INGENIERÍA MECATRÓNICA

DI_CERO

DIEGO CERVANTES RODRÍGUEZ

INSTRUMENTACIÓN VIRTUAL

NI LABVIEW 2020 (32-BIT)

Bucle for y Shift Register

Contenido

Introducción Teórica de LabVIEW: 2

Introducción al Entorno de LabVIEW: 2

Front Panel: Ventana Gris con la Interfaz del Programa 4

Block Diagram: Ventana Blanca con la Lógica del Programa (Bloques) 4

Front Panel o Block Diagram - Show Context Help: Descripción de Bloques 5

Front Panel y Block Diagram: Navegar de una Ventana a Otra 6

Block Diagram - Cambiar Nombre a los Bloques: Nombre de los elementos en el **Front Panel** 7

Block Diagram - Highlight Execution: Correr Más Lento el Programa 8

Coertion dot: Conversión Automática de Datos por Parte de LabVIEW 8

Block Diagram - Clean Up Diagram: Organizar Automáticamente los Bloques del VI 8

Programa: Bucle for y Shift Register 9

Desarrollo del Programa: Movimiento de Datos en un Shift Register 9

Block Diagram - Bucle For: Uso de Memorias (Registros) Shift Register 9

Block Diagram - Bucle For - Shift Register: Registros de Memoria dentro de un Ciclo 10

Block Diagram - Add: Suma de dos Números Cualquiera 12

Block Diagram - Wait Until Next ms Multiple: Temporizador en milisegundos 13

Block Diagram - Highlight Execution: Correr Más Lento el Programa 14

Ejecución del Programa: Movimiento de Datos Shift Register 15

Block Diagram - Bucle For - Túnel (index): Sacar Información del Bucle en un Vector 15

Block Diagram - Greater ?: Operación Lógica Mayor Que (>) 16

Block Diagram - Index Array: Indicar la Posición de un Array 17

Front Panel - Waveform Graph: Ventana que Muestra una Señal **(Array)** 18

Ejecución del Programa: Waveform Graph (Arrays) de un Shift Register 20

Ejecución del Programa: Waveform Graph de un Shift Register y su Array 22

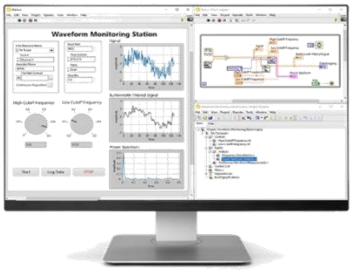
Introducción Teórica de LabVIEW:

LabView sirve para poder usar la computadora como instrumento de medición, monitoreo, control y análisis de procesos y operaciones, esto se hace a través de una frecuencia de muestreo que se relaciona con mediciones de los dispositivos digitales y tiene que ver con la señal de reloj de la tarjeta de desarrollo, indicando cada cuánto tiempo se hará un muestreo de cualquier señal del mundo real.

La diferencia entre los instrumentos virtuales de medición y los reales es más que nada el precio, ya que un osciloscopio cuesta alrededor de $10,000 y se puede hacer la misma función con LabView y un Arduino, que cuesta alrededor de $170, además de que es modular, esto implica que se pueden agregar o quitar funcionalidades. La mejor tarjeta de desarrollo para hacer esto es la de NI Instruments, que es la creadora de LabVIEW.

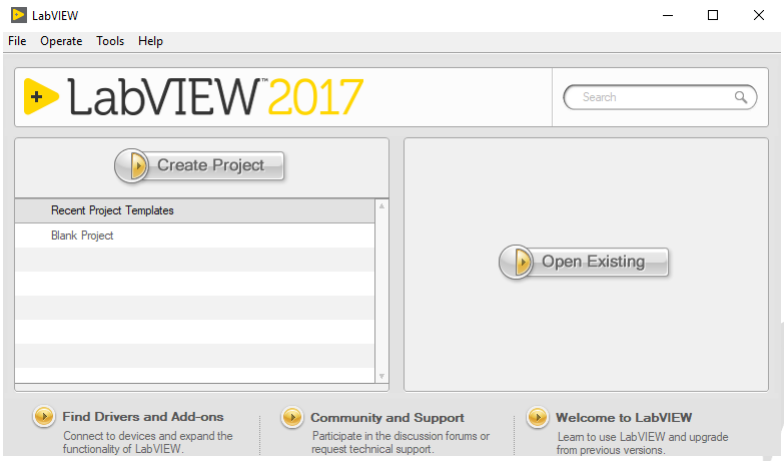
- **Instrumentación Tradicional:** El hardware es más usado, como por ejemplo con los circuitos integrados de un osciloscopio.
- **Instrumentación Virtual:** El software es el más utilizado y sus funciones son modulares, como lo es en una tarjeta de desarrollo de National Instruments.

La instrumentación virtual es empleada para la gestión de sistemas industriales y muy utilizado en compañías como: Ford, SpaceX, Accenture, Bosch, etc.

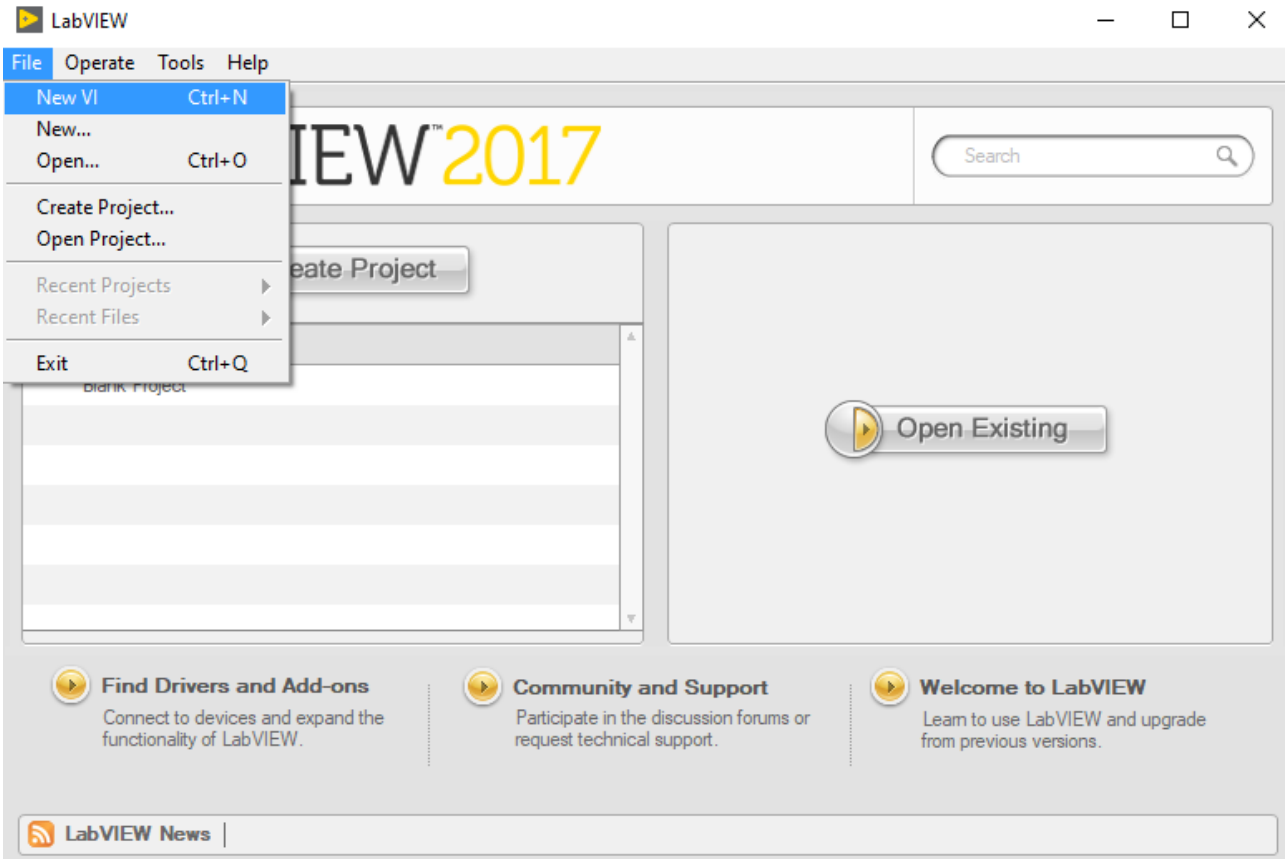


Introducción al Entorno de LabVIEW:

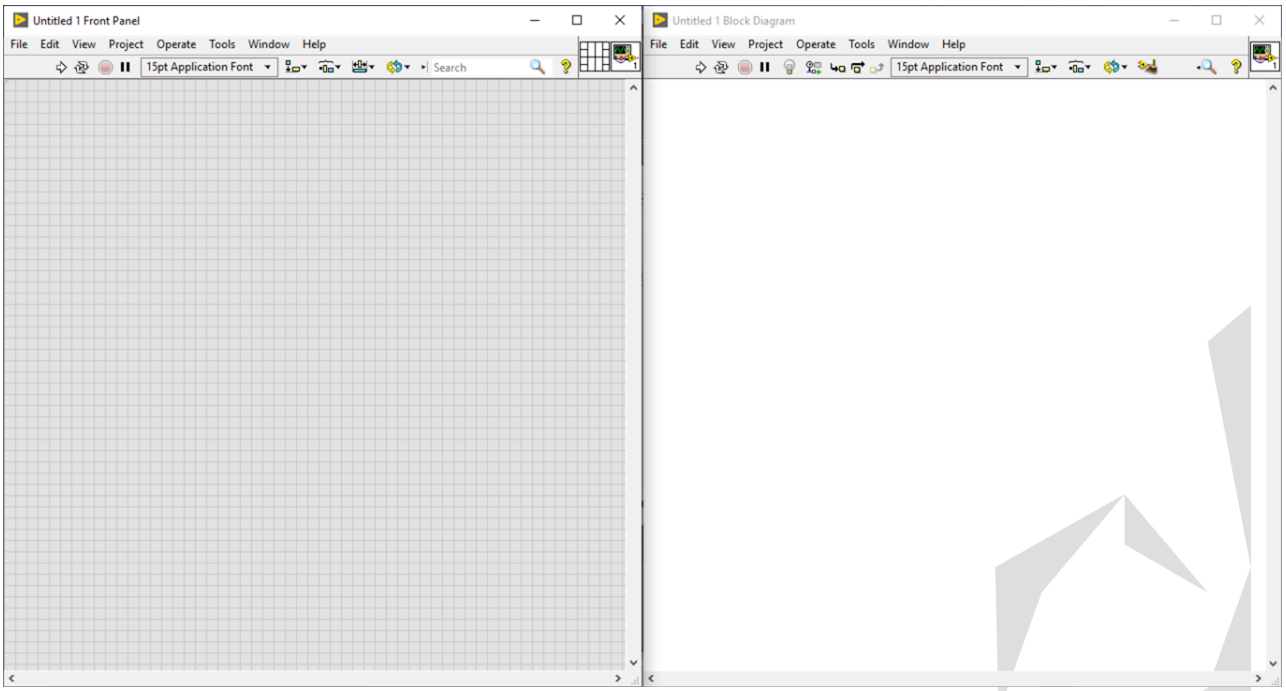
Un nuevo proyecto de LabView se abre por medio del botón de Create project que aparece inmediatamente cuando abra el programa.



VI se refiere a Virtual Instrument.

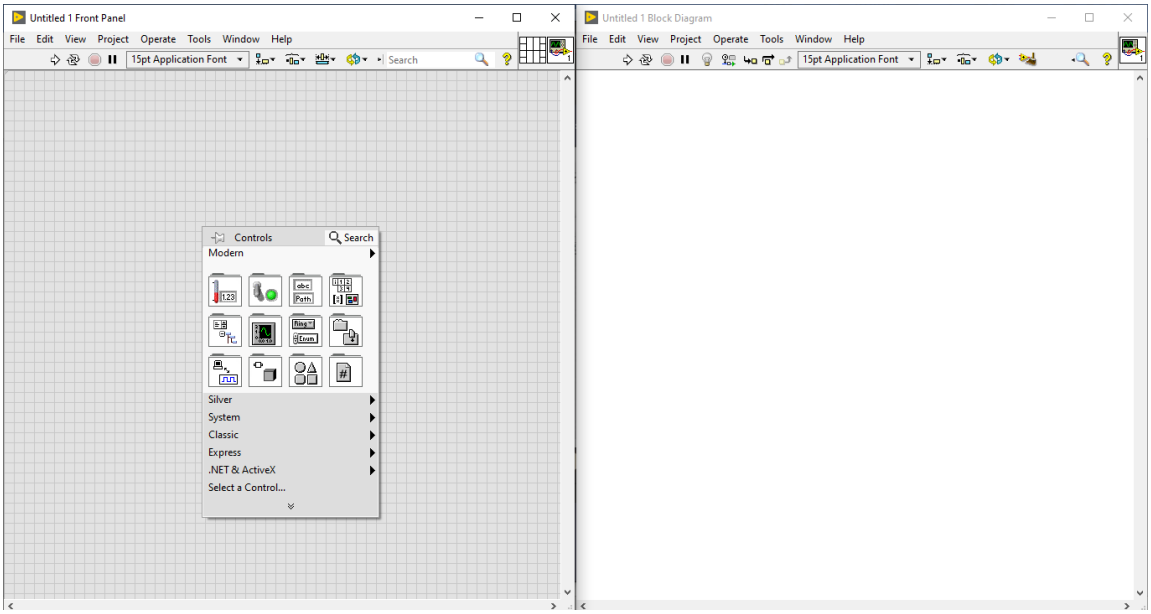


Al hacerlo me abrirá estas dos ventanas, en una de ellas se creará el programa con bloques (Ventana Block Diagram) y en la otra se verá la interfaz (Ventana Front Panel).



