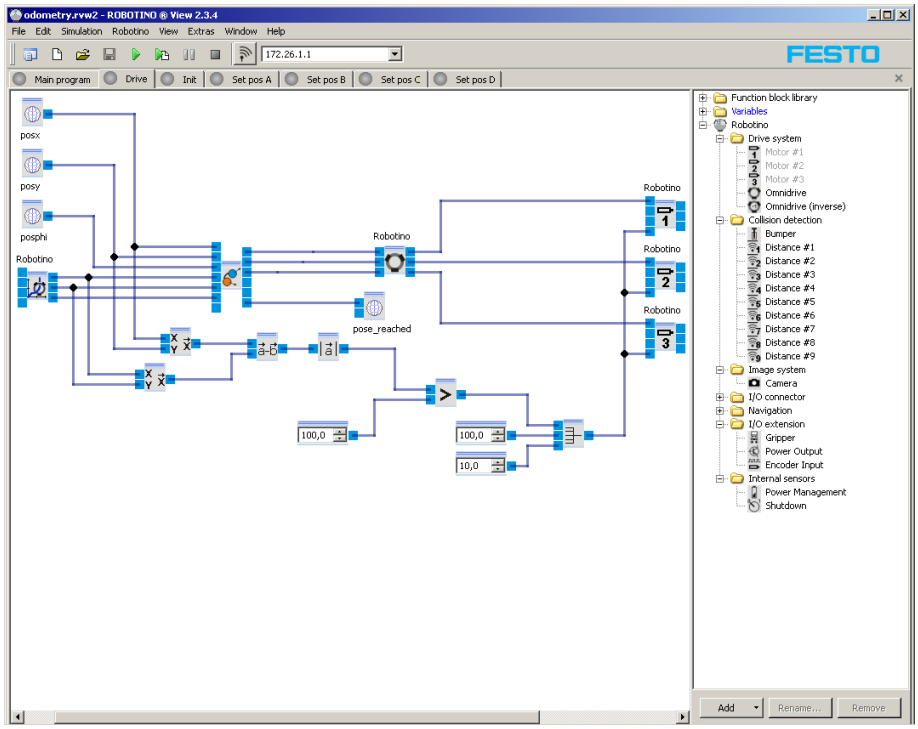
FESTO

Robotino® View 2



© 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 Bienvenue

Robotino® View est l'environnement graphique intuitif de programmation pour Robotino®. Robotino® View permet de créer et d'exécuter des programmes de commande pour Robotino®.

1.1 Nouveautés

Robotino® View 2 combine convivialité, évolutivité et utilisation intuitive. Nous avons veillé à préserver dans tous les perfectionnements les nombreux aspects positifs de Robotino® View 1. L'utilisateur familier de Robotino® View 1 retrouvera de nombreux éléments connus. Notamment, la bibliothèque de fonctions et la barre d'outils qui permet de se connecter à Robotino. A première vue, Robotino® View 2 ressemble fort à son prédécesseur. Mais si on y regarde de plus près, on constate qu'il y a eu de grands changements.

Qu'est-ce qui n'a pas changé ?

- Les programmes sont toujours représentés comme graphes de flux de données. La bibliothèque de blocs de fonction met à disposition les blocs de fonction à interconnecter.
- La connexion à Robotino peut toujours être établie ou coupée au moyen de la barre d'outils.

Quoi de neuf ?

- La commande séquentielle a été remplacée par un "véritable" langage séquentiel, issu de la programmation d'API et conforme à DIN EN 61131.
- Robotino® View 2 peut utiliser simultanément non seulement Robotino mais aussi autant de "dispositifs" d'autres marques que voulez. Vous pouvez donc piloter simultanément autant de Robotino que vous voulez à partir d'un seul programme.
- Les sous-programmes sous réutilisables dans un programme séquentiel autant de fois qu'on le souhaite.
- Les sous-programmes peuvent être importés dans des projets.
- L'utilisateur peut créer ses propres blocs de fonction en C++ et les charger comme plug-in dans la bibliothèque de blocs de fonction.
- L'utilisateur peut créer ses propres dispositifs en C++ et les charger comme plug-in dans le gestionnaire de dispositifs.

Modifications de la version actuelle :

- Le gestionnaire de dispositifs a été intégré à la bibliothèque de blocs de fonction. Voir Créer et éditer des dispositifs 122.
- Des projets peuvent être chargés sur Robotino et y être exécutés (à partir de la carte mémoire Robotino version 2.0). Voir Charger des projets 20.
- Nouveaux dispositifs pour l'échange de données via le réseau. Voir Dispositifs d'échange de données 158.

1.2 Installation, mise à jour et désinstallation

Vous devez posséder des droits d'administrateur pour pouvoir installer Robotino® View.

Pour installer Robotino® View conformez-vous aux instructions affichées dans les boîtes de dialogue.

Si vous voulez autoriser l'utilisation de Robotino® View par des utilisateurs sans droits d'administrateur, ouvrez

dans le centre de sécurité de Windows® XP, les paramètres du pare-feu et inscrivez Robotino® View dans

la liste des programmes exemptés des restrictions (port 80 et port 8080).

1.3 Changement de langue

Robotino® View identifie automatiquement au démarrage la langue activée de votre système Windows® et sélectionne la traduction correspondante de Robotino® View. Vous pouvez modifier ce paramétrage automatique à volonté à l'aide de la commande de menu Outils ▸ Changer de langue Le changement de langue est immédiatement visible, sans redémarrage, dans tous les dialogues de Robotino® View.

2 Familiarisation avec le plan de travail

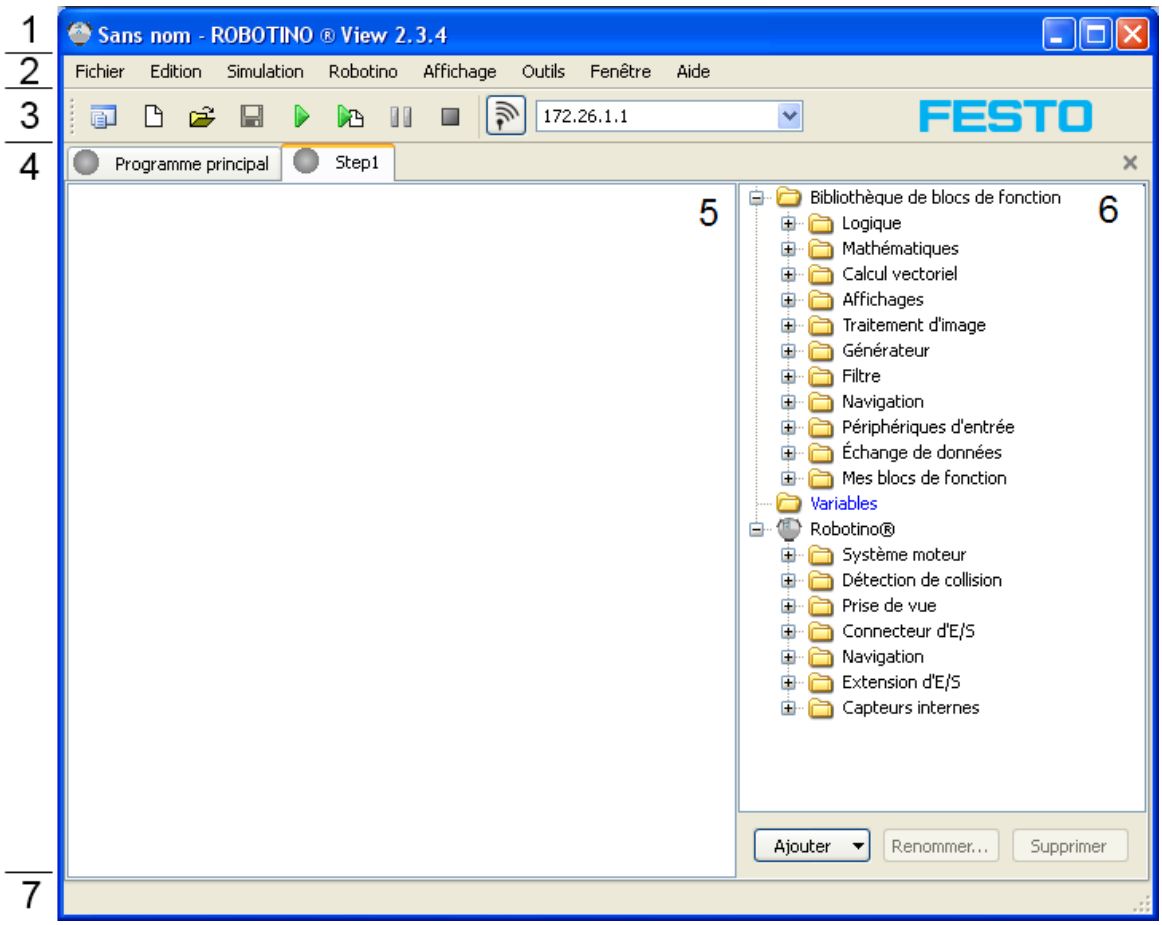
Familiarisez-vous d'abord avec le plan de travail et les désignations employées dans Robotino® View pour comprendre plus facilement le reste de la documentation.

Cette rubrique vous renseignera sur

- la structure et le concept de l'interface utilisateur,
- les désignations employées dans Robotino® View.

2.1 Structure et concept de l'interface utilisateur

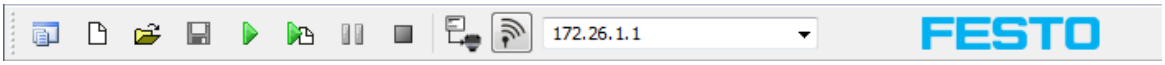
Au démarrage, Robotino® View affiche un projet vierge uniquement équipé du dispositif "Robotino". Le projet couvre l'ensemble du plan de travail.



Numéro	Désignation	Description
1	Barre de titre	• Affiche le nom du projet courant (ici sans nom). Si le projet a été modifié, le nom du projet est précédé d'un *. • Il est suivi du nom de l'application et du numéro de version (en l'occurrence Robotino View version 2.2.4). • Boutons standard pour réduire, agrandir et fermer l'application
2	Barre de menus	Menus permettant d'ouvrir/enregistrer ou éditer un projet, modifier l'affichage ...
3	Barre d'outils	• Boutons permettant d'accéder rapidement aux fonctions des menus. • Boutons permettant de démarrer et arrêter la simulation • Champ de saisie de l'adresse IP de Robotino et bouton de connexion (voir Barre d'outils Robotino 124). • Logo Festo avec lien vers la page d'accueil de Festo.
4	Sélection de programme	Vous pouvez sélectionner ici le programme principal ou les sous-programmes d'un projet. Le sous-programme "Step1" est actuellement visible.

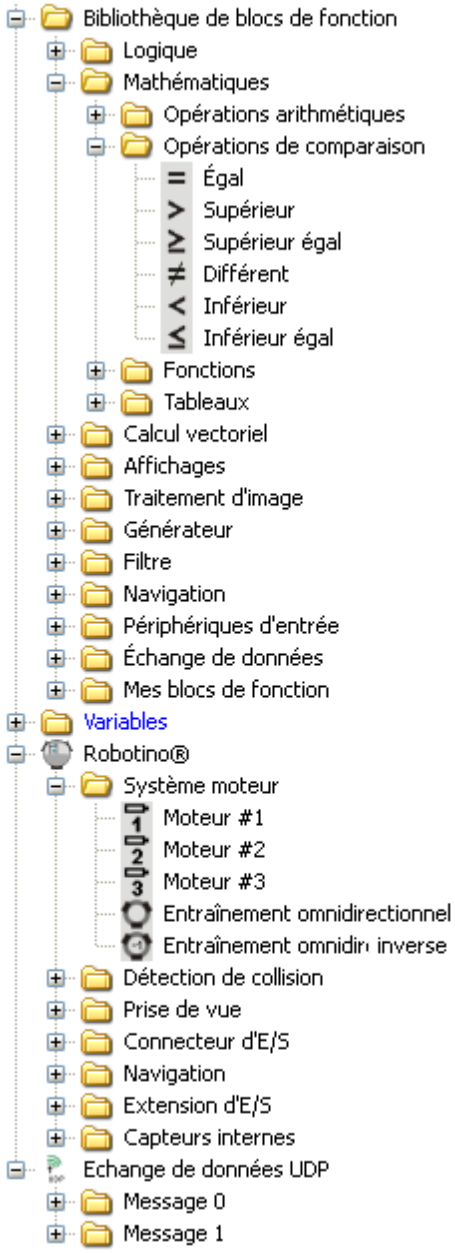
5	Création d'un programme	C'est ici que le programme est affiché et édité. Le sous-programme "Step1" est manifestement vide.
6	Bibliothèque de blocs de fonction 12	Elle affiche les blocs de fonction disponibles pour la programmation.
7	Barre d'état	Affiche des informations sur l'état du projet et de l'application.

2.1.1 Barre d'outils



	Créer un projet
	Créer un sous-programme
	Ouvrir un projet
	Enregistrer le projet
	Démarrer le programme principal
	Démarrer le programme actuellement visible
	Suspendre la simulation
	Arrêter la simulation
Entrée d'adresse et bouton de connexion	voir Barre d'outils Robotino 124
FESTO	Logo Festo avec lien vers la page d'accueil de Festo.

2.1.2 Bibliothèque de blocs de fonction



Dans le dossier Bibliothèque de blocs de fonction figurent tous les blocs de fonction disponibles dans chaque projet. On y trouve les blocs de fonction "Égal" à "Inférieur égal" du dossier "Opérations de comparaison".

Le dossier Robotino® contient les blocs de fonction qui sont mis à disposition par le dispositif "Robotino® 124". Un nouveau projet contient toujours le dispositif "Robotino® 124". Ce dernier comprend les blocs de fonction Robotino "Moteur1" à "Entraînement omnidirectionnel (inverse)" du dossier "Système moteur".

Le dossier Variables contient les blocs de fonction de lecture et d'écriture de variables.

Les blocs de fonction sont ajoutés au programme par glisser-déplacer après sélection dans la bibliothèque de blocs de fonction.

Les blocs de fonction de dispositifs sont souvent liés à des ressources matérielles concrètes. Le "Moteur 1" n'existe qu'une seule fois par Robotino. C'est la raison pour laquelle ce bloc de fonction ne peut être ajouté qu'une seule fois dans le sous-programme en cours. Si le "Moteur 1" existe déjà dans le programme, le bloc de fonction correspondant est "grisé" dans la bibliothèque de blocs de fonction.

2.2 Désignations

Bloc de fonction	Plus petite unité fonctionnelle constitutive d'un sous-programme. L'interconnexion de plusieurs blocs de fonction permet d'obtenir un comportement de robot complexe.
Sous-programme	Un sous-programme est constitué de plusieurs blocs de fonction interconnectés de la bibliothèque.
Programme principal	Une combinaison temporelle de sous-programmes, programmée dans le langage séquentiel.
Projet	Un projet se compose d'un programme principal et de plusieurs sous-programmes. Les projets sont chargés et enregistrés.
Réseau	Les blocs de fonction sont interconnectés au sein d'un ou de plusieurs réseaux.
Point de réseau	Ils se trouvent sur le réseau et permettent de structurer le réseau et de l'agencer graphiquement. Un point de réseau peut être le point de départ d'un nouveau sous-réseau.

3 Utilisation de Robotino® View

Robotino® View permet de réaliser des programmes de commande de Robotino. Vous apprendrez dans cette rubrique

- à créer un projet
- à charger un programme du projet
- à insérer des blocs de fonction dans un sous-programme
- à créer des réseaux entre les blocs de fonction
- à exécuter un sous-programme
- à exécuter un programme principal
- à vous connecter à Robotino

3.1 Créer un projet

Vous pouvez créer un projet de deux manières

- Par la commande de menu : Fichier ▸ Nouveau
- Par l'icône "Créer un projet" de la barre d'outils

3.2 Charger un projet

Il existe également trois possibilités pour charger un programme

- Par la commande de menu Fichier ▸ Ouvrir
- Par l'icône "Charger projet à partir d'un fichier" de la barre d'outils
- Par le raccourcis clavier Ctrl + O

Les projets possèdent l'extension de fichier .rvw2

3.3 Ajouter des blocs de fonction

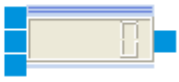
Après avoir créé ou chargé un programme, vous pouvez commencer à développer votre propre programme de commande ou à éditer le programme existant.

Exemple :

Vérifiez que le programme affiché n'est pas le programme principal mais un sous-programme. Dans tout nouveau projet, on trouve le sous-programme "Step1". Ce sous-programme s'affiche après la création d'un projet. La bibliothèque de blocs de fonction n'est visible que si un sous-programme est affiché.

Dans la bibliothèque de blocs de fonction, ouvrez le dossier Logique. Cliquez avec la souris sur le "Compteur-totalisateur 35" et placez-le par glisser-déplacer dans le sous-programme. Relâchez le bouton gauche de la souris à l'emplacement où vous souhaitez positionner le bloc de fonction.

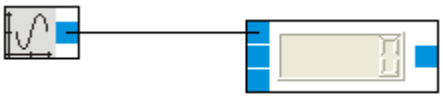
Dans la bibliothèque de blocs de fonction, ouvrez le dossier Générateur. Sélectionnez le générateur d'ondes arbitraire 92 et placez-le à gauche du "compteur-totalisateur 35".



3.4 Créer des réseaux entre les blocs de fonction

L'interconnexion des entrées et sorties des blocs de fonction crée un réseau entre ces derniers.

L'exemple de programme actuel contient le générateur d'ondes arbitraire 92 et le "compteur-totalisateur 35". Reliez la sortie du générateur d'ondes arbitraire 92 à l'entrée supérieure du "compteur-totalisateur 35".



Cliquez avec le bouton gauche de la souris sur la sortie du générateur d'ondes arbitraire 92. L'une des extrémités de la ligne du réseau qui apparaît alors est reliée à la sortie du générateur d'ondes arbitraire 92 tandis que l'autre est accrochée au pointeur de la souris.

En cliquant du bouton gauche de la souris, vous créez des points de réseau. Pour clore le réseau, cliquez sur l'entrée voulue du compteur-totalisateur 35".

