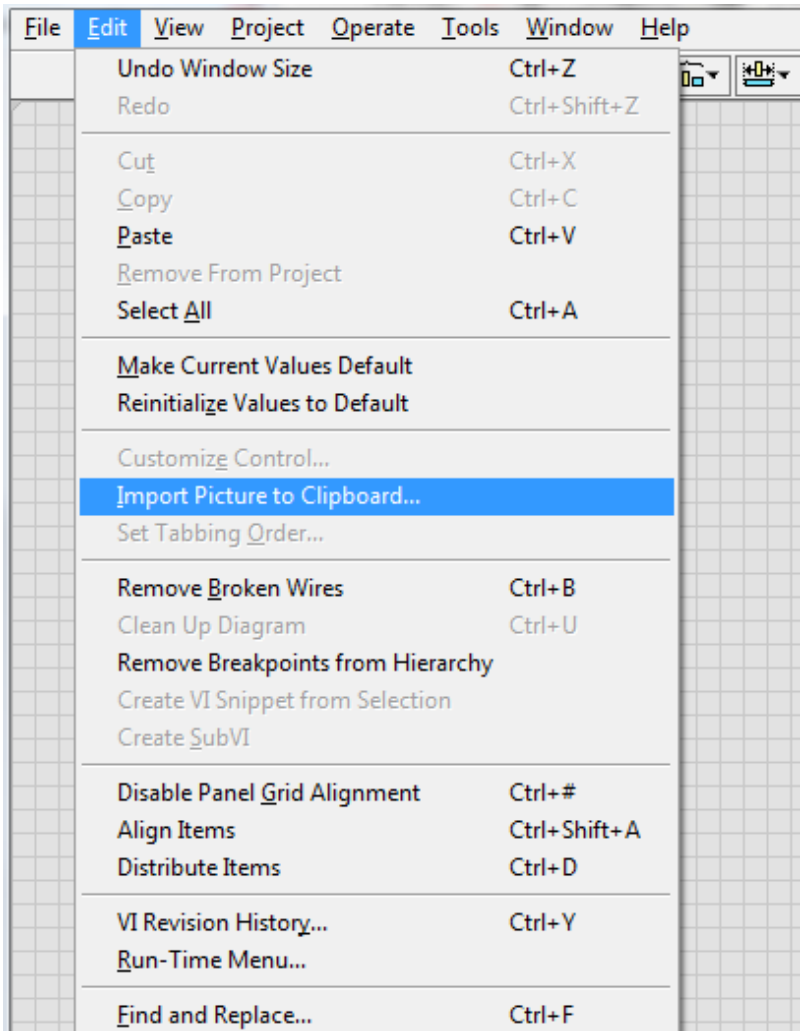


Imagen #60: Insertar imágenes en el panel frontal.

MÁQUINAS DE ESTADO

Una máquina de estados se usa para el desarrollo de algoritmos, siendo una forma muy eficaz de implementación.

Se sigue una serie de pasos para lograr que la máquina funcione, esto se puede observar en el siguiente diagrama:

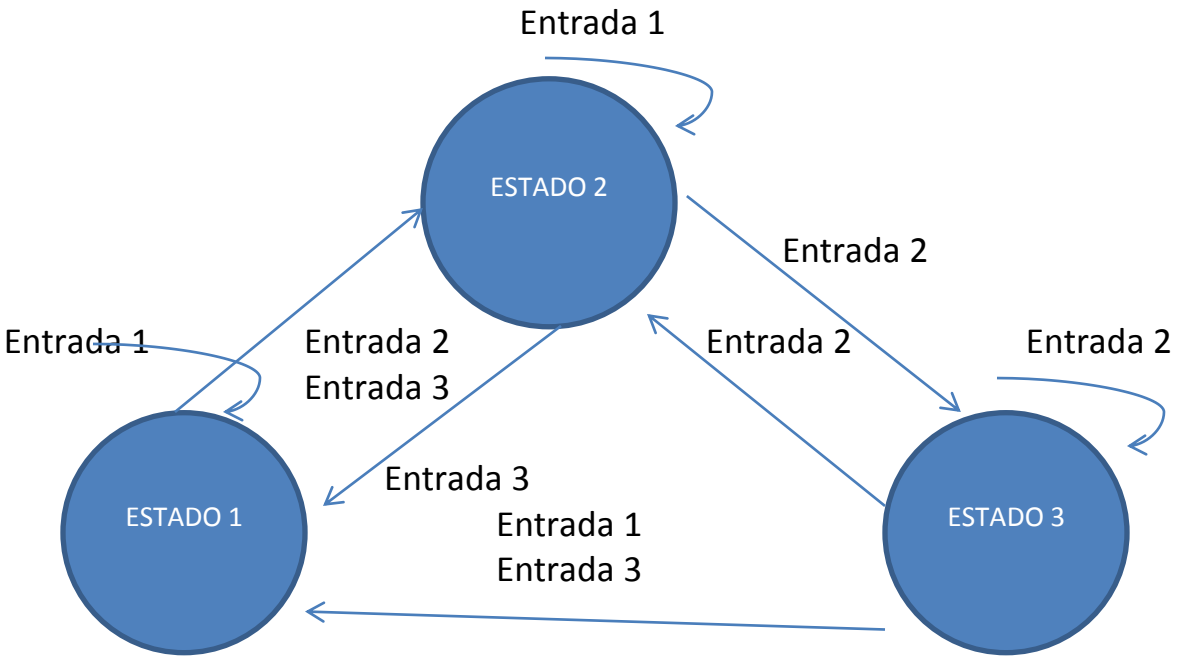


Imagen #61: Diagrama de estados.

En LabVIEW, se utilizan máquinas de estados para resolver problemas en determinadas condiciones.
Los elementos básicos de una máquina de estados en LabVIEW son: Un ciclo while, una estructura case, un enum y los elementos básicos de las estructuras como lo son el timing y el botón de paro del ciclo while.

Para abrir una plantilla de máquina es estados en LabVIEW, sólo se da clic en File>new>For template>Frameworks>Design Patterns>Standard state machine.

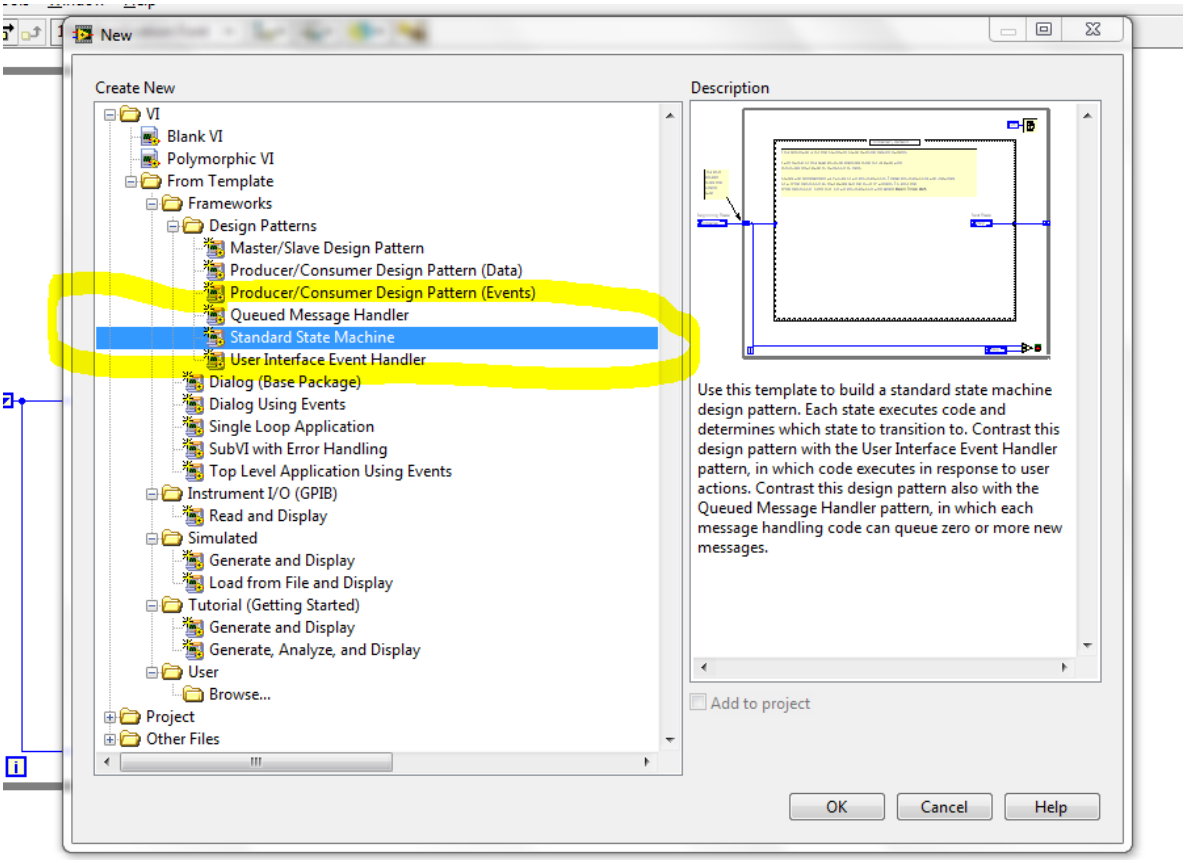


Imagen #62: Proceso de apertura de plantilla de máquina de estados.

Al dar clic en OK, se abre en el diagrama de bloques una plantilla de máquina de estados, en la estructura case se añaden los estados y se controlan con un Enum conectado a un shift register.