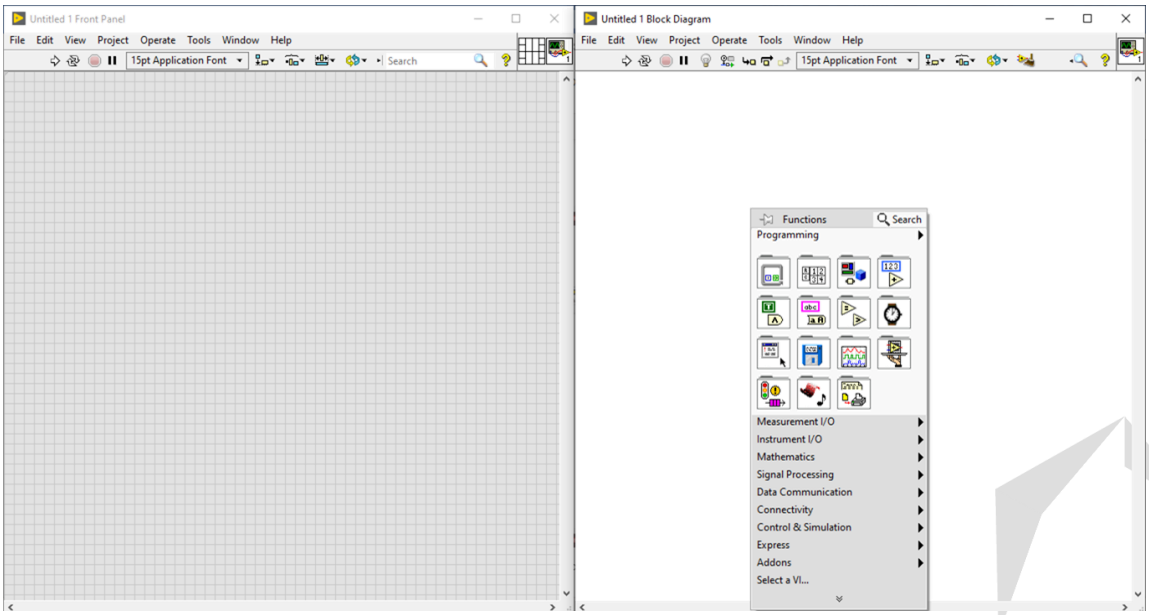
Front Panel: Ventana Gris con la Interfaz del Programa

En la ventana gris llamada *Front Panel*, es donde se observa la interfaz del Programa y se cuenta con el control pallete que sirve para poder añadir elementos gráficos a la interfaz y aparece dando clic derecho en la pantalla gris. *Si no aparece la otra ventana (blanca) por default, se debe seleccionar la opción Window → Show Block Diagram y con ello aparecerá.*



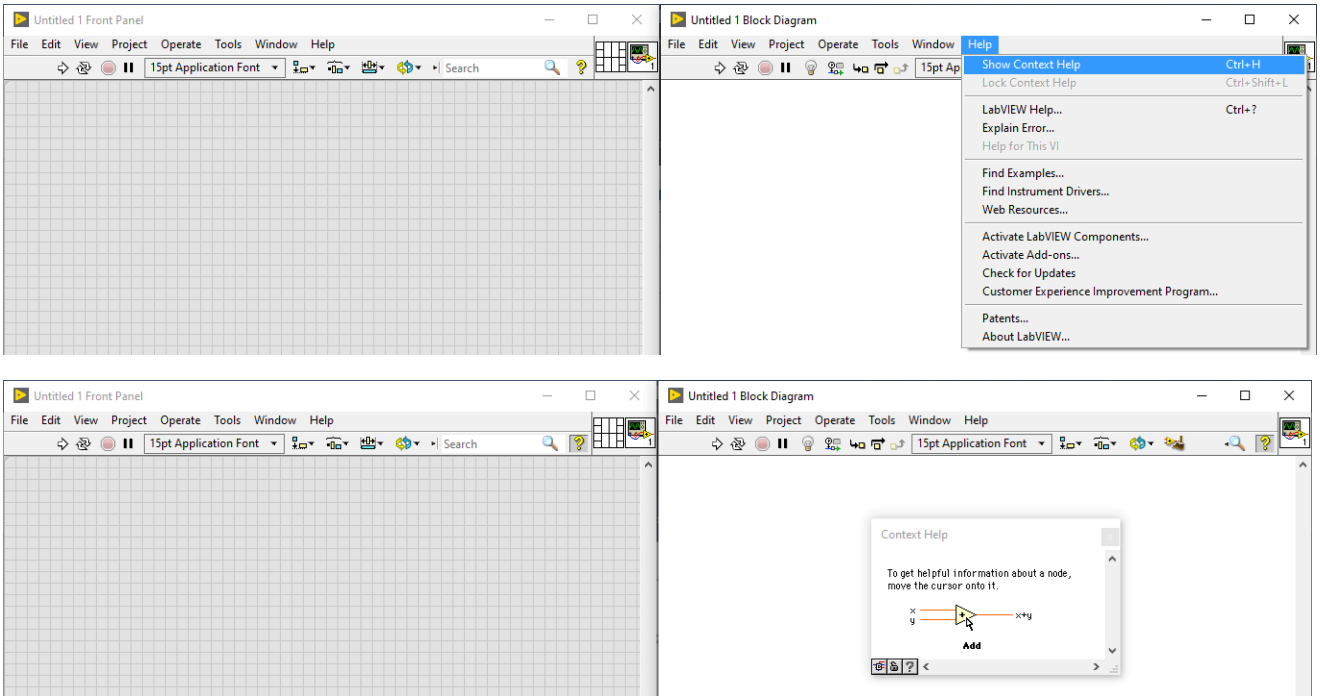
Block Diagram: Ventana Blanca con la Lógica del Programa (Bloques)

En la ventana blanca llamada *Block Diagram* aparece la paleta de funciones que sirve para introducir los elementos de programación en forma de bloques que se conectarán entre ellos y describirán la función del programa, aparece dando clic derecho en la pantalla gris. Si no aparece la ventana gris se debe seleccionar la opción Windows → Show Front Panel y con ello aparecerá.

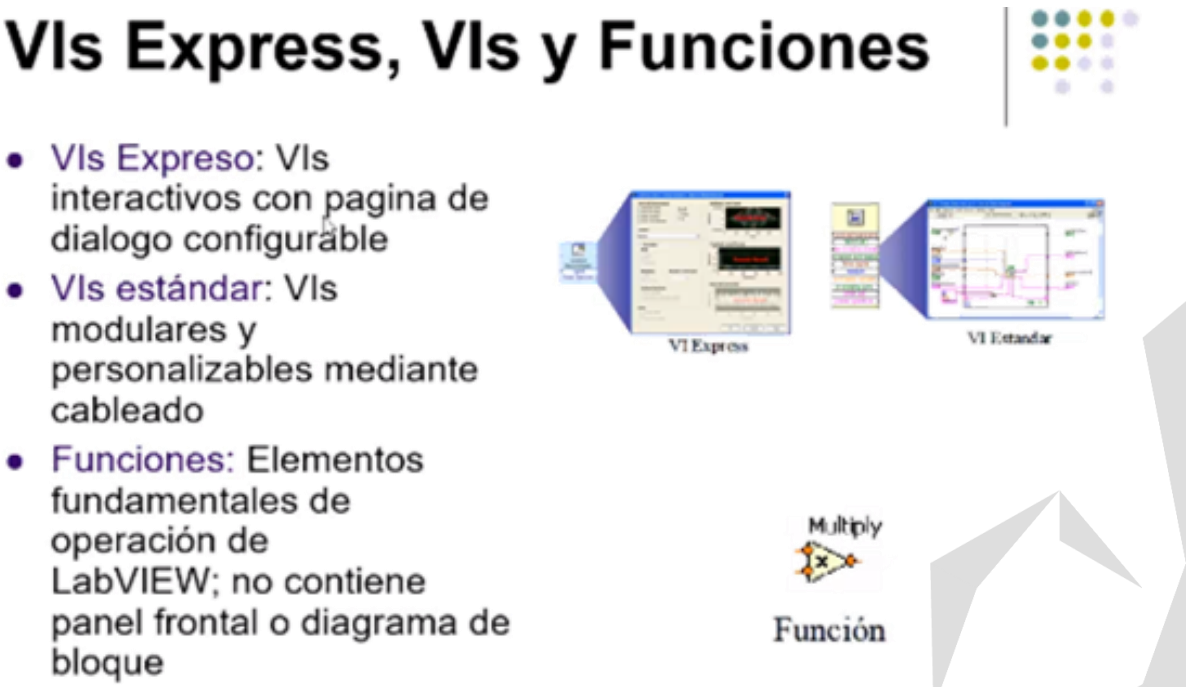


Front Panel o Block Diagram - Show Context Help: Descripción de Bloques

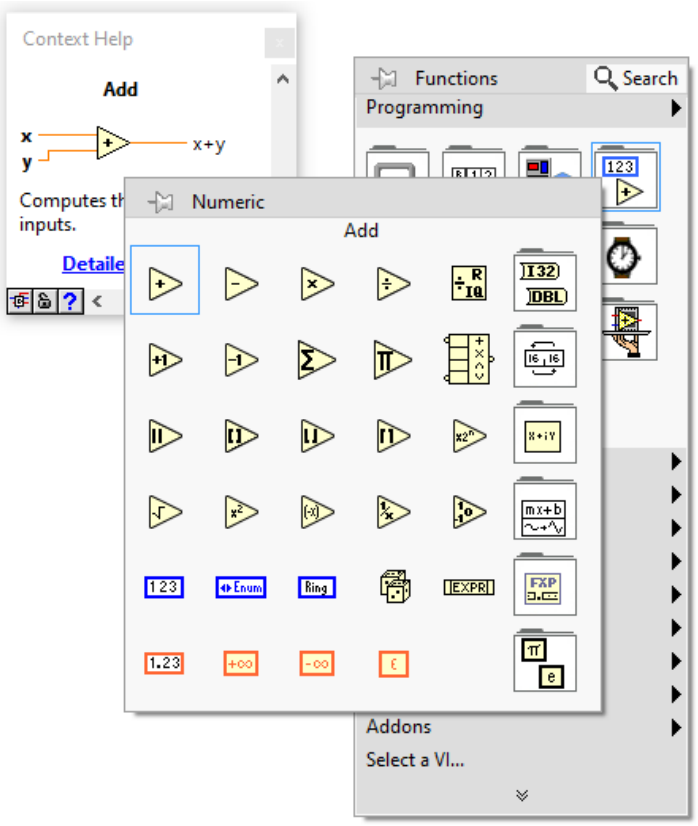
Seleccionando la opción de Help → Show Context Help, aparecerá una ventana emergente que explicará las propiedades de los bloques que se puede seleccionar, mostrando una descripción de su función, imágenes explicativas y significado de sus pines de entrada y salida.



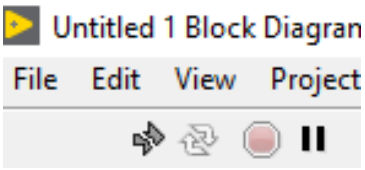
Las funciones o subrutinas son los elementos más básicos que pueden existir en LabView, dentro de ellas existe un código de bloque propio que describe sus funciones, pero además se cuenta con otros elementos:



En un bloque de código, l**as terminales que aparezcan en negritas son las que a fuerza deben estar conectadas a algo**, las que no estén en negritas no deben estar conectadas a nada forzosamente.

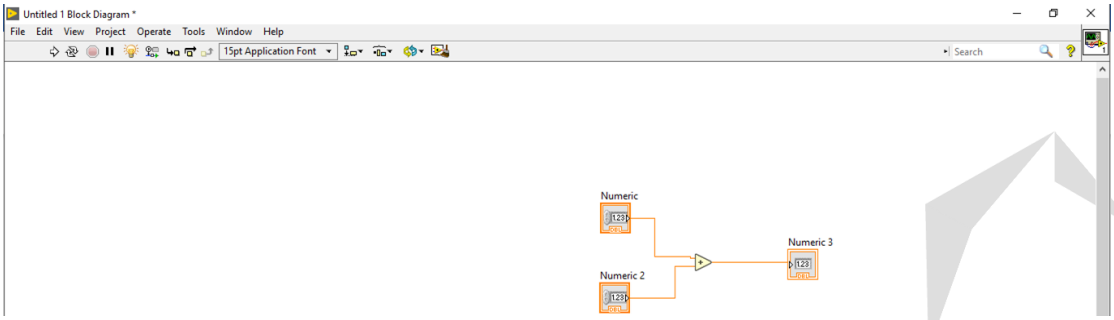


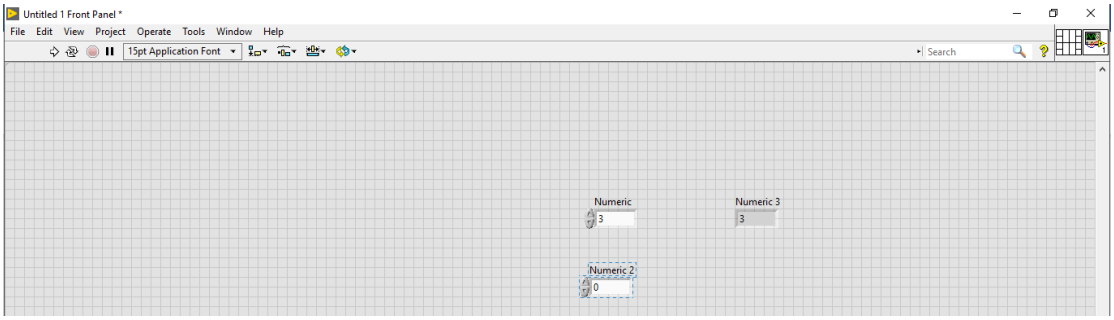
El programa es autocompilable, es decir que se corre por sí solo, por lo que si la flechita aparece rota es porque hay un error en el programa.

