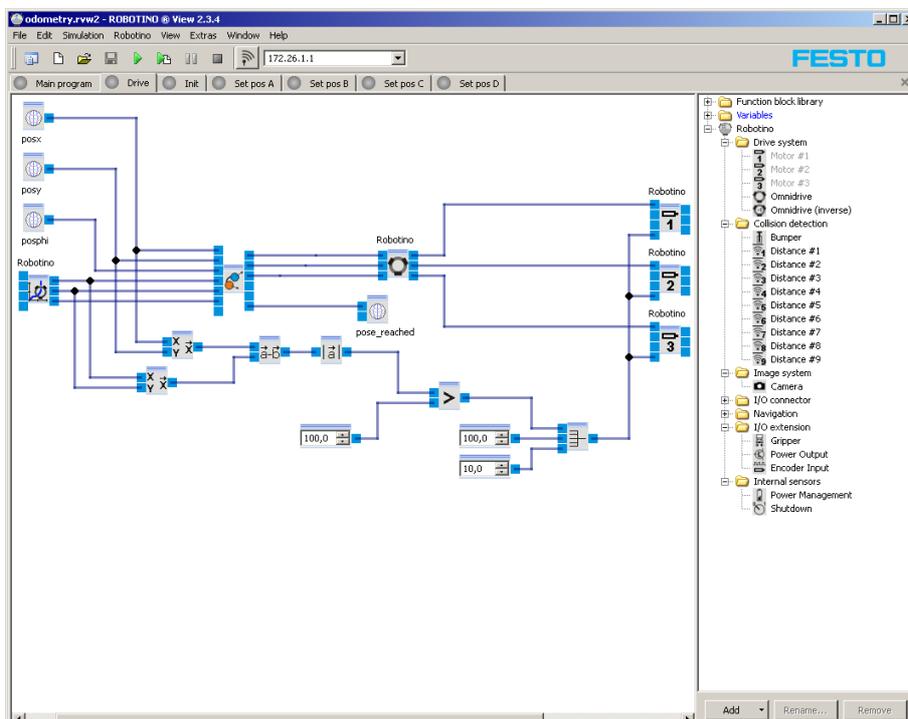


FESTO

Robotino® View 2



Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts verboten, soweit nicht ausdrücklich gestattet. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten, insbesondere das Recht, Patent-, Gebrauchsmuster- oder Geschmacksmusteranmeldungen durchzuführen.

The copying, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation is prohibited. Offenders will be held liable for the payment of damages. All rights reserved, in particular the right to carry out patent, utility model or ornamental design registration.

Sin nuestra expresa autorización, queda terminantemente prohibida la reproducción total o parcial de este documento, así como su uso indebido y/o su exhibición o comunicación a terceros. De los infractores se exigirá el correspondiente resarcimiento de daños y perjuicios. Quedan reservados todos los derechos inherentes, en especial los de patentes, de modelos registrados y estéticos.

Toute diffusion ou reproduction du présent document, de même que toute exploitation ou communication de son contenu sans l'accord express de l'auteur est proscrite. Toute contravention pourra donner lieu à des demandes de dommages et intérêts. Tous droits réservés, notamment en termes de demande de brevet, de modèle déposé et de protection par dessin ou modèle.

© Festo Didactic GmbH & Co. KG, 73770 Denkendorf, Germany, April 2010
Internet: www.festo-didactic.com
e-mail: did@de.festo.com

Inhalt / Contents / Contenido / Sommaire

1	Bienvenida	8
1.1	Mejoras	8
1.2	Instalación, Actualización y Desinstalación	9
1.3	Cambio de idioma	9
2	Familiarización con el espacio de trabajo	9
2.1	Estructura y concepto del interface de usuario	9
2.1.1	Barra de herramientas	11
2.1.2	Librería de bloques de función	12
2.2	Terminología	13
3	Utilización de Robotino® View	13
3.1	Crear un nuevo proyecto	14
3.2	Cargar un proyecto existente	14
3.3	Insertar bloques de función en un subprograma	14
3.4	Interconexión de bloques de función	15
3.5	Variables globales	15
3.6	Ejecutar un subprograma	16
3.7	Iniciar el programa principal	17
3.8	Conectar con Robotino®	18
3.9	Atajos de teclado	19
3.10	Conversión de tipos	20
3.11	Actualizaciones	20
3.12	Cargar proyectos a Robotino y ejecutarlos.	20
3.12.1	Explorar Robotino	21
3.12.2	Cargar y ejecutar	22
3.13	Actualizar paquetes del Robotino	23
3.13.1	Instalación del Robotino firmware	25
3.13.2	Interna	26
4	Ejemplos	26
4.1	Programas de control	26
4.1.1	Tutorial 2	27
4.2	Lógica	34
4.2.1	Multiplexor	34
4.2.2	FlipFlop	34
5	Librería de bloques de función	35
5.1	Lógica	35
5.1.1	Contador incremental	35
5.1.2	Contador decremental	39
5.1.3	Multiplexor	40
5.1.4	Desmultiplexor	41
5.1.5	AND	42
5.1.6	AND FL	44
5.1.7	NAND	46
5.1.8	NAND_FL	48
5.1.9	OR	49
5.1.10	XOR	51
5.1.11	NOT	52
5.1.12	NOR	52

5.1.13	Relé de enclavamiento	54
5.1.14	Elemento de muestreo y retención	55
5.2	Matemáticas	55
5.2.1	Operaciones aritméticas	55
5.2.2	Operaciones de comparación	60
5.2.3	Funciones	62
5.2.4	Matrices	69
5.3	Cálculo vectorial	71
5.3.1	Operaciones con vectores	72
5.3.2	Operaciones con elementos	75
5.3.3	Transformaciones	77
5.4	Visualizador	79
5.4.1	Osciloscopio	80
5.4.2	Pantalla de datos del telémetro láser	81
5.5	Procesamiento de imágenes	82
5.5.1	Segmentador	82
5.5.2	Extractor de segmentos	86
5.5.3	Detector de líneas	88
5.5.4	ROI	90
5.5.5	Información de la imagen	92
5.5.6	Conversión de espacio de color	94
5.6	Generadores	94
5.6.1	Generador de forma de onda arbitraria	95
5.6.2	Constante	96
5.6.3	Temporizador	97
5.6.4	Generador aleatorio	98
5.7	Filtro	99
5.7.1	Filtro de valor medio	99
5.8	Navegación	99
5.8.1	Controlador de posición	100
5.8.2	Pose constante	104
5.8.3	Componer pose	105
5.8.4	Descomponer pose	106
5.8.5	Componer ruta	107
5.8.6	Descomponer ruta	107
5.8.7	Controlador de ruta	109
5.8.8	Evasión de obstáculos	117
5.9	Dispositivos de entrada	119
5.9.1	Panel de control	119
5.9.2	Corredera	121
5.10	Intercambio de datos	122
5.10.1	Lector de imagen	122
5.10.2	Grabador de imágenes	124
5.11	Variables	125
6	Dispositivos	125
6.1	Añadir y editar	125
6.2	Mostrar diálogos	126
6.3	Robotino	127
6.3.1	Barra de herramientas	127
6.3.2	Diálogo	128
6.3.3	Boques de función	128
6.4	Joystick	154
6.4.1	Diálogo	154
6.4.2	Boques de función	155

6.5	Cámara local	156
6.5.1	Diálogo	156
6.5.2	Boques de función	157
6.6	Cliente OPC	158
6.6.1	Diálogo	159
6.6.2	Boques de función	161
6.7	Intercambio de datos	163
6.7.1	Servidor	163
6.7.2	Cliente	166
6.7.3	Boques de función	170
6.8	Intercambio de datos UDP	170
6.8.1	Protocolo	170
6.8.2	Diálogo	174
6.8.3	Boques de función	175
6.8.4	Ejemplo	176
7	Programación	176
7.1	Mis bloques de función	177
7.1.1	Tutorial 1	178
Index	181

1 Bienvenida

Robotino® View es el entorno de programación gráfica intuitivo de Robotino®. Robotino® View le permite crear y ejecutar programas de control para Robotino®.

1.1 Mejoras

Robotino® View 2 combina conceptos operativos modernos, posibilidad de ampliación por el usuario y un uso intuitivo. Todas las innovaciones mantienen muchos de los aspectos positivos conocidos de Robotino® View 1. El usuario que esté familiarizado con Robotino® View reconocerá muchas de las características de la versión anterior. Estas son, por ejemplo, la librería de bloques de función o la barra de herramientas con la que establece la conexión con Robotino. A primera vista Robotino® View 2 se parece mucho a su predecesor.

¿Qué características se mantienen?

- Los programas se diseñan como programas de control de flujo. La librería de bloques de función contiene las unidades a partir de las cuales se construye el gráfico del flujo de datos.
- La conexión con el Robotino se establece a través de la barra de herramientas.

¿Qué hay de nuevo?

- El programa de control de secuencia es reemplazado por un programa de control "real" conocido de la programación de PLC según DIN EN 61131.
- Robotino® View 2 no sólo es capaz de controlar un Robotino, sino cualquier dispositivo y en cantidades ilimitadas. Por ejemplo, puede controlarse cualquier número de Robotinos simultáneamente desde un proyecto Robotino View.
- Los subprogramas pueden ser reutilizados dentro del programa principal.
- Los subprogramas pueden ser importados desde diferentes proyectos.
- El usuario puede diseñar e implementar bloques de función personalizados que se hayan cargado en la librería de bloques de función.
- El usuario puede diseñar e implementar dispositivos personalizados y cargarlos como Plugin en el administrador de dispositivos.

Cambios en esta versión:

- El administrador de dispositivos ha sido integrado en la librería de bloques de función. Véase [Añadir y editar dispositivos](#)^[125].
- Los proyectos pueden cargarse al Robotino y también ser ejecutados en el Robotino (Se necesita la tarjeta Robotino CF Versión 2.0). Véase [cargar proyectos](#)^[20].
- Nuevos dispositivos para el intercambio de datos a través de la red. Véase [Dispositivos para intercambio de datos](#)^[163].

1.2 Instalación, Actualización y Desinstalación

Deberá tener derechos de administrador para poder instalar Robotino® View.
Para instalar Robotino® View siga las instrucciones de los cuadros de diálogo.
Si van a utilizar Robotino® View usuarios sin derechos de administrador, necesitarán incluir los programas liberados de las restricciones en Windows® XP en el centro de seguridad para el ajuste del firewall de Robotino® View (Port 80 y 8080).

1.3 Cambio de idioma

Robotino® View reconoce automáticamente el idioma seleccionado en su sistema operativo Windows® y selecciona la correspondiente traducción de Robotino® View.

Puede cambiar el ajuste automático en cualquier momento a través del elemento de menú Extras ▶ Idiomas. El nuevo ajuste surge efecto inmediatamente.

2 Familiarización con el espacio de trabajo

Una vez que se haya familiarizado con el espacio de trabajo y las denominaciones utilizadas en Robotino® View, será más fácil para usted seguir el resto de la documentación.

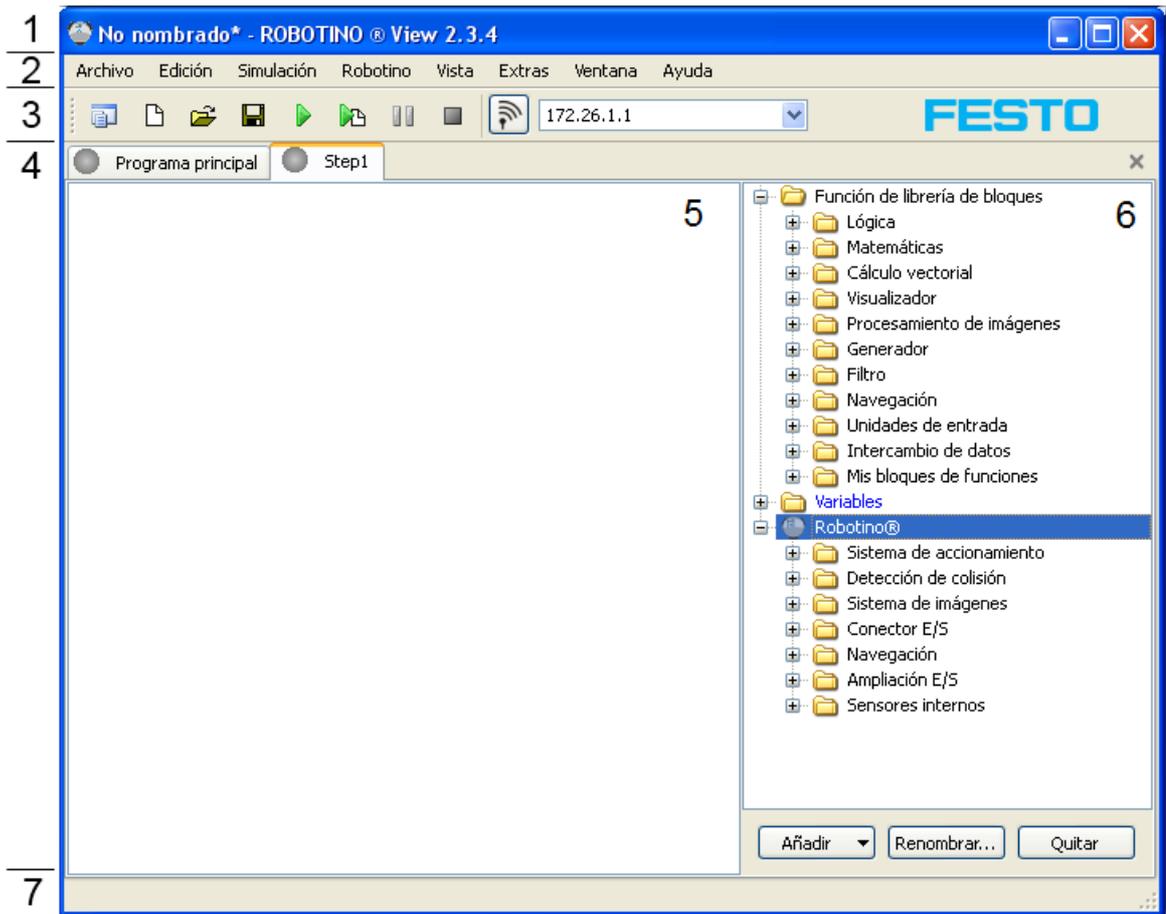
En esta sección aprenderá más acerca de:

- El diseño y concepto del interface de usuario,
- La terminología utilizada en Robotino® View.

2.1 Estructura y concepto del interface de usuario

Al poner en marcha Robotino® View se abre un proyecto vacío con un dispositivo "Robotino". El proyecto ocupa todo el espacio de trabajo.

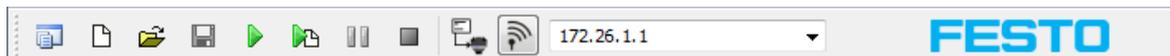
Familiarización con el espacio de trabajo



Número	Nombre	Descripción
1	Barra de título	<ul style="list-style-type: none"> Muestra el nombre del proyecto actual (No nombrado). Si en el proyecto se han realizado cambios sin guardar, al nombre del proyecto le sigue un asterisco (*). Al lado del nombre del proyecto se muestra el nombre y la versión de la aplicación (aquí Robotino® View 2.3.4). Botones habituales para minimizar, maximizar y cerrar la aplicación.
2	Barra de menú	Menús para Abrir/Guardar, Edición, Vista ...
3	Barra de herramientas	<ul style="list-style-type: none"> Botones de acceso rápido a las funciones de los menús. Botones de iniciar y detener la simulación. Campo de entrada para la dirección IP del Robotino y botón de conexión (véase Barra de herramientas de Robotino (127)). Logo de Festo con enlace a la página de inicio de Festo.

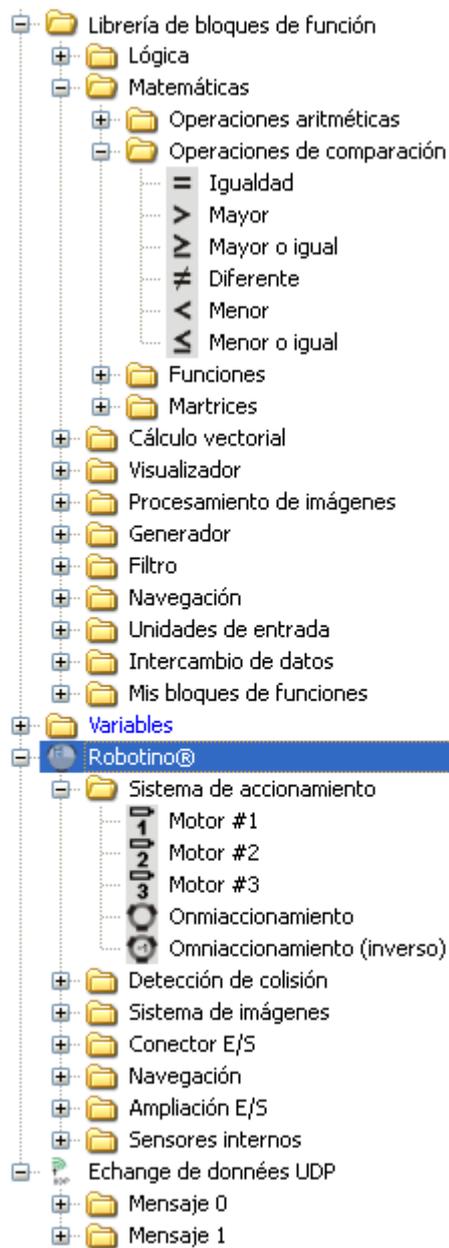
4	Pestañas de selección de programas	Aquí puede cambiar entre el programa principal y los subprogramas de un proyecto. En este momento, se halla visible el subprograma "Step1".
5	Espacio de trabajo del subprograma	Aquí se muestra y se edita el subprograma. Obviamente, en un nuevo proyecto el subprograma "Step1" se halla vacío.
6	Librería de bloques de función ^[12]	Aquí se muestran los bloques de función disponibles para la programación,.
7	Barra de estado	Muestra información acerca del estado del proyecto y de la aplicación.

2.1.1 Barra de herramientas



	Crear un nuevo proyecto
	Crear un nuevo subprograma
	Abrir un proyecto existente
	Guardar el proyecto actual
	Iniciar programa principal
	Iniciar el programa actualmente visible
	Suspender la simulación
	Detener la simulación
	Cargar el proyecto actual al Robotino
Entrada de dirección IP y botón de conexión	véase Barra de herramientas de Robotino ^[12]
	Logo de Festo con enlace a la página de inicio de Festo.

2.1.2 Librería de bloques de función



La carpeta de Librería de bloques de función contiene los bloques de función que están disponibles en cada proyecto. Actualmente visibles son los bloques de función Igualdad, Mayor ... Menor o igual de la subcarpeta Operaciones de comparación.

La carpeta Robotino@ contiene los bloques de función proporcionados por el dispositivo "[Robotino@](#)". Un nuevo proyecto siempre contiene un "[Robotino@](#)". Los bloques de función actualmente visibles "[Motor1](#)" hasta "[Omniaccionamiento \(inverso\)](#)" de la carpeta "[Sistema de accionamiento](#)".

La carpeta Variables contiene bloques de función para leer y escribir Variables globales.

Puede añadir bloques de función arrastrándolos y soltándolos (Drag&Drop) al subprograma actual.

Los bloques de función de dispositivos están limitados a los recursos concretos de hardware. "Motor1^[129]" está disponible una sola vez en Robotino. Por ello, sólo puede añadirse "Motor1^[129]" una vez a un subprograma. Si "Motor1^[129]" ya ha sido añadido a un subprograma, el icono de "Motor1^[129]" aparece en gris en la librería de bloques de función.

2.2 Terminología

Boques de función	Unidad funcional mínima de la que puede estar formado un subprograma. Conectando varios bloques de función, es posible obtener movimientos complejos del Robotino.
Subprograma	Los bloques de función están interconectados en un subprograma.
Programa principal	Programa de control escrito como diagrama de bloques secuencial que une los subprogramas.
Proyecto	Un proyecto consiste en un programa principal y varios subprogramas. Los proyectos pueden guardarse y cargarse.
Red	Los bloques de función se unen mediante una o varias redes (conexiones).
Punto de red	Los puntos de red permiten la estructuración y representación gráfica de una red. De un punto de red puede iniciarse una nueva sub-red.

3 Utilización de Robotino® View

Robotino® View se utiliza para crear programas de control para Robotino®. En esta sección aprenderá como:

- crear un nuevo proyecto
- cargar un proyecto existente
- insertar bloques de función en un subprograma
- interconectar bloques de función por medio de redes
- ejecutar un subprograma y el programa principal
- establecer una conexión con el Robotino®

3.1 Crear un nuevo proyecto

Hay dos posibilidades para crear un nuevo proyecto:

- A través de la barra de menú Archivo ▶ Nuevo
- A través de la [barra de herramientas](#)^[11] con el botón "Crear un nuevo proyecto" 

3.2 Cargar un proyecto existente

Hay tres posibilidades para cargar un proyecto existente:

- A través de la barra de menú Archivo ▶ Abrir
- A través de la [barra de herramientas](#)^[11] con el icono "Cargar proyecto desde archivo" 
- Por medio del atajo de teclado Ctrl + O

Los proyectos guardados tienen la extensión .rvw2

3.3 Insertar bloques de función en un subprograma

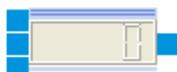
Tras crear un nuevo proyecto o tras cargar un proyecto existente de un archivo, puede empezar a desarrollar su propio programa de control o a modificar uno de existente.

Ejemplo:

Asegurarse de que la vista actual corresponde a un subprograma. Cuando se ha creado un nuevo proyecto, siempre hay un subprograma denominado "Step1". El subprograma "Step1" se muestra después de crear un nuevo proyecto. La [Librería de bloques de función](#)^[12] sólo es visible cuando se visualiza un subprograma.

Desplegar la carpeta "[Lógica](#)^[35]" en la librería de bloques de función. Arrastrar "[Contador incremental](#)^[35]" de la librería de bloques de función y soltarlo en el subprograma.

Desplegar la carpeta "[Generadores](#)^[94]" en la librería de bloques de función. Arrastrar "[Generador de forma de onda arbitraria](#)^[95]" y soltarlo a la izquierda del "[Contador incremental](#)^[35]".



3.4 Interconexión de bloques de función

Uniendo conectores de salida con conectores de entrada de bloques de función, los datos pasan de la salida de unos bloques de función a la entrada de otros bloques de función. La conexión es visualizada por una línea denominada red (network). Una red siempre está conectada a un conector de salida y por lo menos a un conector de entrada.

El siguiente subprograma de ejemplo contiene un "[Generador de forma de onda arbitraria](#)^[95]" y un bloque de función "[Contador incremental](#)^[35]". Conecte la salida del "[Generador de forma de onda arbitraria](#)^[95]" a la entrada superior del "[Contador incremental](#)^[35]".



Haga clic con el botón izquierdo del ratón en el conector de salida del "[Generador de forma de onda arbitraria](#)^[95]". Con ello está creando una línea de red que está unida a la salida del "[Generador de forma de onda arbitraria](#)^[95]" y por el otro extremo al puntero del ratón.

Haciendo clic con el botón izquierdo en cualquier parte del subprograma puede crear un punto de red. Para cerrar la red haga clic en la entrada superior del "[Contador incremental](#)^[35]".

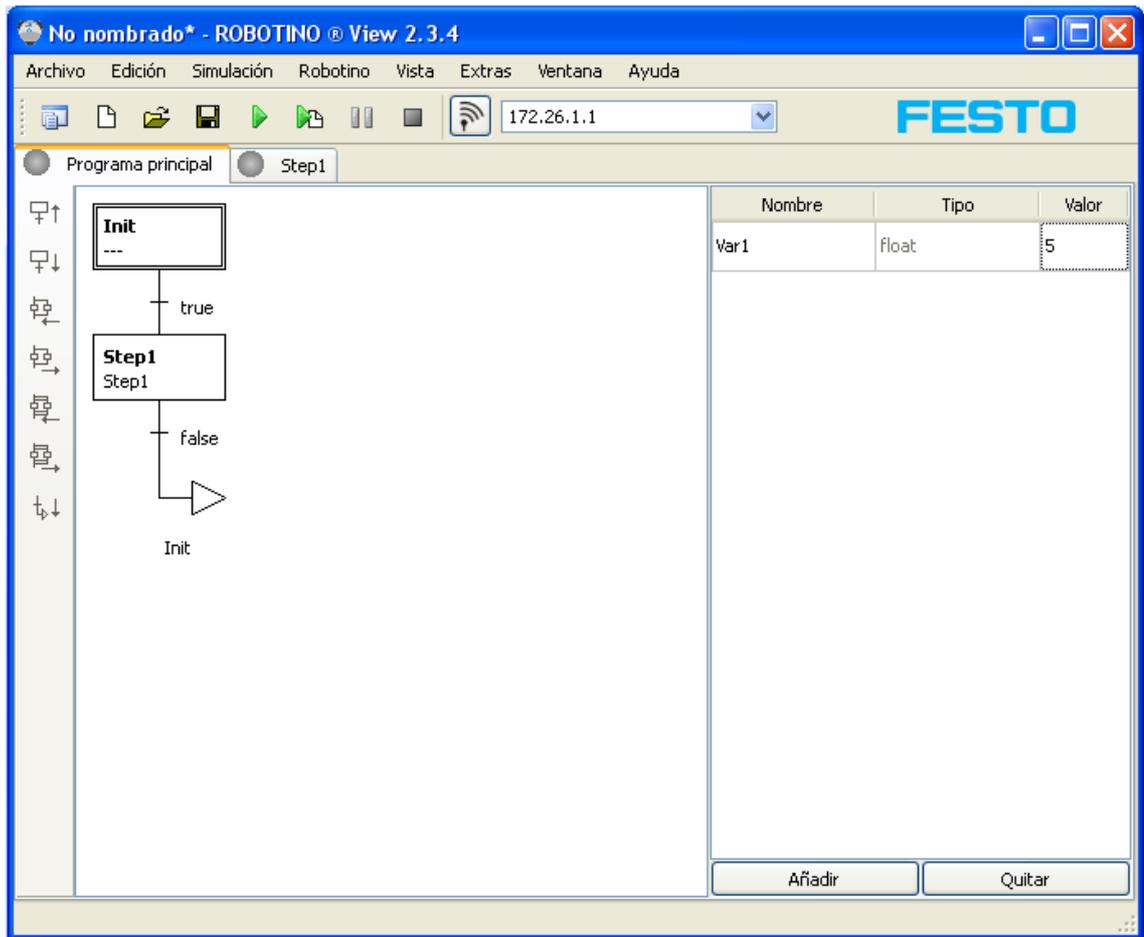
Para borrar un punto de red márkuelo haciendo clic con el botón derecho y elija **Eliminar**.

Para eliminar una línea de red márkuela haciendo clic con el botón izquierdo y pulse **Supr.** Con esto puede eliminar toda la red.

3.5 Variables globales

Las variables globales pueden leerse y escribirse en todos los subprogramas de un proyecto; en el programa principal pueden utilizarse como condiciones de transición.

En la vista de programa principal, el administrador de variables se halla en la lado derecho. Allí es posible añadir, quitar y renombrar variables, así como asignarles valores iniciales.



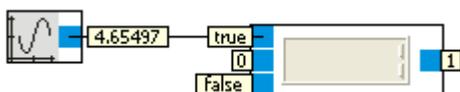
Programa principal con administrador de variables

Las variables globales solamente almacenan números reales o de coma flotante (float). La posibilidad de almacenar otros tipos de datos se incluirá en futuras versiones de Robotino View. Tras crear una variable, se dispone de un bloque escritor y bloques lectores en la librería de bloques de función.

3.6 Ejecutar un subprograma

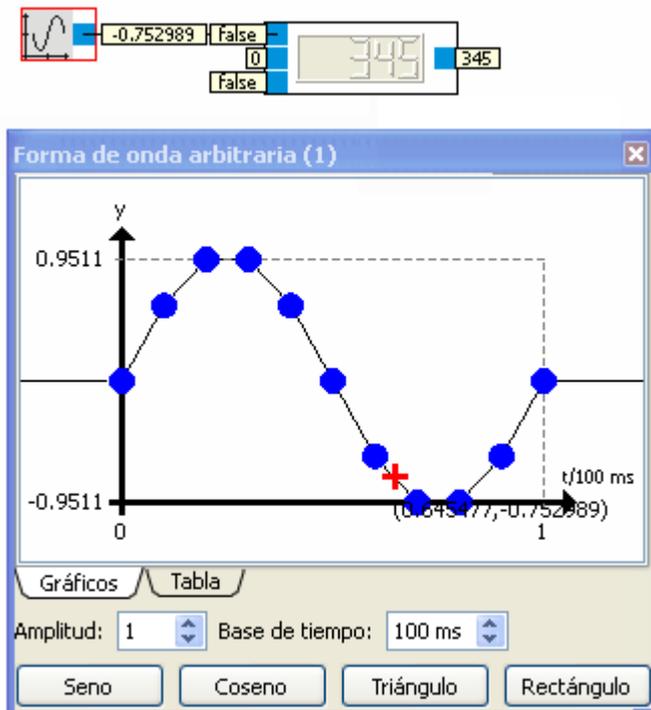
Tras conectar el "[Generador de forma de onda arbitraria](#)^[95]" con el "[Contador incremental](#)^[35]" puede empezar la simulación del subprograma haciendo clic en "Iniciar"  que aparece en la [barra de herramientas](#)^[11].

Pueden visualizarse los valores generados por el "[Generador de forma de onda arbitraria](#)^[95]" y "[Contador incremental](#)^[35]" seleccionando Vista ▶ Mostrar valores de conector o pulsando **Ctrl + D**.



Puede verse el "[Generador de forma de onda arbitraria](#)^[95]" generando valores entre 0 y 10. "[Contador incremental](#)^[35]" incrementa su salida cuando la entrada cambia de false (0) a true (1). Por el momento esto sólo sucede cuando se inicia el subprograma. Véase [conversión de tipos](#)^[20] para saber cómo se convierten los números reales o de coma flotante a valores booleanos. Por otro lado, es muy poco probable que la salida del "[Generador de forma de onda arbitraria](#)^[95]" coincida exactamente con 0.

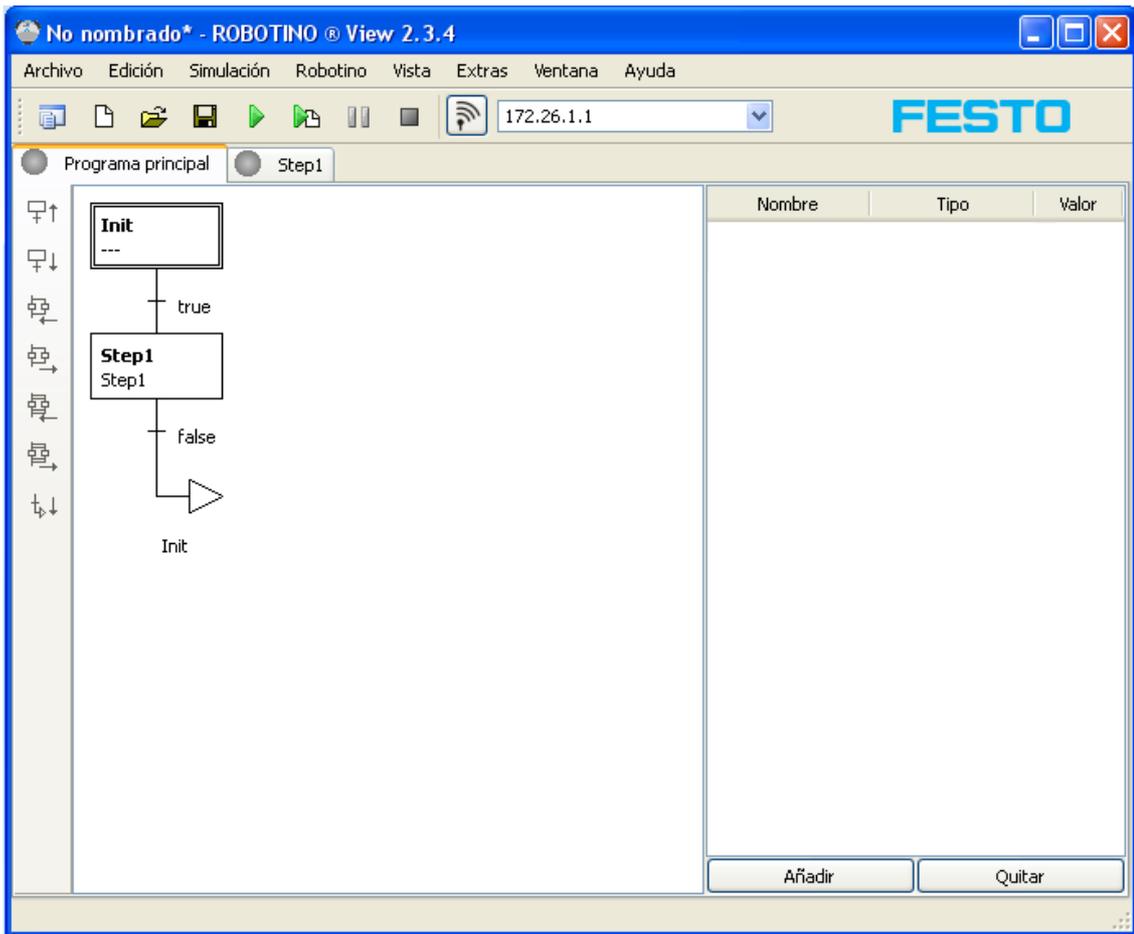
Para ver el contador incrementando su valor, seleccione "Rectángulo" en el diálogo del [Generador de forma de onda arbitraria](#)^[95]. La salida generada se halla ahora en el margen -1 a 1.



3.7 Iniciar el programa principal

Haciendo clic en "Iniciar"  en la [barra de herramientas](#)^[11], se inicia la simulación del programa actualmente visible. Si sólo es visible el programa "Step1", sólo se simulará "Step1" "Step1" es parte del programa principal, que también puede simularse.

Utilice las pestañas de [Selección de programas](#)^[9] para hacer que el programa principal sea el programa actual.



Haciendo clic en "Iniciar"  en la [barra de herramientas](#)^[11] se inicia la simulación del programa principal. El paso inicial sólo funciona una vez, dado que la condición de transición que sigue al paso Init es verdadera (true). Dado que la condición de transición que sigue al Step1 es continuamente falsa (false), se ejecuta el Step1 y el subprograma que lleva asignado (también denominado Step1).

Siempre puede iniciarse la simulación del programa principal, sin importar qué pestaña esté actualmente visible, haciendo clic en el botón "Iniciar programa principal"  en la [barra de herramientas](#)^[11].

3.8 Conectar con Robotino®

Introduzca la dirección IP del Robotino en el campo de dirección IP de la [barra de herramientas](#)^[11]. La dirección predeterminada es 172.26.1.1. Haga clic en el botón de conexión a la izquierda del campo de entrada de la dirección. Si el botón de conexión cambia de gris a verde, significa que se ha establecido la conexión y que hay intercambio de datos entre el Robotino y Robotino View.

3.9 Atajos de teclado

Función	Atajos de teclado
Abrir archivo	Ctrl + O
Guardar archivo	Ctrl + S
Guardar como	May. + Ctrl + S
Salir de Robotino® View	Ctrl + Q
Deshacer	Ctrl + Z
Rehacer	Ctrl + Y o May. + Ctrl + Z
Eliminar selección	Supr
Cortar selección	Ctrl + X
Copiar selección	Ctrl + C
Pegar selección	Ctrl + V
Mover objeto arriba	
Mover objeto abajo	
Mover objeto a izquierda	
Mover objeto a derecha	
Mover vista arriba	Ctrl + 
Mover vista abajo	Ctrl + 
Mover vista a izquierda	Ctrl + 
Mover vista a derecha	Ctrl + 
Borrar selección	Esc
Seleccionar todo	Ctrl + A
Reducir vista	F3
Ampliar vista	May. + F3
Ampliar rejilla	F4
Reducir rejilla	May. + F4
Alternar visibilidad de la librería de bloques de función	Ctrl + L
Alternar mostrar valores de los conectores en los bloques de función	Ctrl + D
Alternar mostrar las descripciones de los conectores en los bloques de función	Ctrl + T

3.10 Conversión de tipos

Tipos datos	de conversión para	implícita	Descripción
int	float, bool		La conversión a bool será cierta si el valor no es 0.
float	int, bool		La conversión a bool será cierta si el valor no es 0.
bool	int, float		Verdadero (true) produce un 1, falso (false) produce un 0.
pose	ruta		Una pose es convertida en una ruta con longitud 1.
ruta	pose		El resultado de la conversión de una ruta a una pose es la primera pose de la ruta. Si la ruta está vacía, la conversión produce una pose no válida.
float	float array		Un número real en coma flotante es convertido a una matriz de números reales con una longitud de 1.

3.11 Actualizaciones

Robotino View dispone de la opción de actualización online. Para verificar la disponibilidad de una nueva versión de software, seleccione "Comprobar actualizaciones" en el menú "Extras". Esta verificación se realiza también automáticamente cada vez que se lanza la aplicación. Si hay disponible una nueva versión, puede descargarse e instalarse automáticamente.

El comportamiento de las opciones de actualización puede configurarse en el menú ("Extras"... ▶ "Preferencias..."). Si sólo puede accederse a Internet a través de un proxy, aquí pueden introducirse la dirección del servidor, puerto, nombre de usuario y contraseña. Pero en redes de empresa, el uso de los ajustes de Internet Explorer suele ser la forma más sencilla.

3.12 Cargar proyectos a Robotino y ejecutarlos.

A partir de Robotino View versión 2.1.0 y Robotino flash card 2.0 es posible cargar proyectos a Robotino a través de FTP y ejecutarlos directamente desde Robotino View. Esta función es accesible a través de: Robotino ▶ Cargar proyecto.

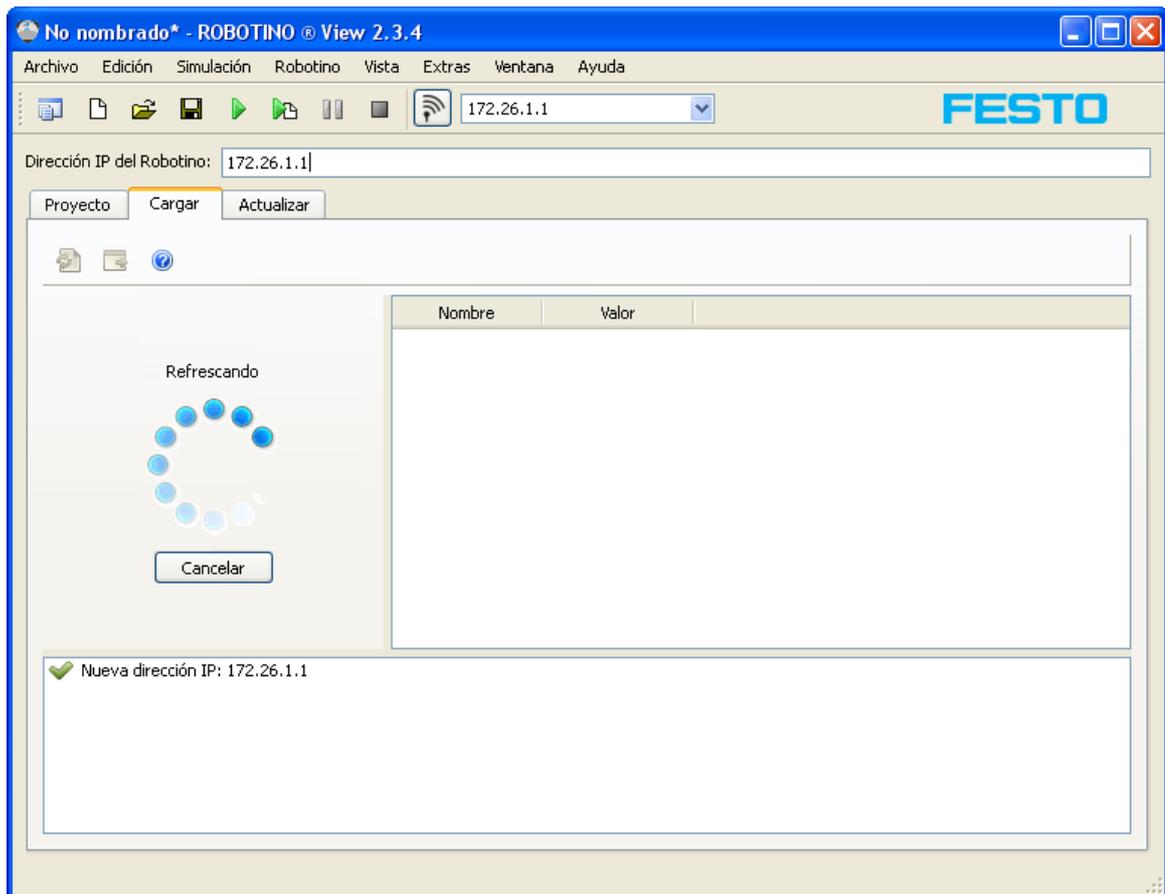
La primera vez que se llama al diálogo de carga, la primera dirección IP del dispositivo Robotino será introducida en el campo de entrada "Dirección IP del Robotino". Si no hay un dispositivo Robotino en el proyecto actual, el campo de entrada permanece vacío.

Cuando se abre el diálogo, la vista del directorio en Robotino será actualizada. La ejecución de la acción se indica con una animación. La actualización de la vista también puede invocarse con el botón . Pueden hallarse más detalles sobre la exploración de la estructura del directorio de Robotino en la sección [Explorar Robotino](#)^[21].

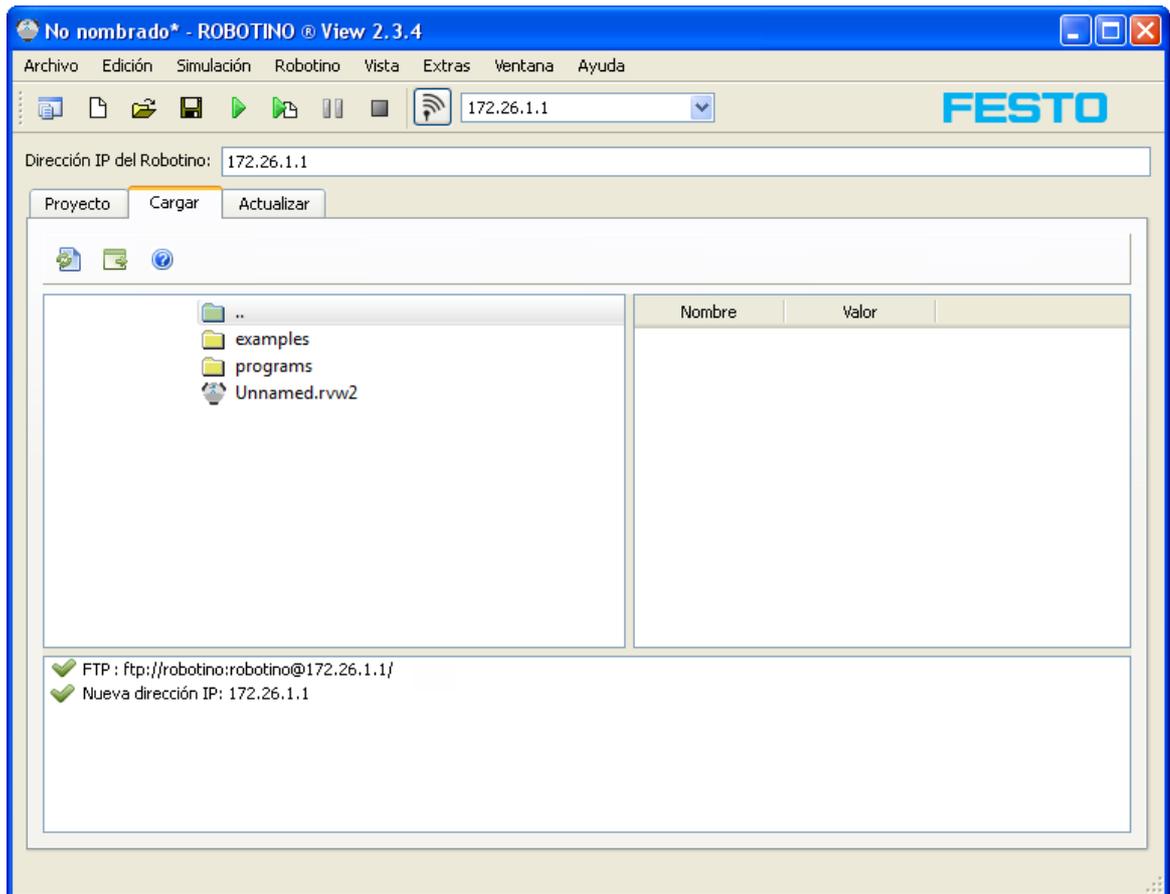
El botón  se utiliza para cargar el proyecto actual en el directorio visualizado actualmente. Más detalles sobre la carga y ejecución de proyectos pueden hallarse en la sección [Cargar y ejecutar](#)^[22].

3.12.1 Explorar Robotino

A partir de la versión 2.0 de Robotino flash card, un servidor Telnet y FTP están instalados en el sistema Linux Ubuntu. FTP se utiliza para visualizar los archivos del Robotino y para cargar proyectos.



Tras el primer inicio de sesión, se muestra el directorio /home/robotino. En este caso, hay los subdirectorios "examples" y "programs" y el proyecto Robotino View "Unnamed" en el directorio actual. Haciendo clic en uno de los directorios, la vista es actualizada y se muestra el contenido del directorio seleccionado. Haciendo clic en un proyecto Robotino View, se inicia la ejecución de este proyecto en Robotino. Véase [Cargar y ejecutar](#) [22].

