

APRENDIENDO A PROGRAMAR

CON DIAGRAMAS DE FLUJO



Build 3.0.8.3

José Manuel Ruiz Gutiérrez

j.m.r.gutierrez@gmail.com

Índice

Introducción

NIVEL I: EJEMPLOS

REALIZAR OPERACIONES MATEMÁTICAS BÁSICAS

1. Realizar una sencilla suma de dos números y mostrar su resultado.

TEMPORIZAR

2. Realizar un sencillo intermitente haciendo uso de la instrucción de temporización.

COMPARAR

3. Realizar la comparación de una magnitud de tipo double (numero real) con un valor predeterminado.
4. Realizar la comparación de dos magnitudes de tipo double (numero real).
5. Realizar el mismo ejercicio anterior pero recogiendo directamente las variables de entrada a través de dos bloques de recogida de datos “Leer”.

CONTAR

6. Realizar un contador que cuente desde 0 a 20 y cada valor que avance lo haga transcurrido un 0,5 seg.
7. Contador que cuente y active una salida.
8. Contador de eventos externos.

NIVEL II: EJEMPLOS

1. Averiguar si un número es par o impar.
2. Averiguar si el resultado de una operación es negativo o positivo.
3. Realización de una aplicación que incluya operaciones de cálculo iterativo
4. Ejecución simultánea de varios diagramas de flujo.

5. Ejecución de un diagrama de flujo con interacción con distintas variables.
6. Comparación de cadenas de texto (strings)
7. Control de un semáforo
8. Realizar el mismo semáforo anterior pero utilizando la tarjeta Velleman.
9. Alarma doméstica
10. Realizar un termostato.
11. Manejo de Subrutinas. Llamadas a procedimientos.
12. Instrucción FOR NEXT
13. Realización de la suma de los N primeros números naturales.
14. Utilización de la instrucción WHILE

NIVEL III: EJEMPLOS PROPUESTOS

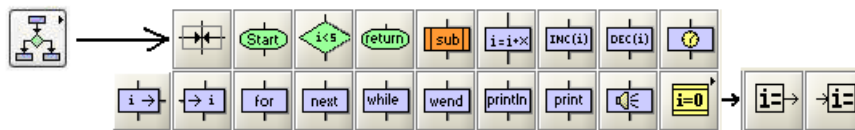
1. ASCENSOR
2. PARKING
3. PUERTA DE ENTRADA A UNA FINCA
4. MÁQUINA DE CAFÉ
5. GASOLINERA


Introducción

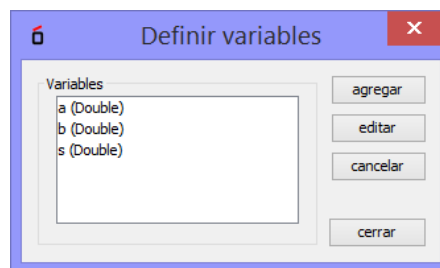
APRENDIENDO A PROGRAMAR MEDIANTE DIAGRAMAS DE FLUJO

En este apartado vamos a realizar una serie de ejercicios que de manera creciente en orden de complejidad nos permitan comprender el lenguaje funcional de los diagramas de flujo que MyOpenlab implementa como una poderosa herramienta de programación de automatismos. Vamos a utilizar la librería que al respecto dispone MyOpenLab:

Librerías Panel Circuito



Debemos tener en cuenta que las variables que intervengan en la aplicación con diagramas de flujo deben definirse previamente mediante el botón 



A la hora de darle nombre a una variable no debe llevar el carácter “_” ni ningún número, es preferible que se nombren con una palabra corta escrita en minúscula.

IMPORTANTE: A continuación se te plantean una serie de ejercicios básicos para realizar mediante el uso de esta librería de funciones de programación.

Es muy recomendable que para abordar con éxito estos ejercicios se hayan leído previamente los manuales:

Guia_usuario_MyOpenLab 3.0.8.0 (J.M.Ruiz)

Guia_Diagramas_Flujo V3.0.8.3 (J.M.Ruiz)

Nota muy importante:

Modificaciones importantes en la sintaxis para la nueva librería Flowchart

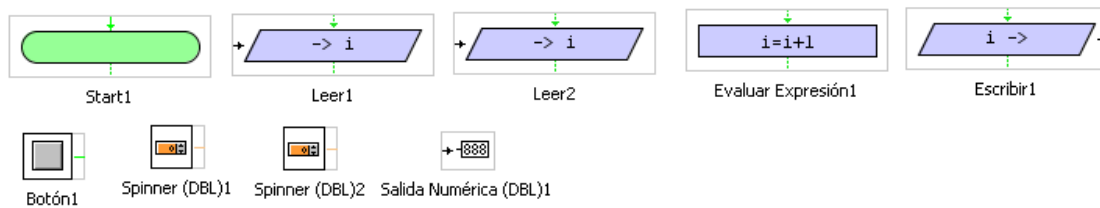
Antes	Ahora
=	==
Valor tipo dbl 10	Valor tipo dbl 10.00
sin(x)/cos(x)/...	Math.sin(x)/Math.cos(x)/..
Operador AND &	Operador AND &&
Operador OR	Operador OR

NIVEL I: EJEMPLOS

REALIZAR OPERACIONES MATEMÁTICAS BÁSICAS

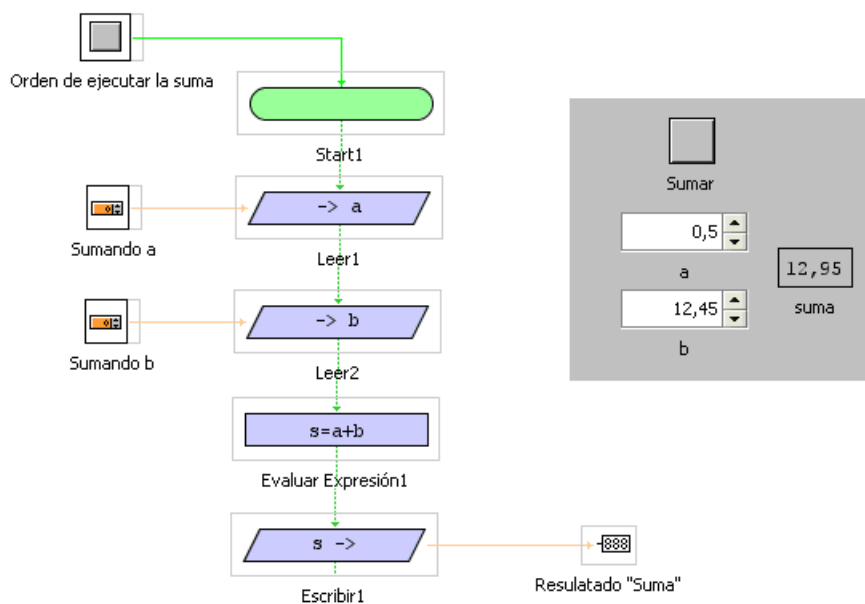
1. Realizar una sencilla suma de dos números y mostrar su resultado.

Los siguientes bloques de función son los que se utilizaran en la realización de este proyecto.



Variables que hay que definir: **a,b,s** de tipo double

Solución: *sumar.vlogic*

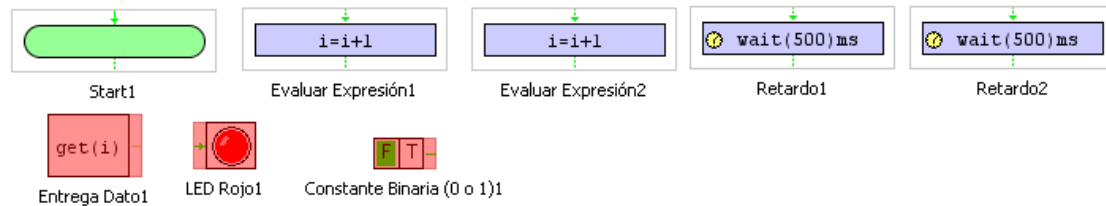


TEMPORIZAR

2. Realizar un sencillo intermitente haciendo uso de la instrucción de temporización.

Se trata de conseguir que una salida (LED) se encienda y apague de manera cíclica teniendo en cuenta que cada ciclo tendrá una duración $T = T_e + T_a$ (Tiempo total igual a tiempo encendido mas tiempo apagado). Los tiempos T_e y T_a no son variables externas del sistema sino parámetros que se asignan en la propia orden de temporización.

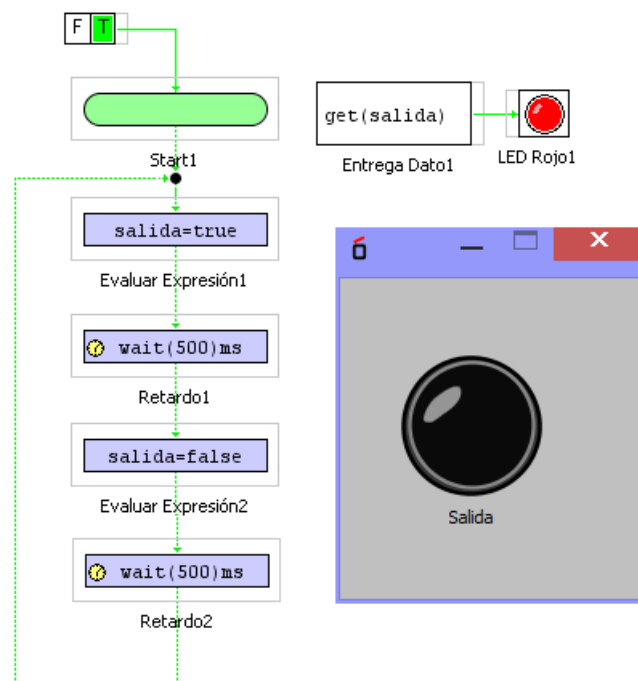
Funciones a utilizar:



Variables a definir:

salida (boolean)= sirve para gobernar el encendido apagado del diodo led

Solución: *intermitente.vlogic*



COMPARAR

3. Realizar la comparación de una magnitud de tipo double (numero real) con un valor predeterminado.

Se trata de recoger una variable de tipo double y compararla con un parámetro (constante) activando una variable de tipo booleana en función del resultado de la comparación.

Variables a definir: **i** (double) valor de entrada

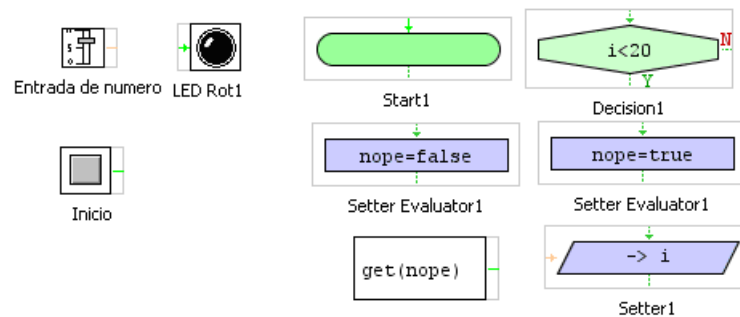
nope (boolean) indicador de resultado de la comparación.

Funcionamiento: Se recoge el valor de **i** y se comparara con “20” resultando que:

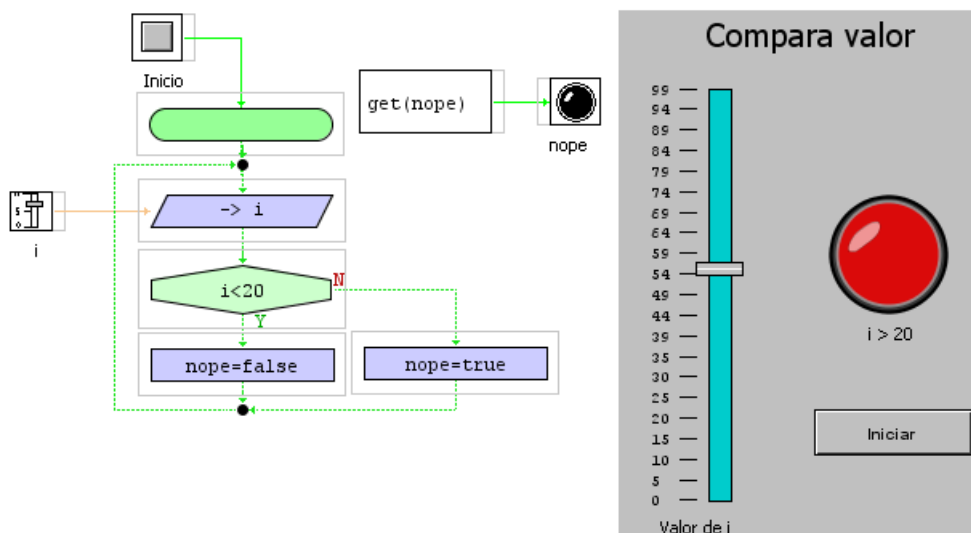
Si **i < 20** entonces **nope=false** (desactivada “0”)

En caso de que **no se cumpla que i < 20** se **nope=true** (se activa “1”)

Los bloques que se deben utilizar en este ejercicio son los siguientes



Solución: *comparar.vlogic*



4. Realizar la comparación de dos magnitudes de tipo double (numero real).

Se trata de recoger dos variables **a** y **b** de tipo double y compararlas activando una variable de tipo booleana **out** en función del resultado de la comparación.