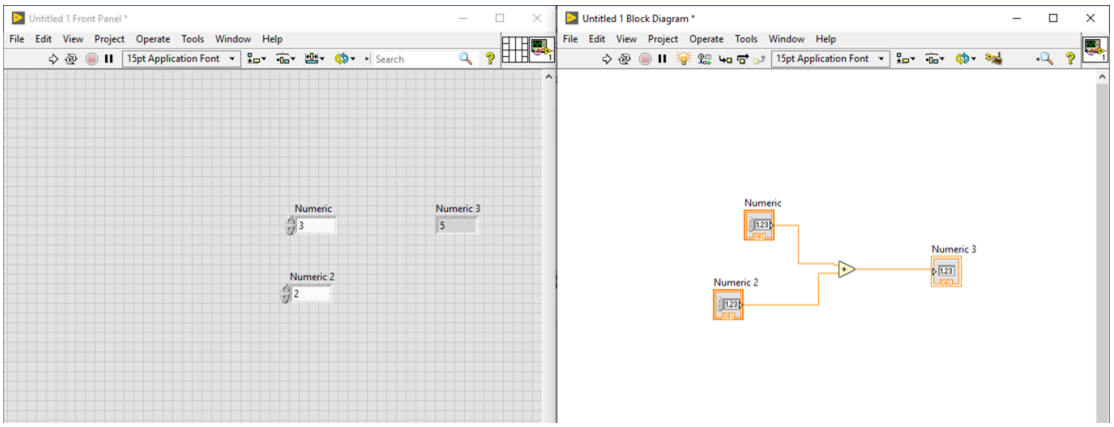


Front Panel y Block Diagram: Navegar de una Ventana a Otra

Al dar doble clic en el bloque de la pantalla blanca, me llevará al punto donde se encuentra el mismo bloque, pero en la pantalla gris.

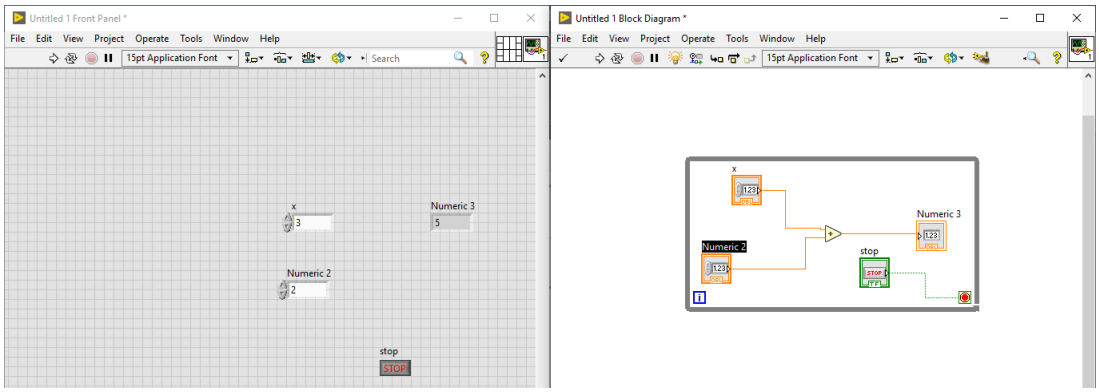


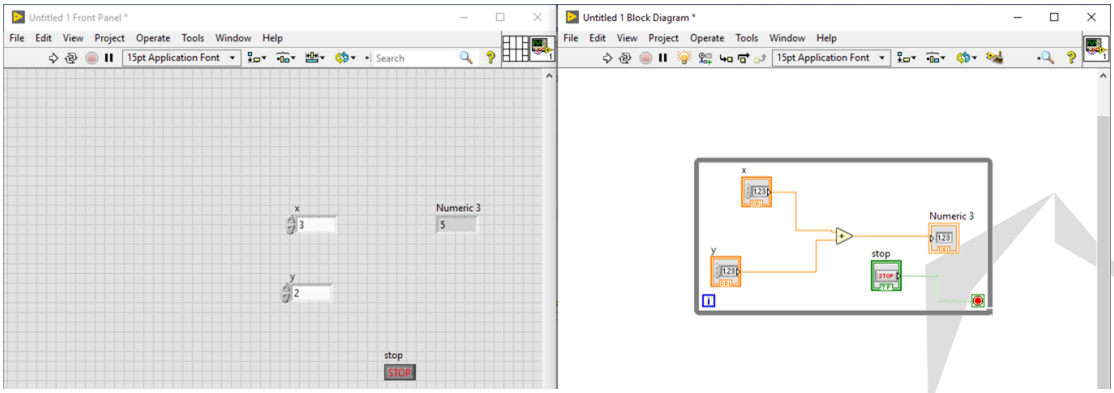


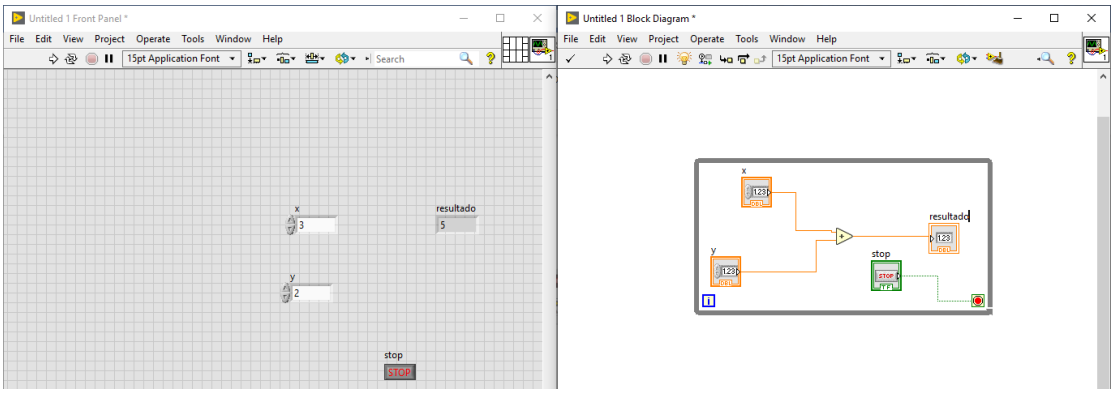


Block Diagram - Cambiar Nombre a los Bloques: Nombre de los elementos en el Front Panel

El nombre de los elementos de las interfaces se puede cambiar desde el Block Diagram, cambiándole literal el nombre a los bloques.

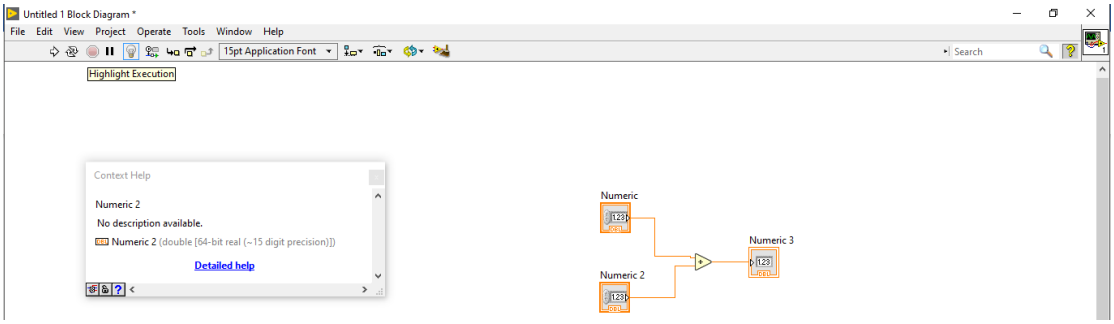






Block Diagram - Highlight Execution: Correr Más Lento el Programa

Podemos presionar el foquito del menú superior para ver el funcionamiento de programa de manera más lenta.



Coertion dot: Conversión Automática de Datos por Parte de LabVIEW

Aparece un punto rojo en la terminal del bloque llamado coertion dot, este lo que me dice es que los tipos de datos en la conexión son distintos, por lo que LabVIEW está forzando una conversión de un tipo de dato a otro, el problema es que en este tipo de conversión yo no sé si se están perdiendo datos, por eso debemos evitar el uso de coertion dots porque usa direcciones de memoria o recursos de la computadora sin que yo tenga control de ellos.

Block Diagram - Clean Up Diagram: Organizar Automáticamente los Bloques del VI

Con el botón de Clean Up Diagram que se encuentra en la parte superior derecha del Block Diagram se organizan mejor y de forma automática mis elementos.

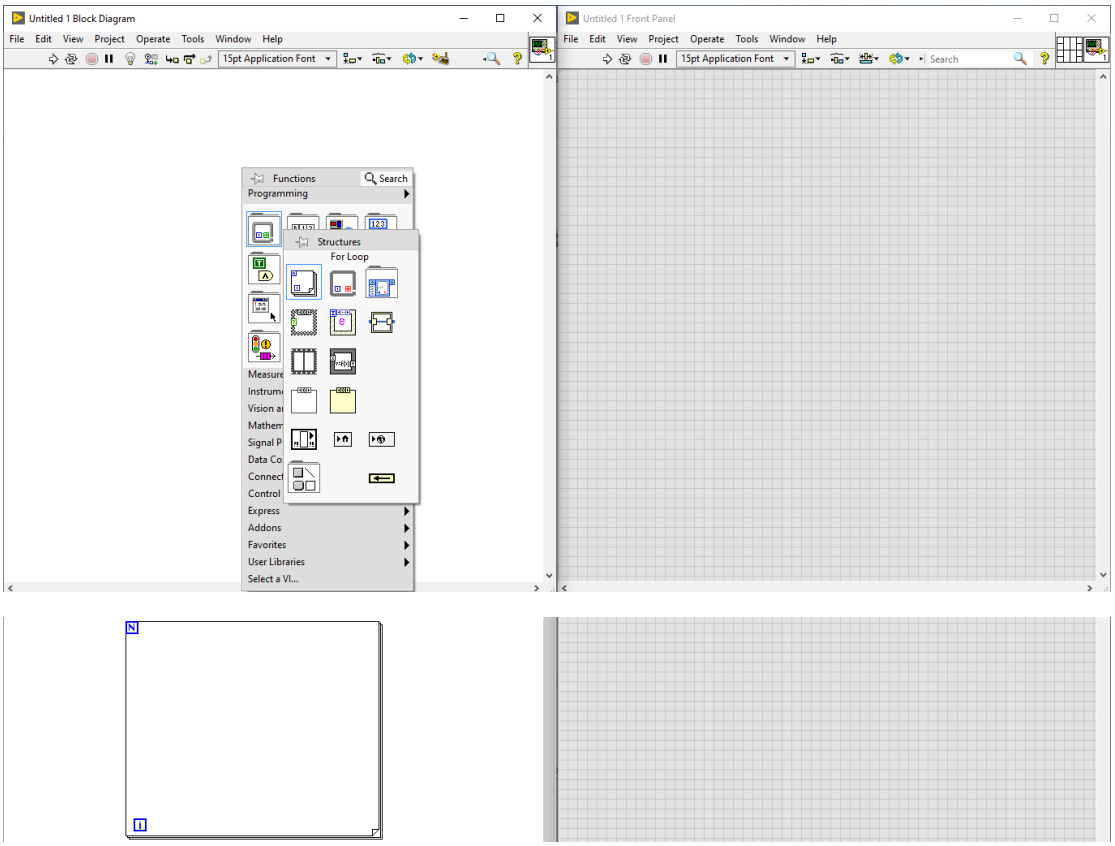
Programa: Bucle for y Shift Register

Movimiento de datos en una memoria de registros Shift Register, proveniente de un bucle for en donde además por medio de un Waveform Chart se grafica un Array obtenido del mismo Shift Register.