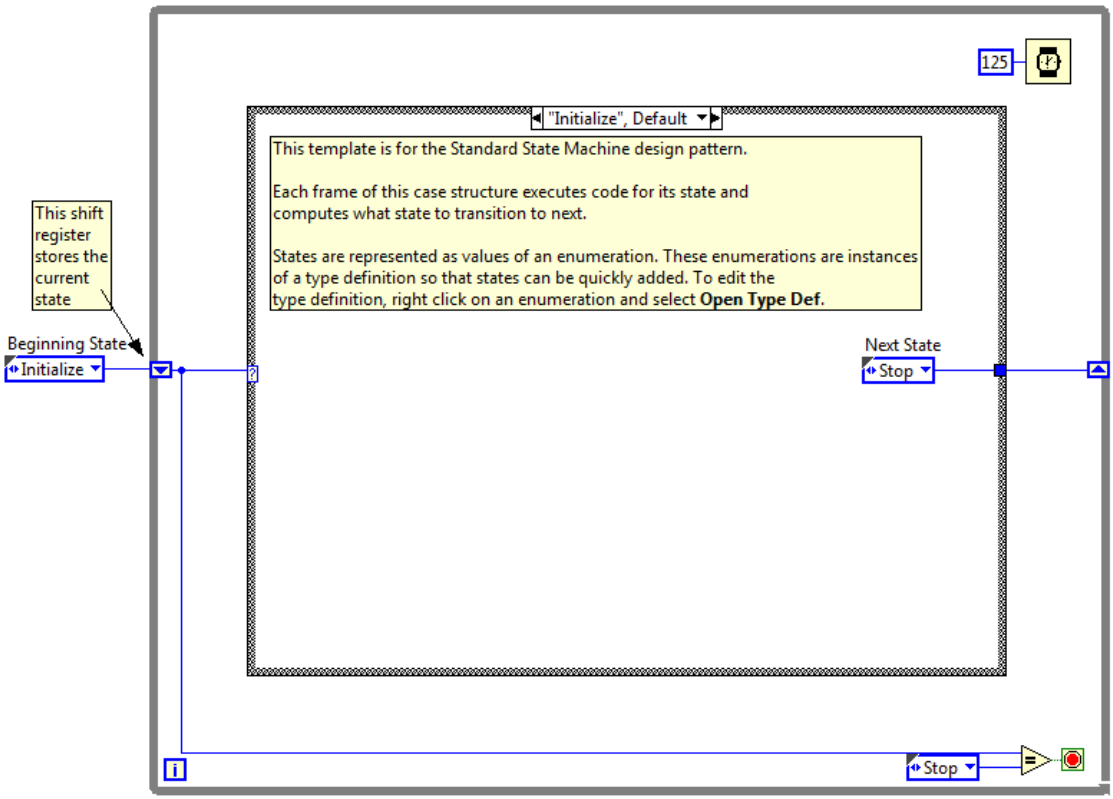


Imagen #63: Plantilla de máquina de estados en LabVIEW.

Ahora veremos un ejemplo del uso de la máquina de estados para crear un programa de llenado de un tanque de Diesel. El panel frontal mostrará un tanque con capacidad de 1000 litros, 3 botones y un indicador numérico.

Para empezar, debemos establecer nuestros estados, salidas, entradas, acciones para poder realizar el diagrama de estados.

Para este programa, tenemos lo siguiente:

Estados:

- Stand by: El programa en estado normal
- Inicio: Se comienza a llenar el tanque
- Detener: Se detiene el proceso de llenado

Entradas:

- Botón de inicio
- Botón de detener

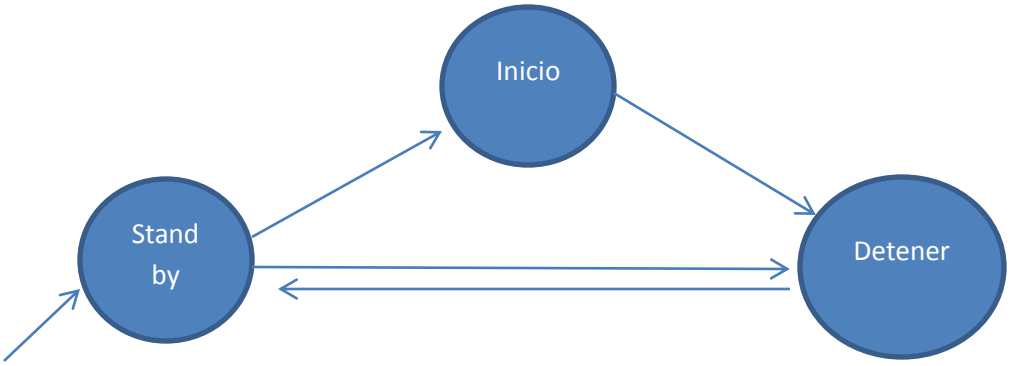
Salidas:

- Llenado del tanque
- Paro del programa

Acciones:

- Si se está Stand by se espera una acción.
- Si se está en inicio se llena el tanque hasta llegar a 1000.
- Si se está en detener, se detiene el proceso de llenado pero no la ejecución del programa.

Con estos datos ya podemos crear el diagrama de estados que nos servirá para poder desarrollar el código del programa en LabVIEW.



Ahora, ya podemos empezar con el código. Colocamos lo básico, un ciclo while, un case y un Enum.

En el panel frontal tendremos un tanque, dos botones que usaremos para Inicio y Detener, un indicador numérico para visualizar los litros y un botón de paro para el ciclo While.

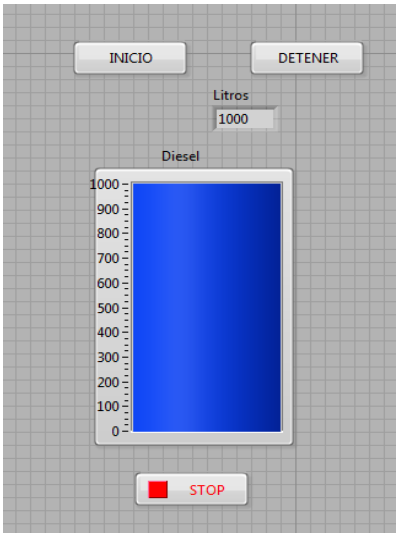


Imagen #64: Panel frontal del ejemplo de máquina de estados.

El diagrama de bloques terminado queda de la siguiente manera:

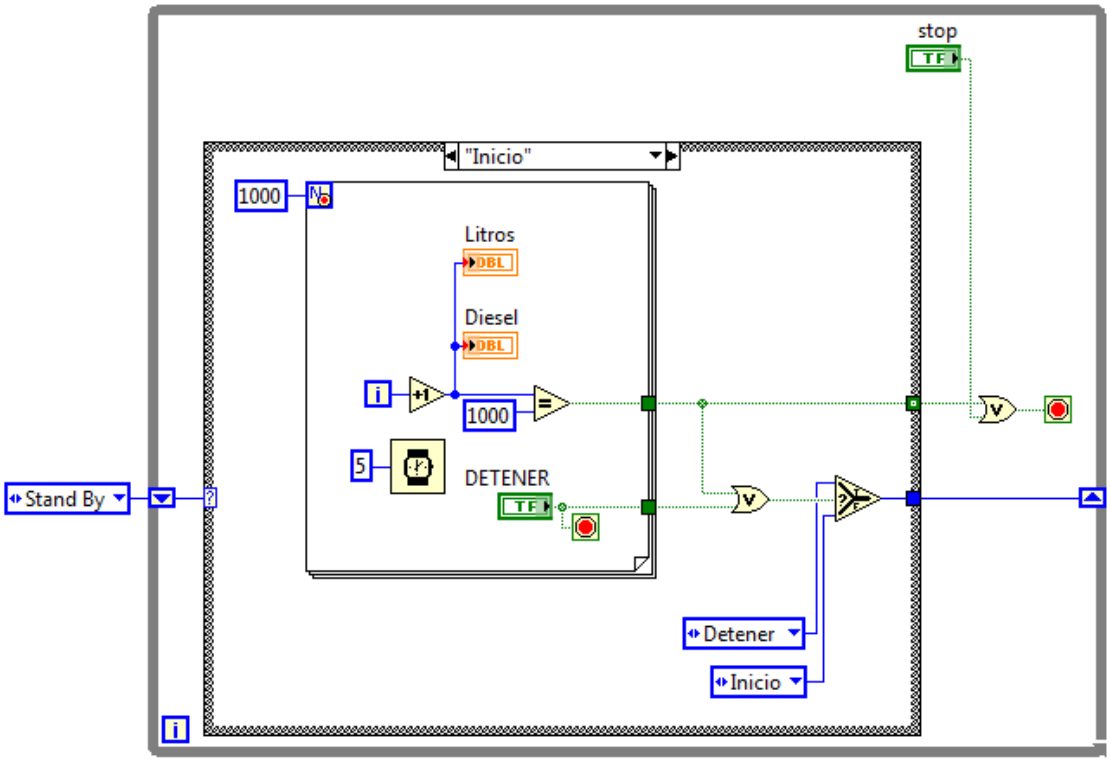


Imagen #65: Diagrama de bloques del ejemplo de máquina de estados.

Como se observa en la imagen, al estar en el caso Inicio, el proceso de llenado funciona con un ciclo for, parecido al de programas anteriores para llenar tanques. Se usan 1000 repeticiones del for, que representan el número de litros. Con el Timing elegimos el tiempo que tarda en llenarse.

Con un incremento los indicadores (tanque e indicador numérico) aumentan de uno en uno cada 5 milisegundos, al ser el valor de los indicadores igual a 1000, se detiene el proceso de llenado.