Variables a definir: **a** y **b** (double) valores de entrada

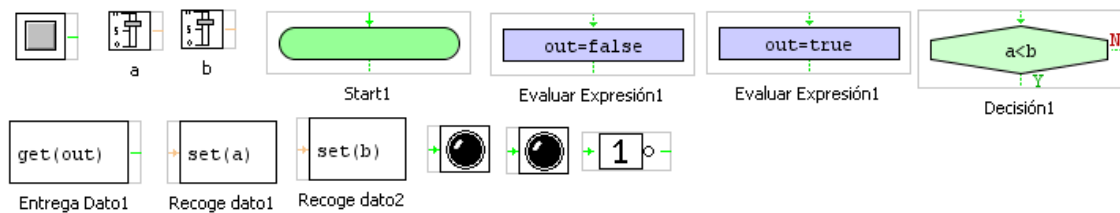
out (boolean) indicador de resultado de la comparación.

Funcionamiento:

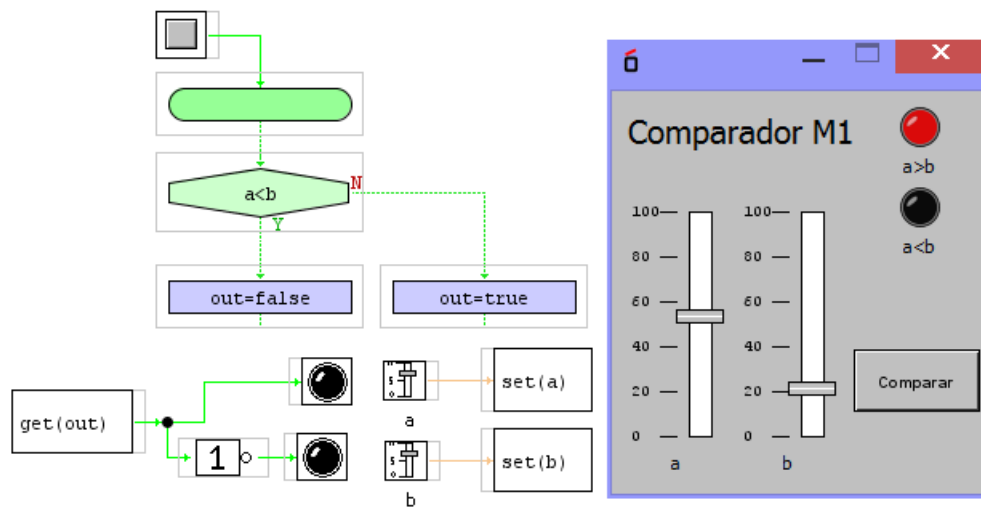
Si $a < b$ entonces **out=true** (activada “1”)
En caso contrario **out=false** (se desactiva “0”)

Convendremos en que cada vez que se quiera realizar la comparación tendremos que pulsar el botón de inicio de ejecución del diagrama de flujo.

Los bloques que se deben utilizar en este ejercicio son los siguientes:



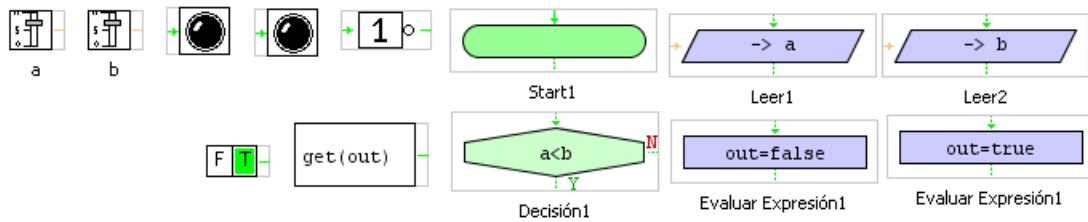
Solución: *Comparador M1.vlogic*



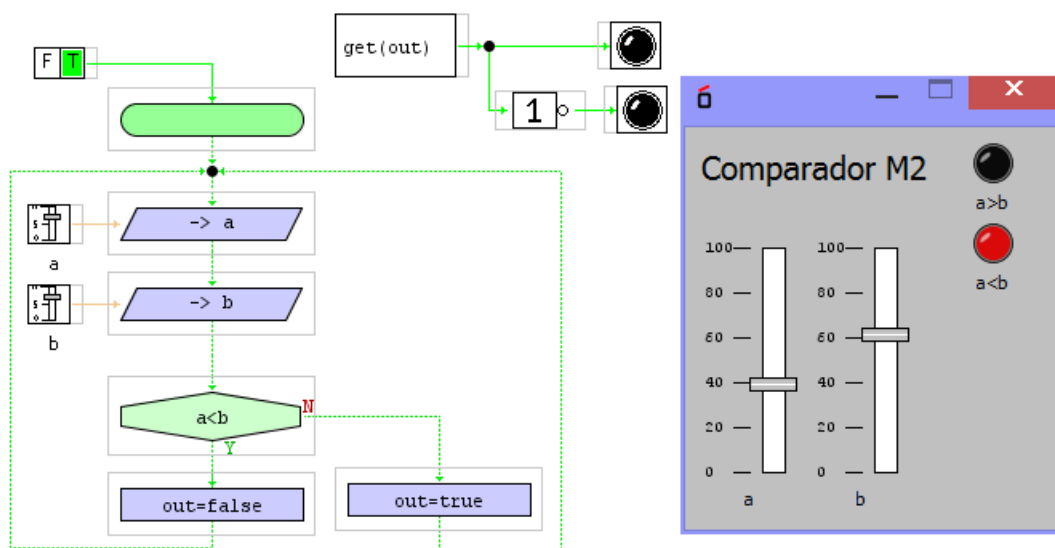
5. Realizar el mismo ejercicio anterior pero recogiendo directamente las variables de entrada a través de dos bloques de recogida de datos “Leer”.

Queremos también en este montaje que el programa se ejecute de manera continua, es decir que una vez realizada la comparación y dada la orden correspondiente en función de la comparación vuelvan automáticamente a leerse las variables y a realizarse la comparación (ejecución cíclica).

Los bloques a utilizar son los siguientes:



Solución: Comparador M2.vlogic





CONTAR

6. Realizar un contador que cuente desde 0 a 20 y cada valor que avance lo haga transcurrido un 0,5 seg.

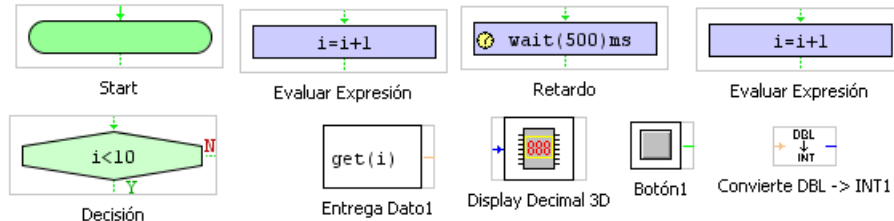
Una vez que termine la cuenta el contador se parará. La variable que hay que definir la nombraremos como **contador** (recuerda que será de tipo double). Para arrancar el programa se utilizará un botón. El valor **contador** se recogerá y se entregará a un

display numérico

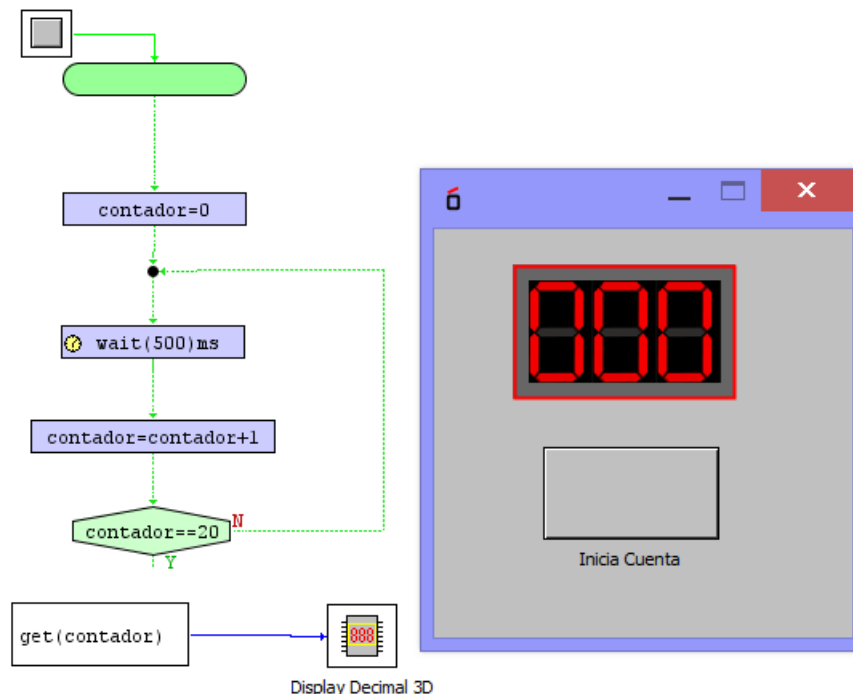


El componente Convierte DBL -> INT1 sirve para pasar el valor de contador (de tipo double) a valor de tipo entero (integer) ya que el display solo admite este tipo de valor. Este componente se encuentra en la librería  -> 

Los componentes a utilizar serán:



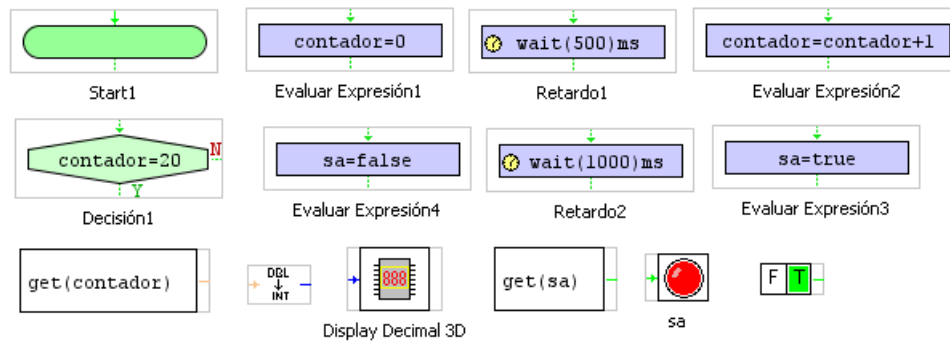
Solución: *contador_basico1.vlogic*



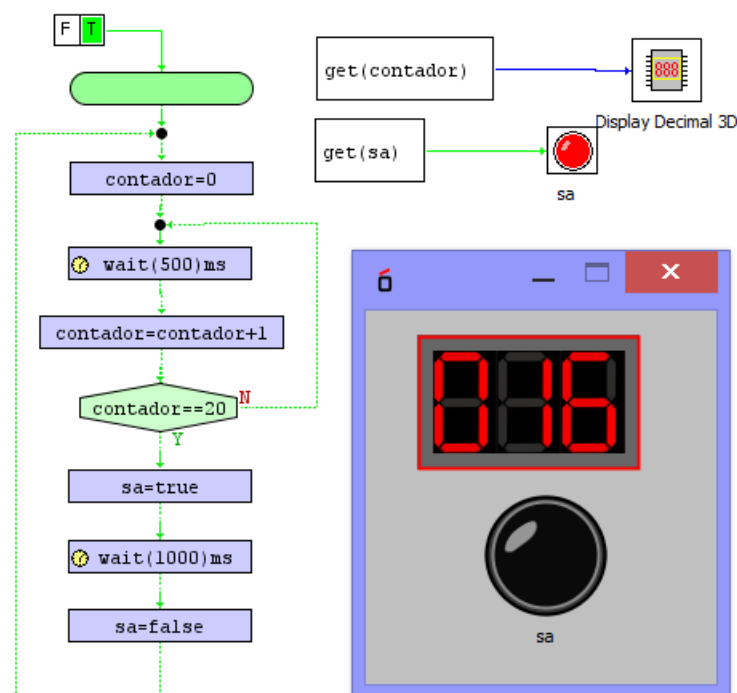
7. Contador que cuente y active una salida.

En este ejercicio se trata de realizar la modificación en el anterior para que el contador este permanentemente contado de 0 a 20 y que además cuando cuente 20 active durante un segundo una salida digital (LED). La salida digital que proponemos se deberá nombrar como **sa** y lógicamente será de tipo boolean. Téngase en cuenta que ahora no se deberá poner un pulsar de arranque porque nada mas iniciada la simulación el contador se pondrá a contar y estará siempre contando.

