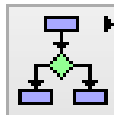


## DIAGRAMAS DE FLUJO (guía rápida)



Para la versión 3.0.8.3 o superiores

*Prof. José Manuel Ruiz Gutiérrez*  
*[j.m.r.gutierrez@gmail.com](mailto:j.m.r.gutierrez@gmail.com)*

## Índice

- 1. Justificación de esta librería.**
- 2. Como usar los componentes de la librería “Diagramas de Flujo” (FlowChart).**
- 3. Bloque de función “Start”**
- 4. Bloque de función “Nudo Unión”**
- 5. Bloque de función “Decisión”: Sintaxis**
- 6. Bloque de Función “Evaluar Expresión”**
- 7. Bloque de función “Incrementar”**
- 8. Bloque de función “Decrementar”**
- 9. Bloque de función “Retardo”**
- 10. Bloque de función “Escribir”**
- 11. Bloque de función “Leer”**
- 12. Bloque de función “Output”**
- 13. Bloque de función “Procedure –procedimiento o rutina-“**
- 14. Bloque de Función “Return”**
- 15. Bloque de función “Entrega Dato”**
- 16. Bloque de función “Recoge Dato”**
- 17. Realización de “Trazado” de un Diagrama de flujo.**
- 18. Ejemplos de utilización de esta librería.**

## 1. Justificación de esta librería.

La incorporación de esta librería en la herramienta MyOpenLab le da un valor importante ya que le permite diseñar simulaciones orientadas al control de procesos a través de un lenguaje muy extendido en los sistemas reales de programación de PLC's, sistemas de control de procesos, microcontroladores (PIC's), etc...

La mayoría de los procesos de control se pueden diseñar haciendo uso de bloques funcionales dispuestos en forma de “diagrama de flujo” haciendo de esta manera mucho más fácil la comprensión de su funcionamiento y la detección de errores.

MyOpenLab incorpora esta librería totalmente compatible al resto de elementos de las demás librerías y con la finalidad, quizá en futuro, de poder asociar las aplicaciones realizadas con sus bloques funcionales a tarjetas de adquisición de datos conectadas al puerto del PC , de la misma manera que lo hacen otras aplicaciones similares a esta.

Las posibilidades didácticas que se añaden con esta librería son muy importantes, ya que el usuario apenas tendrá que tener conocimientos de electrónica o electricidad para ser capaz de realizar simulaciones con éxito.

### Nota muy importante:

Modificaciones importantes en la sintaxis para la nueva librería Flowchart

Antes	Ahora
=	==
Valor tipo dbl 10	Valor tipo dbl 10.00
sin(x)/cos(x)/...	Math.sin(x)/Math.cos(x)/..
Operador AND &	Operador AND &&
Operador OR	Operador OR

## 2. Como usar los componentes de la librería

### “Diagramas de Flujo” (FlowChart).

La librería FlowChart esta formado por los componentes que se muestran en la figura 1

#### Librerías Panel Circuito

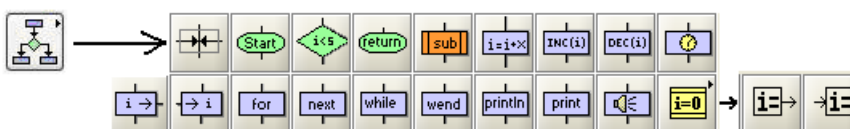
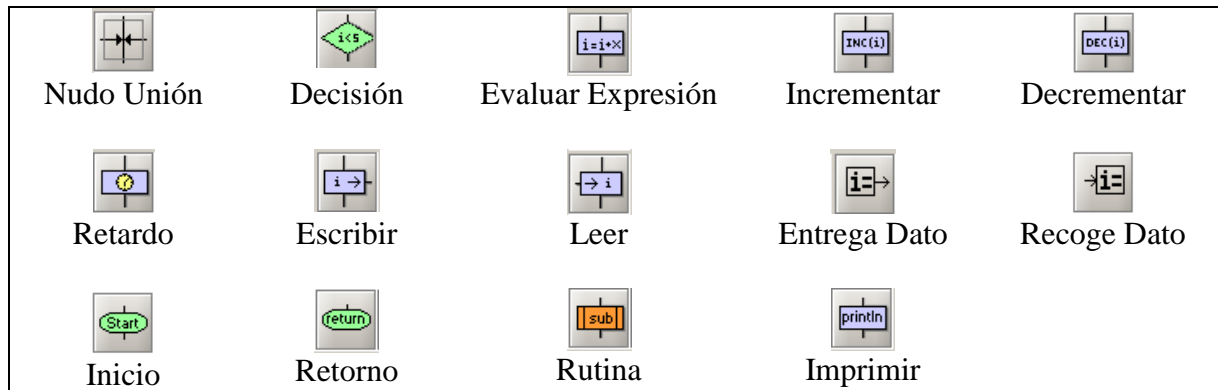


Figura1

En la tabla siguiente colocamos el dibujo y el nombre de cada uno de los bloques.




Los pasos a seguir para realizar una aplicación con esta librería son:

## 2.1. Definir las variables que se manipularán en el proceso.

Este paso es el primero y consiste sencillamente en definir las distintas variables que vamos a manejar. Los tipos de variables para esta versión de librería son tres:

**Boolean (Binaria)**  
**Double (Decimal)**  
**String (cadena de caracteres)**

Para llevar a cabo la definición se hace mediante la herramienta  “Define Variables” que se puede invocar desde la barra de botones del menú o bien desde el menú desplegable (figura 2):

*VM-> Define variables*

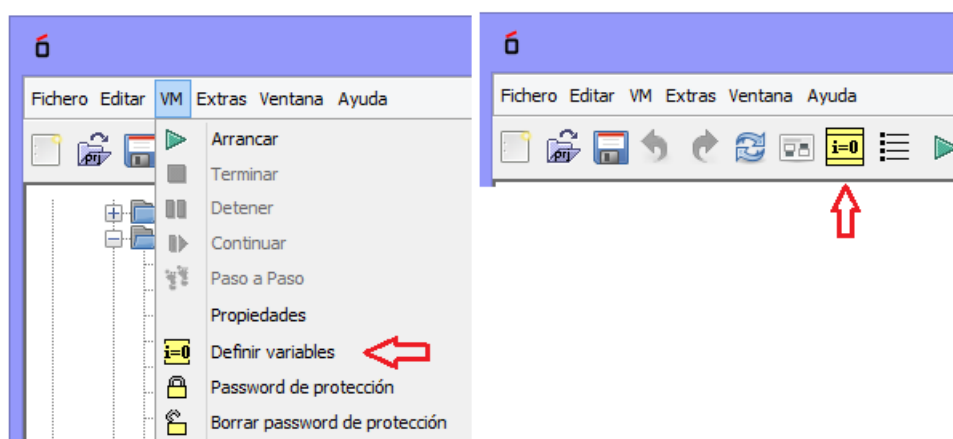


Figura 2

Esta opción, una vez invocada abre una ventana como la mostrada en la figura 3 a través de la cual podemos definir y/o editar las variables.

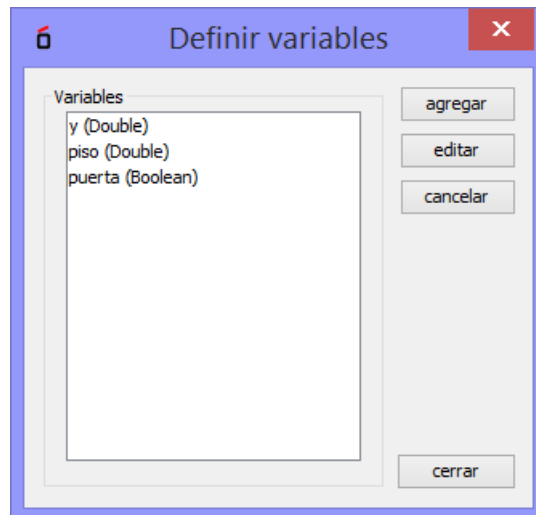


Figura 3

Mediante el botón Add añadimos una nueva variable a través de la ventana de la figura 4

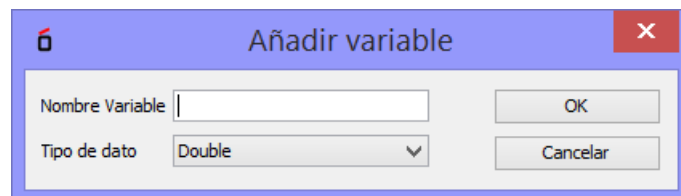


Figura 4

Pulsando sobre el botón de menú “Tipo de dato” seleccionamos el tipo de dato, tal como se ve en la figura 5