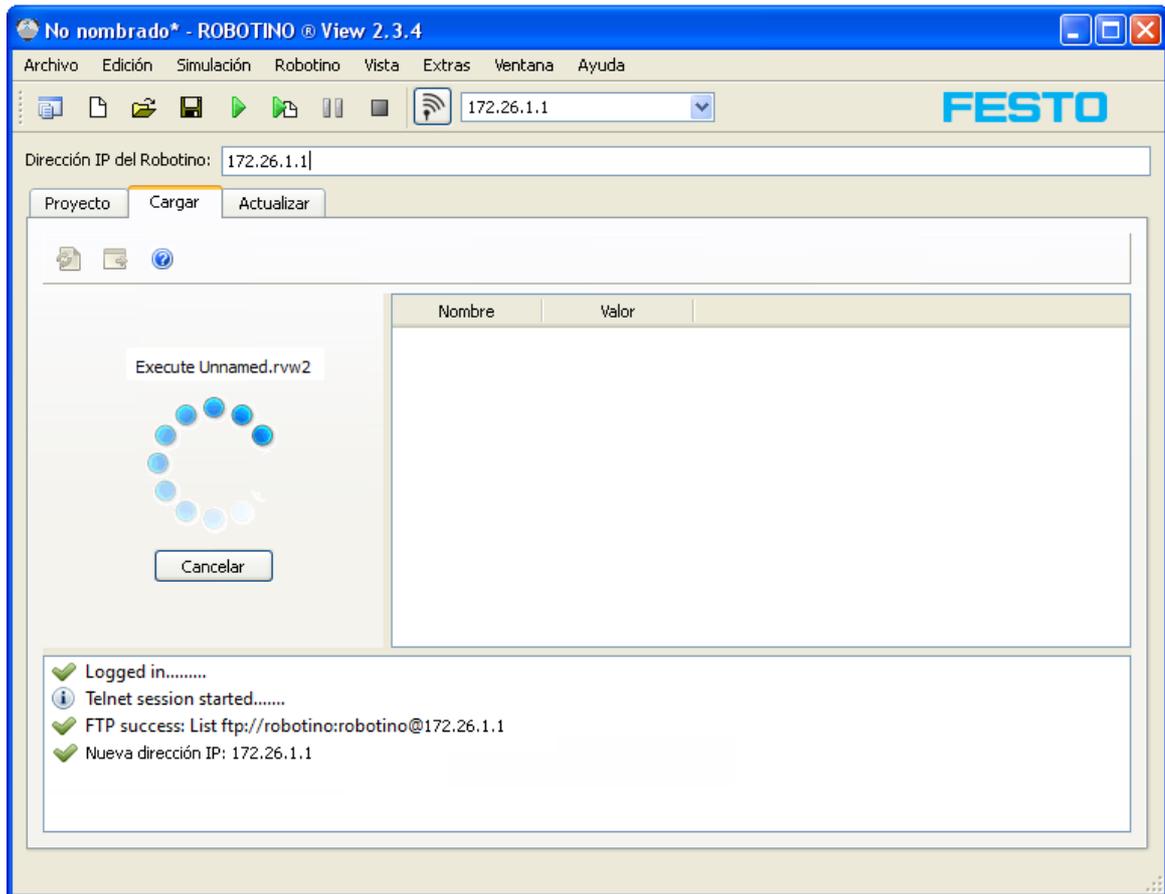


El cliente FTP integrado en Robotino View utiliza el login de usuario "robotino" con la contraseña "robotino". Con ello es posible iniciar sesión, por ejemplo con FileZilla, y crear subdirectorios o eliminar proyectos cargados.

3.12.2 Cargar y ejecutar

Antes de ejecutar un proyecto es recomendable verificar si la versión de Robotino View instalada en Robotino es la más reciente. La actualización del paquete Robotino se describe en la sección [Actualizar paquetes del Robotino](#)^[23].

Haciendo clic en un proyecto Robotino View  en la vista del directorio, se invoca la ejecución de este proyecto en Robotino con el Robotino View Interpreter. Antes de que empiece la ejecución del proyecto, debe cargarse el intérprete. Este proceso toma algunos segundos. La ventana de registro (log) muestra el estado actual.



Tras hacer clic en un proyecto Robotino View se establece una sesión Telnet con el nombre de usuario "robotino" y la contraseña "robotino". Inmediatamente después de aparecer el mensaje "Cargando proyecto", empieza su ejecución. El proceso puede cancelarse en cualquier momento.

En la ventana, junto al indicador de progreso, se muestran los valores de las variables globales del proyecto ejecutado en Robotino. La velocidad de actualización puede ser configurada en Extras ▶ Opciones... ▶ Cargar y ejecutar ▶ Intervalo depuración

3.13 Actualizar paquetes del Robotino

A partir de la versión 2.4.0 de Robotino View y la 2.0 de Robotino flash card, es posible actualizar los paquetes de Linux instalados en Robotino desde el propio Robotino View. Esta función es accesible a través de: Robotino ▶ Actualizar software.

La primera vez que se llama al diálogo de actualización, la primera dirección IP del dispositivo Robotino será introducida en el campo de entrada "Dirección IP del Robotino". Si no hay un dispositivo Robotino en el proyecto actual, el campo de entrada permanece vacío.

Utilización de Robotino® View

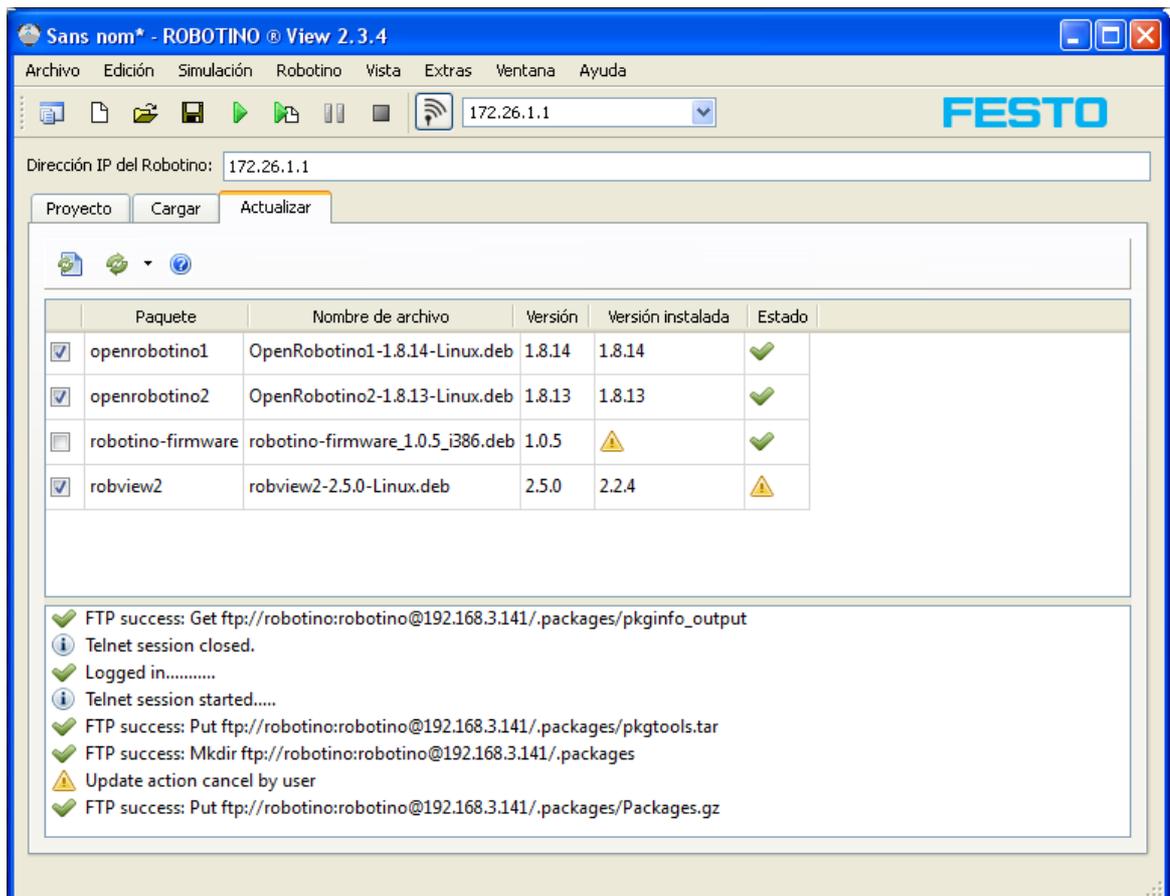
Cuando se abre el diálogo, la información del paquete será actualizada. Durante la actualización, toda la aplicación queda bloqueada. No obstante, la acción puede cancelarse fácilmente.

La actualización puede forzarse mediante el símbolo .

Tras una actualización correcta, se muestran las versiones de los paquetes locales y los paquetes instalados en Robotino. Los símbolos de estado tienen los siguientes significados:

 No hay información disponible o la versión instalada no está actualizada.

 El paquete instalado en el Robotino está actualizado



El paquete "robotino-firmware" es especial. La rutina de actualización verifica si hay una tarjeta EA09 IO en el Robotino. Si se encuentra una tarjeta EA09 IO, el número de la versión se toma directamente de la tarjeta IO. Si no hay instalada una tarjeta EA09, se muestra el símbolo  en lugar del número de la versión. Sin embargo, el estado del paquete es  puesto que el paquete "robotino-firmware" no necesita ser instalado.

En la primera columna de la vista de la versión, pueden añadirse o quitarse paquetes del proceso de actualización. De forma predeterminada, los paquetes "openrobotino1", "openrobotino2" y "robview2" están señalados para actualizar.

En la pantalla superior, el paquete "robview" instalado en Robotino no está actualizado. La versión local es 2.5.0. En Robotino, está instalada la antigua versión 2.2.4. La instalación del nuevo paquete se invoca con el símbolo . El diálogo de actualización muestra que la acción se está realizando. En la ventana de seguimiento puede verse el progreso. Cuando termina la instalación, la vista de la versión es actualizada.

3.13.1 Instalación del Robotino firmware

El paquete "robotino-firmware" es especial. La rutina de actualización verifica si hay una tarjeta EA09 IO en el Robotino. Si se encuentra una tarjeta EA09 IO, el número de la versión se toma directamente de la tarjeta IO. Si no hay instalada una tarjeta EA09, se muestra el símbolo  en lugar del número de la versión. Sin embargo, el estado del paquete es  puesto que el paquete "robotino-firmware" no necesita ser instalado.

Como sea que la actualización de firmware del Robotino por el paquete "robotino-firmware" es crítico, este paquete no será actualizado de forma predeterminada. Solamente si se conoce la razón exacta para una actualización, estos paquetes podrán añadirse al proceso de actualización. La instalación del firmware se describe en la sección Instalación de Robotino firmware.

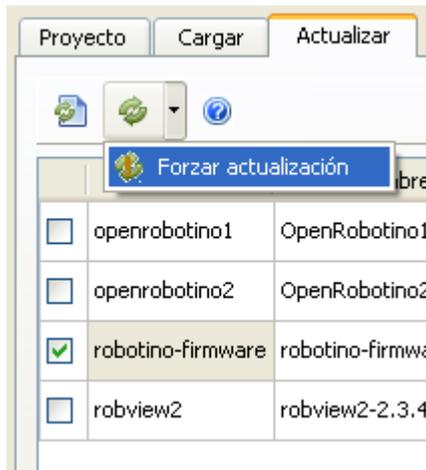
El firmware del microcontrolador (un NXP LPC 2378) en esta tarjeta IO, puede actualizarse desde la PC104 del Robotino. Este proceso es crítico. Un fallo en la actualización del firmware produce los siguientes efectos:

1. El Robotino no puede desconectarse presionando el botón On/Off.
2. Al presionar el botón On/Off el Robotino se enciende. Al soltar el botón, el Robotino se apaga inmediatamente.

referente a 1) Sacando el puente de mando, el Robotino puede apagarse

referente a 2) El botón On/Off debe mantenerse presionado hasta que se realice otra actualización del firmware con éxito.

Para actualizar solamente el firmware (o para repararlo), sólo debe seleccionarse el paquete "robotino-firmware". A continuación, la instalación puede forzarse por medio del botón  "Forzar actualización".



3.13.2 Interna

El proceso de actualización está basado en una combinación de Telnet, FTP y comandos Linux concernientes a apt.

Primero se copia el archivo pkgtools.tar del directorio install_folder\packages en /home/robotino/.packages. A través de Telnet, el archivo es desempquetado. El script pkginfo.sh proporciona información sobre los paquetes instalados..

Los paquetes a instalar son copiados a través de FPT de install_folder\packages a /home/robotino/.packages. Adicionalmente se copia el archivo Packages.gz. Contiene informaciones sobre el paquete.

Inicialmente, el script pkginstall.sh modifica /etc/apt/sources.list y entra el directorio /home/robotino/.packages como paquete fuente solamente. Luego se utiliza apt-get para instalar los paquetes.

pkgremove.sh fuerza la eliminación de los paquetes.

startOpenRobotino1.sh se invoca para reinicar Robotino deamons.

4 Ejemplos

4.1 Programas de control

En este capítulo se realiza un programa de control sencillo con ramales alternativos.

4.1.1 Tutorial 2

Este ejercicio muestra cómo se crea un programa de control con ramales alternativos. El programa completo se halla en el archivo: `examples/sfc/tutorial2.rvw2`.

El programa de control completo tiene el aspecto que muestra la figura 1.

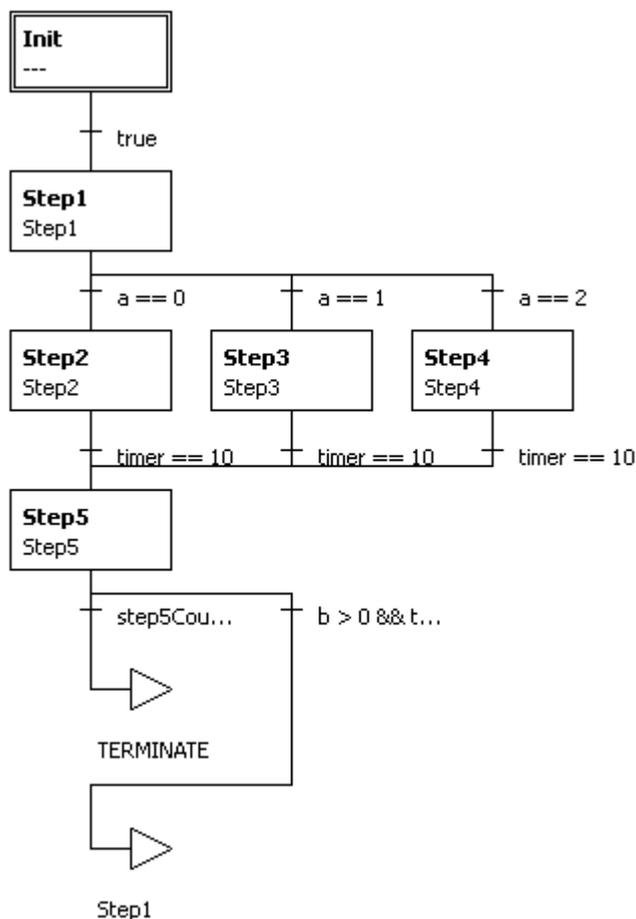


Fig. 1: el programa de control completo

En el Step1, el valor de `a` cambia. Así, en cada ciclo del programa se ejecutará uno de los pasos Step2, Step3 y Step4. Step5 compara el resultado producido por los pasos anteriores. Tras la 6ª ejecución de Step5, el programa se detiene. De lo contrario continúa con Step1.

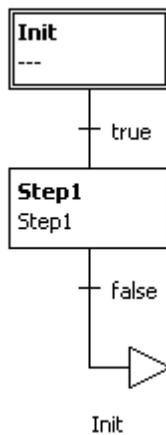
Crear un nuevo proyecto

Crear un nuevo proyecto seleccionando

- Archivo Nuevo
- presionando Ctrl + N
- seleccionando el símbolo para crear un nuevo proyecto en la barra de herramientas 

Ejemplos

El programa principal contiene los pasos Init y Step1.



Crear las variables globales

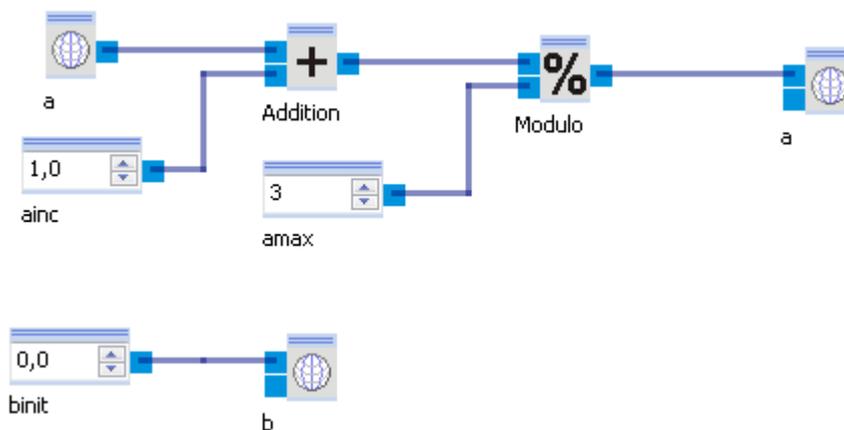
Primero crear las siguientes [Variables globales](#)¹⁵:

- timer
- a
- b
- step2count
- step3count
- step4count
- step5count

Asignar el valor inicial -1 a "a". Mantener el valor inicial de 0 para todas las demás variables.

Programa Step1

En este subprograma, la variable global "a" es incrementada en 1. Para asegurar que el valor de "a" se halla siempre entre 0 y 2, "a" será calculada con Módulo 3 y reescrita en "a". El valor de "b" se dejará en 0.

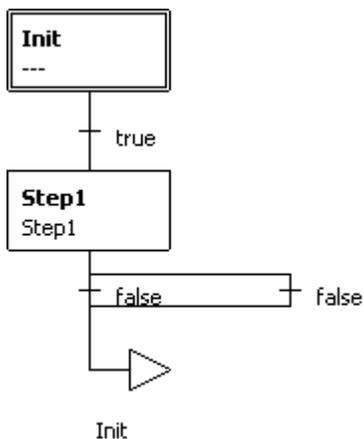


Crear los pasos Step2, Step3 y Step4

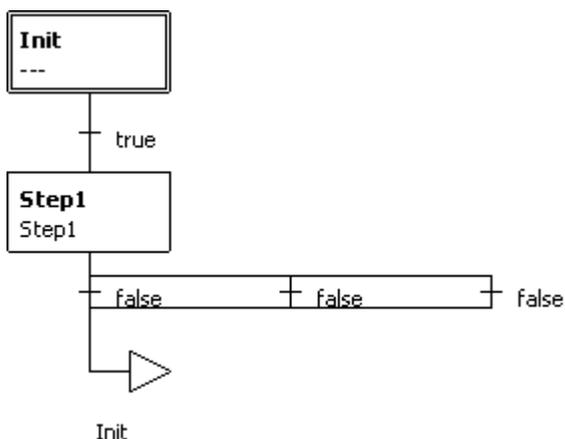
Ahora los pasos se crearán uno junto a otro en ramales alternativos. Para ello, seleccione la condición de transición debajo de Step1 con un doble clic.

Puede ver que la condición de transición seleccionada muestra una línea de trazos a su alrededor.

Ahora haga clic en el símbolo para añadir un ramal alternativo a la derecha. .

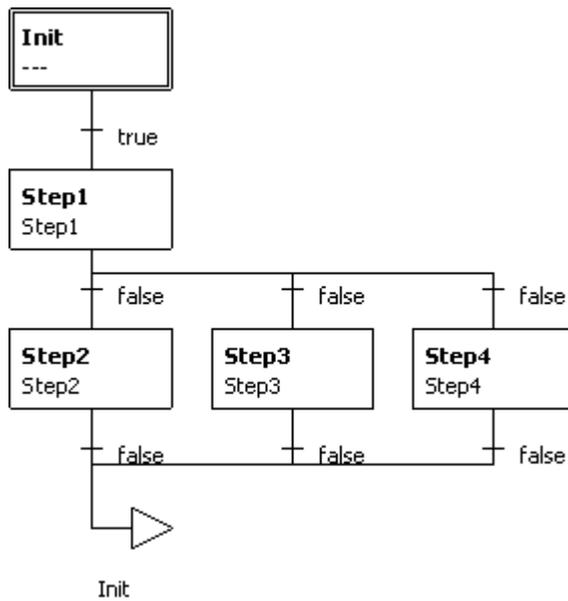


Siga ampliando el ramal que acaba de crear, seleccionando la condición de transición de la derecha y seleccionando del símbolo  "Ramal alternativo derecho" de nuevo.

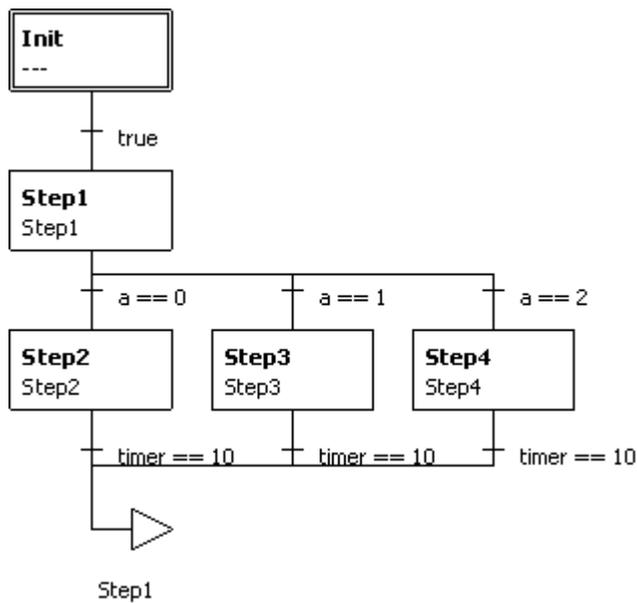


Ahora cree tres Pasos (Steps) en estos tres ramales alternativos y nómbralos como Step2, Step3 y Step4. Para ello, seleccione la condición de introducción de un ramal y haga clic en el símbolo "Insertar paso después" . Luego asigne un subprograma del mismo nombre a cada paso. Para ello, haga doble clic en el paso e introduzca el nombre del subprograma en la siguiente caja de diálogo. Alternativamente, puede crear un nuevo subprograma con el botón de la barra de herramientas "Crear nuevo subprograma".  y asignarlo al Paso (Step).

Ejemplos



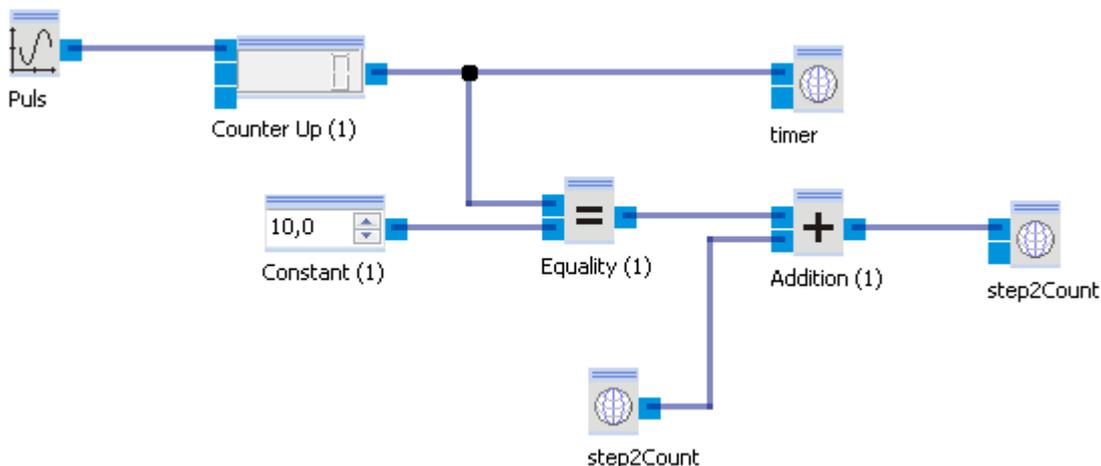
Las condiciones de entrada y salida de los tres ramales alternativos son falsas (false) por el momento. Cambie las condiciones de entrada a: $a == 0$, $a == 1$ y $a == 2$. Utilice $timer == 10$ como condición de salida para todos los ramales. Finalmente, cambie el destino del salto final de Init a Step1.



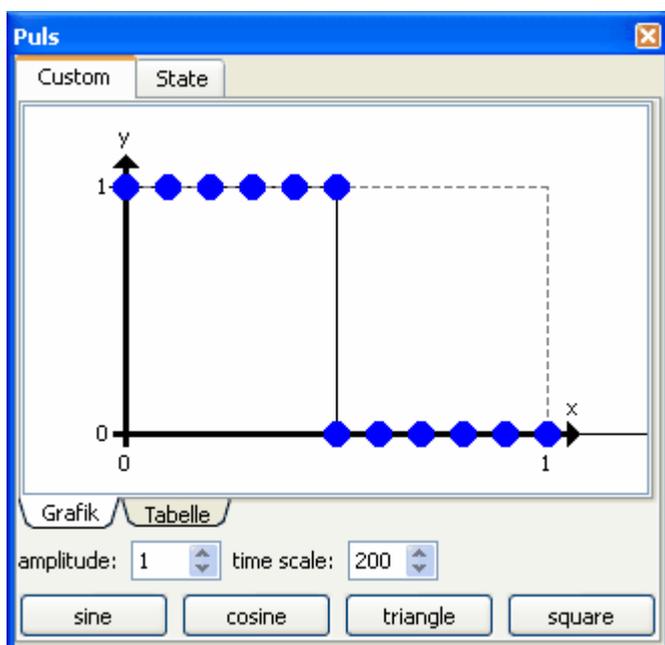
Si ahora inicia el programa principal, el programa se detendrá en Step2, puesto que "a" es 0 durante el primer ciclo y la variable global "timer" no se ha alterado.

Programas Step2, Step3 y Step4

Los subprogramas asignados a los pasos Step2 a Step4 están vacíos por el momento. El subprograma Step2 se muestra a continuación.



Cada 200 ms, el Generador de forma de onda arbitraria crea un pulso de 100 ms de ancho y una altura de 1. Los ajustes para el Generador de forma de onda arbitraria se muestran debajo.



Es decir, cada 200 ms hay un flanco ascendente de 0 a 1. Con cada flanco ascendente, el contador incrementa su valor en 1. Tras 2 s, el valor será 10. Cuando el valor del contador sea 10, el resultado de la comparación de la constante y el valor del contador se añadirá al valor actual de step2Count. Mientras el resultado de la comparación sea falso, se añade 0. Cuando el resultado de la comparación sea true, se añade 1. Al final de cada paso de cálculo del subprograma, se evalúa la condición de transición bajo el paso del programa principal. Cuando la variable global "timer" tiene el valor 10, se abandona el subprograma.

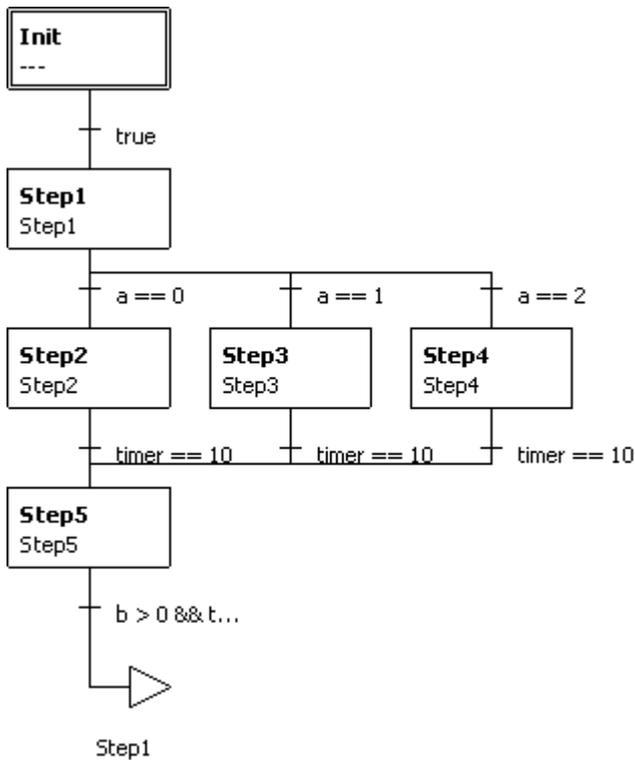
Los subprogramas Step3 y Step4 se construyen de forma equivalente. Seleccione todo en Step2 (Ctrl+A), Copie y Pegue en Step3 y Step4. La única diferencia consiste en el hecho que se leen y escriben respectivamente step3count y step4count.

Ejemplos

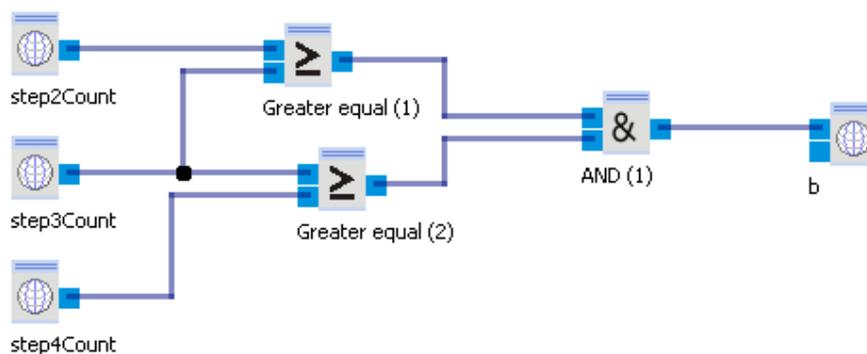
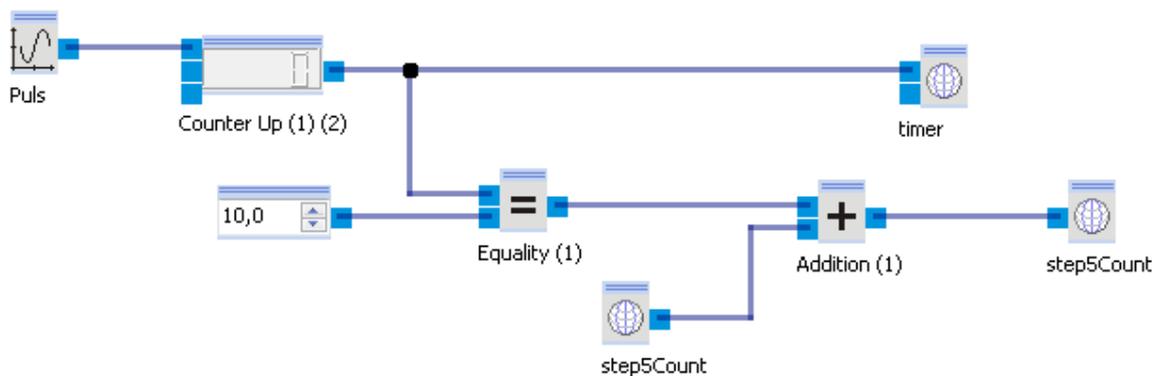
Una vez iniciado el programa principal, Step2, Step3 y Step4 se ejecutarán cíclicamente durante 2 s cada uno.

Crear el programa Step5

Para añadir un nuevo paso tras el ramal alternativo, seleccione el salto final y haga clic en el símbolo para añadir un paso antes. . Ahora cree un subprograma denominado Step5 y asígnelo al Step5 acabado de crear. Cambie la condición de transición debajo de Step5 a: $b > 0 \ \&\& \ \text{timer} == 10$.



El subprograma Step5 es similar a Step2 ... Step4. Copie Step2 a Step5 y cambie step2count por step5count. Tras establecer las variables globales "timer" y "step5count" también se realiza una verificación de si se cumple la condición $\text{step2count} \geq \text{step3count} \geq \text{step4count}$. Si es éste el caso, la variable global "b" se pone en 1. De lo contrario "b" es 0. La condición siempre debe ser verdadera (true) durante la correcta ejecución del programa, puesto que Step2, Step3 y Step4 se ejecutan uno tras otro ya que Step1 incrementa "a" en 1 a cada ciclo.



Si ahora se inicia el programa principal, Step5 permanece activo durante 2 s si "b" es mayor de 0.

Crear la terminación del programa y saltar al Step1.

Ahora el programa debería finalizar cuando el valor de "step5count" haya alcanzado 6. Para ello, inserte un ramal alternativo debajo de Step5. Seleccione la condición de transición debajo de Step5 y haga clic en el símbolo para insertar un ramal alternativo a la izquierda. . Seleccione la nueva condición de transición del ramal (por el momento es false) y haga clic en el símbolo para crear un nuevo salto. . Cambie la condición de transición a `step5count == 6` y seleccione TERMINATE como nuevo destino de salto.

El programa principal tiene ahora el aspecto que se mostró al principio.

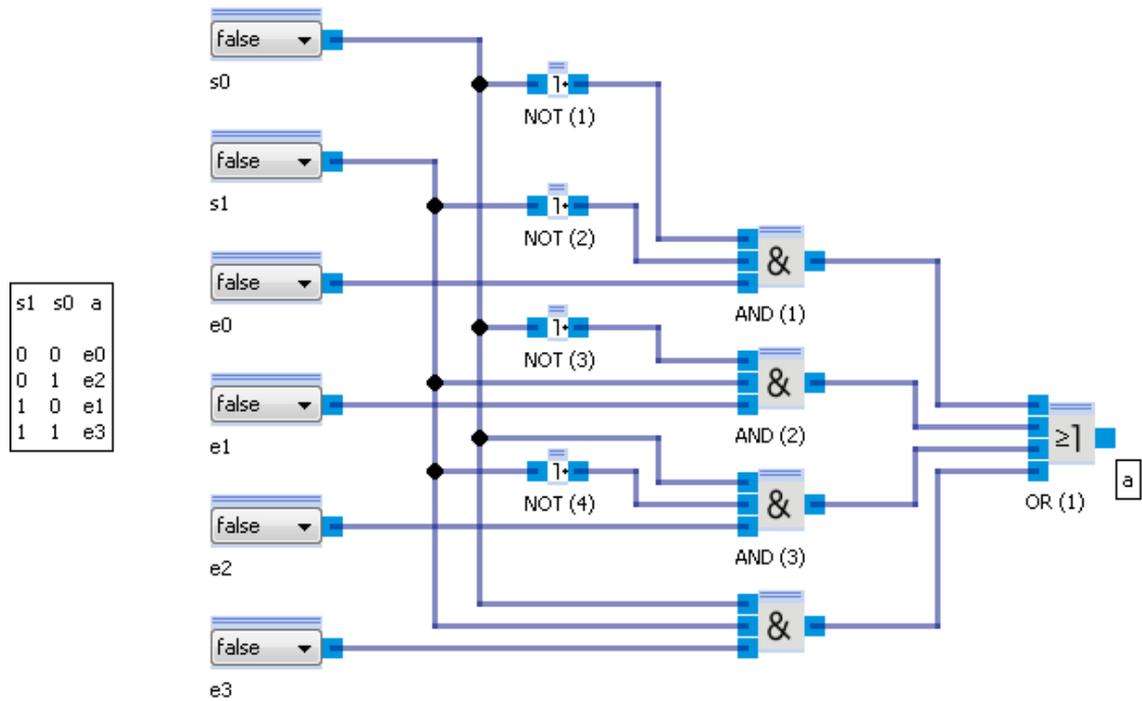
Por cierto, el ramal alternativo que contiene el salto a TERMINATE debe hallarse a la izquierda del ramal con la condición `b>0 && timer == 10`, puesto que las condiciones iniciales de los ramales alternativos se evalúan de izquierda a derecha. En los primeros 6 ciclos, no se cumple la condición `step5count == 6`. Por ello se evalúa la condición del segundo ramal.

Ahora la ejecución del programa principal dura 24 s.

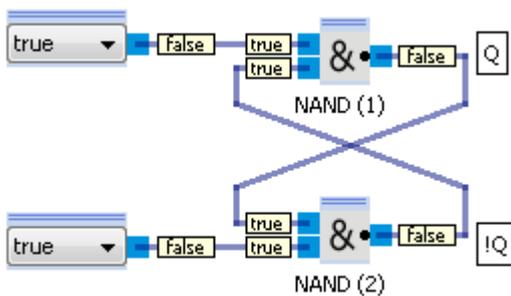
4.2 Lógica

En este apartado, se realizan los conocidos circuitos eléctricos por medio de funciones lógicas.

4.2.1 Multiplexor



4.2.2 FlipFlop



5 Librería de bloques de función

Los programas de control creados con Robotino® View, consisten en bloques de función enlazados.

Estos se hallan en la [Librería de bloques de función](#) y pueden insertarse en un subprograma arrastrándolos y soltándolos (Drag&Drop).

Los bloques de función están asignados a diferentes categorías. Haciendo doble clic en el nombre de una categoría con el botón izquierdo del ratón, se despliega la capeta de la categoría. Hay disponibles las siguientes categorías:

Nombre	Descripción
Lógica	Los componentes son reconocidos a partir de módulos lógicos electrónicos
Matemáticas	Operaciones matemáticas simples
Cálculo vectorial	Análisis utilizando vectores de dos dimensiones
Visualizador	Osciloscopio gráfico y visualizador de vectores de dos dimensiones
Procesamiento de imágenes	Funciones básicas de procesamiento de imágenes
Generador	Generación de señales
Filtro	Suavizado de señales
Navegación	Accionamiento de Robots móviles
Dispositivos de entrada	Bloques de función para la interacción del usuario con el programa de control
Intercambio de datos	Intercambio de datos con programas externos
Mis bloques de función	Tutoriales para el desarrollo de bloques de función propios.

5.1 Lógica

La categoría de Lógica contiene componentes que son reconocidos como módulos o funciones lógicas electrónicas.

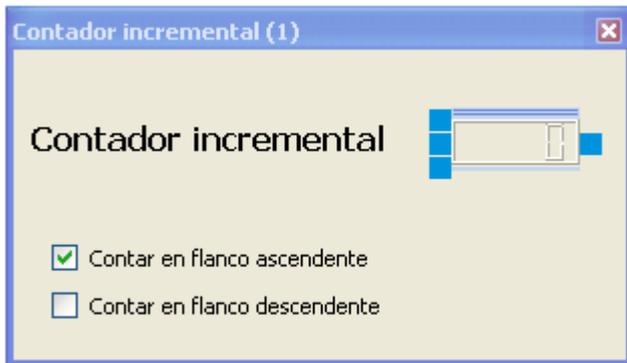
5.1.1 Contador incremental



El contador cuenta el número de eventos en su conector de Entrada

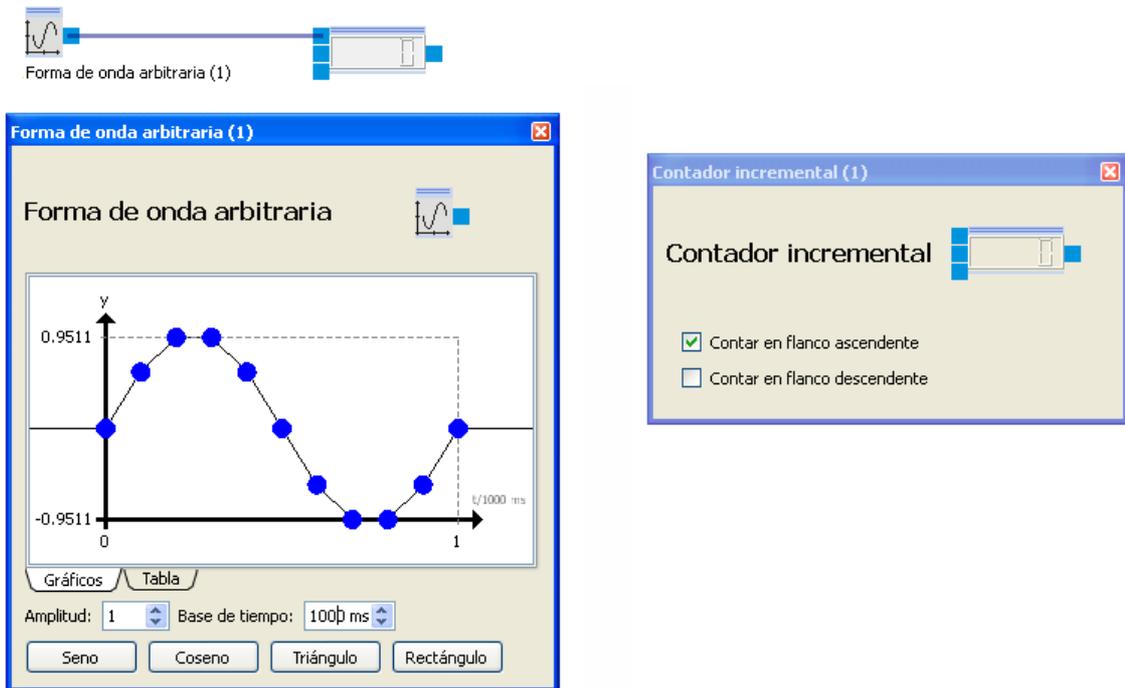
Entradas	Tipo	Predeterminado	Descripción
Entrada	bool	false	Entrada de conteo. El valor de conteo es incrementado si la entrada cambia de falso (false) a verdadero (true).
Valor inicial	int32	0	El conteo empieza con el valor aplicado aquí al iniciarse el subprograma o después que la entrada Reset haya sido true.
Reset	bool	false	El contador es restaurado a su valor inicial si esta entrada es true.
Salidas			
Salida	int32		Valor de conteo

5.1.1.1 Diálogo



Contar en flanco ascendente	Incrementa el contador en 1 si la entrada en el momento t es falsa (false) y en el momento t+1 es verdadera (true).
Contar en flanco descendente	Incrementa el contador en 1 si la entrada en el momento t es verdadera (true) y en el momento t+1 es falsa (false).

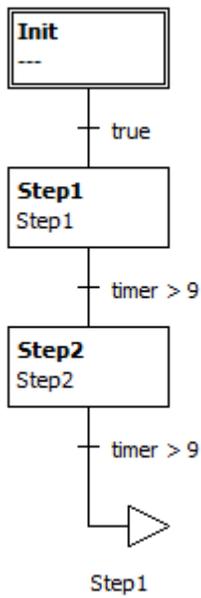
5.1.1.2 Ejemplo



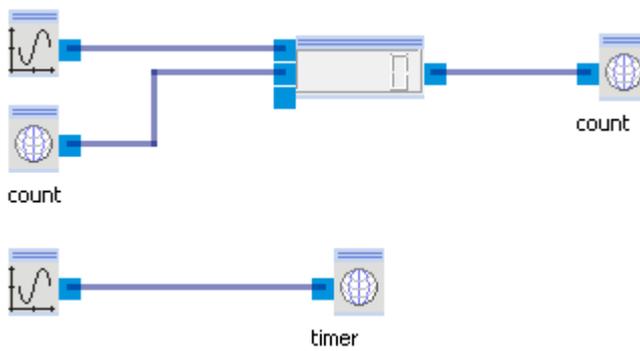
El "[Generador de forma de onda arbitraria](#)^[95]" genera una forma de onda senoidal con amplitud 2 y frecuencia de 1Hz. La salida del generador es del tipo real (float). Los valores menores o iguales a 0 son convertidos a falso (false). Los valores mayores de 0 son convertidos a verdadero (true) (véase [conversión de tipos](#)^[20]). El contador cuenta sobre el flanco ascendente, es decir, cuando la entrada cambia de falso (false) a verdadero (true). Esto sucede exactamente una vez por segundo al principio de la onda senoidal. Por ello el valor del contador representa el tiempo en segundos desde el inicio del subprograma.

El siguiente ejemplo muestra cómo utilizar la entrada de valor inicial para contar en los límites de un subprograma. El programa principal ejecuta Step1 y Step2 secuencialmente. Una vez finalizado Step2, volvemos a empezar con Step1.

Programa principal

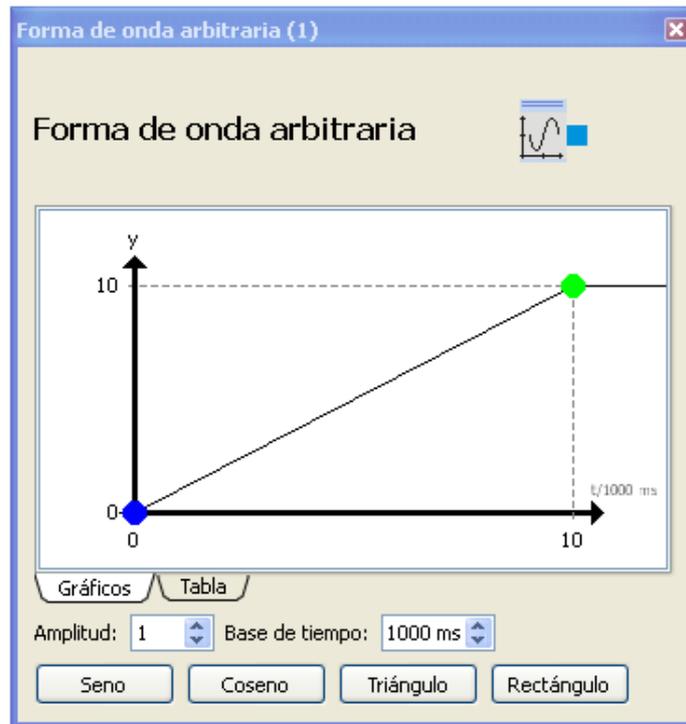
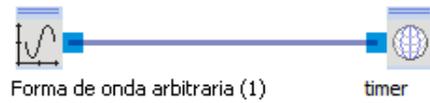


Step1



El contador escribe su resultado en la variable global "count". Tras el reinicio de Step1 la variable global "count" se utiliza como valor inicial para el Contador. Step1 está activo hasta que el segundo "[Generador de forma de onda arbitraria](#)" genera un valor mayor de 9. Esto sucede transcurridos 10s.

Step2



Step2 está también 10s activo.