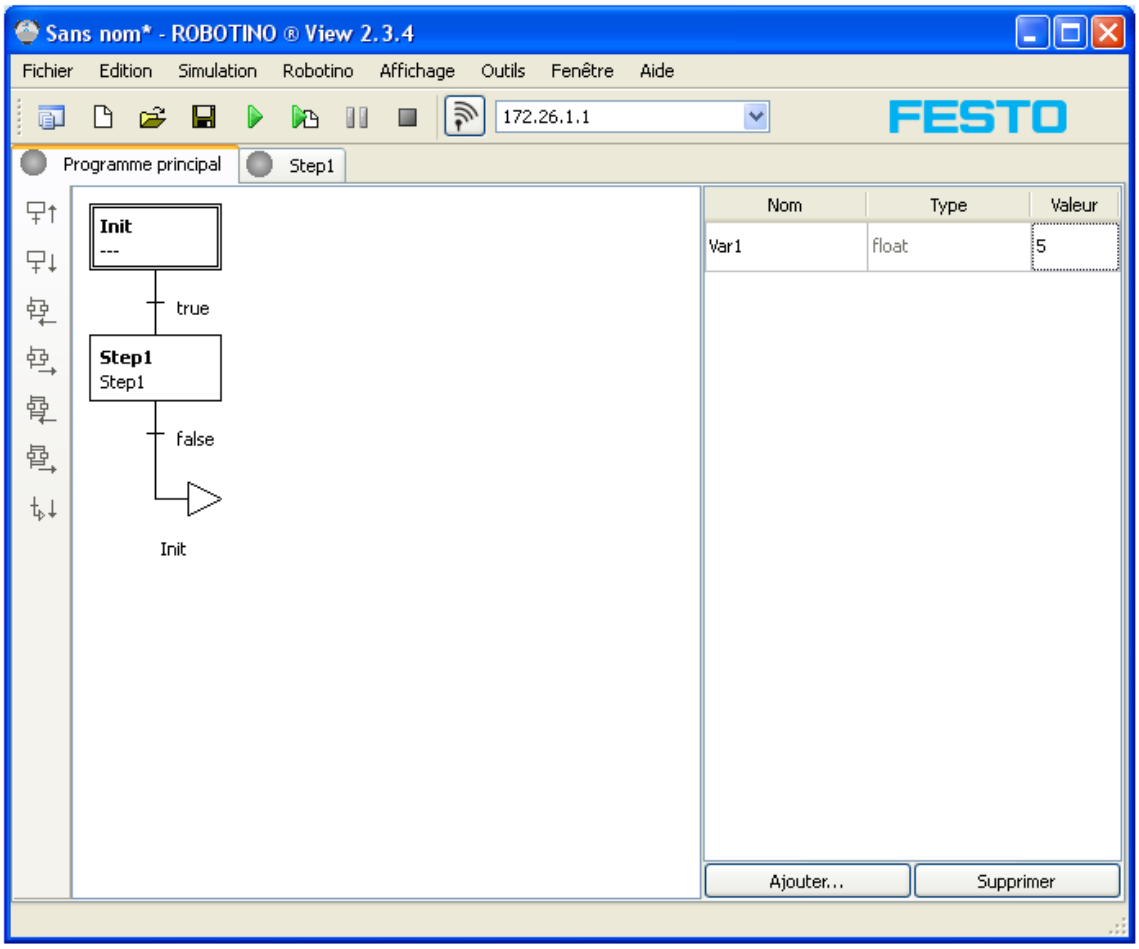
Suppression d'un point de réseau : Sélectionnez le point de réseau en cliquant dessus puis appuyez sur la touche **Suppr**.

Suppression d'une ligne de réseau entre deux points : Sélectionnez la ligne puis appuyez sur la touche **Suppr**

3.5 Variables globales

Les variables globales peuvent être lues et inscrites dans tous les sous-programmes d'un projet ; dans le programme principal, elles peuvent être utilisées comme conditions de transition.

Lorsque le programme principal est affiché, le gestionnaire de variables qui permet de créer, de supprimer ou de renommer les variables et de leur affecter une valeur de départ, se trouve dans le volet droit de l'écran.



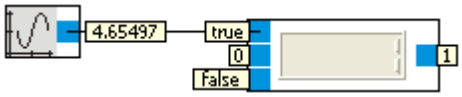
Programme principal avec gestionnaire de variables

Les variables globales enregistrent uniquement des nombres à virgule flottante. La prise en charge d'autres types de donnés sera implémentée dans les futures versions de Robotino View. Dès que la variable a été créée, les blocs de fonction pour la lecture et l'écriture de la variable sont disponibles dans la bibliothèque de blocs de fonction.

3.6 Exécuter un sous-programme

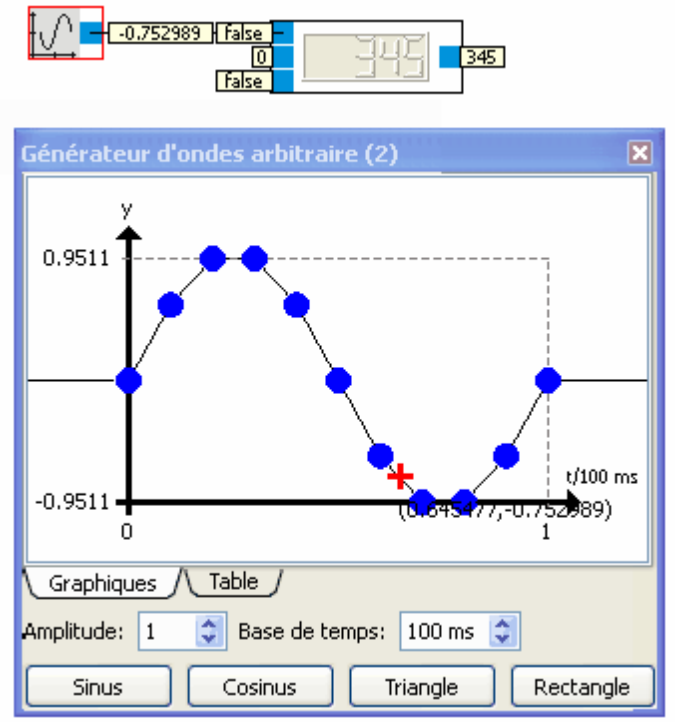
Après avoir interconnecté le générateur d'ondes arbitraire 92 et le "compteur-totalisateur 35", vous pouvez exécuter le sous-programme en cliquant dans la barre d'outils sur "Démarrer" .

Faites afficher les valeurs générées par le générateur d'ondes arbitraire 92 et le "compteur-totalisateur 35". Sélectionnez pour ce faire dans le menu Affichage ▸ Afficher valeurs des connexion ou utilisez le raccourci clavier **Ctrl + D**.



Vous voyez à présent que le générateur d'ondes arbitraire [92] génère des valeur de 0 à 10. Le "compteur-totalisateur [35]" incrémente sa sortie à chaque fois que l'entrée passe de faux (false) à vrai (true). Ceci n'intervient pour le moment qu'au démarrage du sous-programme. Ceci est dû à la conversion des nombres à virgule flottante (float) en valeurs logiques vrai ou faux. Voir à ce propos Transtypage [19]. Par ailleurs, lors du balayage de la fonction du générateur d'ondes arbitraire [92] le 0 n'est pas pris en compte.

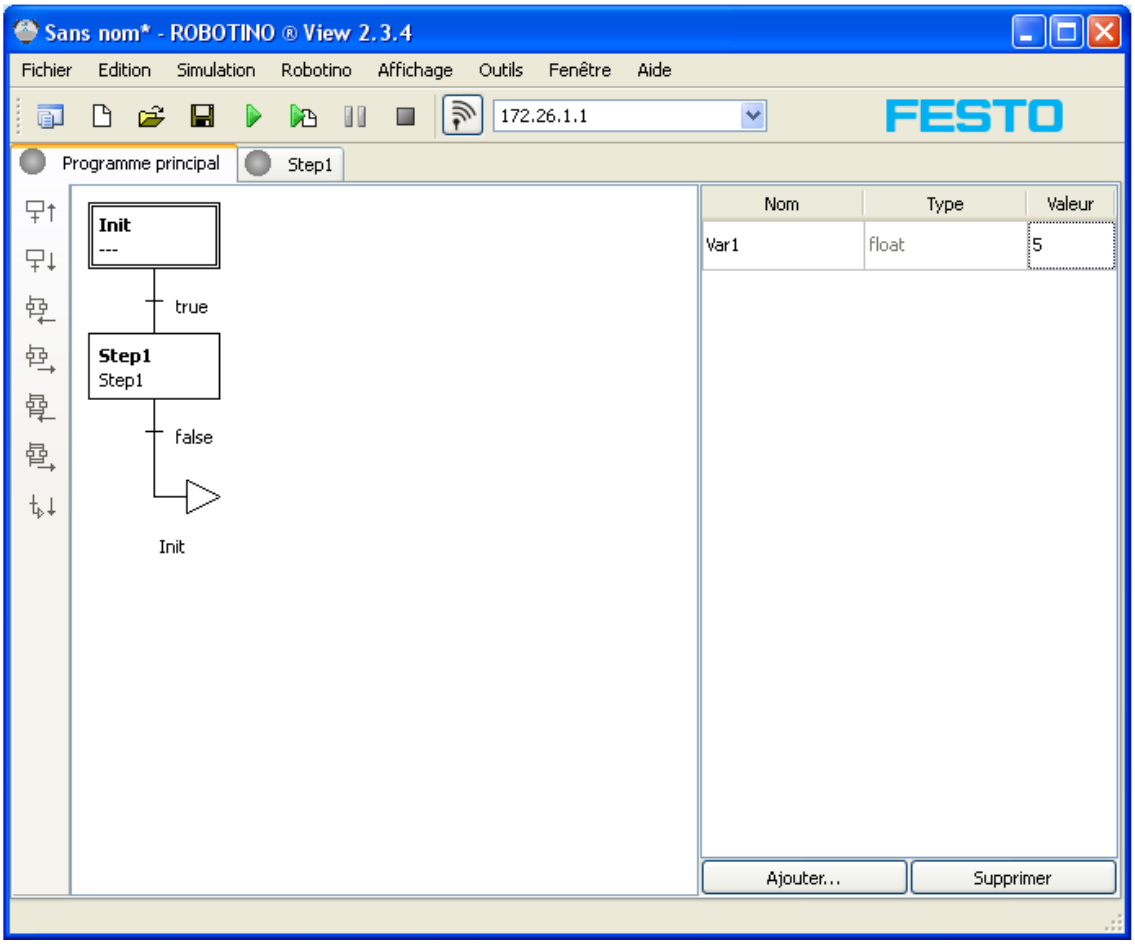
Pour voir le compteur compter, basculez le générateur d'ondes arbitraire [92] sur la génération d'une sinusoïde.



3.7 Exécuter le programme principal

L'actionnement de "Démarrer" dans la barre d'outils, exécute le programme affiché. Dans notre cas, le sous-programme "Step1" était visible et c'est donc "Step1" qui est exécuté. Dans un nouveau projet le sous-programme "Step1" est déjà intégré dans un programme principal.

Utilisez la sélection de programme [9], pour passer à l'affichage du programme principal.



L'actionnement de "Démarrer" dans la barre d'outil ne déclenche pas dans ce cas le sous-programme "Step1", mais le programme principal. Mais là aussi seul le sous-programme "Step1" correspondant à l'étape Step1 sera exécuté car d'une part l'étape Init est immédiatement quittée parce que la condition de transition est vrai (true) et d'autre part la condition de transition de l'étape succédant à l'étape Step1 est fausse (false) en permanence.

Le programme principal peut toujours être exécuté en appuyant sur "Démarre le programme principal" indépendamment du programme momentanément visible.

3.8 Se connecter à Robotino®

Dans le champ d'adresse entrez à présent l'adresse IP de Robotino (généralement 172.26.1.1). Cliquez sur le bouton de connexion à gauche du champ d'adresse. La connexion est établie dès que la couleur du bouton de connexion passe de gris à vert.

3.9 Clavier

Fonction	Raccourcis clavier

Ouvrir fichier	Ctrl + O
Enregistrer fichier	Ctrl + S
Enregistrer fichier sous	Maj + Ctrl + S
Quitter Robotino® View	Ctrl + Q
Annuler une action du plan de travail	Ctrl + Z
Restaurer	Ctrl + Y ou Maj + Ctrl + Z
Supprimer l'objet sélectionné	Suppr
Couper l'objet sélectionné	Ctrl + X
Copier l'objet sélectionné	Ctrl + C
Coller l'objet sélectionné	Ctrl + V
Déplacer l'objet vers le haut	
Déplacer l'objet vers le bas	
Déplacer l'objet vers la gauche	
Déplacer l'objet vers la droite	
Déplacer la vue vers le haut	Ctrl + 
Déplacer la vue vers le bas	Ctrl + 
Déplacer la vue vers la gauche	Ctrl + 
Déplacer la vue vers la droite	Ctrl + 
Annuler la sélection	Esc
Sélectionner tout	Ctrl + A
Zoom arrière	F3
Zoom avant	Maj + F3
Agrandir la grille	F4
Réduire la grille	Maj + F4
Afficher/masquer la bibliothèque	Ctrl + L
Afficher/masquer les données d'entrée et de sortie des blocs de fonction	Ctrl + D
Afficher/masquer les descriptions des entrées et sorties des blocs de fonction	Ctrl + T

3.10 Transtypage

Type de données	Transtypage implicite en	Description

int	float, bool	Lors du transtypage en bool, vrai est délivré si la valeur est différente de 0.
float	int, bool	Lors du transtypage en bool, vrai est délivré si la valeur est différente de 0.
bool	int, float	Vrai égale 1, faux égale 0.
pose	path	Une pose est converti en un itinéraire de longueur 1.
path	pose	Le résultat de la conversion d'un itinéraire en une pose est la première pose de l'itinéraire. Si l'itinéraire est vide, la conversion délivre une pose non valide.
float	float array	Un nombre à virgule flottante est converti en tableau de tableau de float array de longueur 1.

3.11 Mises à jour

Robotino View dispose d'une fonction de mise à jour en ligne. Pour vérifier l'existence d'une nouvelle version du logiciel, sélectionnez la commande "Rechercher mise à jour" dans le menu "Outils". Cette vérification est également exécutée automatiquement au démarrage du programme. S'il existe une nouvelle version, son téléchargement est proposé et elle peut être immédiatement installée automatiquement.

Le comportement de la fonction de mise à jour peut être configuré dans la boîte de dialogue Options ("Outils" ▸ "Préférences..."). Si la connexion à Internet n'est possible que via un proxy (serveur mandataire), vous pouvez également indiquer ici l'adresse, le port et les données d'accès au serveur proxy. Dans les réseaux d'entreprise, il est cependant souvent plus simple de reprendre les paramètres d'Internet Explorer ("Utiliser les paramètres de l'Internet Explorer").

3.12 Charger et exécuter des projets sur Robotino

À partir de la version 2.1.0. de Robotino View et de la carte mémoire version 2.0, il est possible de charger des projets via FTP sur Robotino et de les exécuter directement à partir de Robotino View. Cette fonction est accessible dans le Menu Robotino avec la commande ▸Charger projet.

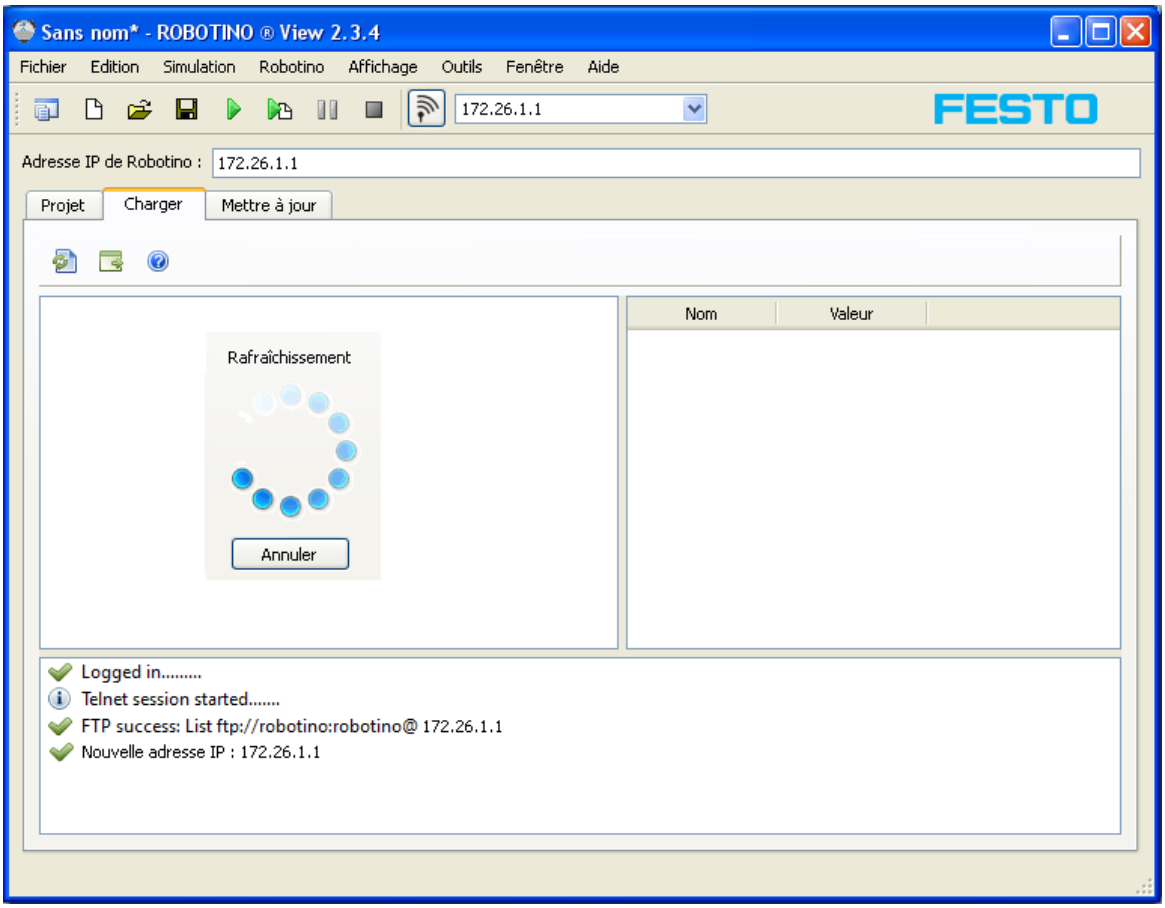
Lors de la première ouverture de la boîte de dialogue de chargement de projets, l'adresse IP momentanée du premier dispositif Robotino est reprise dans le champ de saisie "Adresse IP de Robotino". S'il n'existe pas de dispositif Robotino dans le projet courant, le champ de saisie reste vide.

Lors de l'ouverture de la boîte de dialogue, la représentation de l'arborescence sur Robotino est mise à jour (si cela n'a pas déjà été effectué auparavant). L'exécution d'une action est représentée par une animation. La mise à jour de la représentation peut également être déclenchée à l'aide du bouton  . Pour plus de détails sur la navigation dans l'arborescence de Robotino, veuillez consulter la rubrique Parcourir Robotino 21.