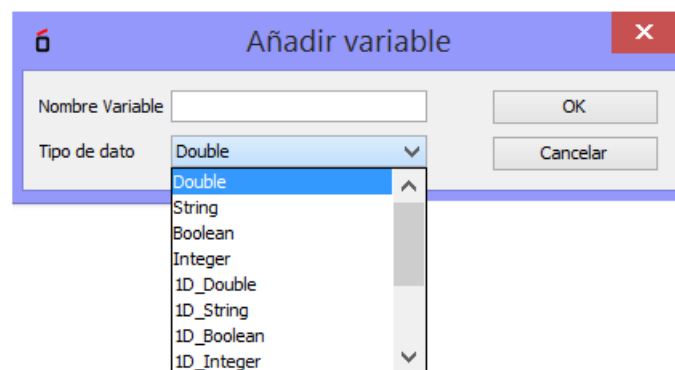


Figura 5

## 2.2. Realizar el “diagrama de flujo”

Una vez definidas las variables procederemos a realizar el “diagrama de flujo” arrastrando del área de la librería los componentes y situándolos en el “Panel de Circuito”. Se recomienda colocar (figura 6):

- (1) Los Bloques y la configuración de los valores de cada uno.
- (2) Unir los Bloques entre si
- (3) Colocar los elementos externos (botón “run” y caja de “entrada de constante” –valor -i-. Bloque “Entrega Dato” y bloque “Salida Numérica”

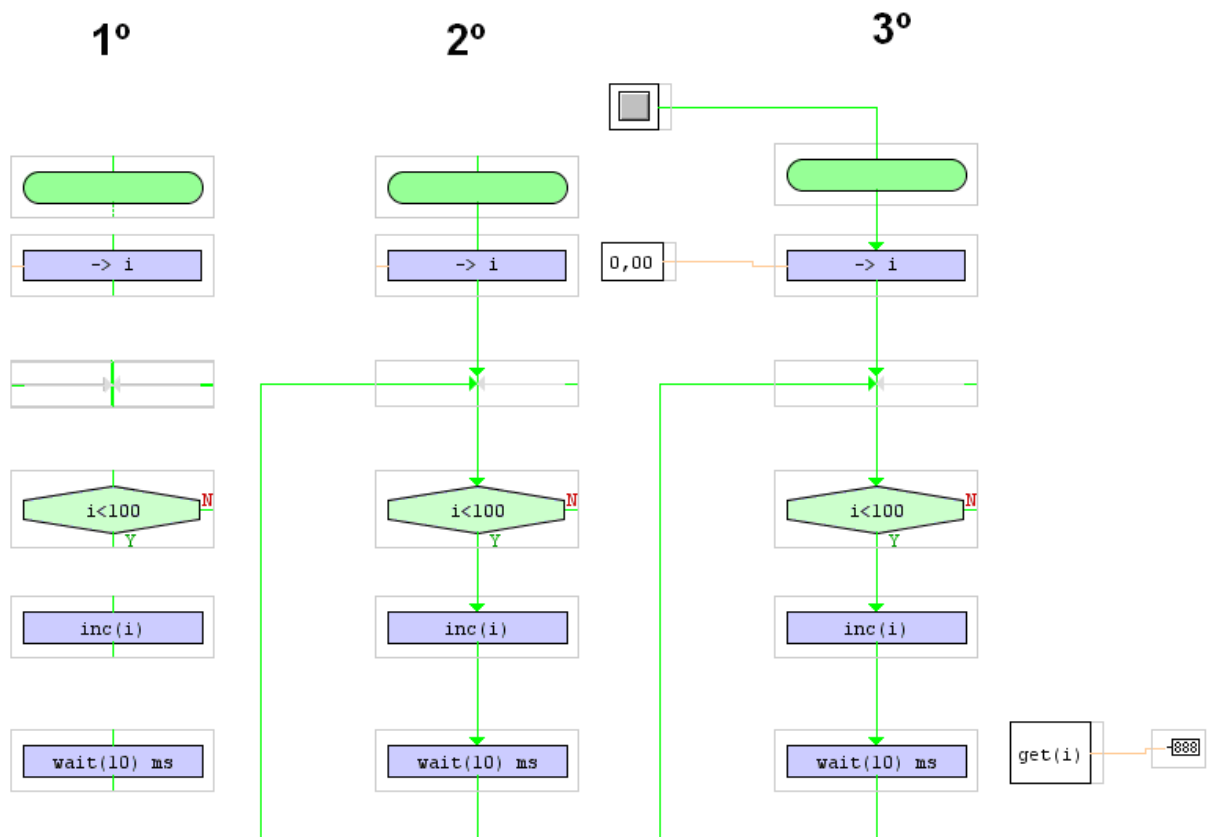


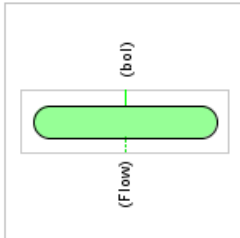
Figura 6

Debemos tener en cuenta que en todos los Diagramas de Flujo se debe colocar una señal de tipo booleano que es la que activa la ejecución del organigrama. En la figura anterior vemos que hemos colocado para este fin un botón.

Los bloques que vemos en la figura 6 tienen unos parámetros de configuración que enumeramos a continuación:

Bloque “ <b>Start</b> ”	se coloca en el comienzo de todos los diagramas de flujo y en las subrutinas
Bloque “ <b>Leer</b> ”	se debe asignar el nombre de la variable que va a leer ( <i>i</i> )
Bloque “ <b>Nudo Unión</b> ”	este bloque no tiene parámetros.
Bloque “ <b>Decisión</b> ”	se debe colocar la expresión condicional del bloque $i > 100$
Bloque “ <b>Incrementar</b> ”	se asigna la variable <i>i</i> como variable a incrementar.
Bloque “ <b>Retardo</b> ”	se asigna el tiempo de retardo que queremos detener la ejecución del organigrama una vez que estemos simulando su funcionamiento

### 3. Bloque de función “Start (Inicio)”



El elemento Start es necesario para iniciar el diagrama de flujo y tiene que colocarse siempre al principio de éste.

Este elemento representa una Función o Procedimiento que se debe usar siempre también para el inicio de una subrutina.

Este bloque en su entrada debe ser activado mediante una señal de tipo Booleano (bol) y su salida es de tipo (Flow) que es el tipo de dato que se utiliza para los diagramas de flujo.

Es muy importante tener en cuenta este detalle dado que la conexión del diagrama con el resto de objetos de nuestro modelo se debe hacer atendiendo a este concepto.

Figura 7

En el caso de que este elemento forme parte del comienzo de una Subrutina o Procedimiento se pueden definir como variable de entrada (a efectos de que recojan valores de otras partes del diagrama de flujo) una serie de datos.

**Pongamos un ejemplo.** Supongamos que queremos definir una subrutina llamada *Test* que maneje las variables *a* (double) variable *b* (string) y variable *c* (boolean). En este caso se escribiría como parámetros del elemento Start el texto: *Test(double a, string b, boolean c)*

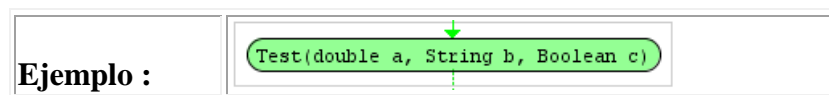


Figura 8

*Test* es el nombre del "Procedimiento/ función"

Todos los parámetros son variables de tipo "local" que pueden ser usadas en el diagrama de flujo.

Se puede usar el elemento Start como una "función" que devuelve un resultado.

Colocando simplemente al final una función "Return"

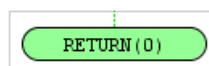


Figura 9

### 3.1. Vemos a continuación un sencillo ejemplo con la función Start.

Se trata de realizar la impresión de unos datos en el terminal haciendo uso de la función “println”.

El programa se ejecuta pulsando el botón *OK*.

La primera orden que recibe el sistema es ejecutar la subrutina *Test*, esto se realiza con la función “Procedure” en la que definimos

$$y = \text{Test}(5,3)$$

En este caso *y* será la variable en la que se recogerá el valor que devuelva la ejecución del Procedimiento *Test*.