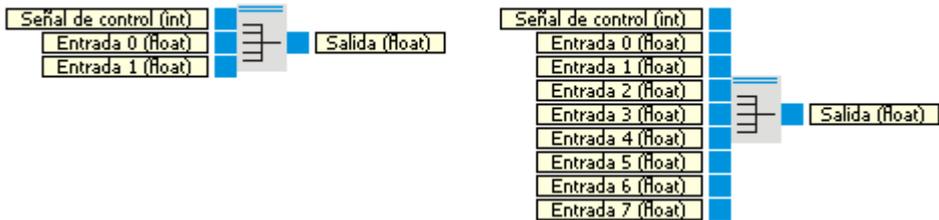
5.1.2 Contador decremental

El contador decremental es similar al [Contador incremental](#)^[35]. La única diferencia es que el valor es decrementado en 1 si se produce un evento de conteo.

5.1.2.1 Diálogo

Véase diálogo de [Contador incremental](#)^[36].

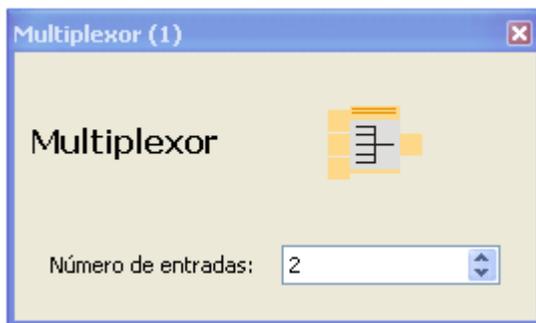
5.1.3 Multiplexor



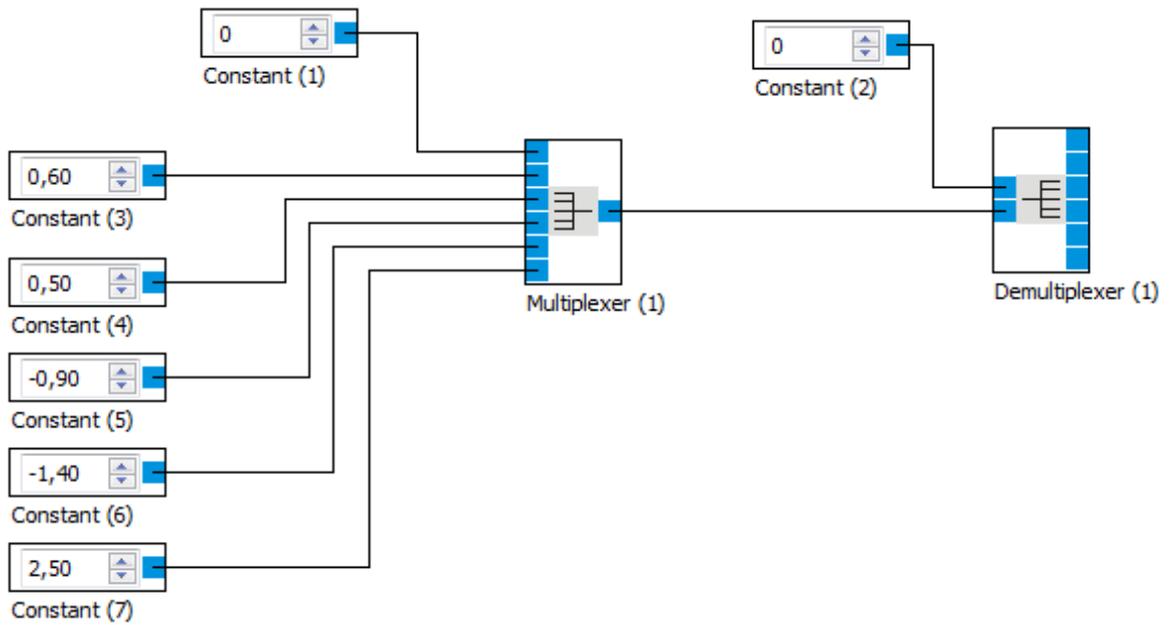
El Multiplexor conecta una salida con una entrada seleccionable.

Entradas	Tipo	Predeterminado	Descripción
Señal de control	int	0	Determina la entrada que se hallará conectada con la salida. Si la señal de control es menor de 0 o mayor o igual que el número de entradas, la salida se pone a 0.
Entrada0	float	0	El valor de la entrada 0 es copiado en la salida si la señal de control es 0
...			
Entrada9	float	0	El valor de la entrada 9 es copiado en la salida si la señal de control es 9
Salidas			
Salida	float		El valor de una entrada, o 0 si la señal de control es inferior a 0 o mayor o igual al número de entradas.

5.1.3.1 Diálogo

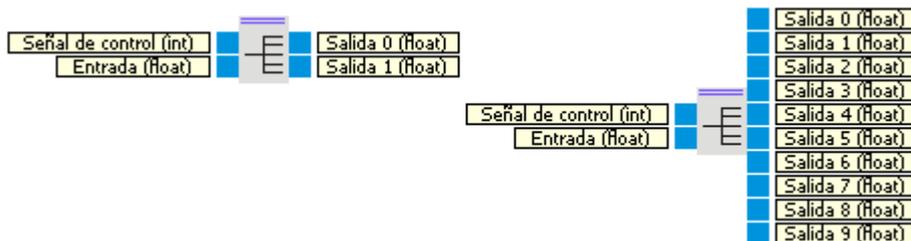


5.1.3.2 Ejemplo



véase también Ejemplos►Lógica►[Multiplexor](#)³⁴

5.1.4 Desmultiplexor



El desmultiplexor conecta una entrada con una salida seleccionable.

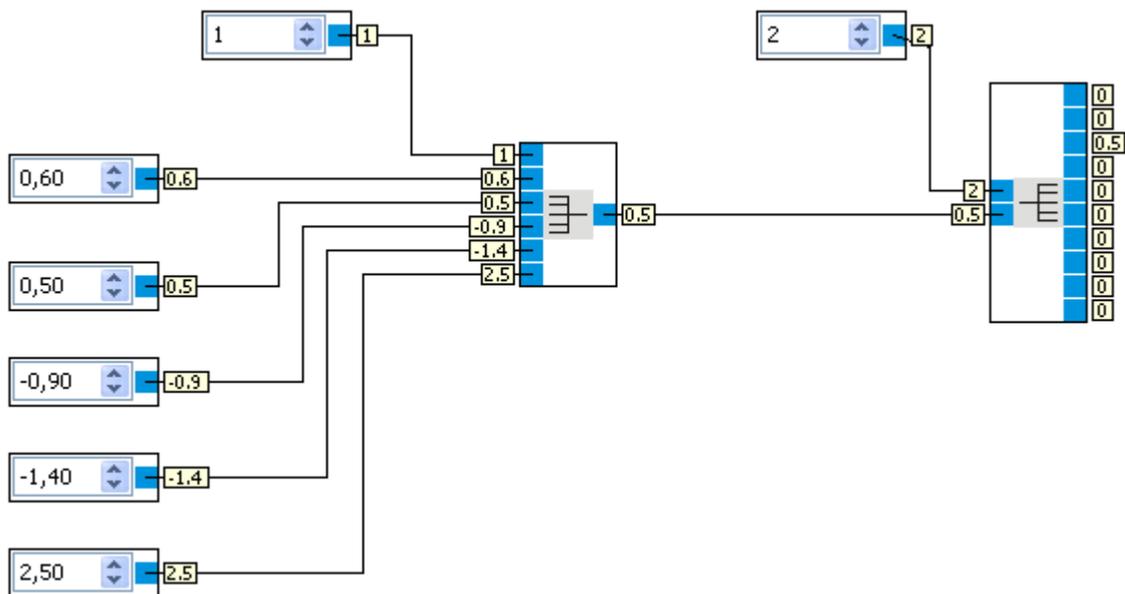
Entradas	Tipo	Predeterminado	Descripción
Señal de control	int	0	Determina la salida que se hallará conectada con la entrada. Si la señal de control es menor de 0 o mayor o igual que el número de salidas, todas las salidas se ponen en 0.
Entrada	float	0	Valor que se aplicará a la salida determinada por la señal de control.
Salidas			
Salida 0	float		Valor de la entrada si la señal de control es 0, de lo contrario 0.

...		
Salida 9	float	Valor de la entrada si la señal de control es 9, de lo contrario 0.

5.1.4.1 Diálogo



5.1.4.2 Ejemplo



5.1.5 AND

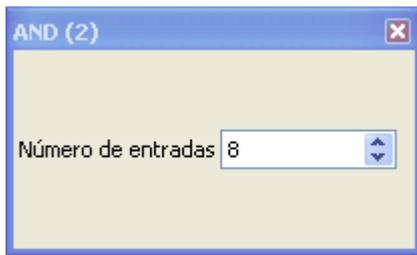


La salida de un bloque AND es verdadera (true) sólo si todas las entradas son verdaderas (true). Véase [conversión de tipos](#)^[20] cómo los números se convierten a valores booleanos (bool).

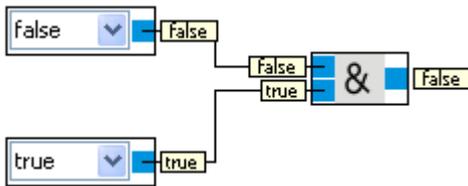
Entradas	Tipo	Predeterminado	Descripción
Entrada 1	bool	true	
...			
Entrada 8	bool	true	
Salidas			
Q	bool		ver tabla inferior

Entradas								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	0
							1	0
						1		0
						1	1	0
					1			0
					1		1	0
					1	1		0
					1	1	1	0
				1				0
				1			1	0
				1		1		0
				1		1	1	0
				1	1			0
				1	1		1	0
				1	1	1		0
				1	1	1	1	0
			1					0
1	1	1	1	1	1	1	1	1

5.1.5.1 Diálogo



5.1.5.2 Ejemplo



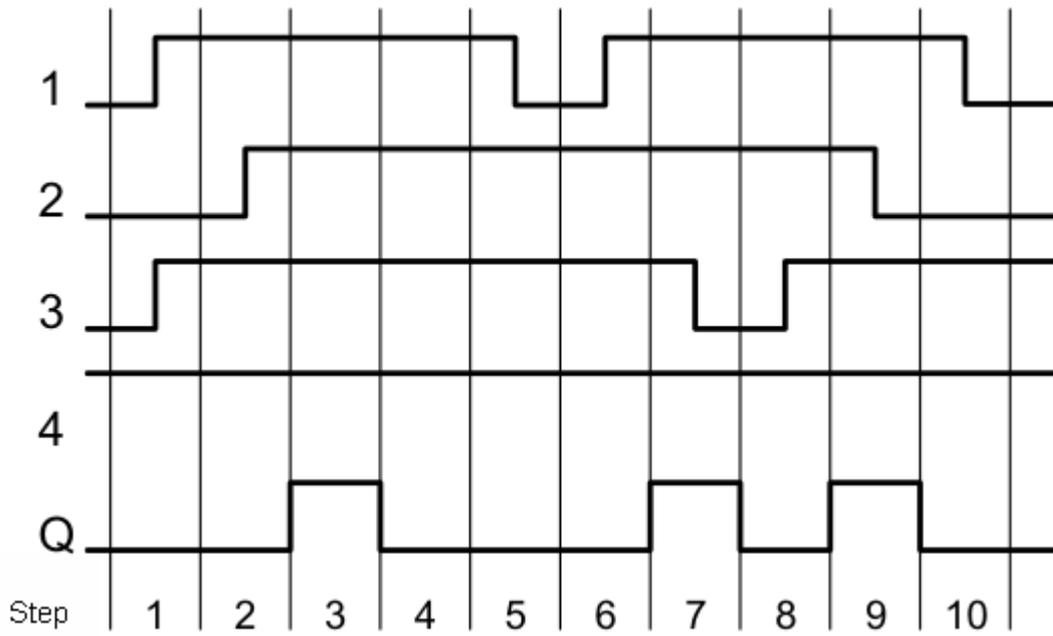
5.1.6 AND FL



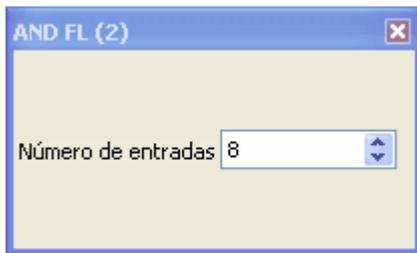
La salida Q del bloque AND FL (control por flancos) sólo se halla en 1 si todas las entradas son verdaderas (true) y si por lo menos una entrada era falsa (false) durante el ciclo anterior. Genera un pulso de un ciclo de duración. Véase [conversión de tipos](#)^[20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada 1	bool	true	
...			
Entrada 8	bool	true	
Salidas			
Q	bool		véase diagrama de tiempos

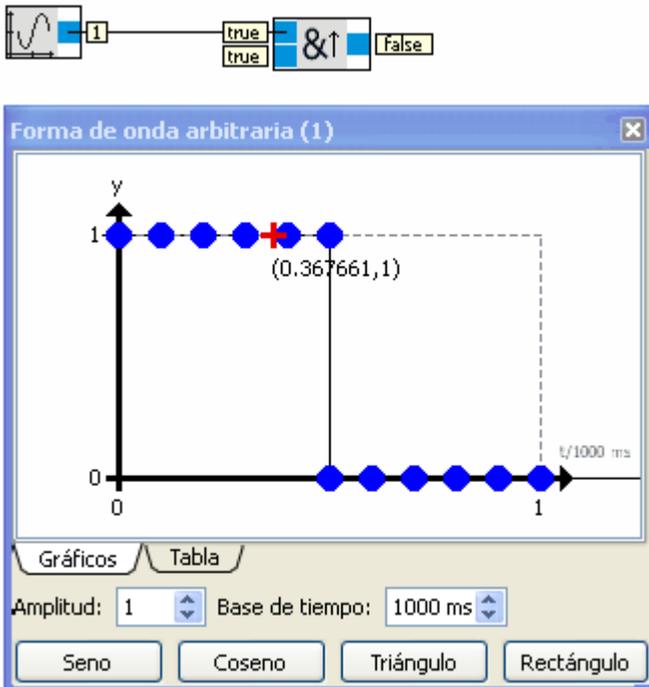
Diagrama de tiempos para el bloque AND FL de control por flancos y cuatro entradas.



5.1.6.1 Diálogo



5.1.6.2 Ejemplo



Cuando la salida del generador cambia de 0 a 1, la salida del bloque AND FL es verdadera (true) durante un ciclo.

5.1.7 NAND



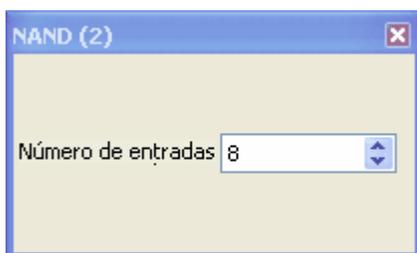
La salida de un bloque NAND es falsa (false) sólo si todas las entradas son verdaderas (true). Véase [conversión de tipos](#) [20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada1	bool	true	
...			
Entrada8	bool	true	

Salidas			
Q	bool		ver tabla inferior

Entradas								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	1
							1	1
						1		1
						1	1	1
					1			1
					1		1	1
					1	1		1
					1	1	1	1
				1				1
				1			1	1
				1		1		1
				1		1	1	1
				1	1			1
				1	1		1	1
				1	1	1		1
				1	1	1	1	1
			1					1
1	1	1	1	1	1	1	1	0

5.1.7.1 Diálogo



5.1.7.2 Ejemplo

véase Ejemplo ▶ Lógica ▶ [FlipFlop](#) [34]

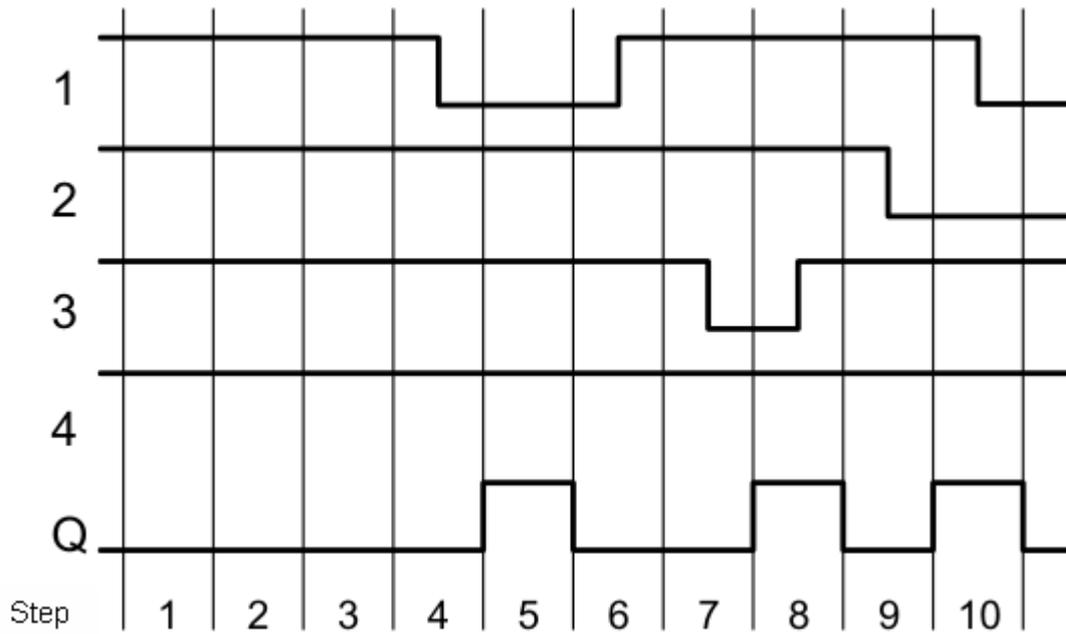
5.1.8 NAND_FL



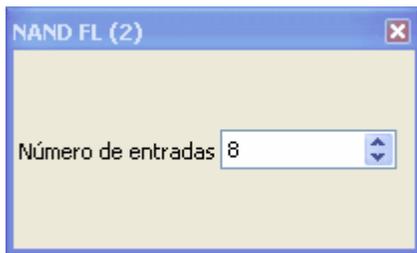
La salida Q del bloque NAND FL (con control por flancos) sólo se halla en 1 si por lo menos una entrada es falsa y si todas las entradas fueron verdaderas (true) durante el ciclo anterior. Genera un pulso de un ciclo de duración. Véase [conversión de tipos](#) [20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada1	bool	true	
...			
Entrada8	bool	true	
Salidas			
Q	bool		véase diagrama de tiempos

Diagrama de tiempos para el bloque NAND FL con control de flancos y cuatro entradas.



5.1.8.1 Diálogo



5.1.9 OR



La salida de la función OR sólo es verdadera (true) si por lo menos una entrada es verdadera (true). Véase [conversión de tipos](#) ^[20] cómo los números se convierten a valores booleanos (bool).

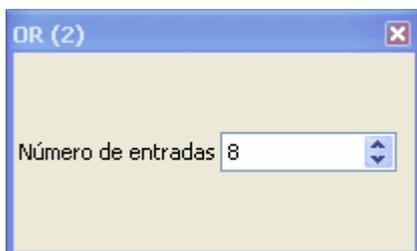
Entradas	Tipo	Predeterminado	Descripción
Entrada1	bool	false	

Librería de bloques de función

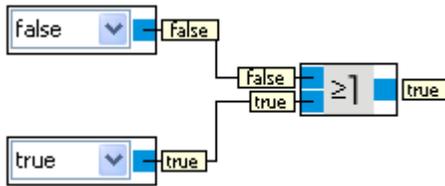
...			
Entrada8	bool	false	
Salidas			
Q	bool		ver tabla inferior

Entradas								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	0
							1	1
						1		1
						1	1	1
					1			1
					1		1	1
					1	1		1
					1	1	1	1
				1				1
				1			1	1
				1		1		1
				1		1	1	1
				1	1			1
				1	1		1	1
				1	1	1		1
				1	1	1	1	1
			1					1
1	1	1	1	1	1	1	1	1

5.1.9.1 Diálogo



5.1.9.2 Ejemplo



5.1.10 XOR

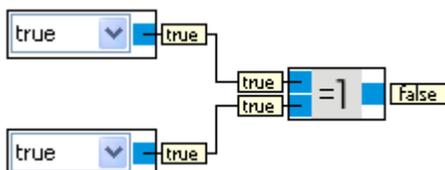


La salida de una función XOR es verdadera (true) si las entradas tienen valores diferentes. Véase [conversión de tipos](#)^[20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada1	bool	false	
Entrada2	bool	false	
Salidas			
Q	bool		ver tabla inferior

Entradas		
1	2	Q
0	0	0
0	1	1
1	0	1
1	1	0

5.1.10.1 Ejemplo



5.1.11 NOT



La salida de una función NOT es verdadera (true) si la entrada es falsa (false). Véase [conversión de tipos](#) [20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada	bool	false	
Salidas			
Q	bool		ver tabla inferior

Entradas	Q
1	0
0	1
1	0

5.1.11.1 Ejemplo



El ejemplo muestra el uso del bloque de función NOT. Los valores de entrada y salida no se muestran junto a sus conectores de entrada o salida. Esto tiene la ventaja de que la función NOT ocupa muy poco espacio y la visualización de su estado no se solapa con los datos mostrados por los bloques de función adyacentes.

5.1.12 NOR

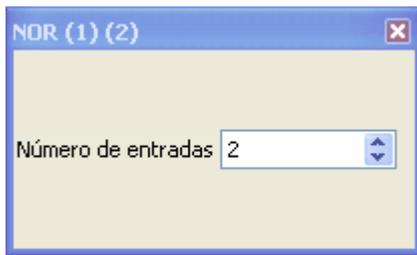


La salida Q de una función NOR es verdadera (true) si todas las entradas son falsas (false). Véase [conversión de tipos](#) [20] cómo los números se convierten a valores booleanos (bool).

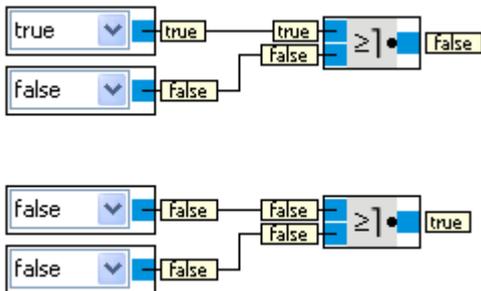
Entradas	Tipo	Predeterminado	Descripción
Entrada1	bool	false	
...			
Entrada8	bool	false	
Salidas			
Q	bool		ver tabla inferior

Entradas								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	1
							1	0
						1		0
						1	1	0
					1			0
					1		1	0
					1	1		0
					1	1	1	0
				1				0
				1			1	0
				1		1		0
				1		1	1	0
				1	1			0
				1	1		1	0
				1	1	1		0
				1	1	1	1	0
			1					0
1	1	1	1	1	1	1	1	0

5.1.12.1 Diálogo



5.1.12.2 Ejemplo



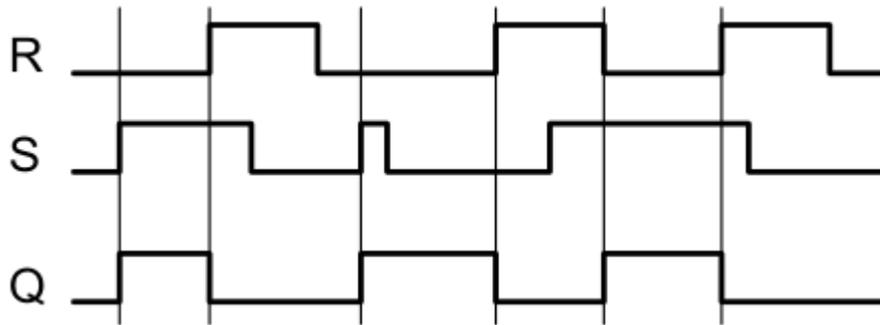
5.1.13 Relé de enclavamiento



La salida Q es activada por la entrada S. La entrada R desactiva la salida Q. Véase [conversión de tipos](#) ^[20] cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
S	bool	false	Si S es verdadera (true), Q se pone en verdadera (true).
R	bool	false	Si R es verdadera (true) Q se pone en falsa (false). R es prioritaria sobre S.
Par	bool	false	Remanencia: false: Sin remanencia true: El estado actual del relé es mantenido en una memoria remanente (independientemente del estado de S o R).
Salidas			
Q	bool		Q es puesta en verdadera (true) por S y se mantiene en true (aunque S se ponga en false) hasta que R se pone en true.

Diagrama de tiempos



5.1.14 Elemento de muestreo y retención



Si Muestra se halla en falso (false), la Salida mantiene su valor actual. Véase [conversión de tipos](#) cómo los números se convierten a valores booleanos (bool).

Entradas	Tipo	Predeterminado	Descripción
Entrada	float	0	Señal de entrada
Muestra	bool	false	Si es verdadera (true) la Salida asume el valor de la Entrada. Si es falsa (false), el valor actual de la salida se "congela".
Salidas			
Salida	float	0	Valor de la Entrada cuando Muestra pasa de false a true.

5.2 Matemáticas

Esta categoría contiene operaciones matemáticas simples.

5.2.1 Operaciones aritméticas

5.2.1.1 Módulo



En matemáticas, la aritmética modular (en ocasiones denominada "aritmética del reloj") es un sistema aritmético para números enteros, en la que los números "dan la vuelta" tras alcanzar cierto valor llamado "Módulo". La aritmética modular fue introducida en 1801 por Carl Friedrich Gauss en su libro *Disquisitiones Arithmeticae*. (Fuente: http://es.wikipedia.org/wiki/Aritmetica_modular)

Entradas	Tipo	Predeterminado	Descripción
Dividendo	int	0	
Divisor	int	1	
Salidas			
Resto	int		Dividendo mod Divisor

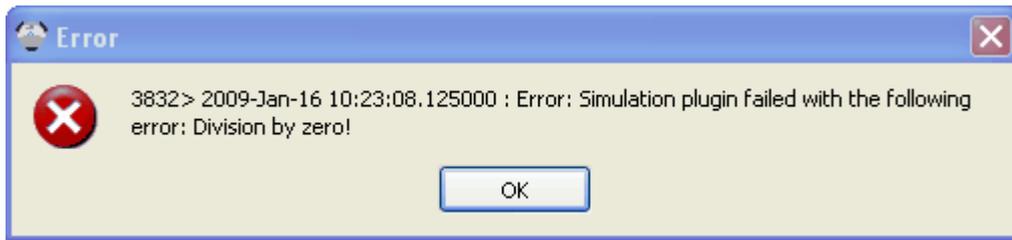
5.2.1.2 División



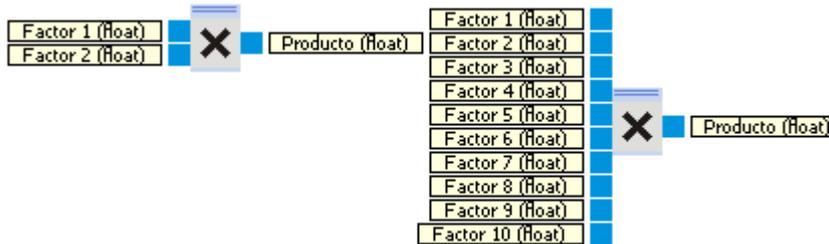
Calcula el cociente a partir del dividendo y del divisor. Véase http://es.wikipedia.org/wiki/División_matemática.

Entradas	Tipo	Predeterminado	Descripción
Dividendo	float	0	
Divisor	float	1	
Salidas			
Cociente	float		El dividendo dividido por el divisor

Si el dividendo es diferente de 0 y el divisor es igual a 0, la simulación se detiene con el siguiente mensaje de error.



5.2.1.3 Multiplicación



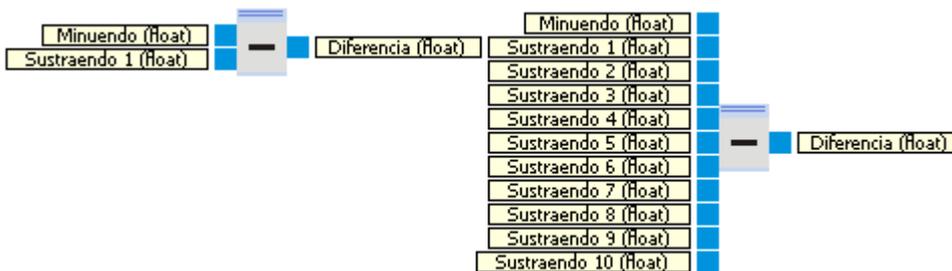
El bloque de función Multiplicación, multiplica números de coma flotante (float). Ver también <http://es.wikipedia.org/wiki/Multiplicación>.

Entradas	Tipo	Predeterminado	Descripción
Factor 1	float	1	
...			
Factor 10	float	1	
Salidas			
Producto	float		"Factor 1" * "Factor 2" * ... * "Factor 10"

5.2.1.3.1 Diálogo



5.2.1.4 Resta



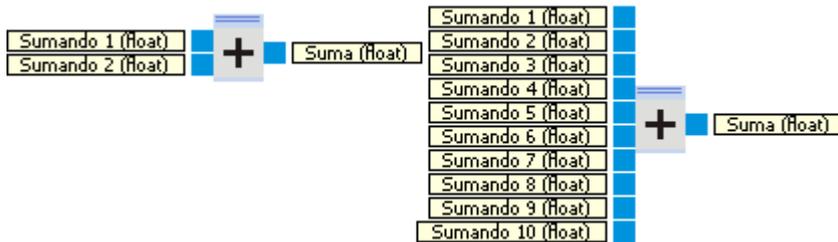
El bloque de función Resta, calcula la diferencia entre un minuendo y hasta 10 sustraendos. Ver también <http://es.wikipedia.org/wiki/Resta>.

Entradas	Tipo	Predeterminado	Descripción
Minuendo	float	0	
Sustraendo 1	float	0	
...			
Sustraendo 10	float	0	
Salidas			
Diferencia	float		Minuendo - "Sustraendo 1" - "Sustraendo 2" - ... - "Sustraendo 10"

5.2.1.4.1 Diálogo



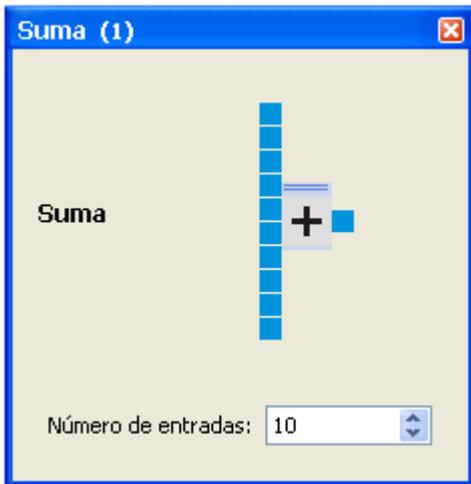
5.2.1.5 Suma



El bloque de función Suma, suma hasta 10 sumandos. Ver también <http://es.wikipedia.org/wiki/Suma>.

Entradas	Tipo	Predeterminado	Descripción
Sumando 1	float	0	
...			
Sumando 10	float	0	
Salidas			
Suma	float		"Sumando 1" + "Sumando 2" + ... + "Sumando 10"

5.2.1.5.1 Diálogo



5.2.2 Operaciones de comparación

5.2.2.1 Diferente



La salida es verdadera (true) si el valor absoluto de $\text{Entrada1} - \text{Entrada2}$ es mayor o igual a epsilon, con $\text{epsilon} = 0.0000002384185792$

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	0	
Entrada2	float	0	
Salidas			
Salida	bool		$\text{fabs}(\text{Entrada1} - \text{Entrada2}) \geq \text{epsilon}$

5.2.2.2 Igual



La salida es verdadera (true) si el valor absoluto de $\text{Entrada1} - \text{Entrada2}$ es inferior a epsilon, con $\text{epsilon} = 0.0000002384185792$

Entradas	Tipo	Predeterminado	Descripción
Entrada 1	float	0	
Entrada 2	float	0	
Salidas			
Salida	bool		$\text{fabs}(\text{Entrada1} - \text{Entrada2}) < \text{epsilon}$

5.2.2.3 Menor o igual



La salida es verdadera (true) si la Entrada1 es menor o igual que la Entrada2.

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	0	
Entrada2	float	0	
Salidas			
Salida	bool		"Entrada1" menor o igual que "Entrada2"

5.2.2.4 Menor



La salida es verdadera (true) si la Entrada1 es menor que la Entrada2.

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	0	
Entrada2	float	0	
Salidas			
Salida	bool		"Entrada1" menor que "Entrada2"

5.2.2.5 Mayor o igual



La salida es verdadera (true) si la Entrada1 es mayor o igual que la Entrada2.

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	0	
Entrada2	float	0	
Salidas			
Salida	bool		"Entrada1" mayor o igual que "Entrada2"

5.2.2.6 Mayor



La salida es verdadera (true) si la Entrada1 es mayor que la Entrada2.

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	0	
Entrada2	float	0	
Salidas			
Salida	bool		"Entrada1" mayor que "Entrada2"

5.2.3 Funciones

5.2.3.1 Valor absoluto



Da el valor absoluto de una entrada

Entradas	Tipo	Predeterminado	Descripción
Entrada	float	0	
Salidas			
Salida	float		abs(Entrada)

5.2.3.2 Función de transferencia



Con la función de transferencia es posible realizar cualquier correlación de la entrada x con la salida y.

Entradas	Tipo	Predeterminado	Descripción
x	float	0	
Salidas			
x	float		véase Diálogo 63

5.2.3.2.1 Diálogo

Con el diálogo del bloque de Función de transferencia es posible definir puntos de interpolación para la asignación $y(x)$. Los puntos de interpolación predeterminados son:

Librería de bloques de función

$$p_0 = (x_0, y_0) = (0, 0)$$

$$p_1 = (x_1, y_1) = (10, 10)$$

Estos puntos definen la siguiente asignación $y(x)$

$$y = y_0 \text{ si } x \leq x_0$$

$$y = x \text{ si } x > x_0 \text{ y } x \leq x_1$$

$$y = y_1 \text{ si } x > x_1$$

Límites

$p_0 = (x_0, y_0)$ es el primer punto de interpolación

$p_n = (x_n, y_n)$ es el último punto de interpolación

$$\text{Si } x < x_0: y = y_0$$

$$\text{Si } x > x_n: y = y_n$$

Asignación

Si se tiene una lista de puntos de interpolación p_0, p_1, \dots, p_n , la asignación $y(x)$ viene dada por:

$$y = y_0 \text{ si } x \leq x_0$$

$$y = (y_1 - y_0) / (x_1 - x_0) * (x - x_0) + y_0 \text{ si } x > x_0 \text{ y } x \leq x_1$$

$$y = (y_2 - y_1) / (x_2 - x_1) * (x - x_1) + y_1 \text{ si } x > x_1 \text{ y } x \leq x_2$$

...

$$y = y_n \text{ si } x > x_n$$

Mover puntos

Los puntos de interpolación pueden moverse, añadirse o quitarse. Para mover un punto de interpolación puede utilizar la vista gráfica y desplazar el punto con el puntero del ratón. En la vista de tabla, puede editarse los valores de los puntos de interpolación x, y . El valor x de un punto de interpolación nunca puede ser menor que el valor x del anterior punto de interpolación ni mayor que el siguiente.

Añadir puntos

En la vista gráfica puede añadirse un nuevo punto en cualquier parte haciendo clic con el botón derecho del ratón y utilizando el menú de contexto.

Insertar punto
Importar del portapapeles
Exportar al portapapeles
Ayuda

En la vista de tabla, el menú de contexto aparece haciendo clic con el botón derecho en una fila.

Insertar punto antes
Insertar punto después
Eliminar punto
Importar del portapapeles
Exportar al portapapeles
Ayuda

Puede elegir insertar el nuevo punto antes o después de la fila actual.

Borrar puntos

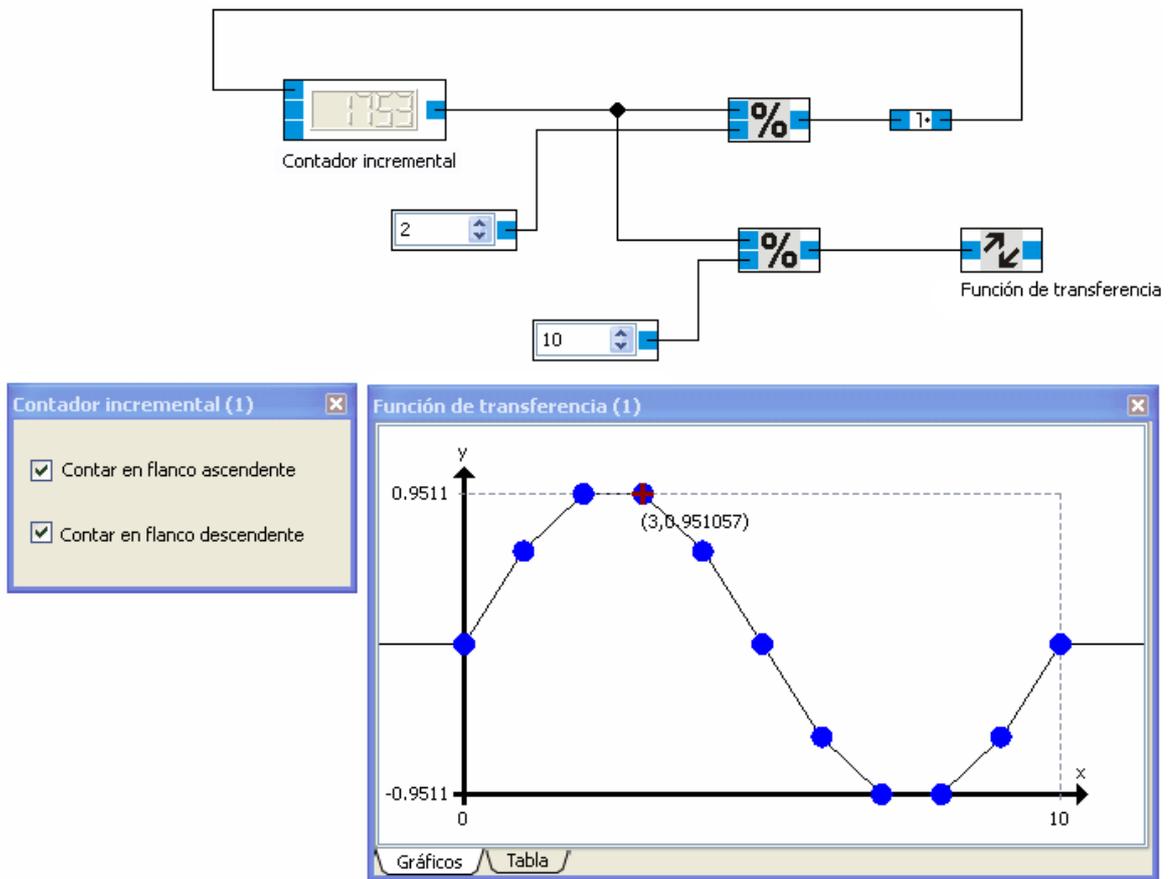
Tanto en la vista gráfica como en la tabla, pueden borrarse puntos haciendo clic con el botón derecho del ratón y seleccionando la opción en el menú de contexto. Si sólo queda un único punto, la función de borrado permanece desactivada.

Importar/Exportar puntos de interpolación

Puede utilizarse el portapapeles para importar y exportar la lista de puntos de interpolación. Con ello pueden intercambiarse datos con programas como Excel o Matlab. La función para Importar/Exportar está disponible por medio del menú de contexto tanto en la vista gráfica como en la de tabla.

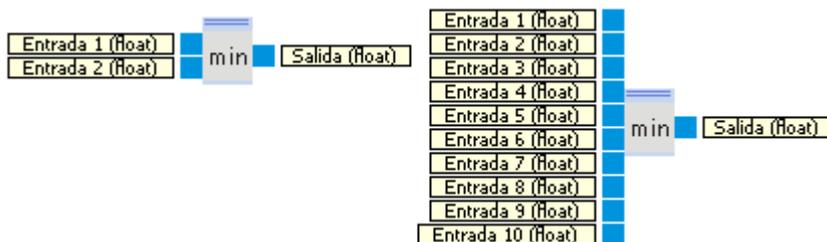
Librería de bloques de función

5.2.3.2.2 Ejemplo



El contador incremental aumenta una unidad en cada etapa de la simulación. El valor de conteo es restringido al margen [0,10]. La Función de transferencia define una onda senoidal con 10 puntos de interpolación

5.2.3.3 Mínimo



El valor de la salida es el valor mínimo de entre todas las entradas.

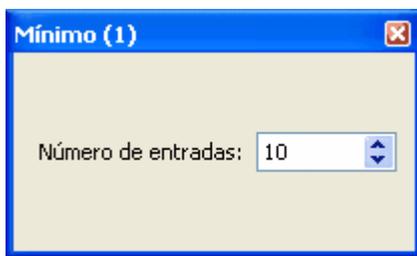
Entradas	Tipo	Predeterminado	Descripción

Entrada1	float	1e+037	
...			
Entrada10	float	1e+037	
Salidas			
Salida	float		min("Entrada1", "Entrada2", ..., "Entrada10")

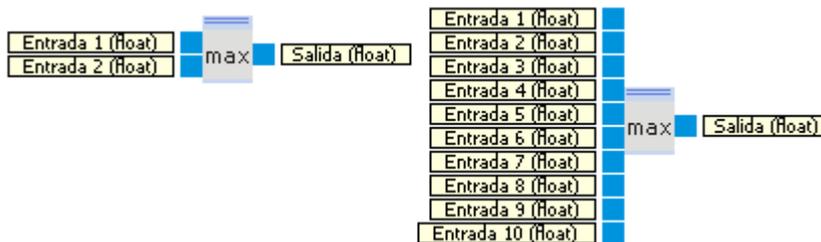
1e+037 = 10 pow 37

número en coma flotante más grande posible

5.2.3.3.1 Diálogo



5.2.3.4 Máximo



El valor de la salida es el valor máximo de entre todas las entradas.

Entradas	Tipo	Predeterminado	Descripción
Entrada1	float	-1e+037	
...			

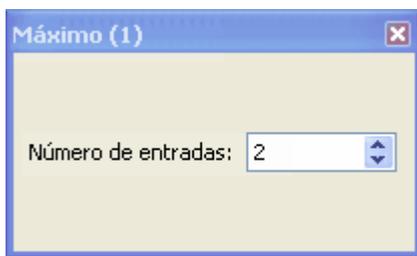
Librería de bloques de función

Entrada10	float	-1e+037	
Salidas			
Salida	float		max("Entrada1", "Entrada2", ..., "Entrada10")

-1e+037 = - (10 pow 37)

número en coma flotante más pequeño posible

5.2.3.4.1 Diálogo



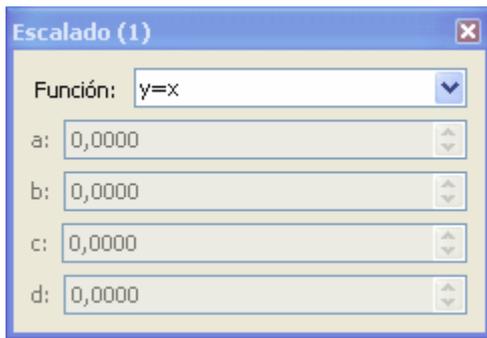
5.2.3.5 Escalado



Escalado sencillo de valores.

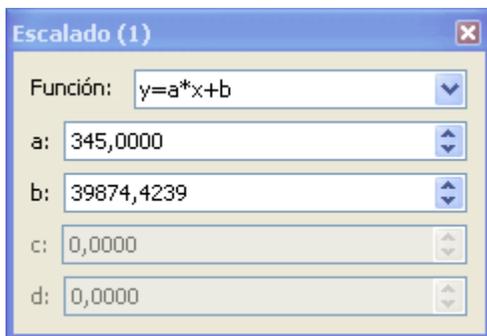
Entradas	Tipo	Predeterminado	Descripción
x	float	0	
Salidas			
y	float		véase Diálogo 69

5.2.3.5.1 Diálogo



Elegir una función de un cuadro combinado. La función predeterminada es la asignación de identidad ($y=x$).

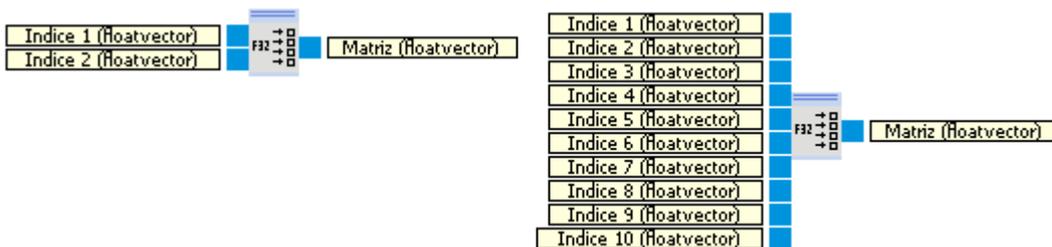
Según la función seleccionada, los parámetros son editables o no. Si elige, por ejemplo, la función " $y=a*x+b$ ", podrá editar los parámetros "a" y "b".



La asignación aquí es $y = 345 * x - 39874,4239$

5.2.4 Matrices

5.2.4.1 Componer matriz de números reales

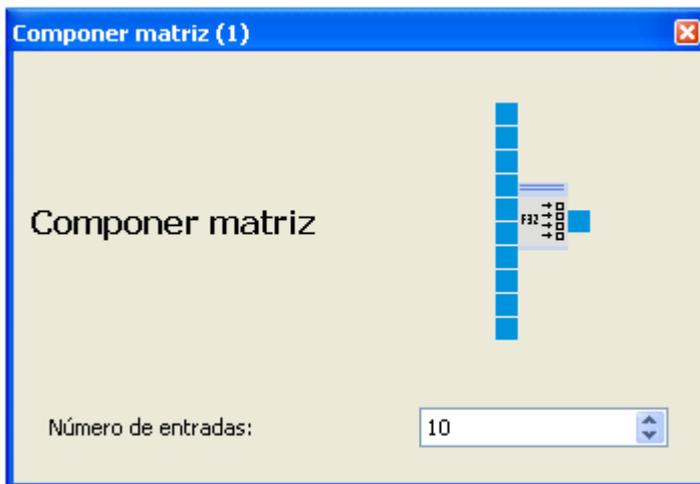


Librería de bloques de función

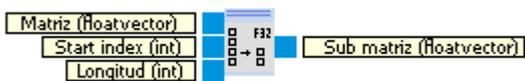
El bloque para componer matrices de números reales o de coma flotante (float array) crea una matriz de hasta 10 valores reales o matrices. Para la conversión de tipo de valores reales o de coma flotante (float values) a matrices de números reales o de coma flotante (float arrays), véase [conversión de tipos](#) [20].