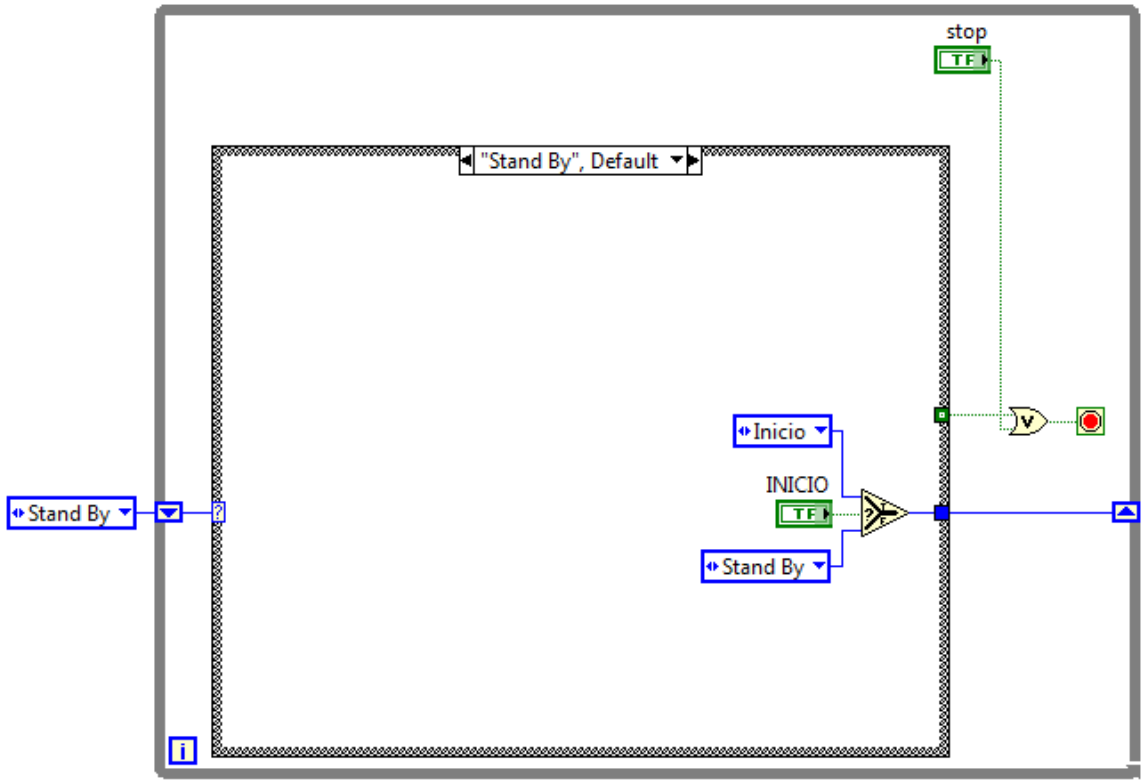


Imagen #66: Caso Stand by.

Al estar en Stand by, el programa solo espera que el usuario presione el botón Inicio para pasar al estado de llenado, de lo contrario se queda en el mismo estado.

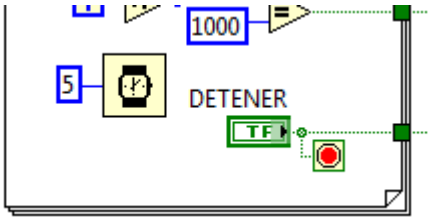


Imagen #67: Funcionamiento del botón detener.

Para que el proceso de llenado del tanque se detenga sin que el programa completo se detenga, hay que ponerle una terminal de paro al ciclo for (for

condicional), de esta manera, al presionar el botón sólo se detendrá el ciclo for y no todo el programa, haciendo más eficaz el uso del mismo.

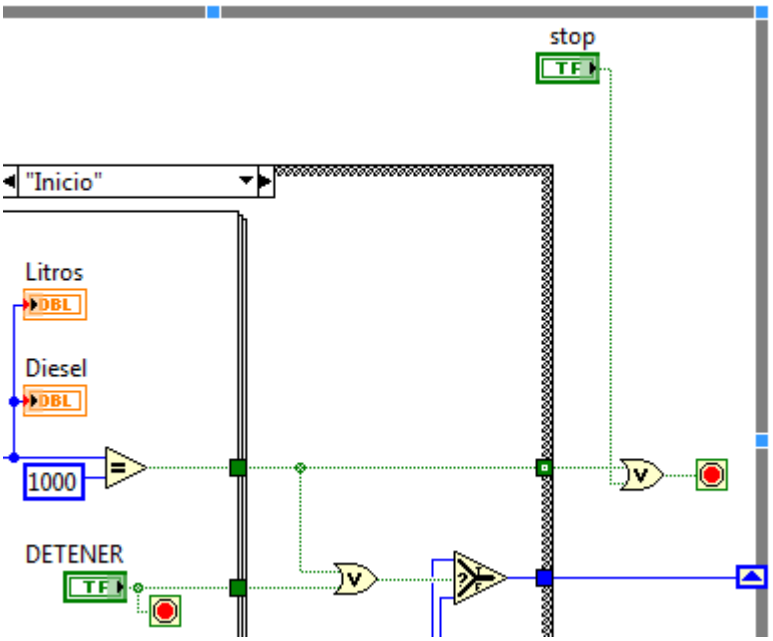


Imagen #68: Paro total del programa.

Si queremos detener por completo el programa, entonces si tenemos que usar la terminal del ciclo while. Nada más que aquí usaremos una compuerta lógica OR para que el programa se detenga si el usuario presiona el botón Stop o si el tanque llega a los 1000 litros.

Con esto se concluye el programa y sólo queda agregar algunos detalles visuales o en el código, dependiendo de las necesidades del programador y del usuario final.

EJERCICIOS FINALES

Hasta aquí ya se ha visto lo más básico para aprender a programar en LabVIEW, por lo que el estudiante ya debe comprender el funcionamiento de cada elemento que ofrece este Software, así como saber resolver problemas relacionados.

Ahora se verán algunos ejemplos para reforzar lo aprendido, en éstos se trabajará con todo lo que se ha visto hasta el momento.

Podemos empezar con un ejercicio aplicado al área del álgebra lineal. Suponiendo que necesitamos resolver una multiplicación de matrices A X B, que mejor forma que hacerlo con algún software como Matlab o Excel, pero al tratarse de ser programadores, podemos crear uno en LabVIEW.

Se puede crear un programa que haga multiplicación de matrices usando arrays y todas las operaciones que esto requiere, pero sería una pérdida de tiempo hacer el programa que realice la operación si LabVIEW ya incluye un menú para trabajar con álgebra lineal. Sólo basta con usar los elementos de este menú.

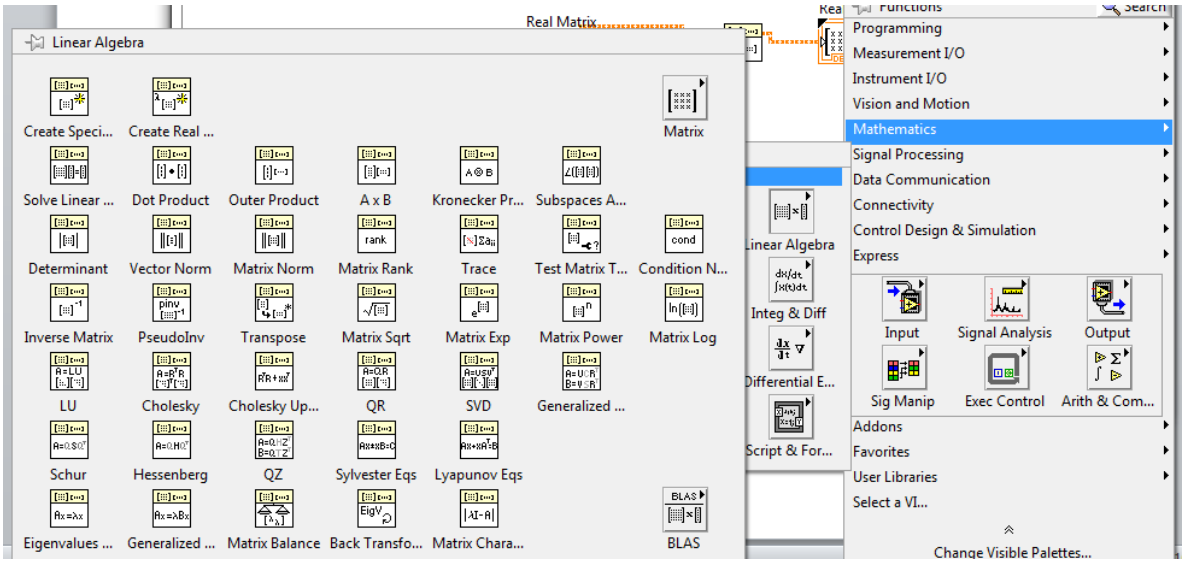


Imagen #69: Menú de elementos para álgebra lineal.

Para acceder a este menú desde el diagrama de bloques seleccionamos donde dice Mathematics>Linear Algebra.

Ahora, en el panel frontal, usaremos tres matrices; A, B y C, siendo A y B las matrices que se van a multiplicar y C el resultado. Con esto ya podemos saber de antemano que A y B serán controles y C indicador.

Podemos encontrar las matrices en el menú donde se encuentran los arrays, ahí usamos Real Matrix para matrices con números reales, ó Complex Matrix para matrices con números complejos.

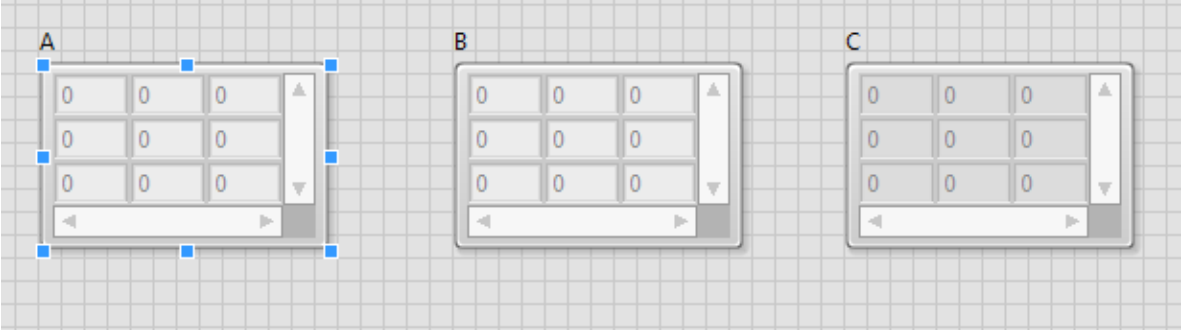


Imagen #70: Matrices para el ejemplo de multiplicación de A X B.

Ahora, en el menú de álgebra lineal, usamos el elemento AXB y conectamos las matrices.