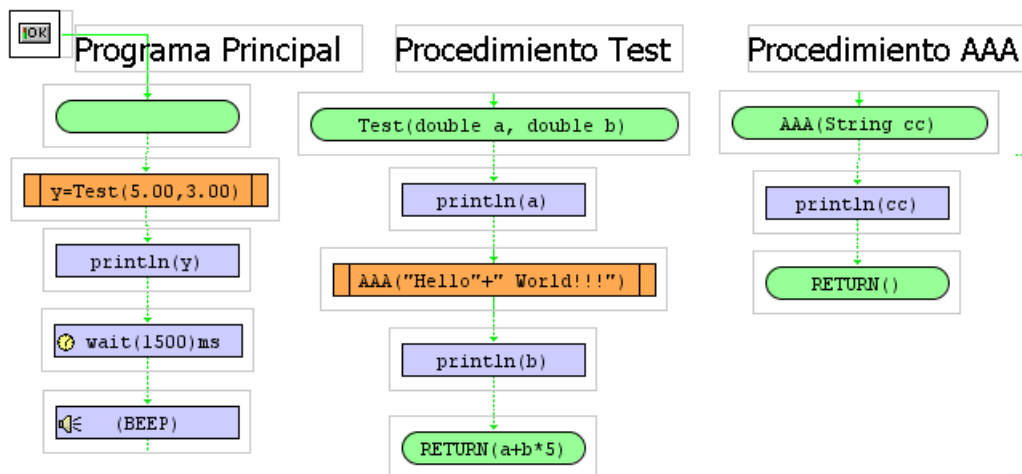


Figura 10

El programa principal a través de la función “Procedure” *Test* entrega los valores

*a=5* y *b=3* al procedimiento *Test*

La función “Start” con la que necesariamente debe comenzar este procedimiento aparece con el texto

*Test(double a, double b)*

en donde se especifica que variables se recogen (*a* y *b*).

El procedimiento acaba con la instrucción “*RETURN(a+b\*5)*” que devuelve el valor a la instrucción que lo invocó `y=Test(5,3)`.

#### Ejecución del procedimiento *Test*

Una vez iniciado el procedimiento *Test* lo primero que se hace ordena es la impresión del valor de la variable *a* “*println a*” que se lleva a cabo en el Terminal de Salida (ver figura ...)



La siguiente operaciones una llamada a un segundo procedimiento o subrutina:

*AAA("Hello"+"world !!")*

En este caso mediante esta instrucción de tipo "Procedure" se llama a la rutina AAA y a la vez se pasa un parámetro de tipo string (*Hello world !!!*)

### **Ejecución del procedimiento AAA**

El procedimiento AAA comienza con su instrucción "Start" recogiendo el string que se le envía y asociándolo a la variable *cc*.

*"AA(string cc)"*

La siguiente instrucción será imprimir en el terminal el valor de *cc* y esto se hace mediante:

*"println(cc)"*

Realizada esta operación termina con RETURN el procedimiento AAA devolviendo el control de la ejecución a la instrucción

*"println(b)"*

perteneciente al procedimiento Test

La siguiente instrucción que se ejecutará es

*RETURN(a+b\*5)*

Una vez ejecutado el procedimiento Test el control del programa pasa al programa principal.

A continuación se imprime en el terminal el valor *y*

*"println(y)"*

A continuación se produce una espera de 1500 ms. Mediante la instrucción

*"wait (1500 ms)"*

Finalmente se ejecuta la instrucción BEEP y se acaba el programa

*"BEEP"*

En la figura vemos el aspecto de la ventana del terminal una vez acabado el programa.

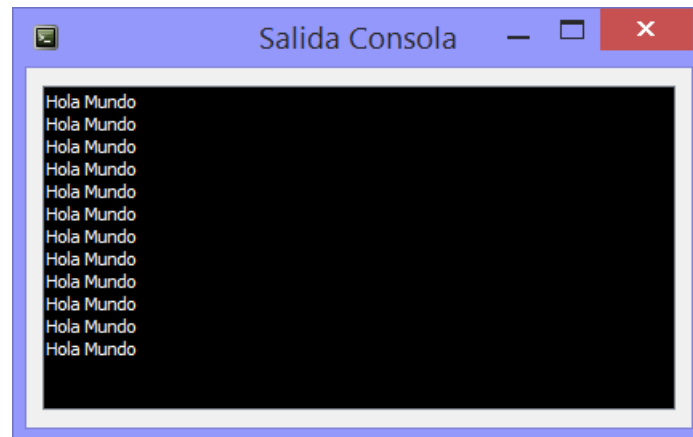


Figura 11

Para estudiar mas detenidamente el flujo de ejecución del programa se puede recurrir a la opción de trazado (Modo Debug).

#### 4. Bloque de función “Nudo Unión”



Este bloque sirve para realizar la conexión de hasta cuatro líneas dentro de un diagrama de flujo. Todas las señales que confluyen en el son de tipo flow



Figura 12.

#### 5. Bloque de función “Decisión”: Sintaxis

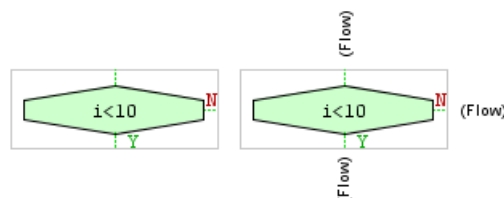


Figura 13

Se trata de un bloque condicional que activa una de sus dos salidas (SI "Y"o NO "N") en función de si se cumple una condición establecida.

La condición se puede asignar de acuerdo con la siguiente sintaxis.

<	menor
<=	menor o igual
>	mayor
>=	mayor o igual
==	igual
<>	distinto

En la figura 14 vemos una aplicación típica en donde aparece un bloque condicional en el que se pregunta por el valor de la variable “i” y se muestra su valor a la vez que se incremente y se temporiza mientras este valor sea menor que 10 “i<10” cuando deje de serlo dejará de ejecutarse la secuencia de visualizar-incrementar-temporizar.

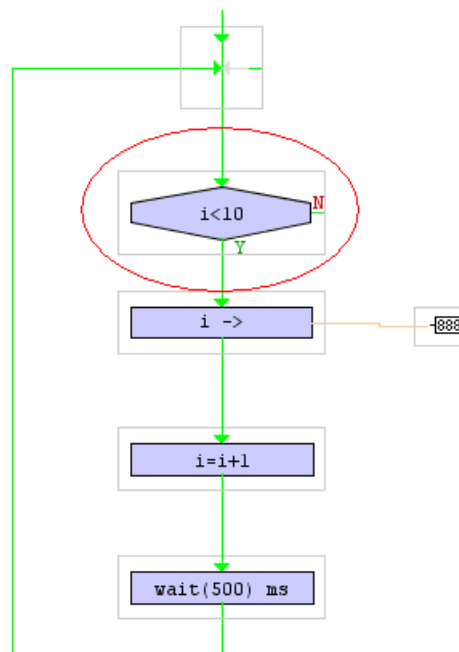
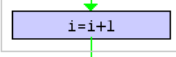
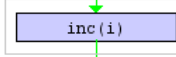


Figura 14

Es importante observar como en este caso para incrementar no se ha hecho uso de la función “Incrementar” sino que se hace simplemente con la función “Evaluar Expresión”, es decir

podríamos sustituir el bloque  por el bloque .

## 6. Bloque de Función “Evaluar Expresión”

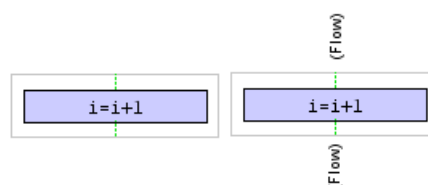


Figura 15

Con este bloque se pueden evaluar expresiones matemáticas en las que figuren variables previamente definidas sometidas a los operadores que se anotan a continuación. Este componente admite variables del tipo: **Boolean, Double y String**

Debe cuidarse en la elaboración de un organigrama el no mezclar tipos distintos de variables.