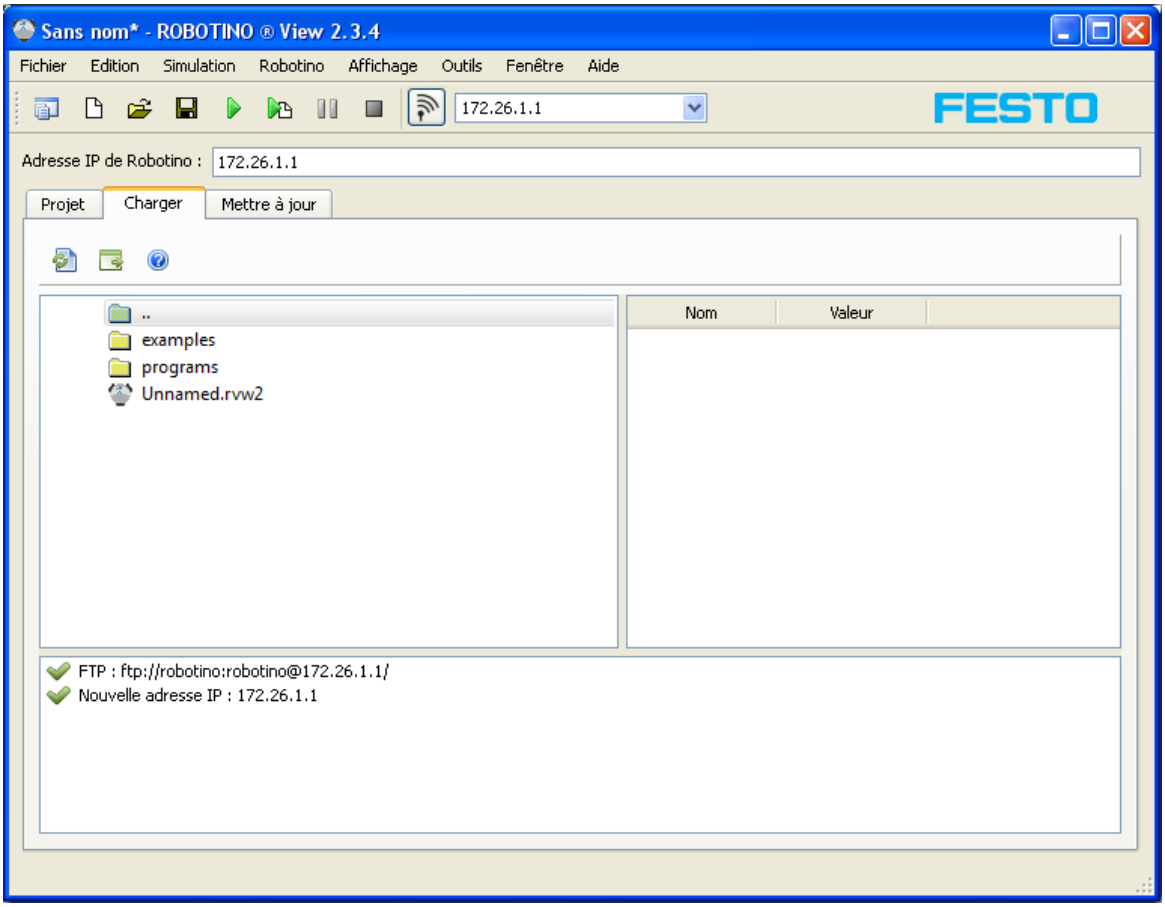
Le bouton  sert à charger le projet courant dans le répertoire actuellement affiché. Pour plus de détails sur le chargement et l'exécution de projets, veuillez consulter la rubrique Charger et exécuter 22.

3.12.1 Parcourir Robotino

Depuis la version 2.0 de la carte flash Robotino, un serveur FTP et Telnet a été installé sur le système fonctionnant sous Linux Ubuntu. Le serveur FTP est utilisé pour la représentation des fichiers enregistrés sur Robotino et pour le chargement de projets.



Après la première connexion, l'utilisateur se trouve dans le répertoire /home/robotino. Dans notre exemple, les répertoires "examples" et "programs" de même que le projet Robotino View "Unnamed2" se trouvent dans le répertoire courant. Si vous cliquez sur l'un des répertoires, la représentation est rafraîchie et affiche le contenu du répertoire sélectionné. Si, par contre, vous cliquez sur un projet Robotino View, l'exécution du projet sur Robotino est déclenchée. Voir à ce propos Charger et exécuter 22.

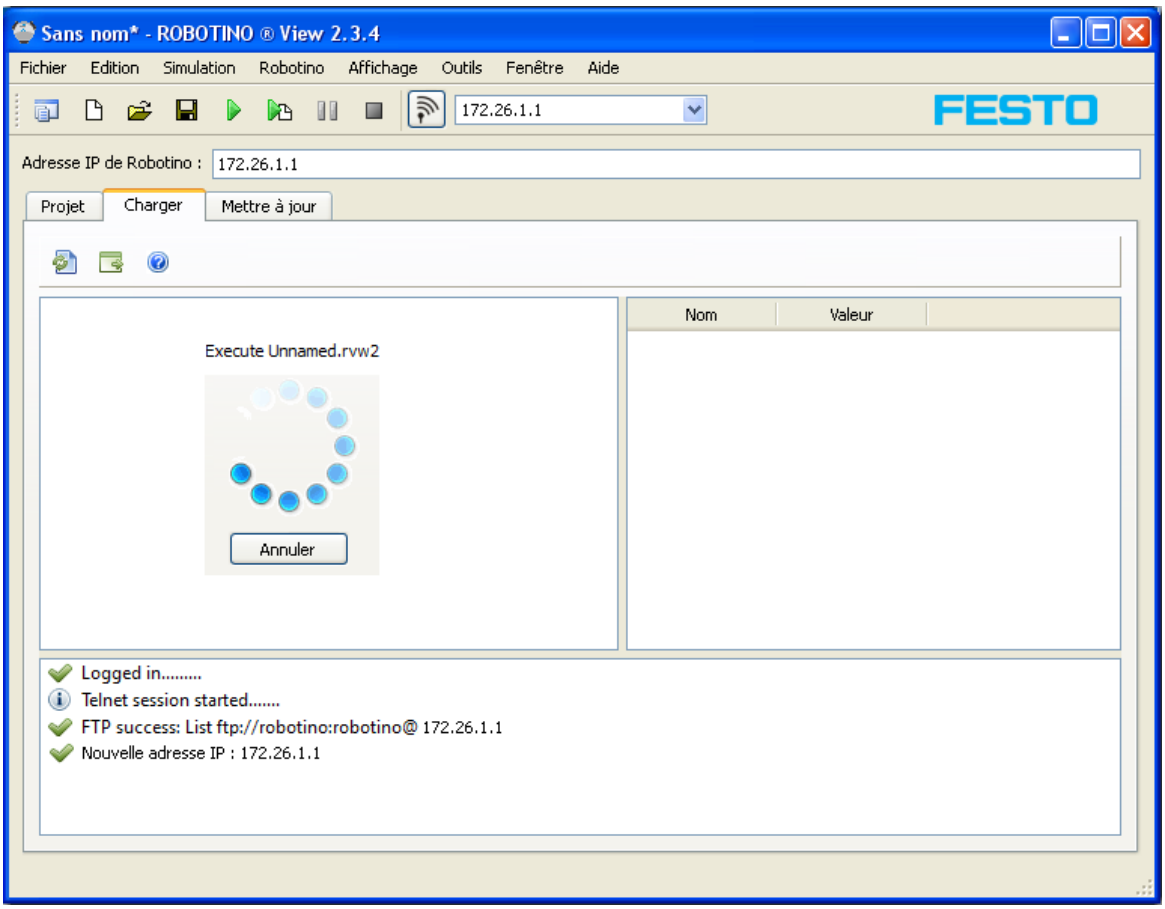


Le client FTP intégré à Robotino View a pour nom d'utilisateur "robotino" et pour mot de passe "robotino". Vous pouvez ainsi vous connecter à Robotino avec FileZilla p. ex. et y créer des répertoires ou supprimer des projets.

3.12.2 Charger et exécuter

Avant de passer à l'exécution, il est conseillé de vérifier que Robotino est bien doté de la dernière version de Robotino View. La mise à jour des paquets sur Robotino est décrite dans la rubrique Mise à jour des paquets Robotino 23.

En cliquant sur un projet Robotino View dans l'arborescence des répertoires, vous activez l'exécution de ce projet sur Robotino au moyen de l'interpréteur de Robotino. Avant que l'exécution du projet ne débute, il faut que l'interpréteur soit chargé. Ce processus dure quelques secondes. Le volet Journal affiche l'état de l'exécution.



Le fait de cliquer sur un projet Robotino View démarre une session Telnet. Il faut ensuite se connecter avec le nom d'utisateur "robotino" et le mot de passe "robotino". L'exécution démarre immédiatement après le message "Chargement du projet en cours". Ce processus peut être interrompu à tout moment.

La fenêtre à côté de l'affichage de la progression, visualise les valeurs des variables globales du projet exécuté sur Robotino. La fréquence de rafraîchissement peut être paramétrée sous Outils ▸ Préférences... ▸ Charger & exécuter ▸ Intervalle de débogage.

3.13 Mise à jour des paquets Robotino

À partir de la version 2.4.0. de Robotino View et de la carte mémoire version 2.0, il est possible de mettre à jour les paquets Linux installés sur Robotino via Robotino View. Cette fonction est accessible dans le Menu Robotino avec la commande ▸ Mise à jour logiciel.

Lors de la première ouverture de la boîte de dialogue de mise à jour des paquets, l'adresse IP momentanée du premier dispositif Robotino est reprise dans le champ de saisie "Adresse IP de Robotino". S'il n'existe pas de dispositif Robotino dans le projet courant, le champ de saisie reste vide.

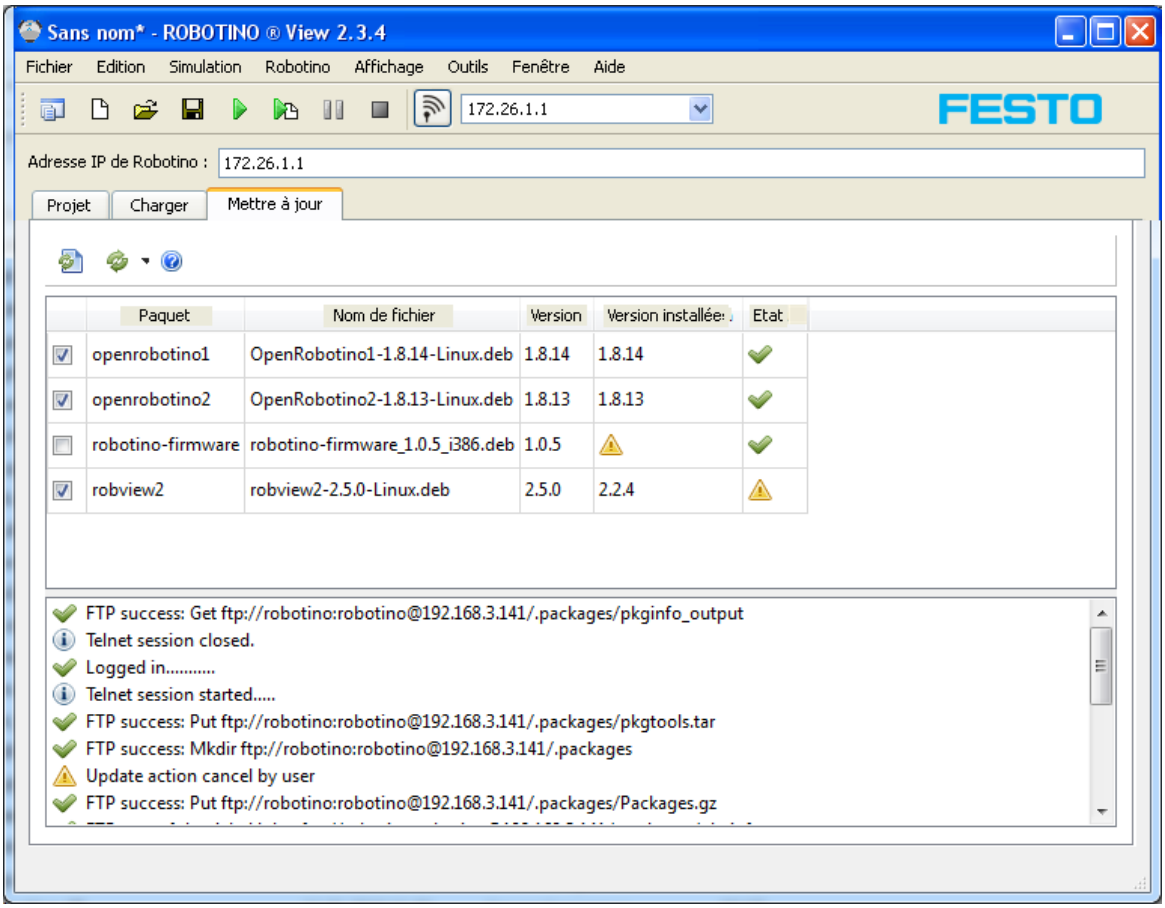
Lors de l'ouverture de la boîte de dialogue, les informations de paquet sont mises à jour (si cela n'a pas déjà été effectué auparavant). Pendant la mise à jour, l'application est entièrement verrouillée. L'opération peut cependant être interrompue sans problème à tout moment.

La mise à jour peut être forcée à l'aide de l'icône .

Après une mise à jour réussie, l'écran affiche les versions des paquets installés localement et celles des paquets sur Robotino. Les symboles d'état signifient :

Pas d'information disponible ou la version installée n'est pas à jour

Le paquet installé sur Robotino est à jour



Dans le première colonne de l'affichage des versions, il est possible d'ajouter des paquets au processus de mise à jour ou d'en exclure. Les paquets "openrobotino1", "openrobotino2" et "robview2" sont inclus par défaut au processus de mise à jour.

Dans la figure affichée ci-dessus, le paquet "robview2", installé sur Robotino, n'est pas à jour. La version locale est 2.5.0. La version installée sur Robotino est cependant 2.2.4. L'installation de nouveaux paquets est démarrée à l'aide de l'icône . La boîte de dialogue de mise à jour signale que l'opération est en cours d'exécution. La volet Journal permet de suivre la progression de l'opération. Après installation, l'affichage des versions est rafraîchi.

3.13.1 Installation du firmware de Robotino

Le paquet "robotino-firmware" est un cas particulier. La routine de mise à jour vérifie la présence sur Robotino d'une carte d'E/S EA09. En présence d'une carte d'E/S EA09, le numéro de version est lu directement sur la carte. En l'absence de carte EA09, l'icône ⚠ est affichée à la place du numéro de version. L'état du paquet est tout de même ✔, parce qu'il n'est pas nécessaire d'installer le paquet "robotino-firmware".

La mise à jour du firmware sur Robotino par le paquet "robotino-firmware" étant un processus critique, il n'est pas mis à jour par défaut. N'intégrez ce paquet au processus de mise à jour que si vous savez exactement pourquoi la mise à jour s'impose. L'installation du firmware est décrite dans la rubrique Installation du firmware de Robotino.

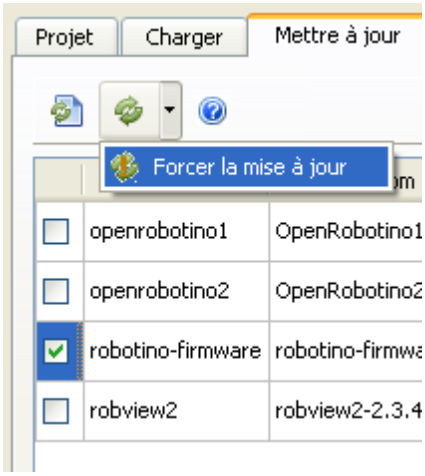
Le firmware du microcontrôleur installé sur cette carte d'E/S (un NXP LPC 2378) peut être mis à jour par le PC104 de Robotino. Ce processus est critique. L'échec du chargement d'un nouveau firmware se solde par les effets suivants :

1. Robotino ne peut plus être mis hors tension en appuyant sur la touche marche/arrêt.
2. Si vous appuyez sur la touche marche/arrêt Robotino est mis sous tension. Il est remis hors tension aussitôt que vous relâchez la touche.

à propos de 1) Vous pouvez mettre Robotino hors tension en déposant le boîtier de commande

à propos de 2) Maintenez la touche marche/arrêt enfoncée jusqu'à ce que le chargement du nouveau firmware ait réussi.

Si vous voulez uniquement mettre à jour ou réparer le firmware, sélectionnez uniquement le paquet "robotino-firmware". Le chargement du paquet peut alors être forcé à l'aide du bouton "Forcer mise à jour de paquet".



3.13.2 Informations internes

Le processus de mise à jour repose sur une combinaison de commandes Telnet, FTP et du gestionnaire de paquets apt de Linux.

Le fichier pkgtools.tar est d'abord copié du répertoire Dossier_d'installation\packages dans le répertoire /home/robotino/.packages. Le fichier est décompressé à l'aide de Telnet. Le script pkginfo.sh fournit des informations sur les paquets installés.

Pour l'installation des paquets, ces derniers sont copiés via FTP du Dossier_d'installation\packages dans le répertoire /home/robotino/.packages. Le fichier Packages.gz est également copié. Il contient les informations sur les paquets.

Le script pkginstall.sh modifie d'abord /etc/apt/sources.list et y inscrit le répertoire /home/robotino/.packages comme unique source de paquets. La commande apt-get est ensuite utilisée pour installer les paquets.

pkgremove.sh force la désinstallation de paquets.

La commande startOpenrobotino1.sh redémarre les services Robotino.

4 Exemples

4.1 Programmes séquentiels

Ce chapitre décrit la réalisation d'un programme séquentiel simple avec branches alternatives.

4.1.1 Tutoriel 2

L'exercice montre comment réaliser un programme séquentiel avec des branches alternatives. Le programme fini se trouve sous examples/sfc/tutorial2.rvw2

Le programme prêt à exécuter se présente comme indiqué dans la figure 1.

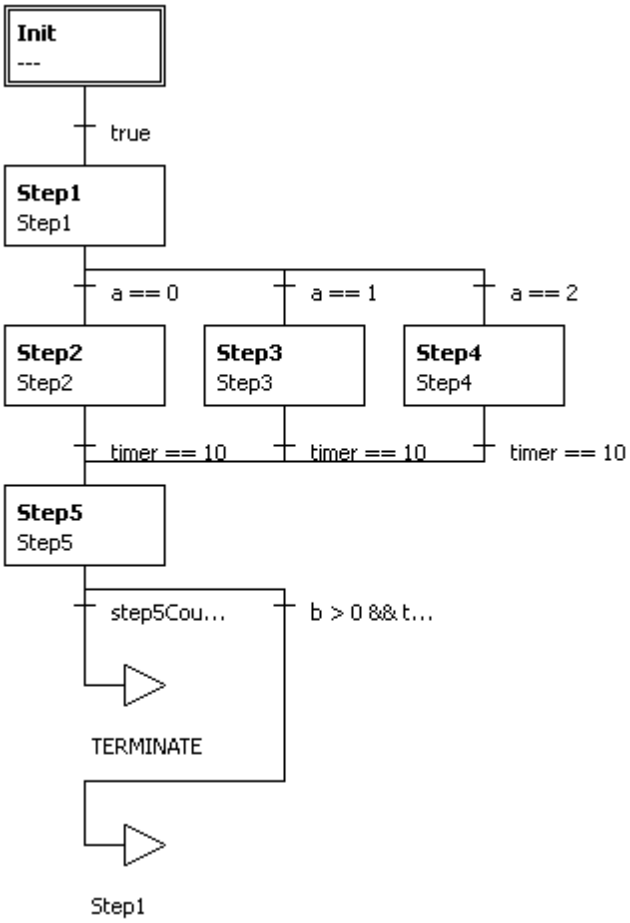


Figure 1 : Le programme séquentiel complet

Dans Step1, la valeur a est modifiée. Ceci se traduit par l'exécution, au cours de chaque cycle du programme, de l'une des étapes Step2, Step3 et Step4. Step5 compare les résultats fournis par les étapes précédentes Au bout du sixième cycle de Step5, le programme s'arrête. Sinon, il se poursuit avec Step1.