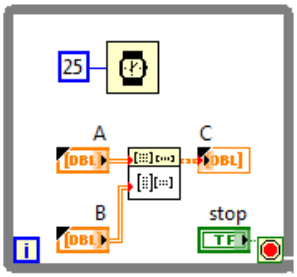


Imagen #71: Diagrama de bloques del programa A X B.

A y B se conectan de lado izquierdo y C del lado derecho. El ciclo while es opcional, pero si se requiere que el programa se mantenga en ejecución es conveniente utilizar un ciclo.

Hay que tener presente el orden de conexión de las matrices, ya que si se coloca A en el lugar de B y B en el de A el resultado será diferente.

En este programa se está trabajando con matrices de 3 X 3, pero hay opción de hacerlas más grandes o más pequeñas.

TAMAÑO DE UN BOTÓN CON NODOS DE PROPIEDAD

En este programa usaremos nodos de propiedad para ajustar el tamaño de un botón. Usaremos el nodo Size con los parámetros Width y Height.

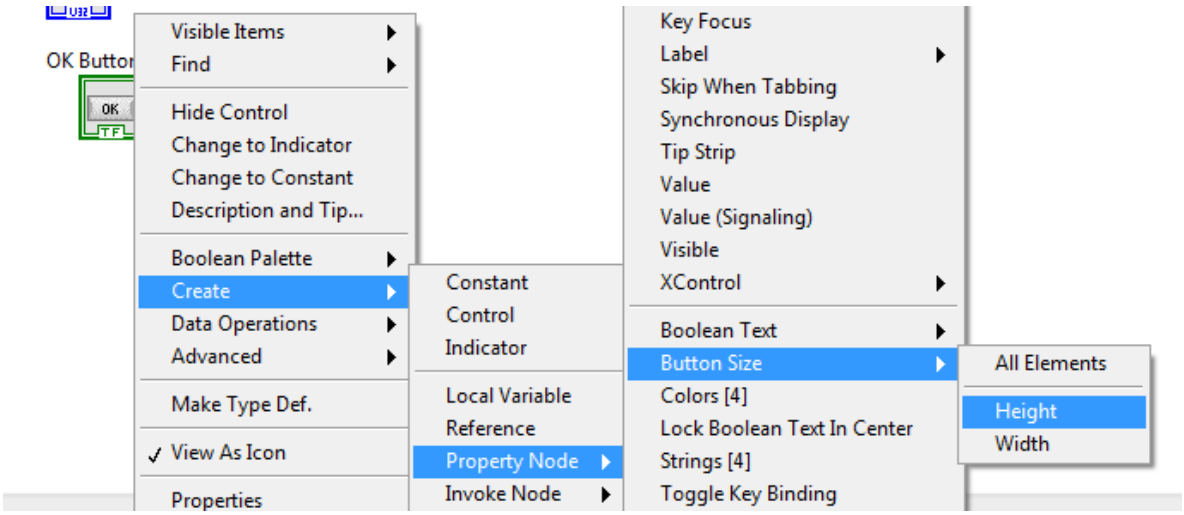


Imagen #72: Ubicación del nodo de propiedad Size.

Con este nodo de propiedad podemos ajustar el ancho y largo del botón. Para esto usaremos dos controles tipo Slide que se conectarán al cluster del nodo de propiedad mediante un bundle by name.

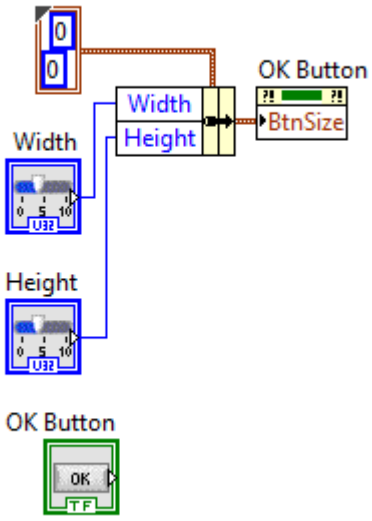


Imagen #73: Código del programa con nodo de propiedad Size.

Imagen #74: Panel frontal del programa.

Para ver el programa funcionando entra a este link: http://www.youtube.com/watch?v=D4oLOfPpWRc

EJERCICIO CON ARRAY DE LEDS

El objetivo de este ejercicio es explotar los conocimientos que se tienen en el uso de arrays, se tiene un array de 11 X 11 de leds en donde el objetivo es que se encienda un led del centro, y vaya haciendo una especie de espiral hasta dejar prendidos todos los leds.

Aparte, el programa tendrá la opción de que el usuario elija hacia qué dirección se encenderá el segundo led: arriba, abajo, izquierda o derecha.

También tendrá un indicador de texto que mostrará un mensaje cuando el programa esté en ejecución o se detenga. Un botón de stop y un botón de salida que hará que se cierre LabVIEW al presionarlo.

El panel frontal terminado queda de la siguiente manera:

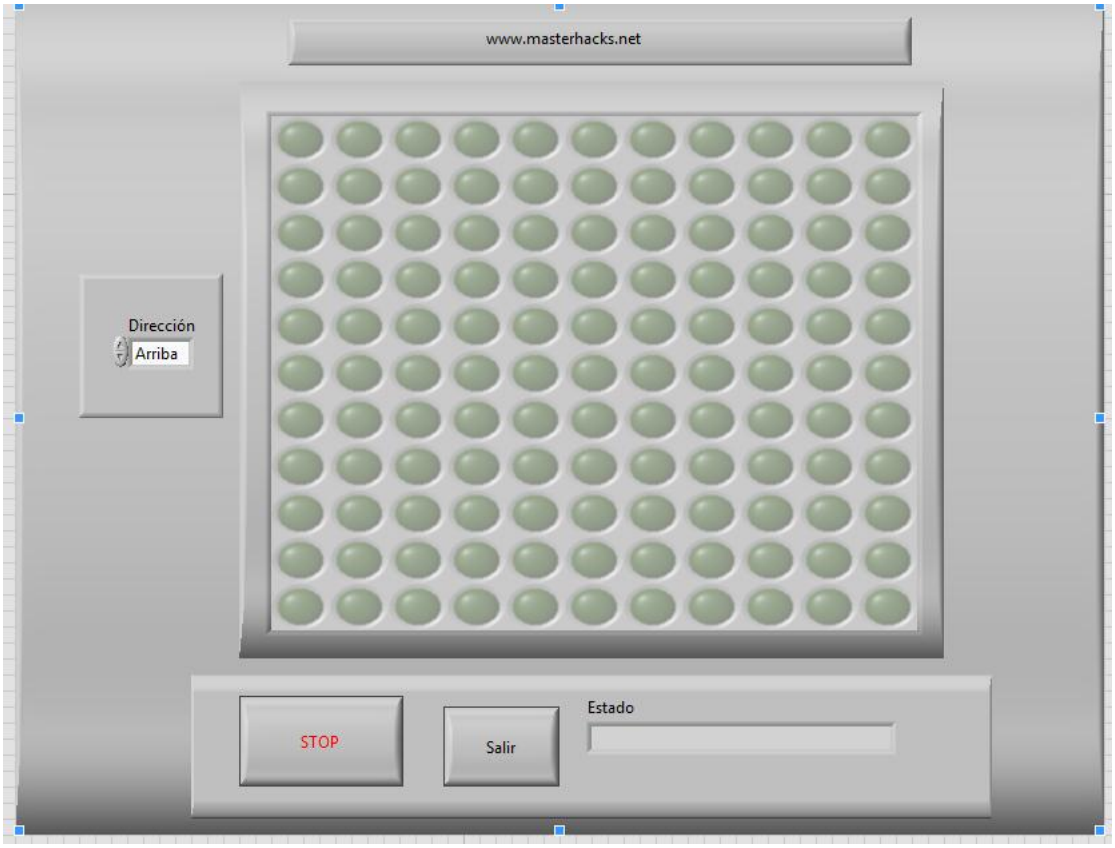


Imagen #75: Panel frontal del programa.

El diagrama de bloques completo es el siguiente:

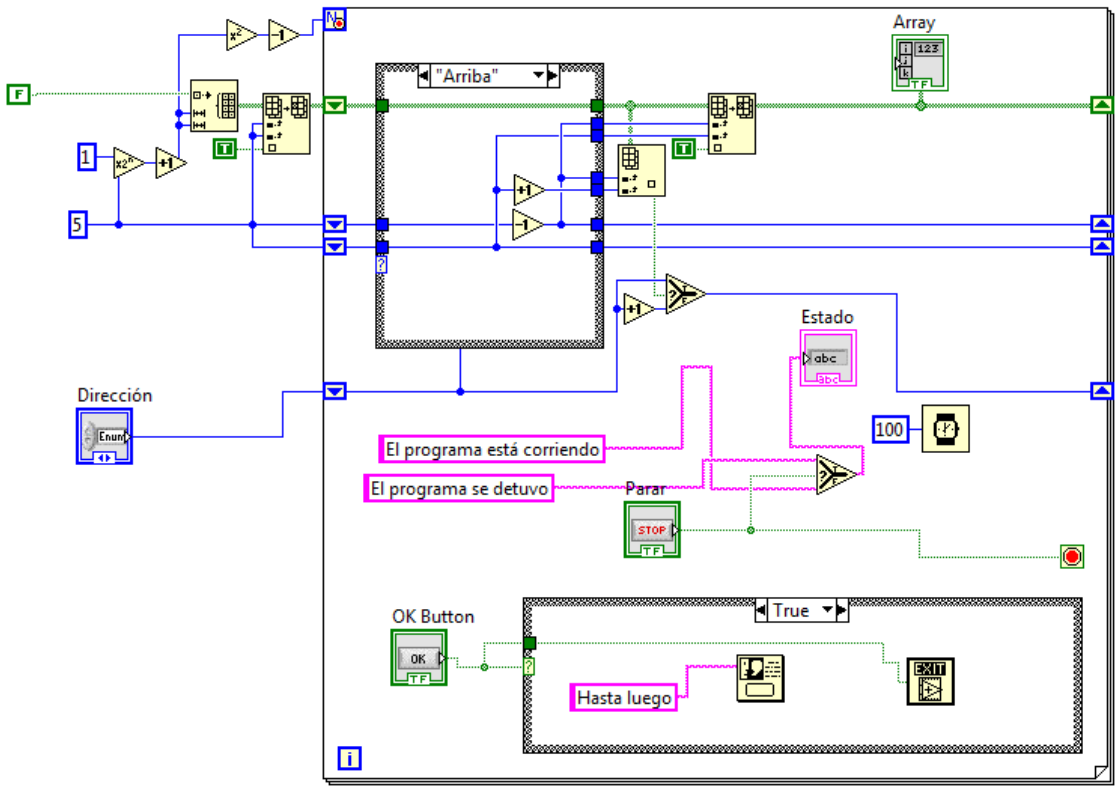


Imagen #76: Diagrama de bloques del programa.