Los operadores que se pueden utilizar en este bloque de función son:

### Operaciones Booleanas:

&&	Producto lógico AND
	Suma lógica OR
!	Negación NOT

### Ejemplo:

A=TRUE, B=FALSE

$Y = (!A) \&\& B = ((\text{NOT TRUE}) \text{ AND FALSE}) = \text{FALSE AND FALSE}$

Resultado : Y= FALSE

### Operaciones con números tipo (double) "con decimales"

%	Modulo
^	Potencia ( $x^y$ )
+	Suma
-	Resta
*	Multipliación
/	División

### Ejemplo :

$y = x + z * 4 - 1;$

Funciones numéricas

<b>Math.sin(x)</b>	<b>Seno</b>
<b>Math.cos(x)</b>	<b>Coseno</b>
<b>Math.tan(x)</b>	<b>Tangente</b>
<b>Math.asin(x)</b>	<b>Arco Seno</b>
<b>Math.acos(x)</b>	<b>Arco Coseno</b>
<b>Math.atan(x)</b>	<b>Arco Tangente</b>
<b>Math.asinh(x)</b>	<b>Seno Hiperbólico</b>
<b>Math.cosh(x)</b>	<b>Coseno Hyperbolico</b>
<b>Math.tanh(x)</b>	<b>Tangente Hiperbólica</b>
<b>Math.abs(x)</b>	<b>Valor Absoluto</b>
<b>Math.log(x)</b>	<b>Logaritmo (decimal)</b>
<b>Math.ln(x)</b>	<b>Logaritmo natural (neperiano)</b>
<b>Math.exp(x)</b>	<b>Exponente</b>
<b>Math.sqrt(x)</b>	<b>Raiz</b>
<b>Math.round(x)</b>	<b>Aleatorio</b>
<b>Math.fak(x)</b>	<b>Factorial</b>
<b>Math.toDeg(x)</b>	<b>Convierte radianes a grados</b>
<b>Math.toRad(x)</b>	<b>Convierte grados a radianes</b>

Todas las funcione llevan la partícula Math por ejemplo: **Math.sin(x)**

### Operaciones con Cadenas de caracteres (string)

usar las cadenas con comillas :“Cadena”

+	<b>Suma de cadenas</b>  <b>Ejemplo : “Hola _” + “mundo”</b>  <b>Resultado= “Hola _mundo”</b>
---	--

## 7. Bloque de función “Incrementar”

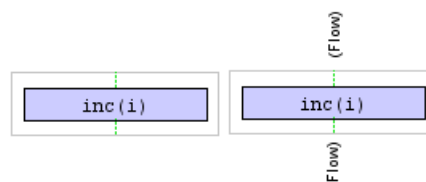


Figura 16

Este bloque permite incrementar una variable que se especifica en el.

Las aplicaciones con este bloque pueden ser muy diversas y entre ellas podemos destacar el contar eventos en un proceso. La variable debe ser de tipo “dbl” (Decimal)

## 8. Bloque de función “Decrementar”

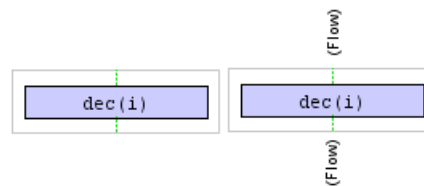


Figura 17

Este bloque hace exactamente lo mismo que el anterior solo que en su caso decrementa la variable a la que esta asociado. La variable debe ser de tipo “dbl” (Decimal)

## 9. Bloque de función “Retardo”

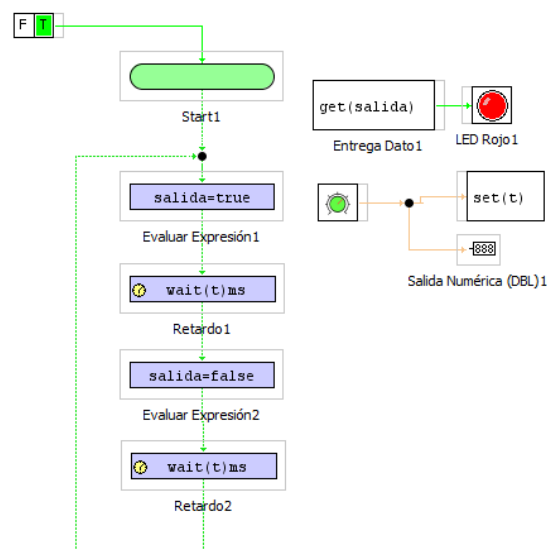


Figura 18

Este bloque introduce un retardo en la ejecución del programa determinado por el valor en ms del parámetro que introducimos.

Se utiliza para generar impulso de duración variable, contar, temporizar, etc.

El valor del tiempo se puede asignar a través de una variable, tal como se muestra en este ejemplo:



## 10. Bloque de función “Escribir”

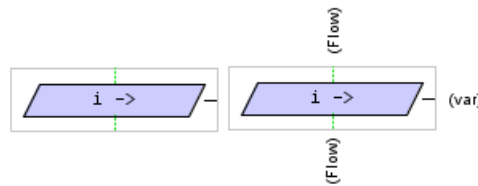


Figura 19

Este bloque permite entregar un valor de una variable tipo bol, dbl o str. al sistema con el fin de poder mostrarla o procesarla en otros bloque de librería.

En la figura 20 vemos como la variable I se deposita en un cuadro de salida numérica.

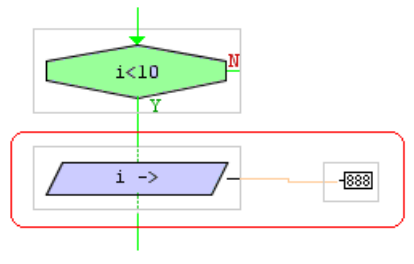


Figura 20

## 11. Bloque de función “Leer”

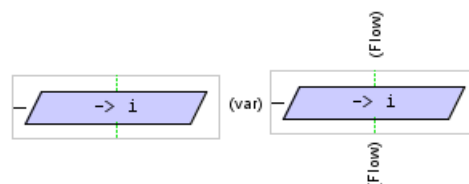


Figura 21

Este bloque permite recoger (leer) un valor de una variable tipo bol, dbl o str. al sistema con el fin de poder hacer uso de ella (procesarla) en el diagrama de flujo.

En la figura 22 vemos como la variable “j” se toma de un bloque “constante”

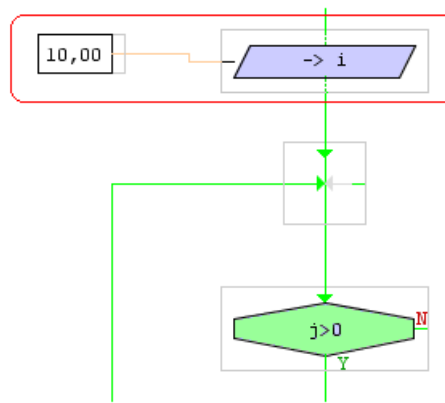


Figura 22

## 12. Bloque de función “Output (escribir dato)”

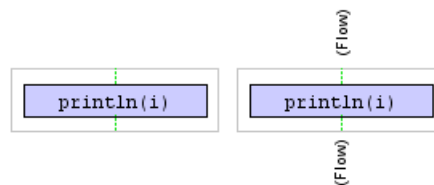


Figura 23

Este elemento imprime el valor de una variable en la ventana se “Salida “.

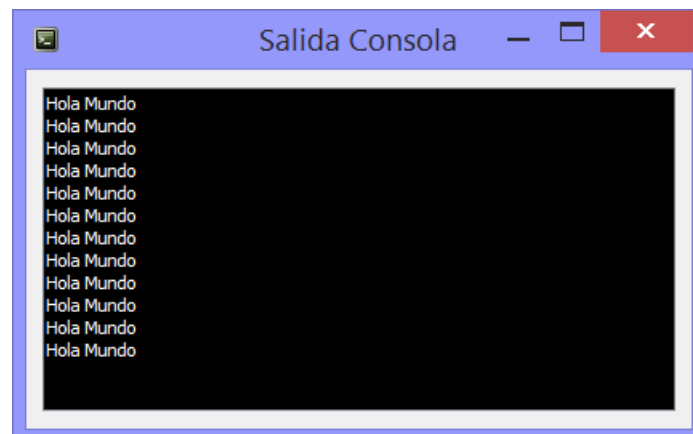


Figura 24

Uso: : Println(<expression>)

Ejemplo : Println("Hello World!"). Println(a) (a = double)



### 13. Bloque de función “Procedimiento(rutina)”

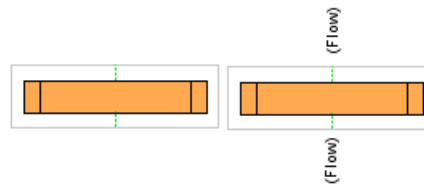


Figura 25

Este elemento invocará siempre a una subrutina o procedimiento que siempre tendrá una estructura que comenzará con una función “Start” y terminará con una función “RETURN”

En este caso la función Start no necesitará ser activada con una entrada de activación (señal de tipo bol) ya que simplemente cuando el nombre que lleve implícito coincida con el nombre de la función “Procedimiento” se activará automáticamente.