Los componentes a utilizar son:

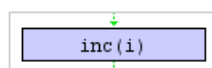


Solución: *contador_basico2.vlogic*



8. Contador de eventos externos.

Ahora se trata de realizar un contador que deberá contar cada vez que pulsemos un botón. Es decir queremos hacer un contador que en lugar de que cuente solo cuente cuando se produzca un evento (por ejemplo contar coches en un parking). Queremos que el contador cuente hasta 10 y después vuelva a empezar de nuevo. La orden de de cuenta (dentro del diagrama de flujo en esta ocasión queremos que se ejecute en



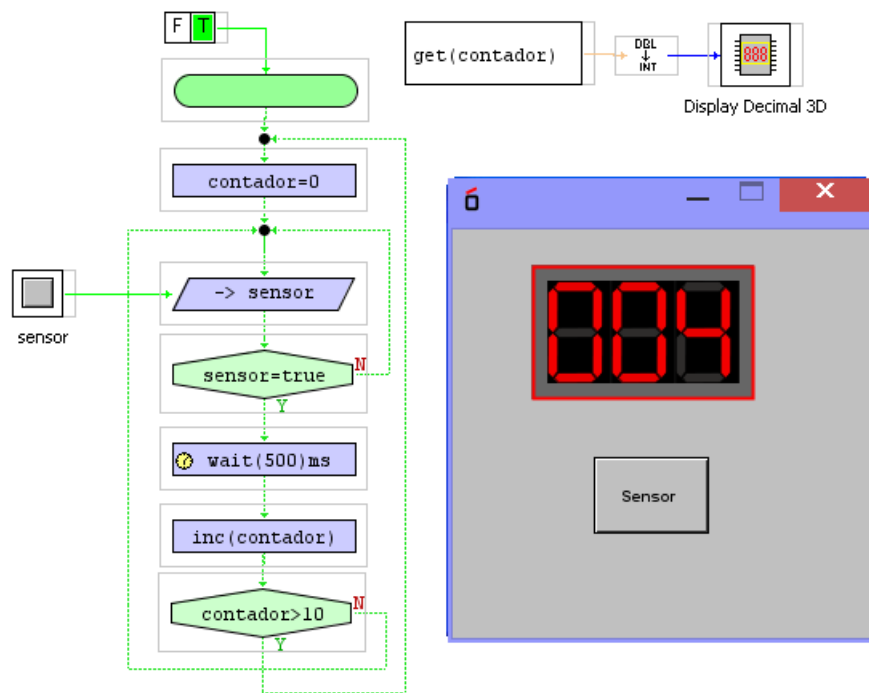
el bloque que lo que hace es incrementar la variable contador. En esta ocasión no pondremos los bloques a utilizar porque suponemos que el alumno ya sabe el bloque de función que deberá elegir.

Las variables que debemos definir serán:

contador (de tipo double) es la que almacena el número de cuenta.

sensor (boolean) es la que hace que cuente el contador

Solución: *contador_basico3.vlogic*



NIVEL II: EJEMPLOS

1. Averiguar si un número es par o impar.

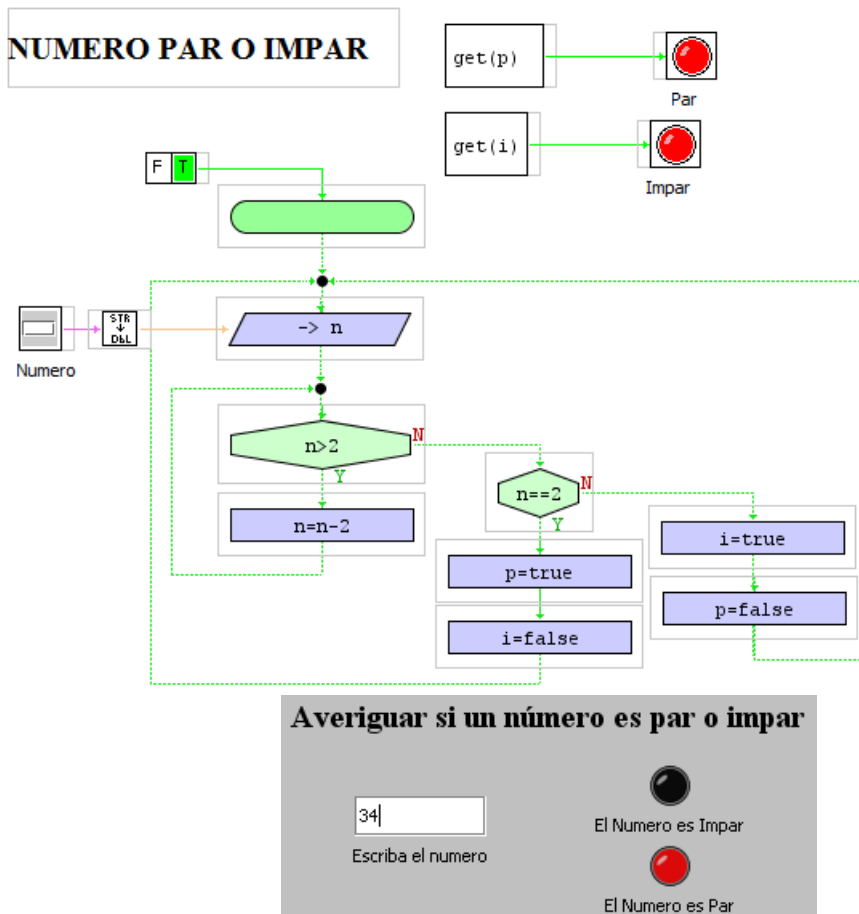
Con este ejemplo se trata de que el sistema, cuando le demos un número (variable double) nos indique si es par o impar.

El algoritmo consistirá en restar sucesivamente 2 al número mientras que este sea mayor que dos. Cuando no sea mayor que dos preguntaremos si es igual a dos, si es dos el número será par y si no es dos el número será impar. Se deberán establecer por lo tanto dos bloque condicionales en el primero preguntaremos si el número es mayor que dos y en el segundo preguntaremos si el número es igual a dos.

Variables a definir:

- n** Variable de tipo double: representa el número
- p** Variable de tipo boolean: indica si el número es par:
- i** Variable de tipo boolean: indica si el número es impar

Solución: *par_impar.vlogic*



2. Averiguar si el resultado de una operación es negativo o positivo.

Se trata de que el sistema averigüe si el resultado de una operación matemática es positivo o negativo.

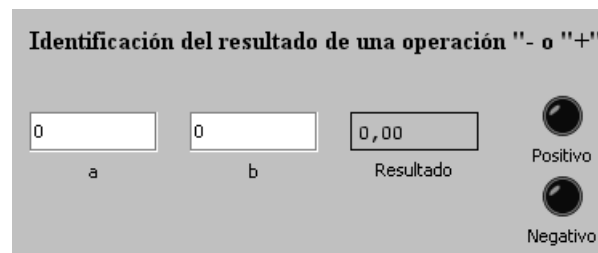
Para realizar la operación matemática recurrimos al bloque de cálculo montando el esquema de la figura que lo que hace es sencillamente calcular el resultado de la resta entre las variables x e y lo entrega al diagrama de flujo a través de la variable a

El algoritmo lo que debe hacer es simplemente preguntar si el numero a (variable de tipo double) es menor que cero (con un bloque tipo condicional) y en función de ello activar la variable **negativo** que es de tipo booleano.

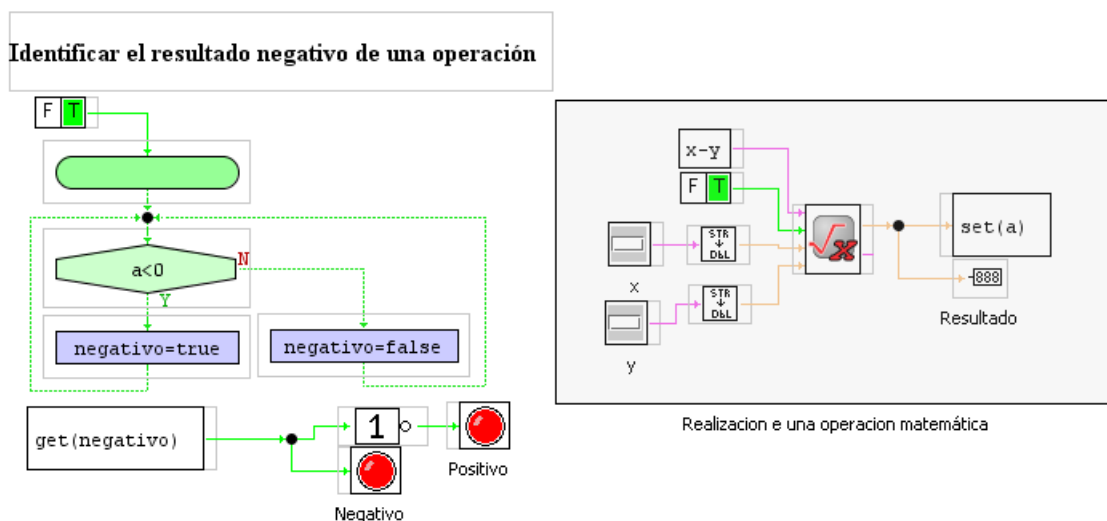
Variables a definir

a	Variable tipo double: el número del que queremos averiguar si es positivo o negativo
negativo	variable de tipo boolean: Se activa (trae) cuando el numero es negativo y se desactiva (false) cuando es positivo

El aspecto del Panel Frontal seria el siguiente:



Solución: *positivo_negativo.vlogic*



3. Realización de una aplicación que incluya operaciones de cálculo interactivo.

En esta aplicación se trata de realizar una serie de operaciones de manera interactiva (mediante un bucle) Se utilizará una variable **x** se vaya incrementando y a la vez se realice el el cálculo del seno de x. En el bucle de cálculo pondremos un retardo con el fin de que podamos ver como se van realizando los cálculos, si no fuera así se verían pasar los valores muy rápidamente.

Se definen dos variables:

- x** variable que va incrementándose en pasos de 0.1
- i** variable que se obtendrá de calcular el seno de x ($\sin(x)$)

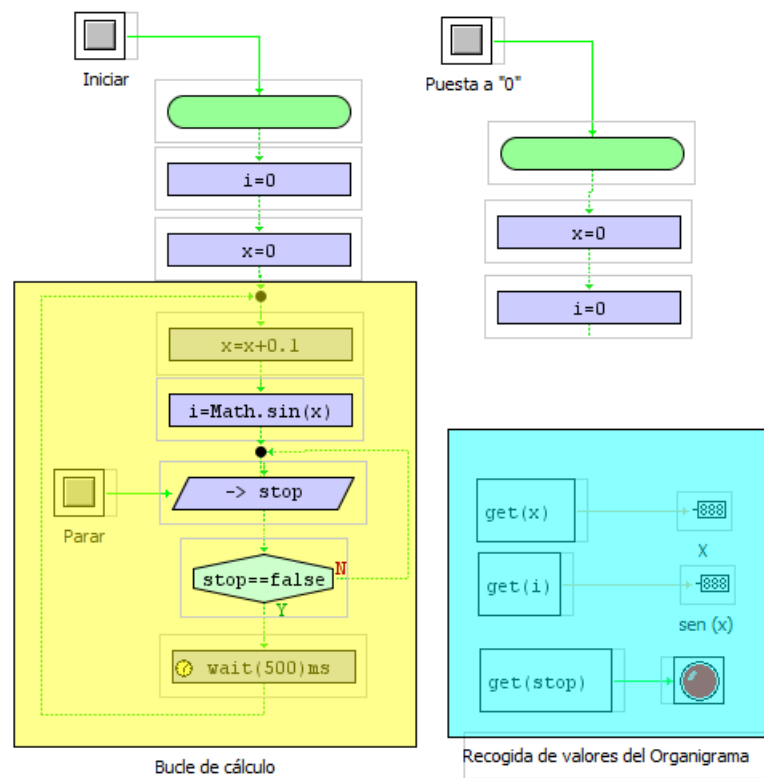
Se dispondrá de tres botones:

- | | |
|---------------------------|-----------------------------------|
| Botón Iniciar | inicia el cálculo. |
| Botón Puesta a "0" | pondrá a cero las variables x e i |
| Botón de Parar | detendrá el cálculo. |