Entradas	Tipo	Predeterminado	Descripción
Índice 1	float array	empty array	
...			
Índice 10	float array	empty array	
Salidas			
Array	float array	empty array	(Índice 1, ..., Índice 10)

5.2.4.1.1 Diálogo



5.2.4.2 Descomponer matriz de números reales



El bloque para descomponer una matriz de números reales (float array) extrae una submatriz de una matriz.

Entradas	Tipo	Predeterminado	Descripción
Array	float array	empty array	La matriz a descomponer.
Índice inicial	int	1	El valor en la posición Índice inicial de la matriz, será el primer valor de la matriz resultante.
Longitud	int	1	La matriz resultante consiste en la cantidad de valores que indica Longitud , empezando en la posición que indica el valor Índice inicial de la matriz de entrada.
Salidas			
Sub array	float array	empty array	(Array[Índice inicial], ..., Array[Índice inicial + Longitud - 1])

5.2.4.3 Acceso al índice de la matriz de números reales



El módulo de acceso al índice permite el acceso a los valores individuales de una matriz de números reales.

Entradas	Tipo	Predeterminado	Descripción
Array	float array	empty array	Matriz a la que se accede.
Índice	int	1	Índice del valor a acceder.
Salidas			
Valor	float	0	Valor de la posición Índice .

5.3 Cálculo vectorial

Esta categoría contiene los métodos básicos de análisis vectorial para vectores de dos dimensiones.

5.3.1 Operaciones con vectores

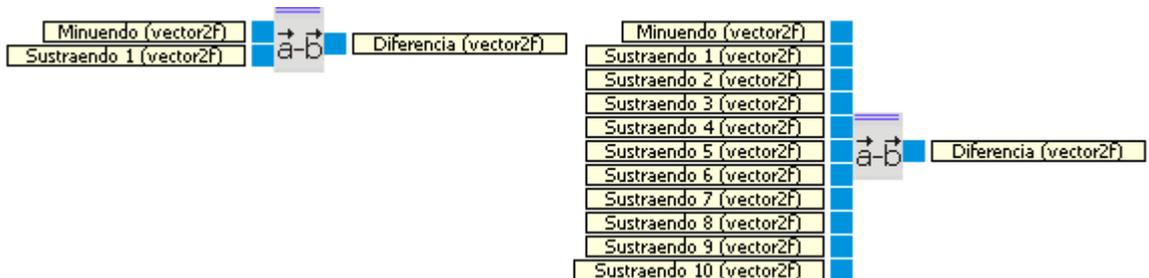
5.3.1.1 Producto escalar



Da el producto escalar (o producto interno, interior o punto) de dos vectores. Ver también http://es.wikipedia.org/wiki/Producto_escalar.

Entradas	Tipo	Predeterminado	Descripción
Vector 1	vector2f	(0, 0)	
Vector 2	vector2f	(0, 0)	
Salidas			
Producto	float		

5.3.1.2 Resta



El bloque de función Resta, calcula la diferencia entre un minuendo y hasta 10 sustraendos. Ver también [http://es.wikipedia.org/wiki/Vector_\(espacio_euclideo\)](http://es.wikipedia.org/wiki/Vector_(espacio_euclideo)).

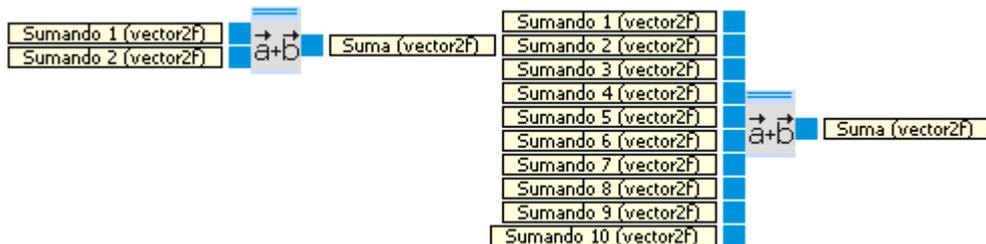
Entradas	Tipo	Predeterminado	Descripción
Minuendo	vector2f	(0, 0)	
Sustraendo 1	vector2f	(0, 0)	
...			

Sustraendo 10	vecto r2f	(0, 0)	
Salidas			
Diferencia	vecto r2f		Minuendo - "Sustraendo 1" - "Sustraendo 2" - ... - "Sustraendo 10"

5.3.1.2.1 Diálogo



5.3.1.3 Suma



El bloque de función Suma de vectores, suma hasta 10 sumandos. Ver también [http://es.wikipedia.org/wiki/Vector_\(espacio_euclídeo\)](http://es.wikipedia.org/wiki/Vector_(espacio_euclídeo)).

Entradas	Tipo	Predeterminado	Descripción
Sumando 1	vecto r2f	(0, 0)	

Librería de bloques de función

...			
Sumando 10	vecto r2f	(0, 0)	
Salidas			
Suma	vecto r2f		"Sumando 1" + "Sumando 2" + ... + "Sumando 10"

5.3.1.3.1 Diálogo



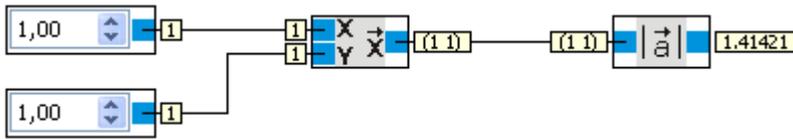
5.3.1.4 Norma



Calcula la norma euclídea del vector de entrada. Ver también http://es.wikipedia.org/wiki/Norma_vectorial.

Entradas	Tipo	Predeterminado	Descripción
Vector	vecto r2f	(0, 0)	
Salidas			
Norma	float		

5.3.1.4.1 Ejemplo



La norma del vector (1, 1) es la raíz cuadrada de $1+1 = 1.41421\dots$

5.3.2 Operaciones con elementos

5.3.2.1 División



Por elemento división de Vector por Divisor.

Entradas	Tipo	Predeterminado	Descripción
Vector	vector2f	(0, 0)	
Divisor	float	1	
Salidas			
Resultado	vector2f		Vector = (x0, x1) Resultado = (x0 / Divisor, x1 / Divisor)

5.3.2.2 Resta



Por elemento resta de Minuendo del Vector.

Entradas	Tipo	Predeterminado	Descripción
----------	------	----------------	-------------

Vector	vecto r2f	(0, 0)	
Minuendo	float	0	
Salidas			
Resultado	vecto r2f		Vector = (x0, x1) Resultado = (x0 - Minuendo, x1 - Minuendo)

5.3.2.3 Suma



Por elemento suma de Sumando a Vector.

Entradas	Tipo	Predeterminado	Descripción
Vector	vecto r2f	(0, 0)	
Sumando	float	0	
Salidas			
Resultado	vecto r2f		Vector = (x0, x1) Resultado = (Sumando + x0, Sumando + x1)

5.3.2.4 Multiplicación



Por elemento multiplicación de vector por factor.

Entradas	Tipo	Predeterminado	Descripción
Vector	vecto r2f	(0, 0)	
Factor	float	1	

Salidas			
Resultado	vecto r2f		Vector = (x0, x1) Resultado = (Factor * x0, Factor * x1)

5.3.3 Transformaciones

5.3.3.1 Vector a Polar



Descompone un Vector en sus componentes polares. Ver también http://es.wikipedia.org/wiki/Sistema_de_coordenadas_polares.

Entradas	Tipo	Predeterminado	Descripción
Vector	vecto r2f	(0, 0)	
Salidas			
Longitud	float		La longitud (norma) del Vector
Phi	float		Ángulo en grados entre el Vector y el eje x

5.3.3.2 Vector a Cartesiano



Descompone un Vector en sus componentes cartesianos. Ver también http://es.wikipedia.org/wiki/Sistema_de_coordenadas_cartesianas.

Entradas	Tipo	Predeterminado	Descripción
Vector	vecto r2f	(0, 0)	

Salidas			
x	float		componente x del Vector
y	float		componente y del Vector

5.3.3.3 Polar a Vector



Crea un vector a partir de su longitud y orientación.

Entradas	Tipo	Predeterminado	Descripción
Longitud	float	0	Longitud (módulo) del Vector.
Phi	float	0	Ángulo en grados entre el vector y el eje x
Salidas			
Vector	vector2f		Vector con longitud Longitud y orientación Phi .

5.3.3.4 Cartesiano a Vector



Crear un vector a partir de sus componentes cartesianos.

Entradas	Tipo	Predeterminado	Descripción
x	float	0	componente x.
y	float	0	componente y.
Salidas			
Vector	vector2f		Vector (x, y).

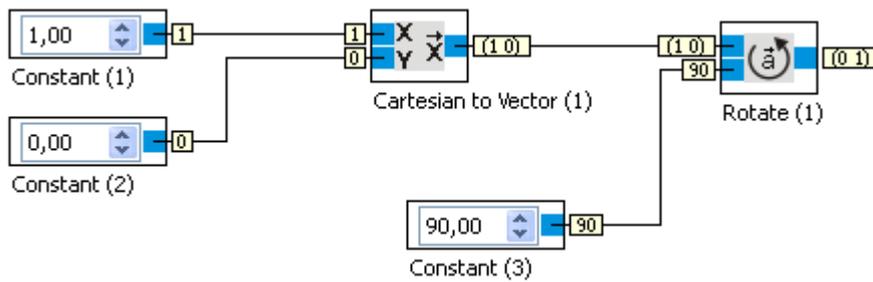
5.3.3.5 Rotar



Rota el vector por el valor especificado en grados.

Entradas	Tipo	Predeterminado	Descripción
Vector	vector2f	(0, 0)	
Phi	float	0	Ángulo de rotación
Salidas			
Resultado	vector2f		Vector girando por el ángulo Phi .

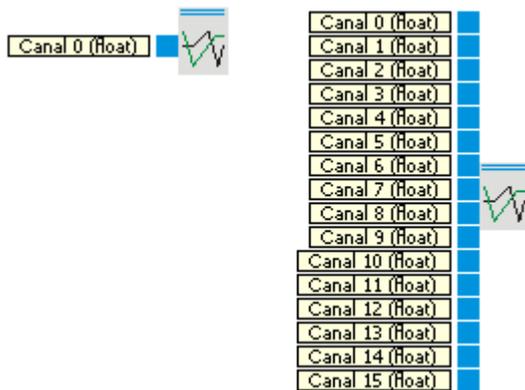
5.3.3.5.1 Ejemplo



5.4 Visualizador

Esta categoría contiene los bloques de función para la visualización de datos.

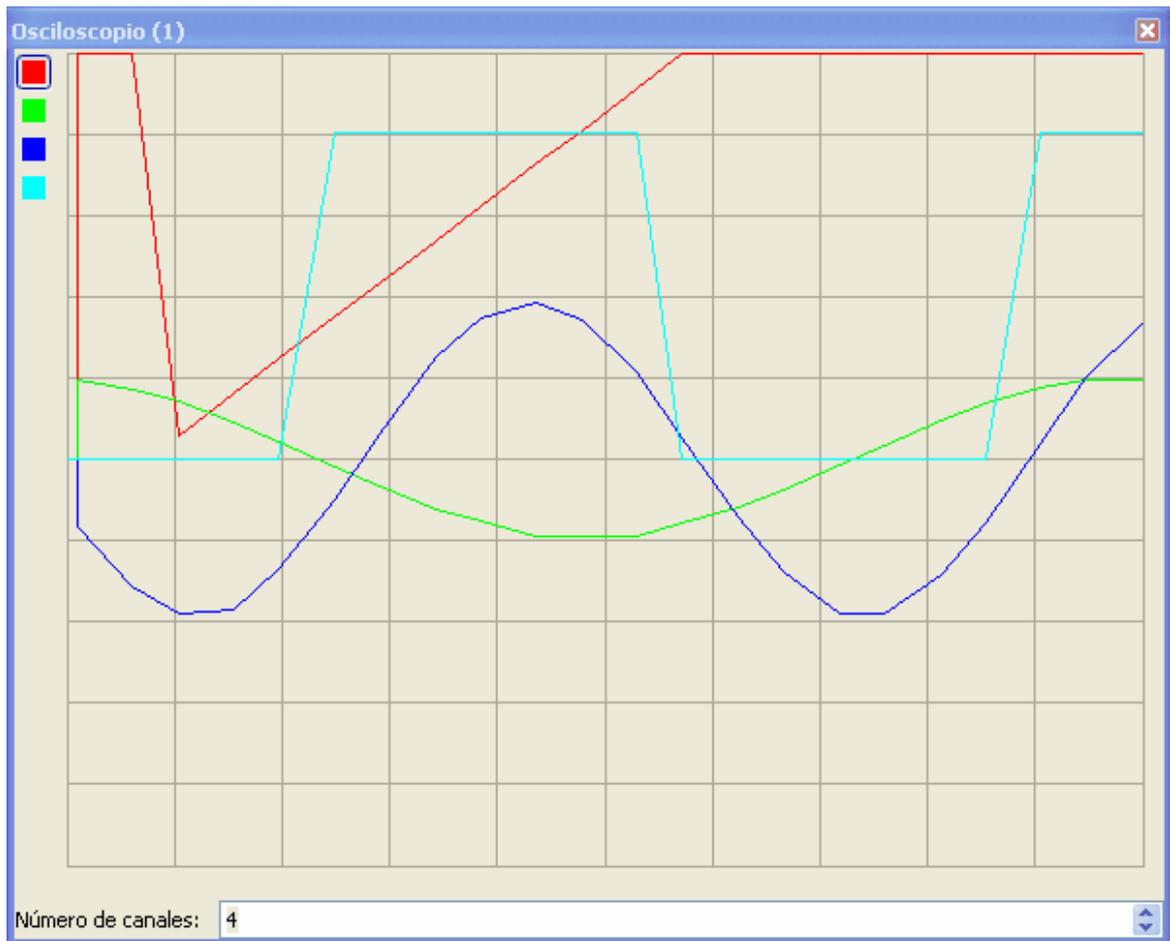
5.4.1 Osciloscopio



El Osciloscopio se utiliza para visualizar hasta 16 canales.

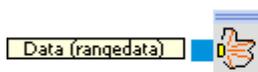
Entradas	Tipo	Predeterminado	Descripción
Canal 0	float	0	
Canal 1	float	0	
...			
Canal 16	float	0	

5.4.1.1 Diálogo



El diálogo visualiza las señales de los canales. Para cada canal pueden cambiarse los ajustes tales como la amplificación, etc.. También es posible desactivar cada uno de los canales.

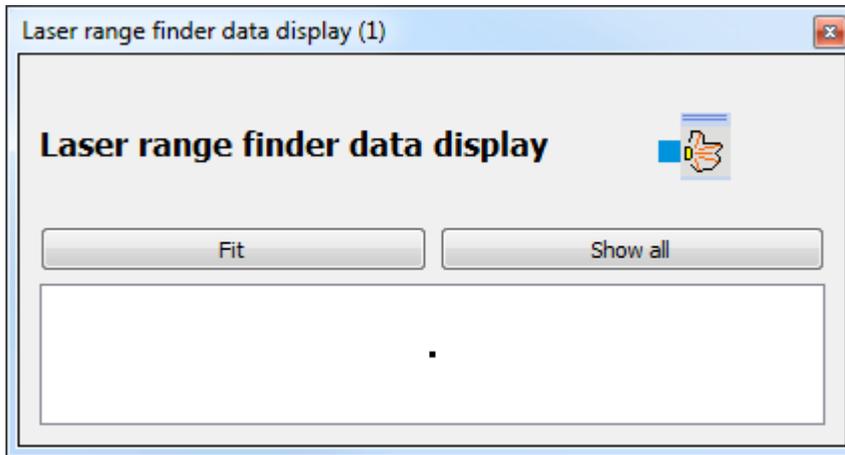
5.4.2 Pantalla de datos del telémetro láser



La pantalla de datos permite visualizar los datos procedentes de un telémetro o sensor de distancia láser.

Entradas	Tipo	Descripción
Datos	datos del telémetro láser	

5.4.2.1 Diálogo



5.5 Procesamiento de imágenes

Esta categoría contiene bloques de función para el procesamiento de imágenes.

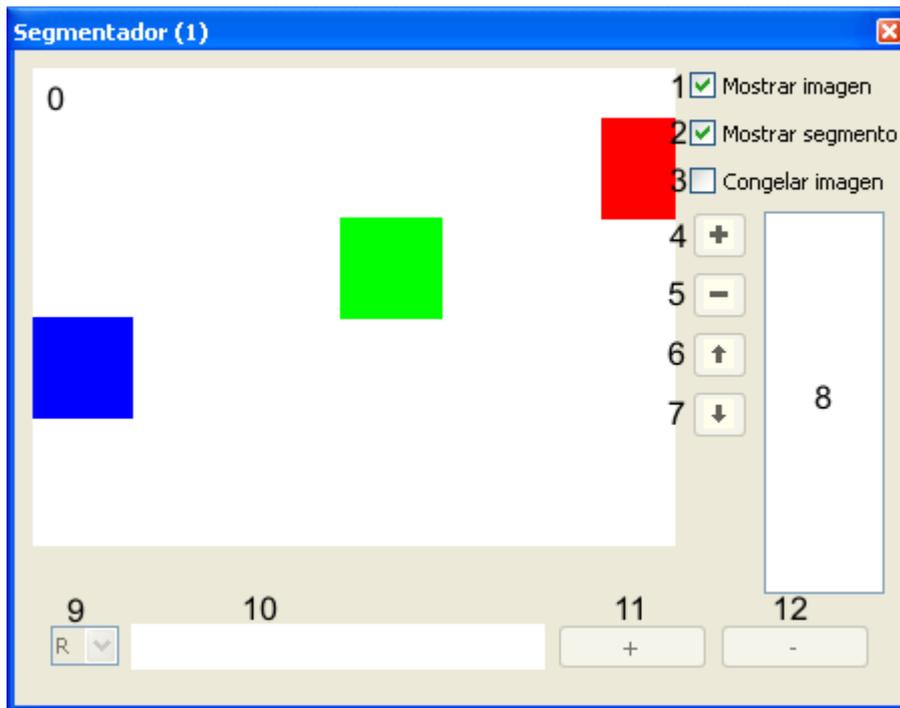
5.5.1 Segmentador



El bloque de función Segmentador, divide la imagen de entrada en múltiples segmentos. La imagen de salida contiene una lista de los segmentos hallados.

Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Imagen de entrada
Salidas			
Salida	imagen		Imagen de salida aumentada con la lista de los segmentos hallados.

5.5.1.1 Diálogo



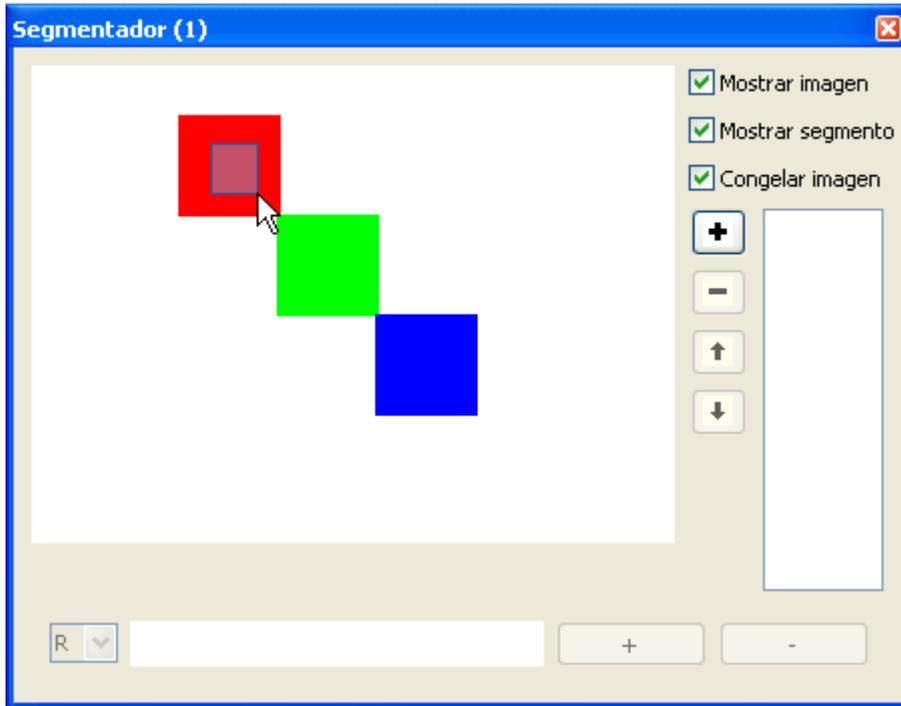
Botón/ Descripción

Displa

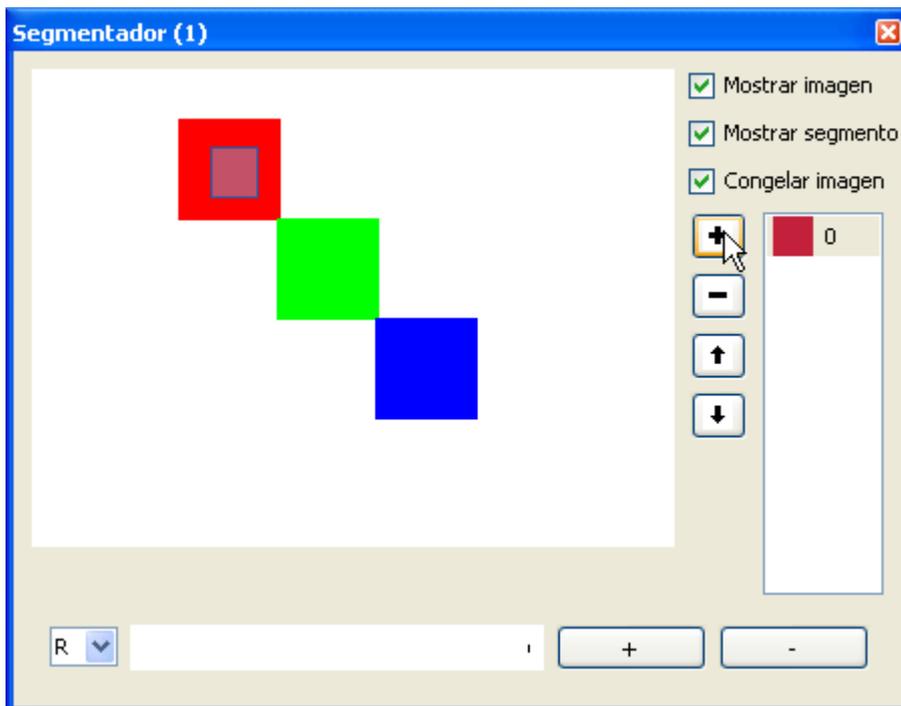
y

- 0 Visualización de la imagen de entrada y de los segmentos encontrados
- 1 Si está marcado, se visualiza la imagen de entrada
- 2 Si está marcado, se visualizan los segmentos encontrados
- 3 Si está marcado, se congela la imagen actual de entrada
- 4 Añadir la selección actual de la imagen de entrada a la lista de segmentos
- 5 Borrar un segmento
- 6 Desplazar segmento arriba
- 7 Desplazar segmento abajo
- 8 Lista de segmentos
- 9 Selector del canal de color para la optimización de segmentos
- 10 Indicación de valores dentro del canal seleccionado del segmento activo actual.
- 11 Cerrar intervalos dentro de los valores del canal seleccionado.
- 12 Dispersar valores del canal seleccionado

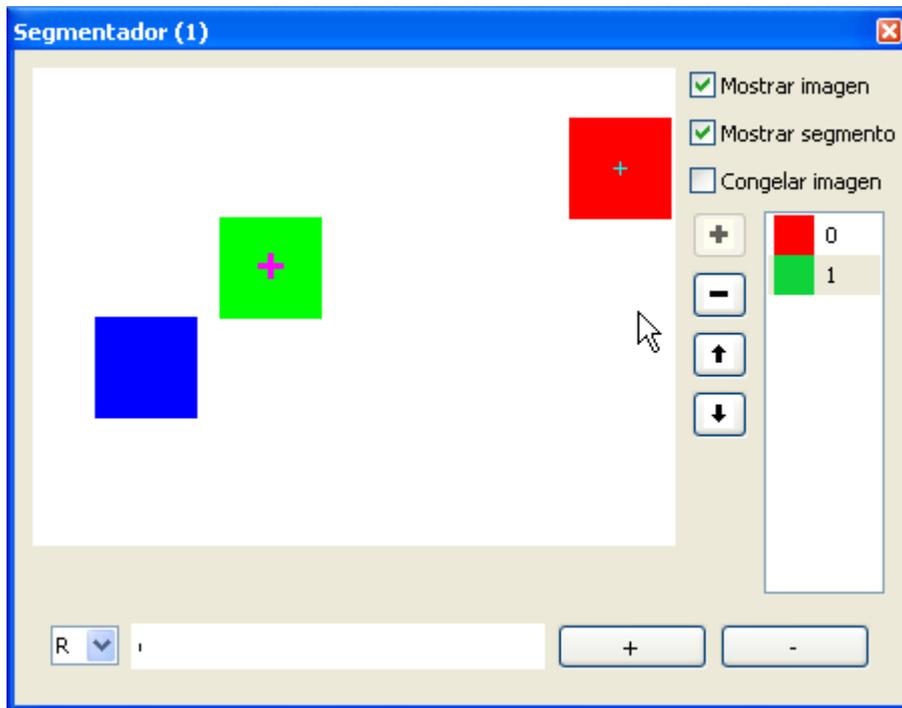
Para reconocer el cuadro rojo como segmento individual, marque una región dentro del cuadro rojo con el ratón.



Haga clic en el signo "+" (botón 4) para añadir la selección a la lista de segmentos.

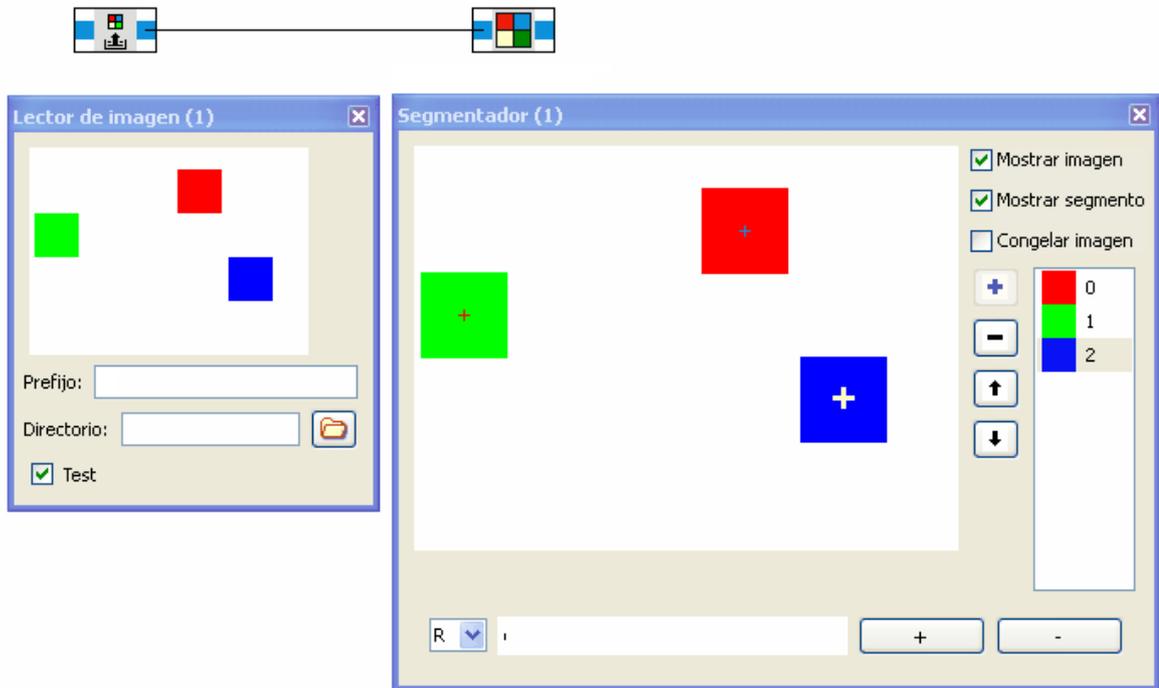


El centro de enfoque del segmento se muestra como una cruz. Cuando cambia la imagen (desactivar la opción congelar imagen) el centro de enfoque se mueve con el cuadrado rojo. Ahora repita el procedimiento para añadir el cuadro verde a la lista de segmentos.



Ahora hay dos segmentos en la lista. El segmento actualmente seleccionado es marcado con una cruz gruesa.

5.5.1.2 Ejemplo



El bloque de función [lector de imagen](#)^[122] funciona en modo de test y genera una secuencia de imágenes de test. El segmentador busca regiones conectadas en la imagen de entrada que se ajusten a los colores de la lista de segmentos. Se muestra el centro de enfoque de los segmentos encontrados.

5.5.2 Extractor de segmentos

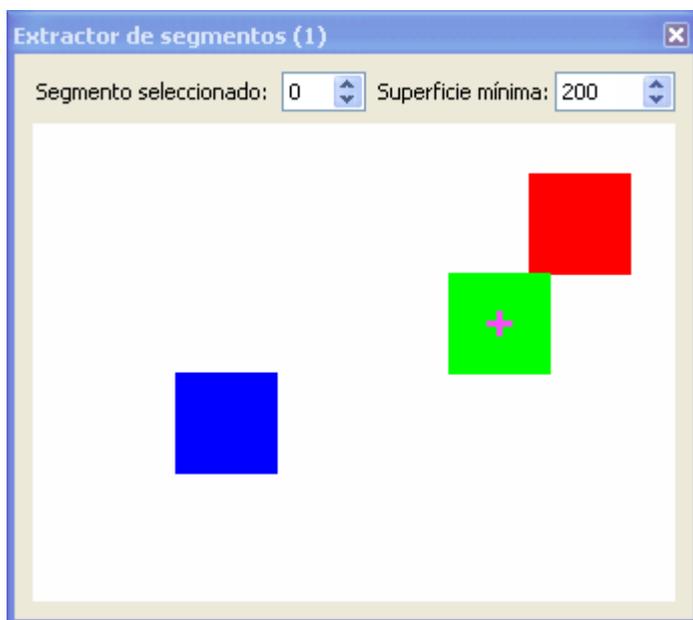


Obtiene la posición y tamaño de un segmento a partir de una imagen aumentada con una lista de segmentos con el bloque de función [Segmentador](#)^[82].

Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Imagen aumentada
Segmento seleccionado	int	0	Número del segmento del que se requiere información.
Superficie mínima	int	200	La salida del segmento hallado sólo será verdadera (true) si el segmento contiene por lo menos el número de píxeles indicado aquí.

Salidas			
x	int		coordenada x del centro de enfoque del segmento hallado. Si no se halla ningún segmento, x=0.
y	int		coordenada y del centro de enfoque del segmento hallado. Si no se halla ningún segmento, y=0.
Área	int		Número de píxeles dentro del segmento. Si no se halla ningún segmento, Área=0.
Segmento encontrado	bool		Verdadero (true) si se ha hallado un segmento. De lo contrario, falso (false).

5.5.2.1 Diálogo



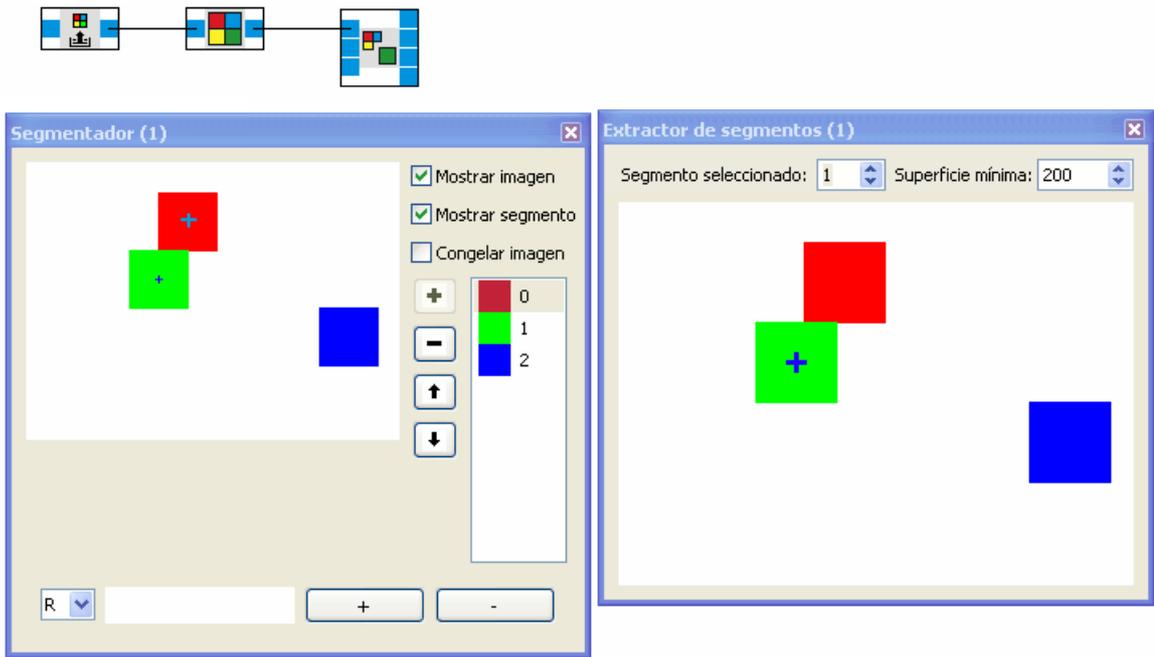
Segmento El spinbox es editable si el conector de entrada "Segmento seleccionado" no está seleccionado conectado. Indica el número de segmento a buscar.

o

Superficie mínima El spinbox es editable si el conector de entrada "Superficie mínima" no está conectado. Indica el número mínimo de píxeles que debe contener el segmento.

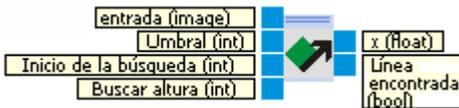
Muestra los segmentos en la imagen de entrada. El segmento seleccionado es marcado con una cruz (si se ha hallado el segmento).

5.5.2.2 Ejemplo



El lector de imagen crea una secuencia de test con tres cuadrados coloreados. El segmentador busca las regiones rojas verdes y azules. El extractor de segmentos busca el segmento con el número 1 (el segmento verde) y marca su centro de enfoque con una cruz.

5.5.3 Detector de líneas

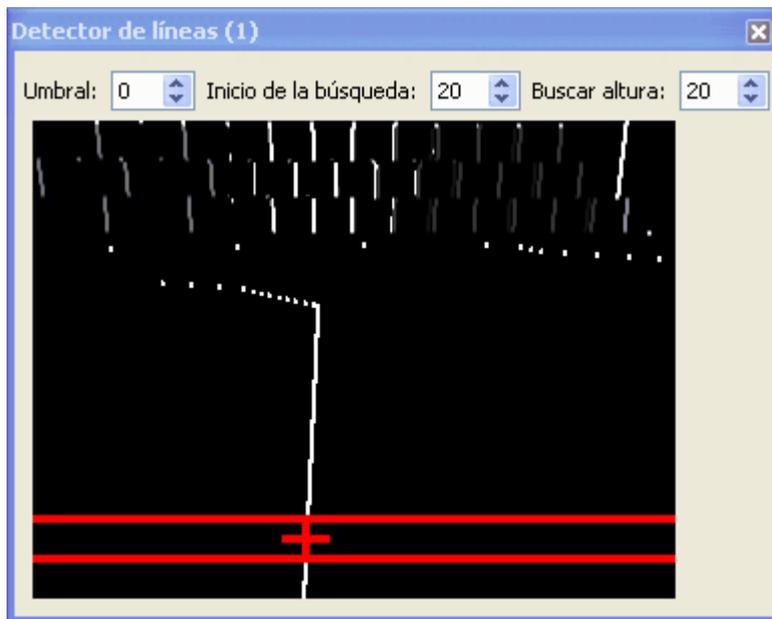


Busca una línea en la imagen de entrada.

Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Imagen de entrada
Umbral	int	0	El umbral define la sensibilidad del algoritmo de detección de líneas respecto a discontinuidades en la imagen. Para eliminar ruidos, elegir un umbral mayor. Margen: [0, 255]
Inicio de la búsqueda	int	20	El algoritmo inicia la búsqueda de una línea empezando en "Inicio de búsqueda" desde abajo.

Altura de búsqueda	int	20	La imagen es buscada desde abajo hacia arriba para la detección de bordes. El valor límite define el número de líneas buscadas en la imagen empezando por abajo, más "inicio de la búsqueda" para detectar un segmento en forma de línea.
Salidas			
x	int		posición x de la línea situada en el borde inferior de la imagen.
Línea encontrada	bool		Verdadero (true) si se ha hallado una línea. De lo contrario, falso (false).

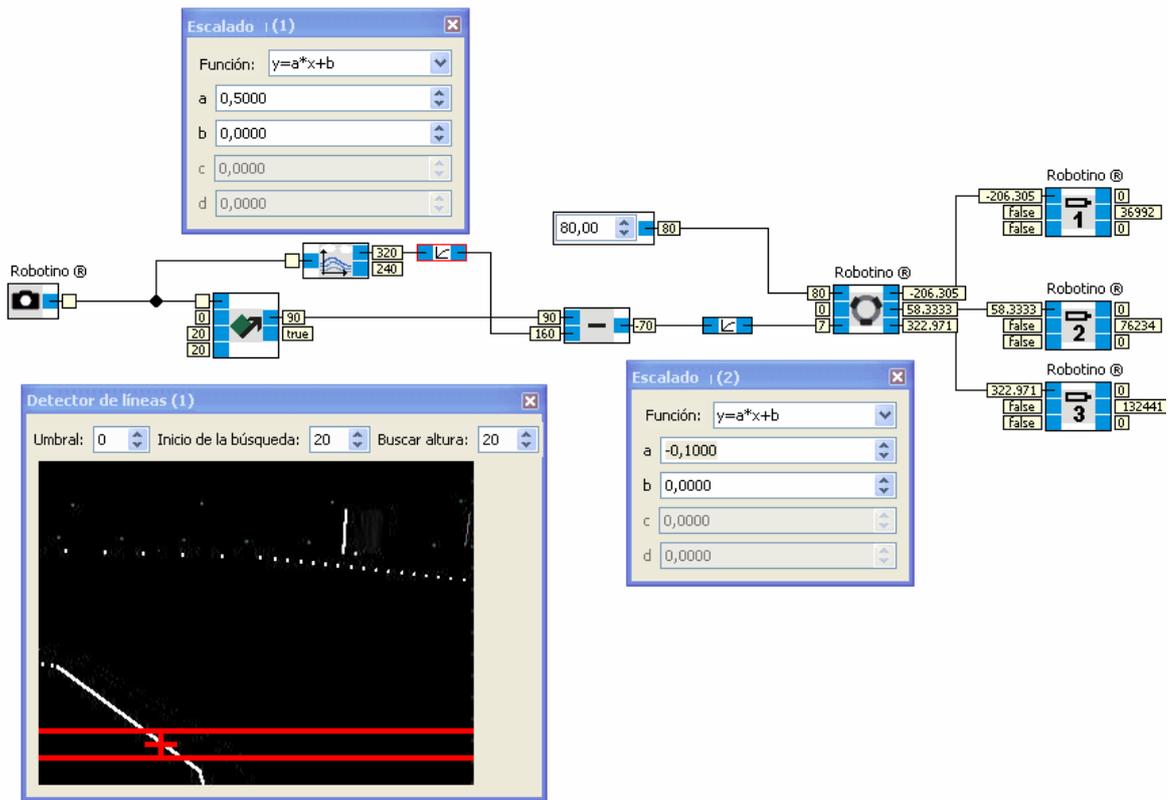
5.5.3.1 Diálogo



El área en la que se busca la línea está marcada por las líneas rojas horizontales. La línea inferior señala el "Inicio de la búsqueda". La línea superior indica "Inicio de la búsqueda" + "Altura de búsqueda".

La cruz roja "+" marca el borde de oscuro a claro de la línea vista de izquierda a derecha.

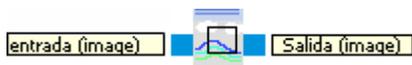
5.5.3.2 Ejemplo



La imagen de la cámara del Robotino (aquí del simulador de Robotino) se utiliza como entrada para el detector de líneas. Utilizamos el bloque de función [Información de la imagen](#)^[92] para asignar ("mapear") la posición x de la línea desde el margen [0, ancho de la imagen] hasta [- ancho de imagen/2, ancho de imagen/2] que en nuestro caso es [-160, 160]. El bloque de función [Escalado](#)^[68] se utiliza para cambiar el signo y para escalar la salida del bloque de función [resta](#)^[58].

El valor obtenido puede ser utilizado directamente para hacer girar el Robotino, de forma que gire a la derecha si la línea se halla a su derecha y a la izquierda si se halla a su izquierda. Con una velocidad de avance constante, Robotino sigue la línea.

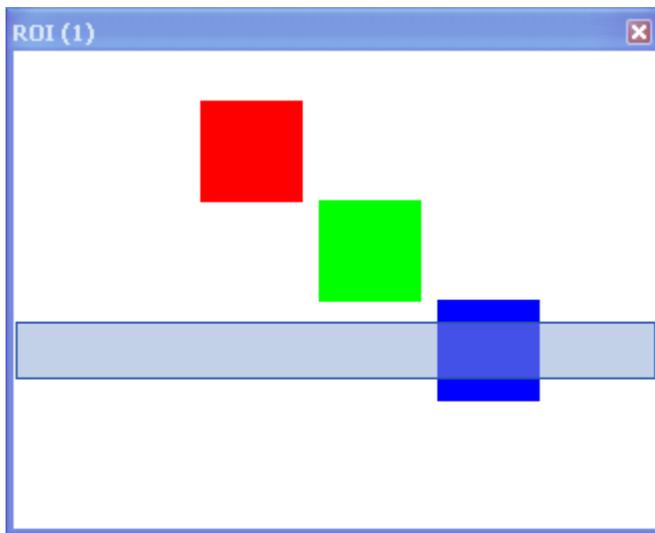
5.5.4 ROI



Seleccionar una región interesante dentro de la imagen de entrada (ROI = Region Of Interest / Región de interés).

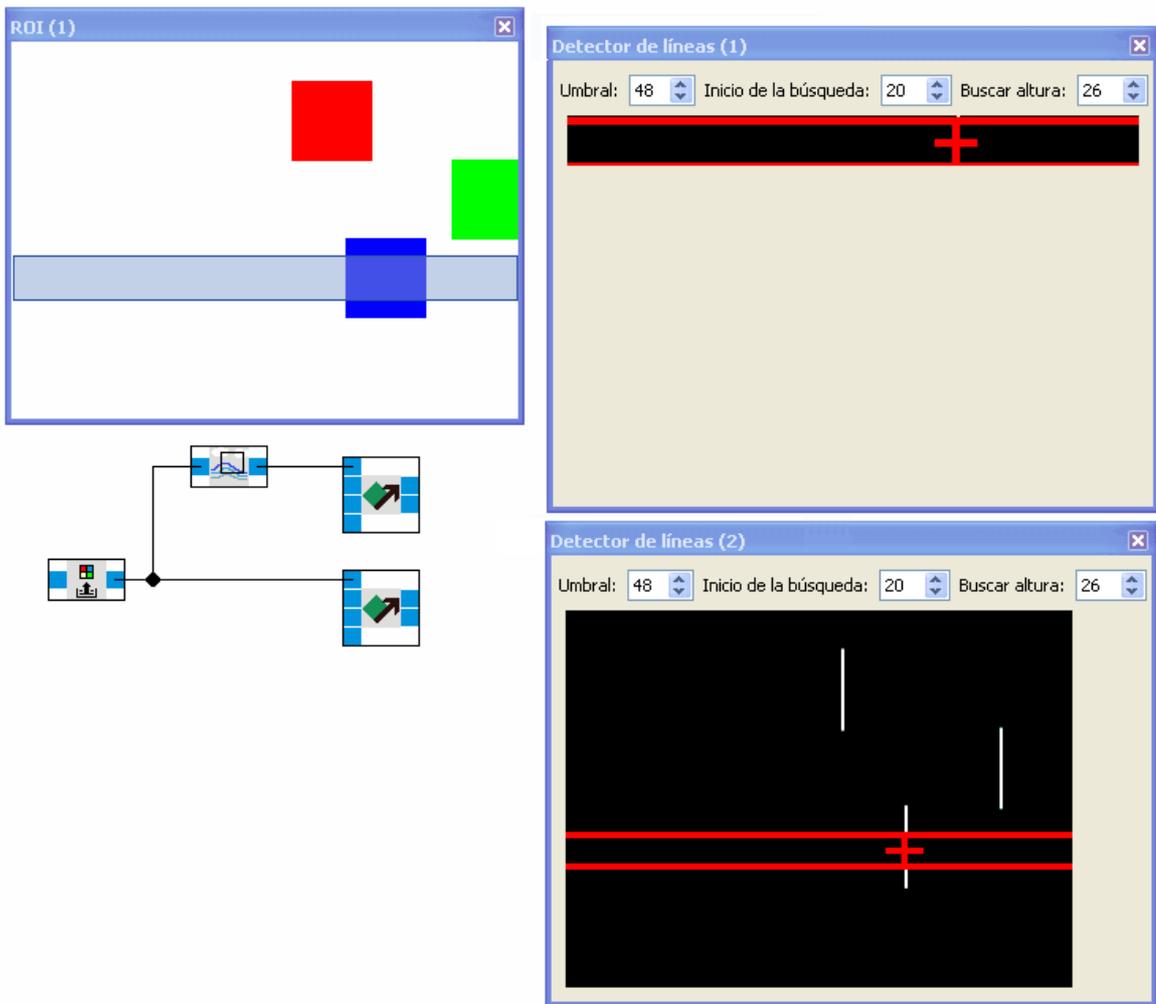
Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Imagen de entrada
Salidas			
Salida	imagen		La imagen de salida es aumentada con la información ROI. El posterior procesamiento de la imagen se realiza sólo en la región o zona señalada como ROI.