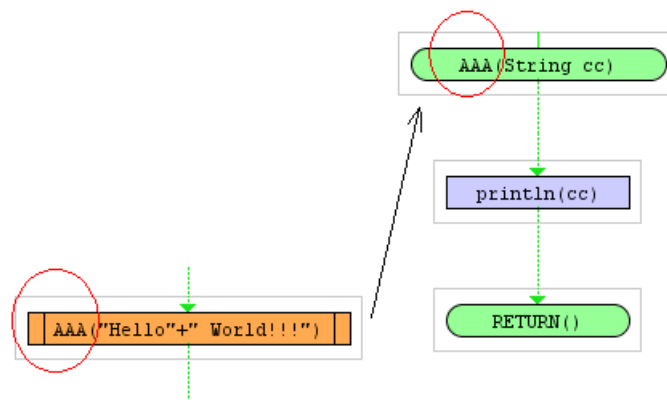


Figura 26

En este caso el procedimiento se llama AAA y el parámetro que pasa es un string asociado a la variable *cc*. Se ve que a la hora de invocar a un procedimiento se pueden entregar a este valores, de la misma manera que el procedimiento una vez ejecutado puede devolver valores tal como se ve en el siguiente ejemplo, en el que el procedimiento pasa los valores 5 y 3 y recoge el valor de *y* que en éste caso es  $y=a+b$ .

La función procedimiento es de una gran importancia y permite realizar numerosas combinaciones a la hora de diseñar un diagrama de flujo.

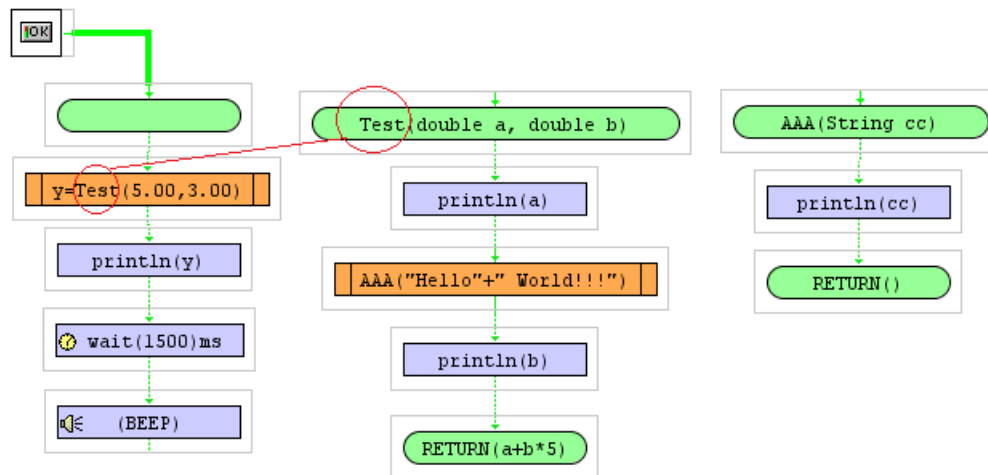


Figura 27

La función “Procedimiento” puede invocar a un procedimiento que no este en el propio diagrama, es decir puede abrir una función que este en otro fichero guardado. Para ello bastará con rellenar el campo vm-Filename del editor de Propiedades de la función. En este campo deberá escribirse el nombre del VM que deseamos abrir. El Filename" deberá contener el nombre del fichero VM (sin la extensión "vlogic")

Ejemplo: sub-vm = subvm.vlogic -> el subvm

Editor de Propiedades	
Nombre	Procedure1
Mostrar Nombre	<input type="checkbox"/>
Izquierda	55
Arriba	155
Anchura	130
Altura	40
vm-Filename	

Figura 28

## 14. Bloque de función “Return (retorno Sub.)”

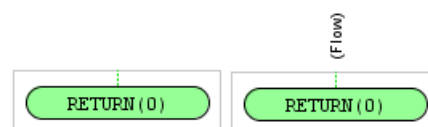


Figura 29

Esta función sirve para cerrar una subrutina o procedimiento.

Puede devolver un valor establecido

En el esquema vemos como el procedimiento Test devuelve a través de la función "Return" el valor de  $a+b*5$  que es el valor de "y=Test(5,3)"

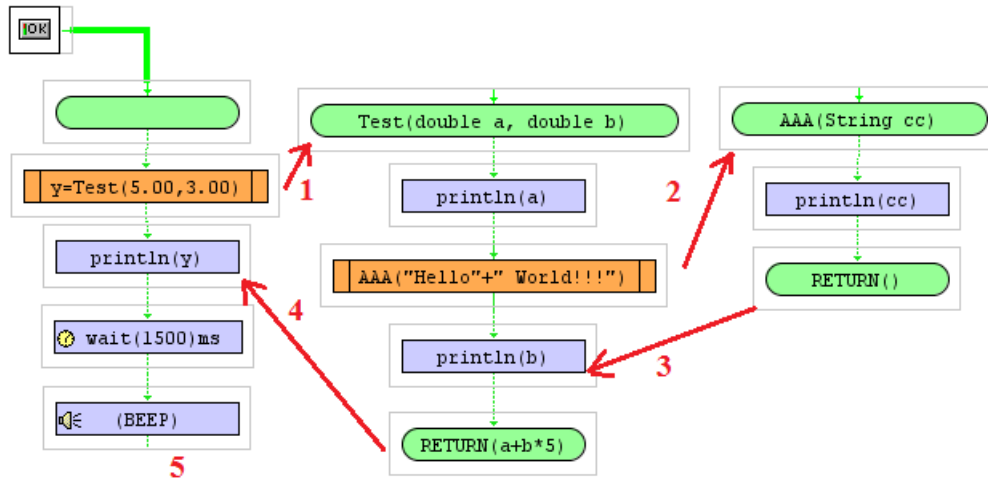


Figura 30

En la figura 30 vemos la secuencia a la hora de ejecutar el diagrama de flujo

## 15. Bloque de función “Entrega Dato”

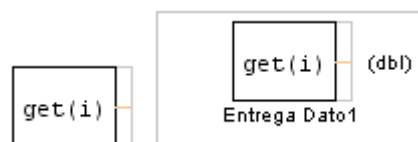


Figura 31

Esta función es parecida a la anterior de “Escribir” ya que lo que hace es enviar un dato de los que hemos definido para trabajar en el diagrama de flujo a cualquier otro bloque del sistema. La diferencia es que mientras que en el bloque “Escribir” solo se lee cuando la secuencia de ejecución pasa por el bloque, aquí se lee en todo momento el valor.

En la figura 32 vemos un sencillo ejemplo en el que se usa esta función para entregar el valor “onx” y sacarlo a un dispositivo de salida digital “led”. La aplicación lo que hace es que la señal booleana “onx” pase de valor 1 a 0 manteniéndose activada 50 ms y manteniéndose apagada 500 ms. Emulando una especie de flash.

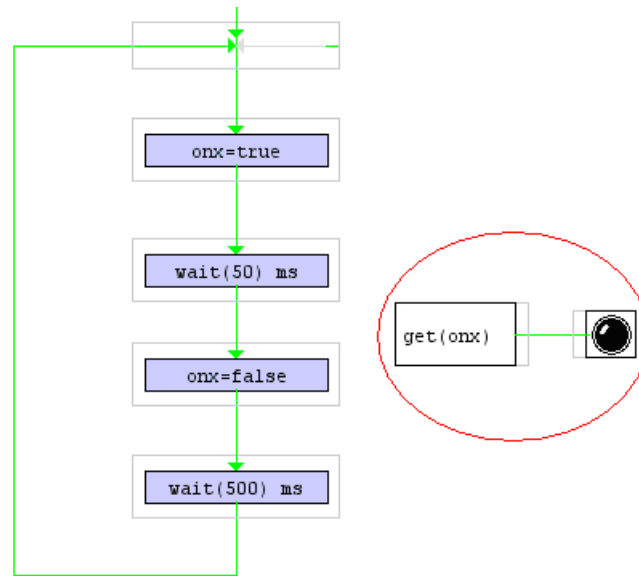


Figura 32

## 16. Bloque de función “Recoge Dato”

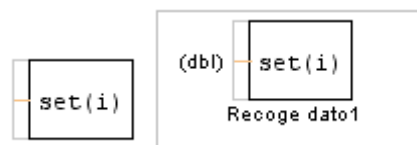


Figura 33

Esta función es parecida a la anterior de “Leer” ya que lo que hace es recoger un dato del sistema y asociarla a una variable del diagrama de flujo. La diferencia con el bloque “Leer” es que la lectura en ese bloque se realiza solo cuando la secuencia de ejecución pasa por el bloque mientras que en este se realiza en todo momento.