Mostraremos en el Panel Frontal los valores de x y del seno de x



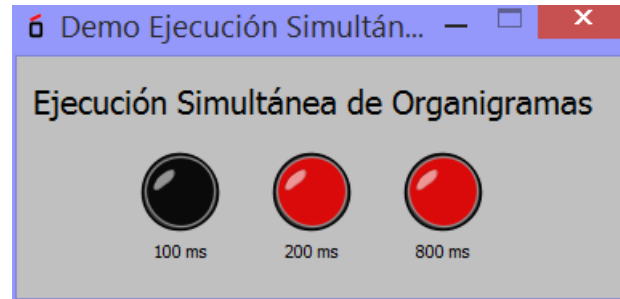
Solución: Cálculo interactivo.vlogic



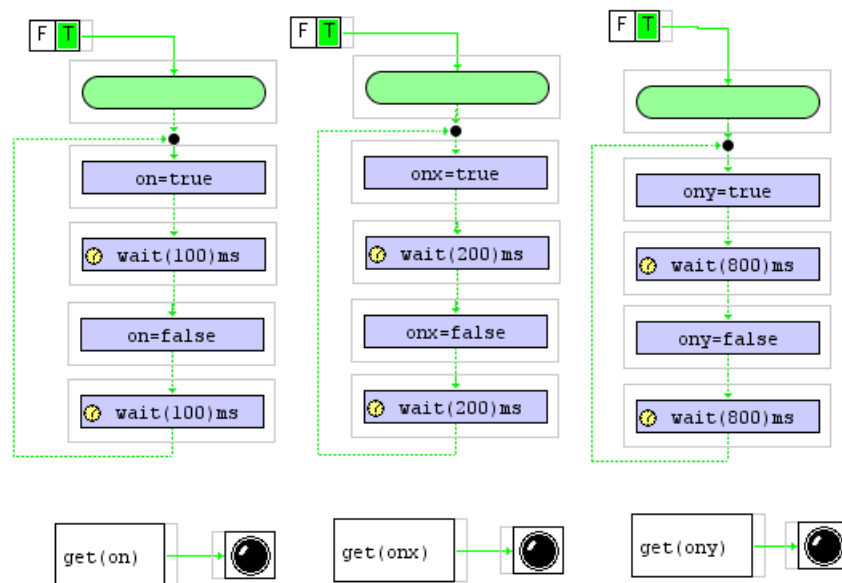
4. Ejecución simultánea de varios diagramas de flujo.

En este ejemplo se trata de comprobar como MyOpenlab es capaz de ejecutar varios diagramas de flujo a la vez. Se crearán hasta tres variables de tipo booleano **-on**, **onx** y **ony**- que se deben activar y desactivan de manera cíclica (modo intermitente) con distintas cadencias de tiempo $T_{on}=100\text{ ms}$, $T_{onx}=200\text{ ms}$ y $T_{ony}=800\text{ ms}$.

En la figura vemos como quedaría el panel de simulación

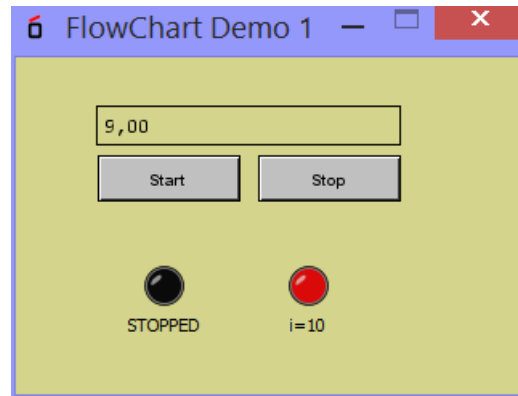


Solución: *ejecución simultánea.vlogic*



5. Ejecución de un diagrama de flujo con interacción con distintas variables.

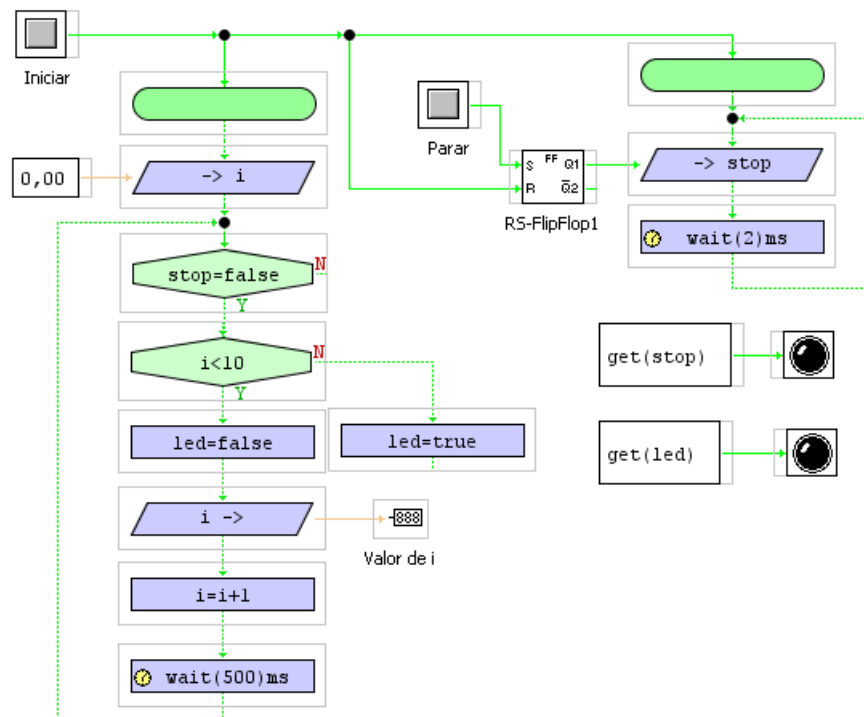
En el presente ejemplo se trata de realizar un contador de variable de cuenta **i** que contará hasta 10. En ese momento deberá activar una salida en forma de diodo led a la llamaremos **led** (variable booleana). Para detener el contador dispondremos de un botón “Parar” que gobernará un biestable tipo RS cuya salida será la que utilicemos en el diagrama de flujo como señal de parada **stop**.



Las variables serán:

- i** Variable de cuenta (tipo double)
- led** Salida cuando $i=10$ (tipo boolean)
- stop** Detiene el proceso de cuenta (tipo boolean)

Solución: *interaccion_variables.vlogic*



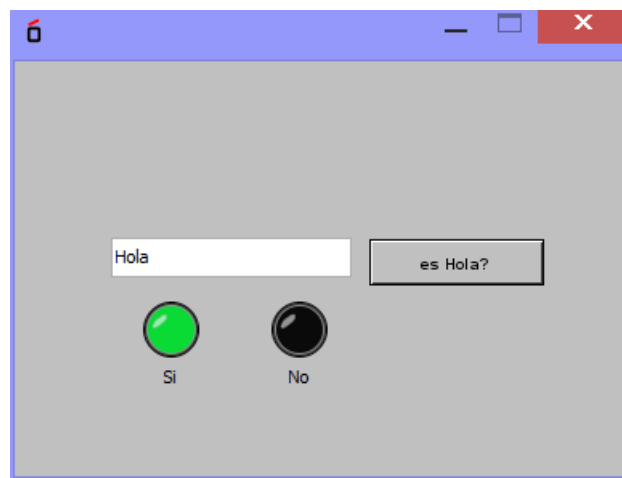
6. Comparación de cadenas de texto (strings)

Con este ejemplo se pretende realizar la comparación de una cadena de texto que recoge el sistema a través de una caja de texto (string) y la compara con otra cadena fija por ejemplo “Hola”, si resultan iguales se activa una salida (true) y si no se permanece desactivada (false).

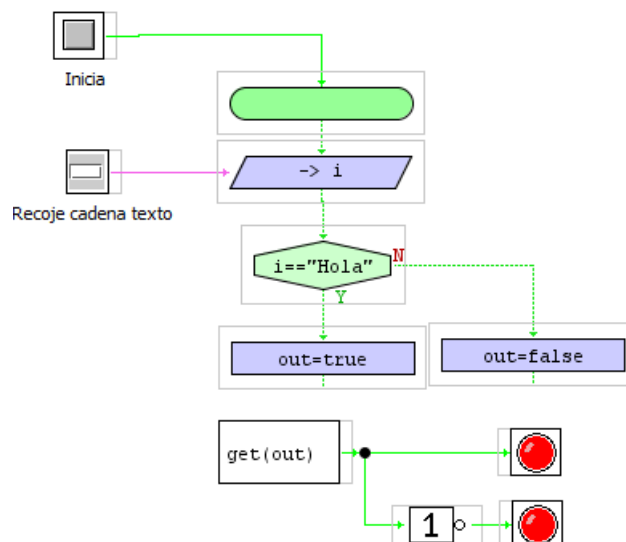
Las variables son:

- i** variable tipo string: cadena de texto de entrada
- out** variable boolean: salida que indica si la cadena introducida es igual que “Hola”.

El Panel Frontal del ejercicio podría ser el siguiente:



Solución: *comparacion_cadenas_texto.vlogic*



7. Control de un semáforo

Se trata de realizar un semáforo que gobierne tres salidas en forma de diodos led (rojo, ámbar y verde)

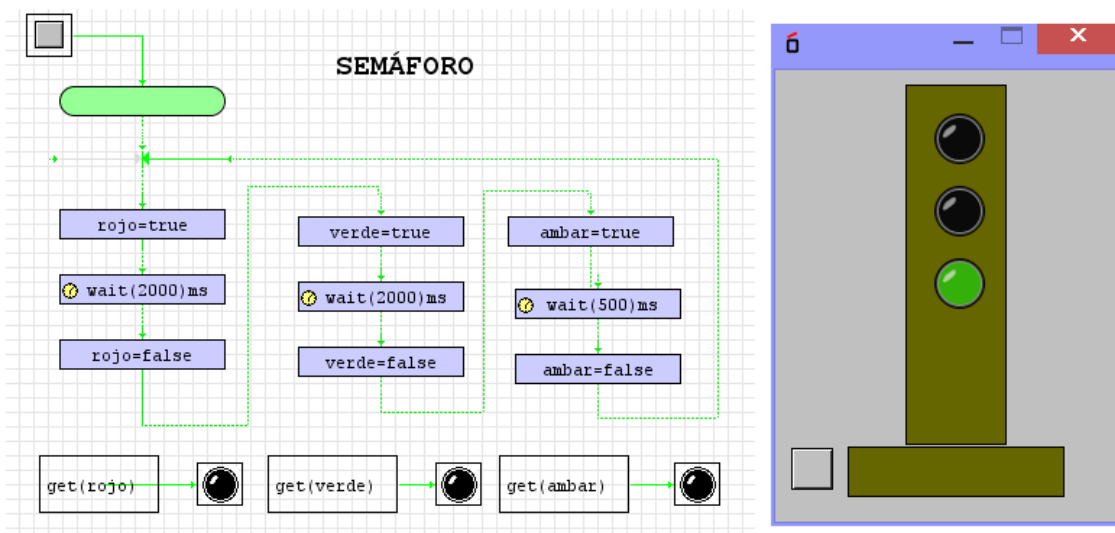
Señales de salida:

roja, ambar y verde (todas de tipo Boolean)

Parámetros:

tiempo_rojo=2 seg. Tiempo_ambar=2 seg. Tiempo_verde=0,5seg.

Solución: *semaforo_flowchart.vlogic*



8. Realizar el mismo semáforo anterior pero utilizando la tarjeta Velleman.

Se trata de realizar un semáforo que gobierne tres salidas de la tarjeta Velleman (rojo, ámbar y verde)

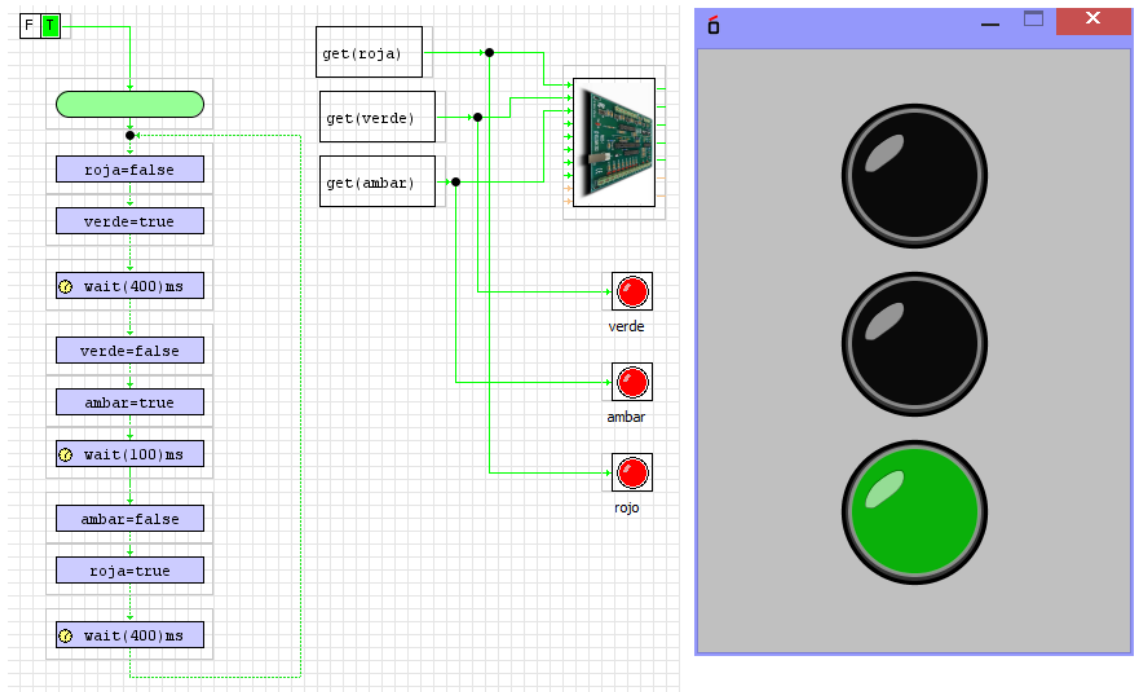
Señales de salida:

roja, ambar y verde (todas de tipo Boolean)

Parámetros:

tiempo_rojo=4 seg. Tiempo_ambar=2 seg. Tiempo_verde=4 seg.