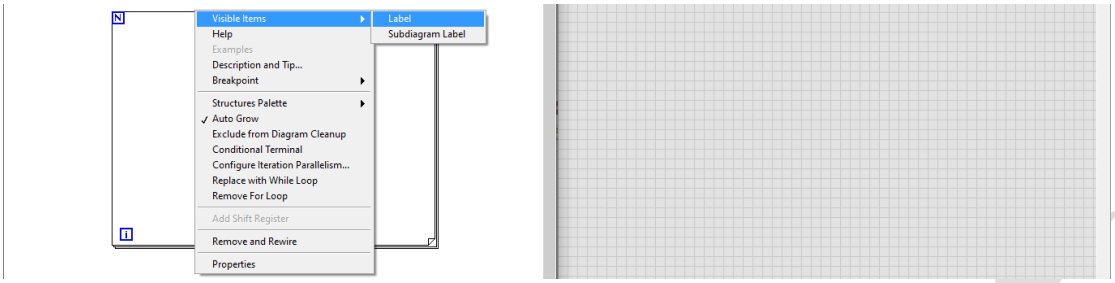
Desarrollo del Programa: Movimiento de Datos en un Shift Register

Block Diagram - Bucle For: Uso de Memorias (Registros) Shift Register

El ciclo for hace que el programa se ejecute un número finito de veces, indicado por las variables N e i.

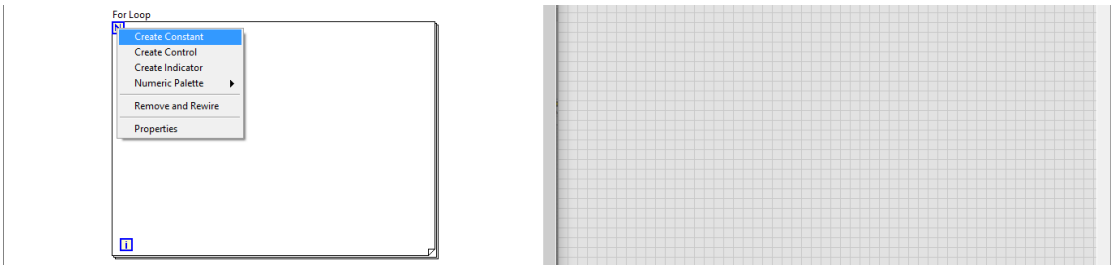


Mostrar nombre del bloque: Clic derecho → Visible Items → Label.



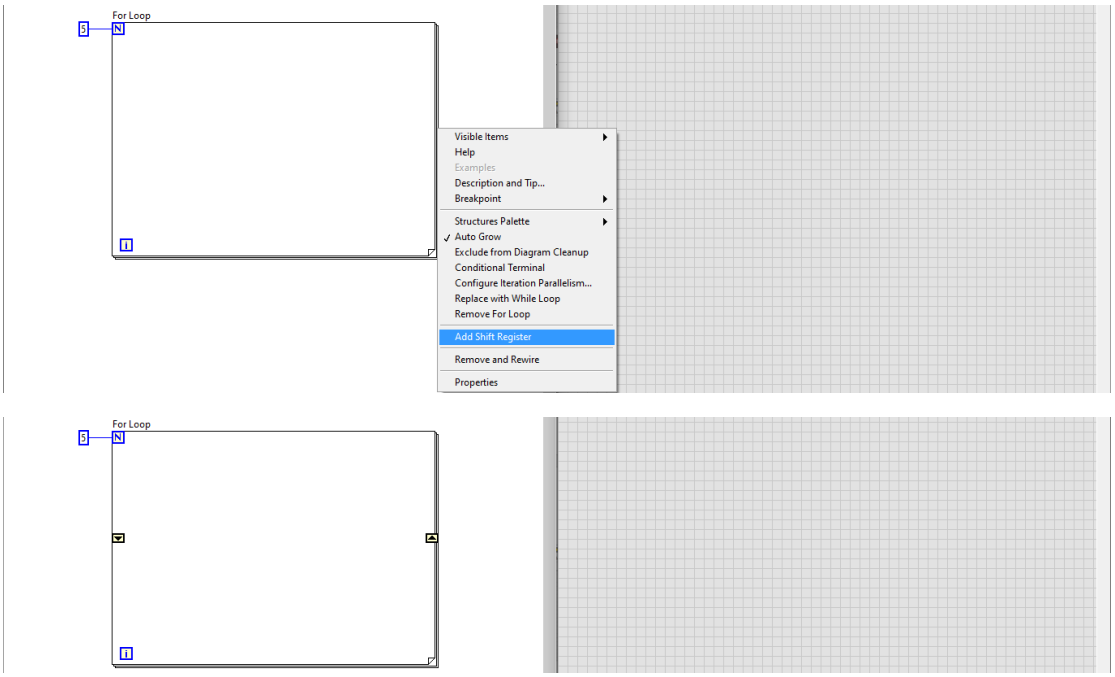
En el Bucle for La N nos dice cuántas iteraciones se van a hacer y la i es una variable que indica el paso del conteo, en otras palabras, nos dice de cuanto en cuanto vamos contando hasta llegar a N.

Crear una Constante para un Bloque: Clic derecho en la terminal del bloque de interés → Create → Constant.

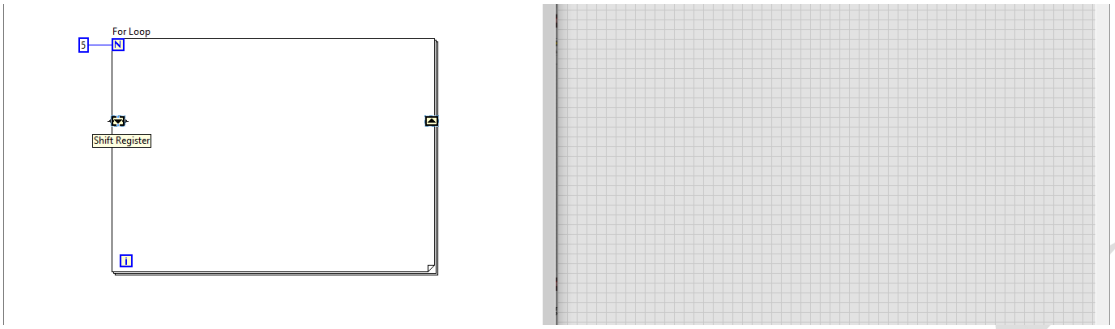


Block Diagram - Bucle For - Shift Register: Registros de Memoria dentro de un Ciclo

Ahora vamos a dar clic derecho al bucle y seleccionar la opción de Add Shift Register, esto es para que almacenemos en una dirección de memoria temporal algún dato generado en un ciclo for o while.



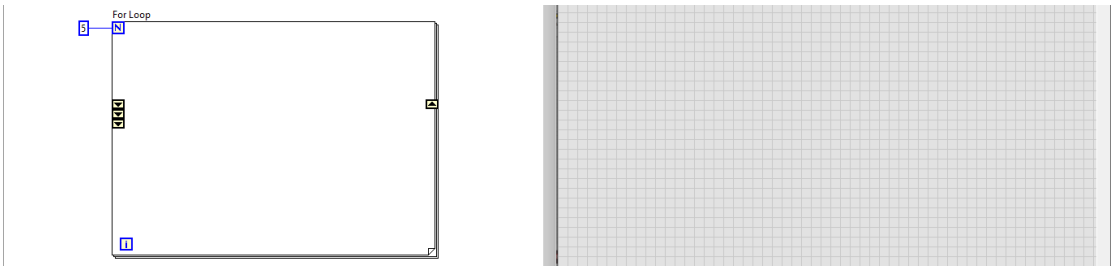
Cuando el Shift Register se muestra de color negro es porque no se le ha asignado un dato.



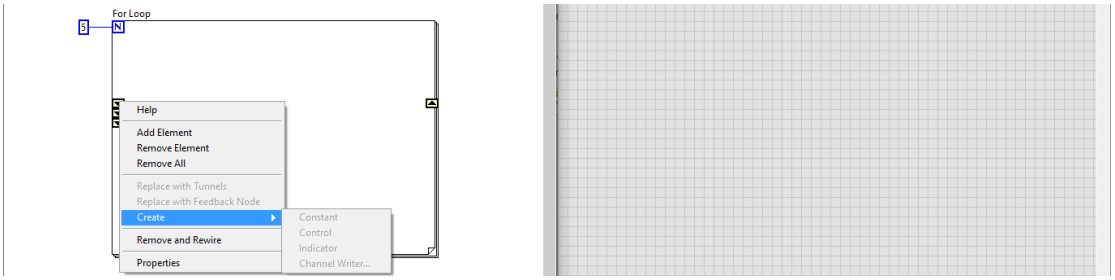
Si lo extiendo puedo ver las direcciones de memoria a donde se van a estar asignando los valores, cuando un dato entre al primer registro (posición de arriba hacia abajo) del Shift Register, si tenía un

dato almacenado ahí, lo va a mover a la siguiente posición (hacia abajo) y así se estarán moviendo y moviendo los datos en los registros del Shift Register mientras vayan entrando más y más datos a la memoria, hasta que el último dato ya no se pueda mover a otra posición, entonces se borrará ese dato.

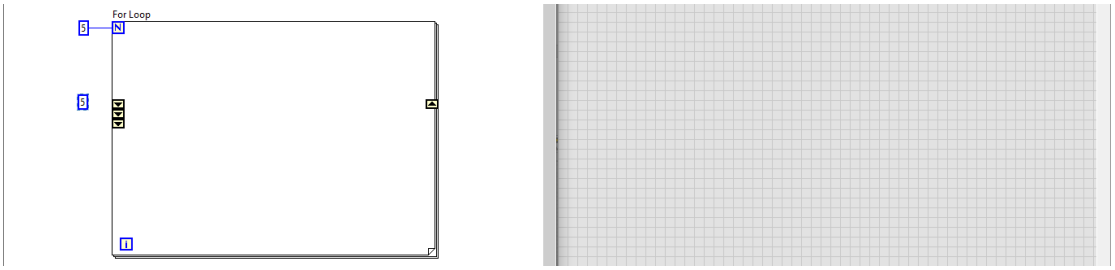
En este caso vamos a asignar 3 posiciones, llamadas registros de corrimiento (los de la izquierda) donde entrarán datos por medio del registro de la derecha.



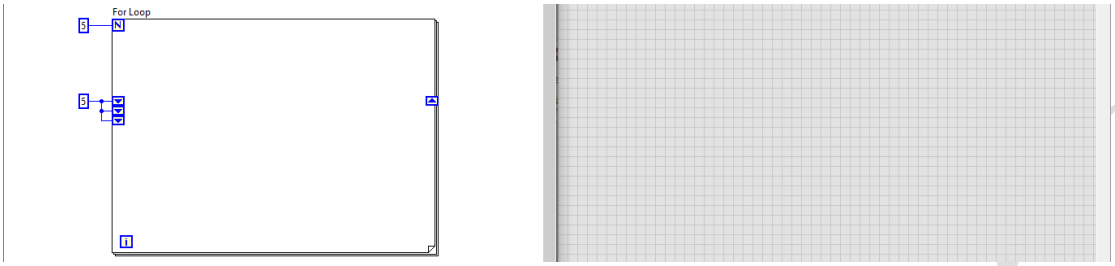
No se pueden crear constantes en el Registro del Shift Register porque no se ha indicado un tipo de dato.



Lo que voy a hacer es copiar y pegar una constante externa que ya tengo, la cual es la que indica la variable N = número de ejecuciones del bucle for.

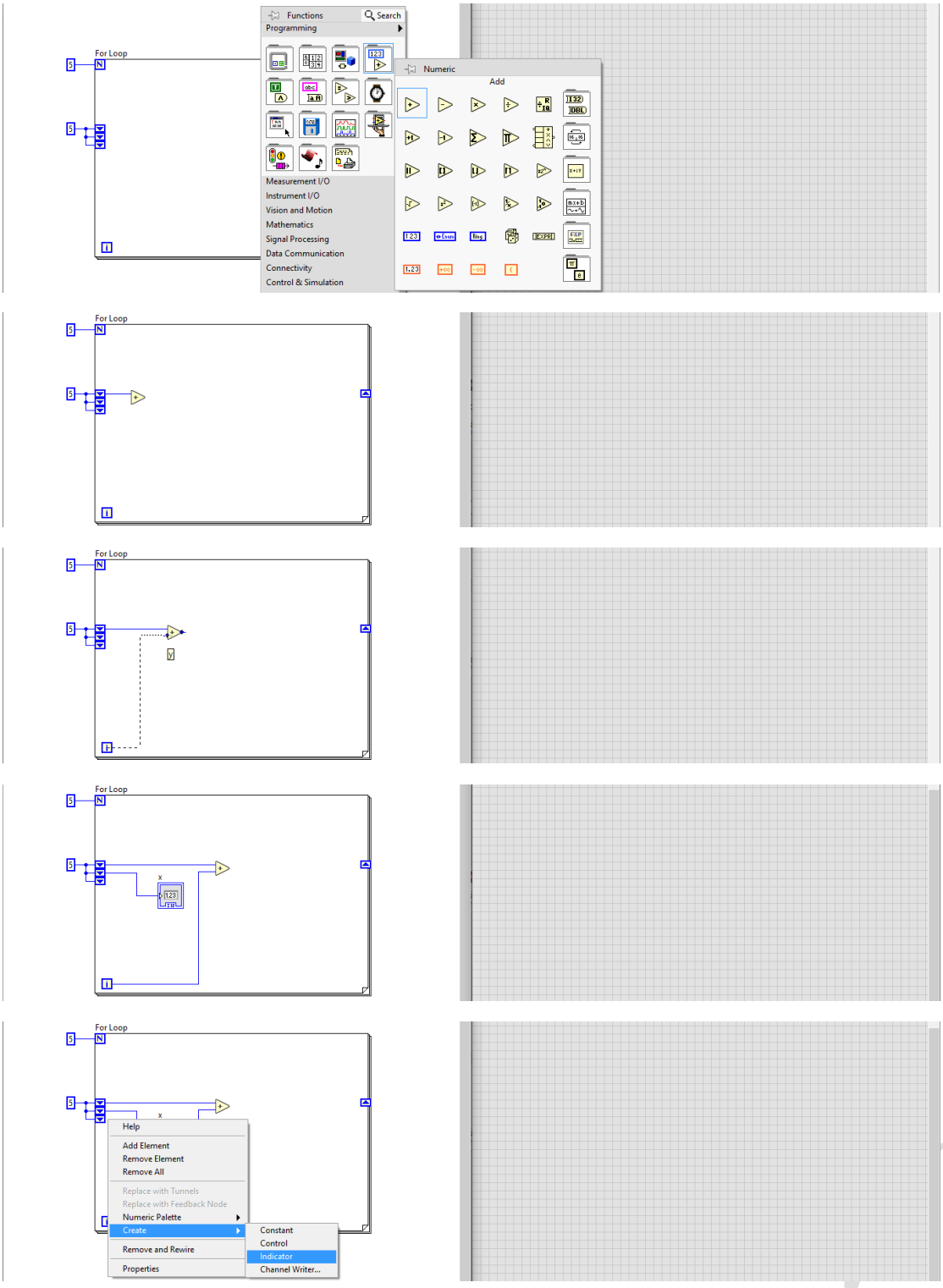


Y la conectaré directo al registro, al hacer esto ya le estaré dando un tipo de dato con un valor inicial de 5, además podemos ver que cambia de color, el cual es el mismo que tiene la constante externa que conecté.