5.5.4.1 Diálogo



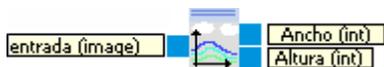
La región de interés puede determinarse con el ratón. Haga clic en un punto y arrastre para delimitar una región.

5.5.4.2 Ejemplo



El lector de imagen genera una secuencia de test de imágenes. El detector de línea inferior utiliza toda la imagen, mientras que el detector de línea superior busca sólo en la ROI.

5.5.5 Información de la imagen

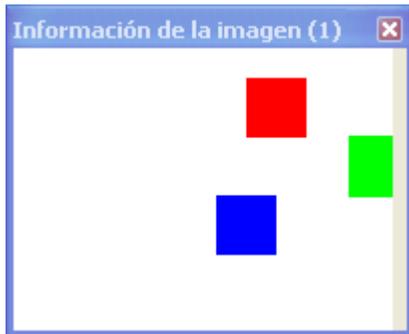


Obtener el ancho y alto de la imagen de entrada.

Entradas	Tipo	Predeterminado	Descripción

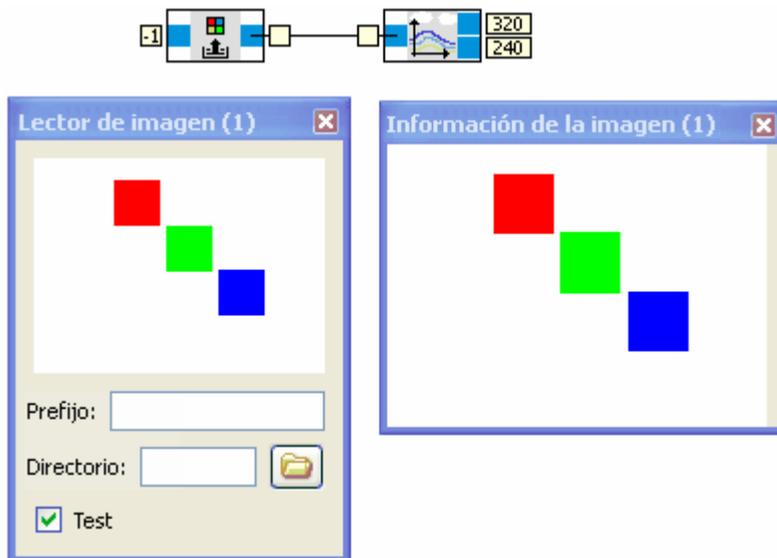
Entrada	imagen		Imagen de entrada
Salidas			
Ancho	int		Ancho de la imagen en píxeles
Alto	int		Alto de la imagen en píxeles

5.5.5.1 Diálogo



El diálogo muestra la imagen de entrada

5.5.5.2 Ejemplo



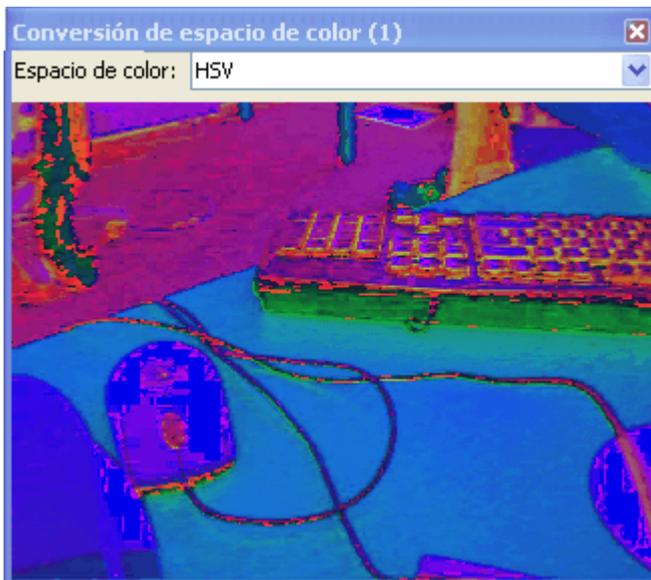
La imagen de la secuencia de test generada por el lector de imagen tiene una resolución de 320 x 240 píxeles.

5.5.6 Conversión de espacio de color



Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Imagen de entrada
Salidas			
Salida	imagen		Imagen convertida

5.5.6.1 Diálogo



En el diálogo de la conversión del espacio de color puede seleccionarse el espacio de color de destino.

5.6 Generadores

Esta categoría contiene diversos bloques de función para generar señales.

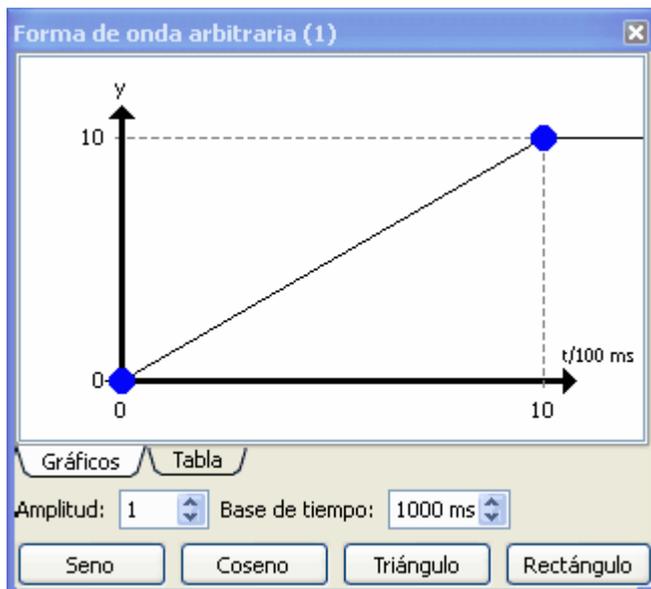
5.6.1 Generador de forma de onda arbitraria



Generador de formas de onda ajustables. Ver también http://en.wikipedia.org/wiki/Arbitrary_waveform_generator.

Entradas	Tipo	Predeterminado	Descripción
Salidas			
Salida	float		La señal generada.

5.6.1.1 Diálogo



La parte superior del diálogo es similar a diálogo conocido de la [Función de transferencia](#) [63].

Además, el generador de forma de onda arbitraria tiene los siguientes parámetros y botones:

Amplitud La salida del generador se multiplica por este valor.

Base de tiempo de la unidad del eje x. En el ejemplo actual, con una base de tiempo de 100ms, se alcanza el valor 10 al cabo de 1s.

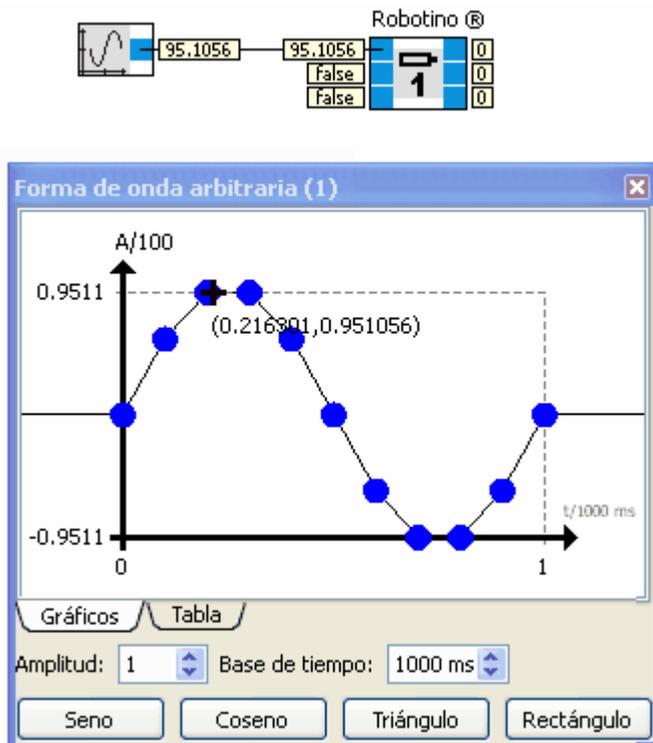
Seno Genera puntos de interpolación que se aproximan a una onda senoidal.

Coseno Genera puntos de interpolación que se aproximan a una onda cosenoidal.

Librería de bloques de función

Triángulo Genera puntos de interpolación que se aproximan a una onda triangular.
Rectángulo Genera puntos de interpolación que se aproximan a una onda rectangular.
o

5.6.1.2 Ejemplo



El Motor 1 del Robotino gira controlado por una onda de forma senoidal.

5.6.2 Constante



Generación de un valor constante El tipo de constante, así como su visualización gráfica, cambian según el tipo de entrada a la que se halla conectada su salida.

La introducción del valor puede realizarse directamente dentro del programa. No se abre un cuadro de diálogo. El valor puede editarse **directamente** .

Entradas	Tipo	Predeterminado	Descripción
Salidas			
Valor	float, int, bool	0 / false	El valor visualizado.

5.6.3 Temporizador



Mide el tiempo en milisegundos desde el inicio del programa. Si la entrada de Reset es verdadera (true), el conteo de tiempo se pone a cero.

Entradas	Tipo	Predeterminado	Descripción
Reset	bool	false	Si es verdadera (true), el conteo de tiempo se pone a cero.
Salidas			
Tiempo	float		Tiempo en milisegundos desde el inicio del programa o desde que Reset cambió de verdadero (true) a falso (false).

5.6.3.1 Ejemplo



Temporizador y [Función de transferencia](#)^[63] genera un pulso de amplitud 10, un segundo después de que se ponga en marcha el programa.

5.6.4 Generador aleatorio



El generador aleatorio proporciona un número aleatorio dentro de un margen especificado.

Entradas	Tipo	Predeterminado	Descripción
Máximo	float	1	Límite superior del margen.
Mínimo	float	0	Límite inferior del margen.
Salidas			
Valor	float	0	Número aleatorio entre Mínimo y Máximo .

5.7 Filtro

Esta categoría contiene bloques de función para filtrar y suavizar señales.

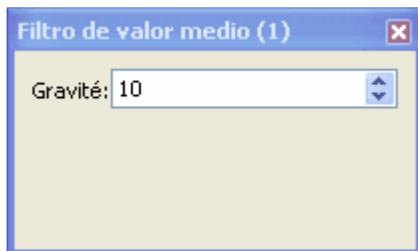
5.7.1 Filtro de valor medio



Calcula la media del valor de entrada para hasta 1000 pasos.

Entradas	Tipo	Predeterminado	Descripción
Entrada	float	0	Señal de entrada
Salidas			
Salida	float		Valor medio

5.7.1.1 Diálogo

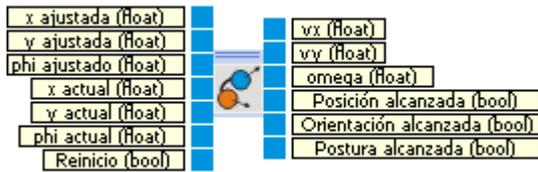


Profundidad es el número pasos de tiempo previos que se consideran para calcular el valor medio.

5.8 Navegación

Esta categoría comprende los bloques de función utilizados para la navegación.

5.8.1 Controlador de posición



El controlador de posición se utiliza para conducir a Robotino a una determinada posición.

El controlador de posición genera los valores de consigna de velocidad y velocidad angular para llevar al Robotino desde su posición actual a la posición fijada.

Entradas	Tipo	Unidad	Descripción
x ajustada	float	mm	coordenada x de la posición ajustada en el sistema global de coordenadas.
y ajustada	float	mm	coordenada y de la posición ajustada en el sistema global de coordenadas.
phi ajustado	float	Grados	orientación phi de la posición ajustada en el sistema global de coordenadas.
x actual	float	mm	coordenada x de la posición actual en el sistema global de coordenadas.
y actual	float	mm	coordenada y de la posición actual en el sistema global de coordenadas.
phi actual	float	Grados	orientación phi de la posición actual en el sistema global de coordenadas.
Reinicio	bool		Reiniciar movimiento
Salidas			
vx	float	mm/s	ajustar velocidad en dirección x en el sistema local de coordenadas del Robotino
vy	float	mm/s	ajustar velocidad en dirección y en el sistema local de coordenadas del Robotino
omega	float	gra/s	velocidad angular ajustada.
Posición alcanzada	bool		Es verdadero (true) si vx=vy=0, es decir, se ha alcanzado la posición ajustada.
Orientación alcanzada	bool		Es verdadero (true) si omega=0, es decir, se ha alcanzado la orientación ajustada.
Pose alcanzada	bool		Es verdadero (true) si se ha alcanzado la posición y la orientación (la pose).

Véase [Movimientos](#) ^[103]

5.8.1.1 Diálogo

Controlador de posición (1)

Controlador de posición

$v/\text{mm/s}$

300

0

0

100

d/mm

Gráficos Tabla

$\omega/\text{deg/s}$

50

0

0

20

d/deg

Gráficos Tabla

1: Accionamiento|Giro Holonómico

Rampa de velocidad: 1000ms

Rampa de velocidad angular: 1000ms

Movimiento activo: 1: Accionamiento|Giro Holonómico

El diálogo se compone en tres partes.

La parte superior refleja la asignación de la distancia a la posición de destino d (en mm) respecto a la velocidad de accionamiento v (en mm/s).

La parte media refleja la asignación de la distancia angular a la orientación de destino d (en 1°) respecto a la velocidad angular ω (en $1^\circ/s$). La distancia angular está en el margen $[0^\circ, 180^\circ]$. Las rotaciones en sentido horario y antihorario son tratadas de forma similar. La rotación se realizará en sentido horario o antihorario, de forma que la distancia a recorrer sea la mínima.

Con la lista desplegable.



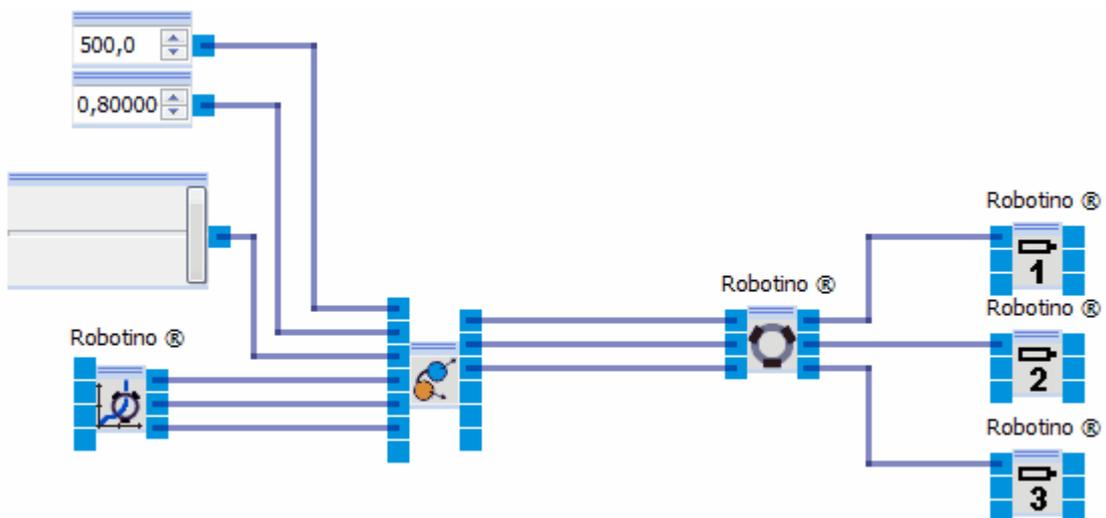
puede seleccionarse la clase de movimiento (véase [Movimientos](#)¹⁰³). El movimiento seleccionado se convierte en el movimiento activo.

1. al iniciarse el programa.
2. cuando la entrada "reiniciar" se pone en true.

La rampa de velocidad es el tiempo en milisegundos tras el cual se alcanza el 100% de la velocidad deseada. Esto evita un salto brusco de la velocidad al inicio del movimiento.

La rampa de velocidad angular es el tiempo en milisegundos tras el cual se alcanza el 100% de la velocidad angular deseada. Esto produce una amortiguación del movimiento cuando se inicia una nueva rotación.

5.8.1.2 Ejemplo



5.8.1.3 Movimientos

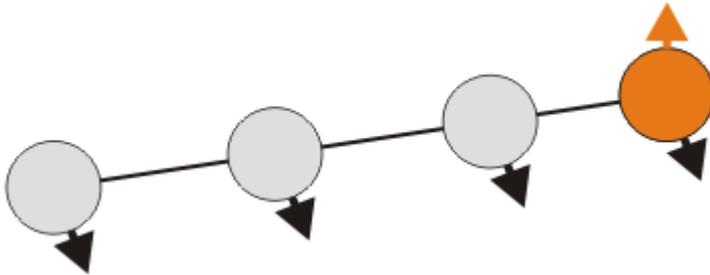
Son posibles cuatro clases de movimientos diferentes. Dos de ellos son aplicables para vehículos holónomos y no-holónomos cada uno de ellos. Como sea que Robotino tiene un accionamiento holónimo – los tres grados de libertad (x, y, giro) pueden ser alterados independientemente – Robotino puede realizar las cuatro clases de movimiento. Para los movimientos no-holónomos, la salida vy es 0.

El movimiento empieza cuando se ejecuta el programa o cuando la entrada "Reiniciar" es verdadera (true). De hecho, en el segundo caso el movimiento empieza cuando la entrada entrada "Reiniciar" se pone en false.

Movimiento 1 - conducir, girar - (holónimo)

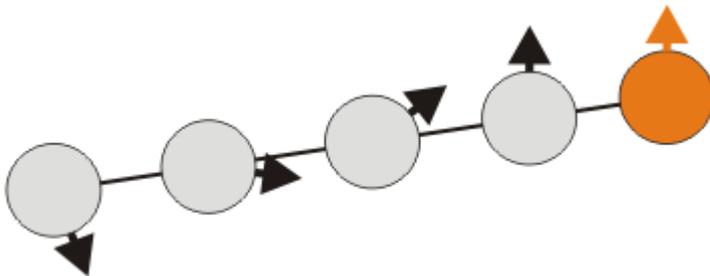
Paso 1: conducir a la posición de destino manteniendo la orientación de la posición inicial.

Paso 2: tras alcanzar la posición de destino, girar hasta alcanzar la orientación de destino.



Movimiento 2 - conducir y girar - (holónimo)

Paso 1: conducir y simultáneamente girar a la orientación de destino

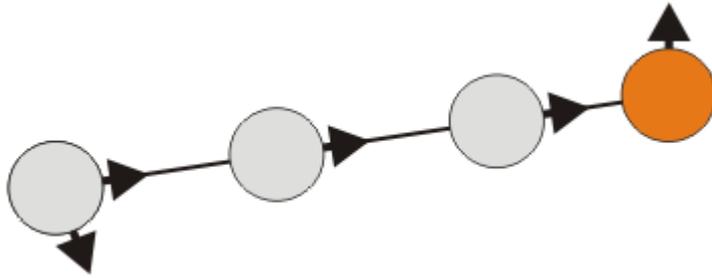


Movimiento 3 - girar, conducir, girar - (no-holónimo)

Paso 1: girar a la dirección de conducción

Paso 2: conducir hasta la posición de destino

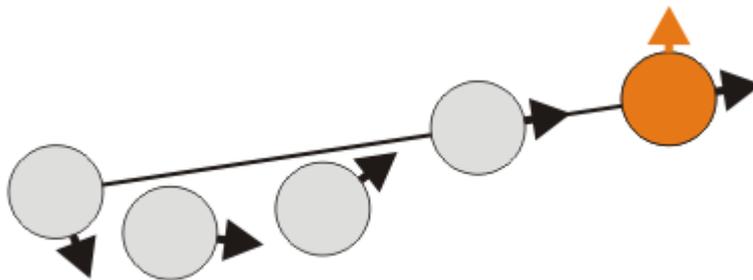
Paso 3: tras alcanzar la posición de destino, girar hasta alcanzar la orientación de destino.



Movimiento 4 - conducir y girar, girar - (no-holónimo)

Paso 1: Conducir y girar en el sentido de la conducción

Paso 2: tras alcanzar la posición de destino, girar hasta alcanzar la orientación de destino.



5.8.2 Pose constante



En el campo de entrada se especifica la pose. Las coordenadas se separan con espacios en blanco.

Entrada	Pose resultante
x y phi	(x, y, phi)
x y	(x, y, invalid)
x	Pose inválida
	Pose inválida

La orientación phi se especifica en grados.

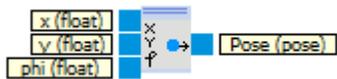
Ejemplo:

10.5 20 120

resultados en $x=10.5$ $y=20$ y orientación= 120° ;

Entradas	Tipo	Predefinido	Descripción
Salidas			
Pose	pose	Pose inválida	La constante del valor de la pose. El valor de la orientación se muestra en radianes en la salida.

5.8.3 Componer pose



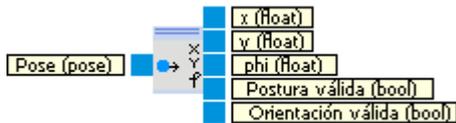
Entradas	Tipo	Unidad	Predefinido	Descripción
x	float		0	El componente x de la pose.
y	float		0	El componente y de la pose.
phi	float	Grados	0	La orientación de la pose en grados. La unidad puede pasarse a radianes en el Diálogo .
Salidas				
Pose	pose		(0, 0)	La pose (x, y, phi) compuesta a partir de sus valores individuales. El valor de la orientación se muestra en radianes en la salida.

5.8.3.1 Diálogo



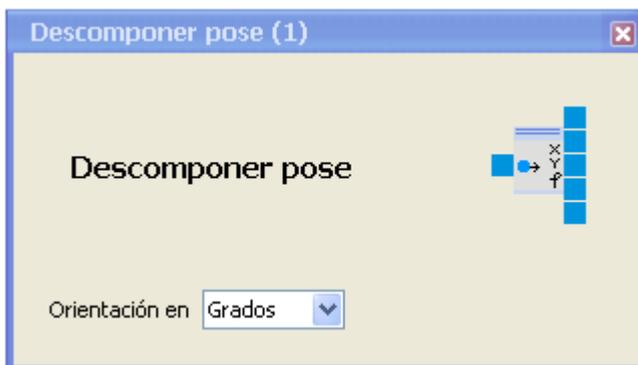
Pose válida	Especifica si la pose es válida. Las poses inválidas serán ignoradas en las rutas.
Orientación	Especifica si la orientación es válida y sus unidades (grados o radianes).

5.8.4 Descomponer pose



Entradas	Tipo	Unidad	Predefinido	Descripción
Pose	pose		(0, 0, 0)	Pose a descomponer
Salidas				
x	float		0	El componente x de la pose.
y	float		0	El componente y de la pose.
phi	float	Grados	0	La orientación de la pose en grados. La unidad puede pasarse a radianes en el Diálogo 105.
Pose válida	bool		false	Especifica si la pose es válida.
Orientación válida	bool		false	verdadero (true) si la orientación guardada en la pose es válida.

5.8.4.1 Diálogo



5.8.5 Componer ruta



Entradas	Tipo	Predeterminado	Descripción
Ruta 1	ruta	ruta vacía	La primera sub-ruta. Una Pose simple también será aceptada, ya que una pose puede convertirse en una ruta. Véase conversión de tipos ^[20] .
...			
Ruta 20	ruta	ruta vacía	La última sub-ruta. Una Pose simple también será aceptada, ya que una pose puede convertirse en una ruta. Véase conversión de tipos ^[20] .
Salidas			
Ruta	ruta	ruta vacía	La ruta compuesta a partir de las sub-rutas Ruta 1 + ... + Ruta 20

5.8.5.1 Diálogo



5.8.6 Descomponer ruta



Corta una sub-ruta de una ruta. Una ruta consta de una lista de poses.

Índice	Pose
1	p1
2	p2
...	

N	pN
---	----

Las entradas **Inicio** y **Longitud** especifican la pose inicial y la longitud de la ruta descompuesta. **Inicio** debe estar en $[1;N]$. Si **Inicio** < 1 , se utiliza el valor 1 internamente. Si **Inicio** $>$ la longitud de la ruta, el resultado será una ruta vacía. **Longitud** debe estar en $[0;N-\text{Inicio}+1]$. Si **Longitud** ≤ 0 el resultado será una ruta vacía.. Si **Longitud** $> N-\text{Inicio}+1$, el resultado será una subruta empezando en el índice **Inicio**.

Ejemplos:

Ruta = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Inicio = 3

Longitud = 5

Subruta = p3, p4, p5, p6, p7

Ruta = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Inicio = 0

Longitud = 1

Subruta = p1

Ruta = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Inicio = 11

Longitud = 1

Subruta = ruta vacía

Ruta = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Inicio = 1

Longitud = 0

Subruta = ruta vacía

Ruta = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Inicio = 2

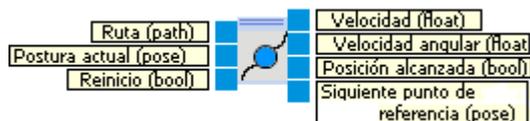
Longitud = 20

Subruta = p2, p3, p4, p5, p6, p7, p8, p9, p10

Entradas	Tipo	Predeterminado	Descripción
Ruta	ruta	ruta vacía	La ruta a descomponer.
Inicio	int	1	La pose en el índice Inicio de la ruta se convierte en la primera pose de la ruta descompuesta.

Longitud	int	1	La ruta descompuesta consiste en Longitud poses empezando en la pose del índice Inicio de la ruta a descomponer.
Salidas			
Subruta	ruta	ruta vacía	La ruta resultante empieza con la pose del índice Inicio y consta de Longitud poses.

5.8.7 Controlador de ruta

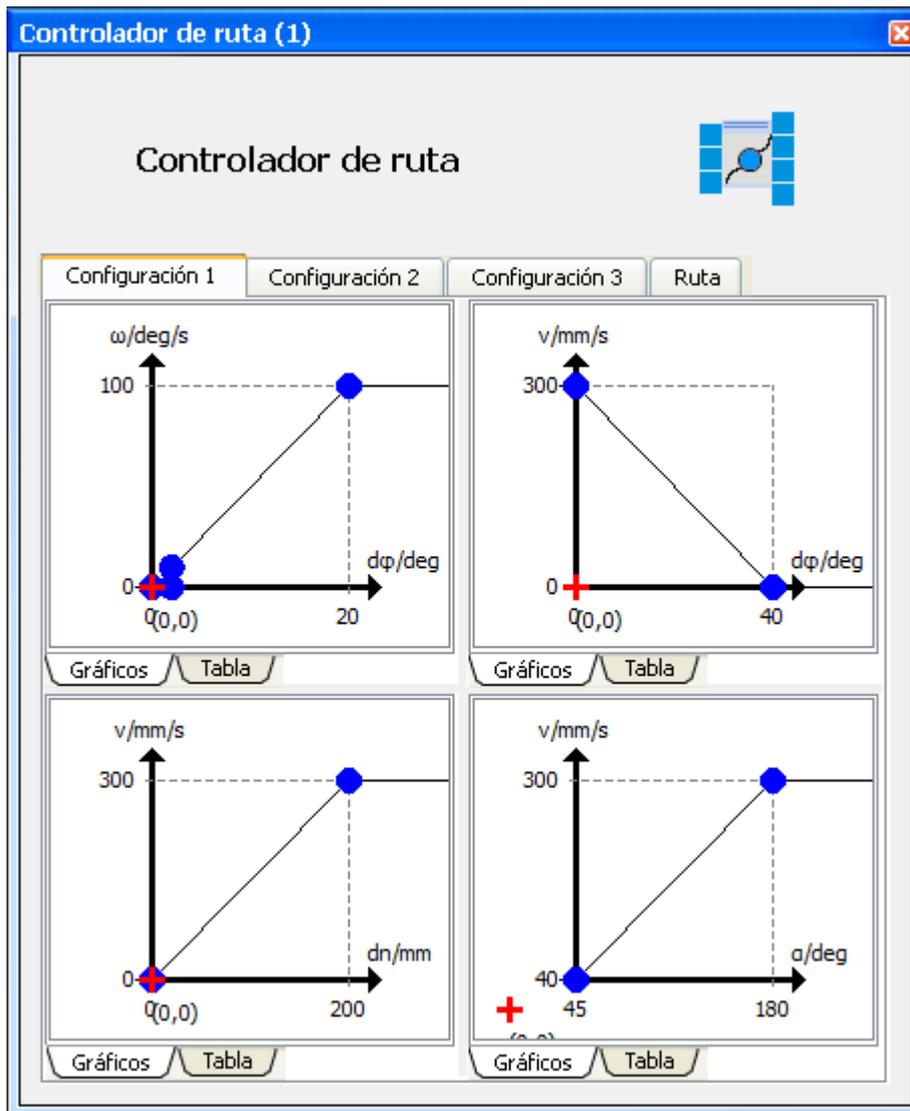


Con el controlador de ruta es posible conducir a lo largo de rutas.

A partir de la ruta y de la pose actual, se calculan la velocidad y la velocidad angular, de forma que Robotino sea conducido de forma rectilínea a lo largo de las poses individuales de la ruta.

Entradas	Tipo	Unidad	Predefinido	Descripción
Ruta	ruta		ruta vacía	La ruta a recorrer.
Pose actual	pose		(0, 0, 0)	La pose actual determinada por odometría o SLAM.
Reinicio	bool		false	Reinicia el movimiento.
Salidas				
Velocidad	float	mm/s		Velocidad de avance.
Velocidad angular	float	gra/s		Velocidad angular
Posición alcanzada	bool			Verdadero (true) si la ruta está vacía. De contrario, true cuando el punto virtual se halla localizado en el último segmento de la ruta y $v(d) = 0$.
Siguiente punto en el recorrido	pose			El siguiente punto objetivo en el recorrido.

5.8.7.1 Diálogo de configuración 1



Arriba a la izquierda

Correlación entre velocidad angular y error angular $d\phi$.

Arriba a la derecha

Correlación entre velocidad de avance y error angular $d\phi$.

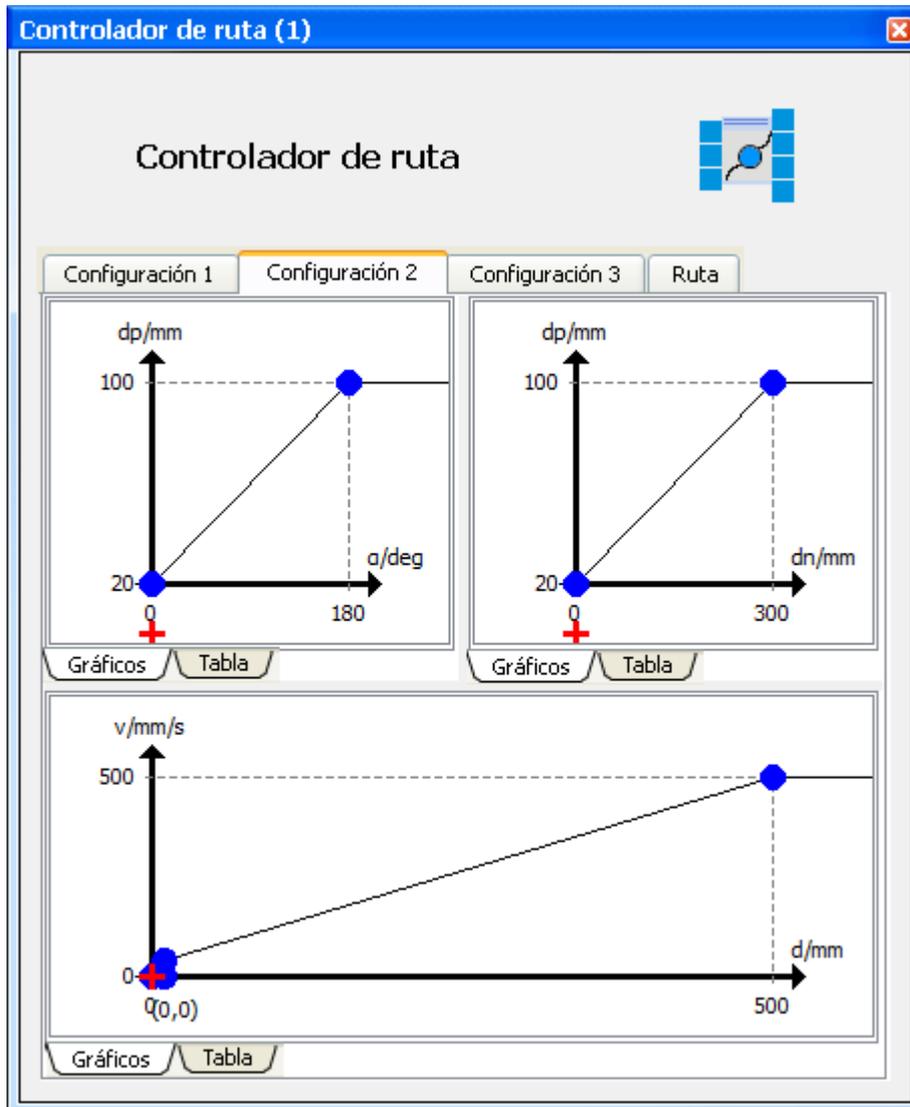
Abajo a la izquierda

Correlación entre velocidad de avance y distancia al siguiente punto del recorrido.

Abajo a la derecha

Correlación entre velocidad de avance y el ángulo al siguiente segmento de la ruta.

5.8.7.2 Diálogo de configuración 2



Arriba a la izquierda

Correlación entre la distancia del robot hasta el punto virtual en el recorrido y el ángulo al siguiente segmento de la ruta.

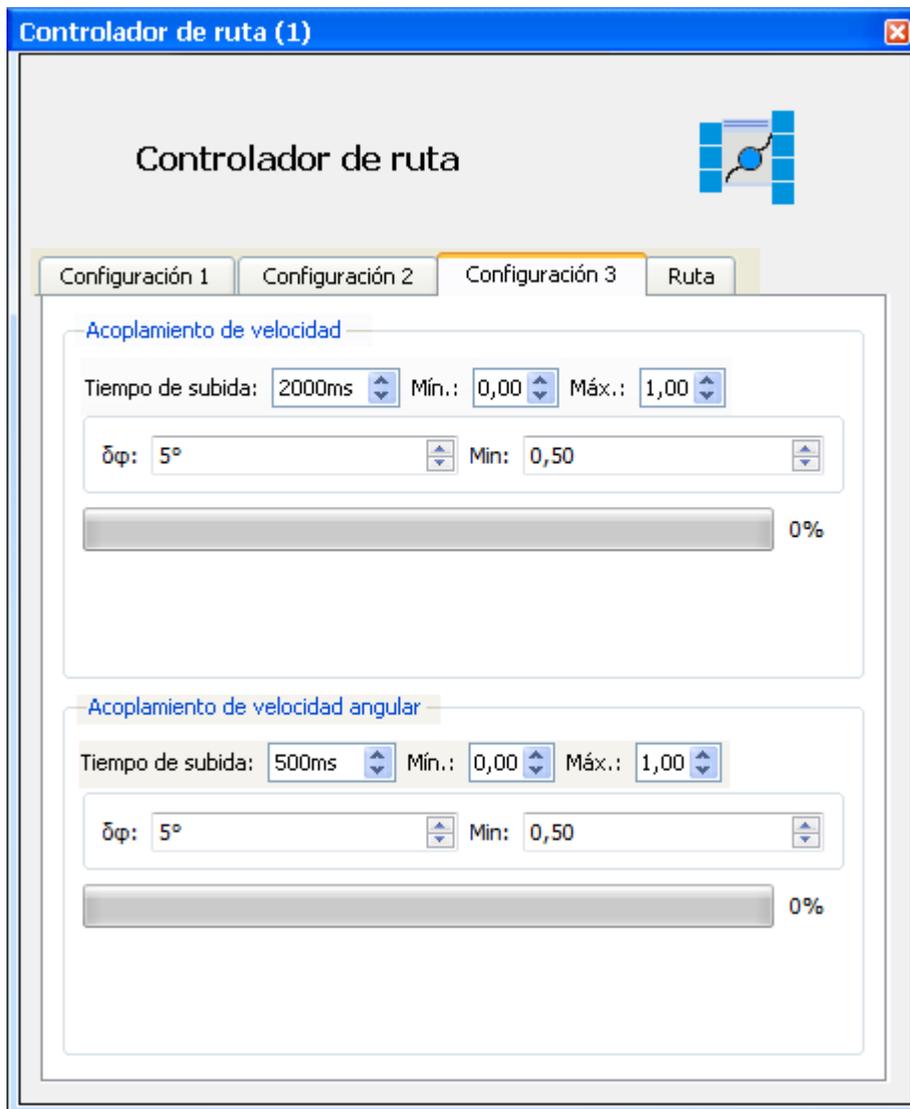
Arriba a la derecha

Correlación entre la distancia del robot hasta el punto virtual en el recorrido y la distancia al siguiente punto del recorrido.

Abajo

Correlación entre velocidad de avance y distancia al final de la ruta.

5.8.7.3 Diálogo de configuración 3



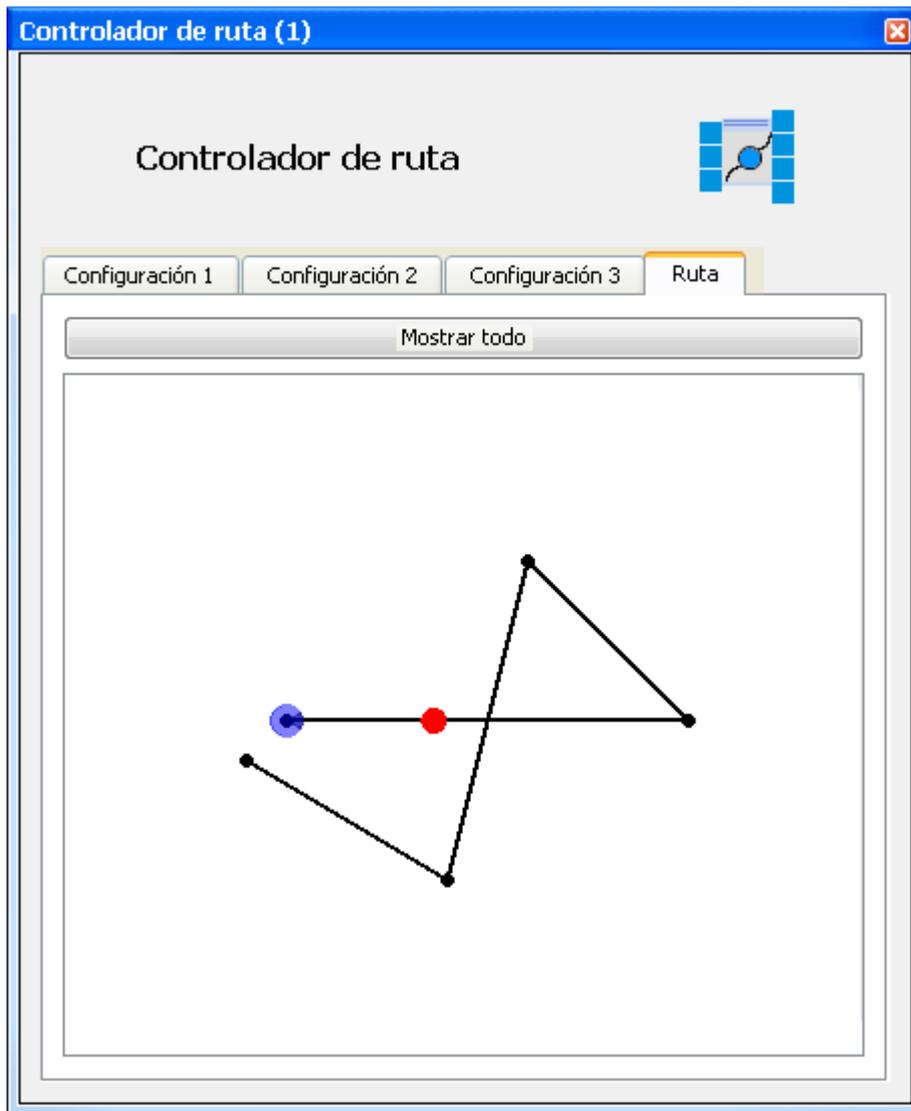
Arriba

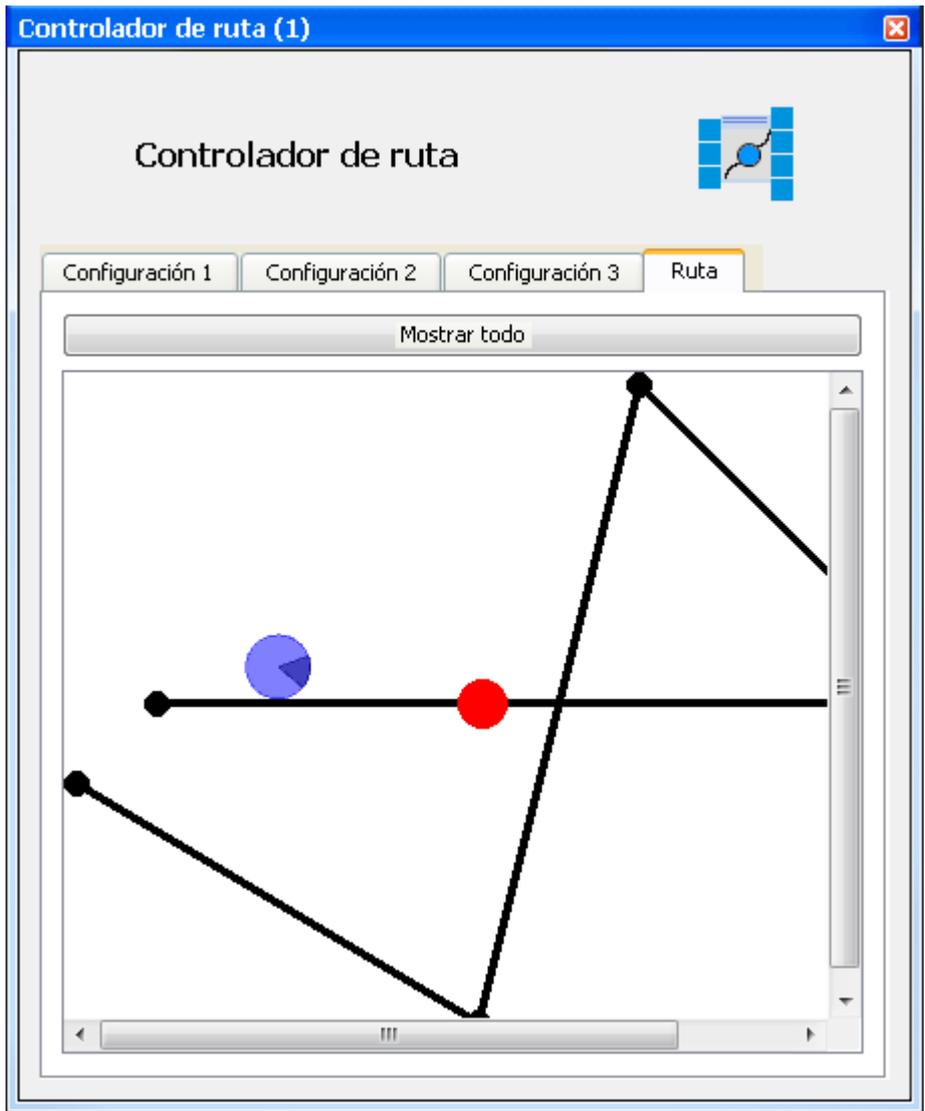
Adaptación del factor de acoplamiento entre la velocidad calculada debida a la configuración de los diálogos 1 y 2, y la velocidad real.

Abajo

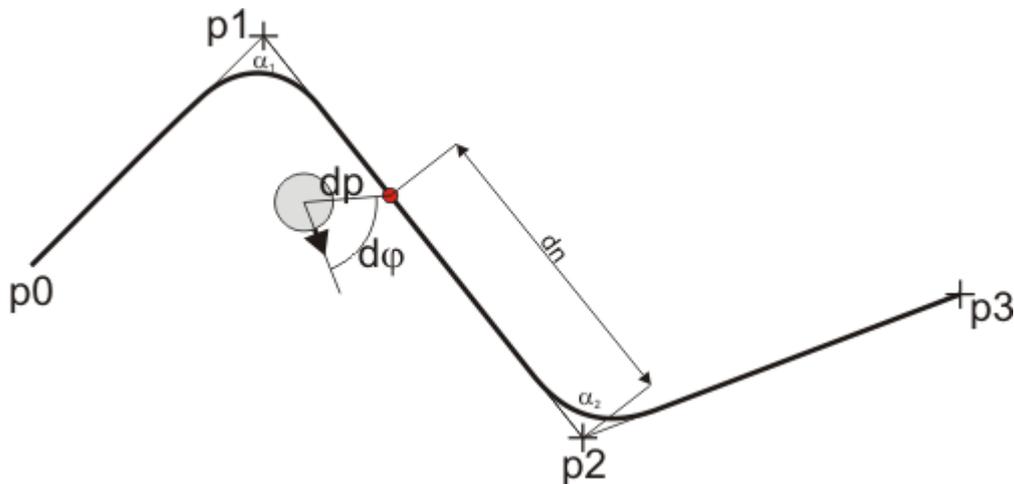
Adaptación del factor de acoplamiento entre la velocidad angular calculada debida a la configuración de los diálogos 1 y 2, y la velocidad angular real.

5.8.7.4 Visor de ruta





5.8.7.5 Estrategia



El bloque de función controlador de ruta crea una ruta que primero conecta los puntos a recorrer con una línea recta.