En la figura 34 vemos un ejemplo en el que las variables “a” y “b” se toman de las barras de control a y b y se utilizan para ser procesadas por el ejemplo sencillo de un diagrama de flujo en el que se compara el valor de ambos

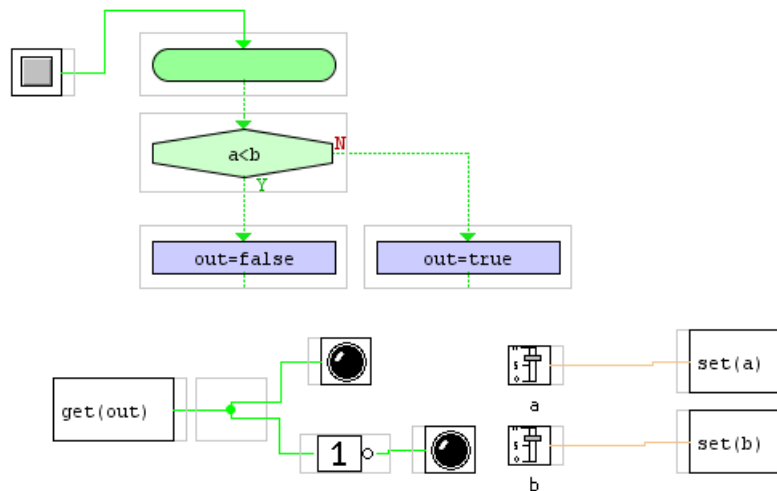



Figura 34

## 17. Realización del “trazado” de un diagrama de flujo.

MyOpen Lab nos ofrece la posibilidad de realizar el trazado de en la ejecución de un diagrama de flujo, permitiendo en este caso ver como evoluciona la información a lo largo de la aplicación.

Para realizar el trazado de un diagrama contamos con la opción  de la barra de botones. Esta opción lanza la aplicación y la va ejecutando de manera lenta. La velocidad de ejecución se puede variar mediante la barra de deslizamiento que aparece en la figura 35

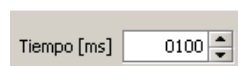


Figura 35

A medida que se va realizando la ejecución se marcan en color más intenso los bloque de función que se van activando, tal como se muestra en la figura 36

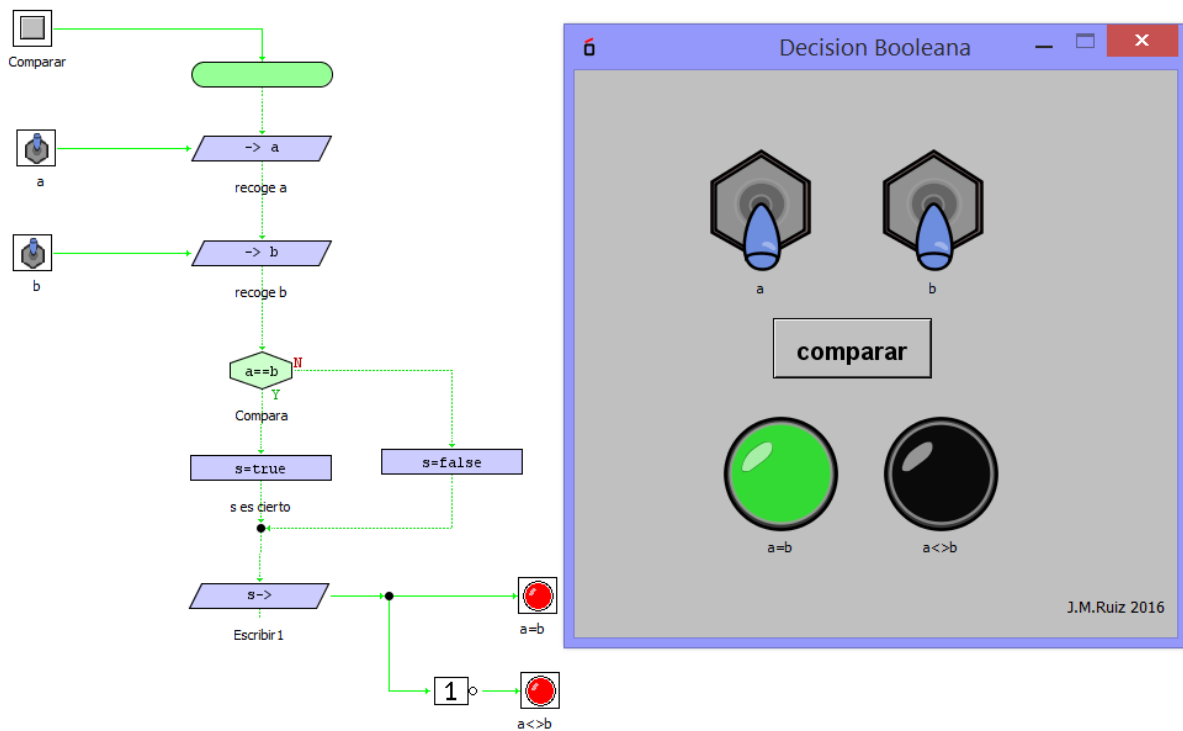


Figura 36

## 18. Ejemplos de utilización de la librería FlowChart

A continuación vamos a comentar algunos ejemplos básicos en los que se utiliza esta librería. No debemos olvidar que con ello se facilita mucho la confección de pequeños automatismos secuenciales además de que se puede ejercitar el uso de algoritmos.


### 18.1. EJEMPLO1

#### EVALUACION DE VARIABLES DE TIPO BOOLEANO (bool)

Evaluación de dos variables de tipo booleano mediante un elemento de salto condicional denominado **“Decisión”**. Ver figura 23

Vemos que las variables se han definido como **i** y **j** y se recogen mediante dos instrucciones del tipo “Leer”. El bloque de decisión evalúa la condición **i=j** y a sus salidas directamente hemos colocado dos dispositivos de visualización de valor digital **“Leds”**.

El resultado es que este sencillo ejemplo realiza la comparación de los valores y en función de su resultado activa una salida u otra

En la figura 37 vemos las variable sen la ventana correspondiente activada por el botón  de la barra de botones o la opción *VM-->Definir Variables* del menú.

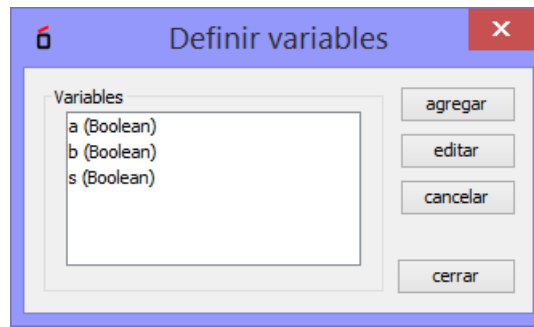


Figura 37

El botón de “start” se deberá activar cada vez que queramos realizar la función de comparación, ya que una vez activado el “led” correspondiente el diagrama de flujo queda detenido (no existe un bucle de realimentación).

En la figura 38 vemos la ventana en el modo de ejecución.



Figura 38

## 18.2. EJEMPLO 2

### REALIZACION DE BUCLES Y EVALUACION DE VALORES DECIMALES (double)

En este caso realizamos el mismo ejemplo pero en lugar de comparar variables de tipo booleano lo hacemos con variables de tipo decimal (double).

En este ejemplo se han colocado dos bucles con dos botones que “disparan independientemente” cada uno de los dos diagramas de flujo, lo cual nos permite afirmar que se pueden hacer convivir dos o mas diagramas y ejecutarse en paralelo.

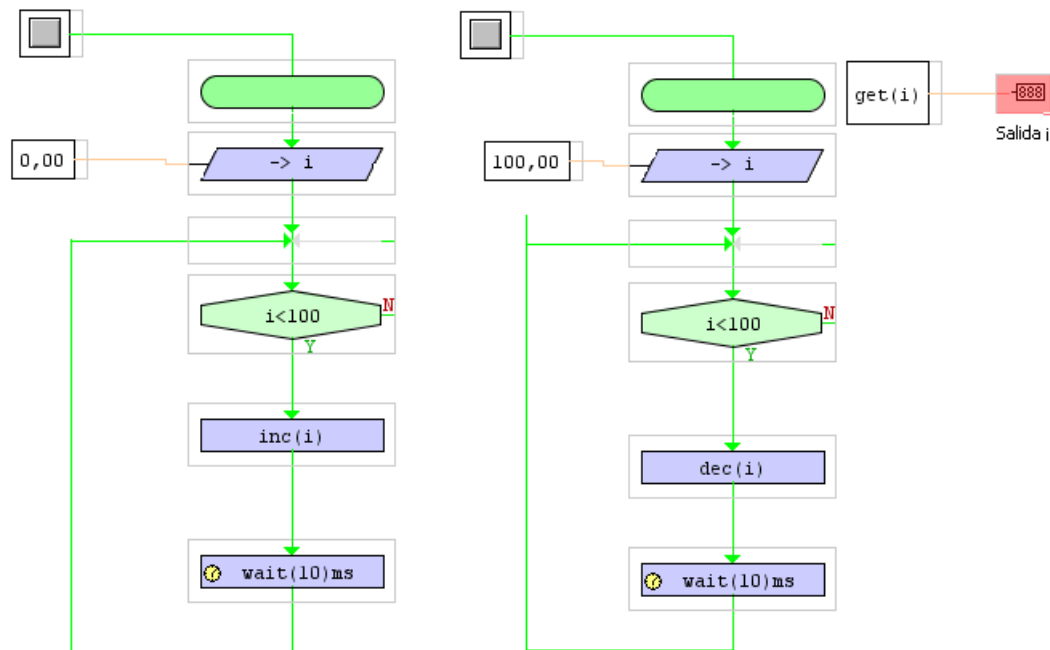
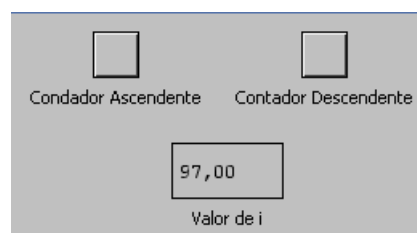