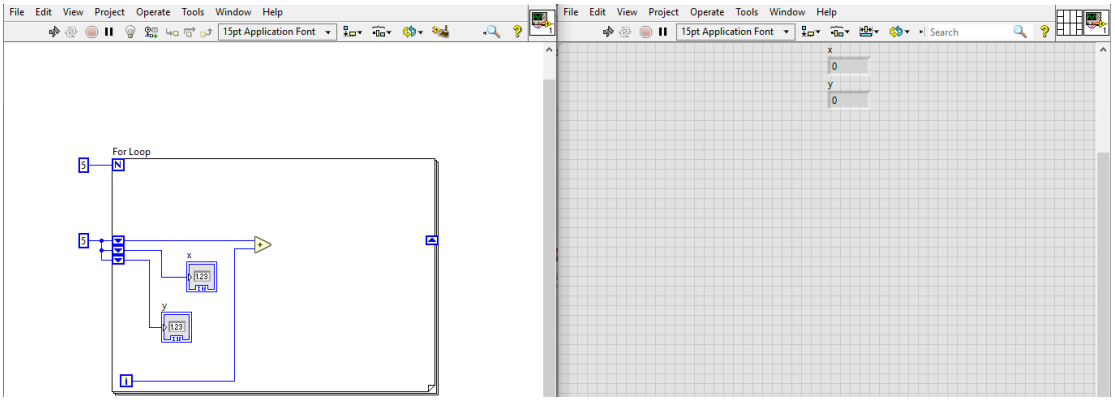


Block Diagram - Add: Suma de dos Números Cualquiera

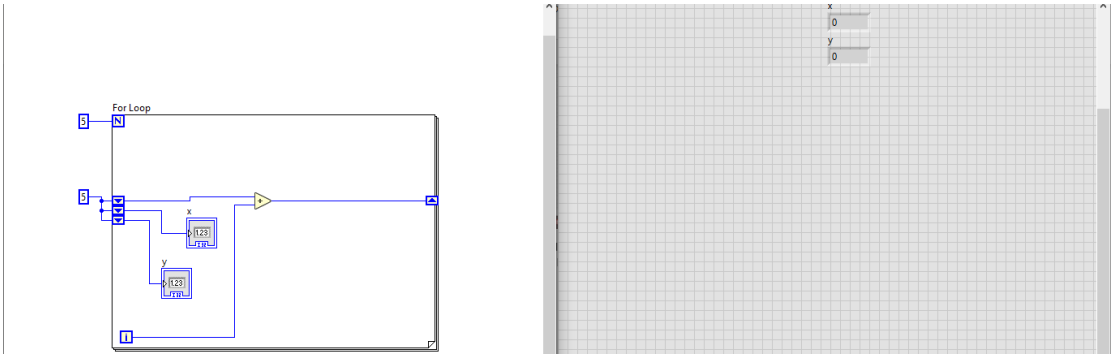
Todos los registros están inicializados con el valor de 5, pero se les realizarán operaciones matemáticas para que eso cambie, específicamente se les sumará el valor de la variable i, que va contando las ejecuciones del bucle for.



Crear un Indicador para un Bloque: Clic derecho en la terminal del bloque de interés → Create → Indicator. Con esto se crea un elemento de la interfaz donde sea visible el valor de la terminal del bloque.



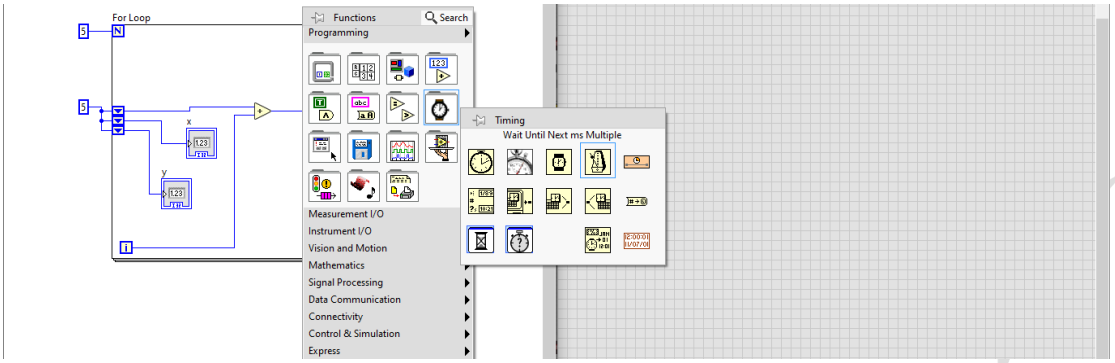
La suma se irá yendo de 1 en 1 porque el sumador está conectado a la i de iteración que cuenta de 0 a N y lo sumará a la condición inicial que está en el shift register.

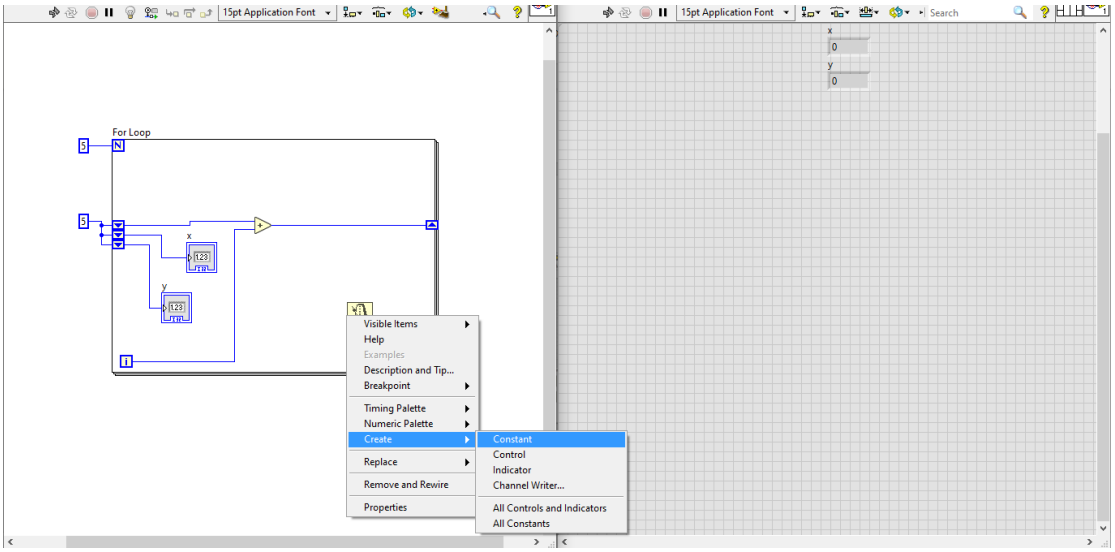


Cuando el resultado de la suma se conecta al shift register de la derecha, va alimentando a las posiciones del registro en la izquierda.

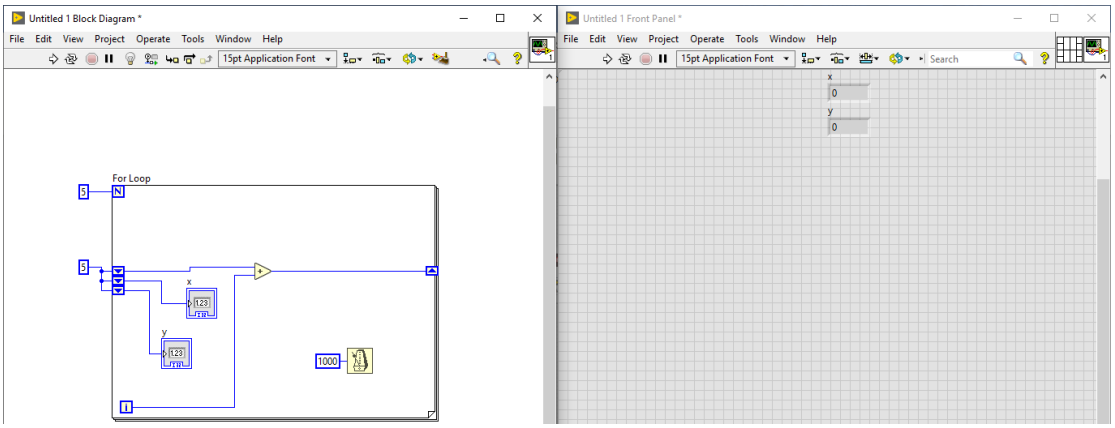
Block Diagram - Wait Until Next ms Multiple: Temporizador en milisegundos

El bloque de wait until se utiliza cuando se debe hacer un retraso de tiempo (delay) por ciertos segundos, para de esta manera parar la ejecución del programa por un cierto tiempo, en específico para que corra este bloque se debe crear una constante dando clic derecho sobre ella y declarando el tiempo en milisegundos.



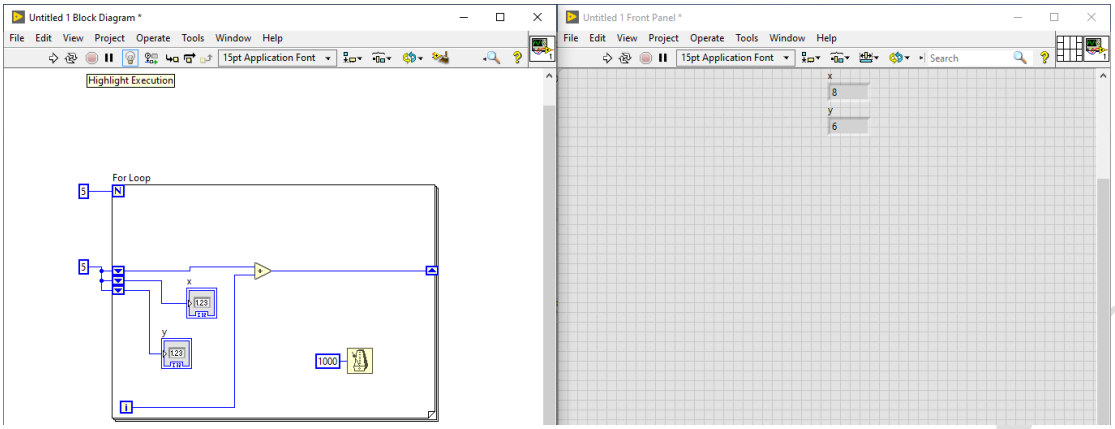


Se indica que tarde 1 segundo = 1000 milisegundos cada iteración del bucle for.