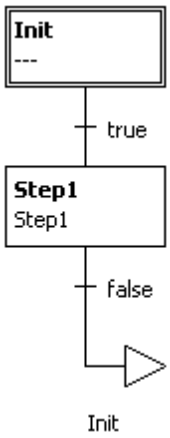
Créer un projet

Créez un nouveau projet en sélectionnant

- dans le menu Fichier▶la commande Nouveau
- en utilisant le raccourci clavier Ctrl+N ou
- en cliquant dans la barre d'outils sur l'icône de création d'un projet .

Le programme principal comporte les deux étapes Init et Step1.

Exemples



Créer une variable globale

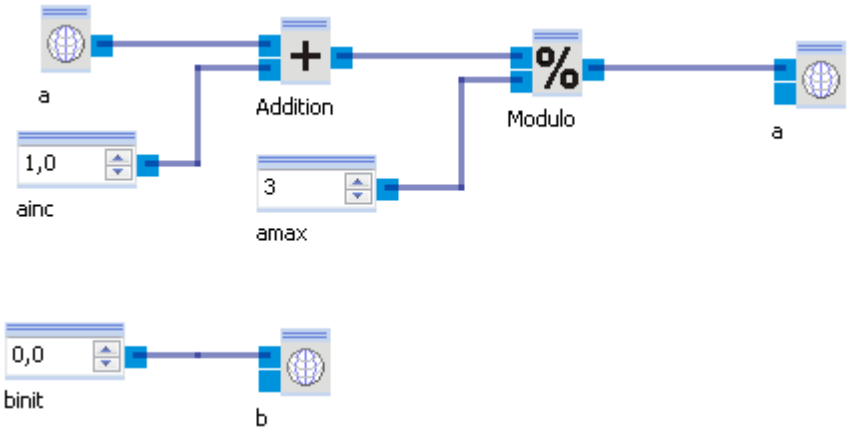
Créez d'abord les Variables globales 15 suivantes :

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

Affectez à la variable "a" la valeur initiale -1. Toutes les autres variables ont pour valeur initiale 0.

Programmer Step1

Dans ce sous-programme, la variable globale "a" est incrémentée de 1. Pour que la valeur de "a" évolue dans les limites de 0 à 2 inclus, on calcule "a" modulo 3 puis on réinscrit le résultat dans "a". La valeur "b" est simplement mise à zéro.

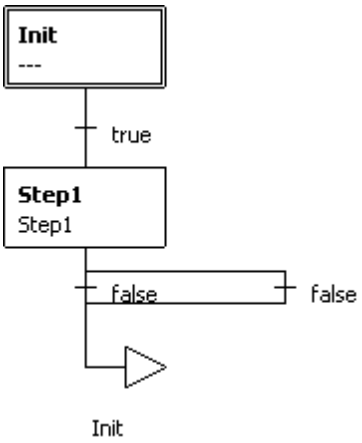


Créer les étapes Step2, Step3 et Step4

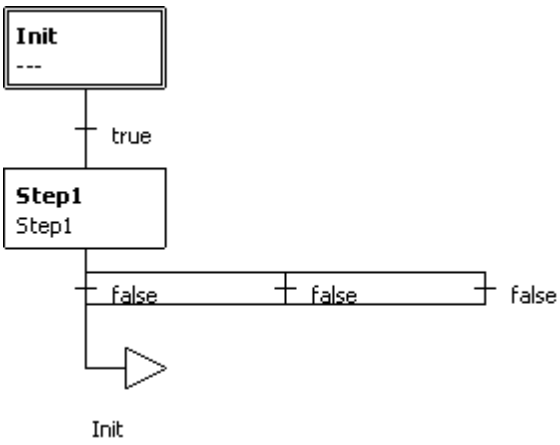
On crée à présent les étapes juxtaposées dans une branche alternative. Sélectionnez pour ce faire la condition de transition sous Step1.

La condition de transition est sélectionnée lorsqu'elle est entourée d'une ligne pointillée.

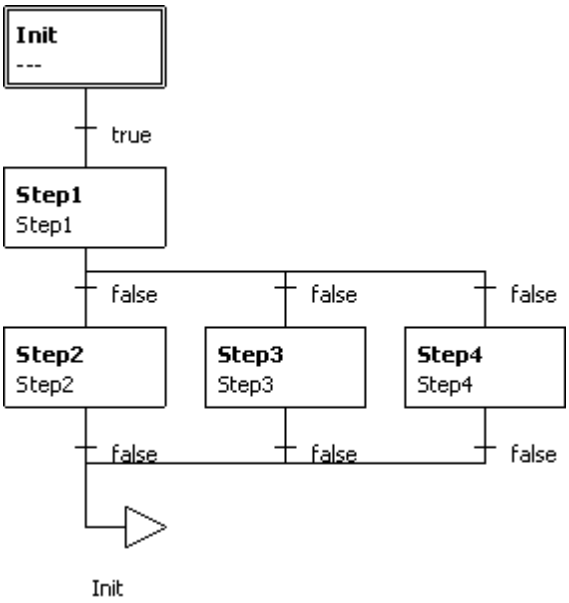
Cliquez à présent sur le symbole d'ajout d'une branche alternative à droite .



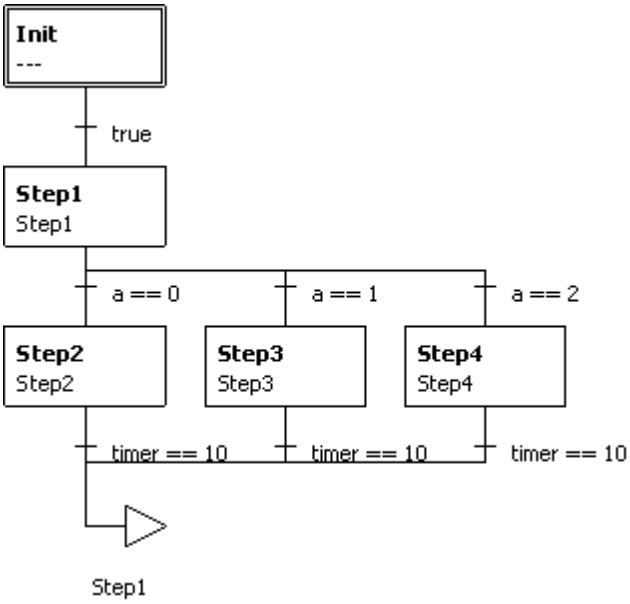
Complétez la branche générée en sélectionnant la condition de transition et en ajoutant une nouvelle branche alternative à droite Cliquez sur le symbole



Créez à présent, dans les trois branches alternatives, trois étapes que vous nommerez Step2, Step3 et Step4. Sélectionnez pour ce faire la condition d'entrée d'une branche puis clique sur le symbole d'ajout d'une étape après . Affectez ensuite au étapes créées des sous-programmes du même nom. Effectuez pour ce faire un double clic sur l'étape voulue et entrez dans la boîte de dialogue qui s'ouvre le nom du sous-programme. Vous pouvez également, avec le bouton "Nouveau sous-programme", créer un sous-programme et l'affecter ensuite à l'étape voulue.



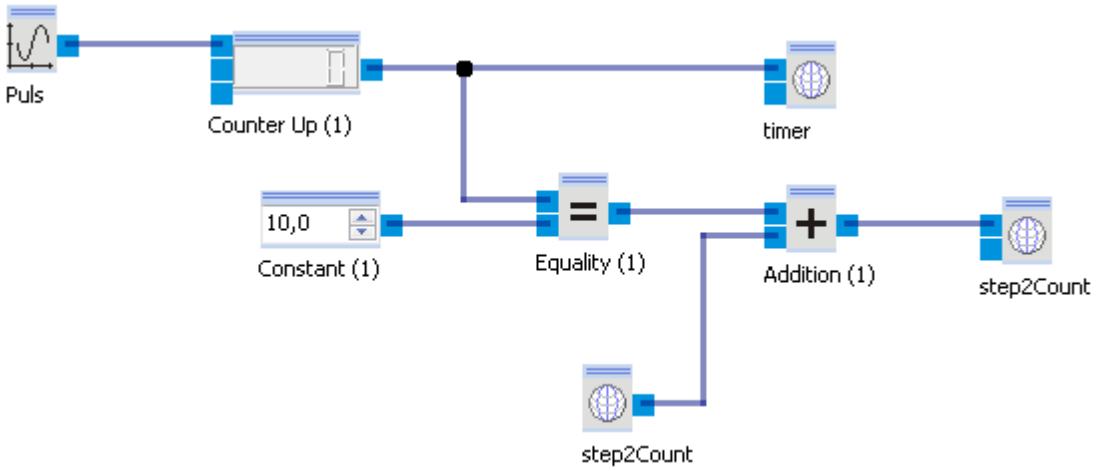
Les conditions d'entrée et de sortie des trois branches alternatives sont actuellement à l'état faux. Modifiez les conditions d'entrée pour obtenir a == 0, a == 1 et a == 2. Prenez comme condition de sortie pour toutes les branches timer == 10. Modifiez ensuite le saut final de Init à Step1.



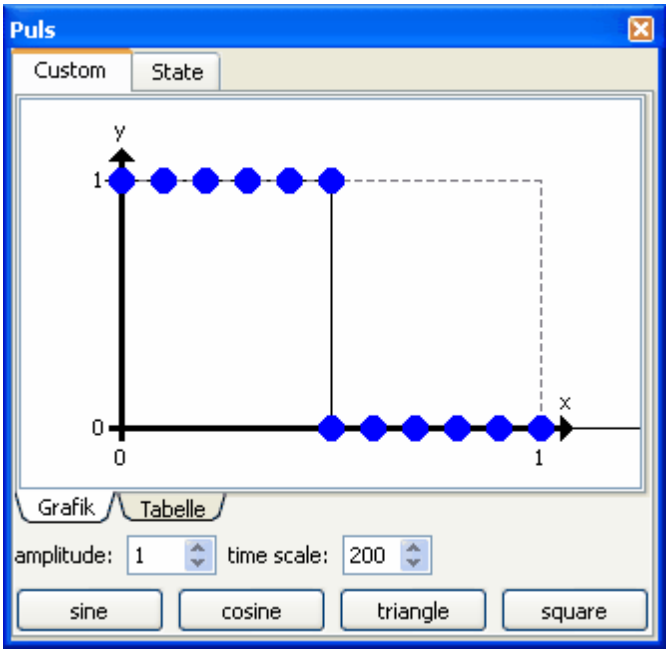
Si vous démarrez maintenant le programme principal, le programme s'arrête dans Step2 parce que "a" est à 0 au premier cycle et que la variable globale "timer" n'est pas modifiée.

Programmer les étapes Step2, Step3 et Step4

Les sous-programmes affectés aux étapes Step2 à Step4 sont momentanément encore vides. Nous allons voir dans ce qui suit comment créer le sous-programme Step2.



Le générateur d'ondes arbitraire génère toutes les 200 ms une impulsion d'une largeur de 100 ms et d'une hauteur égale à 1. La figure ci-après visualise le paramétrage du générateur d'ondes arbitraire.



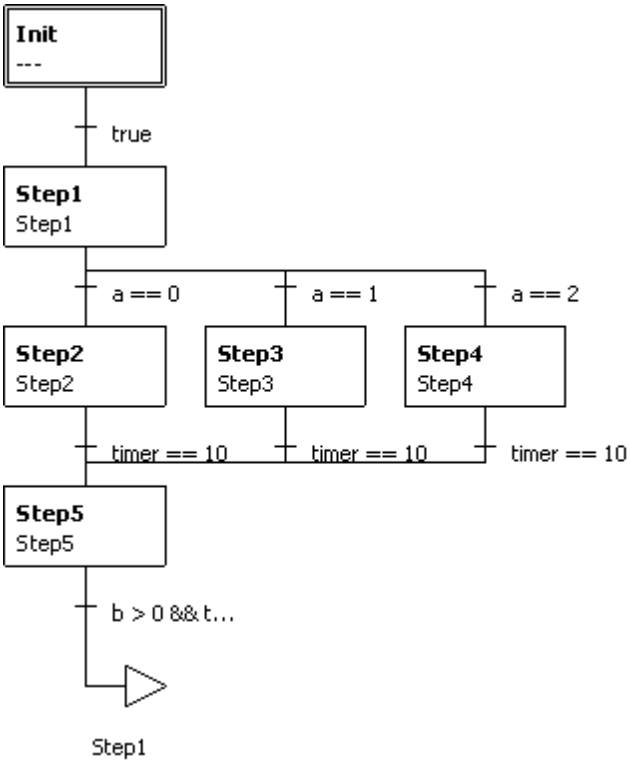
En d'autre termes, une transition de 0 à 1 intervient toutes les 200 ms. A chaque transition, le compteur incrémente son résultat de 1. Au bout de 2 s, sa valeur est donc de 10. Lorsque la valeur du compteur est égale à 10, la valeur momentanée de step2Count est additionnée au résultat de la comparaison de la constante et de la valeur du compteur. Tant que la comparaison est sanctionnée par faux, la valeur ajoutée est 0. Dès que la comparaison est sanctionnée par vrai, la valeur ajoutée est 1. Pour clore chaque étape de calcul du sous-programme, on vérifie la condition de transition suivante du programme principal. Si la variable globale "timer" a atteint la valeur 10, le sous-programme est terminé.

Les sous-programmes des étapes Step3 et Step4 possèdent une structure équivalente. Sélectionnez tout dans le sous-programme Step2 (Ctrl+A) et copiez le tout dans Step3 et Step4. La seule différence réside dans le fait que les valeurs sont lues et écrites dans les variables step3count et step4count.

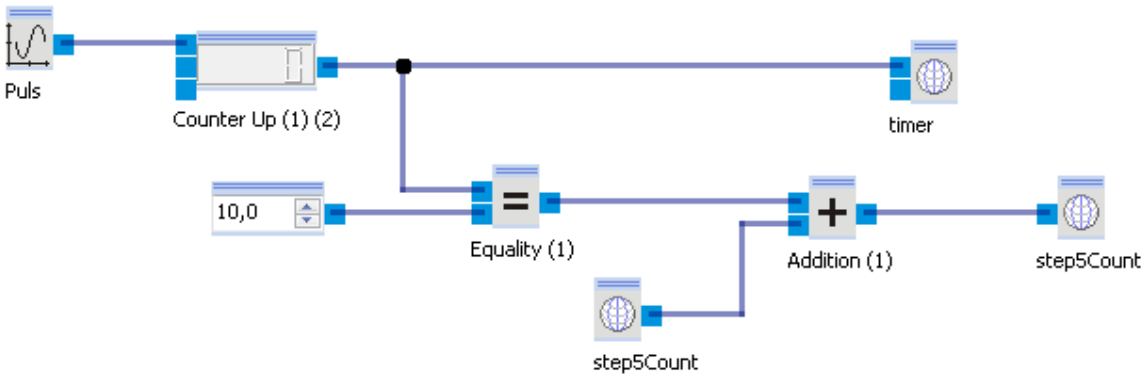
Si vous démarrez le programme principal par un clic sur l'icône , les étapes Step2, Step3 et Step4 sont exécutée cycliquement durant respectivement 2 s.

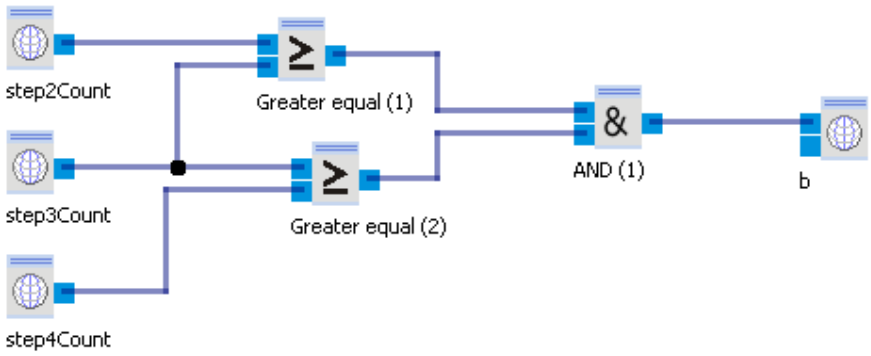
Créer et programmer Step5

Pour créer une étape après la branche alternative, sélectionnez le saut final puis cliquez sur l'icône de création d'une étape avant . Créez un sous-programme nommé Step5 et affectez-le à l'étape Step5 qui vient d'être créée. Modifiez les conditions de transition de Step5 pour obtenir b›0 && timer == 10.



Le sous-programme Step5 ressemble aux programmes Step2 à Step4. Copiez Step2 dans Step5 et modifiez step2count en step5count. Dans ce sous-programme, les variables globales "timer" et "step5count" sont positionnées mais on vérifie également la réalisation de la condition step2count ›= step3count ›= step4count. Si c'est le cas, la variable globale "b" est mise à 1. Sinon "b" est à 0. La condition doit cependant toujours être vraie en cas d'exécution correcte du programme, étant donné que les étapes Step2, Step3 et Step4 sont exécutée successivement parce que Step1 incrémente la variable globale "a" de 1 à chaque cycle.





Si vous démarrez le programme principal maintenant, Step5 reste activé durant 2 s, à condition que "b" soit supérieur à 0.

Créer l'arrêt du programme et le saut à Step1

On souhaite que le programme s'arrête lorsque la variable globale "step5count" atteint la valeur 6. Pour y parvenir, on ajoute une branche alternative sous Step5. Sélectionnez la condition de transition sous Step5 puis cliquez sur l'icône d'ajout d'une branche alternative à gauche . Sélectionnez la condition de transition de la nouvelle branche (la condition est actuellement à l'état faux) puis cliquez sur l'icône d'ajout d'un saut . Modifiez la condition de transition de step5count == 6 et sélectionnez "TERMINATE" comme destination du saut.

Le programme principal est maintenant identique à ce que vous avez vu au début.