El programa consta de un ciclo For y dos estructuras case, el mayor grado de complejidad está en la parte de la dirección de encendido. Se explicará por partes para una mejor comprensión.

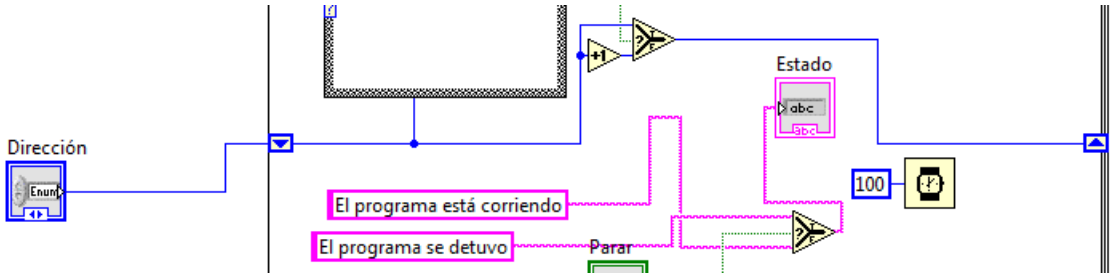


Imagen #77: Uso de shift register con enum.

Se utiliza un Enum con 4 valores; arriba, abajo, izquierda y derecha, éste es el que se encarga de asignar la dirección hacia donde se encenderán los leds. Se conecta a un shift register que a su vez se conecta a una estructura case. Esta estructura tendrá una configuración para cada uno de los cuatro casos en donde prácticamente sólo varía el orden de los incrementos y decrementos.

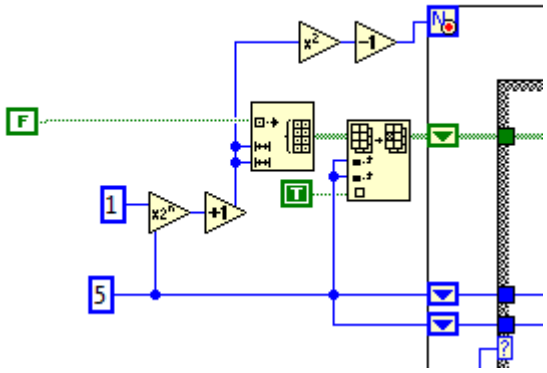


Imagen #78: Configuración del ciclo for y del array de leds.

Se utiliza un initialize array que no servirá para establecer un cómo empezará a trabajar el array. La constante False indica que al ejecutar el programa, los leds estarán apagados, de ahí, las dos terminales de abajo hacen referencia a la dimensión del array, en este caso de 11 X 11, así que se usaron dos constantes, 5 y 1, donde x vale 5 y n vale 1. Usando el elemento que multiplica a X por 2 elevado a la n potencia, tenemos que 5 X 1 = 10^1 lo que nos deja un valor de 10, a esto le sigue un incremento que nos deja un valor de 11, con esto ya tenemos que nuestro array es de 11 X 11.

Todo lo anterior se puede sustituir por un simple 11 en cada terminal, para hacer las cosas más sencillas.

Luego podemos observar que el valor 11 se conecta a un elemento que eleva a la segunda potencia, lo que hace que nuestra X tenga un nuevo valor de 121, luego se conecta a un decremento dando como resultado 120.

Este número 120 representa el número total de leds en el array, y a su vez, la cantidad de repeticiones que el ciclo for ejecutará.

El resultado del Initialize array se conecta a un Replace array subset, que va a reemplazar los leds apagados por leds encendidos con ayuda de los shift register. Nótese que se siguen usando los valores de 11 X 11.

Este último elemento se conecta a un shift register que pasa por la estructura case, luego se conecta a otro Replace array subset que seguirá el orden de encendido de los casos y dejará en valor verdadero los leds que se vayan encendiendo, luego se conecta al array de leds para mostrar el resultado.

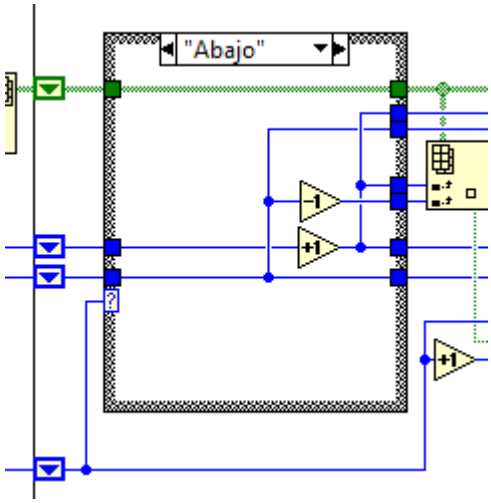


Imagen #79: Estructura case para el orden de encendido.

Para establecer hacia donde se van a encender los leds, sólo se usan dos elementos, incremento y decremento. Por ejemplo, para “Abajo”, al estar encendido el led del centro se enciende el de abajo con un decremento, luego, con un incremento se enciende el de su derecha y así funcionan los demás casos, sólo se juega con los incrementos.

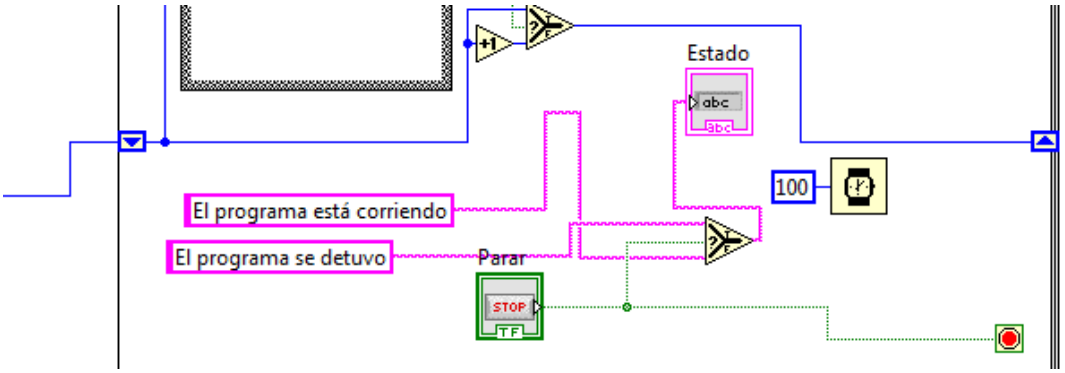


Imagen #80: Textos para el estado del programa.

En la parte de estado, el programa muestra en un indicador de texto si el programa está ejecutándose o si se detuvo. Para esto usamos dos constantes de texto conectadas a un select y éste conectado al botón de stop. Si el botón no se presiona, por default al ejecutar el programa saldrá el mensaje de que se está corriendo ya que los leds empiezan a encenderse. Al presionar el botón, se muestra el mensaje de que el programa se detuvo y luego se detiene el programa.

Cabe mencionar que el Timing que se ve en la figura 80 nos sirve también para la velocidad de encendido de los leds.

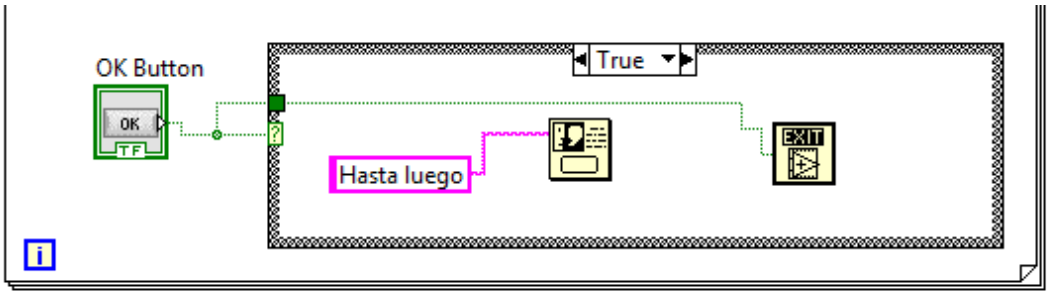


Imagen #81: Mensaje de despedida y salida de LabVIEW.

Para salir de LabVIEW usamos otro botón, éste se conecta a un case, si el botón está en false no se hace nada, si se presiona, cambia a true y el programa mostrará un mensaje emergente con la frase “Hasta luego” y con el elemento Exit, saldrá de LabVIEW.

Para ver el programa funcionando, puedes ver el siguiente video:

http://www.youtube.com/watch?v=jLspyWmOTFM

MÁQUINA TRAGAMONEDAS

El siguiente programa es muy extenso, se utilizan muchos elementos, en especial leds, y varias estructuras case. Se trata de una máquina tragamonedas como las que se ven en las tenditas.

El funcionamiento es simple, se inserta el dinero y se escoge una fruta, cada fruta tiene un valor, este valor es el que el usuario puede ganar si el led se enciende en su casilla. Al insertar el dinero y presionar el botón de la fruta, se enciende un led que va recorriendo todas las casillas, empezando con una velocidad alta que va disminuyendo aleatoriamente hasta detenerse.

Si al terminar, el led se enciende en la fruta seleccionada, se gana el valor indicado, si no, se descuenta un peso del dinero ingresado.

Si se presiona el botón de jugar sin haber ingresado dinero, se muestra un mensaje de alerta advirtiendo que no hay dinero suficiente. Si el usuario gana el premio seleccionado, se muestra otro mensaje de felicitaciones.