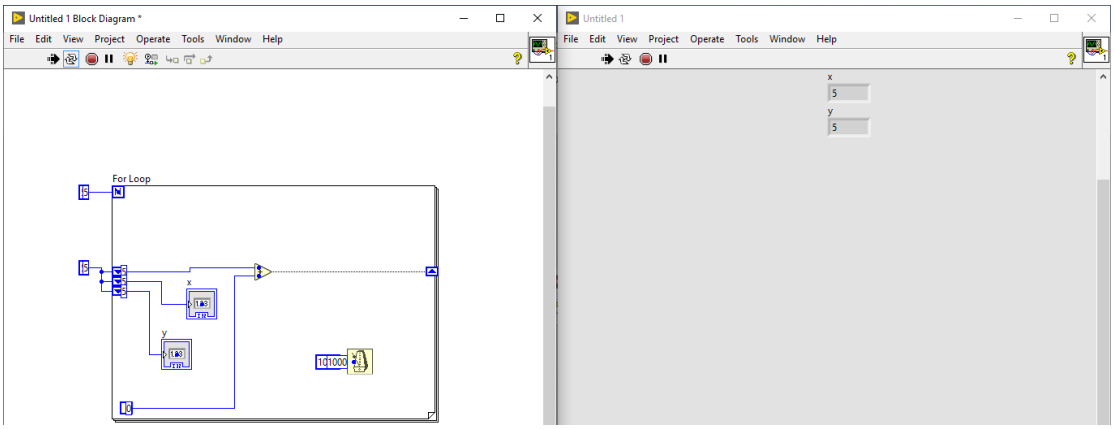


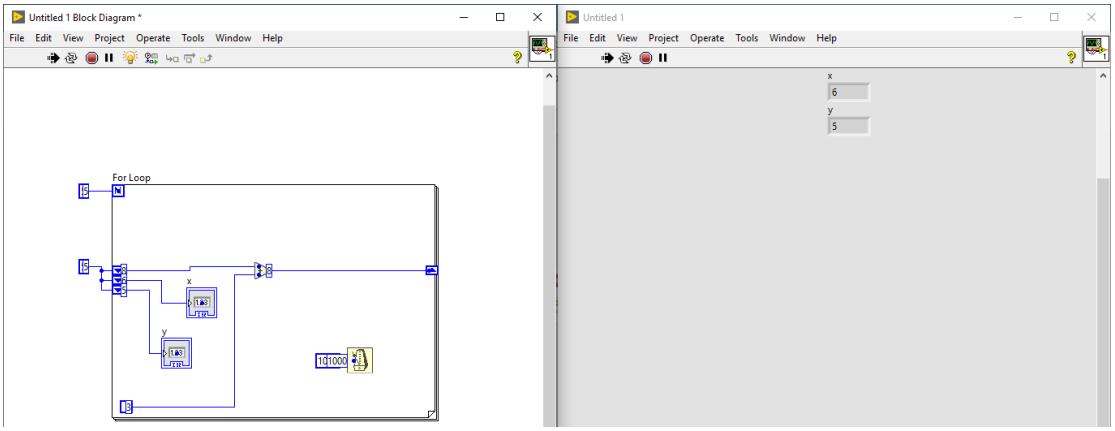
Block Diagram - Highlight Execution: Correr Más Lento el Programa

Podemos presionar el foquito del menú superior para ver el funcionamiento de programa de manera más lenta, esto se hace para ver cómo se van moviendo los datos en los registros del Shift Register.



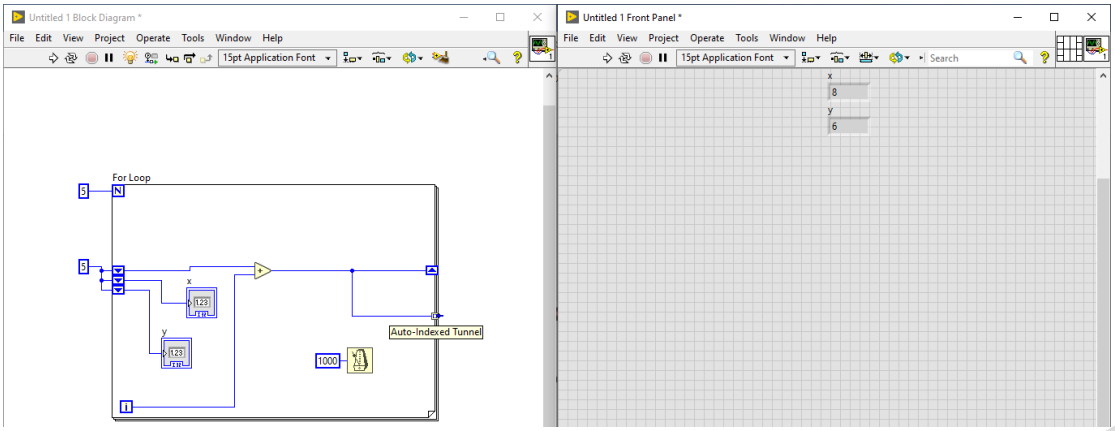
Ejecución del Programa: Movimiento de Datos Shift Register





Podemos observar que los datos se van apilando en los 3 registros de la izquierda cada vez que hay una nueva iteración, ya que se van sumando con los nuevos valores que va adoptando la variable i.

Ahora del resultado de la suma voy a conectar una bifurcación al perímetro del bloque.



Block Diagram - Bucle For - Túnel (index): Sacar Información del Bucle en un Vector

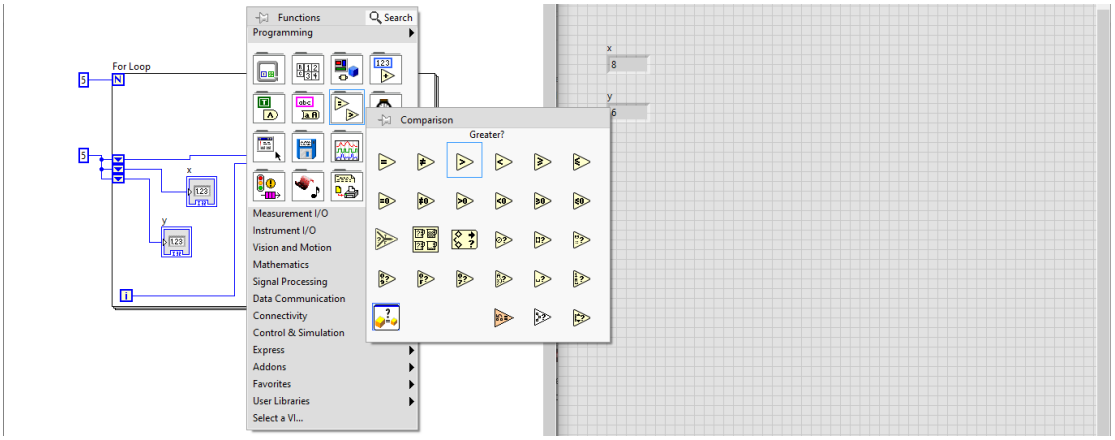
Un túnel es una parte de la pared del bucle que sirve sacar valores de su estructura y que se puedan utilizar afuera, en las demás partes del programa.

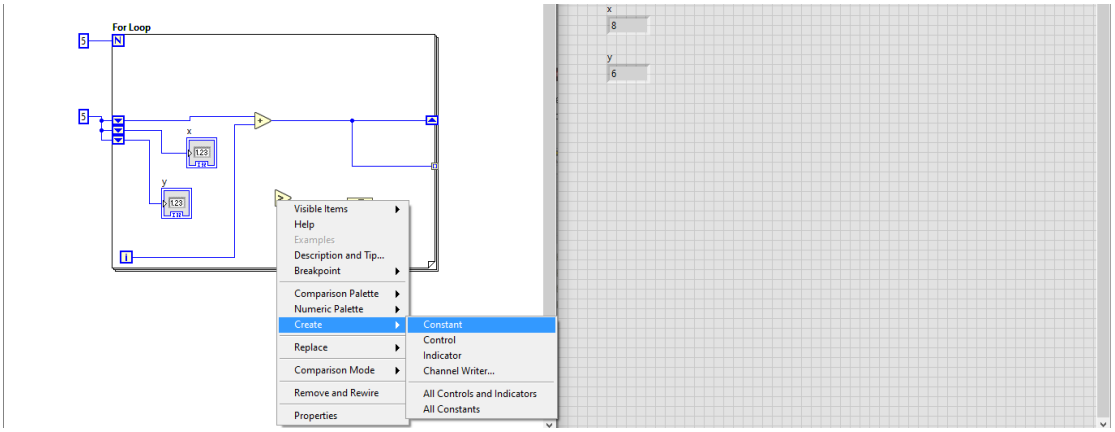
Por medio de un túnel del bucle for puedo obtener un escalar que tenga el último valor que se generó del bloque de números aleatorios para sacarlo y mostrarlo.

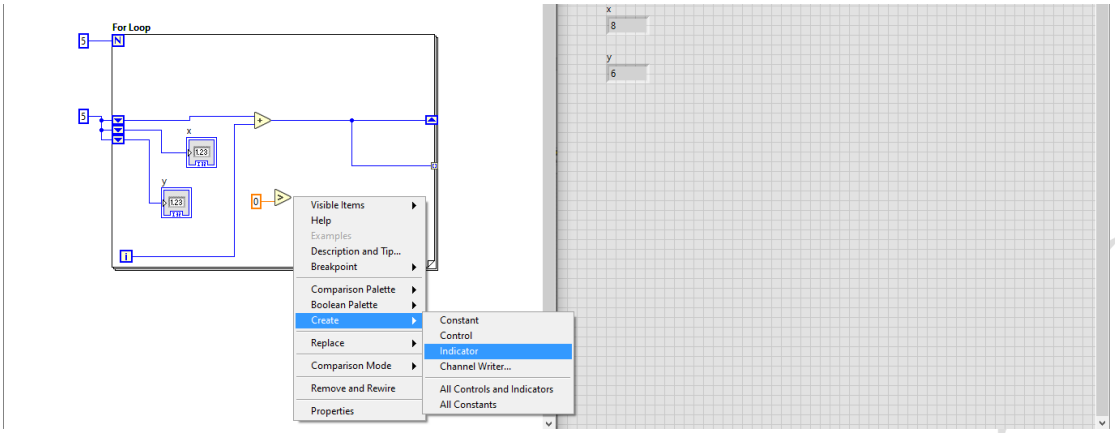
Con la opción de indexing aplicada al túnel, estoy generando un vector que almacene lo obtenido del bucle for.

El túnel del ciclo for lo que hará es aparecer indexado, esto implica que lo que vamos a estar generando aquí es un vector, por lo tanto, los datos se van a estar almacenando en un vector que podemos obtener de ese punto.

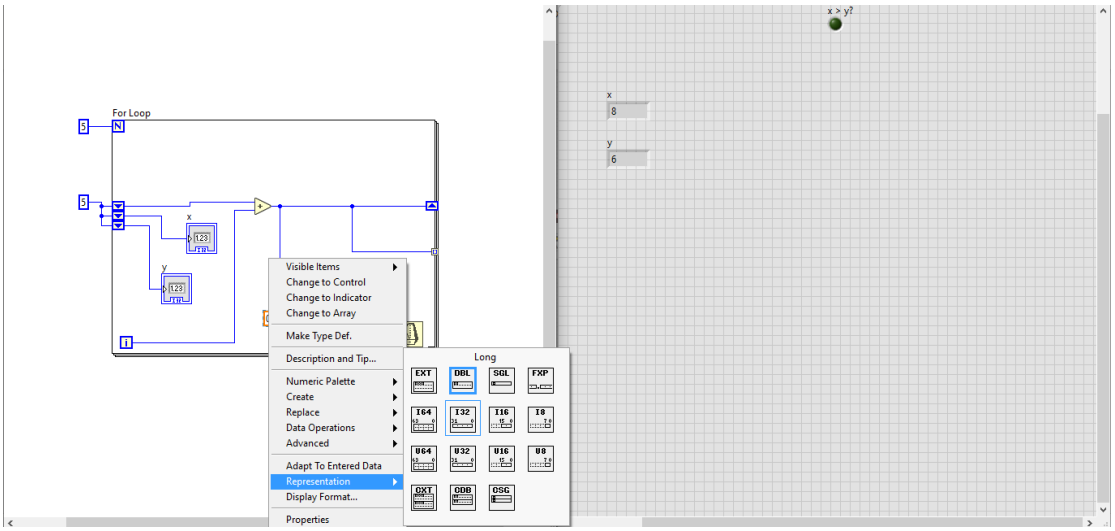
Block Diagram - Greater ?: Operación Lógica Mayor Que (>)



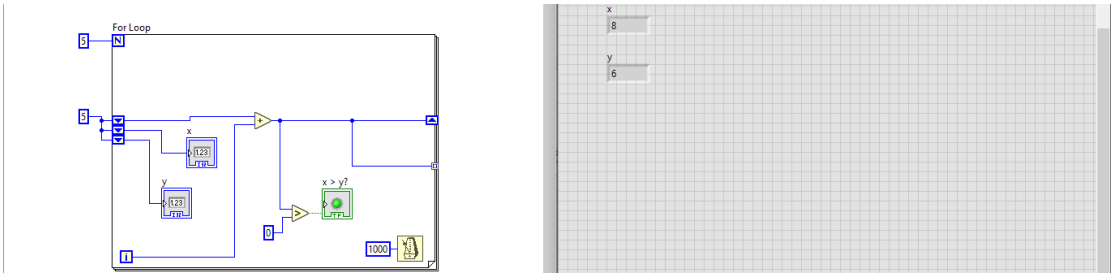




Cambiar el tipo de dato de mi elemento: Clic derecho en el bloque → Representation → Tipo de Dato. Como cree primero la constante, debo cambiar el tipo de dato.