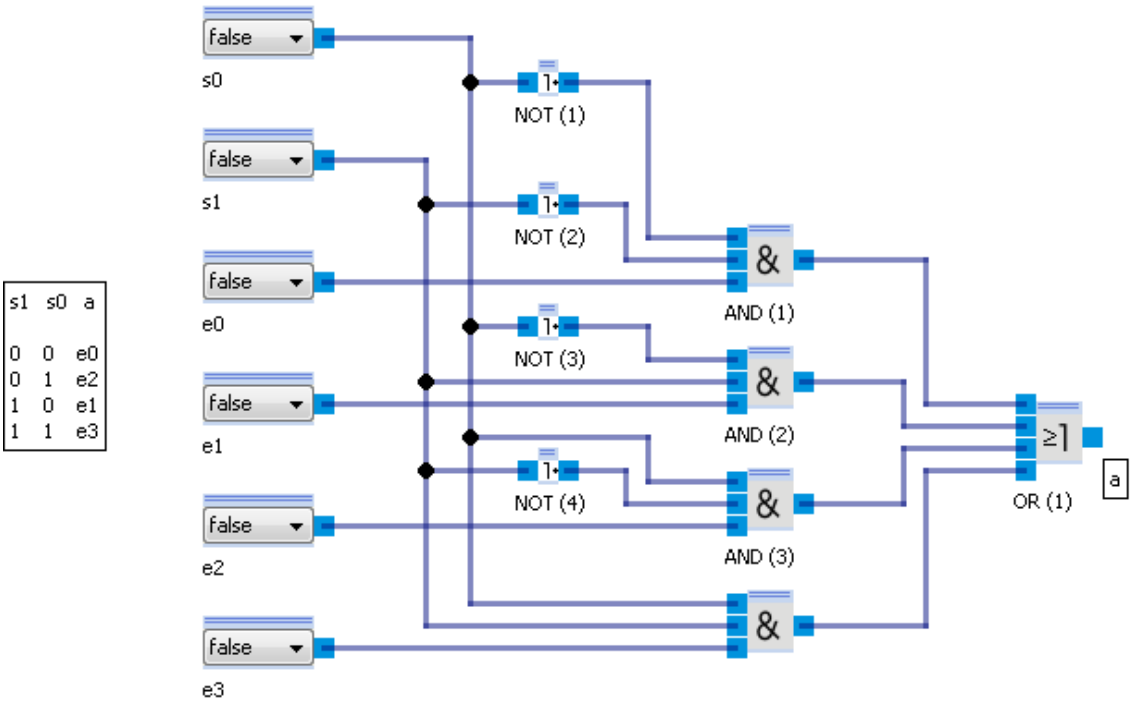
La branche alternative avec le saut vers TERMINATE doit d'ailleurs figurer à gauche de la branche avec la condition b>0 && timer == 10 parce que les conditions initiales d'une branche alternative sont traitées de gauche à droite. Au cours des 6 premier cycles, la condition step5count == 6 n'est pas remplie, de sorte que la condition de la seconde branche est vérifiée.

L'exécution du programme principal dure 24 s.

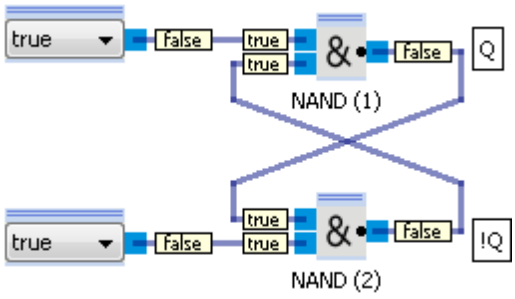
4.2 Logique

Ce chapitre décrit la réalisation de circuits électriques avec des blocs logiques.

4.2.1 Multiplexeur



4.2.2 Bascule



5 Bibliothèque de blocs de fonction

Les programmes de commande réalisés sous Robotino® View se composent de blocs de fonction interconnectés. Les blocs de fonction qui se trouvent dans la bibliothèque de blocs de fonction, 9 peuvent être insérés dans un programme par glisser-déplacer.

Les blocs de fonction sont classés en différentes catégories. Cliquez avec le bouton gauche de la souris sur le nom de catégorie pour faire afficher tous les blocs de fonction de cette même catégorie. Les catégories suivantes sont disponibles :

Nom	Description
Logique 35	Composants similaires aux blocs logiques de l'électronique
Mathématique 54	Opérations et fonctions mathématiques
Calcul vectoriel 69	Calcul avec des vecteurs bidimensionnels
Affichage 76	Blocs de fonction de visualisation
Traitement d'images 79	Blocs de fonction pour le traitement d'images (de caméra)
Générateur 92	Génération de signaux
Filtre 95	Lissage de signaux
Navigation 96	Blocs de fonction pour éditer et parcourir des itinéraires
Dispositifs d'entrée 116	Blocs de fonction d'interaction de l'utilisateur avec le programme de commande
Échange de données 119	Blocs de fonction pour l'échange de données avec des programmes externes
Blocs de fonction personnels 170	Tutoriels de conception de blocs de fonction personnels

5.1 Logique

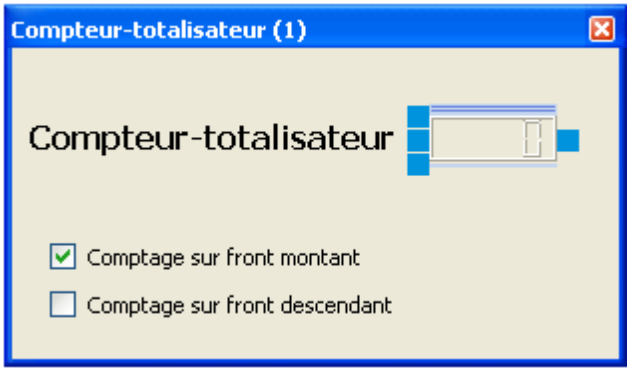
La catégorie Logique contient des composants similaires aux blocs logiques de l'électronique.

5.1.1 Compteur-totalisateur

Le compteur compte le nombre de changements d'état d'une entrée.

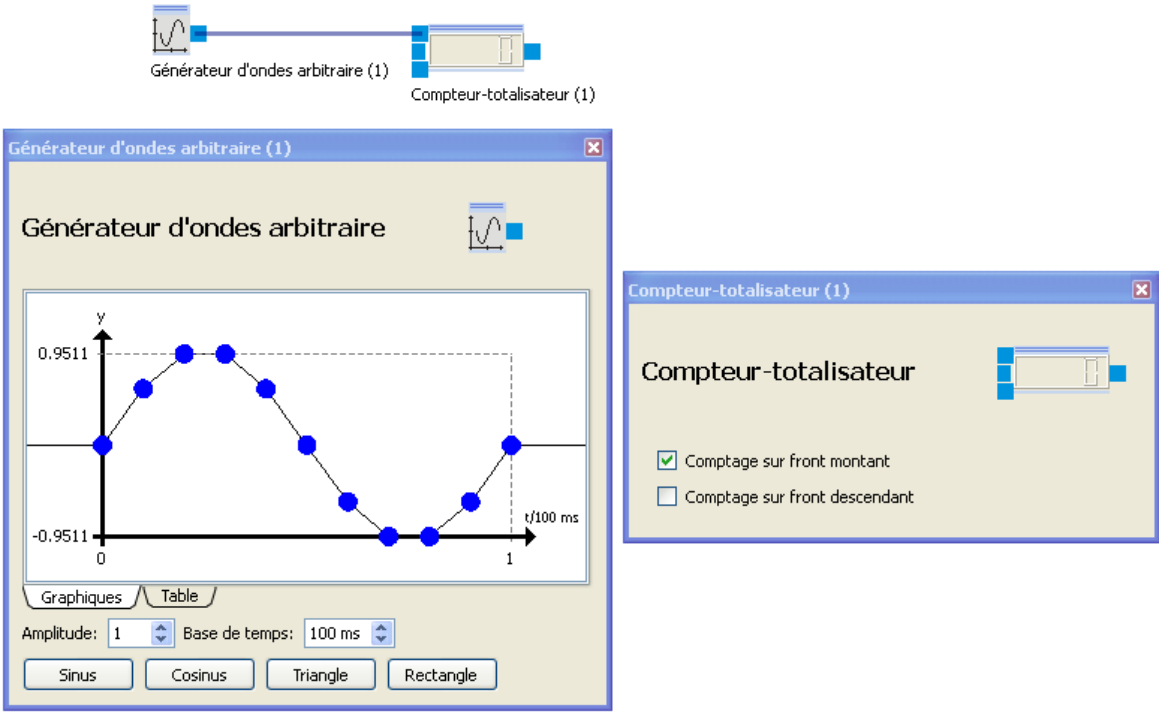
Entrées	Type	Standard	Description
Entrée	bool	false	Entrée du compteur Le compteur est incrémenté à chaque transition de faux (false) à vrai (true) et/ou à chaque transition de vrai à faux.
Valeur initiale	int	0	Le comptage s'effectue à partir de cette valeur au démarrage du programme et lorsque Réinitialiser est vrai (true).
Réinitialiser	bool	false	Si vrai (true), le compteur est remis à la valeur initiale.
Sorties			
Sortie	int		Valeur comptée

5.1.1.1 Dialogue



Comptage sur front montant	Incrémentation de la valeur du compteur de 1 si l'entrée est à l'état faux (false) à l'instant t et si elle est à l'état vrai (true) à l'instant t+1.
Comptage sur front descendant	Incrémentation de la valeur du compteur de 1 si l'entrée est à l'état vrai (true) à l'instant t et si elle est à l'état faux (false) à l'instant t+1.

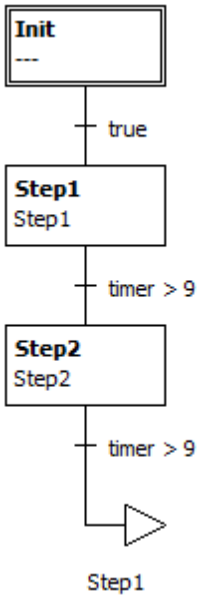
5.1.1.2 Exemple



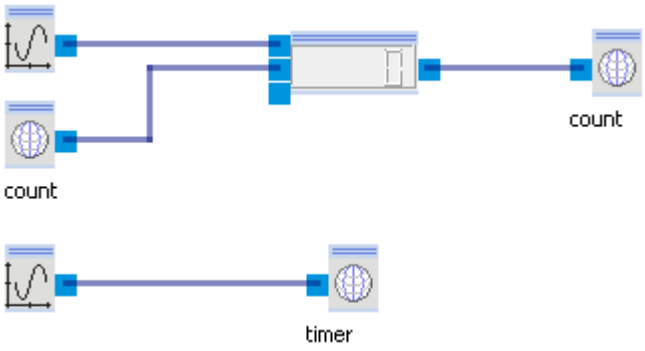
Le générateur d'ondes arbitraire génère une sinusoïde d'amplitude 2 et de fréquence 1 Hz. La sortie du générateur est de type float. Les valeurs numériques inférieures ou égales à 0 sont assimilées à faux (false). Les valeurs numériques supérieures à 0 sont assimilées à vrai (true) (voir Transtypage 19). Le compteur compte sur front montant, c.-à-d. lors de la transition de faux à vrai. Cet événement survient exactement une fois par seconde à chaque début de sinusoïde. La valeur du compteur correspond donc au temps en secondes, écoulé depuis le démarrage du programme.

Dans l'exemple ci-après, l'entrée est utilisée comme valeur initiale pour procéder à un comptage se poursuivant au-delà des limites du sous-programme. Le programme principal exécute les sous-programmes Step1 et Step2 successivement. A la fin du sous-programme Step2, il reprend avec Step1.

Programme principal

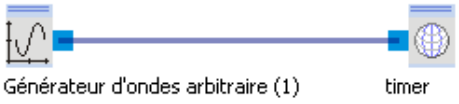


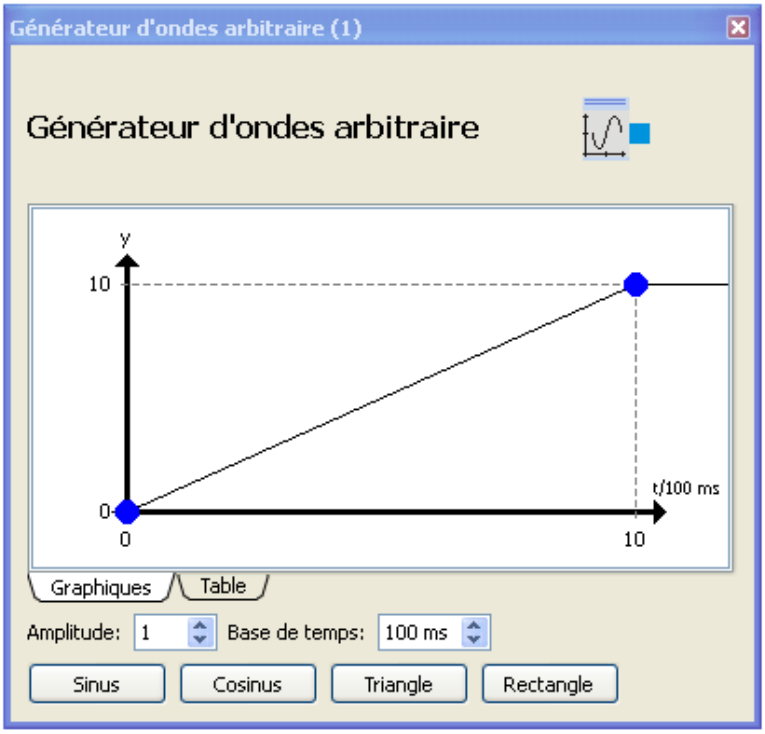
Step1



Le compteur inscrit son résultat dans la variable globale "count". Au redémarrage du sous-programme Step1 la valeur de count1 est utilisée comme valeur initiale. Step1 reste actif jusqu'à ce que le générateur d'ondes arbitraire produise une valeur supérieure à 9. C'est le cas au bout de 10 s.

Step2





Step2 est actif durant 10 s, sinon il n'a pas d'autre fonction.

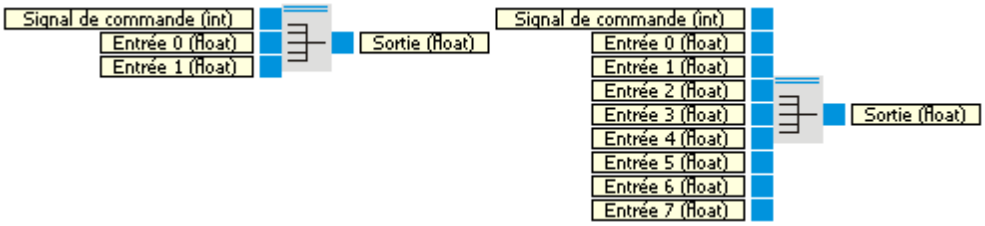
5.1.2 Décompteur

Le décompteur est identique au compteur-totalisateur 35 sauf qu'il n'ajoute pas 1 à la valeur du compteur mais qu'il retranche 1 lorsque survient un événement.

5.1.2.1 Dialogue

Voir dialogue du compteur-totalisateur 36, sauf que la valeur n'est pas ajoutée mais retranchée.

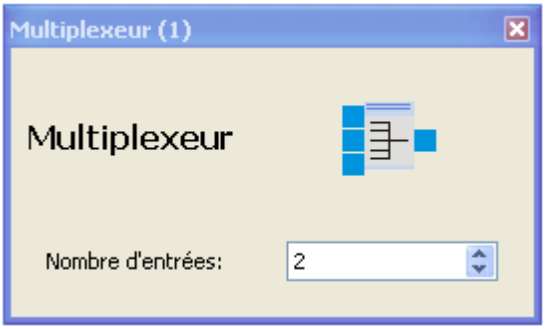
5.1.3 Multiplexeur



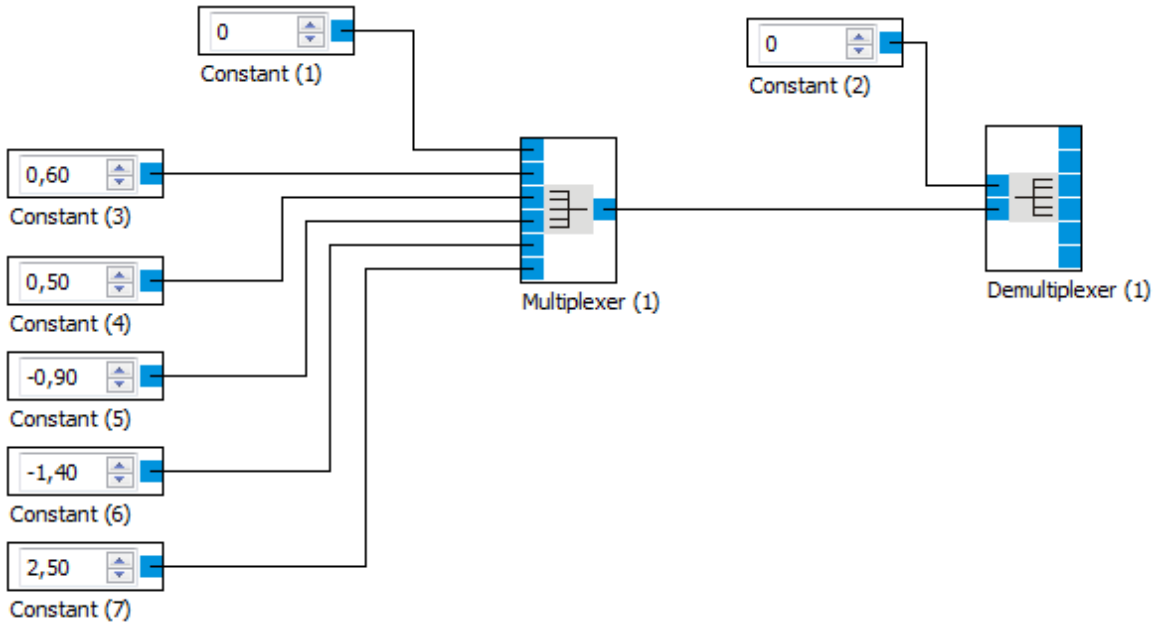
Le multiplexeur associe une sortie à une entrée sélectionnable.

Entrées	**Type**	**Standard**	**Description**
Signal de réglage	int	0	Définit l'entrée à laquelle la sortie est reliée. Si le signal de commande est inférieur à 0 ou supérieur ou égal au nombre des entrées, la sortie est à 0.
Entrée 0	float	0	La valeur de l'entrée 0 est appliquée à la sortie si le signal de commande est égal à 0.
...			
Entrée 9	float	0	La valeur de l'entrée 9 est appliquée à la sortie si le signal de commande est égal à 9.
Sorties			
Sortie	float		La valeur d'une entrée. 0, si le signale de commande est inférieur à 0 ou supérieur ou égal au nombre des entrées.