Si el usuario quiere sacar su dinero sobrante o ganado, se presiona el botón cobrar y la cantidad va disminuyendo simulando que fueran cayendo las monedas.

El panel frontal es un poco grande, debido a que se utilizan las imágenes de una máquina real, sólo insertando los elementos con los que se trabaja en LabVIEW.

Imagen #82: Panel frontal del programa "Máquina tragamonedas".

En el panel frontal, los elementos que se usarán son los leds que hay en cada fruta, los botones de la parte derecha, también el control numérico y el indicador de texto y el botón de stop.

Debido a que el código es demasiado grande, no se mostrará completo como imagen, pero se puede ver en el video de muestra o descargando el VI desde nuestra página web.

Para el manual, se explicará el código por partes.

Para empezar, utilizamos un ciclo while para mantener corriendo el programa, hasta que se presione el botón de stop. Esto podría considerarse como la simulación de encendido y apagado de la máquina.

Dentro del while, se colocó el botón de jugar, éste se conectó a un case, en el caso falso no se hace nada, en el caso verdadero, se evalúa si el control numérico “dinero” es mayor o igual a uno y eso se conecta a otro case anidado. En caso de ser falso, se muestra un mensaje emergente advirtiendo que falta dinero. Si es verdadero, se ejecuta la mayor parte del programa.

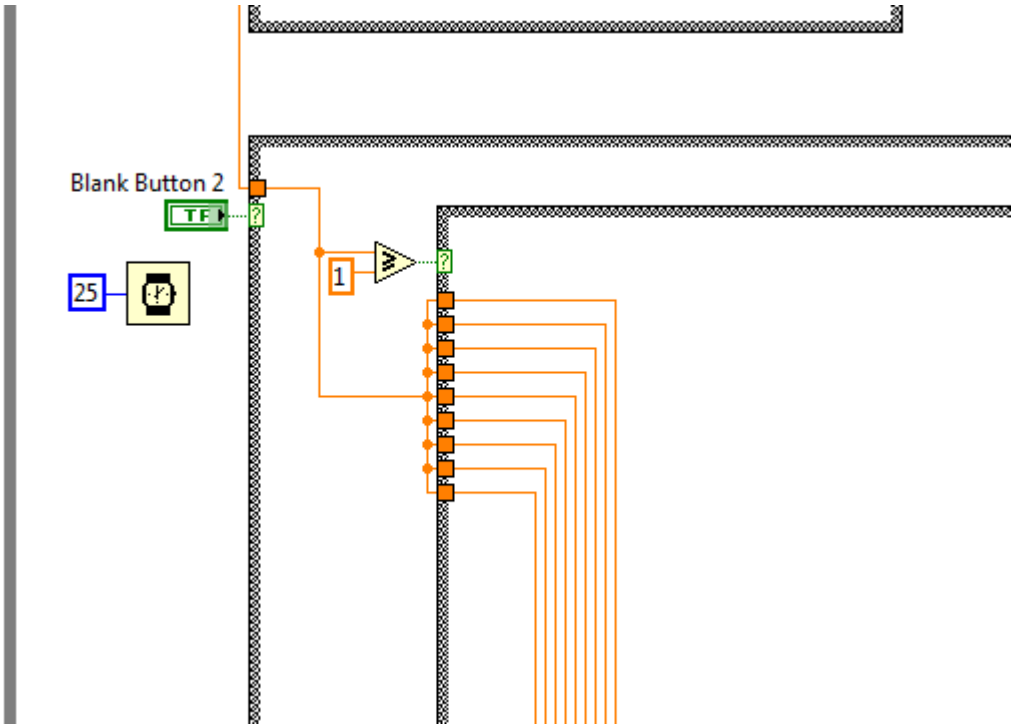


Imagen #83: Se evalúa si dinero es mayor o igual a 1.

Aquí es donde viene la parte más interesante, hacer que el led recorra todas las frutas, encendiéndose uno y apagando los que ya recorrió, empezando rápido hasta terminar lento.

Lo más viable para esto sería un ciclo for. El número de repeticiones del ciclo será la cantidad de vueltas que dará el led encendido en el tablero, ésta cantidad debe variar, ya que se trata de un juego de azar.

Entonces, podemos utilizar un número aleatorio (random), éste se encuentra en el menú numérico. Ahora, como se trata de un número aleatorio, podría variar entre 0 e infinito, entonces tendríamos que ponerle un límite. En este caso fue de cero a 150.

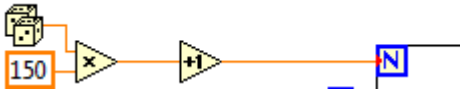


Imagen #83: Configuración del número de repeticiones.

Ahora, para que el led camine con una velocidad descendente, podemos usar un contador con un shift register. Se inicializa el shift register en cero, se conecta a un incremento y éste a un timing.

Esto ocasionará que el timing empiece en cero y vaya avanzando en uno conforme avanzan las repeticiones del ciclo.

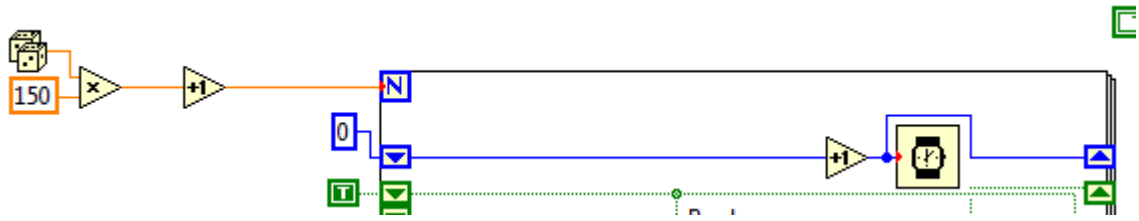


Imagen #84: Configuración de la velocidad de avance del led.

Para que al ejecutarse el ciclo for no se queden encendidos los leds, se inicializa el shift register en true y los demás, que corresponden a cada led, se quedan en false. Esto hará que no se queden encendidos, únicamente lo estará uno.

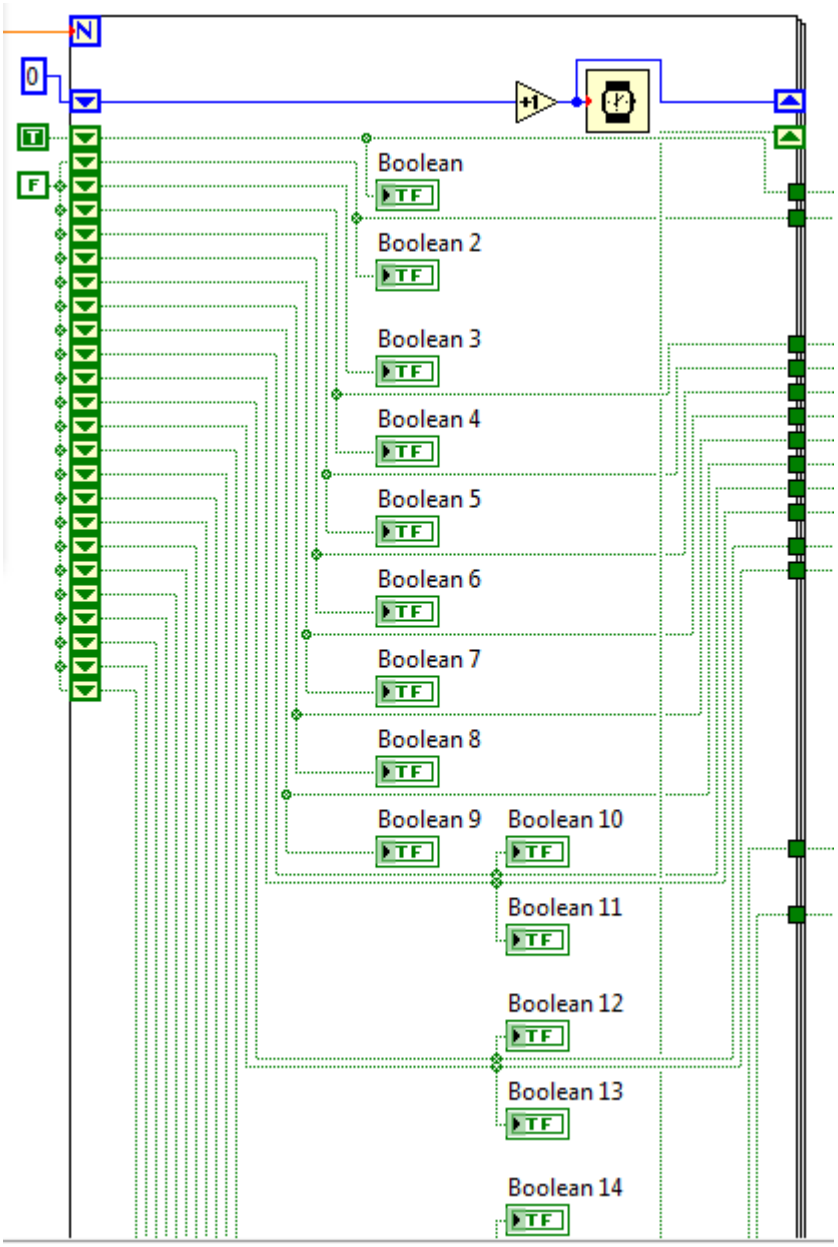


Imagen #85: Inicialización de shift registers.

Al correr el programa, el indicador de texto nos muestra un mensaje, éste se encuentra dentro del while, y se mostrará durante toda la ejecución del programa.

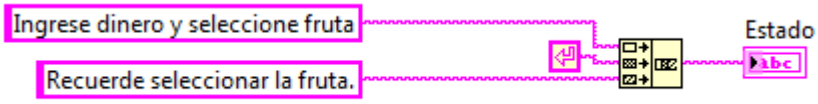


Imagen #86: Mensaje de texto del programa.

Aquí se usan dos frases para el usuario, entre ellas usamos un Carriage Return Constant que nos servirá para colocar una oración debajo de la otra, es decir; deja un espacio como si se hubiera presionado un enter al escribir o parecido al uso de \n en el lenguaje C.