Solución: *semaforo_Velleman.vlogic*



9. Alarma doméstica

Se trata de realizar mediante la tarjeta Velleman un sistema de alarma domestico.

Señales de entrada:

s_ventana, s_puerta, (Boolean)

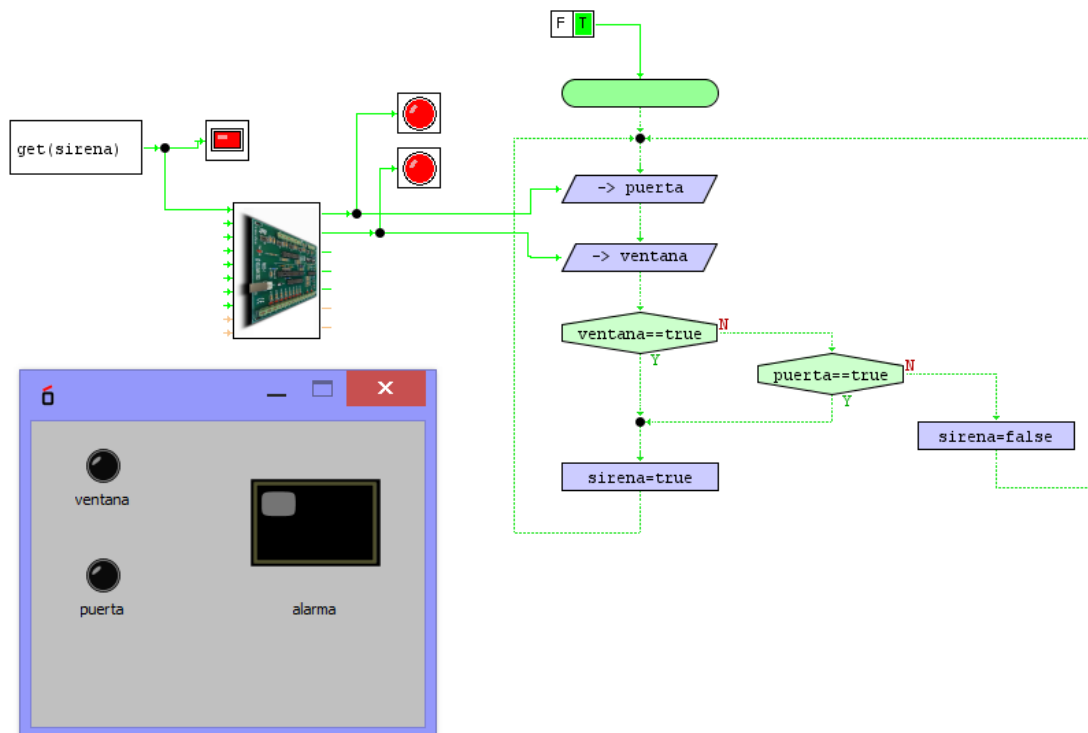
Señales de salida:

sirena (boolean)

Funcionamiento: cada una de las dos señales de entrada se recogerán de dos entradas de la tarjeta Velleman (pulsadores). La salida se asignara a una de las salidas de la tarjeta.

Cuando cualquiera de las entradas se ponga en “1” (activada deberá de ponerse en modo intermitente la salida sirena.

Solución: *alarma_Velleman.vlogic*



10. Realizar un termostato.

Se trata de simular el comportamiento de un termostato que gobierna un elemento calefactor.

Variables de entrada:

Temperatura leída a través de un sensor **sensor** (tipo Double)
Valor de Consigna **consigna** (Tipo Double)

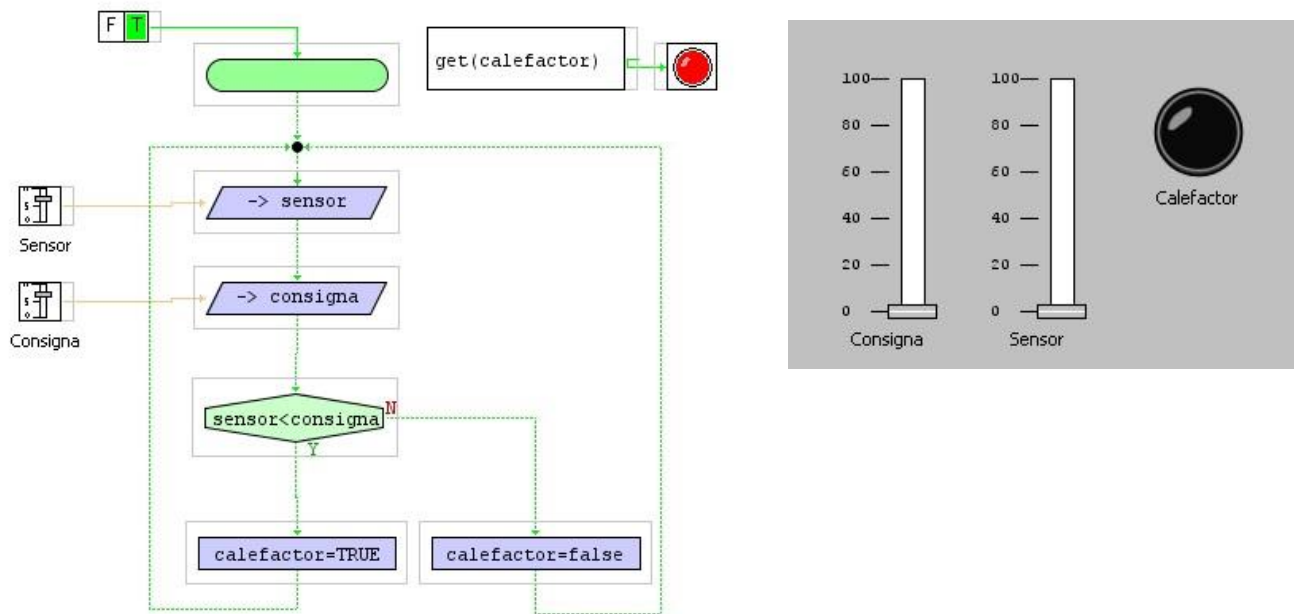
Variables de salida:

Elemento Calefactor **calefacción** (Tipo Boolean)

Funcionamiento: El programa se realizará haciendo uso de la librería de diagramas de flujo.

Se colocará un valor de ajuste (consigna) y después modificando el valor de la temperatura de sonda (sensor) se debe activar o desactiva la salida (calefacción) en función de la comparación de ambos valores (función condicional)

Solución: *termostato.vlogic*



11. Manejo de Subrutinas. Llamadas a procedimientos.

Con este ejemplo que ya se da resuelto se trata de demostrar la posibilidad de manejar “subrutinas” o procedimientos en el entorno de MyOpenlab con las librerías de tipo Flowchart.

Lo que hace el programa: *Demo Pocedimiento.vlogic*

- 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

$$y=Test(5,3)$$

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

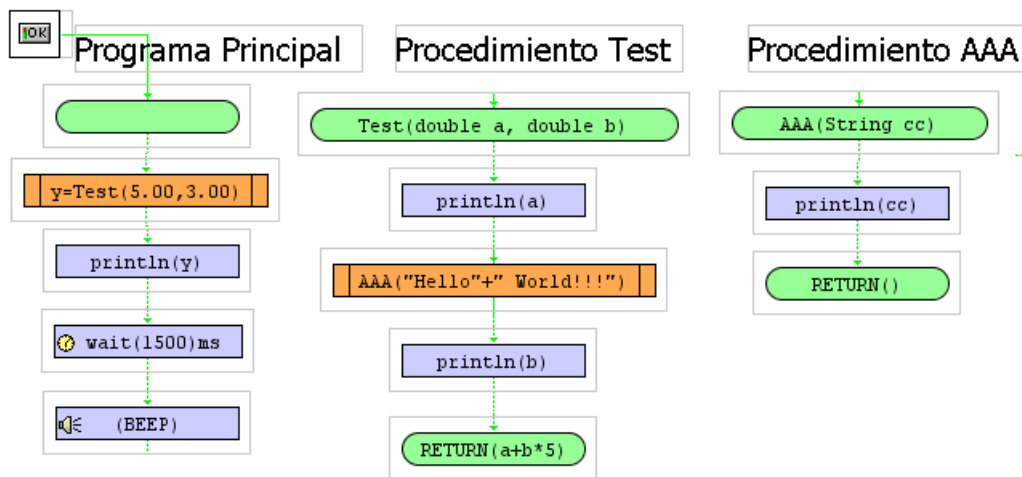


Figura 10

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

a=5.00 y b=3.00 al procedimiento Test

(No olvidar poner los valores de tipo *dbl* con dos decimales)

- 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

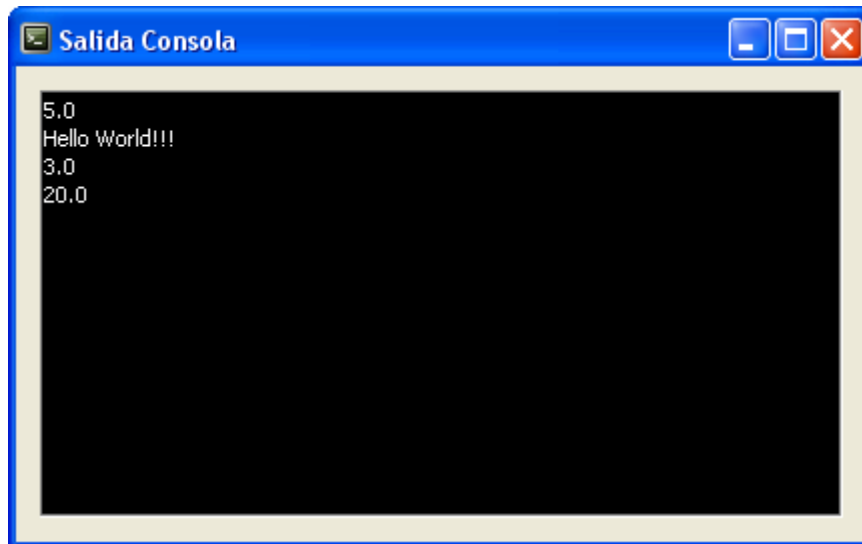
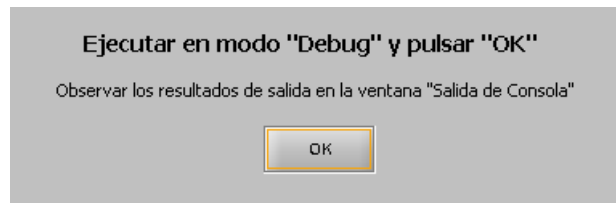
“wait (1500 ms)”

- Finalmente se ejecuta la instrucción BEEP y se acaba el programa

“BEEP”

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

Aspecto de las pantallas de Panel y Consola a la hora de ejecutar el programa



12. Instrucción FOR NEXT

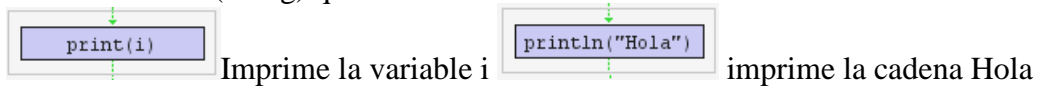
Este programa utiliza el bloque “FOR” y el bloque “NEXT” para configura un clásico bucle de los que se utilizan comúnmente en los lenguajes de programación y que resultan muy útiles en los sistemas de automatización y control. En este ejemplo se trata de imprimir un numero de veces la palabra “Hola”, el numero de veces (variable **a**) se lo daremos nosotros en modo ejecución.