5.1.3.1 Dialogue

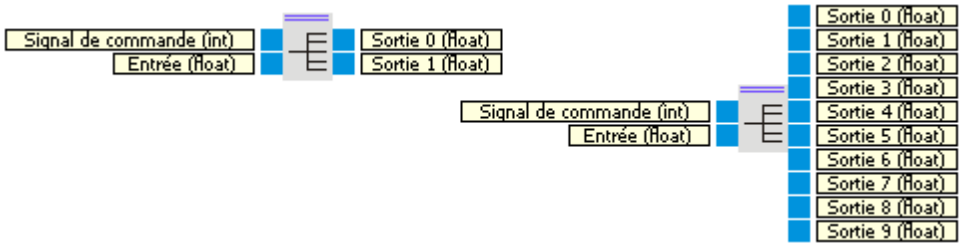


5.1.3.2 Exemple



Voir aussi Exemples▸Logique▸Multiplexeur 34

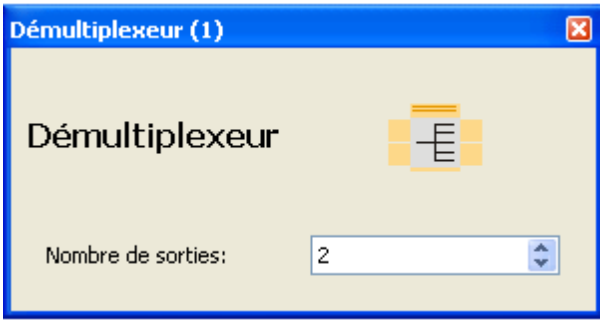
5.1.4 Démultiplexeur



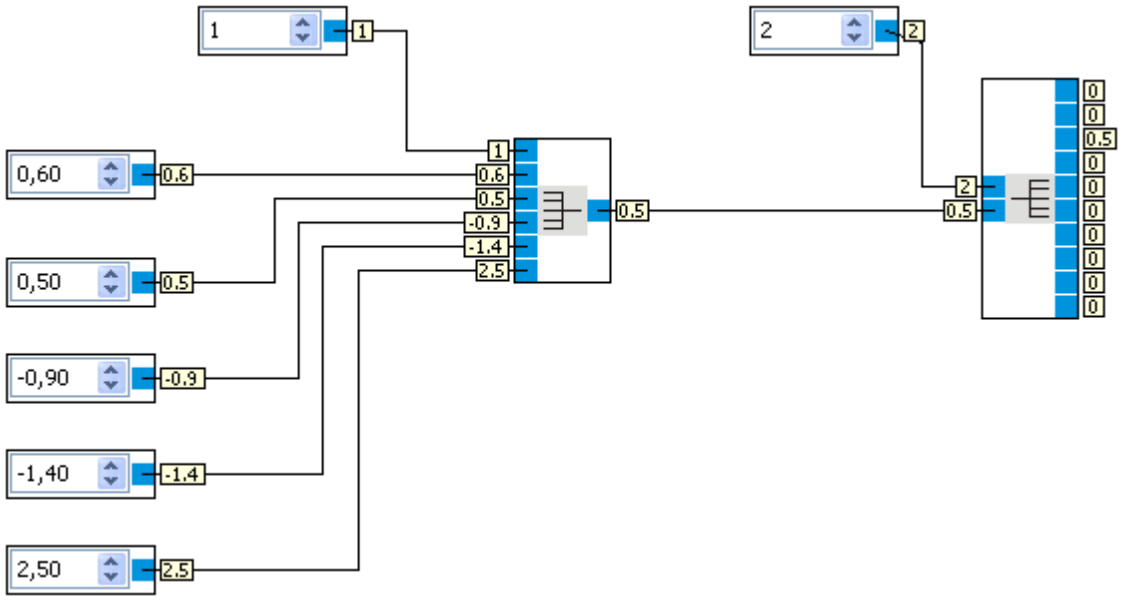
Le démultiplexeur répartit un signal d'entrée sur, au maximum, 10 sorties sélectionnables.

Entrées	Type	Standard	Description
Signal de réglage	int	0	Définit la sortie à laquelle l'entrée est reliée. Si le signal de commande est inférieur à 0 ou supérieur ou égal au nombre des sorties, toutes les sorties sont remises à 0.
Entrée	float	0	La valeur de l'entrée est appliquée à "sortie + signal de commande".
Sorties			
Sortie 0	float		Valeur de l'entrée si le signal de commande est égal à 0.
...			
Sortie 9	float		Valeur de l'entrée si le signal de commande est égal à 9.

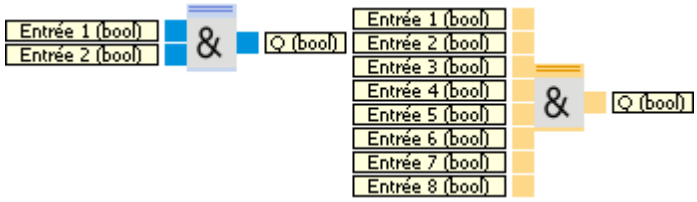
5.1.4.1 Dialogue



5.1.4.2 Exemple



5.1.5 AND

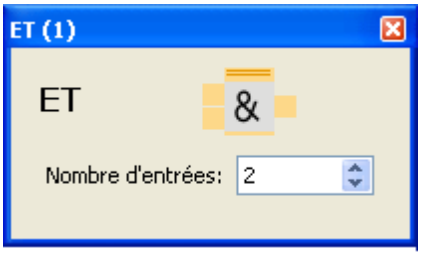


La sortie du AND ne passe à l'état vrai (true) que si toutes les entrées sont à l'état vrai. Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

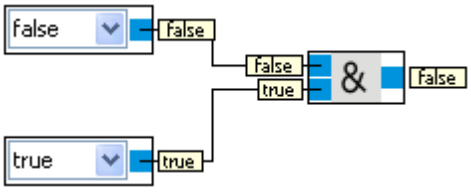
Entrées	Type	Standard	Description
Entrée 1	bool	true	
...			
Entrée 8	bool	true	
Sorties			
Q	bool		voir table de vérité

Entrées								
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

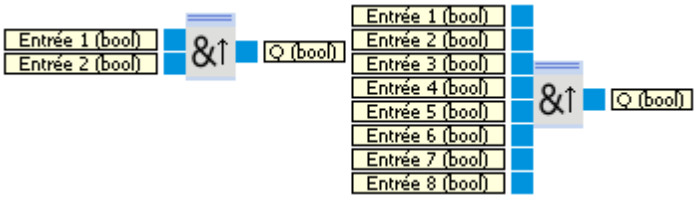
5.1.5.1 Dialogue



5.1.5.2 Exemple



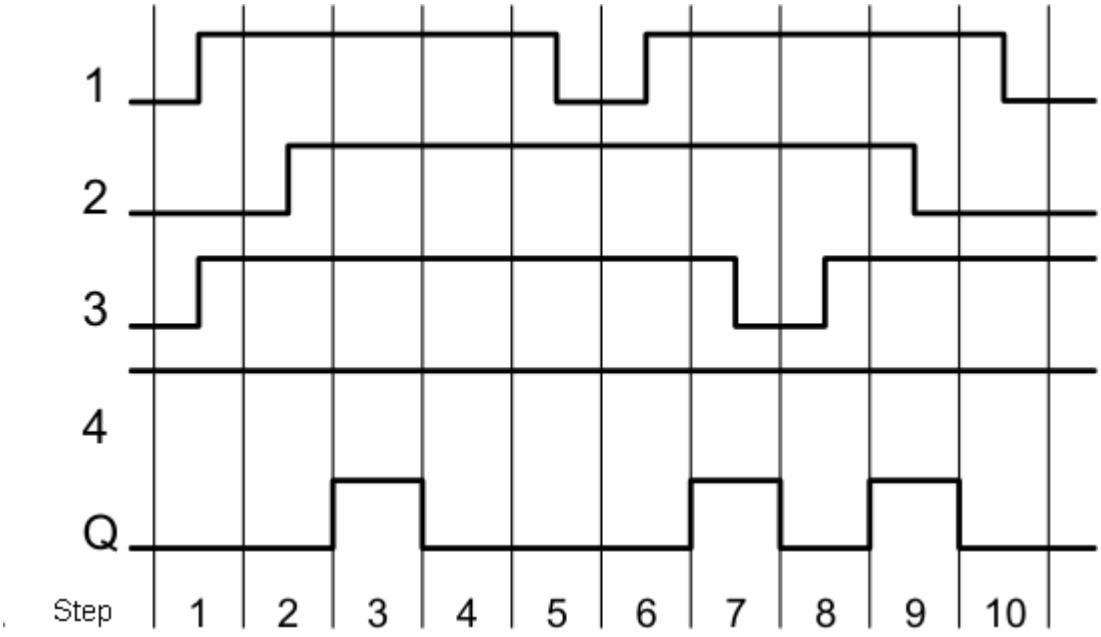
5.1.6 AND FL



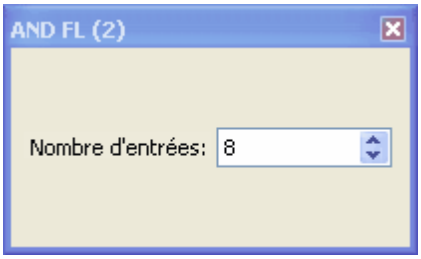
La sortie du AND avec détection de fronts ne passe à l'état vrai (true) que si toutes les entrées sont à l'état vrai et si au cycle précédent au moins une entrée était à l'état faux (false). Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	Type	Standard	Description
Entrée 1	bool	true	
...			
Entrée 8	bool	true	
Sorties			
Q	bool		Diagramme des temps

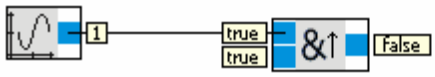
Diagramme des temps du AND avec détection de front à 4 entrées.

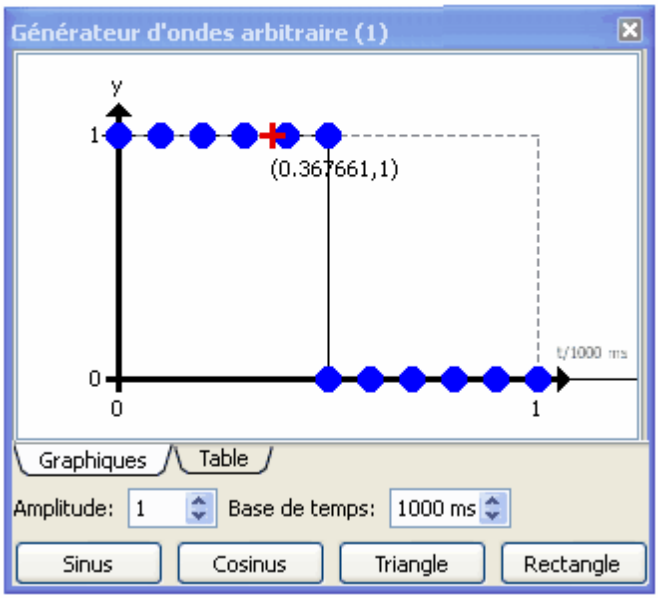


5.1.6.1 Dialogue



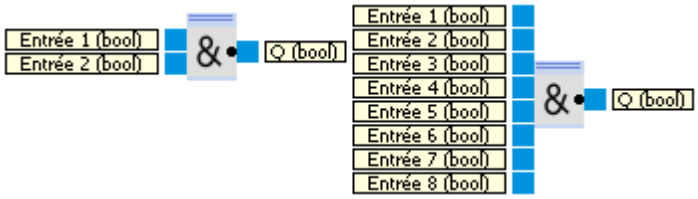
5.1.6.2 Exemple





A chaque transition du générateur de 0 à 1, la sortie du AND à détection de fronts est à l'état vrai (true) durant un cycle.

5.1.7 NAND

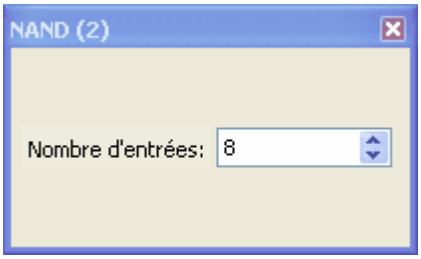


La sortie du NAND ne passe à l'état faux (false) que si toutes les entrées sont à l'état vrai (true). Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	Type	Standard	Description
Entrée 1	bool	true	
...			
Entrée 8	bool	true	
Sorties			
Q	bool		voir table de vérité

Entrées								
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

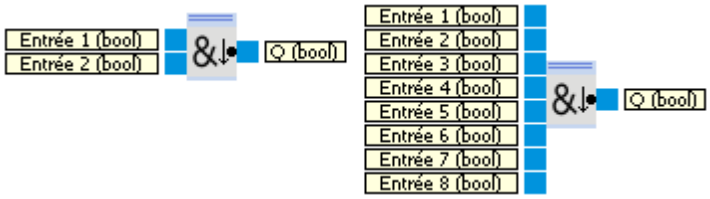
5.1.7.1 Dialogue



5.1.7.2 Exemple

voir exemples▸Logique▸Bascule 34

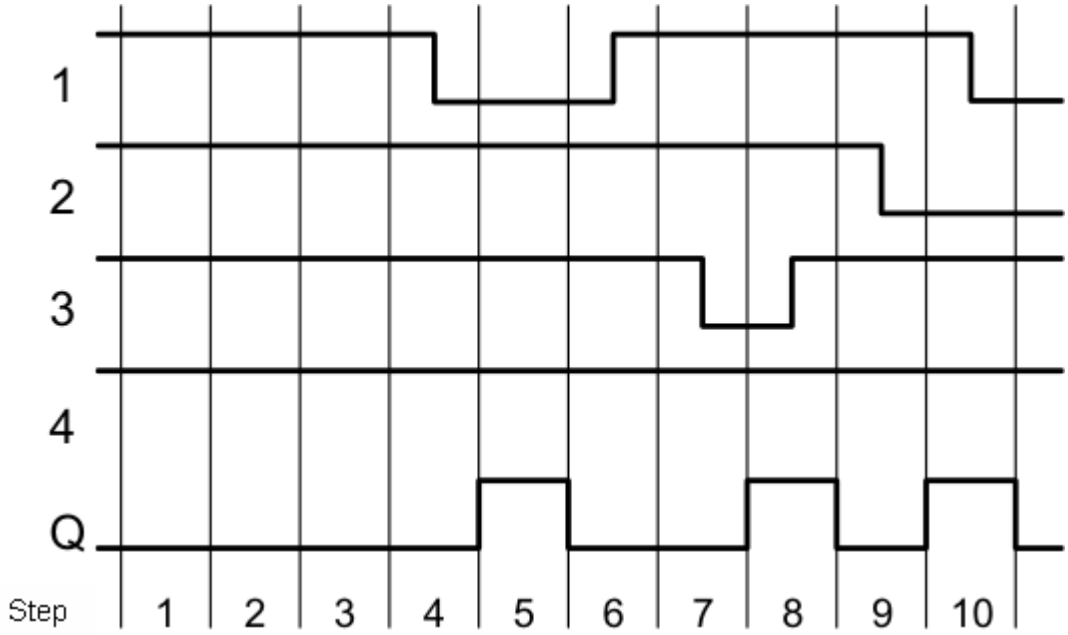
5.1.8 NAND_FL



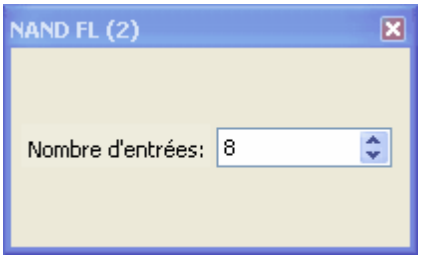
La sortie du NAND avec détection de fronts ne passe à l'état vrai (true) que si toutes les entrées sont à l'état faux (false) et si au cycle précédent toutes les entrées étaient à l'état vrai (true). Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	bool	true	
...			
Entrée 8	bool	true	
Sorties			
Q	bool		voir Diagramme des temps

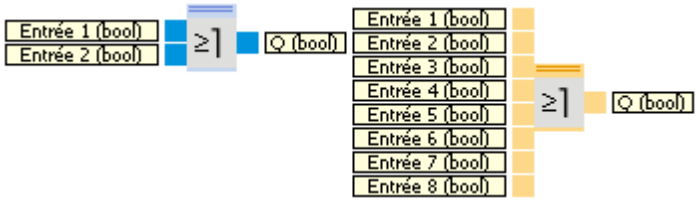
Diagramme des temps du NAND avec détection de fronts à 4 entrées.



5.1.8.1 Dialogue



5.1.9 OR



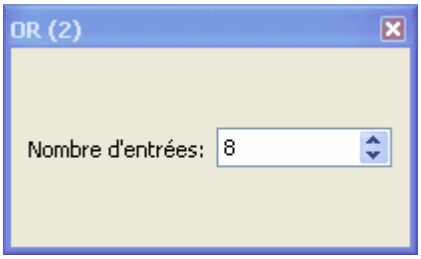
La sortie du OR passe à l'état vrai (true) si au moins une entrée est à l'état vrai. Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	bool	false	
...			
Entrée 8	bool	false	
Sorties			
Q	bool		voir table de vérité

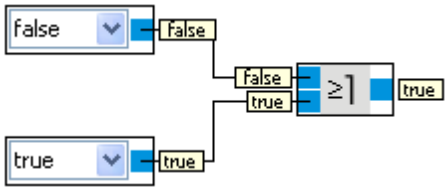
Entrées								
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Q**
0	0	0	0	0	0	0	0	0
							1	1
						1		1
						1	1	1
					1			1
					1		1	1
					1	1		1
					1	1	1	1
				1				1
				1			1	1

				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

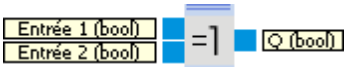
5.1.9.1 Dialogue



5.1.9.2 Exemple



5.1.10 XOR

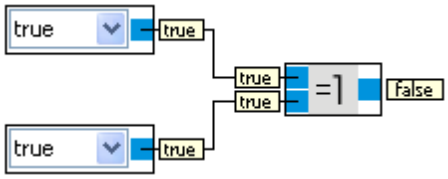


La sortie du XOR est mise à l'état vrai (true) si l'état des sorties n'est pas identique. Pour la conversion de valeurs numériques en booléen voir Tanstypage 19.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	bool	false	
Entrée 2	bool	false	
Sorties			
Q	bool		voir table de vérité

Entrées		
1	**2**	**Q**
0	0	0
0	1	1
1	0	1
1	1	0

5.1.10.1 Exemple



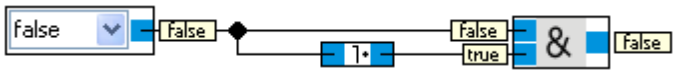
5.1.11 NOT

La sortie du NOT ne passe à l'état vrai (true) que si l'entrée est à l'état faux (false). Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	Type	Standard	Description
Entrée	bool	false	
Sorties			
Q	bool		voir table de vérité

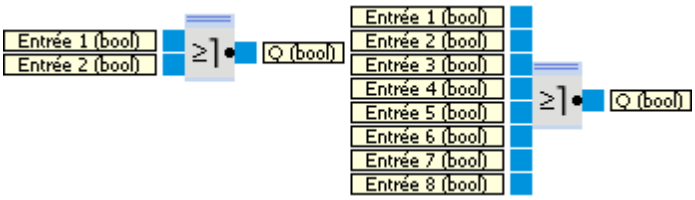
Entrées	
1	**Q**
0	1
1	0

5.1.11.1 Exemple



L'exemple illustre une particularité du bloc de fonction NOT. A la différence des autres blocs de fonction, les données ne sont pas affichées à côté des connexions. Le bloc de fonction NOT peut donc de ce fait être positionné très près des autres blocs de fonction sans que les affichages de données se chevauchent ou se recouvrent.