Figura 39



El diagrama de la izquierda es un contador ascendente de 0 a 100 que se implementa en torno a la variable de cuenta **i** que es inicializada mediante el correspondiente valor 0,00 y que posteriormente es testado su valor con la condición  $i < 100$  saliendo del bucle cuando no se cumple la condición establecida. La variable se va incrementando cada vez que pasa por el bloque de función “**Incrementar**” ( $inc(i)$ ). Se ha colocado un bloque de espera “**Retardo**” ( $wait(10)ms$ ) con el fin de poder visualizar como los valores van cambiando.

Figura 40

El diagrama de flujo de la derecha realiza la misma función pero en este caso decrementando el valor de la variable **i**, es decir se comienza la cuenta asignando a **i** el valor 100 y se decrementa hasta que se deja de cumplir la condición del bloque “**Decisión**”  $i > 0$ .

En la figura 40 vemos la pantalla de visualización

### 18.3. EJEMPLO3

#### COMPARACIÓN DE VARIABLES TIPO STRING

En este ejemplo vemos el caso de una comparación de cadenas de caracteres que se recogen a través de dos cajas de recogida de texto; Variables **i** y **j**



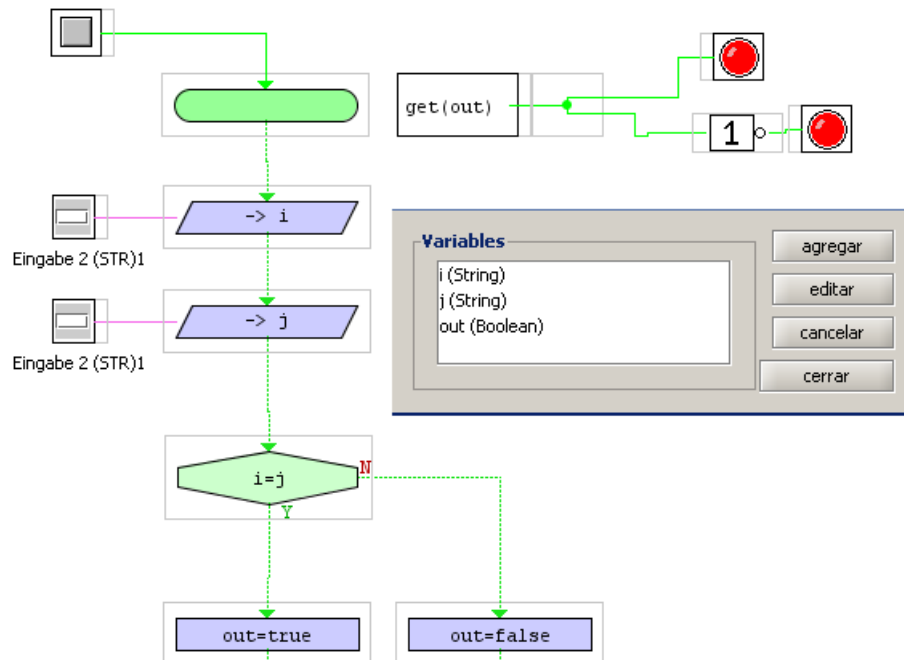


Figura 41

El resultado de la comparación se muestra mediante dos diodos “led”. La comparación se ordena mediante el botón “compare”

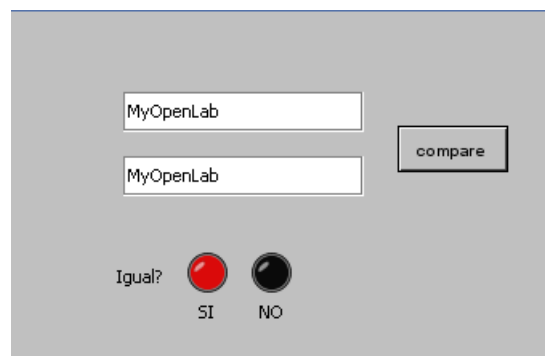


Figura 42

## 18.4. EJEMPLO 4

### LUCES DE ALARMA EN COCHE

Se trata de crear una aplicación que active de manera alternativa dos luces que simulan las de una ambulancia o coche de policía y además una luz que actúe de manera que simule un flash

Para ello se crean dos organigramas habiendo definido previamente las variables on y onx como variables de tipo booleano. Figura 43

Las variables se recogen del diagrama de flujo a través de la función “Entrega Dato” *get(on)* y *get (onx)*

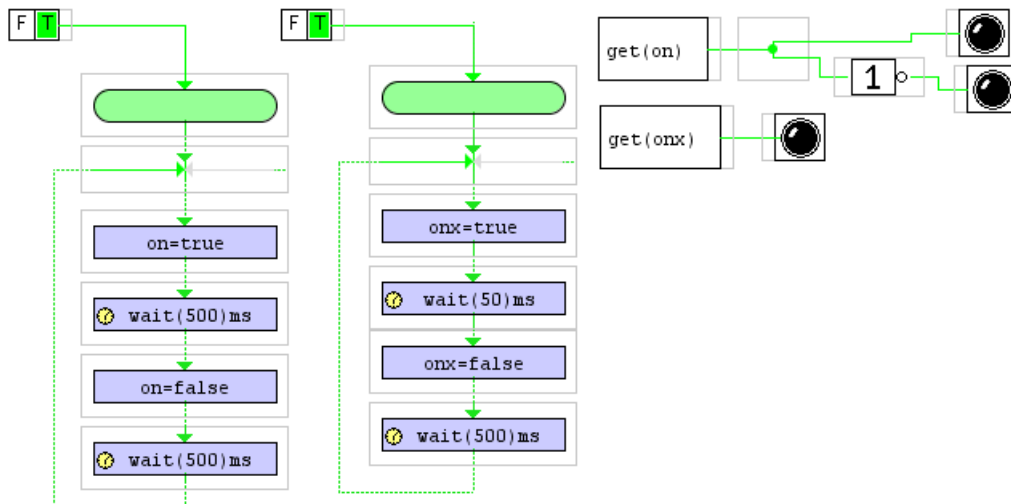


Figura 43

Los dos organigramas son de tipo realimentado (en bucle) y para conseguir los efectos de intermitencia y flash se recurre a la instrucción “Retardo” *wait*.



Para que estén permanentemente activadas las luces se inicia cada diagrama de flujo con una constante de tipo booleano en cada uno de ellos.

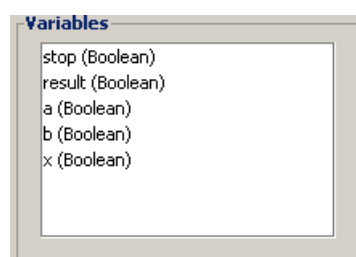
La pantalla de presentación esta formada por las lámparas (objetos diodo led) y una imagen de un coche. Figura 44.

Figura 44

## 18.5. EJEMPLO 5

### ENTREGA Y RECOGIDA DE VALORES DE UN DIAGRAMA DE FLUJO

En este ejemplo (figura 45) vemos la posibilidad de recoger y/o entregar valores desde un diagrama de flujo hacia el resto de funciones de MyOpenLab.