Variables:

- i** variable contador del bucle FOR NEXT (tipo double)
- a** valor máximo que alcanzará el contador (tipo double)

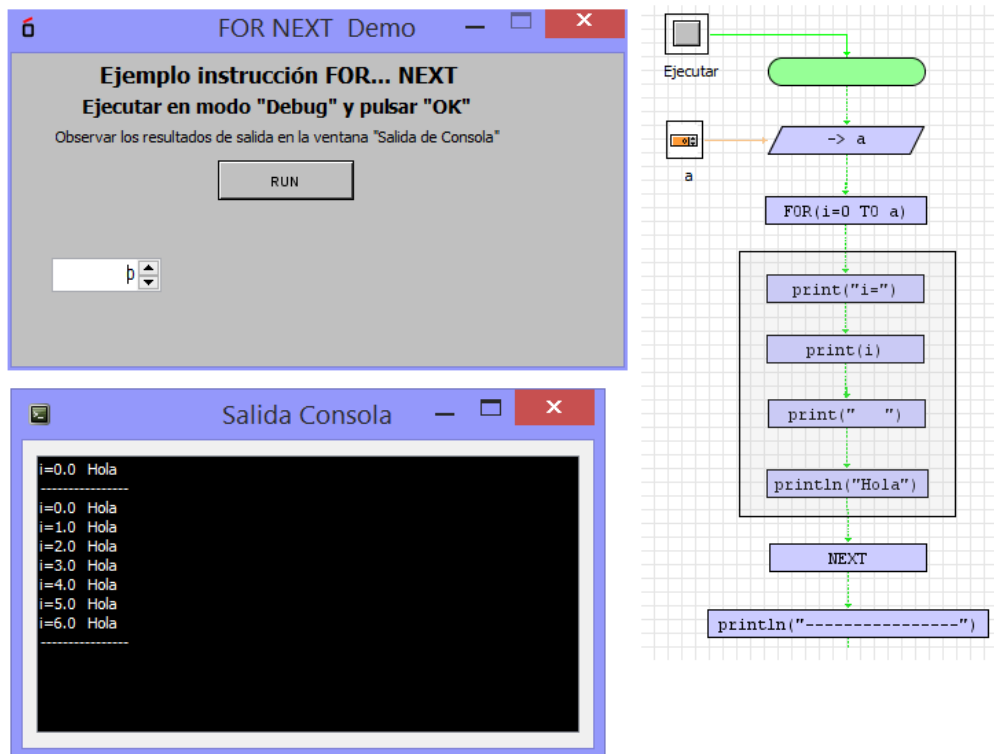
¿Como funciona?

El bucle a la hora de darle parámetros tenemos que indicará cual es la variable que vamos utilizar de “puntero” o contador en el bloque (en este ejemplo es **i**) así como el rango entre el cual va a variar (en este caso entre **0** y **a**), **a** es una variable que debemos definir como tipo double. A continuación de la instrucción FOR se colocaran un conjunto de instrucciones, en este caso “**Print**” que lo que hace es imprimir bien una cadena de texto (string) que le demos entre comillas o bien una variable



La estructura FOR siempre acaba con una instrucción NEXT que es la que realmente cierra o acota el número de instrucciones dentro del bucle.

Solución: *For_Next1.vlogic*



13. Realización de la suma de los N primeros números naturales.

En este ejemplo se trata de realizar el algoritmo que es capaz de sumar los N primeros números naturales. Pongamos unos ejemplos.

Si $n=2$ $\text{suma}=0+1+2=3$

Si $n=3$ $\text{suma}=0+1+2+3=6$

Si $n=5$ $\text{suma}=0+1+2+3+4+5+6=21$

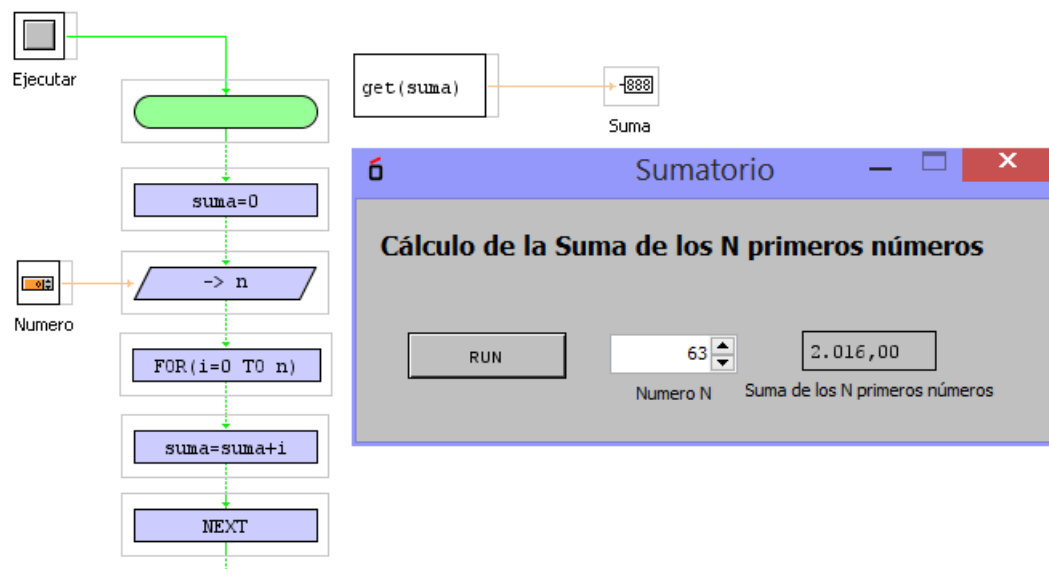
El programa lo que hará será pedirnos el numero “n” y nos devolverá el valor de “suma”

Las variables del ejercicio serán:

n Numero a calcular (double)

suma Valor de la suma de los n primeros números (double)

Solución: *sumatorio.vlogic*



14. Utilización de la instrucción WHILE

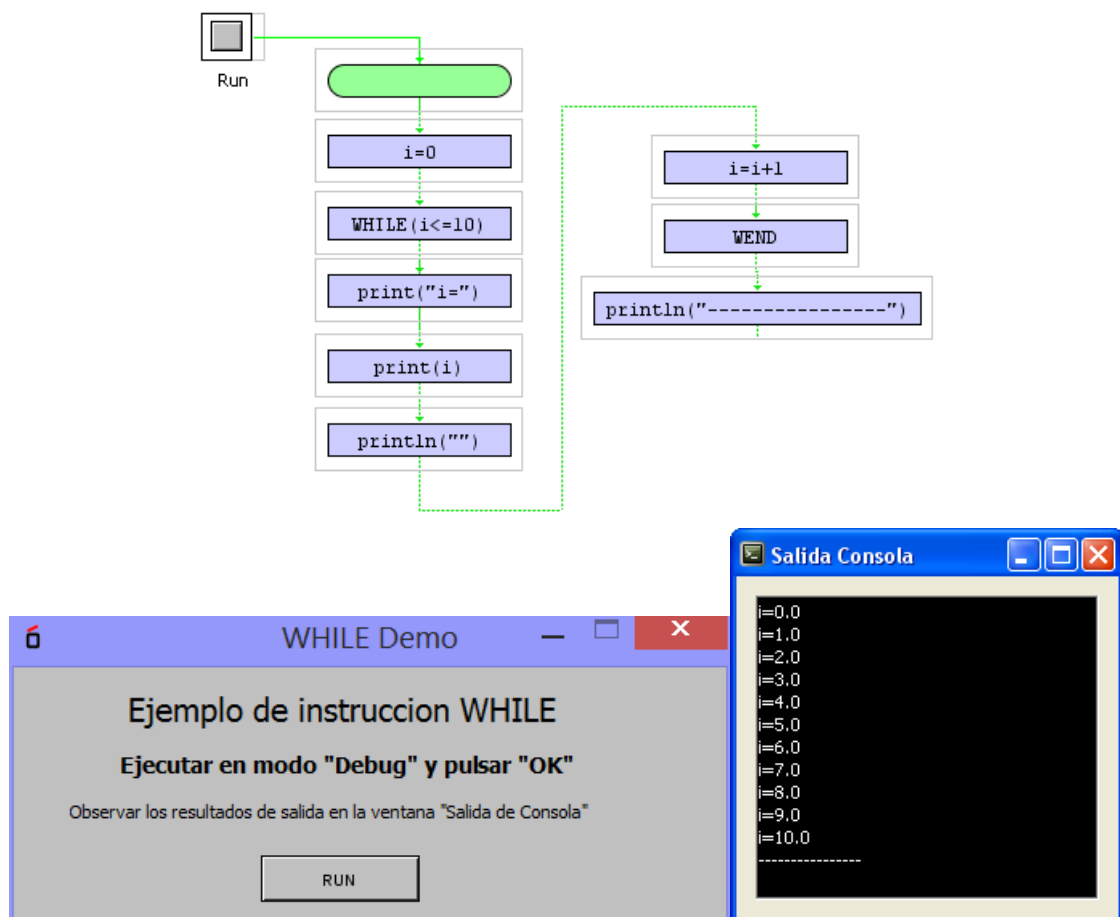
El bloque de instrucción **WHILE** es parecido al anterior FOR NEXT, en este caso se trata de encerrar una serie de instrucciones en una estructura tipo “bucle” y establecer una condición de salida. Si no se cumple la condición de salida se estarán ejecutando las instrucciones de manera repetitiva.

En el ejemplo que mostramos se trata de que se realice una cuenta ascendente de una variable **i** poniendo como condición de parada que el valor de esta sea **i ≤ 10**. Las instrucciones que vamos a realizar dentro del bucle serán simplemente imprimir el valor de **i**.

Esta instrucción se debe cerrar siempre con el bloque “**WEND**” que viene a ser el fin del bucle.

Variables: **i** Contador(double)

Solución: *While_Wend1.vlogic*

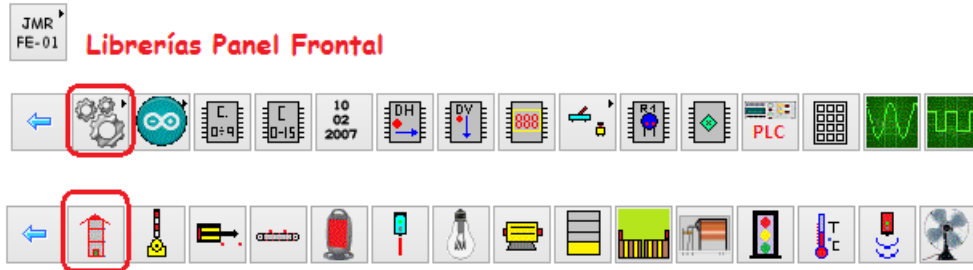


En el ejemplo *While_Wend2.vlogic* se tiene una variación de éste, que consiste en recoger la variable de control desde fuera como variable “**a**”

NIVEL III: EJEMPLOS PROPUESTOS

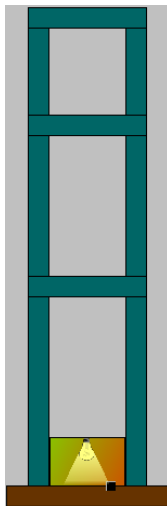
ASCENSOR

1. Utilizando el componente “Ascensor” de la librería de componentes de “Panel Frontal”



realiza un programa para que gobierne el ascensor de acuerdo a la siguiente secuencia.