5.1.12 NOR



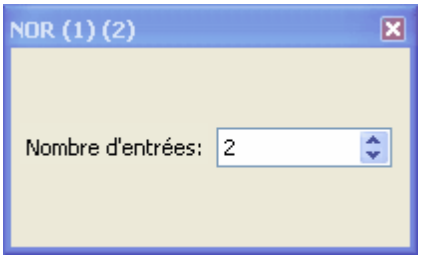
La sortie du NOR ne passe à l'état vrai (true) que si toutes les entrées sont à l'état faux (false). Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	bool	false	
...			
Entrée 8	bool	false	
Sorties			
Q	bool		voir table de vérité

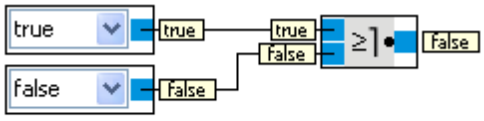
Entrées								
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Q**
0	0	0	0	0	0	0	0	1
							1	0
						1		0
						1	1	0
					1			0
					1		1	0
					1	1		0
					1	1	1	0
				1				0

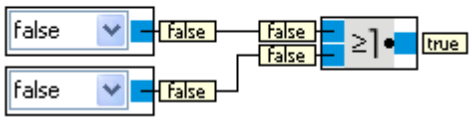
				1			1	0
				1		1		0
				1		1	1	0
				1	1			0
				1	1		1	0
				1	1	1		0
				1	1	1	1	0
			1					0
1	1	1	1	1	1	1	1	0

5.1.12.1 Dialogue



5.1.12.2 Exemple





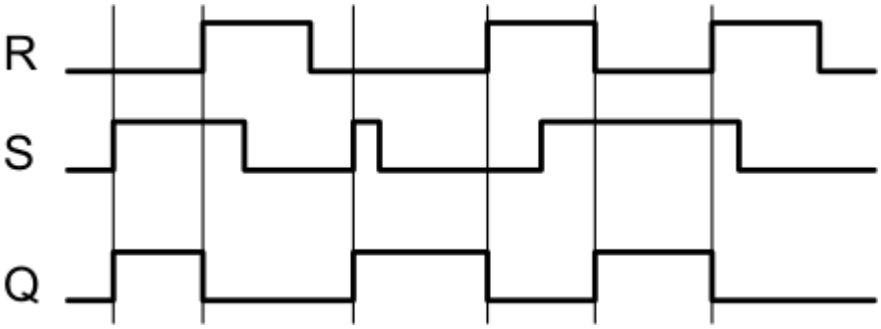
5.1.13 Relais à automaintien

La sortie Q est mise à 1 par l'entrée S. Elle est remise à 0 par l'entrée R. Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

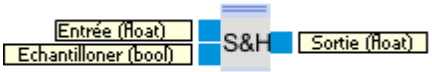
Entrées	Type	Standard	Description

S	bool	false	La sortie Q est mise à l'état vrai (true) par l'entrée S.
R	bool	false	La sortie Q est remise à 0 par l'entrée R. Si S et R sont simultanément à 1, la sortie est remise à zéro.
Par	bool	false	Rémanence : faux (false) : pas de rémanence vrai (true) : enregistrement rémanent de l'état actuel (indépendamment de S ou de R)
Sorties			
Q	bool		Q est mise à l'état vrai (true) par S et le reste jusqu'à ce que l'entrée R passe à vrai (true).

Diagramme des temps



5.1.14 Échantillonneur-bloqueur



Si l'échantillonnage est mis à l'état faux, le signal d'entrée peut être maintenu à la valeur momentanée aussi longtemps que l'on veut. Pour la conversion de valeurs numériques en booléen voir Transtypage 19.

Entrées	**Type**	**Standard**	**Description**
Entrée	float	0	Signal d'entrée
Échantillonnage	bool	false	Si vrai, le signal d'entrée est émis en sortie. Si faux, la valeur momentanée est "figée" en sortie.
Sorties			
Sortie	float	0	Valeur du signal d'entrée lorsque **l'échantillonnage** est passé de vrai à faux.

5.2 Mathématique

Cette catégorie contient des opérations mathématiques élémentaires.

5.2.1 Opérations arithmétiques

5.2.1.1 Modulo

Modulo (du lat. modulus, ablatif modulo : "par mesure" ou aussi "avec mesure", pluriel moduli), symbole de formule mathématique, représenté dans de nombreux langages de programmation par %, est une fonction mathématique indiquant le reste de la division de deux entiers. (Source : http://fr.wikipedia.org/wiki/Modulo)

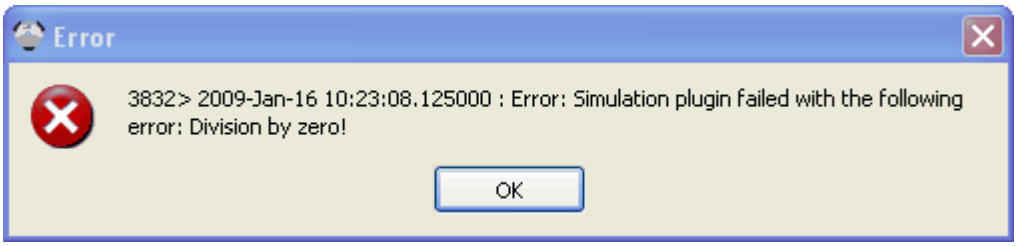
Entrées	Type	Standard	Description
Dividende	int	0	
Diviseur	int	1	
Sorties			
Reste	int		Dividende mod diviseur

5.2.1.2 Division

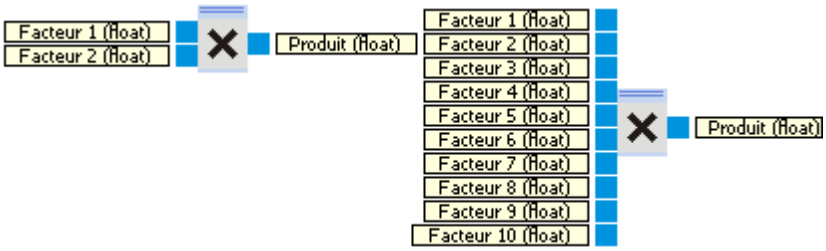
Calcule le quotient à partir du dividende et du diviseur. Voir aussi http://fr.wikipedia.org/wiki/Division.

Entrées	Type	Standard	Description
Dividende	float	0	
Diviseur	float	1	
Sorties			
Quotient	float		Dividende divisé par diviseur

Si le dividende est différent de 0 et si le diviseur est égal à 0, la simulation est arrêtée avec affichage du message d'erreur suivant :



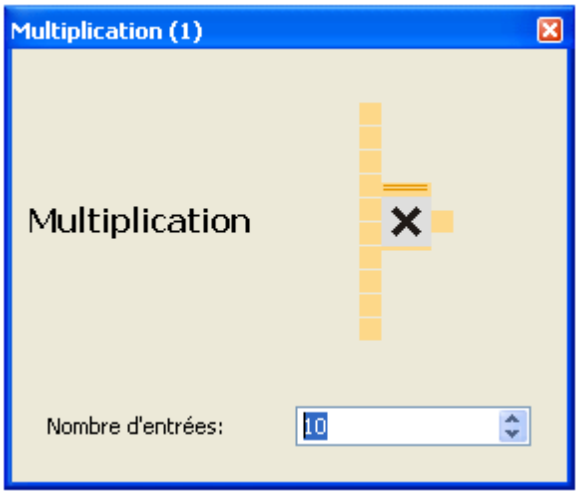
5.2.1.3 Multiplication



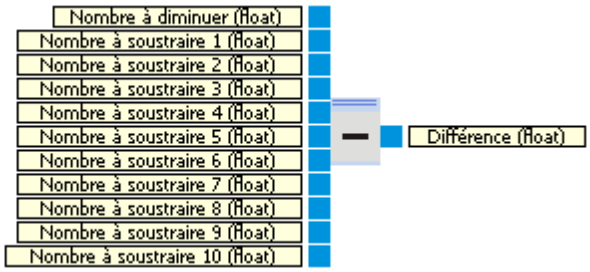
Le module de multiplication multiplie jusqu'à 10 valeurs d'entrée. Voir aussi http://fr.wikipedia.org/wiki/Multiplication.

Entrées	**Type**	**Standard**	**Description**
Facteur 1	float	1	
...			
Facteur 10	float	1	
Sorties			
Produit	float		"Facteur 1" * "Facteur 2" * ... * "Facteur 10"

5.2.1.3.1 Dialogue



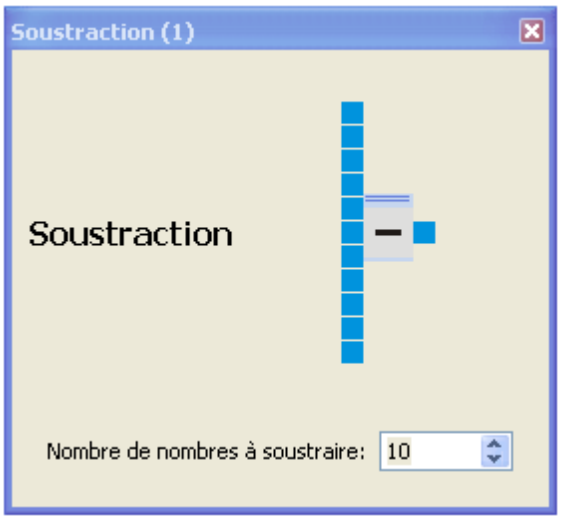
5.2.1.4 Soustraction



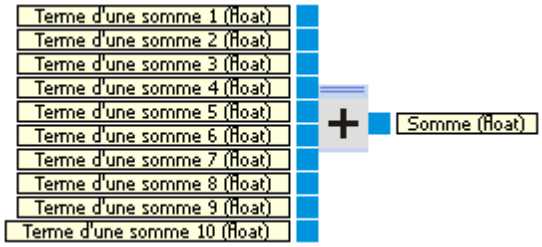
Le module de soustraction déduit jusqu'à 10 valeurs du premier opérande de la soustraction. Voir aussi http://fr.wikipedia.org/wiki/Soustraction.

Entrées	**Type**	**Standard**	**Description**
Nombre à diminuer	float	0	
Nombre à soustraire 1	float	0	
...			
Nombre à soustraire 10	float	0	
Sorties			
Différence	float		Nombre à diminuer - "Nombre à soustraire 1" - "Nombre à soustraire 2" - ... "Nombre à soustraire 10"

5.2.1.4.1 Dialogue



5.2.1.5 Addition



Le module d'addition additionne jusqu'à 10 valeurs d'entrée. Voir aussi http://fr.wikipedia.org/wiki/Addition.

Entrées	**Type**	**Standard**	**Description**
Terme 1	float	0	
...			
Terme 10	float	0	
Sorties			
Total	float		"terme 1" + "terme 2" + ... + "terme 10"

5.2.1.5.1 Dialogue

5.2.2 Opérations de comparaison

5.2.2.1 Différent de

La sortie passe à vrai (true) si la valeur absolue de entrée 1 - entrée 2 est supérieure ou égale à epsilon, avec epsilon = 0,0000002384185792.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		fabs(entrée 1 - entrée 2) >= epsilon

5.2.2.2 Egal à

La sortie passe à vrai (true) si la valeur absolue de entrée 1 - entrée 2 est inférieure à epsilon, avec epsilon = 0,0000002384185792.

Entrées	Type	Standard	Description
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		fabs(entrée 1 - entrée 2) ‹ epsilon

5.2.2.3 Inférieur ou égal à

La sortie passe à l'état vrai (true) si l'entrée 1 est inférieure ou égale à l'entrée 2

Entrées	Type	Standard	Description
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		"entrée 1" inférieure ou égale à "entrée 2"

5.2.2.4 Inférieur à

La sortie passe à l'état vrai (true) si l'entrée 1 est inférieure à l'entrée 2

Entrées	Type	Standard	Description
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		"entrée 1" inférieure à "entrée 2"

5.2.2.5 Supérieur ou égal à

La sortie passe à l'état vrai (true) si l'entrée 1 est supérieure ou égale à l'entrée 2

Entrées	Type	Standard	Description
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		"entrée 1" supérieure ou égale à "entrée 2"

5.2.2.6 Supérieur à

La sortie passe à l'état vrai (true) si l'entrée 1 est supérieure à l'entrée 2

Entrées	Type	Standard	Description
Entrée 1	float	0	
Entrée 2	float	0	
Sorties			
Sortie	bool		"entrée 1" supérieure à "entrée 2"

5.2.3 Fonctions

5.2.3.1 Montant

Forme la valeur absolue.

Entrées	Type	Standard	Description

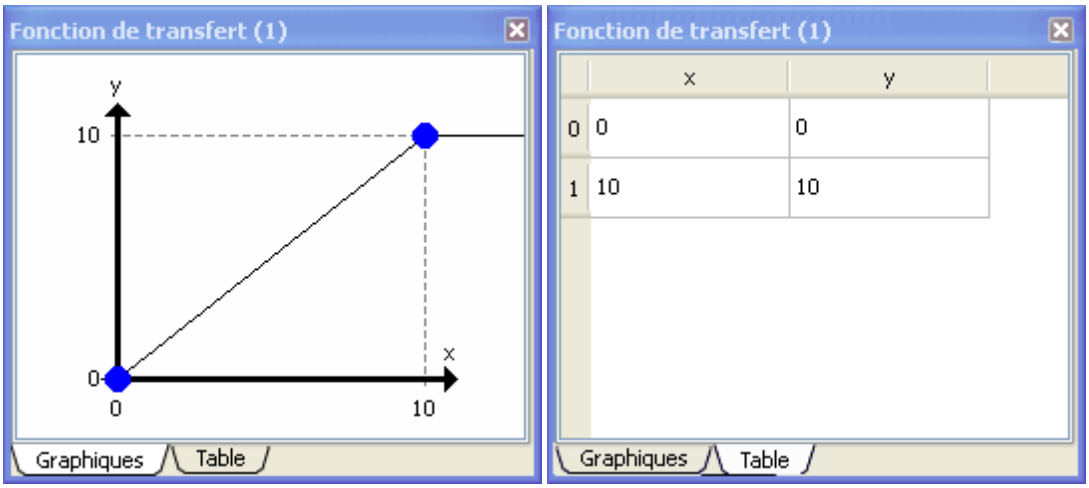
Entrée	float	0	
Sorties			
Sortie	float		abs(entrée)

5.2.3.2 Fonction de transfert

Ce bloc de fonction permet de transformer le signal d'entrée x en un signal de sortie y quelconque.

Entrées	Type	Standard	Description
x	float	0	
Sorties			
x	float		voir Dialogue 61

5.2.3.2.1 Dialogue



Cette boîte de dialogue permet de définir les points de contrôle de la fonction de transfert y(x). Les points de contrôle sont par défaut

p0 = (x0, y0) = (0, 0)

p1 = (x1, y1) = (10, 10)

. On obtient donc la transformation suivante :

y = y0 pour la plage x ‹= x0
y = x pour la plage x › x0 et x ‹= x1
y = y1 pour la plage x › x1

Zones limites

p0 = (x0, y0) est le premier point de contrôle.
pn = (xn, yn) est le dernier point de contrôle.

Si x inférieur à x0 : y = y0
Si x supérieur à x0 : y = yn

Fonction de transfert

Pour une liste de points de contrôle p0, p1, ... pn la transformation y(x) est :

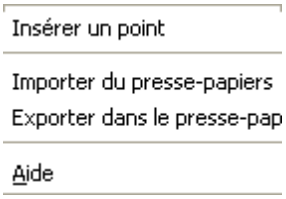
y = y0 pour x ‹= x0
y = (y1 - y0) / (x1 - x0) * (x - x0) + y0 pour x › x0 et x ‹= x1
y = (y2 - y1) / (x2 - x1) * (x - x1) + y1 pour x › x1 et x ‹= x2
...
y = yn für x › xn

Décalage des points

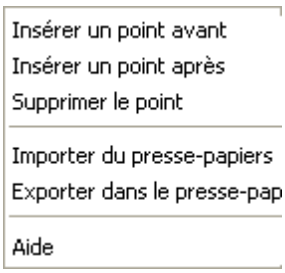
Les points de contrôle peuvent être décalés, ajoutés et supprimés. Pour décaler un point de contrôle, vous pouvez le déplacer dans le graphique avec la souris. Dans le tableau, vous pouvez éditer les valeur de x et y. La valeur x d'un point de contrôle ne peut ce faisant jamais être inférieure à la valeur x du point précédent et jamais supérieure à la valeur x du point suivant.

Ajouter des points

Dans le graphique vous pouvez ajouter un point à un emplacement quelconque à l'aide du menu contextuel qui s'ouvre sur un clic droit de la souris.



Dans le tableau, le menu contextuel s'ouvre lorsque vous cliquez à droite sur une cellule du tableau.



Vous pouvez indiquer ici que le point doit être inséré avant ou après le point actuel.

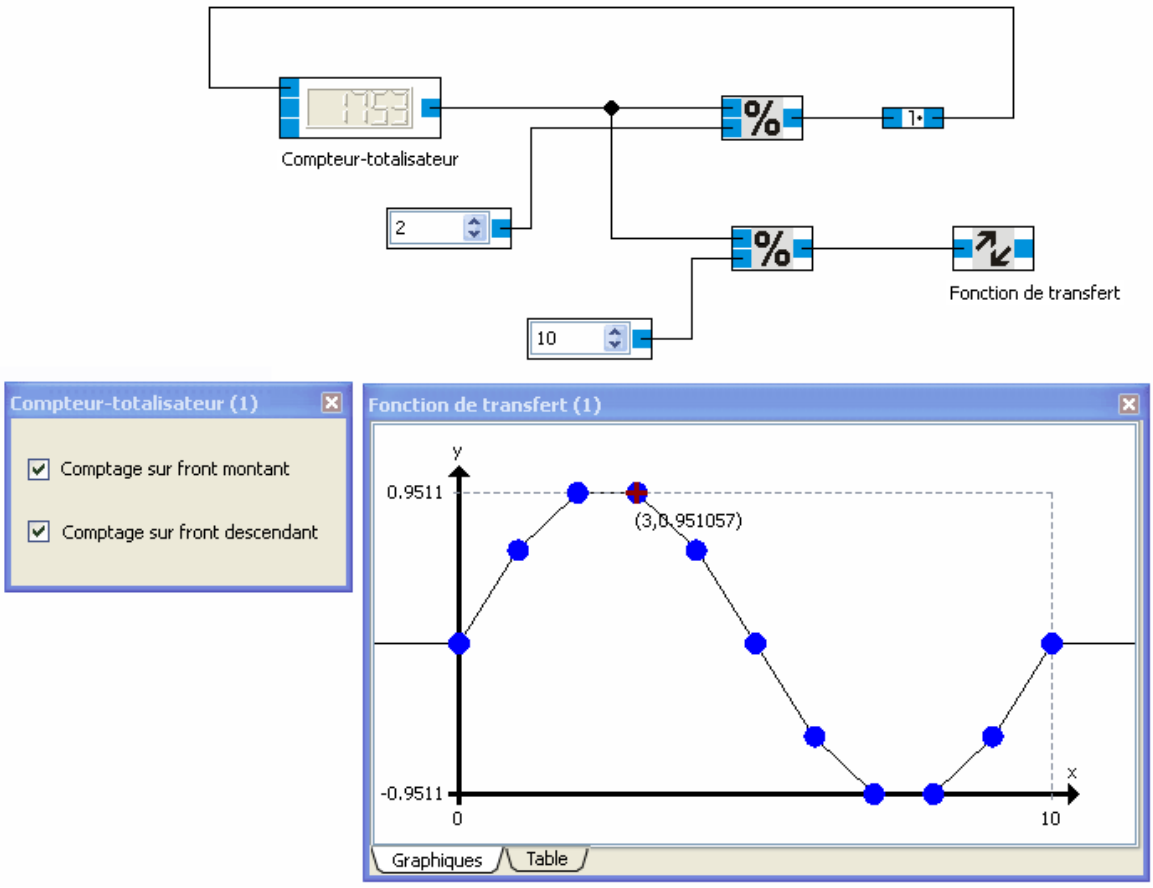
Supprimer des points

Dans le graphique ou dans le tableau, cliquez avec le bouton droit de la souris puis supprimez le point via le menu contextuel. S'il ne reste plus qu'un seul point de contrôle, la fonction de suppression du point est désactivée.