El robot es conducido con un punto de recorrido virtual (indicado con un punto rojo en la figura superior). Dada la posición actual del robot, el punto de recorrido virtual se colocará en la ruta de forma que la distancia entre el robot y el punto virtual del recorrido sea d_p (distancia al punto virtual). El punto virtual del recorrido sólo puede moverse a lo largo de la ruta hacia el final, es decir, si el robot se mueve alejándose del punto virtual del recorrido, permanece inalterable. Debido a la regulación en el punto virtual, la ruta es alisada (suavizada). Cuando mayor sea d_p , mayor será el alisamiento.

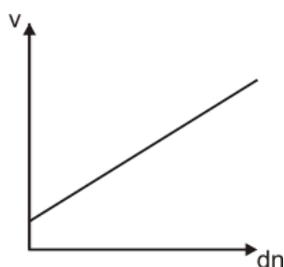
Parametrización de la velocidad angular

La velocidad angular $\omega(d\phi)$ se especifica por medio del diálogo del bloque de función, dependiendo del error angular $d\phi$. $d\phi$ es el ángulo entre la actual orientación del robot y la línea desde el centro del robot hasta el punto virtual del recorrido.

Parametrización de la velocidad

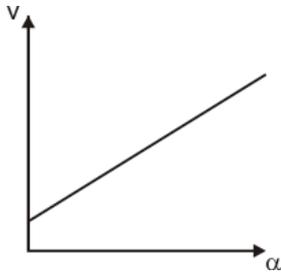
La velocidad también se especifica dependiendo de d_n y se denomina $v(d_n)$. Así que es posible ralentizar el movimiento si el robot no está correctamente orientado.

Para poder reducir la velocidad si la ruta tiene una curva, la velocidad también viene especificada como una función $v(d_n)$ de la distancia entre el punto virtual y el siguiente punto virtual del recorrido. Una típica curva de $v(d_n)$ es



Es decir, la velocidad disminuirá si el robot está cerca del punto del recorrido.

Pero queremos hacer andar más despacio al robot según el ángulo α_n . α_n es el ángulo entre el segmento actual y el siguiente en la ruta. Si $\alpha_n = 180^\circ$; (es decir, la ruta discurre de forma recta a lo largo del punto en el recorrido) la velocidad no va a reducirse. Si α_n se acerca a 0° ; (una curva muy pronunciada) el robot debe reducir su velocidad drásticamente. Por lo tanto, la función $v(\alpha_n)$ es necesaria. Una típica curva de $v(\alpha_n)$ se parece a esto



Es decir, cuanto más pequeña sea α_n más baja será la velocidad.

Estos tres perfiles de velocidad ($v(d)$, $v(d_n)$ y $v(\alpha)$) se utilizan para calcular la velocidad total $V(d, d_n, \alpha)$:

$$V_p(d, d_n, \alpha) = \min(v(d), \max(v(d_n), v(\alpha)))$$

Conducir al último punto del recorrido

Para reducir velocidad cuando se alcanza el final de la ruta, la velocidad, que depende de la distancia a conducir, se especifica y se denomina $v(d)$. Se supone que se ha alcanzado el objetivo cuando la velocidad, en función de la distancia remanente a conducir, es cero.

La velocidad sin alisamiento resulta:

$$V(d, d, d_n, \alpha) = \min(v(d), V_p(d, d_n, \alpha))$$

Alisamiento de la velocidad y la velocidad angular

Hay otros dos parámetros disponibles para alisar el movimiento.

El acoplamiento de velocidad es el tiempo en milisegundos que se necesita para el acoplamiento v_{CC} entre la velocidad calculada $V_p(d, d_n, \alpha)$ y la velocidad real para alcanzar el valor 1.

El acoplamiento de la velocidad angular es el tiempo en milisegundos que se necesita para el acoplamiento ω_{CC} entre la velocidad angular calculada $\omega(d)$ y la velocidad real ω para alcanzar el valor 1.

$$dv = v_{CC} * (V_p_t - V_p_{t-1})$$

$$\text{velocidad} = V_p_{t-1} + dv$$

$$d\omega = \omega_{CC} * (\omega(d)_t - \omega(d)_{t-1})$$

$$\text{velocidad} = \omega(d)_{t-1} + d\omega$$

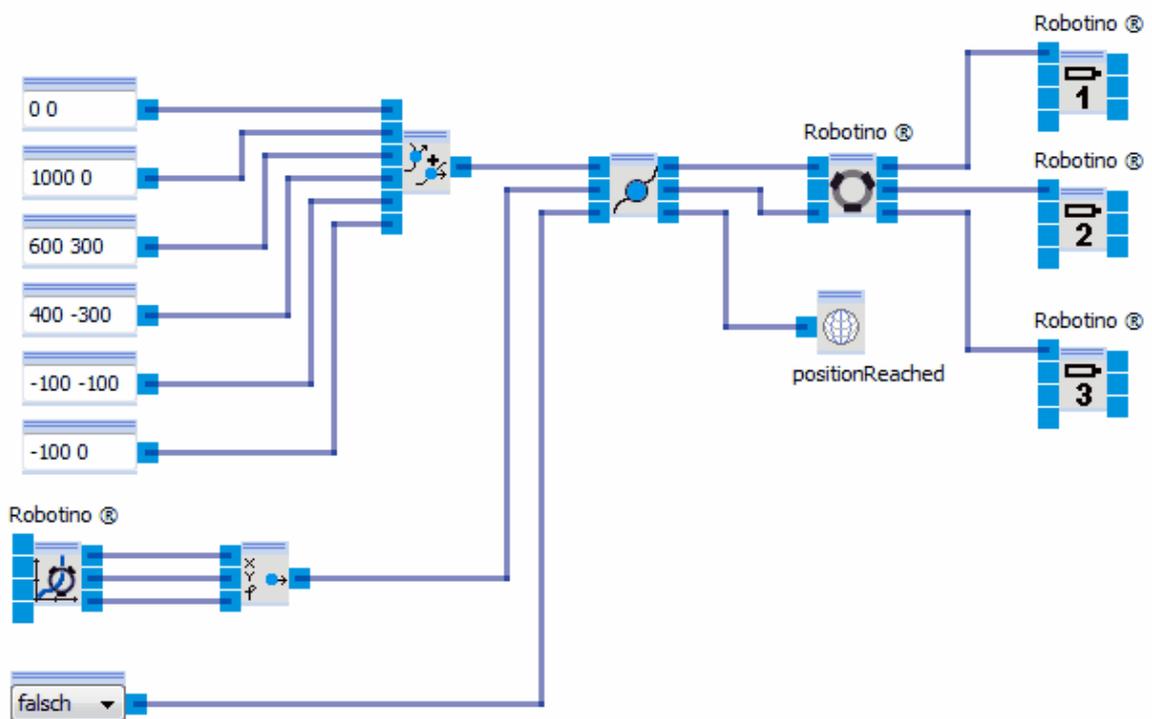
El subíndice t significa el valor en el tiempo t . $t-1$ significa el valor un paso de tiempo antes de t .

Al reiniciar vCC se inicializa a 0 y aumenta a 1 dentro del tiempo especificado por el acoplamiento de velocidad.

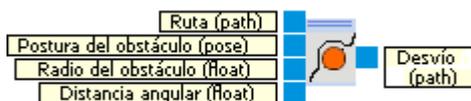
Al reiniciar ωCC se inicializa a 0 y aumenta a 1 dentro del tiempo especificado por el acoplamiento de velocidad angular.

Si el punto virtual salta a un nuevo segmento de la ruta, vCC y ωCC son restablecidos a 0.

5.8.7.6 Ejemplo



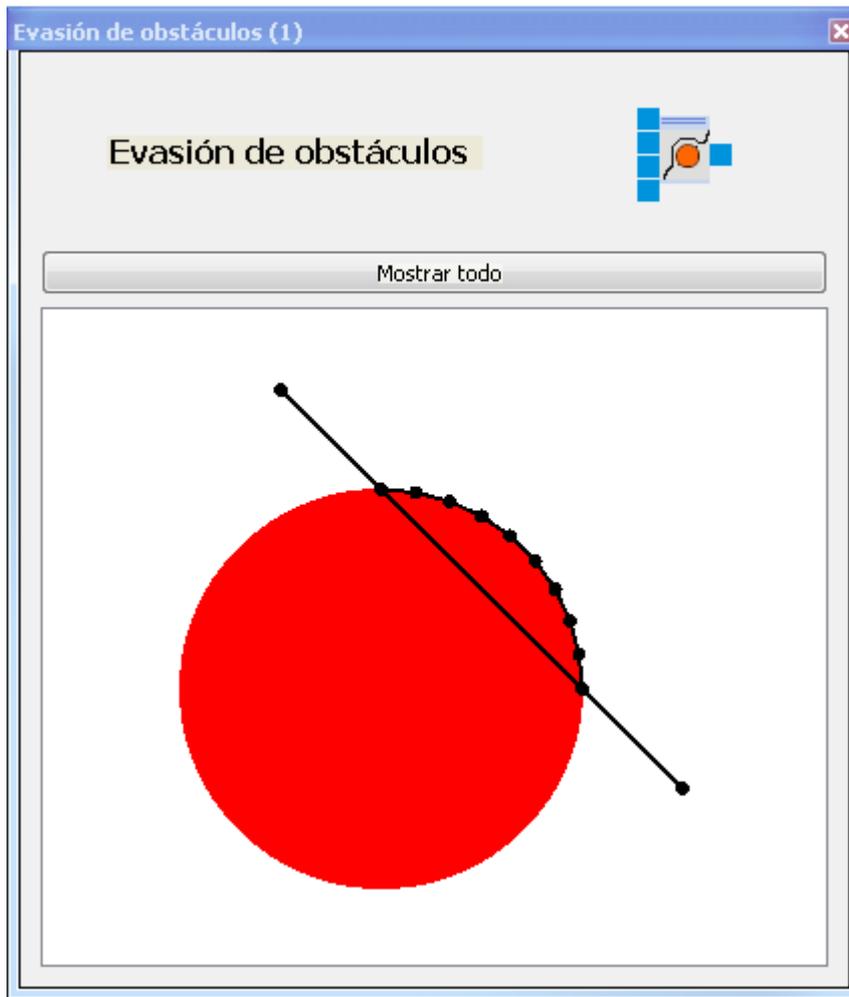
5.8.8 Evasión de obstáculos



El módulo de Evasión de obstáculos calcula un recorrido para hacer un rodeo que circunde un obstáculo circular.

Entradas	Tipo	Unidad	Predefinido	Descripción
Ruta	path		ruta vacía	La ruta a seguir.
Pose del obstáculo	pose		(0, 0, 0)	La posición del obstáculo circular.
Radio del obstáculo	float	mm	100	El radio del obstáculo circular.
Distancia angular	float	Grados	10	La distancia angular máxima entre dos puntos del rodeo alrededor del obstáculo.
Salidas				
Rodeo	ruta		ruta vacía	Rodeo alrededor del obstáculo.

5.8.8.1 Diálogo



El diálogo muestra la ruta original, el obstáculo y el rodeo.

5.9 Dispositivos de entrada

Esta categoría proporciona los bloques de función para realizar la interacción con el usuario.

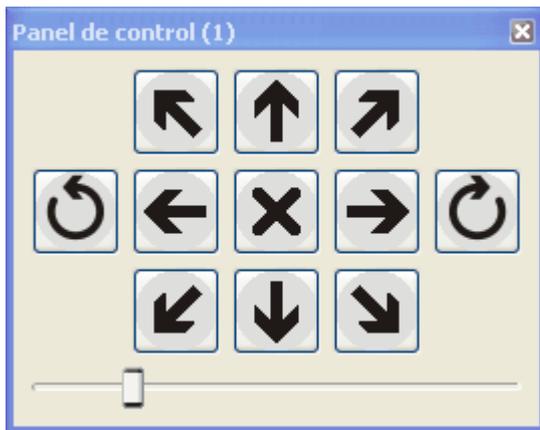
5.9.1 Panel de control



Consiste en un panel de control utilizable con el ratón.

Salidas	Tipo	Descripción
vx	float	Velocidad en dirección x
vy	float	Velocidad en dirección y
omega	float	Velocidad angular

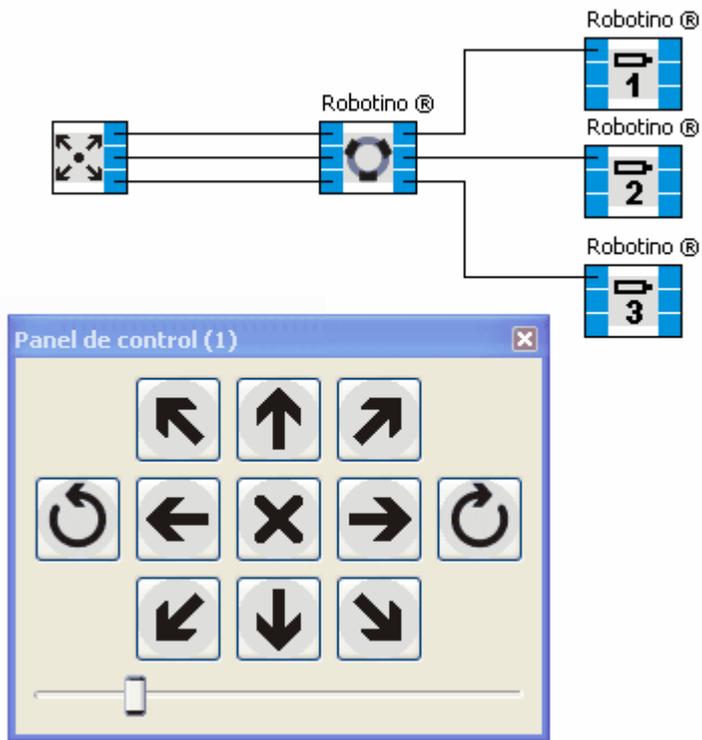
5.9.1.1 Diálogo



El panel de control puede utilizarse como sigue:

- Haciendo clic en uno de los botones el sistema robot se mueve en el sentido de la flecha.
- Haciendo clic en una de las dos flechas circulares se realiza una rotación en el sentido correspondiente.
- Haciendo clic en el botón central se detiene el movimiento.
- La velocidad de los movimientos se ajusta con la corredera.

5.9.1.2 Ejemplo

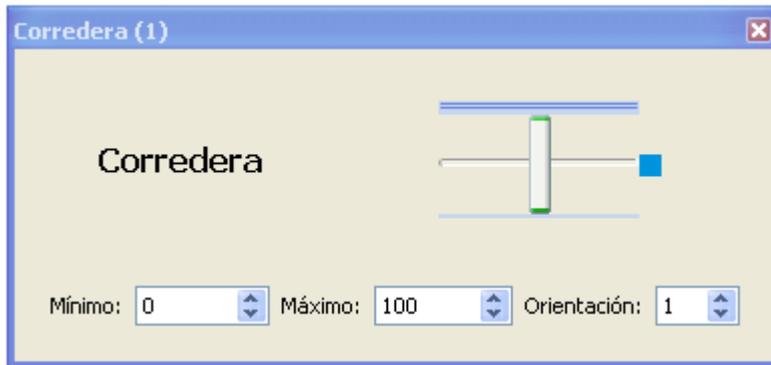


5.9.2 Corredera



La corredera crea cualquier número entero dentro del margen especificado.

5.9.2.1 Diálogo



En el diálogo puede ajustarse el margen de valores (Mínimo / Máximo) y la orientación de la corredera (1 = horizontal, 0 = vertical).

5.10 Intercambio de datos

Esta categoría contiene bloques de función para el intercambio de datos con Robotino® View o con aplicaciones externas.

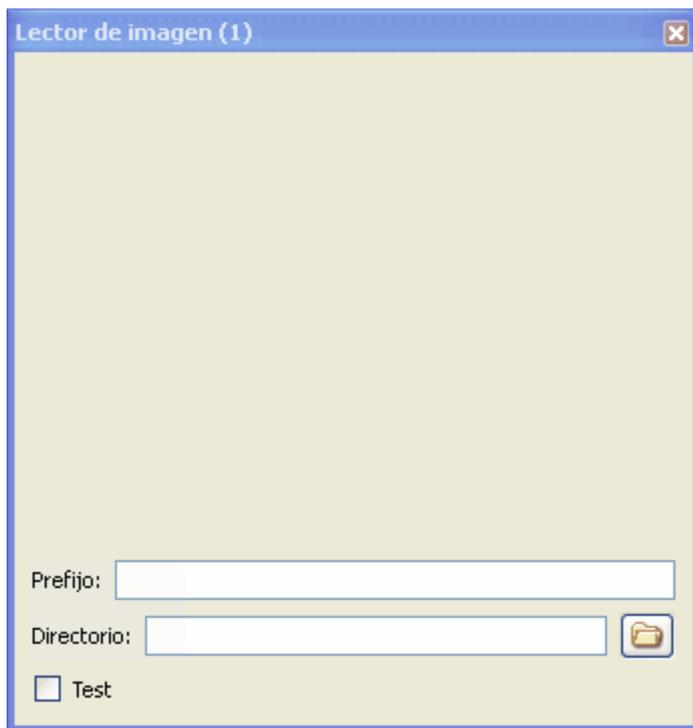
5.10.1 Lector de imagen



El lector de imagen lee imágenes JPEG de una secuencia de fotografías tomadas del sistema de archivos. Puede especificarse la ruta y el prefijo en el [diálogo](#)^[123].

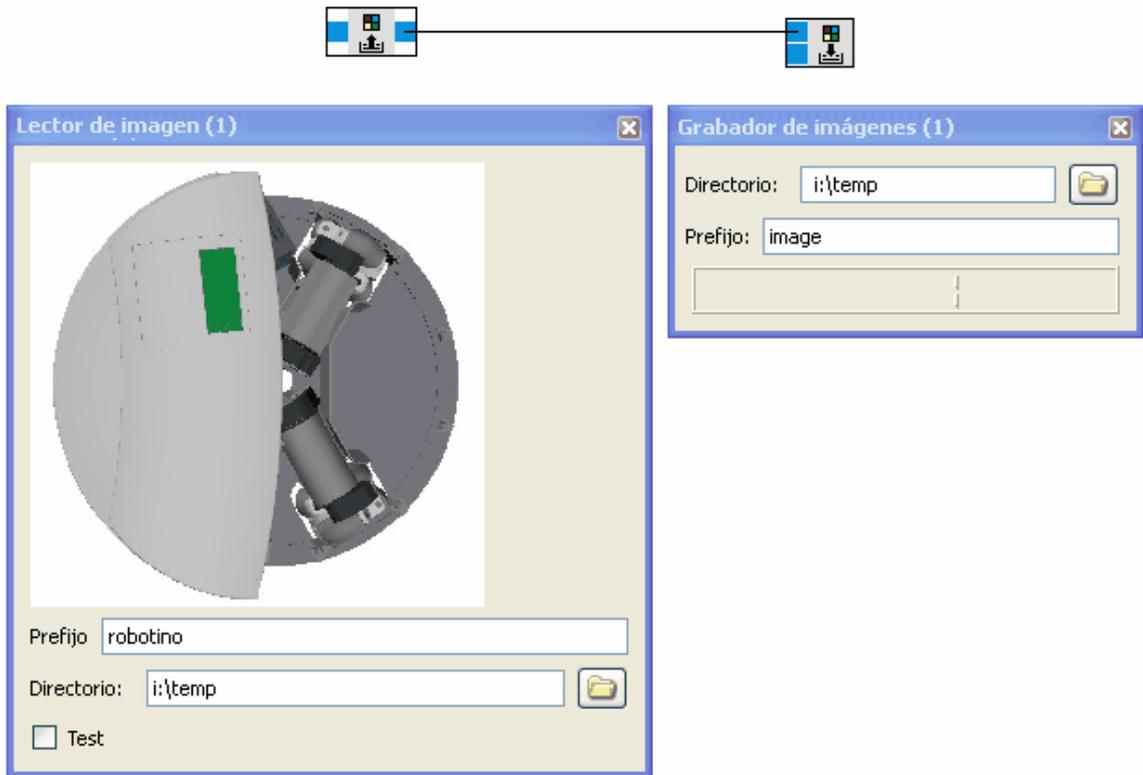
Entradas	Tipo	Predeterminado	Descripción
Número	int16	-1	Número de la imagen deseada. Si el Número = -1, el número de la imagen es incrementado automáticamente en 1 a cada paso, empezando por 0.
Salidas			
Salida	imagen		Imagen JPEG del archivo "<Directorio>/<Prefijo><Número>.jpg" o bien "<Directorio>/<Prefijo>_<Número>.jpg". Si el archivo no existe, se antepondrán ceros al número hasta una longitud total de 4, hasta que se encuentre el archivo de imagen.

5.10.1.1 Diálogo



En el diálogo es posible especificar el directorio y el prefijo de la secuencia de fotos a leer.

5.10.1.2 Ejemplo



5.10.2 Grabador de imágenes



El grabador de imágenes escribe una secuencia de imágenes JPEG en el sistema de archivos. Puede especificarse la ruta y el prefijo en el [diálogo](#)^[125]. El número de la imagen es incrementado en 1 a cada paso, empezando con 0.

Cada imagen (foto) es guardada como "<Directorio>/<Prefijo>_<Número>.jpg". El número tiene por lo menos 4 dígitos, incluyendo los ceros a la izquierda.

Entradas	Tipo	Predeterminado	Descripción
Entrada	imagen		Siguiente imagen de la secuencia.
Habilitar	bool	true	El grabador de imágenes está activo.

5.10.2.1 Diálogo



5.10.2.2 Ejemplo

Véase ejemplo [lector de imagen](#)^[124].

5.11 Variables

Las variables globales son de un tipo especial. Para todas las variables globales hay bloques de función para leer y escribir, disponibles en todos los subprogramas.. Estos bloques de función siempre muestran el nombre de la variable.

Es posible añadir, renombrar y quitar variables globales en el [Administrador de variables \(vista del programa principal\)](#)^[15]. Además, se les puede asignar un valor inicial.

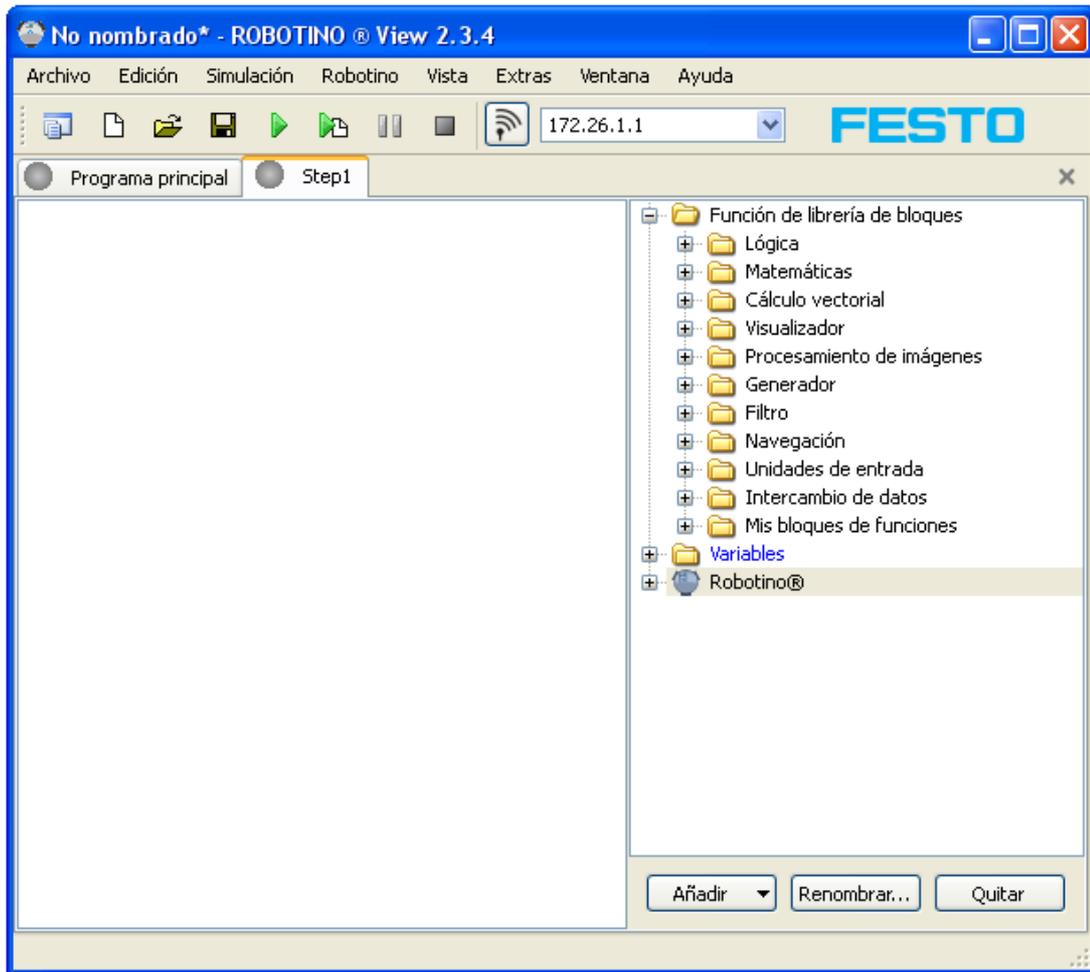
También es posible añadir, quitar o renombrar variables globales en la librería de bloques de función haciendo clic con el botón derecho en el dispositivo "Variables" y eligiendo "Añadir", o haciendo clic con el botón derecho en el lector o el escritor de la variable y seleccionando "Quitar" o "Renombrar".

6 Dispositivos

Los Dispositivos establecen la conexión entre Robotino View y el mundo exterior. El dispositivo "Robotino" puede comunicarse con un Robotino real o con uno de simulado. El dispositivo "Joystick" puede leer las posiciones de los ejes de un joystick conectado al ordenador.

6.1 Añadir y editar

Cuando se crea un nuevo proyecto, se añade automáticamente el dispositivo "Robotino". Para añadir más dispositivos, debe cambiar de la vista principal a la de un subprograma.



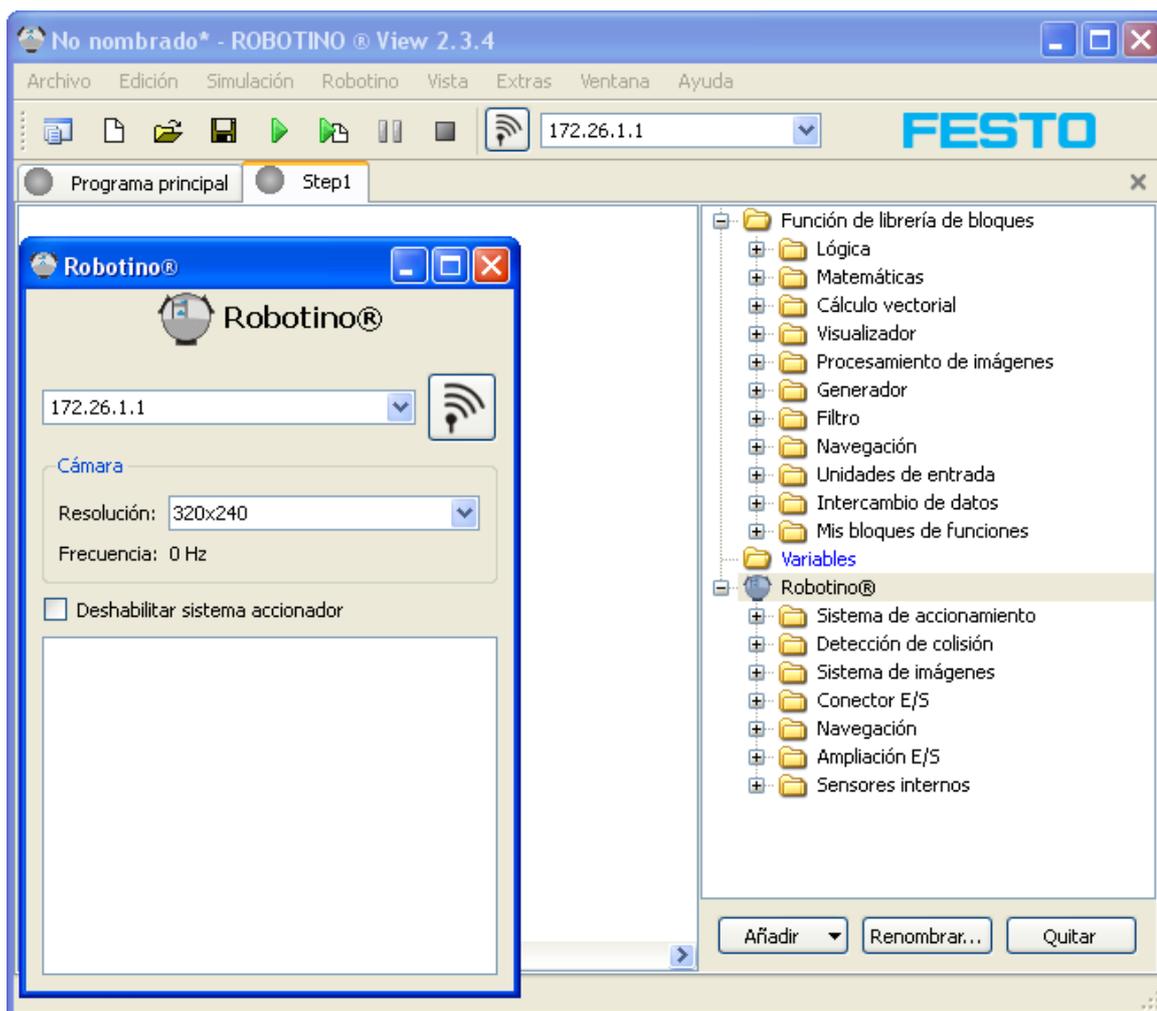
Bajo la librería de bloques de función, pueden añadirse dispositivos utilizando el botón "Añadir". El dispositivo elegido aparecerá debajo del dispositivo "Robotino" en la librería de bloques de función.

A los nuevos dispositivos se les asigna un nombre único. Este nombre puede cambiarse utilizando el botón "Renombrar...", si previamente se ha seleccionado el dispositivo en la librería de bloques de función.

El botón "Quitar" se utiliza para quitar dispositivos del proyecto actual. Esta función sólo está disponible si en el proyecto no se utiliza el bloque de función del dispositivo.

6.2 Mostrar diálogos

Los dispositivos configurables tienen una ventana de diálogo para su configuración. El diálogo se abre haciendo doble clic en el dispositivo una vez colocado en el subprograma.



6.3 Robotino

El dispositivo Robotino proporciona acceso a los sensores y actuadores del Robotino®.

6.3.1 Barra de herramientas



Puede hallar el campo de introducción de la dirección IP y el botón de conexión en la [barra de herramientas principal](#)^[11]. La entrada de la dirección IP y el botón de conexión se refieren al primer dispositivo Robotino de la lista de dispositivos en el Administrador de dispositivos. La función del campo de entrada de la dirección IP y del botón de conexión es idéntica a la del diálogo del dispositivo.

6.3.2 Diálogo



El diálogo del dispositivo Robotino aparecerá tras un doble clic en el dispositivo correspondiente.

1	Entrada de la dirección IP	La dirección predeterminada de Robotino es 172.26.1.1. Si desea conectarse con Robotino Sim (funcionando en el mismo ordenador que Robotino View) la dirección IP es 127.0.0.1:8080. 8080 es el número del puerto en el cual Robotino server escucha conexiones de entrada. Si se simula más de un Robotino, el número de puerto puede ser mayor.
2	Botón de conexión	Haciendo clic en este botón, se establece o se cierra la conexión con Robotino.
3	Resolución	La resolución de las imágenes requerida por la cámara del Robotino.
4	Frecuencia	Frecuencia de actualización de las imágenes
5	Deshabilitar sistema accionador	Si está marcado, los motores del Robotino están desactivados.
6	Ventana de mensajes	Muestra los diferentes mensajes en forma de texto.

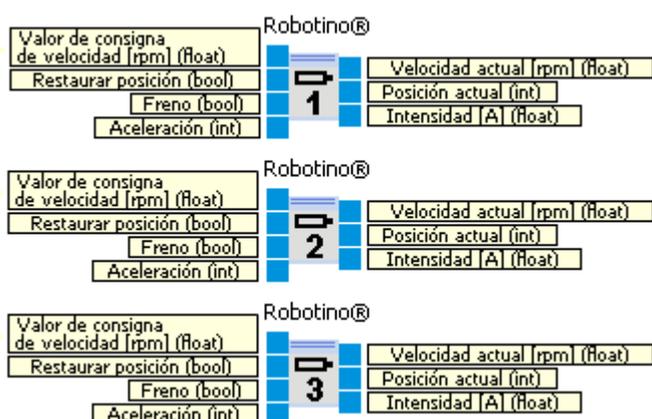
6.3.3 Boques de función

Los bloques de función permiten el uso del dispositivo Robotino en un subprograma.

6.3.3.1 Sistema de accionamiento

Esta carpeta contiene bloques de función para controlar el sistema de accionamiento del Robotino.

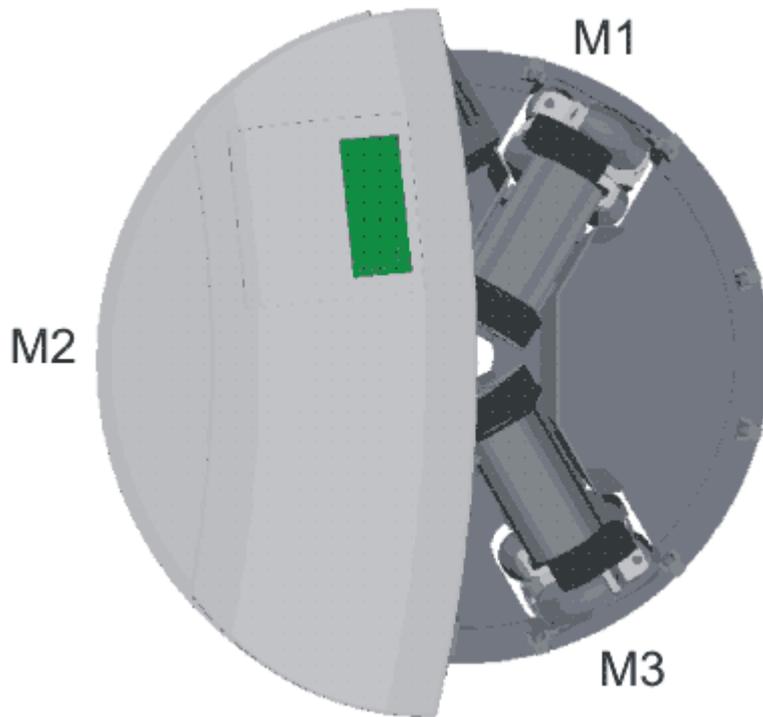
6.3.3.1.1 Motor



Acceso a los motores del Robotino. Se muestra el número del motor.

Entradas	Tipo	Unidad	Predefinido	Descripción
Valor de consigna de velocidad	float	rpm	0	El valor de consigna de velocidad del regulador del motor en vueltas por minuto. Tenga en cuenta que hay un reductor con una relación 16:1 entre el motor y la rueda del Robotino.
Restablecer posición	bool		false	Si es verdadera (true) el contador de pulsos del encoder del motor se restablece a 0.
Freno	bool		false	Si es verdadera (true) el motor está parado.
Aceleración	int		100	Acoplamiento del valor de consigna de velocidad en la entrada y el valor de consigna de velocidad realmente transmitido (véase Diálogo 130)
Salidas				
Velocidad actual	float	rpm		La velocidad actual del motor.
Posición actual	int			El número de pulsos contados desde que se ha aplicado tensión al Robotino o desde que "Restablecer posición" ha sido verdadero (true) y luego falso (false). Los pulsos son generados por el encoder del motor, que genera 2000 pulsos por vuelta.
Intensidad	float	A		La intensidad (corriente) medida en el puente H del motor.

Dispositivos



6.3.3.1.1 Diálogo



Parámetro	Descripción
-----------	-------------

Aceleración	Factor de Aceleración/Deceleración. Con el valor máximo, se dan directamente 100 valores de consigna de velocidad al controlador del motor. Con valores inferiores, las diferencias entre los valores de consigna de velocidad se alisan con el tiempo. Esto puede utilizarse para generar movimientos suaves en el Robotino.
kp	Término proporcional del regulador PID del motor.
ki	Término integral del regulador PID del motor.
kd	Término diferencial del regulador PID del motor.
Usar parámetros predeterminados	Utilizar los valores de kp, ki y kd implementados en el firmware del Robotino. También se usan estos valores predeterminados si se establece kp=ki=kd=255.
Restablecer al inicio	Inicializar la posición Actual a 0 al inicio del programa

El control de velocidad de cada motor se realiza con un regulador PID.

$$u(t) = K_p \left(e(t) + \frac{1}{T_N} \int_0^t e(t') dt' \right) + K_d \dot{e}(t)$$

Los parámetros son:

$$K_p$$

$$K_i = 1/T_n$$

$$K_d$$

A partir de los valores establecidos en el diálogo, los parámetros de regulación se calculan de la siguiente manera:

$$K_p = k_p / 2$$

$$K_i = k_i / 1024$$

$$K_d = k_d / 2$$

Los valores predeterminados son:

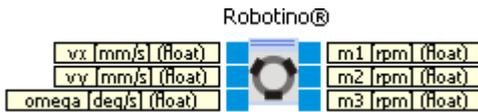
$$k_p = 25$$

$$k_i = 25$$

$$k_d = 25$$

Dispositivos

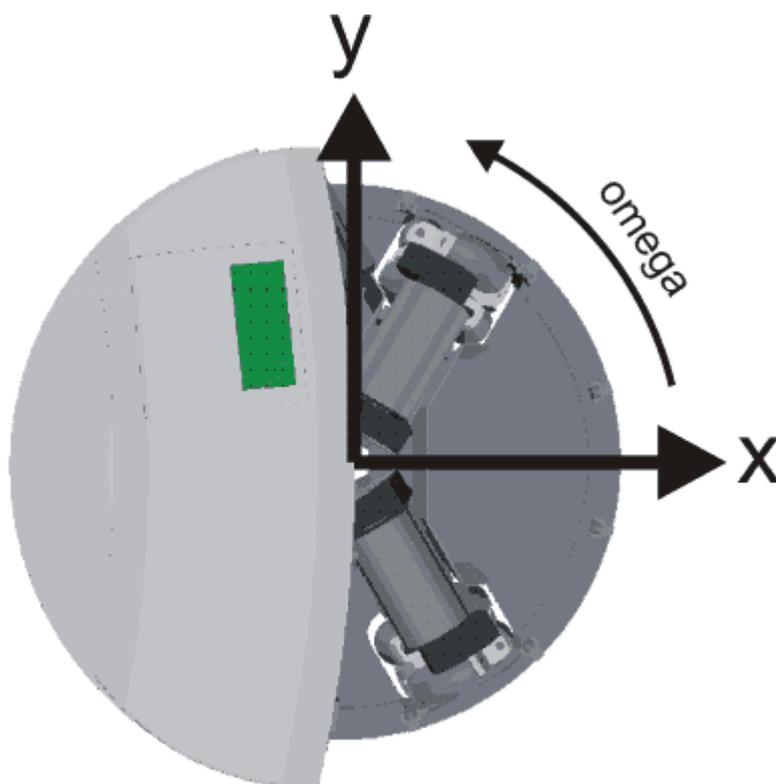
6.3.3.1.2 Omniaccionamiento



Calcula los valores de consigna de la velocidad de los motores 1, 2 y 3, según las velocidades de consigna vx, vy y omega.

Entradas	Tipo	Unidad	Predefinido	Descripción
vx	float	mm/s	0	Velocidad en dirección x en el sistema local de coordenadas del Robotino.
vy	float	mm/s	0	Velocidad en dirección y en el sistema local de coordenadas del Robotino.
omega	float	gra/s	0	Velocidad de rotación.
Salidas				
m1	float	rpm		Valor de consigna de velocidad motor 1
m2	float	rpm		Valor de consigna de velocidad motor 2
m3	float	rpm		Valor de consigna de velocidad motor 2

El bloque de función "Omniaccionamiento (inverso)", calcula vx, vy y omega a partir de las velocidades de rotación de los motores.



La figura muestra el sistema de coordenadas local de Robotino. Una velocidad de rotación positiva ω , genera una rotación en sentido antihorario mirando el Robotino desde arriba.

6.3.3.2 Detección de colisión

Aquí puede encontrar bloques de función relacionados con sensores para detectar obstáculos.

6.3.3.2.1 Paragolpes

Robotino®

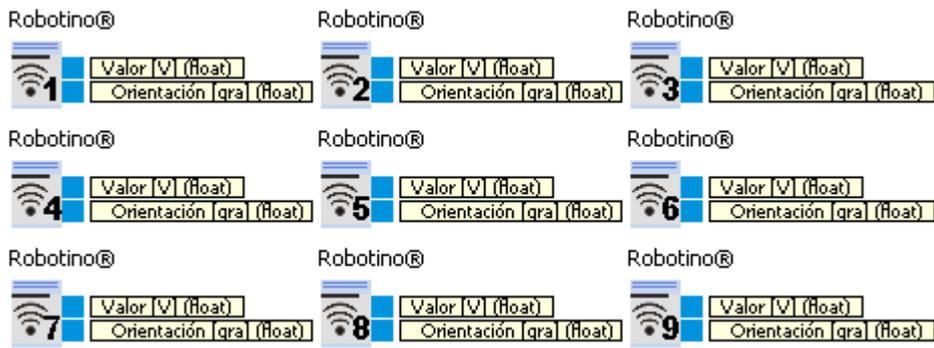


El paragolpes tiene integrado un sensor táctil. Al tocarlo, el sensor emite una señal de salida.

Entradas	Tipo	Predeterminado	Descripción
Salidas			
Valor	bool		Verdadero (true) si hay contacto.

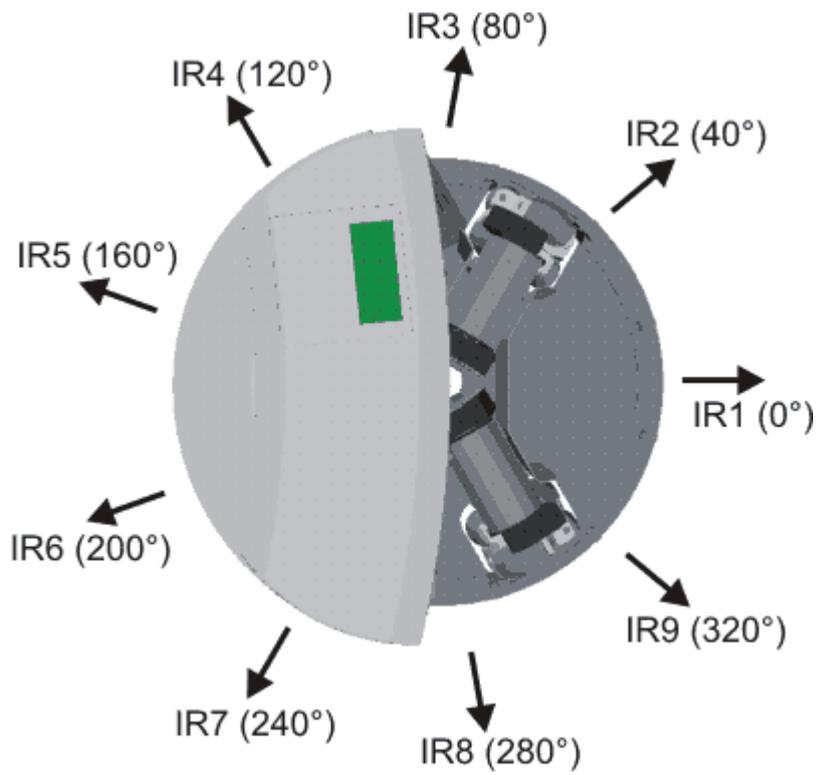
Dispositivos

6.3.3.2.2 Sensores de distancia



Permite la lectura de un sensor de distancia.

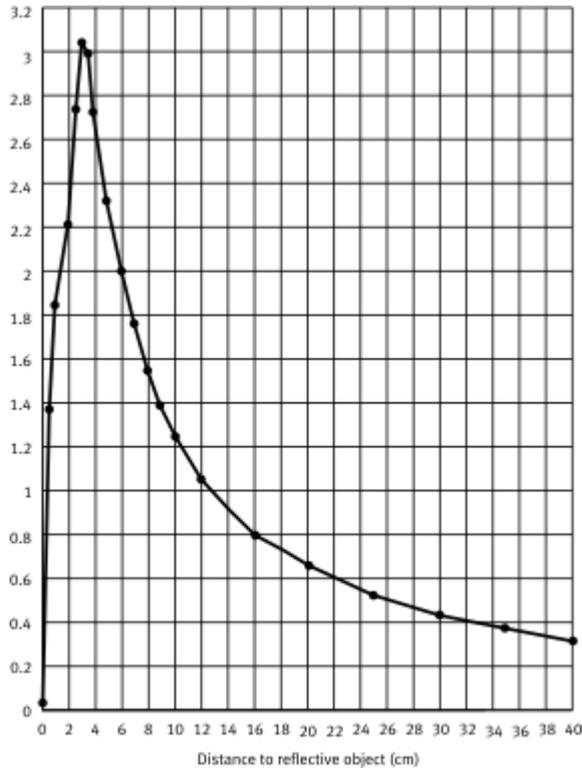
Entradas	Tipo	Unidad	Predefinido	Descripción
Salidas				
Valor	float	Volt		Lectura analógica de un sensor de distancia en V. El escalado y la conversión del valor debe ser realizado por el usuario.
Orientación	float	Grados		La orientación del sensor en el sistema de coordenadas local de Robotino (ver imagen siguiente). La orientación se calcula a partir del número del sensor como sigue: Orientación = 40° x (Número - 1)



6.3.3.2.2 Ejemplo

La hoja de datos del sensor de distancia (es un Sharp GP2D120) muestra la correlación entre la distancia a un objeto en cm y la señal de salida analógica del sensor en Voltios.