ASCENSOR



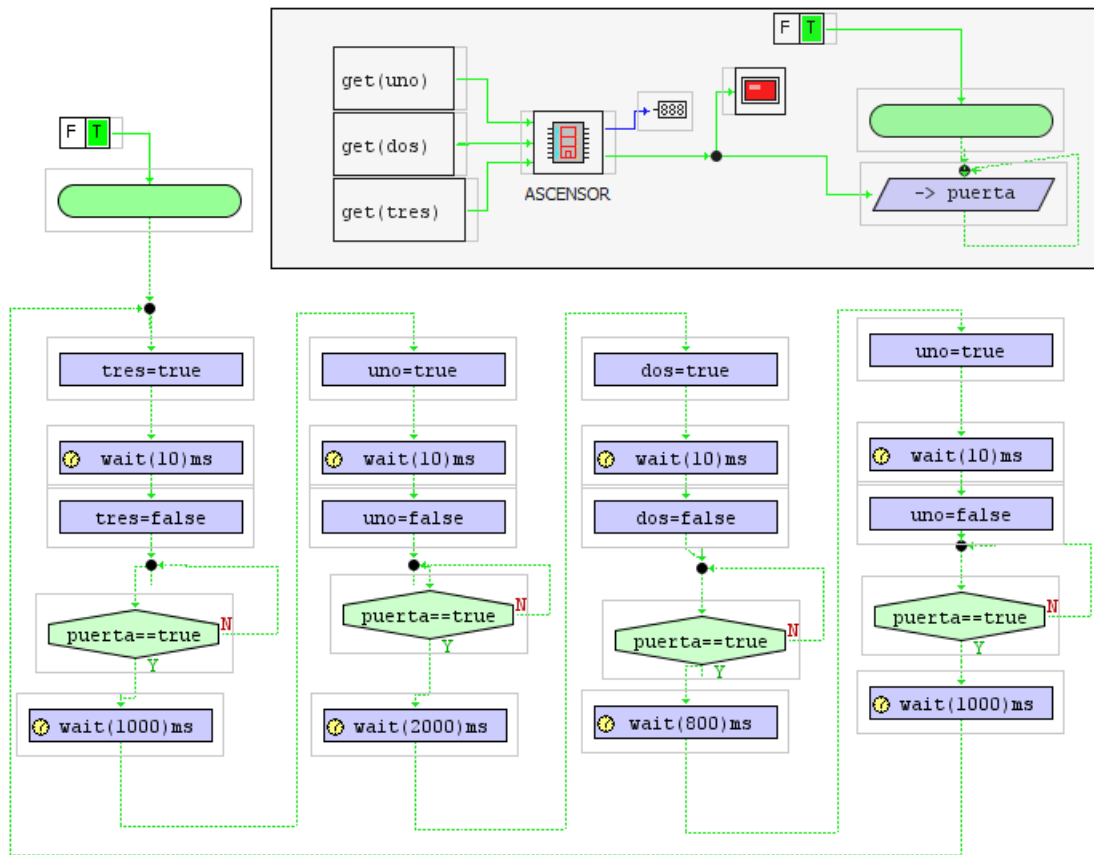
1. Que vaya al piso 3 y espere un tiempo $T1=1$ seg
2. Que vaya al piso 1 y espere un tiempo $T2=2$ seg.
3. Que vaya al piso 2 y espere un tiempo $T3=0.8$ seg.
4. Que vaya al piso 1 y espere un tiempo $T4=1$ seg.
5. Que vuelva al primer punto

Señales a utilizar:

- Entrada **puerta** (de tipo boolean) indica el número de piso en el que se encuentra el ascensor.
- Entradas **uno, dos tres** (booleanas) indican el piso al que enviamos el ascensor.

El Componente ascensor tiene una salida que indica el Numero de Piso y esta salida es la que debemos utilizar en los condicionales que pongamos a la hora de realizar el programa.

Solución: *ascensor.vlogic*



PARKING

2. Diseñar un Parking de acuerdo a las siguientes características:

Los coches al entrar tienen que recoger un ticket junto a la “barrera de entrada” e inmediatamente que lo recojan se levantará esta dejando pasar el coche. A la entrada habrá un semáforo con dos lámparas una verde (libre) y otra roja (lleno).

Se dispondrá de un contador de coches que nos indicará en todo momento los coches que hay dentro. Este contador se debe incrementar cada vez que llega un coche y decrementar cada vez que sale. La salida del coche se detectará con un sensor y se subirá la “barrera de salida”. En el parking caben 11 coches

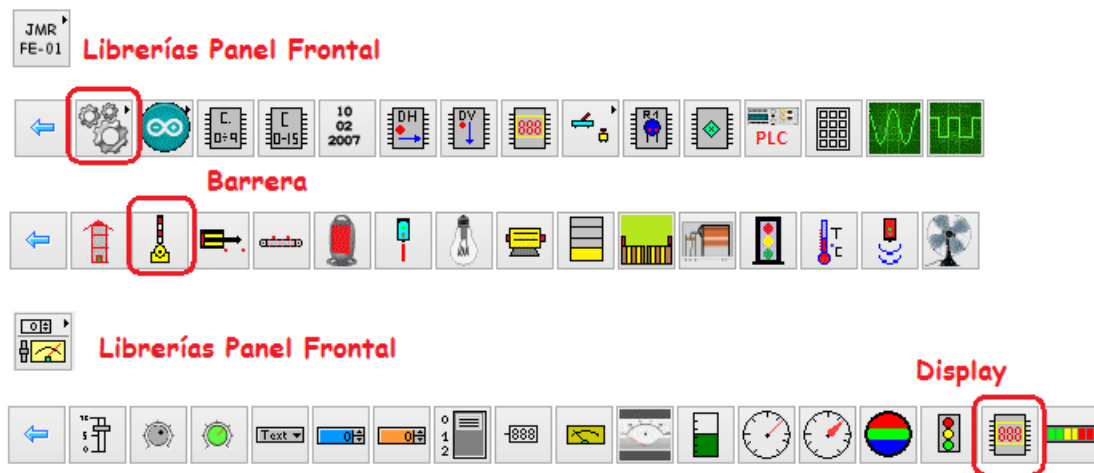
La actuación de las barreras se simplificará de tal manera que cuando se recibe la orden de subir (sensor de entrada o sensor de salida) se suben y transcurrido 1,5 seg. Se bajan.

Señales a tener en cuenta:

Sensor de entrada de coche
 Sensor de salida de coche
 Numero de coches dentro del parking
 Parking lleno
 Sube barrera de entrada
 Sube barrera de salida

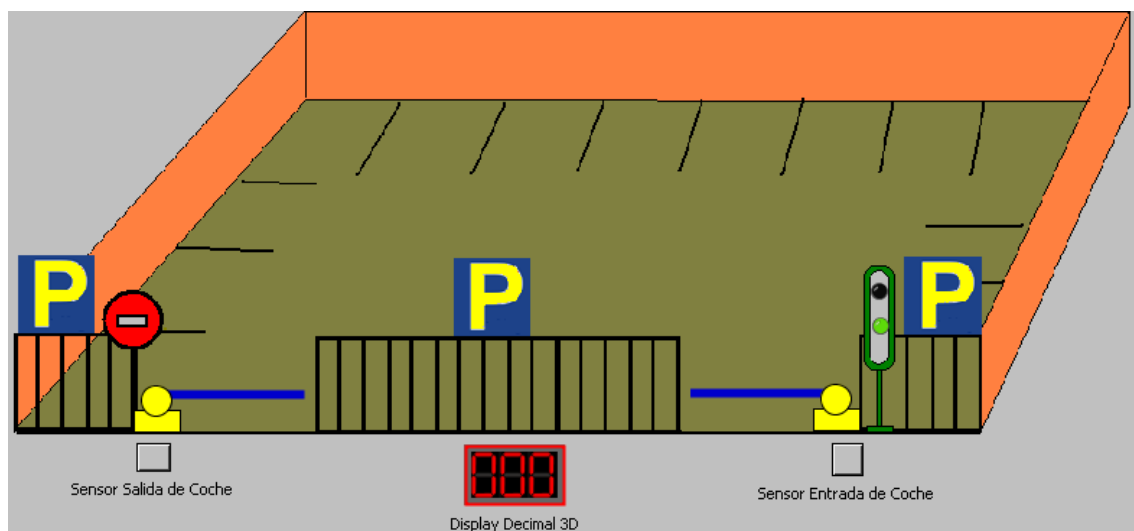
se (booleana)
ss (booleana)
coches (double)
lleno (booleana)
sbe (booleana)
sbs (booleana)

Librerías especiales que se deben utilizar:

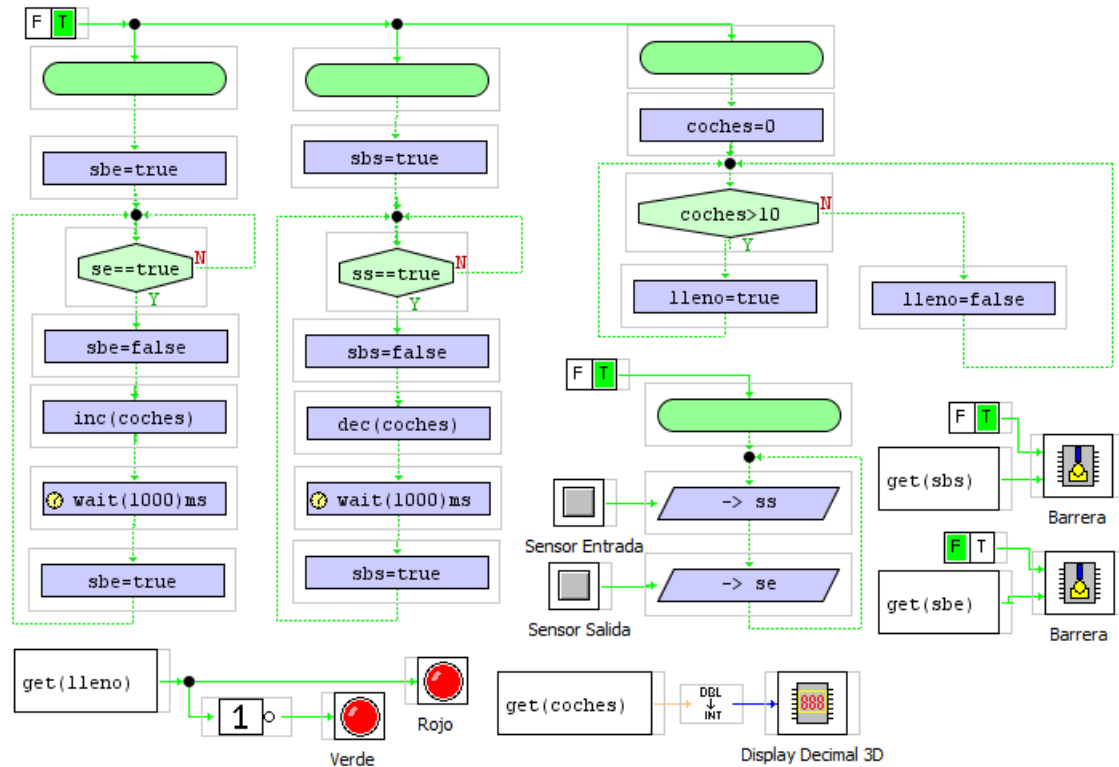


Se podrá dibujar el esquema del parking más o menos como el que se pone en la solución.

Solución: *parking.vlogic*



Esquema



Comentario a la solución.

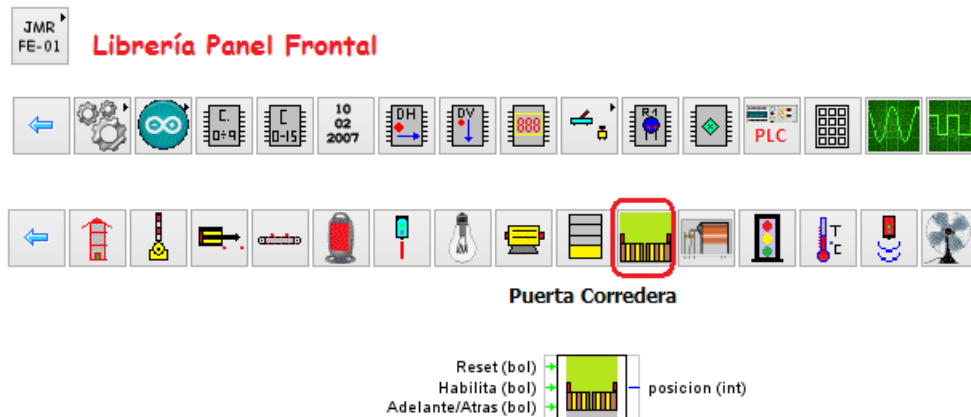
Obsérvese que se han realizado varios diagramas de flujo con el fin de facilitar la comprensión del funcionamiento. No debemos olvidar que se pueden ejecutar varios diagramas a al vez.

Diagramas realizados:

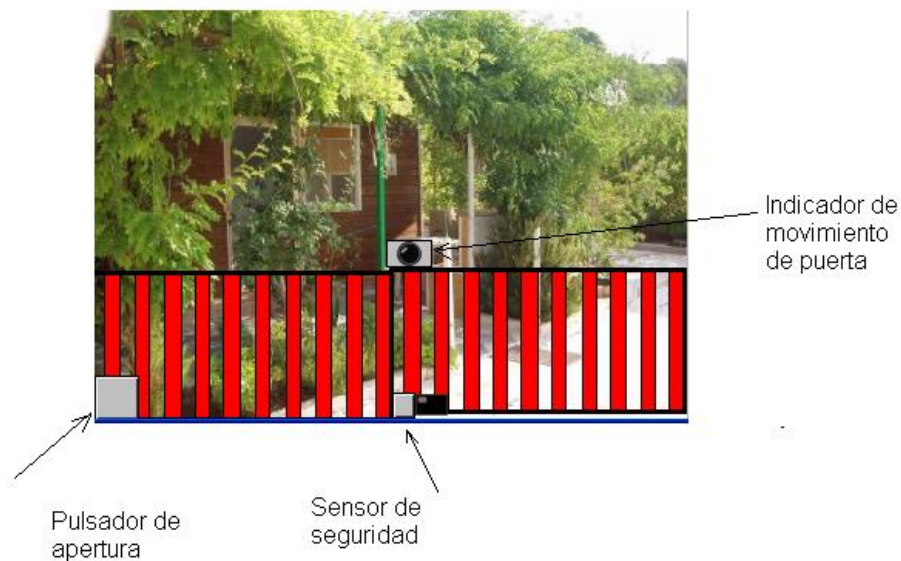
- Tratamiento de entrada de coche (se testea la señal se)
- Tratamiento de salida de coche (se testea la señal ss)
- Lectura permanente de los sensores se y ss
- Realización del conteo de los coches (entrantes y salientes)

PUERTA DE ENTRADA A UNA FINCA

3. Se trata de diseñar el automatismo de una puerta de una finca haciendo uso de la librería que representa una puerta que se desplaza sobre un carril y es gobernada de acuerdo con las señales que se indican en la siguiente figura.