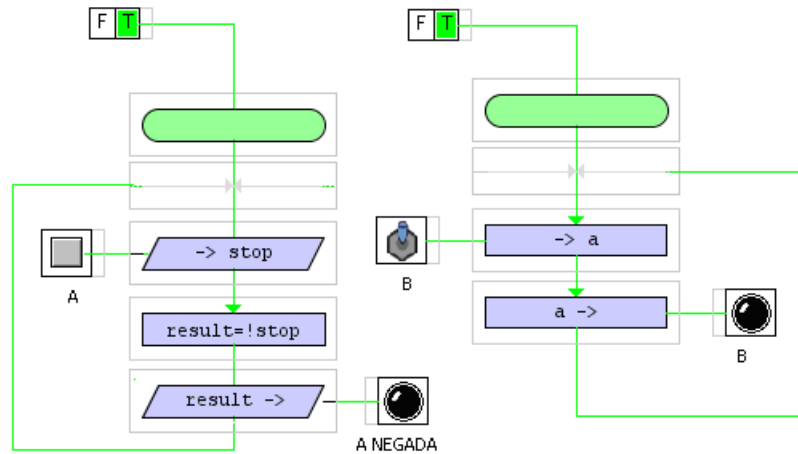


Figura 45

En el diagrama de flujo de la izquierda vemos como el valor emitido por el “Botón A” es recogido y mostrado por el diodo “Led A NEGADA”.

En el diagrama de flujo de la derecha se recoge la variable “B” y se entrega en el “Diodo Led B”

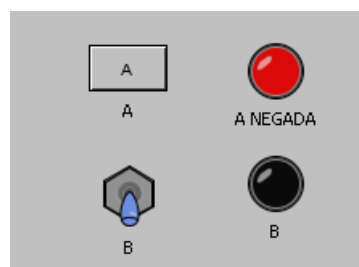


Figura 46

## 18.6. EJEMPLO 6

### SEMÁFORO

En el siguiente ejemplo se ha relacionado la simulación del funcionamiento de un semáforo en el que básicamente se establece una secuencia fija en el encendido y apagado de cada una de las lámparas del semáforo (rojo, verde, amarillo).

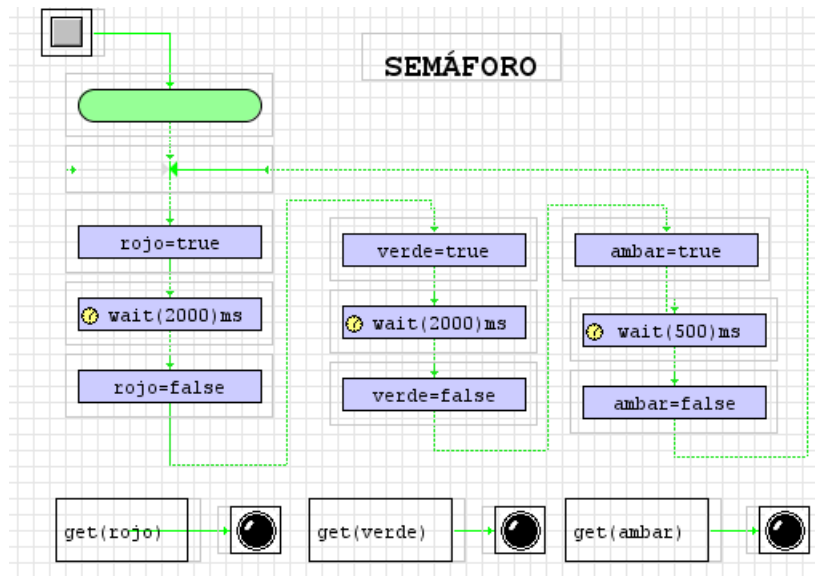


Figura 47

Las instrucciones que se utilizan son la de “Evaluación de Expresión “ en la que se fuera a “true” (activado) o a “false” (desactivado) cada una de las lamparas. Por otro lado los tiempos de encendido de cada lampara se fijan mediante la instrucción “Retardo”.

Las variables que se han creado se muestran en la figura 48.

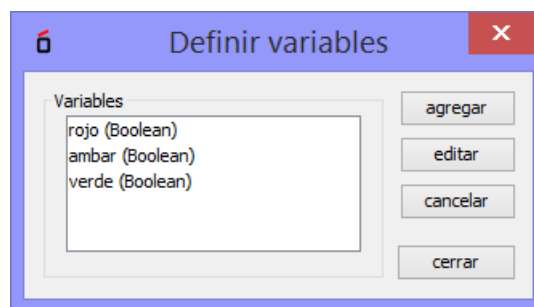


Figura 48

Las variables de salida de nuestro sistema se recogen mediante la función “Entrega dato” y se lleva los diodos led indicadores de cada uno de los colores.

En la figura 49 vemos el panel de visualización en estado de ejecución de la simulación.

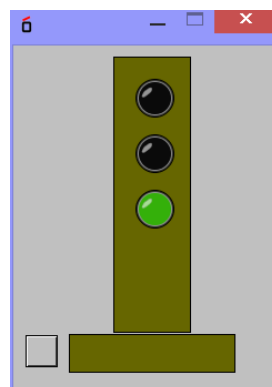



Figura 49

## OTRA FORMA DE REALIZAR LA SIMULACIÓN DE UN SEMÁFORO

Este mismo modelo de simulación se puede llevar acabo utilizando ele elemento de “librería de usuario”  .

Este bloque de función incorpora un semáforo completo en el que tenemos como variables de entrada los valores de los tiempos de encendido de cada lámpara y sus salidas son directamente conectables a los diodos leds indicadores de lámpara. En la figura 50 vemos el esquema del bloque semáforo y después la aplicación de este en una simulación:

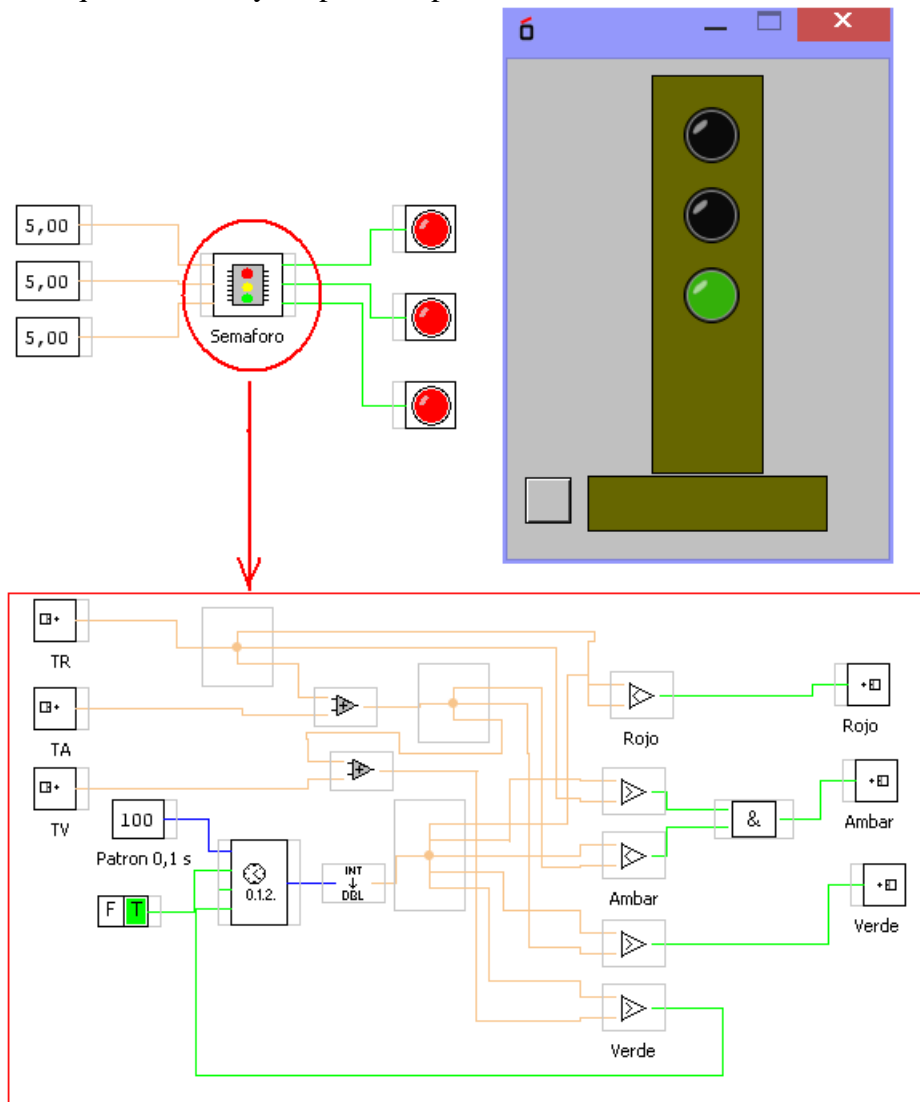


Figura 50