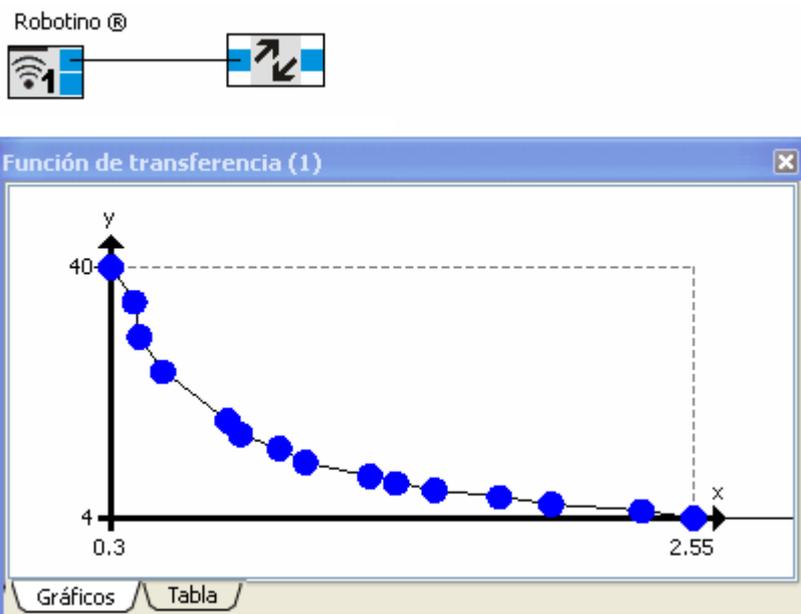
Dispositivos



Con esta correlación es fácil configurar una [función de transferencia](#)^[63] de forma que la tensión analógica se corresponda con la distancia al objeto en cm. Tenga en cuenta que esta correlación es la inversa de la mostrada arriba. Esto significa que debemos omitir distancias inferiores a 4 cm. Una distancia inferior a 4 cm no puede distinguirse de una distancia superior a 4 cm, puesto que la tensión analógica de la salida del sensor es la misma.

Además el convertidor analógico digital sólo mide voltajes de hasta 2,55 V.

La correlación entre la tensión analógica y la distancia también viene influida por el material del objeto a detectar. Por todo ello, es una buena práctica establecer la correlación por medios propios en cada caso.



Los valores de la [función de transferencia](#)^[63] se dan a continuación- Puede Copiar y Pegar estos valores en su propia [función de transferencia](#)^[63].

0.3	40
0.39	35
0.41	30
0.5	25
0.75	18
0.8	16
0.95	14
1.05	12
1.3	10
1.4	9
1.55	8
1.8	7
2	6
2.35	5
2.55	4

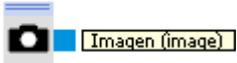
6.3.3.3 Sistema de imágenes

Esta carpeta contiene bloques de función para utilizar la cámara del Robotino.

Dispositivos

6.3.3.3.1 Cámara

Robotino®

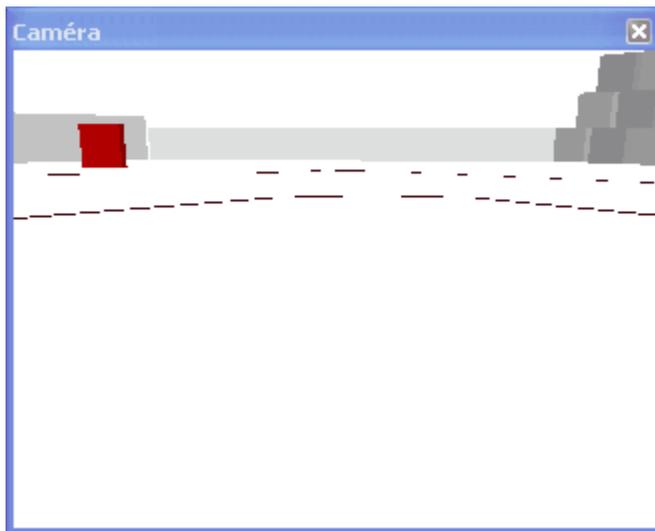


La imagen en directo de la cámara del Robotino.

Entradas	Tipo	Predeterminado	Descripción
Salidas			
Imagen	imagen		La imagen de la cámara del Robotino

Para ajustar la resolución de la imagen, puede utilizar el [diálogo del dispositivo](#)^[128].

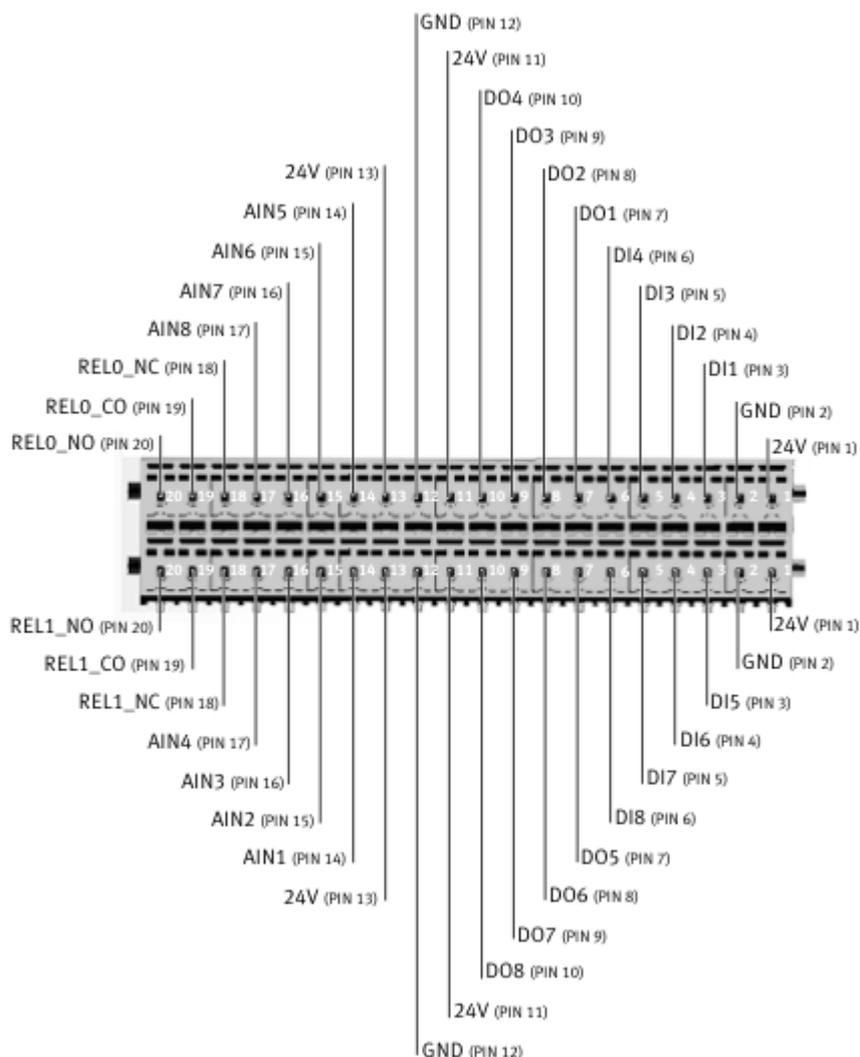
6.3.3.3.1 Diálogo



Muestra la última imagen. Para ajustar la resolución de la imagen, véase [diálogo del dispositivo](#)^[128].

6.3.3.4 Conector E/S

Aquí puede encontrar bloques de función para acceder al conector de E/S del Robotino.



6.3.3.4.1 Relé

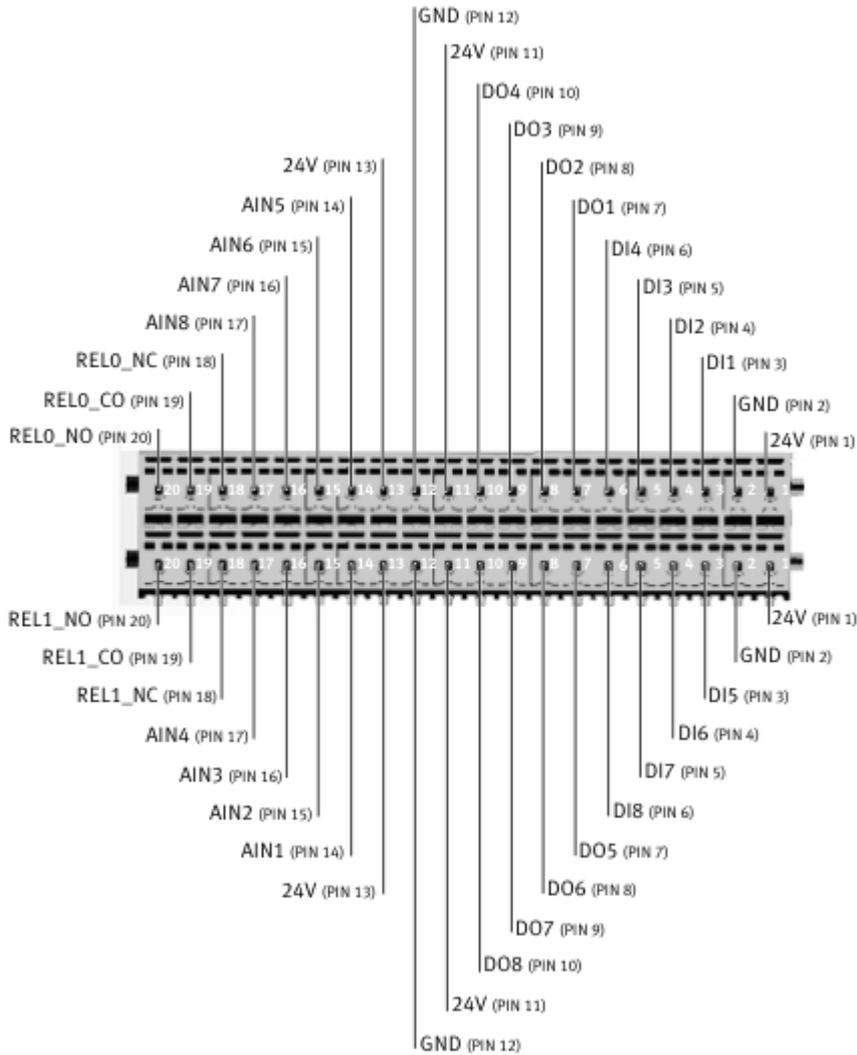


Relés conmutadores 1 y 2

Entradas	Tipo	Predeterminado	Descripción
Valor	bool	false	Si es falsa (false) el relé está desactivado.

Las conexiones del relé 1 son REL1_NO, REL1_CO y REL1_NC.

Las conexiones del relé 2 son REL2_NO, REL2_CO y REL2_NC.



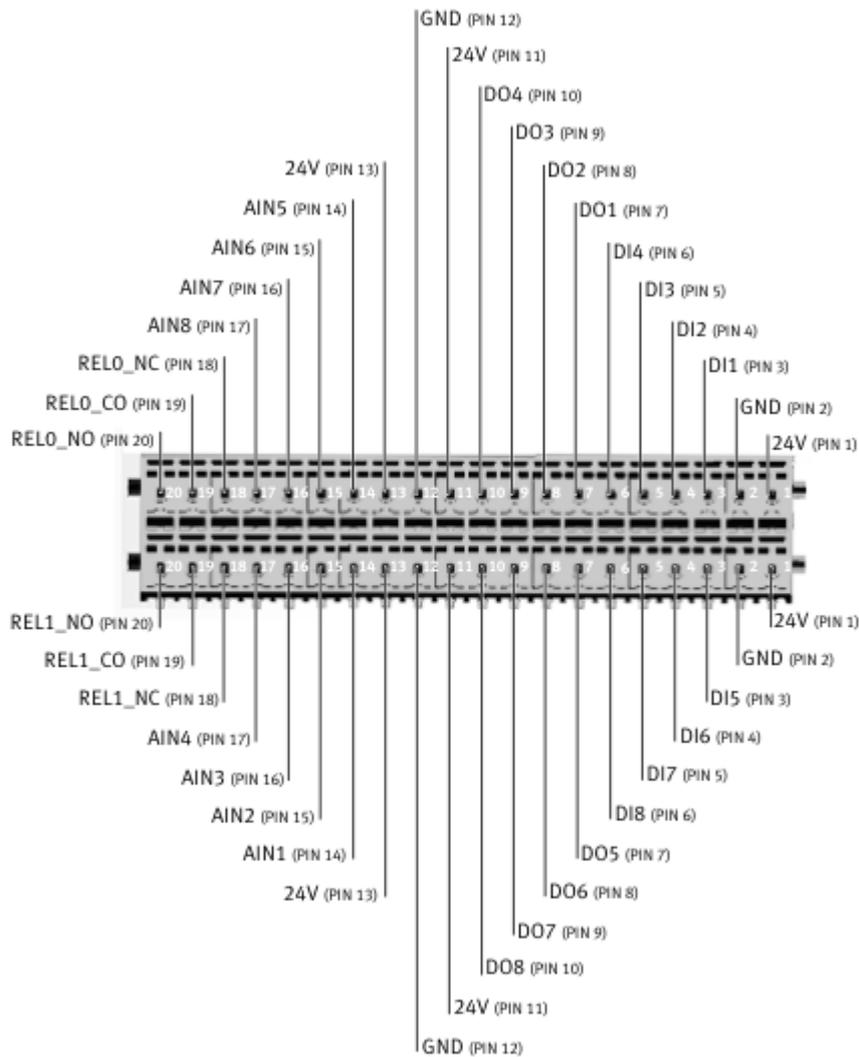
6.3.3.4.2 Salida digital



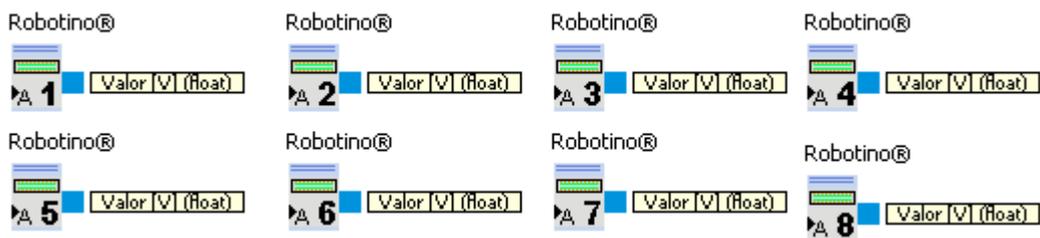
Activa una salida digital

Entradas	Tipo	Predeterminado	Descripción
Valor	bool	false	Si es verdadera (true) la salida en el conector de E/S del Robotino es +10V. De lo contrario, la salida es 0V

El conector para la salida digital x es DOx con x en [1;8].



6.3.3.4.3 Entrada analógica

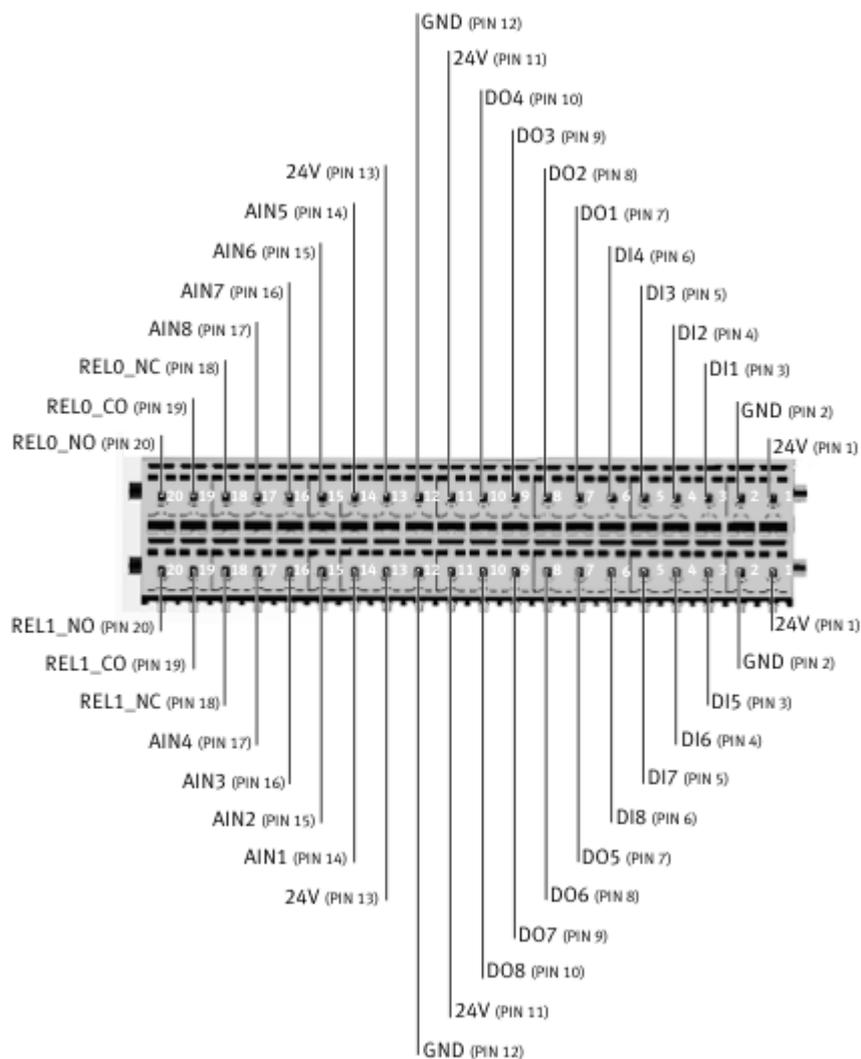


Lee el valor de una entrada analógica.

Salidas	Tipo	Unidad	Descripción
---------	------	--------	-------------

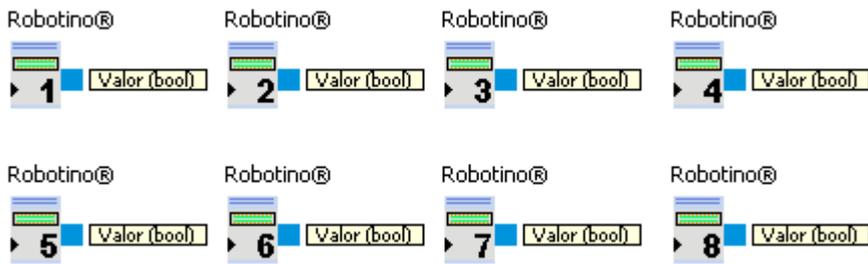
Valor	float	Volt	El voltaje medido. Margen [0;10].
-------	-------	------	-----------------------------------

El conector para la entrada analógica x es AINx con x en [1;8].



Dispositivos

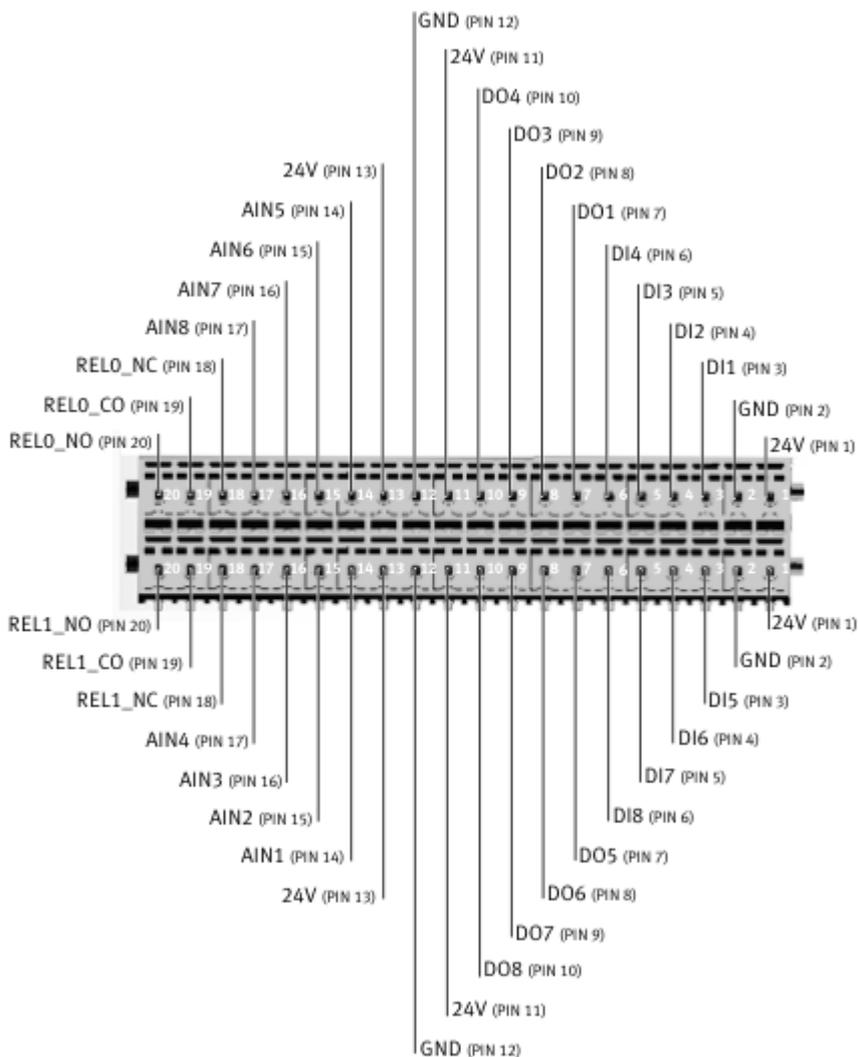
6.3.3.4.4 Entrada digital



Lee el valor de una entrada digital.

Salidas	Tipo	Descripción
Valor	bool	El valor del conector de E/S del Robotino. Los voltajes inferiores a 5,75 V son considerados como falsos (false). Los valores superiores a 8.6V son considerados como verdaderos (true). Si el voltaje en el conector se halla entre 5,75V y 8,6V el valor permanece inalterable.

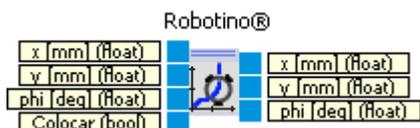
El conector para la entrada digital x es Dlx con x en [1;8].



6.3.3.5 Navegación

Esta carpeta contiene bloques de función para la localización del Robotino.

6.3.3.5.1 Odometría



Para esta función se necesita una tarjeta de memoria Compact-Flash de 1GB para Robotino (V1.7 o superior).

(No funciona con tarjetas de memoria de 256MB ni versiones <=1.6)

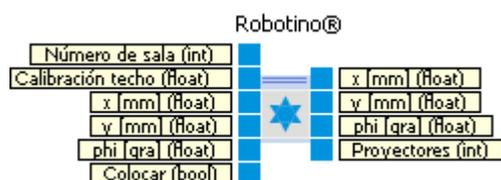
Dispositivos

La odometría es la utilización de datos procedentes del movimiento de los actuadores para estimar los cambios de posición en el tiempo. Véase <http://es.wikipedia.org/wiki/Odometria>.

La rotación de las ruedas se mide con la mayor resolución posible en el tiempo. A cada paso de tiempo, se calcula la distancia recorrida por el vehículo a partir de la velocidad de rotación de las ruedas. Estas distancias muy pequeñas obtenidas de los pasos, se integran a lo largo del tiempo. Con ello se obtiene la posición actual relativa a la posición de partida. Con este método se obtienen unos buenos resultados locales. En largas distancias o bajo condiciones adversas (deslizamiento de las ruedas debido a polvo en el suelo, deriva producida por un sentido preferente del piso), este método produce errores muy apreciables. Teniendo esto en cuenta, la odometría siempre se combina con otros métodos para compensar los errores citados.

Entradas	Tipo	Unidad	Predefinido	Descripción
x	float	mm	0	La nueva posición x La odometría se reinicia a la nueva posición si "Set" es verdadero (true).
y	float	mm	0	La nueva posición y La odometría se reinicia a la nueva posición si "Set" es verdadero (true).
phi	float	Grados	0	La nueva orientación La odometría se reinicia a la nueva posición si "Set" es verdadero (true).
Set	bool		false	Si es verdadera (true), la odometría se establece a los valores de las entradas x, y y phi. Para reiniciar la odometría a (0,0,0) sólo es necesario poner esta entrada en verdadero (true) durante un paso. No es necesario que estén conectadas las demás entradas, puesto que los valores predeterminados son 0.
Salidas				
x	float	mm		La posición x actual de la odometría en las coordenadas globales.
y	float	mm		La posición y actual de la odometría en las coordenadas globales.
phi	float	Grados		La orientación actual de la odometría en las coordenadas globales.

6.3.3.5.2 North Star



North Star® es un sensor que determina la posición absoluta del Robotino® con la ayuda de proyectores.

Entradas	Tipo	Unidad	Predefinido	Descripción
Número de sala	int		1	El número de la sala en la que se halla actualmente Robotino. Las salas se numeran empezando por 1.
Calibración techo	float	mm	1	Distancia entre el detector y el techo. Si la distancia al techo es de 3 m, la distancia entre el detector y el techo es de aproximadamente 2800 mm.
x	float	mm	0	posición x del origen, establecido por la entrada "Set".
y	float	mm	0	posición y del origen, establecido por la entrada "Set".
phi	float	Grados	0	Orientación del origen, establecido por la entrada "Set".
Set	bool		false	Si es verdadera (true), se utiliza la pose actual (x,y,phi) como origen.
Salidas				
x	float	mm		La posición x actual en las coordenadas globales.
y	float	mm		La posición y actual en las coordenadas globales.
phi	float	Grados		La orientación actual en las coordenadas globales.
Proyectores	int			Número de proyectores visibles.

Room ID	Projector type	Switch-Setting	Frequency [Hz]
0	ProjectorKit NS1	Spot 1 SW2 = 0 Spot 2 SW2 = 8	1040 2350
1	ProjectorKit NS1	Spot 1 SW2 = 1 Spot 2 SW2 = 9	1150 2450
2	ProjectorKit NS1	Spot 1 SW2 = 2 Spot 2 SW2 = A	1250 2560
3	Projector NS2-50Hz	1	Spot A 3025 Spot B 3925
4	Projector NS2-50Hz	2	Spot A 3125 Spot B 4025
5	Projector NS2-50Hz	3	Spot A 3225 Spot B 4125



ProjectorKit NS1



Projector NS2

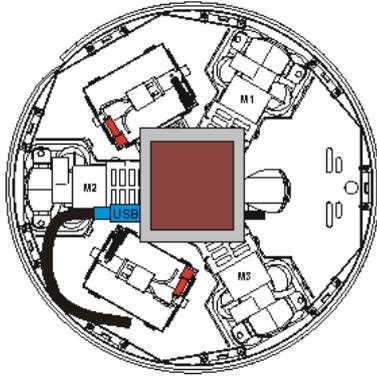
El detector NorthStar puede fijarse al Robotino de diferentes formas. Según la configuración, el archivo: /etc/robotino/robotino.xml del robotino debe ser adaptado con un editor de texto. El valor de la orientación debe establecerse según la figura inferior.

```
<NorthStar>
  <!--The orientation of the northstar sensor. See www.openrobotino.org-->
```

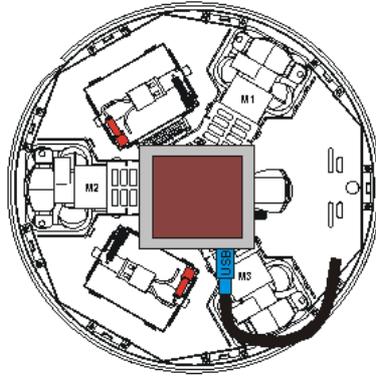
Dispositivos

```
<Orientation value="1" />  
</NorthStar>
```

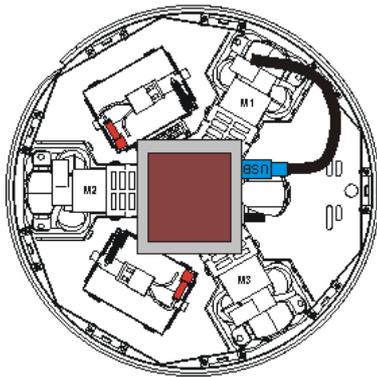
0



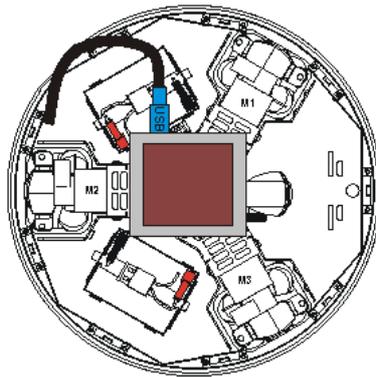
1



2



3



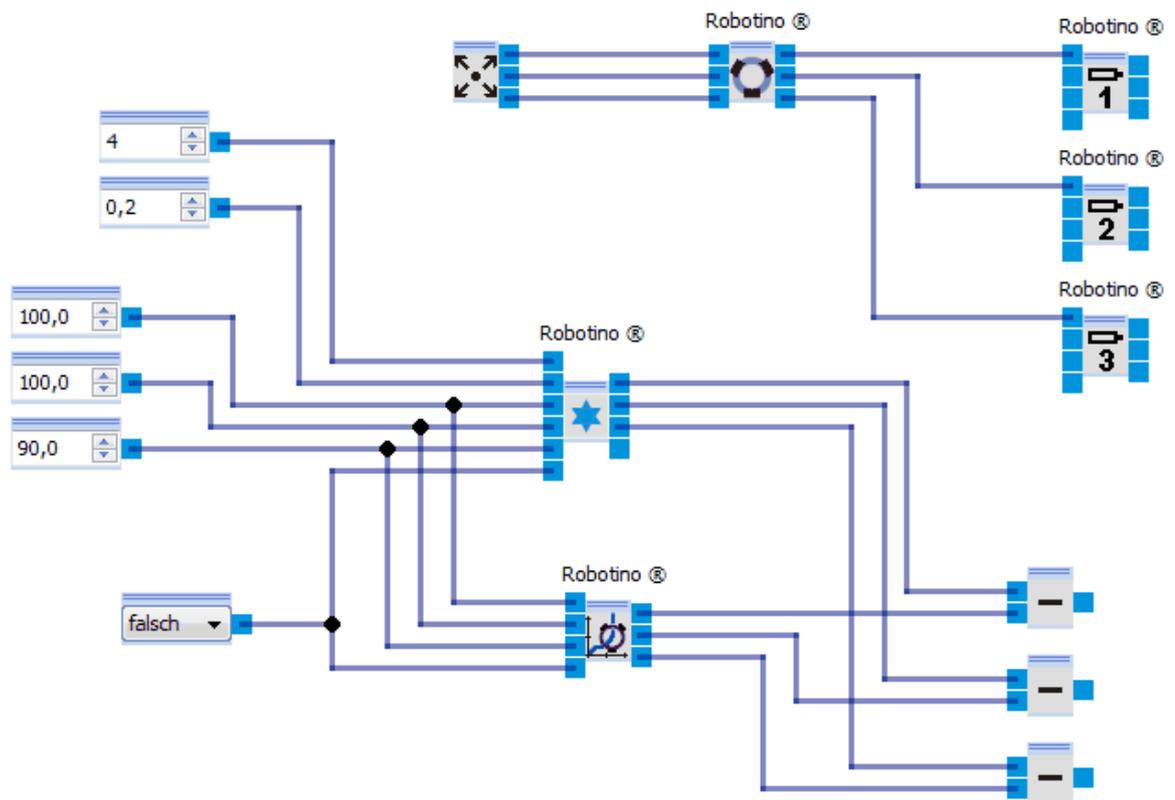
6.3.3.5.2 Diálogo



Punto A	Intensidad del primer punto de luz emitido por el proyector.
Punto B	Intensidad del segundo punto de luz emitido por el proyector.
Sala actual	El número de sala detectado por el sensor NorthStar.
Número de proyectores	Número de proyectores visibles.
Número de secuencia	El número de secuencia es incrementado en una unidad cada vez que el sensor NorthStar proporciona nuevas lecturas.

Dispositivos

6.3.3.5.2 Ejemplo



El panel de control se utiliza para mover el Robotino.

NorthStar detectará el proyector que pertenece a la sala número 4. El nuevo proyector NorthStar debe establecerse para la sala 1.



A través de la constante booleana (true/false) el sistema de coordenadas de NorthStar y la odometría pueden transformarse de forma que "pose NorthStar actual" == "pose odometría actual" == (100,100,90).

Restando los componentes de NorthStar y odometría, puede verse el error que se produce entre odometría y NorthStar.

6.3.3.6 Ampliación E/S

Esta carpeta contiene bloques de función para otros interfaces de hardware del Robotino.

6.3.3.6.1 Entrada del transmisor giratorio



Este bloque de función lee valores del transmisor giratorio (encoder) del Robotino.

La entrada del transmisor giratorio interpreta las señales de un encoder digital (Canales A-B, código de Gray).

Se tienen en cuenta los flancos ascendente y descendente. Esto produce una resolución efectiva de 4 cuadrantes.

Ejemplo: El encoder de los motores de accionamiento del Robotino tiene una resolución de 500 pulsos. De forma efectiva, se cuentan 2000 pulsos.

Entradas	Tipo	Unidad	Predefinido	Descripción
Restablecer posición	bool		0	Si es verdadera (true) la posición actual se restablece a 0.
Salidas				
Velocidad actual	int	pulsos/s		El voltaje medido.
Posición actual	int	pulsos		La suma de todos los pulsos medidos desde la puesta en marcha de Robotino o desde que "Restablecer posición" = true.

6.3.3.6.2 Salida de potencia



Este bloque de función asigna valores de consigna a la salida de potencia del Robotino (anteriormente Motor 4). La salida de potencia sólo puede utilizarse si el subprograma no utiliza la pinza.

La salida es materializada por un puente H, que puede suministrar hasta 5A de corriente continua.

El puente H es controlado por una señal PWM de alta frecuencia y un bit para la dirección.

El signo del valor de consigna dado por la entrada corresponde al bit de dirección.

El valor absoluto del valor de consigna influye la señal PWM.

Un valor de consigna de 0 no genera ninguna señal PWM, es decir, el puente H no suministra corriente alguna.

Un valor de consigna de 50 produce una relación high-low (alto-bajo) de la señal PWM del 50%.

Un valor de consigna de 100 genera una señal permanente alta, es decir, el puente H suministra la máxima corriente.

Entradas	Tipo	Unidad	Predefinido	Descripción
Valor de consigna	int		0	Establece el bit de dirección y la señal PWM. Margen -100 a 1000. Los valores inferiores a -100 son interpretados como -100. Los valores superiores a 100 son interpretados como 100.
Salidas				
Intensidad	float	A		Intensidad (corriente) que fluye por el puente H.

La intensidad suministrada por la salida de potencia está limitada de forma predeterminada. Para cambiar o inhabilitar esta limitación, debe editarse el archivo: /etc/robotino/robotino.xml en el Robotino.

Los nuevos valores se asignan al cabo de 2 segundos.

6.3.3.6.3 Pinza



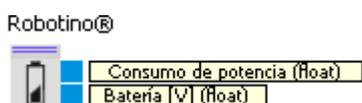
Utilice este módulo con una Pinza Robotino de Festo. La Pinza sólo puede utilizarse si el subprograma actual no contiene ninguna [salida de potencia](#) ¹⁵².

Entradas	Tipo	Predeterminado	Descripción
Abrir	bool	false	Si es verdadera (true) la pinza está abierta.
Salidas			
Abierta	bool		Verdadera (true) si la pinza ha alcanzado su posición abierta.
Cerrada	bool		Verdadera (true) si la pinza ha alcanzado su posición cerrada.

La Pinza debe conectarse el puerto X15 en la PCB detrás de la batería: cable marrón (+) a la izquierda, cable azul (-) a la derecha.

6.3.3.7 Sensores internos

6.3.3.7.1 Supervisión de la alimentación



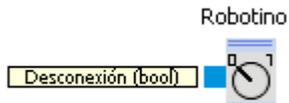
El módulo de supervisión de la alimentación del Robotino.

Entradas	Tipo	Unidad	Predeterminado	Descripción
Salidas				

Dispositivos

Consumo de potencia	float	A		La intensidad actual consumida por las baterías del Robotino.
Batería	float	Volt		Tensión de la batería.

6.3.3.7.2 Desconexión



Desconexión y apagado del Robotino.

Entradas	Tipo	Predeterminado	Descripción
Desconexión	bool	false	Si es verdadera (true), Robotino se desconecta y se apaga.

6.4 Joystick

El dispositivo "Joystick" permite el acceso a una palanca de mando conectada localmente.

6.4.1 Diálogo



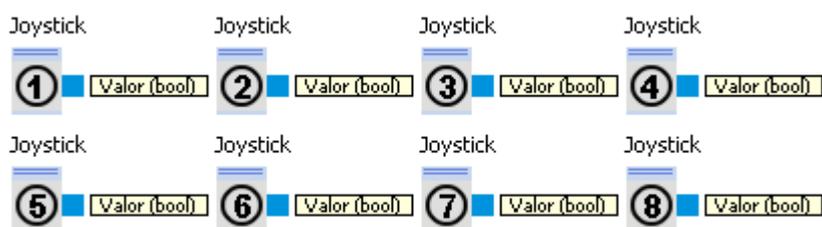
1	Lista de joysticks disponibles	El cuadro combinado contiene una entrada para cada joystick disponible en el ordenador. La lista se actualiza cada vez que se añade o se retira un joystick del ordenador. Al seleccionar un joystick, sus botones y ejes están disponibles a través de los correspondientes bloques de función.
2	Número de ejes	El número de ejes del joystick seleccionado.

3	Número de botones	El número de botones del joystick seleccionado.
---	-------------------	---

6.4.2 Boques de función

Bloques de función para leer estados de ejes y botones.

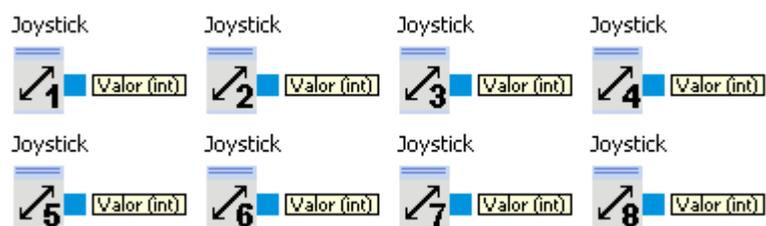
6.4.2.1 Botón



Leer el estado de un botón del joystick.

Salidas	Tipo	Descripción
Valor	bool	Verdadero (true) si el botón se halla presionado, de lo contrario, falso (false).

6.4.2.2 Eje



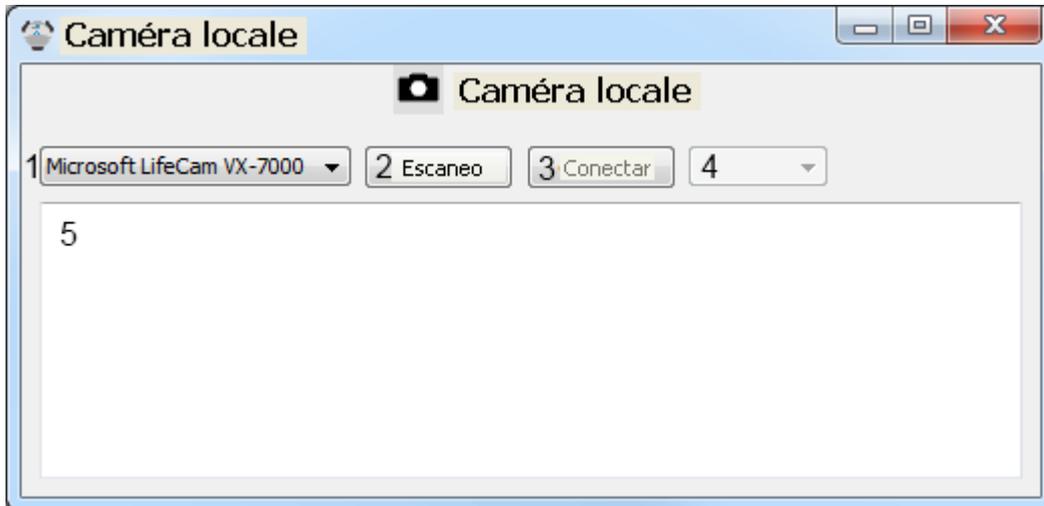
Leer la posición del eje de un joystick.

Salidas	Tipo	Descripción
Valor	int	Margen -1000 a 1000.

6.5 Cámara local

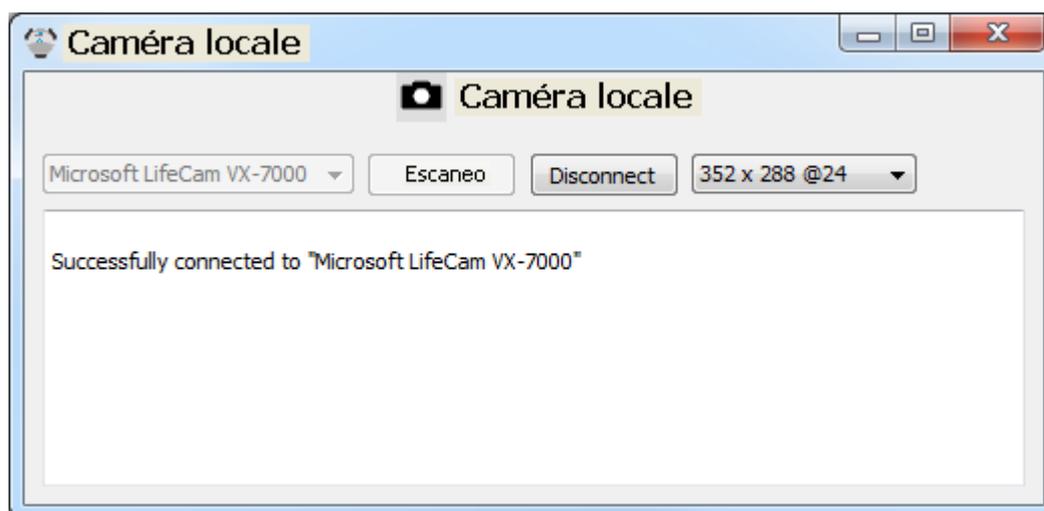
El dispositivo "Cámara local" permite el acceso a una cámara que esté conectada al ordenador (p.ej. una webcam).

6.5.1 Diálogo



1	Lista de cámaras disponibles	Aquí se muestran todas las cámaras conectadas al sistema. La lista se actualiza cuando se conecta o se retira una nueva cámara del ordenador.
2	Escaneo	Actualiza la lista de cámaras disponibles.
3	Conectar/Desconectar	Establece/Cierra la conexión con la cámara seleccionada.
4	Resolución/Profundidad de color	Aquí puede ajustar la resolución y la profundidad de color. Se muestran todas las resoluciones soportadas por la cámara.
5	Ventana de mensajes	Muestra los diferentes mensajes en forma de texto.

Tras seleccionar una cámara, se establece la conexión. Entonces es posible ajustar la resolución.



6.5.2 Boques de función

Los bloques de función permiten utilizar el dispositivo "Cámara local" en un subprograma.

6.5.2.1 Cámara

Caméra locale



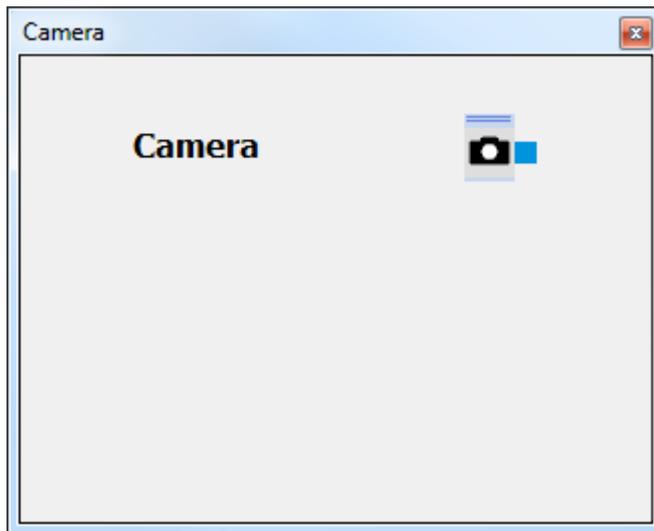
Imagen en directo de la cámara local.

Entradas	Tipo	Predeterminado	Descripción
Salidas			
Imagen	imagen		Imagen en directo

La profundidad de color y la resolución pueden seleccionarse en el [diálogo del dispositivo](#)¹⁵⁶.

Dispositivos

6.5.2.1.1 Diálogo



El diálogo de la cámara muestra la imagen actual. Para ajustar la resolución de la imagen, véase [diálogo del dispositivo](#) ^[156].

6.6 Cliente OPC

OPC es un interface estandarizado entre diferentes aplicaciones de software y controladores de diferentes módulos de hardware (p.ej. PLCs).

A un mismo Servidor OPC pueden conectarse varios Clientes OPC.

Un Servidor OPC (especial) a menudo es ofrecido por los fabricantes de PLC más conocidos.

En la muestra que sigue, Festo EasyPort se conectará al Robotino View a través del Servidor Festo EzOPC gratuito.

El Servidor EzOPC asigna las entradas/salidas de hasta 4 EasyPorts utilizando los denominados "Grupos (Groups)" y "Etiquetas (Tags)":

- El Dispositivo 1 mostrado como grupo "EasyPort1"
- Por lo tanto, la salida 1 aparece como etiqueta "EasyPort1.OutputPort1"

Seguir los pasos siguientes:

1. Instalar el EzOPC-Server.
2. Ejecutar el EzOPC Server y elegir "Process Simulation..." y "PLC via EasyPort".
3. Poner en marcha Robotino View.
4. Añadir el dispositivo "OPC Client"
5. En el menú de diálogo del dispositivo, seleccionar Ajustes predeterminados ▶ "Festo EzOPC EasyPort". Los valores predeterminados para el EasyPort serán cargados.

6. Si es necesario, seleccionar "FestoDidactic.EzOPC.1".
7. Iniciar la conexión.
8. Usar los bloques de función OPC Client para acceder a los datos OPC de las entradas/salidas del EasyPort.