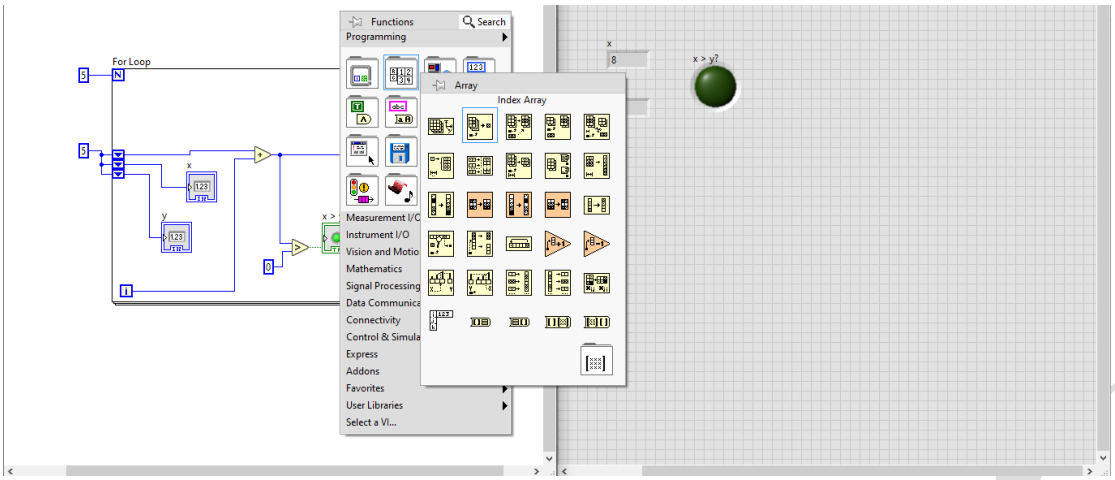


Y ya con esto se quita el coertion dot.

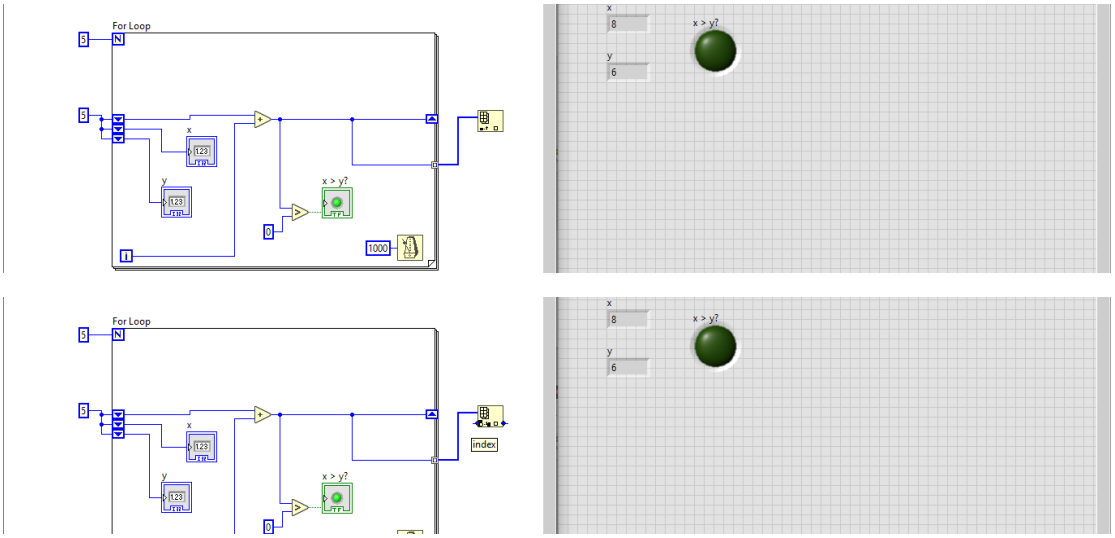


Block Diagram - Index Array: Indicar la Posición de un Array

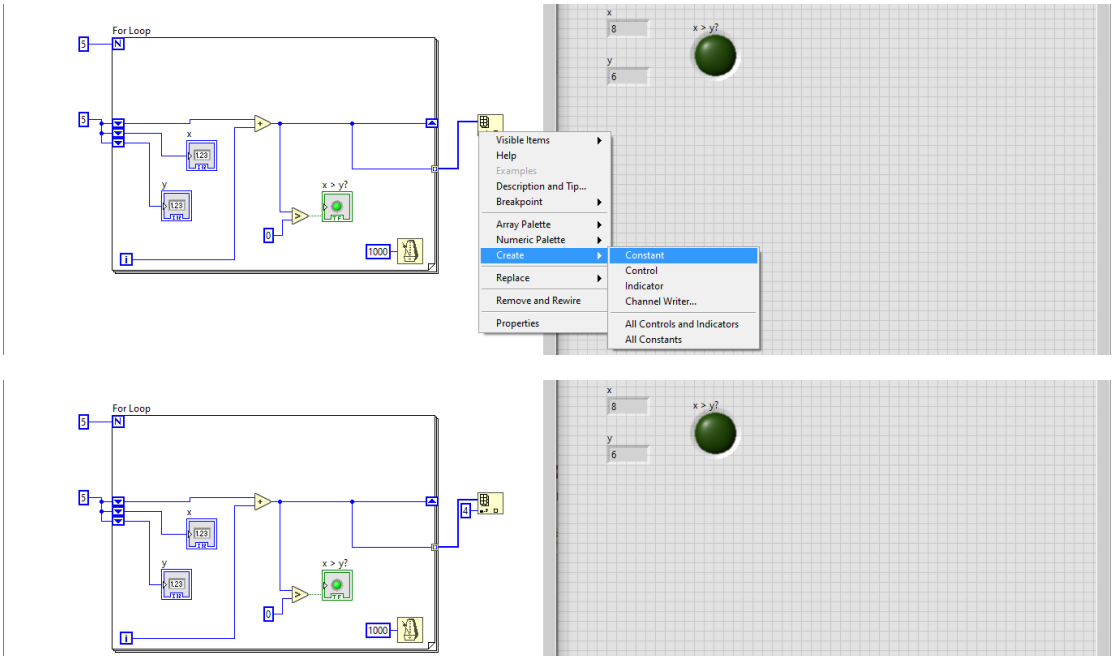
El bloque de Index Array se utiliza cuando se quiere colocar un elemento en cierta posición de un array vacío.



Desde el túnel del ciclo for pondremos el vector de 5 valores fuera del bloque.

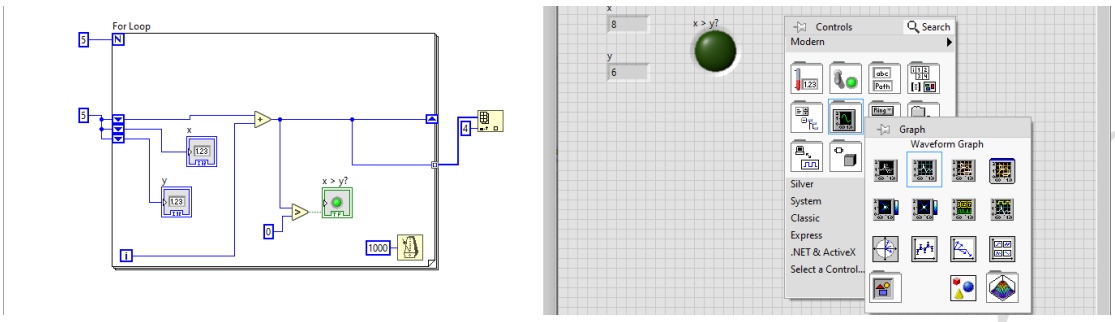


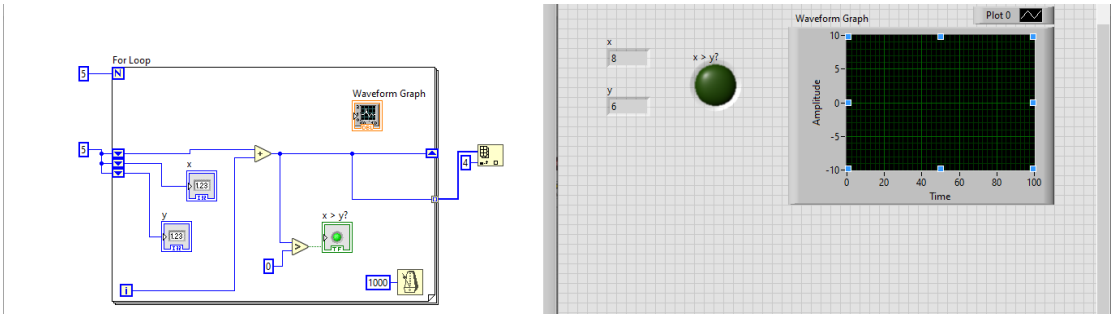
Aquí pondré un 4 porque cuenta de 0 a 4, que son todas las iteraciones del bucle for, en total son 5.



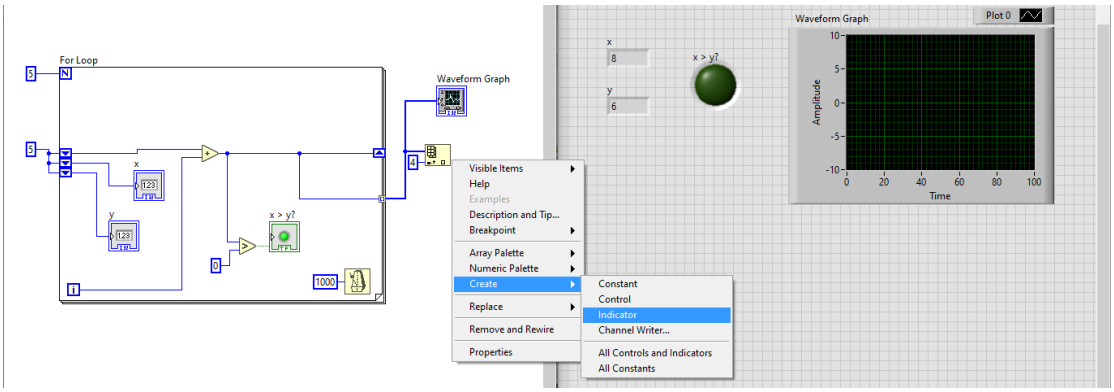
Front Panel - Waveform Graph: Ventana que Muestra una Señal (Array)

El Waveform Graph muestra gráficas de tipo Array, las cuales son cualquier tipo de grupos de números.

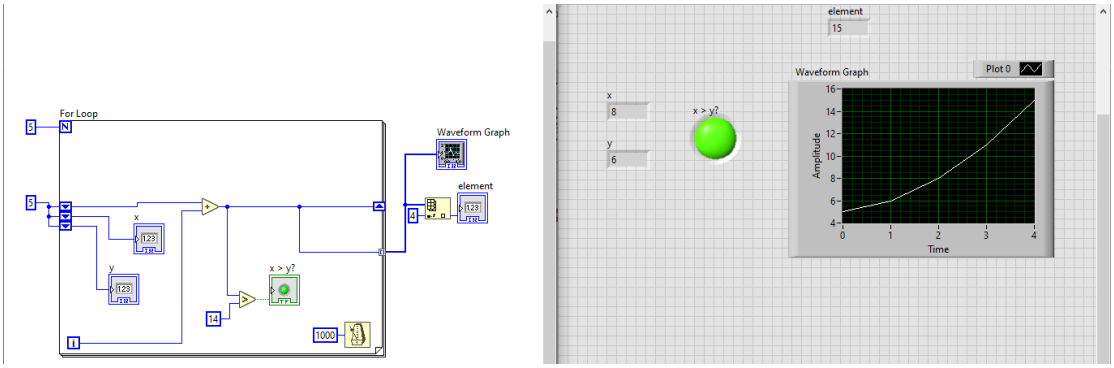


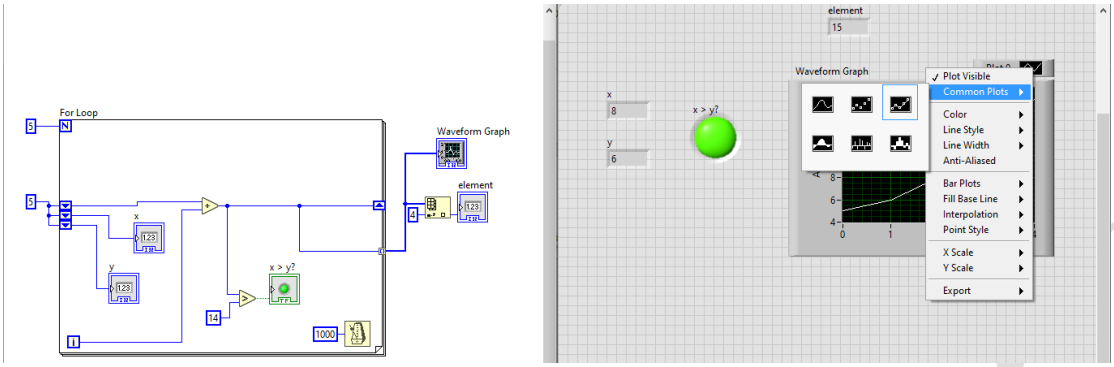


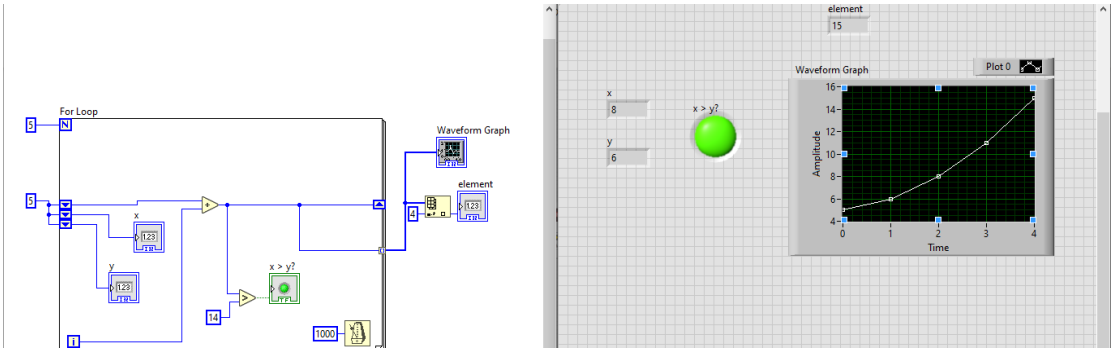
Se colocó un Waveform Graph porque no es necesario que se esté graficando el mismo vector dinámicamente, sino que cada que le dé STOP al programa, se graficará como fue cambiando el valor del vector.