Para que se haga la resta del dinero del usuario cada vez que presiona el botón jugar, utilizamos una variable local del control numérico.

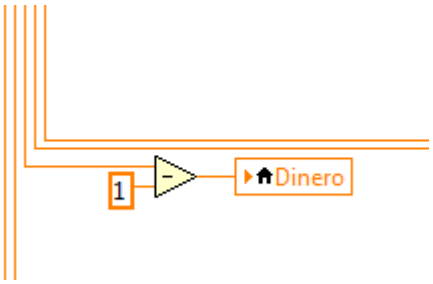


Imagen #87: Resta del dinero al presionar el botón "jugar".

Entonces, se conecta el control numérico "dinero" a una operación de resta, y el resultado se conecta a una variable local del mismo control numérico.

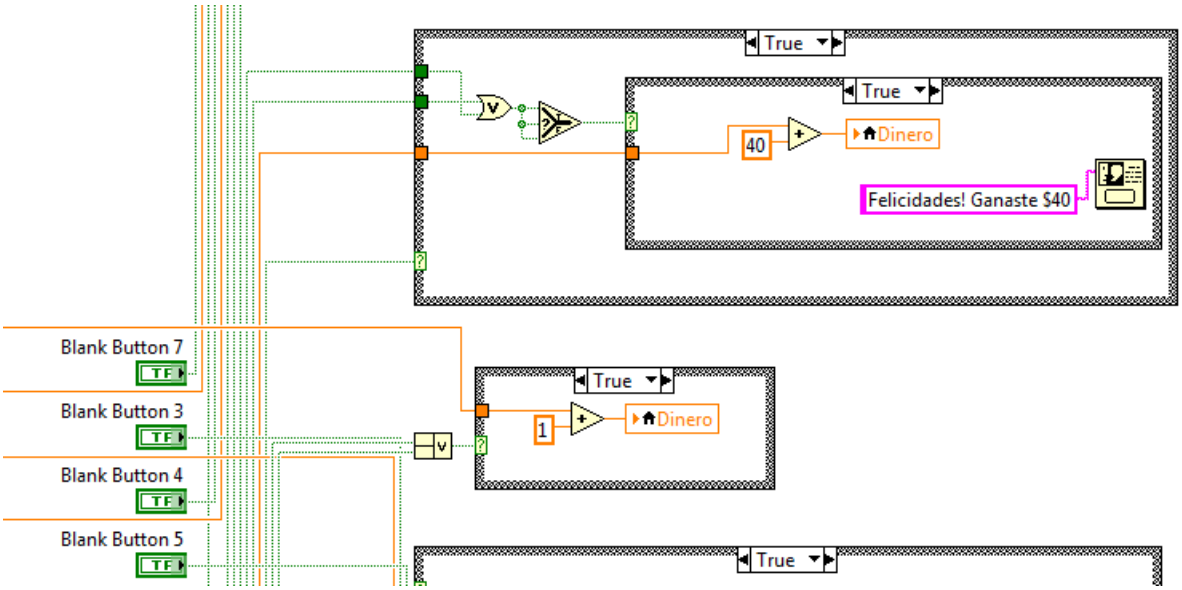


Imagen #88: Conexión de los botones a estructuras case.

Cada botón que representa las frutas del tablero se conecta a una estructura case, en esta, se usa una compuerta lógica OR para advertir que si X led se queda encendido se cumpla el caso verdadero. Si el case debe responder a más de dos leds, se utiliza un Compound Arithmetic para poder conectar más elementos a la compuerta.

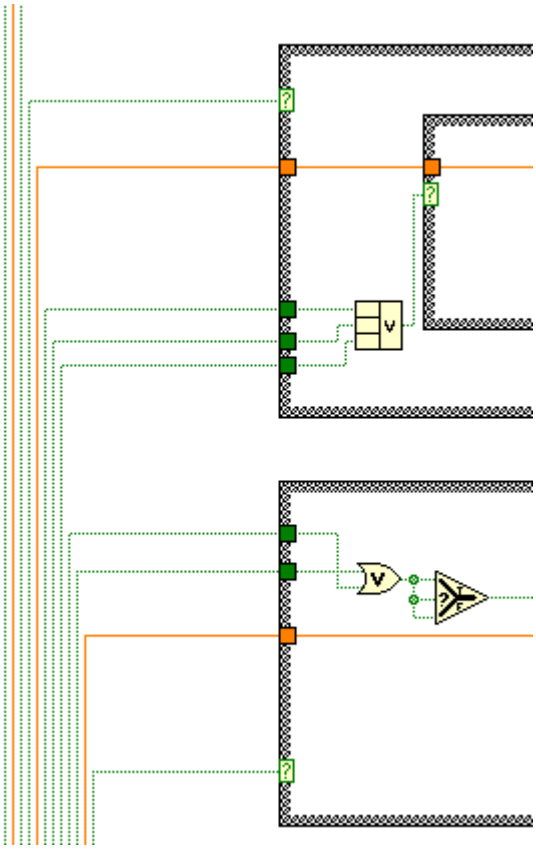


Imagen #89: Uso de compuerta AND en estructura case.

Esa compuerta se conecta a otro case anidado, en donde el caso falso no contiene nada, pero el verdadero alberga la operación que se realiza si el led cae en la fruta que el usuario eligió.

Por ejemplo, en el caso de la naranja, la estructura case es la siguiente:

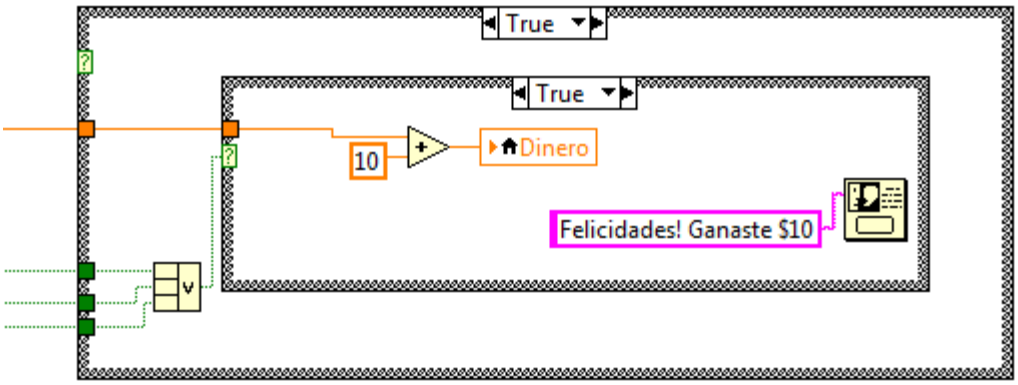


Imagen #90: Casos para leds representativos de la naranja.

La naranja se repite tres veces en el tablero, por lo tanto son 3 leds los que se usan para esa fruta, si uno de esos tres queda encendido al terminar el ciclo for, se utiliza una operación de suma para añadirle el valor de la fruta, 10 en este caso, al dinero del usuario, para esto se utiliza una variable local del control “dinero”.

También se utiliza el elemento One Button Dialog para mostrar un mensaje emergente al usuario felicitándolo.

Lo mismo se usa para cada botón, entonces se utilizan nueve estructuras case que a su vez anidan otra.

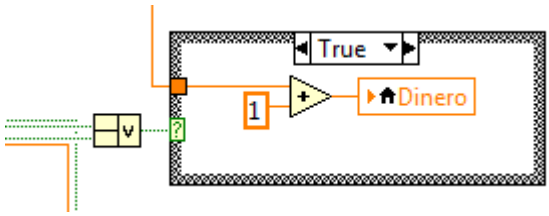


Imagen #91: Casos para salto de turno.

Si el led se detiene en la figura del muñequito que hay a los lados del tablero, sólo se regresa la cantidad que el usuario gastó.

Para terminar nos queda la parte donde el usuario decide cobrar el dinero que ganó o el dinero que ingresó.

Para esto utilizaremos otro contador, pero esta vez descendente.

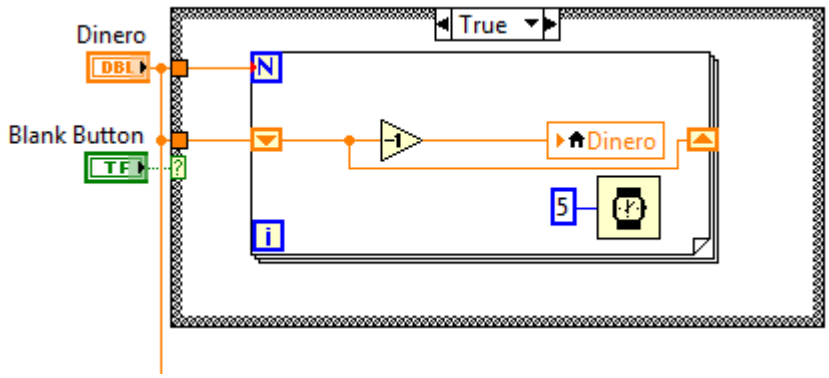


Imagen #92: Contador descendente para botón de cobro.

Se conecta el botón de “cobrar” a una estructura case, en el caso falso no se hace nada, en el verdadero, se utiliza un ciclo for.

El número de repeticiones del ciclo for será la cantidad que haya en el control numérico “dinero”. Se utiliza un shift register, que se inicializa también con el control numérico “dinero”, se conecta a un decremento y luego a una variable local del mismo control numérico.

Con un timing de 5 mili segundos determinamos la velocidad del contador, entonces, al presionar el botón, los números en el control irán corriendo descendentemente, simulando la caída de dinero de la máquina.

Es importante aclarar que la configuración del botón cobrar quedará en la acción mecánica Switch when pressed. El objetivo de esto es que mientras se tenga presionado el botón, se ejecute la acción. Si se presiona y se suelta, sólo se ejecutará el for una o dos veces, pero si se deja presionado, se ejecutará hasta que se quede en ceros. Esto le agrega más realismo al juego.