Sugerencia: Si desea utilizar un PLC de otro fabricante, necesitará el OPC Server y el OPC Client de este fabricante. Utilice un OPC Client para ver qué etiquetas están disponibles en el OPC Server de su PLC.

Puede hallarse información adicional y descargas en: <http://www.opcconnect.com/>

6.6.1 Diálogo

1 FestoDidactic.EzOPC.2		
		2 Conectar
		3 Escaneo
	Tag	Valeur
DO_Port_1	EasyPort1.OutputPort1	4
DO_Port_2	EasyPort1.OutputPort2	
DO_Port_3		
DO_Port_4		
DI_Port_1	EasyPort1.InputPort1	
DI_Port_2	EasyPort1.InputPort2	
DI_Port_3		
DI_Port_4		
AO_Port_1	EasyPort1.AnalogOut0	
AO_Port_2	EasyPort1.AnalogOut1	
AO_Port_3		
AO_Port_4		
AI_Port_1	EasyPort1.AnalogIn0	
AI_Port_2	EasyPort1.AnalogIn1	
AI_Port_3	EasyPort1.AnalogIn2	
AI_Port_4	EasyPort1.AnalogIn3	

1	Seleccionar OPC Server	La lista muestra los Servidores OPC disponibles localmente.
---	------------------------	---

Dispositivos

2	Conectar	Establece una conexión con el OPC Server seleccionado.
3	Escaneo	Actualiza la lista de servidores OPC
4	Asignación	Esta tabla define la asignación de los bloques de función a los "Tags" OPC.

Fila	Bloque de función
DO_Port_1	Salida digital 1
DO_Port_2	Salida digital 2
DO_Port_3	Salida digital 3
DO_Port_4	Salida digital 4
DI_Port_1	Entrada digital 1
DI_Port_2	Entrada digital 2
DI_Port_3	Entrada digital 3
DI_Port_4	Entrada digital 4
AO_Port_1	Salida analógica 1
AO_Port_2	Salida analógica 2
AO_Port_3	Salida analógica 3
AO_Port_4	Salida analógica 4
AI_Port_1	Entrada analógica 1
AI_Port_2	Entrada analógica 2
AI_Port_3	Entrada analógica 3
AI_Port_4	Entrada analógica 4

El menú de contexto proporciona la siguiente funcionalidad:

Ajustes predefinidos ▶
Carga
Guardar
Ayuda

Ajustes predefinidos ▶ Festo EzOPC VirtualPLC	Carga una asignación adecuada para VirtualPLC
Ajustes predefinidos ▶ Festo EzOPC EasyPort	Carga una asignación adecuada para EasyPort
Load	Carga una asignación desde un archivo
Save	Guarda la asignación actual en un archivo.
Help	Muestra esta página de ayuda.

6.6.2 Boques de función

Los bloques de función permiten utilizar el dispositivo "OPC Client" en un subprograma.

6.6.2.1 Entradas

6.6.2.1.1 Entrada analógica

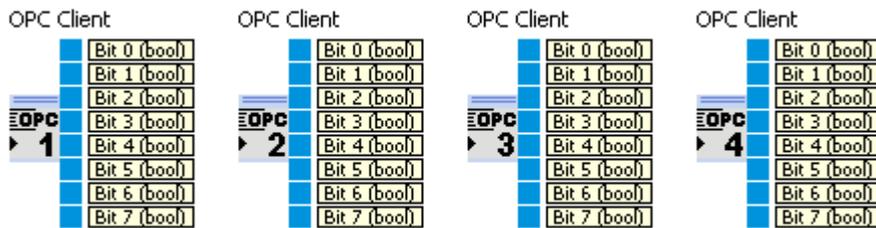


Lectura del "Tag" asignado a AI_Port_x con x en [1;4]

Salidas	Tipo	Descripción
Valor	int	Margen 0 a 65535

Dispositivos

6.6.2.1.2 Entrada digital



Lee valores de bit del "Tag" asignado a DI_Port_x con x en [1;4]

Salidas	Tipo	Descripción
Bit 0	bool	True si el bit 0 está activo
...		
Bit 7	bool	True si el bit 7 está activo

6.6.2.2 Salidas

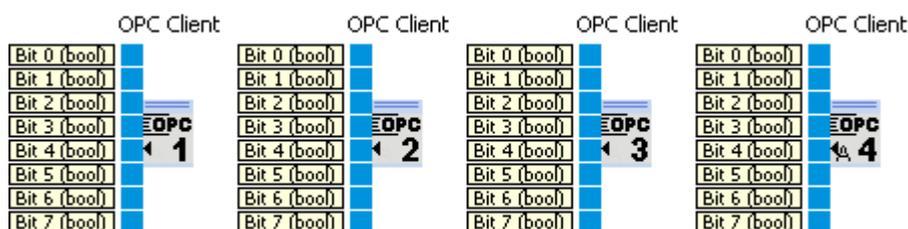
6.6.2.2.1 Salida analógica



Escribir un valor en el "Tag" asignado a AO_Port_x con x en [1;4]

Entradas	Tipo	Descripción
Valor	int	Margen 0 a 65535

6.6.2.2.2 Salida digital



Activa bits del "Tag" asignado a DO_Port_x con x en [1;4]

Entradas	Tipo	Descripción
Bit 0	bool	Si es verdadero (true) el valor enviado al OPC server es aumentado por $2^0 = 1$.
...		
Bit 7	bool	Si es verdadero (true) el valor enviado al OPC server es aumentado por $2^7 = 128$.

6.7 Intercambio de datos

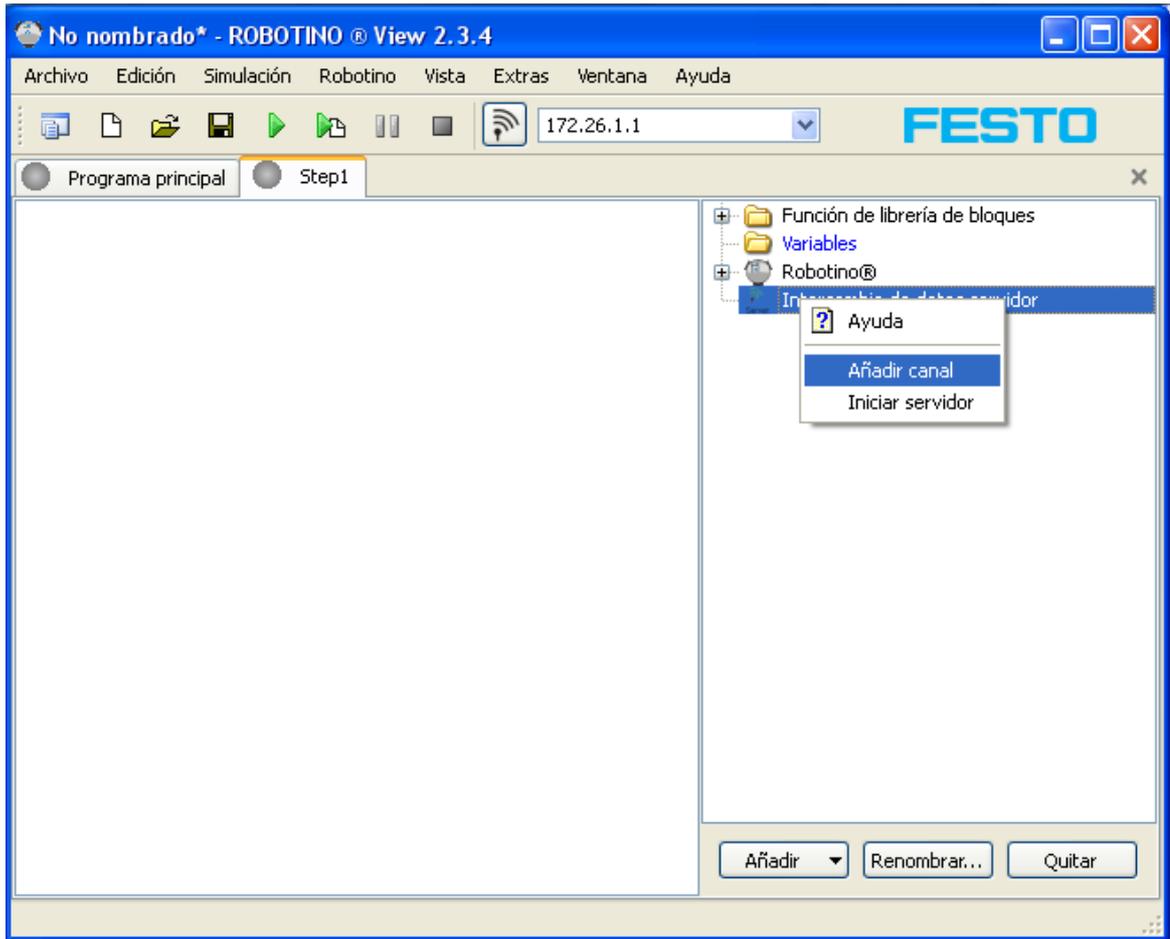
Los dispositivos de esta categoría se utilizan para intercambiar datos entre diferentes instancias de Robotino View en una red.

6.7.1 Servidor

El servidor de intercambio de datos puede ser utilizado por un número cualquiera de clientes para intercambiar datos a través de un número cualquiera de canales de comunicación. Los canales de comunicación se crean en el lado del servidor. Los canales de comunicación creados son difundidos a todos los clientes. Los clientes pueden elegir los canales con los que van a intercambiar datos con el servidor.

Servidor y clientes tienen los mismos derechos durante el intercambio de datos. Cuando un cliente escribe datos en un canal de comunicación, los datos se transfieren al servidor y de allí a todos los demás clientes. Si más de un participante está escribiendo en el mismo canal, es impredecible saber qué dato contendrá finalmente el canal de comunicación.

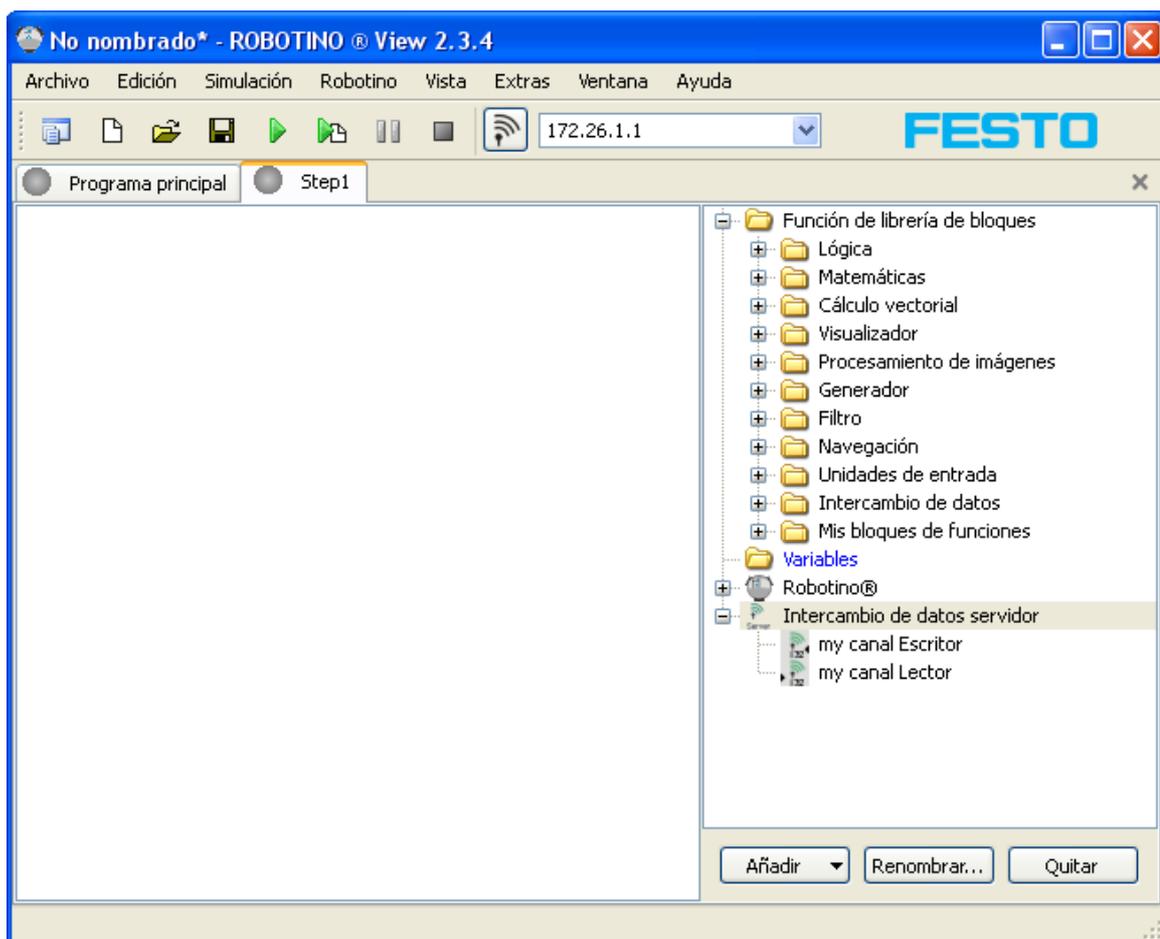
Después de añadir el dispositivo servidor de intercambio de datos en la librería de bloques de función, pueden añadirse los canales de comunicación por medio del menú de contexto de los servidores.



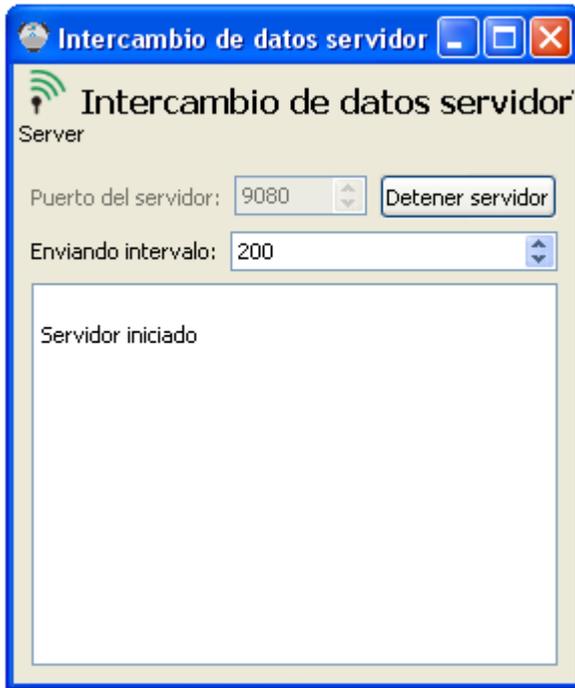
Aparece un diálogo para introducir el nombre y el tipo del nuevo canal.



El nombre del canal debe ser único y debe contener sólo caracteres ASCII, excluyendo el carácter "/". Presionando "Aceptar" se crea el canal. En la librería de bloques de función aparecen dos nuevos bloques con el nombre "nombre del canal Escritor" y "nombre del canal Lector". Estos bloques de función se utilizan para escribir o para leer datos de un canal de comunicación.



6.7.1.1 Diálogo



El diálogo del servidor de intercambio de datos se abre haciendo doble clic en el símbolo del dispositivo en la librería de bloques de función.

El puerto del servidor es el puerto TCP por el que escucha el servidor para conexiones de entrada.

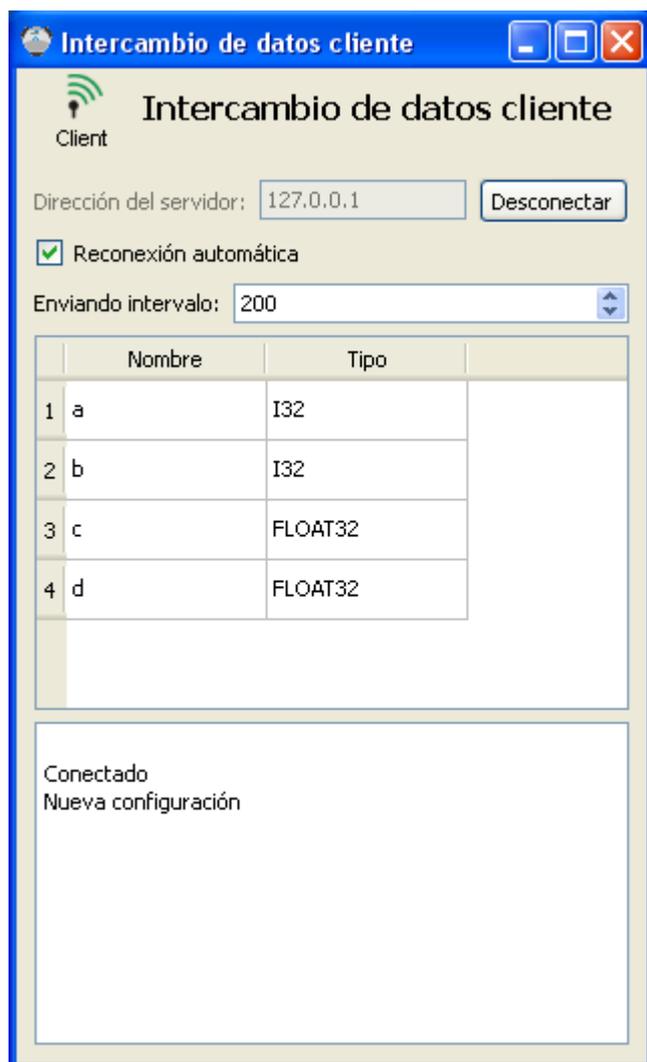
El intervalo de envío es el intervalo de tiempo que debe transcurrir tras una transmisión hasta que se permita la siguiente transmisión.

Con "Iniciar servidor" el servidor empieza a escuchar. A partir de entonces, los clientes pueden conectarse con el servidor.

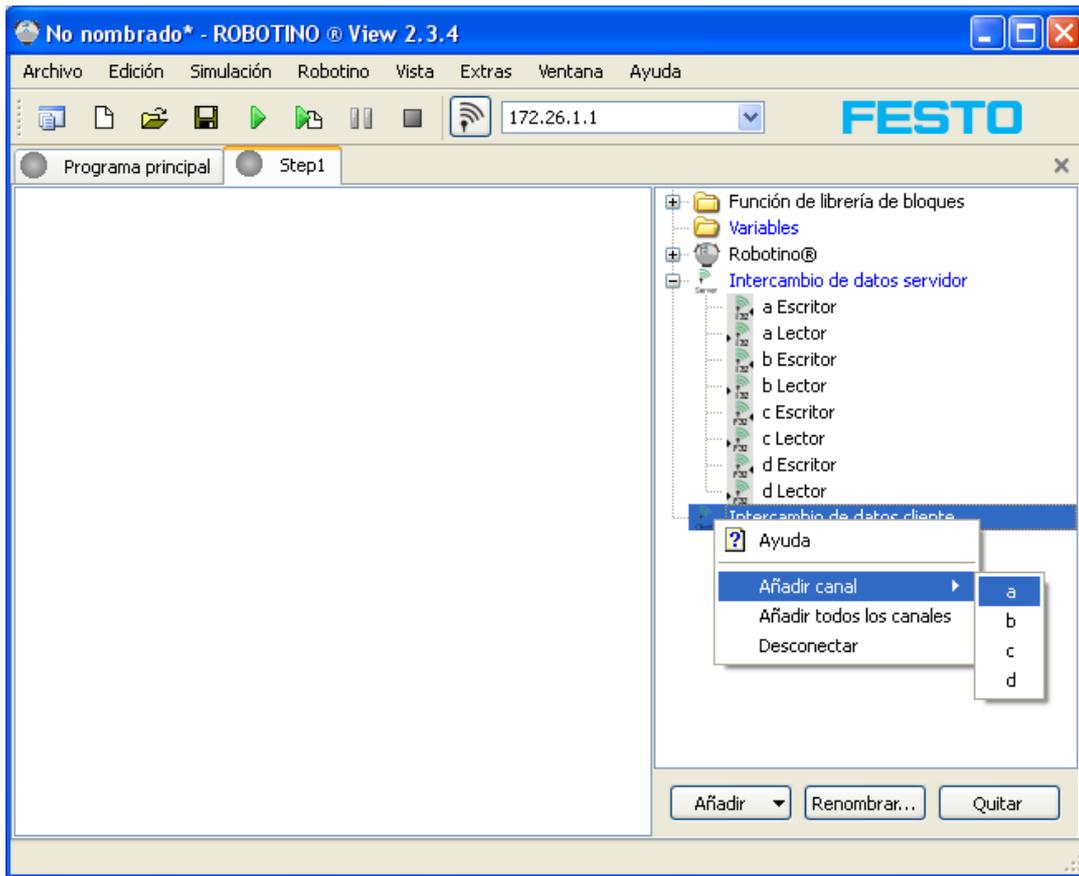
6.7.2 Cliente

El cliente de intercambio de datos se conecta a un [servidor de intercambio de datos](#)^[163]. Tras ello, pueden intercambiarse datos con el servidor utilizando los canales de comunicación del servidor.

Una vez que el cliente se ha conectado correctamente con el [servidor de intercambio de datos](#)^[163] se dispone de una lista de canales de comunicación.

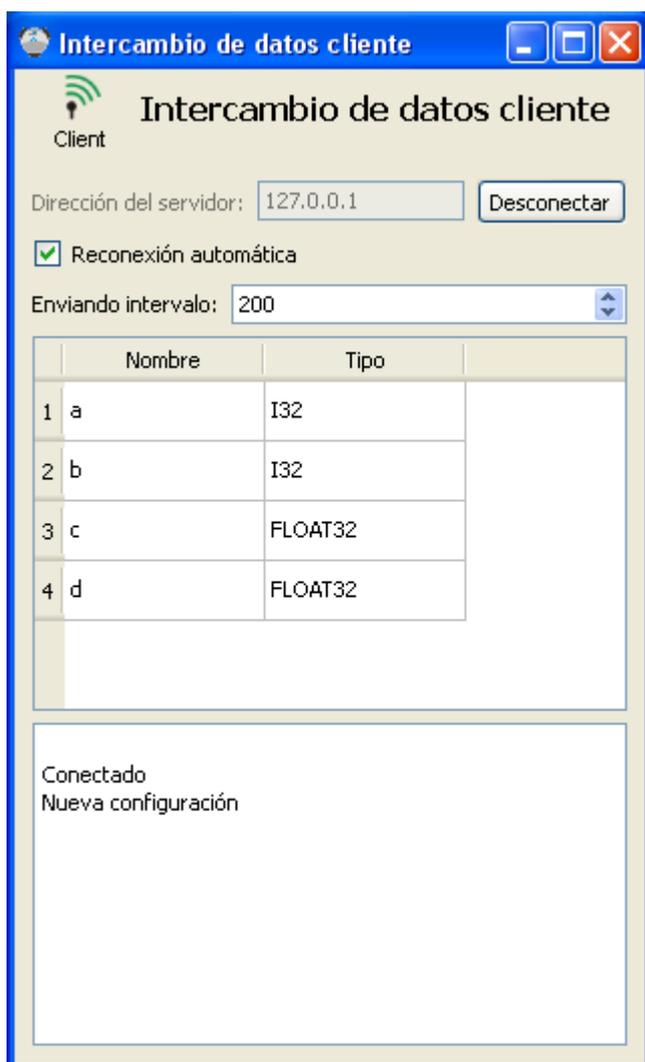


En el lado del servidor se han creado los canales de comunicación a,b del tipo I32 (entero (integer) de 32 bit) y c,d del tipo FLOAT32 (número real de coma flotante de 32 bit). Ahora pueden añadirse los canales al cliente en la librería de bloques de función.



Como en el [servidor de intercambio de datos](#)¹⁶³ la librería de bloques de función muestra dos bloques de función tras añadir un canal al cliente. A través del menú de contexto del cliente, pueden añadirse canales uno por uno o todos a la vez. Utilizando la entrada "Conectar" del menú de contexto puede establecerse la conexión con el servidor sin utilizar el diálogo del cliente.

6.7.2.1 Diálogo



La dirección del servidor es la dirección IP del servidor con el cual desea conectarse el cliente. Si sólo se da la dirección IP, la conexión se establece utilizando el puerto predeterminado del servidor 9080. Si el servidor está escuchando por un puerto diferente, puede especificarse el número del puerto en la dirección IP, separándolo por ":".

Si el servidor está escuchando en el host local por el puerto 8000, la dirección del servidor para el cliente deberá ser 127.0.0.1:8000.

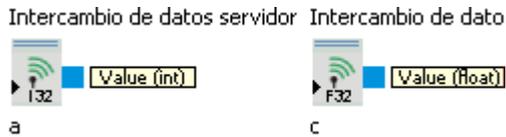
Si está activa la opción "Reconexión automática" los clientes tratan de establecer una nueva conexión si se pierde la conexión actual.

El intervalo de envío es el intervalo de tiempo que debe transcurrir tras una transmisión hasta que se permita la siguiente transmisión.

6.7.3 Boques de función

Los bloques de función se utilizan para intercambiar datos con los dispositivos.

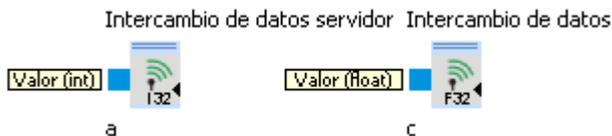
6.7.3.1 Lector



El Lector lee datos de un canal de comunicación.

Salidas	Tipo	Descripción
Valor	int, float, float array, laser range data	El valor del canal de comunicación.

6.7.3.2 Escritor



El Escritor escribe datos en un canal de comunicación.

Entradas	Tipo	Predeterminado	Descripción
Valor	int, float, float array, laser range data	0	El valor es enviado al servidor y de allí es difundido a todos los clientes.

6.8 Intercambio de datos UDP

Con el dispositivo de intercambio de datos UDP, pueden intercambiarse datos entre Robotino View y aplicaciones externas a través de UDP.

6.8.1 Protocolo

Especificación de la estructura de los datos

Byte	Función
------	---------

0	Message ID
1-2	Número de bytes de todo el mensaje N. El tipo es UINT16 ⁽¹⁷²⁾
3	Checksum (a inicializar con 0 cuando se genera el paquete, véase Suma de prueba ⁽¹⁷¹⁾)
N-1	Último byte del mensaje

6.8.1.1 Suma de prueba

Si la longitud del mensaje es inferior a 100 bytes, la suma s0 se calculará a partir de los bytes de todo el paquete. Si el mensaje contiene 100 o más bytes, s0 se calculará a partir del primero y los últimos 50 bytes del mensaje.

En ambos casos el byte de suma de prueba (checksum) debe inicializarse con 0. La suma de prueba se calcula para

checksum = 0xff - s0

```
unsigned char checksum( const unsigned char* payload, unsigned int payloadLength ) const
{
    unsigned char s0 = 0;

    if( payloadLength < 100 )
    {
        for( int i = 0; i < payloadLength; ++i )
        {
            s0 += payload[i];
        }
    }
    else
    {
        for( int i = 0; i < 50; ++i )
        {
            s0 += payload[i];
        }
        for( int i = payloadLength-1; i >= payloadLength - 50; --i )
        {
            s0 += payload[i];
        }
    }

    return ( 0xFF - s0 );
}
```

Para verificar si el paquete ha sido transmitido correctamente, los bytes de todo el mensaje serán acumulados al byte de suma s1 si la longitud del mensaje es inferior a 100 bytes. Si contiene 100 bytes o más, s1 se calcula a partir del primero y los últimos 50 bytes del mensaje.

El paquete es correcto si

s1 = 0xFF

6.8.1.2 Tipos de datos

Tip o	Ancho en bytes	Descripción
UINT16	2	<p>Byte0: low Byte1: high</p> <p>En un pequeño sistema endian, un valor de datos UINT16 puede copiarse directamente en el payload.</p> <p>Ejemplo:</p> <pre>//encoding uint16 value = 9873; char payload[2]; uint16* p = reinterpret_cast<uint16*>(payload); *p = value; //decoding value = *(reinterpret_cast<const uint16*>(payload));</pre>
INT32	4	<p>Byte0: low Byte3: high</p> <p>En un pequeño sistema endian, un valor de datos INT32 puede copiarse directamente en el payload.</p> <p>Ejemplo:</p> <pre>//encoding int32 value = -3459873; char payload[4]; int32* p = reinterpret_cast<int32*>(payload); *p = value; //decoding value = *(reinterpret_cast<const int32*>(payload));</pre>
UINT32	4	<p>Byte0: low Byte3: high</p> <p>En un pequeño sistema endian, un valor de datos UINT32 puede copiarse directamente en el payload.</p> <p>Ejemplo:</p> <pre>//encoding uint32 value = 3459873; char payload[4]; uint32* p = reinterpret_cast<uint32*>(payload); *p = value; //decoding value = *(reinterpret_cast<const uint32*>(payload));</pre>

6.8.1.3 Mensaje 0

Byte	Función
0	0
1	36
2	0
3	Suma de prueba $\overline{171}$
4-7	INT0 del tipo INT32 $\overline{172}$
8-11	INT1 del tipo INT32 $\overline{172}$
12-15	INT2 del tipo INT32 $\overline{172}$
16-19	INT3 del tipo INT32 $\overline{172}$
20-23	INT4 del tipo INT32 $\overline{172}$
24-27	INT5 del tipo INT32 $\overline{172}$
28-31	INT6 del tipo INT32 $\overline{172}$
32-35	INT7 del tipo INT32 $\overline{172}$

6.8.1.4 Mensaje 1

Byte	Función
0	1
1	36
2	0
3	Suma de prueba $\overline{171}$
4-7	INT0 del tipo INT32 $\overline{172}$
8-11	INT1 del tipo INT32 $\overline{172}$
12-15	INT2 del tipo INT32 $\overline{172}$
16-19	INT3 del tipo INT32 $\overline{172}$
20-23	INT4 del tipo INT32 $\overline{172}$
24-27	INT5 del tipo INT32 $\overline{172}$
28-31	INT6 del tipo INT32 $\overline{172}$
32-35	INT7 del tipo INT32 $\overline{172}$

6.8.2 Diálogo



El diálogo del dispositivo de intercambio de datos UDP se abre haciendo doble clic en el dispositivo en la librería de bloques de función.

En la ventana de diálogo pueden configurarse los datagramas de envío y recepción UDP:

Con "Puerto del servidor" se configura el número de puerto UDP en el cual escucha datagramas el servidor y desde el cual se envían.

Con "Iniciar servidor" el servidor empieza a escuchar. Una vez se ha puesto en marcha el servidor, los paquetes de datos UDP se reciben, se interpretan y se envían.

El "intervalo" es el intervalo de tiempo que debe transcurrir tras una transmisión hasta que se permita la siguiente transmisión.

Para cada mensaje (mensaje 0 ó mensaje 1) el envío puede activarse (on) o desactivarse (off) individualmente.

Las direcciones IP y puertos de los receptores de datos pueden introducirse en la tabla "Receptores" (Listeners). Si no se especifica un puerto, se utilizará el 9180 de forma predeterminada.

6.8.3 Boques de función

Los bloques de función se utilizan para intercambiar datos con los dispositivos.

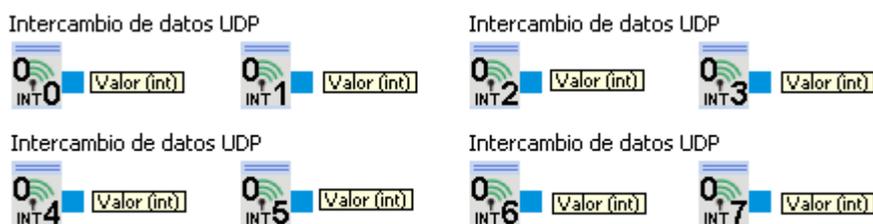
6.8.3.1 Mensaje 0

Los bloques de función en la categoría Mensaje 0 permiten el envío y la recepción de datos.

6.8.3.1.1 Entrada

Las entradas del mensaje 0 proporcionan los valores recibidos.

6.8.3.1.1 Lector



El lector lee datos y emite los datos recibidos. Hay un lector para cada uno de los INTO a INT7.

Salidas	Tipo	Descripción
Valor	int	El valor recibido

6.8.3.1.2 Salida

Las salidas se utilizan para enviar valores.

6.8.3.1.2 Escritor



Dispositivos

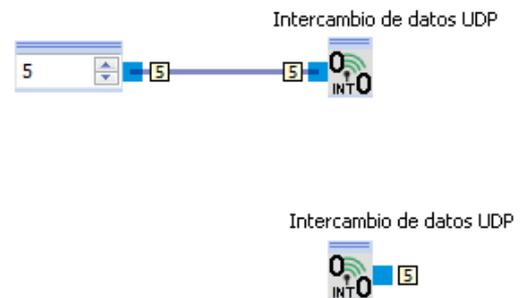
El escritor toma los datos a enviar y los pasa al dispositivo para enviarlos a los receptores a través de UDP. Hay un escritor para cada uno de los INTO a INT7.

Entradas	Tipo	Descripción
Valor	int	El valor a enviar

6.8.3.2 Mensaje 1

Mensaje 1 es idéntico al [Mensaje 0](#)^[175].

6.8.4 Ejemplo



7 Programación

Para compilar bloques de función y dispositivos de Robotino® View 2 es necesario el API.

7.1 Mis bloques de función

Puede hallar los siguientes ejemplos en

```
%ProgramFiles%\Festo\RobotinoView2\units\robview\MyFunctionsBlocks
```

o respectivamente

```
%ProgramFiles(x86)%\Festo\RobotinoView2\units\robview\MyFunctionsBlocks
```

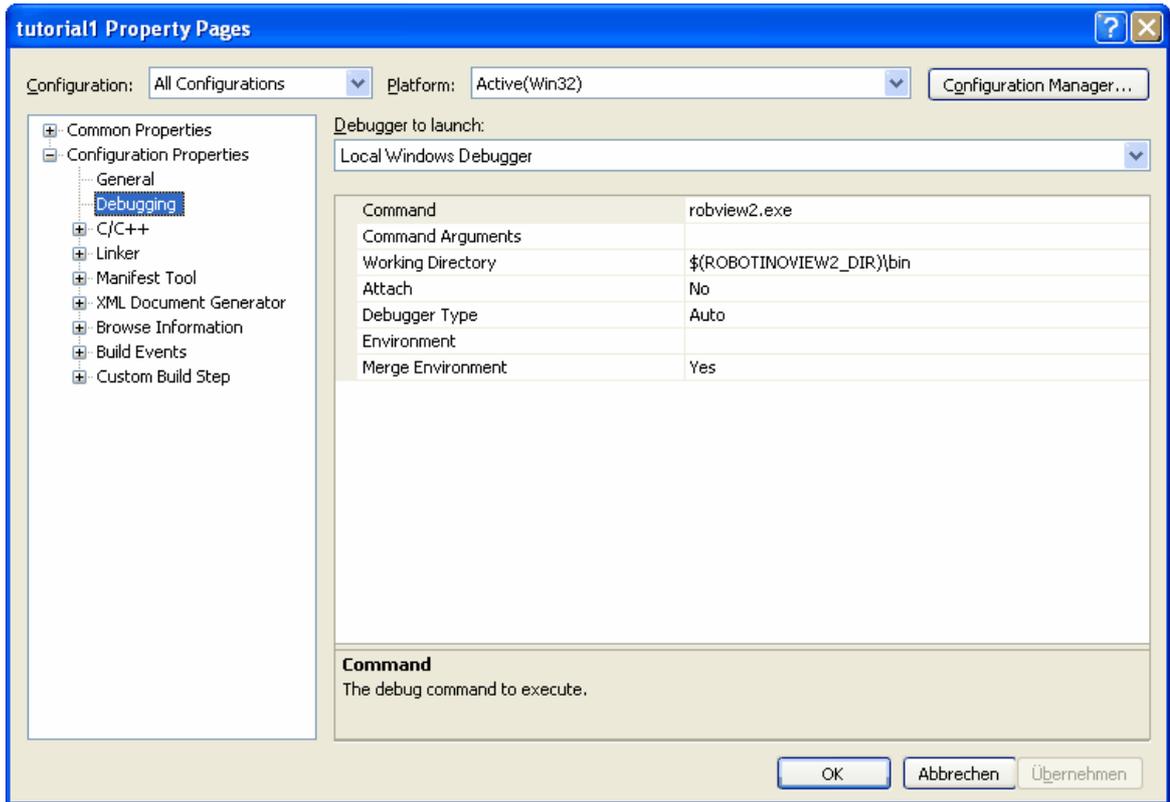
en sistemas de 64 bits. La variable de entorno %ProgramFiles% almacena la ruta a la aplicación instalada. Normalmente esta es "C:\Program Files".

Antes de abrir el Visual Studio Solution tutorialx.sln debería ejecutar el script

RUN THIS FIRST THEN START VS.cmd

desde la carpeta de tutorial actual. El script genera ajustes específicos del usuario, que no pueden ser almacenados en el archivo sln y permite la depuración de bloques de función.

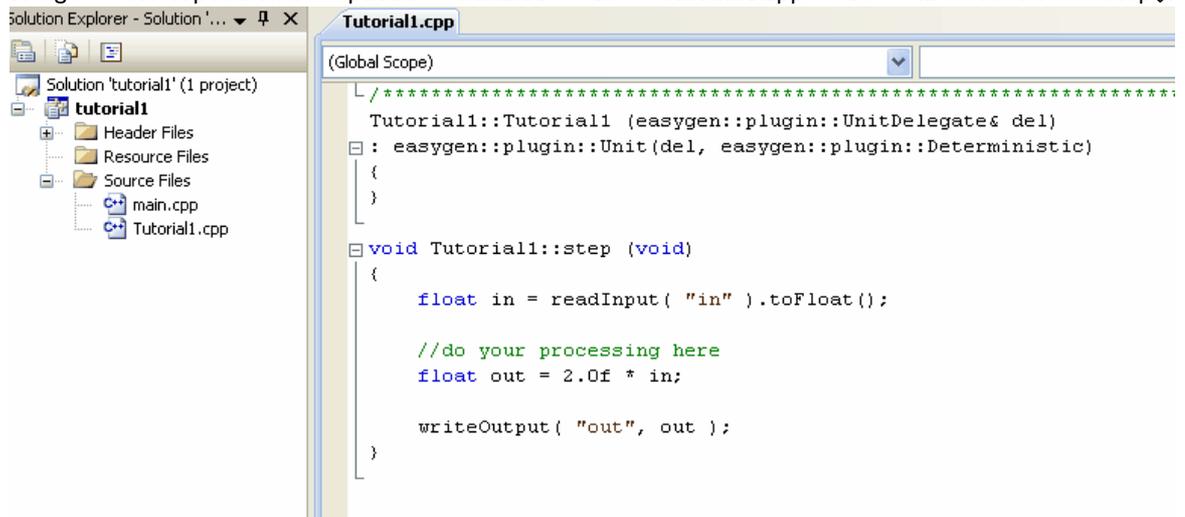
Para que funcione la depuración, Robotino View 2 debe especificarse como ejecutable con el directorio de trabajo correcto en los ajustes de proyecto como se muestra a continuación. Estos ajustes serán establecidos automáticamente si ha ejecutado antes "RUN THIS FIRST THEN START VS.cmd" como se ha descrito arriba.



7.1.1 Tutorial 1

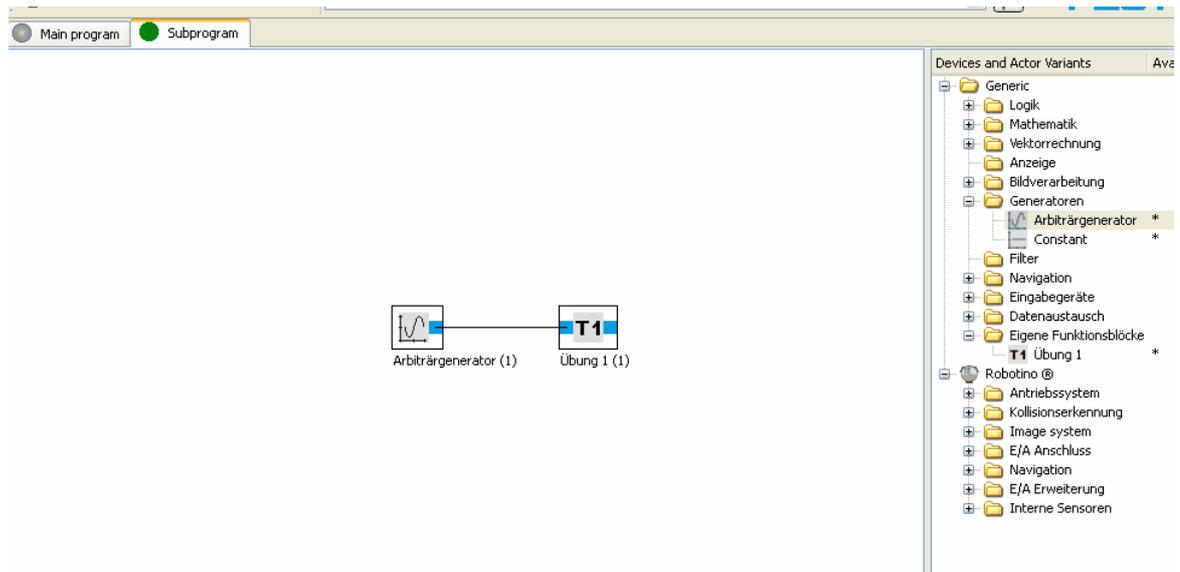
Carpeta: tutorial1.unit

Este tutorial explica cómo crear un bloque de función con un conector de entrada y uno de salida. El código correspondiente puede hallarse en Tutorial1.cpp en la función step().



El valor de entrada "in" es multiplicado por 2 y luego escrito en la salida. Haga aquí lo que desee. Para ver lo que sucede en su código, inicie el depurador presionando la tecla F5. El bloque de función es compilado y enlazado y se inicia Robotino View 2. Ignore el mensaje de diálogo acerca de que Robotino View 2 no contiene información de depuración. Como sea que usted no desea depurar Robotino View 2, sino sólo su bloque de función, este mensaje es irrelevante.

Cree un subprograma en Robotino View 2 conteniendo el bloque de función Tutorial 1 de Mis bloques de función. Ejecute la simulación del subprograma.



Coloque un punto de interrupción en su método step().

Programación

```
Tutorial1.cpp  
Tutorial1  
step()  
/***** Implementation of class Counter *****/  
Tutorial1::Tutorial1 (easygen::plugin::UnitDelegate& del)  
: easygen::plugin::Unit (del, easygen::plugin::Deterministic)  
{  
}  
void Tutorial1::step (void)  
{  
    float in = readInput ( "in" ).toFloat ();  
  
    //do your processing here  
    float out = 2.0f * in;  
  
    writeOutput ( "out", out );  
}
```

	Value	Type
in	1.6312256	float
out	3.2624512	float
this	0x02494f60	Tutorial1

Call Stack	
	Name
0	unit_robview_myfunctionblocks_tutorial1_simulation.dll!Tutorial1::step() Line 21
1	robview2.exe!00525e73()
2	[Frames below may be incorrect and/or missing, no symbols loaded for robview2.exe]
3	robview2.exe!004e9ab9()
4	robview2.exe!00525a0a()
5	robview2.exe!005208ba()
6	robview2.exe!005259c3()

- A -

Abs 62
 absoluto 62
 Actualización 9
 Adición 59
 Analógica 142
 AND 42
 AND FL 44
 Atajos de teclado 19

- C -

C++ 177
 Cámara 138, 156
 Cartesiano 77, 78
 Cliente 158, 163, 170
 comparación 60, 61, 62
 conectar 18
 Conexiones entre módulo 15
 constante 96, 104
 Contador 35, 39
 Controlador de posición 100
 Controlador de ruta 109
 Conversión del espacio de color 94
 Corredera 121
 crear 15

- D -

Desinstalación 9
 Desmultiplexor 41
 Detector de líneas 88
 Diferente 60
 Digital 141, 144
 Distancia 134
 División 56, 75

- E -

Ejemplos 26, 34, 177
 Entrada 119, 121, 142, 144, 151
 Entrada del transmisor giratorio 151
 Escalado 68
 Escalar 72, 76
 Estado 16
 Estructura y concepto del interface de usuario 9
 Evasión de obstáculos 117
 Extractor de segmentos 86

- F -

Filtro 99
 Función de transferencia 63
 Funciones 62, 63, 66, 67, 68

- G -

Generador 95
 Generadores 96, 97

- I -

Idioma 9
 Igual 60, 61
 imágenes 82, 86, 88, 90, 92, 94
 Información de la imagen 92
 Instalación 9
 Intercambio de datos 158, 163, 170

- J -

Joystick 154

- M -

Matemáticas 56, 57, 58, 59
 Matrices 69
 Máximo 67
 Mayor 62
 Mejoras 8
 Menor 61, 62
 Mínimo 66
 Mis bloques de función 177
 Módulo 14, 56
 Motor 129
 Multiplexor 40
 Multiplicación 57, 76

- N -

NAND 46
 NAND FL 48
 Navegación 100, 104, 105, 106, 107, 109, 117,
 132, 145, 146
 NOR 52
 Norma 74
 North Star 146
 NOT 52

- O -

Odometría 145

Index

Omniaccionamiento 132
OPC 158, 163
OR 49
Osciloscopio 80

- X -

XOR 51

- P -

Panel de control 119
Pinza 153
Polar 77, 78
pose 104, 105, 106
Programa de mando 14

- R -

Rectángulo 95
Región de interés 90
Relé 139
Resta 58, 72, 75
Robotino 129, 132, 133, 134, 138, 139, 141, 142,
144, 145, 153, 156
ROI 90
Rotar 79
RS 54
ruta 107

- S -

Salida 139, 141, 152, 153
Salida de potencia 152
Segmentador 82
Sensor táctil 133
Sensores 134
Servidor 158, 163, 170
Suavizado 99
Suma 59, 73, 76
Supervisión 153

- T -

Temporizador 97
Terminología 13
Tutorial 26, 34, 177

- U -

UDP 170

- V -

Vector 72, 73, 74, 75, 76, 77, 78, 79
Visualizador 80, 81