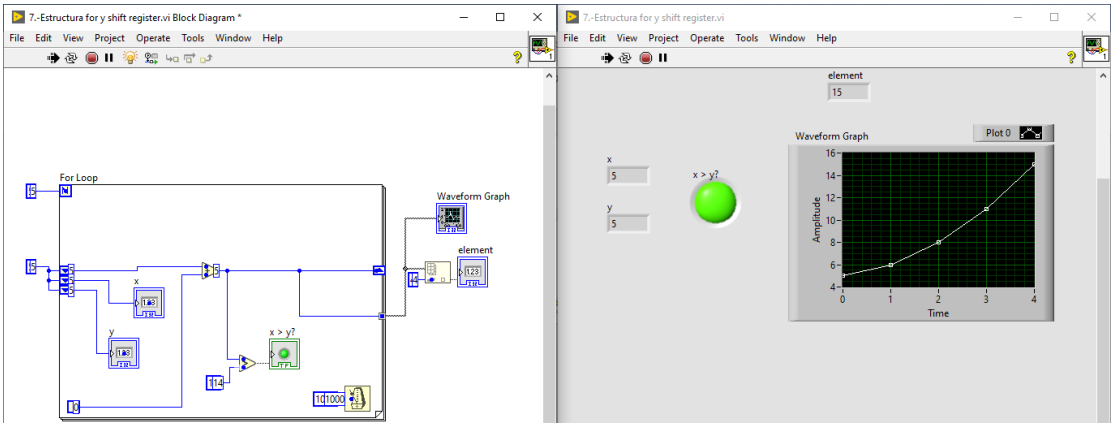
La gráfica no se mostrará hasta que deje de correr el programa.



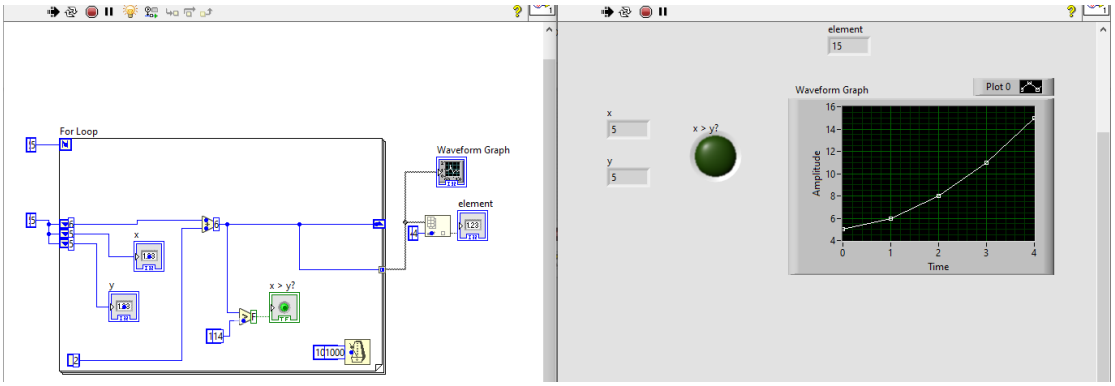


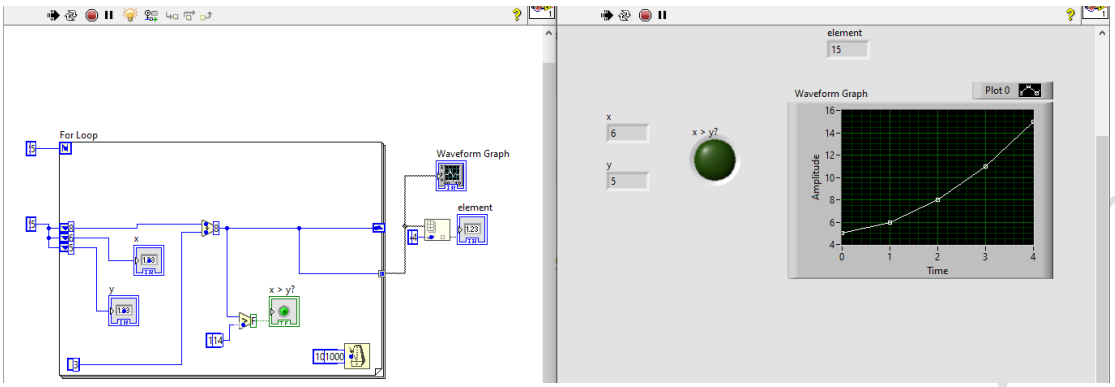


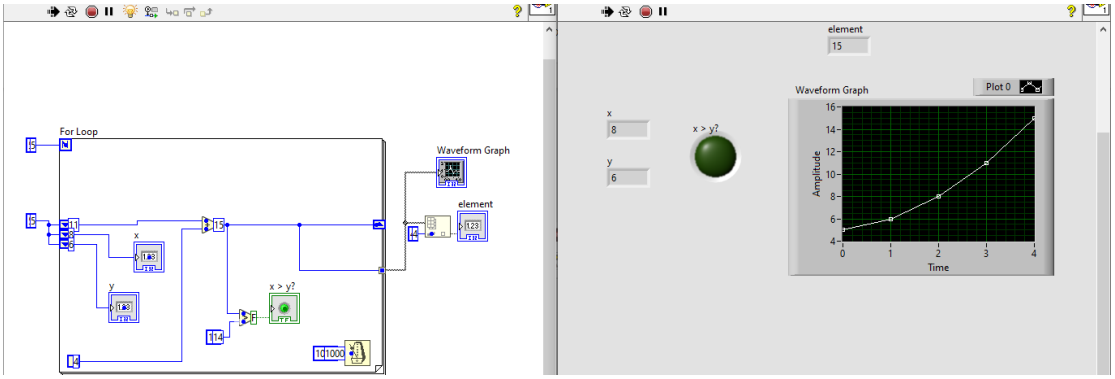
Ejecución del Programa: Waveform Graph (Arrays) de un Shift Register

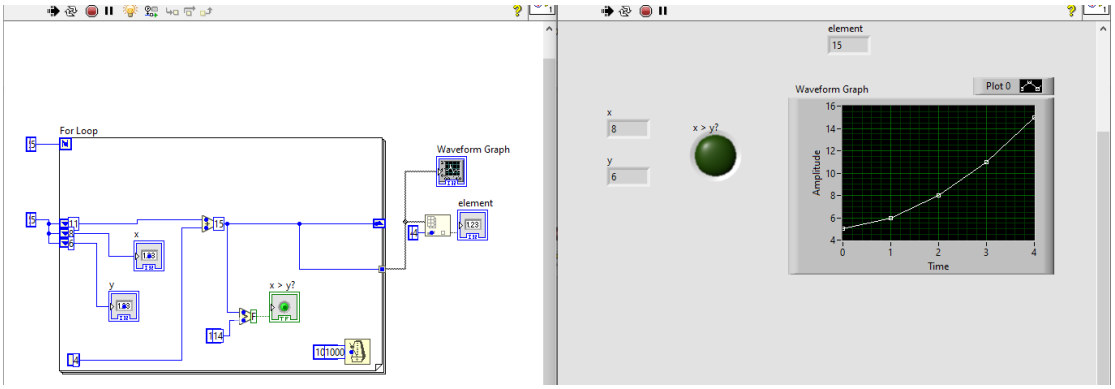


La gráfica no se actualizará hasta que detenga el programa y debo borrar su historial manualmente.

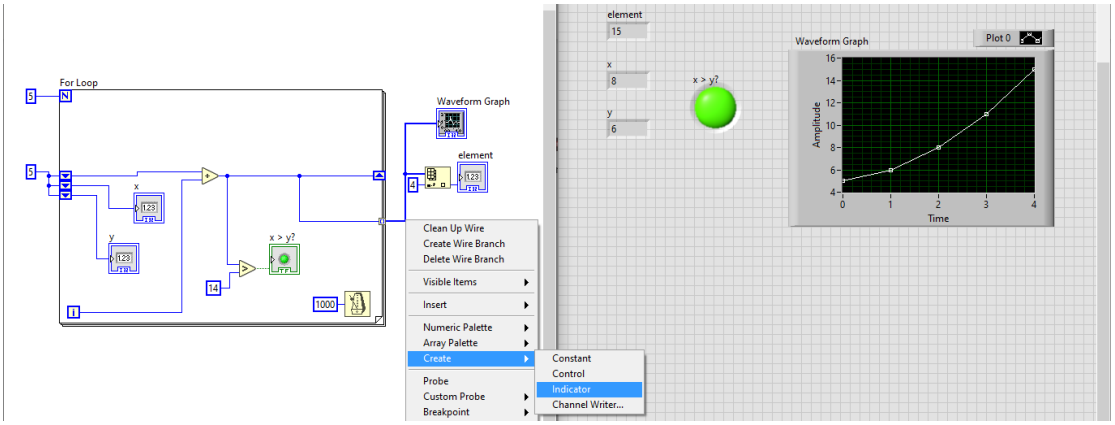


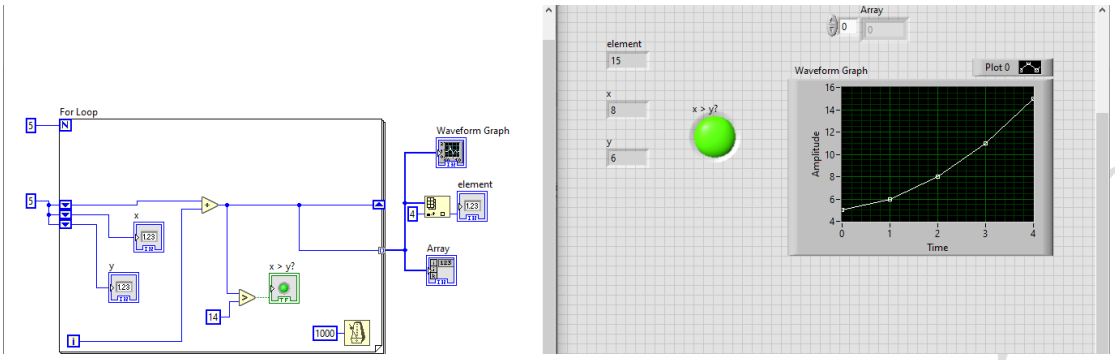




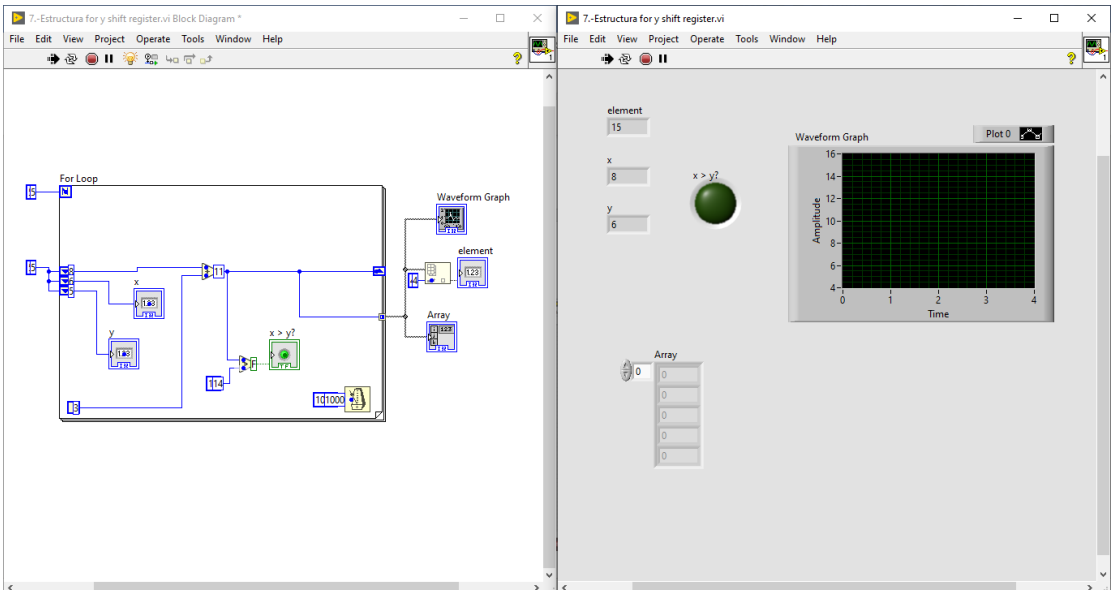


Se le puso 14 como límite del comparador porque el máximo es el 15 después de la suma de las 5 iteraciones. Ahora para que el array indexado que está saliendo del ciclo for pueda ser visible, vamos a agregar un indicador, porque en sí el punto del array es que se puedan observar sus valores.





Ejecución del Programa: Waveform Graph de un Shift Register y su Array



Tanto los valores del Array como el resultado de la gráfica se verán cuando detenga la ejecución del programa.