Esto también se aplica para los botones de las frutas.

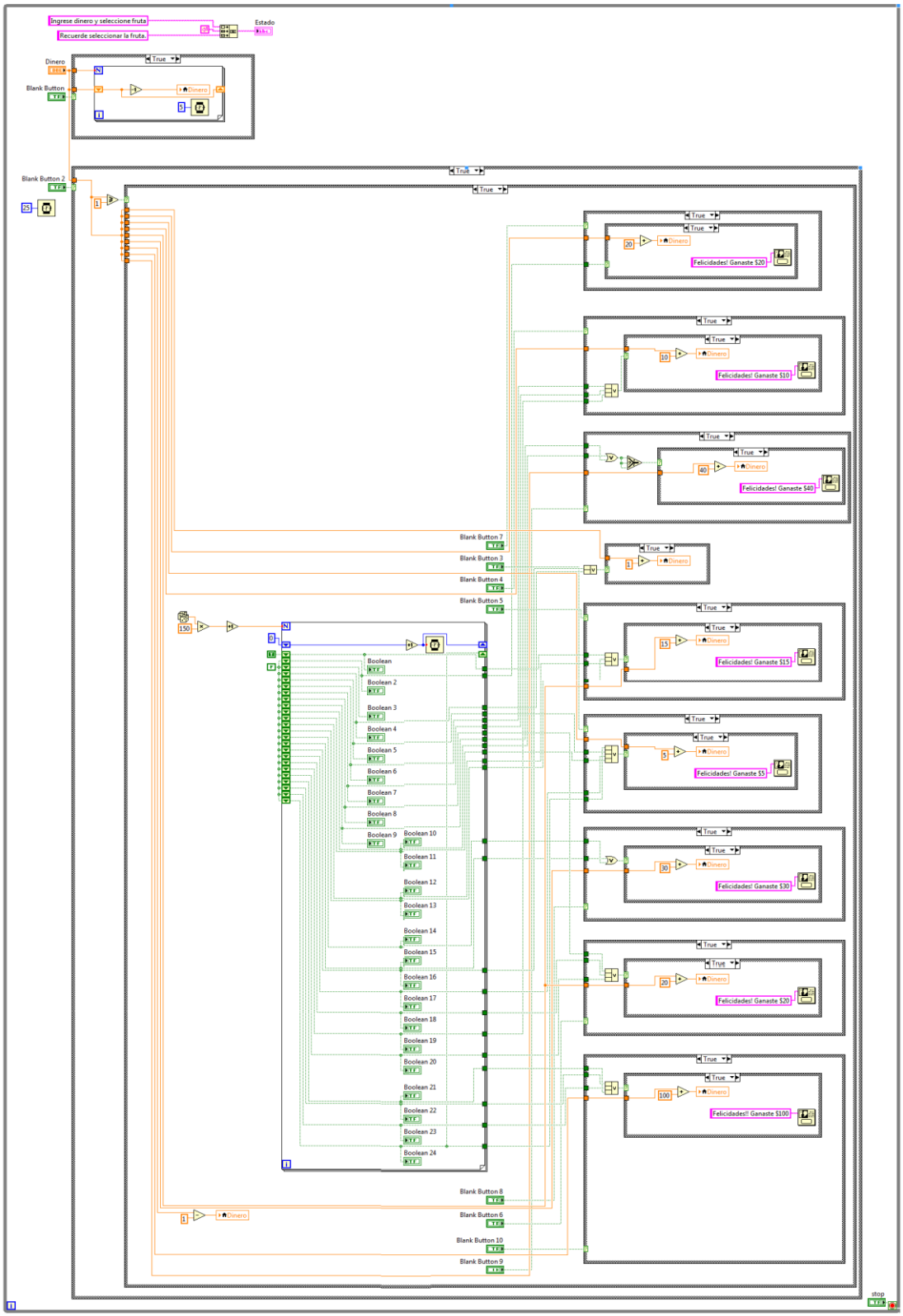


Imagen #93: Código completo de la máquina tragamonedas.

Puedes ver un video del programa funcionando aquí:
http://www.youtube.com/watch?v=b_wL9Ncb_nA

EDICIÓN DEL ÍCONO DE UN PROGRAMA

Como ya habíamos mencionado anteriormente, para editar el ícono del programa que estemos creando, nos vamos a la esquina superior derecha y damos un clic derecho sobre el ícono, luego damos clic en edit icon.

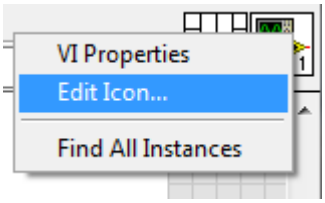


Imagen #94: Cómo entrar al editor de íconos.

También se puede acceder dando doble clic al ícono.

Al hacer eso, nos aparecerá una ventana como la siguiente:

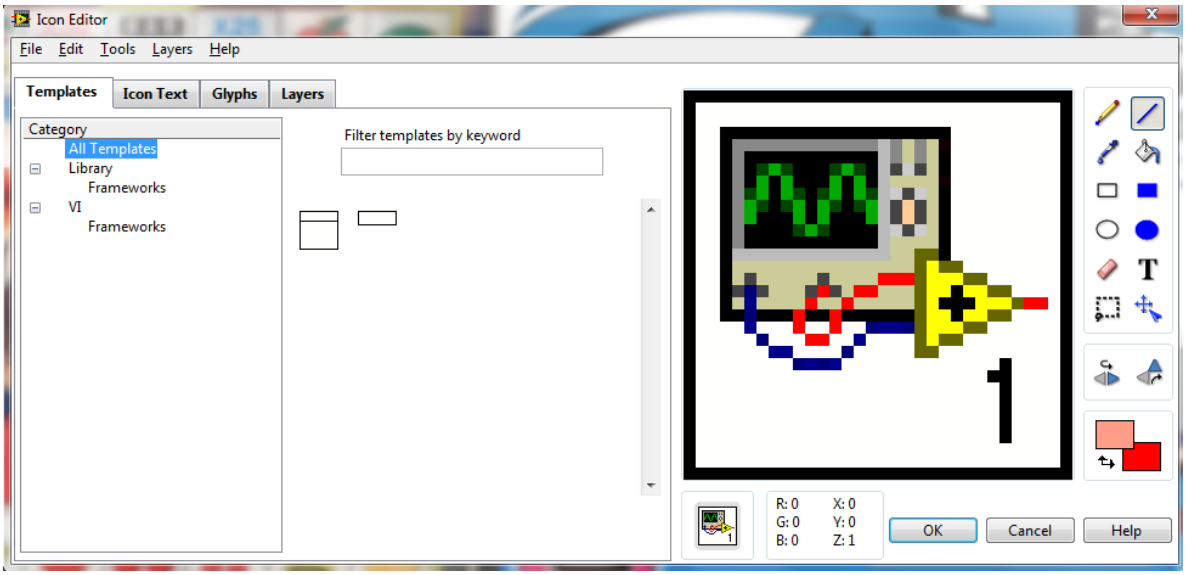


Imagen #95: Editor del ícono del programa.

Ahí podemos editar el ícono como se requiera, se pueden utilizar las imágenes de la pestaña Glyphs.

Y desde ahí se puede editar el ícono de la forma que se requiera, usando imágenes, las herramientas del editor, entre otros.

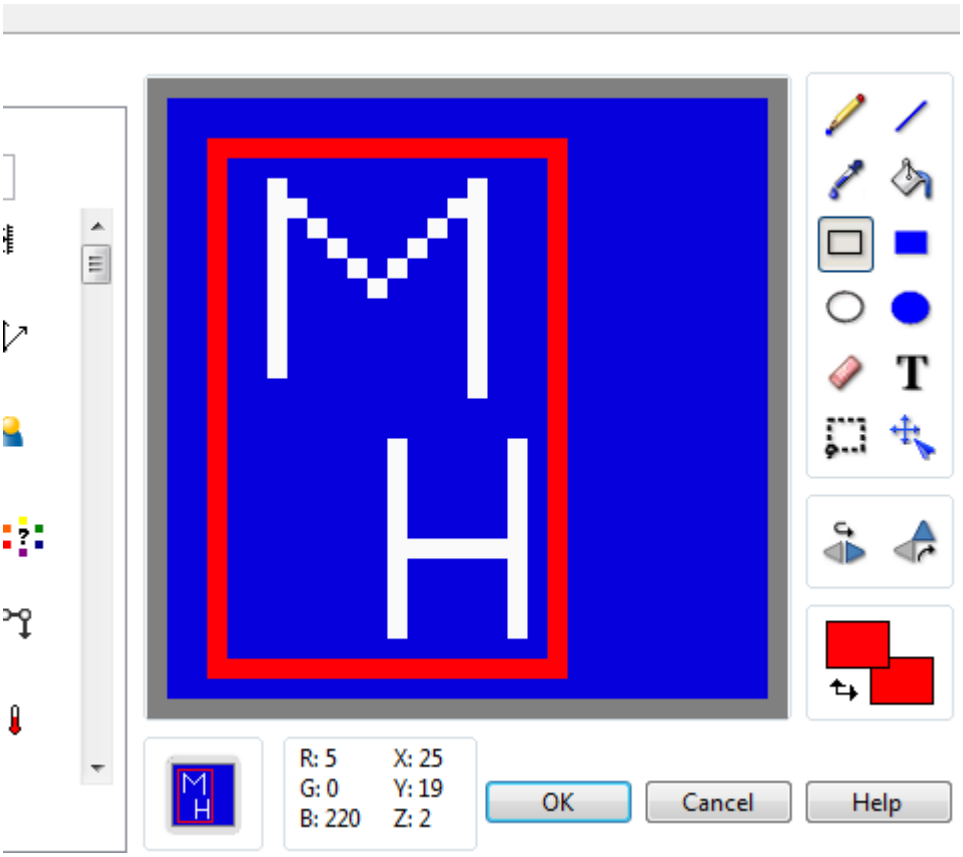


Imagen #96: Ejemplo de ícono.

Con esto se termina el contenido de este manual.

Primera edición terminada el 26 de febrero del 2013.