Se colocara de fondo una imagen que represente una casa o finca para darle más realismo a la simulación



La forma de actuar debe ser la siguiente: Cuando se pulsa en el “Pulsador de llamada” la puerta comienza a abrirse (desplazamiento a la izquierda) hasta que se abre del todo. Una vez abierta estará un tiempo y comenzará la fase de cierre. Si cuando esta cerrándose la puerta se interfiere el sensor de seguridad (célula infrarroja) automáticamente la puerta se detendrá hasta que desaparezca esta señal de seguridad y pueda continuarse el cierre.

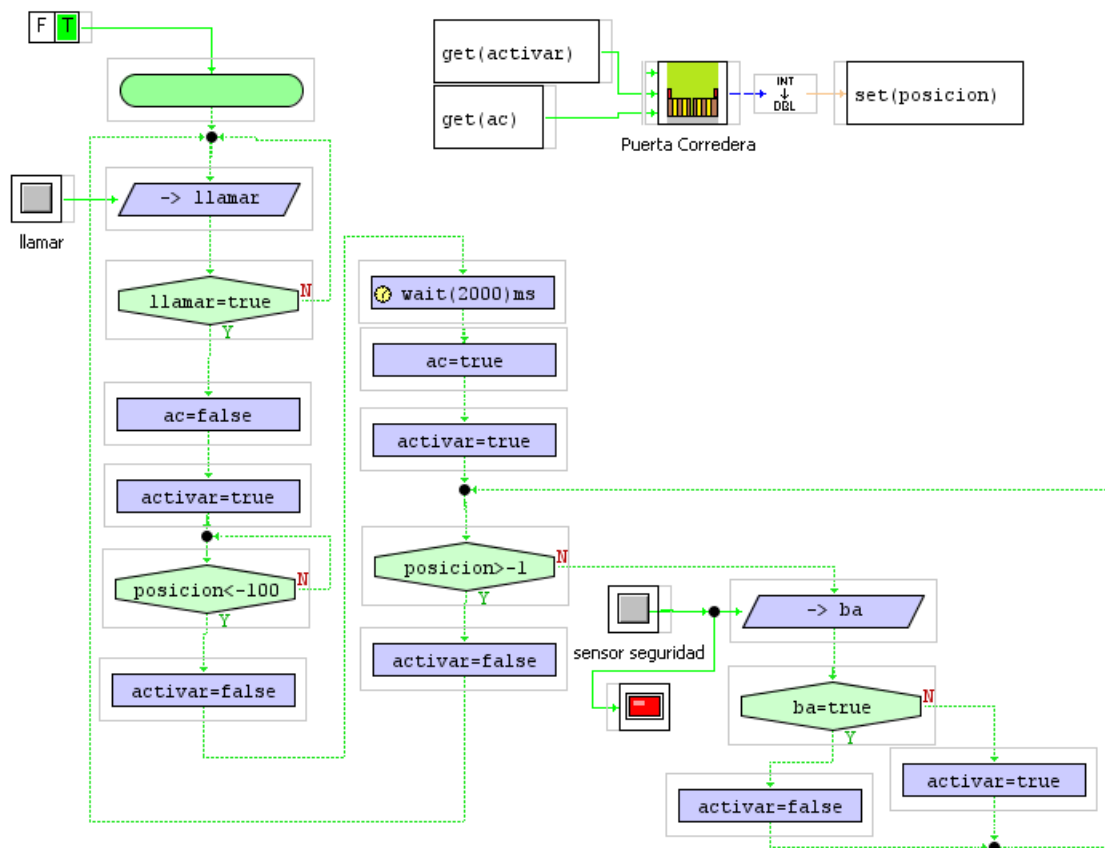
Las señales a tener en cuenta son:

- **llamar** (boolean) inicia el ciclo de apertura de la puerta.

- **activar** (booleana) activa el movimiento de la puerta.
- **posición** (double) nos indica la posición en la que se encuentra la puerta.
- **ac** (boolean) da la orden del sentido de movimiento de la puerta (Adelante/Atrás)
- **ba** (boolean) sensor de seguridad de la puerta, se activa cuando hay algún obstáculo

La señal de posición debemos considerarla para que en el movimiento de apertura se detenga en un punto (posición)

Solución: *puerta_casa.vlogic*



MÁQUINA DE CAFÉ

4. Se trata de realizar la simulación del funcionamiento de una máquina expendedora de bebidas (café, te y manzanilla).

Para su funcionamiento se deben cumplir las siguientes condiciones:



El ciclo de trabajo se iniciará cuando coloquemos una moneda en el lugar correspondiente (pulsador moneda). A continuación se deberá colocar un vaso en el lugar correspondiente lo cual hará que se produzca un impulso en el sensor de vaso (pulsador vaso). A continuación debemos seleccionar una de las tres opciones de bebida a suministrar: té, café o manzanilla. Se pulsará el correspondiente pulsador y se activará la electro válvula de salida de la bebida que se mantendrá activada un tiempo de 2 seg. Transcurrido este tiempo la máquina debe estar dispuesta a realizar otro ciclo de suministro de bebida.

Señales a tener en cuenta:

moneda (boolean) pulsador que simula la entrada de la moneda.

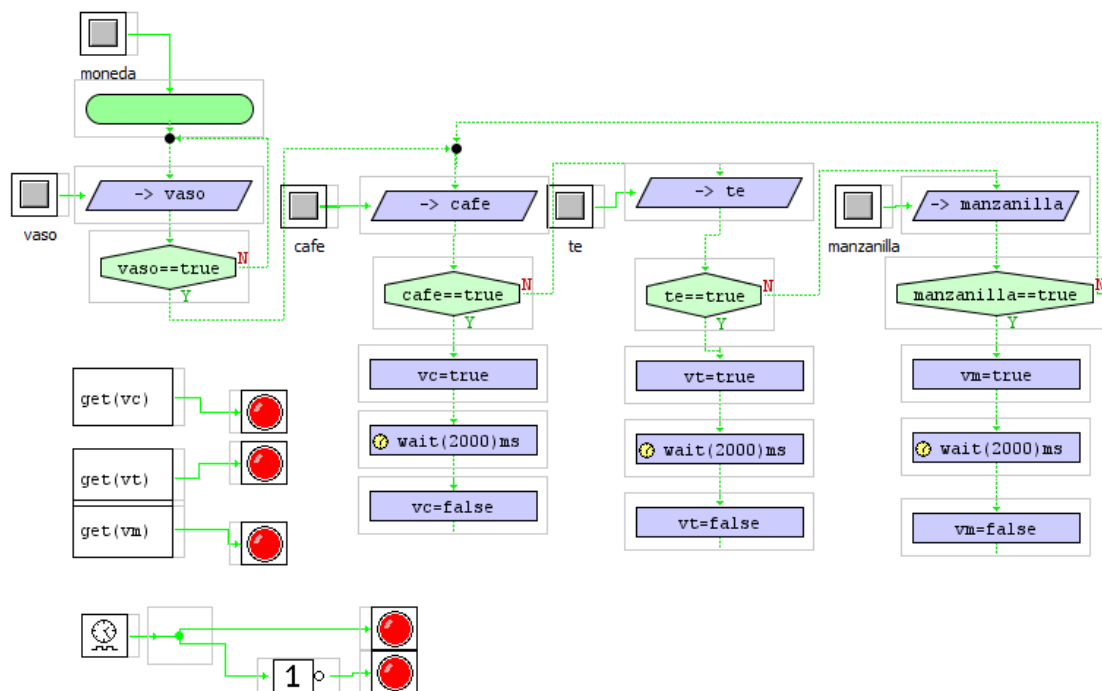
vaso (boolean) pulsador que simula la existencia de un vaso.

café (boolean) pulsador de petición de café

te (boolean) pulsador de petición de te

manzanilla (boolean) pulsador de petición de manzanilla

Solución:: *bebidas.vlogic*



GASOLINERA

5. Se trata de elaborar la simulación de una gasolinera de acuerdo con las siguientes condiciones de funcionamiento:

Se expenderá solo gasolina de un tipo. Dispondrá de un depósito en el que se almacenará el combustible, y un contador en el que veremos como se van marcando los litros que nos está suministrando el sistema.

Para realizar un servicio de deberá marcar la cantidad de litros que queremos y se pulsará un botón para iniciar la descarga de la gasolina en el vehículo.

Deberemos llevar el control del nivel de gasolina del depósito de tal manera que cada vez que nos servimos debemos restar la cantidad de litros extraída del total almacenado. Tendremos también un sistema de recargar la gasolina al depósito de tal manera que podremos marcar la cantidad de gasolina que almacenamos cuando llega el camión cisterna de reparto.

Si el nivel de gasolina del deposito es menor de 10 litros o la cantidad que queremos echar a nuestro depósito es superior a la que queda en el depósito de la gasolinera el sistema no nos permitirá el suministro. Desactivándose una señal que llamaremos “servicio”.

Variables a definir:

cs	cantidad de litros que queremos echar en nuestro coche
cr	cantidad de litros que añadimos al tanque de almacenamiento cuando llega la cisterna de reparto de gasolina.
servicio	Nos indicará si disponemos de servicio en el aparato de dispensar la gasolina
valvula	Válvula que abre el paso de la gasolina para llenar en nuestro coche.
nivel	Nivel de gasolina que hay en el tanque de distribución.
csumins	Cantidad de gasolina que metemos en el tanque de distribución cuando llega el camión cisterna de reparto

Los botones de acción que vamos a utilizar serán:

Botón “**Repostar**” orden de servir la gasolina
Botón “**Recargar**” cargar el tanque de distribución

Las entradas de valor numérico a utilizar serán:

Entrada **cs** de cantidad de litros a repostar (variable cs)
Entrada **rs** cantidad de litros a almacenar en el tanque de distribución

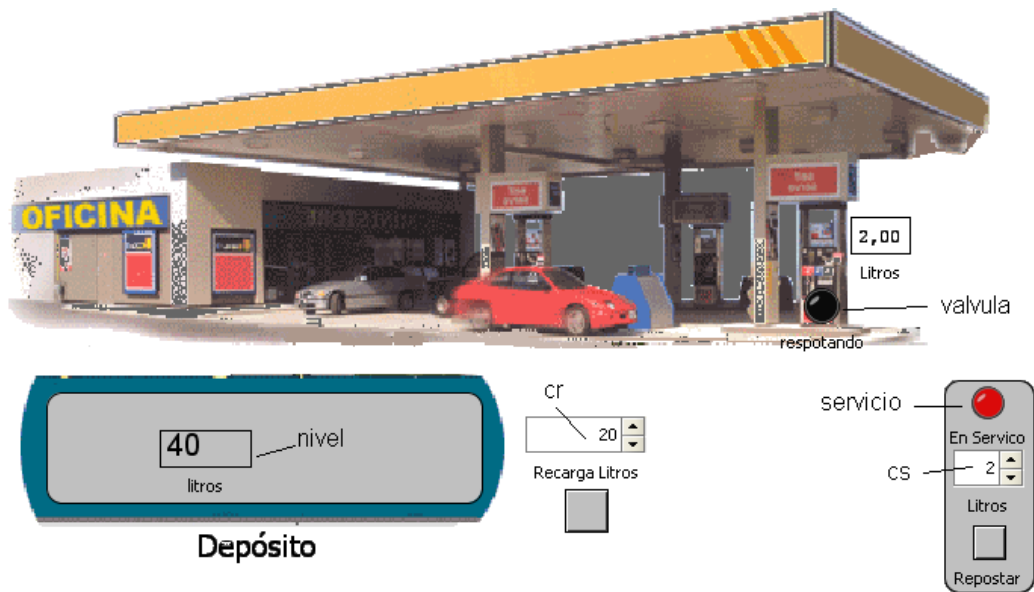
Salidas de valor numérico:

Valor **nivel** : nivel del tanque
Valor **csumins**: los litros que se van sirviendo a nuestro vehículo

Indicadores de tipo LED

Indicador de **servicio** (variable servicio) que nos dice si el sistema está en disposición de darnos gasolina.

Indicador de **válvula** de llenado (variable válvula) nos indica manteniéndose encendido el tiempo transcurrido hasta que se realiza el suministro de la cantidad solicitada.



Solución: gasolinera.vlogic