Importation/Exportation des points de contrôle

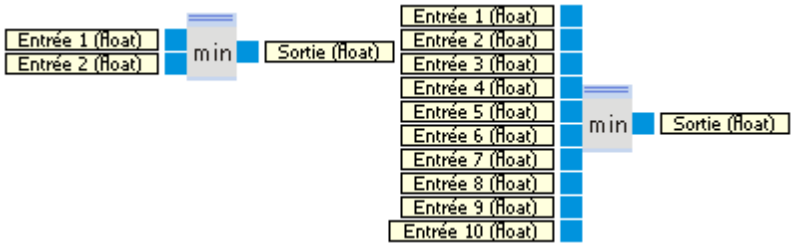
La fonction peut être exportée ou importée sous forme de liste séparée par des tabulateurs par le biais du presse-papier. Ceci permet d'échanger des données avec des programmes tels que Matlab ou Excel. Les fonctions sont disponibles via le menu contextuel aussi bien dans le graphique que dans le tableau.

5.2.3.2.2 Exemple



Le compteur est connecté de sorte qu'à chaque cycle de simulation, la valeur du compteur soit incrémentée de 1. La valeur du compteur est limitée par modulo à la plage [0, 10]. C'est l'entrée de la fonction de transfert. Les 10 points de contrôle sont reliés par un sinusoïde.

5.2.3.3 Minimum



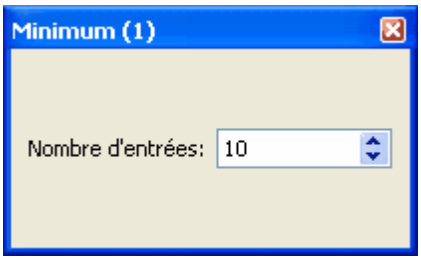
Délivre en sortie la plus petite valeur d'entrée.

Entrées	Type	Standard	Description
Entrée 1	float	1e+037	

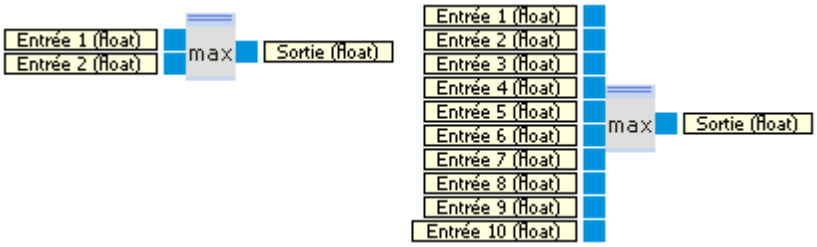
...			
Entrée 10	float	1e+037	
Sorties			
Sortie	float		min("Entrée 1", "Entrée 2", ... , "Entrée 10")

1e+037 = - (10 puissance 37)
plus grand nombre à virgule flottante possible

5.2.3.3.1 Dialogue



5.2.3.4 Maximum



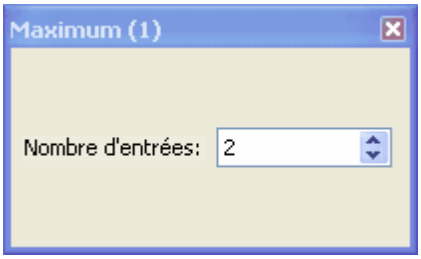
Délivre en sortie la plus grande valeur d'entrée.

Entrées	**Type**	**Standard**	**Description**
Entrée 1	float	-1e+037	
...			
Entrée 10	float	-1e+037	
Sorties			
Sortie	float		max("Entrée 1", "Entrée 2", ... , "Entrée 10")

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

plus petit nombre à virgule flottante possible

5.2.3.4.1 Dialogue

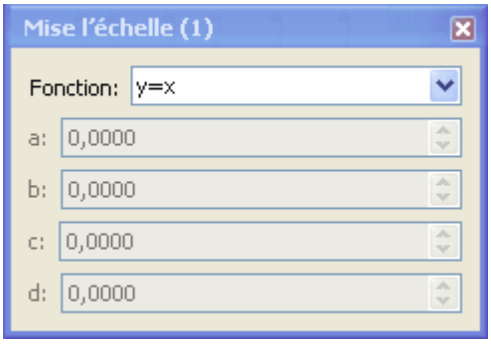


5.2.3.5 Mise à l'échelle

Permet une mise à l'échelle aisée de valeurs.

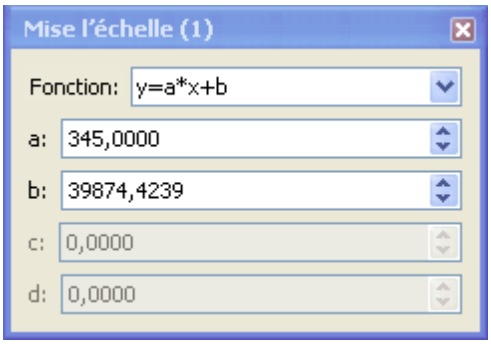
Entrées	Type	Standard	Description
x	float	0	
Sorties			
y	float		Voir dialogue

5.2.3.5.1 Dialogue



On peut dans un premier temps sélectionner une fonction. L'identité est sélectionnée par défaut, c.-à-d. que la valeur d'entrée x est reproduite inchangée à la sortie y.

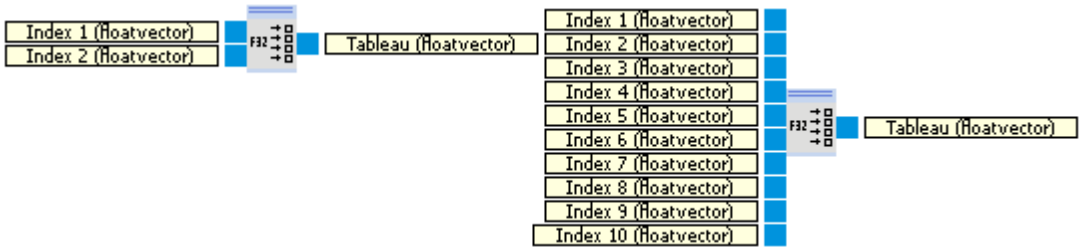
Selon la fonction, les paramètres a, b, c et d sont éditables. Si on sélectionne la fonction y=a*x+b, les champs a et b sont accessibles.



La fonction de transfert est dans ce cas y = 345 * x - 39874,4239

5.2.4 magnétiques

5.2.4.1 Composeur de float array

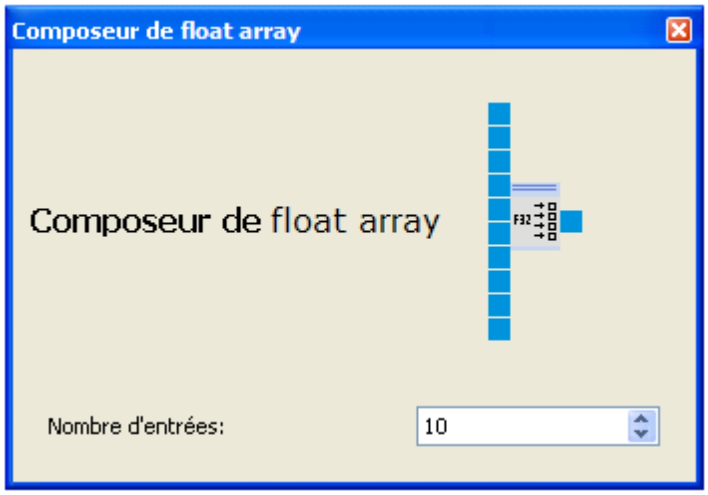


Un composeur de tableau de nombres à virgule flottante (float array) génère un tableau de nombres à virgule flottante à partir de 10 nombres ou tableaux de nombres à virgule flottante. Pour la conversion de nombres à virgule flottante en tableaux de nombres à virgule flottante voir Transtypage 19.

Entrées	**Type**	**Standard**	**Description**
Indice 1	float array	tableau vide	
...			
Indice 10	float array	tableau vide	
Sorties			

Tableau	float array	tableau vide	(Indice 1, ..., indice 10)

5.2.4.1.1 Dialogue

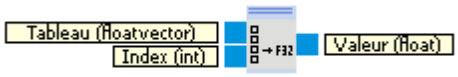


5.2.4.2 Décomposeur de float array

Le décomposeur de tableau de nombres à virgule flottante (float array) extrait un élément d'un tableau de nombres à virgule flottante.

Entrées	**Type**	**Standard**	**Description**
Tableau	float array	tableau vide	Le tableau à décomposer
Indice de départ	int	1	La valeur à la position **Indice de départ** du tableau à décomposer devient la première valeur du tableau décomposé.
Longueur	int	1	Le tableau décomposé est constitué de **longueur** valeurs commençant par la valeur à la position **Indice de départ** du tableau à décomposer.
Sorties			
Elément de tableau	float array	tableau vide	(**Tableau**[**Indice de départ**], ..., **Tableau**[**Indice de départ** + **Longueur** - 1])

5.2.4.3 Accès indexé au float array



Le module d'accès indexé permet d'accéder individuellement à des valeurs du tableau de nombres à virgule flottante (float array).

Entrées	Type	Standard	Description
Tableau	float array	tableau vide	Tableau de nombres à virgule flottant auquel on souhaite accéder.
Indice	int	1	L'indice de la valeur souhaitée.
Sorties			
Valeur	float	0	La valeur à la position **Indice**.

5.3 Calcul vectoriel

Cette catégorie contient les opérations de base du calcul vectoriel pour vecteurs bidimensionnels.

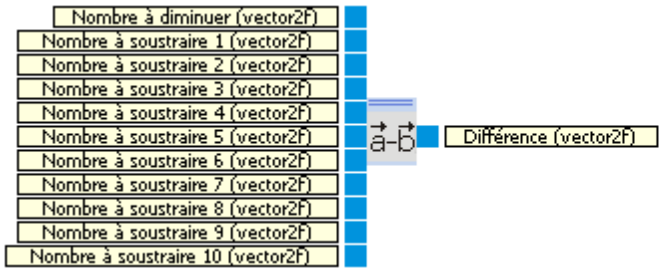
5.3.1 Opérations vectorielles

5.3.1.1 Produit scalaire

Forme le produit scalaire de deux vecteurs. Voir aussi http://fr.wikipedia.org/wiki/Produit_scalaire.

Entrées	Type	Standard	Description
Vecteur 1	vecteur2f	(0, 0)	
Vecteur 2	vecteur2f	(0, 0)	
Sorties			
Produit	float		

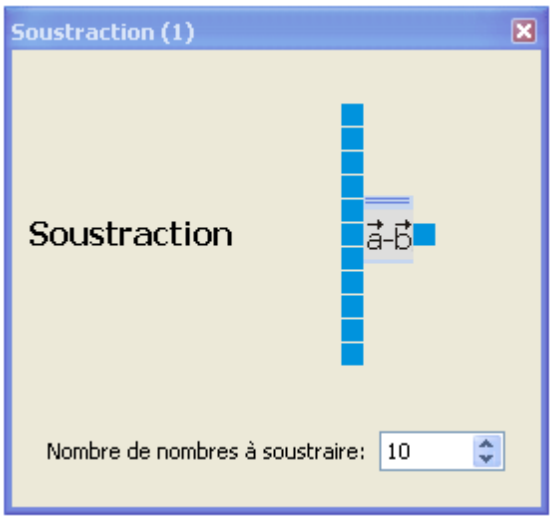
5.3.1.2 Soustraction



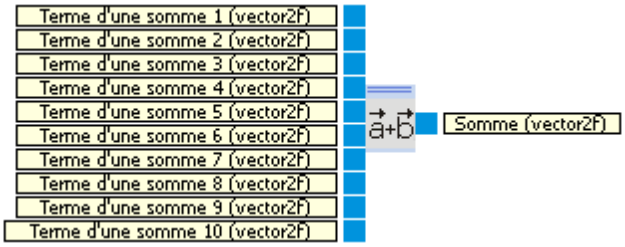
Soustraction d'au plus 10 vecteurs. Voir aussi http://fr.wikipedia.org/wiki/Addition#Addition_vectorielle.

Entrées	**Type**	**Standard**	**Description**
Nombre à diminuer	vecteur2f	(0, 0)	
Nombre à soustraire 1	vecteur2f	(0, 0)	
...			
Nombre à soustraire 10	vecteur2f	(0, 0)	
Sorties			
Différence	vecteur2f		Nombre à diminuer - "Nombre à soustraire 1" - "Nombre à soustraire 2" - ... "Nombre à soustraire 10"

5.3.1.2.1 Dialogue



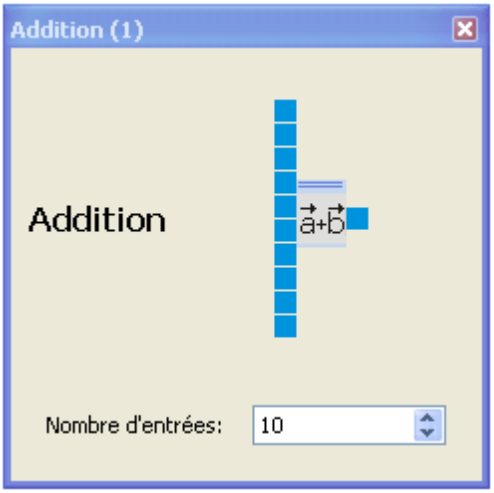
5.3.1.3 Addition



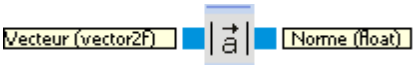
Addition d'au plus 10 vecteurs. Voir aussi http://fr.wikipedia.org/wiki/Addition#Addition_vectorielle.

Entrées	Type	Standard	Description
Terme 1	vecteur2f	(0, 0)	
...			
Terme 10	vecteur2f	(0, 0)	
Sorties			
Total	vecteur2f		"terme 1" + "terme 2" + ... + "terme 10"

5.3.1.3.1 Dialogue



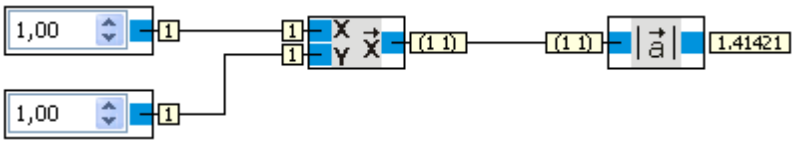
5.3.1.4 Norme



La norme (ou longueur) du vecteur. Voir aussi http://fr.wikipedia.org/wiki/Vecteur.

Entrées	Type	Standard	Description
Vecteur	vecteur2f	(0, 0)	
Sorties			
Norme	float		

5.3.1.4.1 Exemple



La longueur du vecteur (1, 1) est égale à racine de 2 (1.41421...).

5.3.2 Opérations élémentaires

5.3.2.1 Division

Division au niveau éléments.

Entrées	Type	Standard	Description
Vecteur	vecteur2f	(0, 0)	
Diviseur	float	1	
Outputs			

Résultat	vecteur2f		Vecteur = (x0, x1) Résultat = (x0 / diviseur, x1 / diviseur)

5.3.2.2 Soustraction

Soustraction au niveau éléments.

Entrées	**Type**	**Standard**	**Description**
Vecteur	vecteur2f	(0, 0)	
Nombre à diminuer	float	0	
Outputs			
Résultat	vecteur2f		Vecteur = (x0, x1) Résultat = (x0 - nombre à diminuer, x1 - nombre à diminuer)

5.3.2.3 Addition

Addition au niveau éléments.

Entrées	**Type**	**Standard**	**Description**
Vecteur	vecteur2f	(0, 0)	
Terme 1	float	0	
Outputs			
Résultat	vecteur2f		Vecteur = (x0, x1) Résultat = (terme + x0, terme + x1)

5.3.2.4 Multiplication

Multiplication au niveau éléments.

Entrées	Type	Standard	Description
Vecteur	vecteur2f	(0, 0)	
Facteur	float	1	
Outputs			
Résultat	vecteur2f		Vecteur = (x0, x1) Résultat = (facteur * x0, facteur * x1)

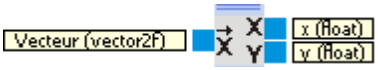
5.3.3 Transformations

5.3.3.1 Vecteur en polaires

Décomposition du vecteur dans ses composantes polaires.

Entrées	Type	Standard	Description
Vecteur	vecteur2f	(0, 0)	
Sorties			
Longueur	float		Longueur (norme) du **vecteur**.
Phi	float		Angle formé par le **vecteur** et l'axe x en degrés

5.3.3.2 Vecteur en cartésiennes



Décomposition du vecteur dans ses composantes cartésiennes.

Entrées	Type	Standard	Description
vecteur	vecteur2f	(0, 0)	
Sorties			
x	float		Composante x du **vecteur**.
y	float		Composante y du **vecteur**.

5.3.3.3 Polaires en vecteur

Génération d'un vecteur à partir de ses composantes polaires.

Entrées	Type	Standard	Description
Longueur	float	0	Longueur (norme) du vecteur.
Phi	float	0	Angle formé par le vecteur et l'axe x.
Sorties			
Vecteur	vecteur2f		Vecteur avec longueur **Longueur** et orientation **Phi**.

5.3.3.4 Cartésiennes en vecteur

Génération d'un vecteur à partir de ses composantes cartésiennes.

Entrées	Type	Standard	Description

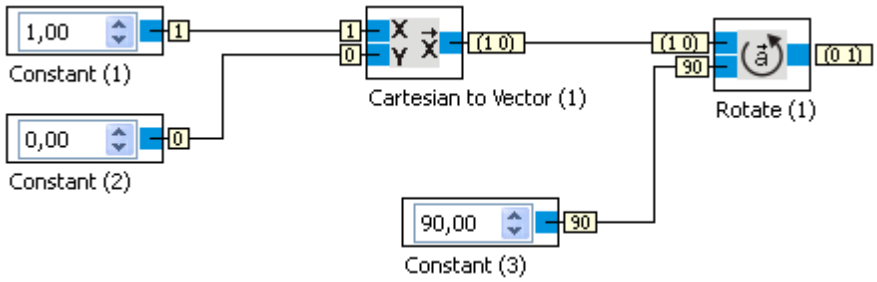
x	float	0	composante x.
y	float	0	composante y.
Sorties			
Vecteur	vecteur2f		Vecteur**x, y**).

5.3.3.5 Rotation

Rotation du vecteur en fonction de la valeur prédéfinie en degrés.

Entrées	**Type**	**Standard**	**Description**
Vecteur	vecteur2f	(0, 0)	
Phi	float	0	Angle de rotation
Sorties			
Résultat	vecteur2f		**Vecteur** tourné de **Phi** .

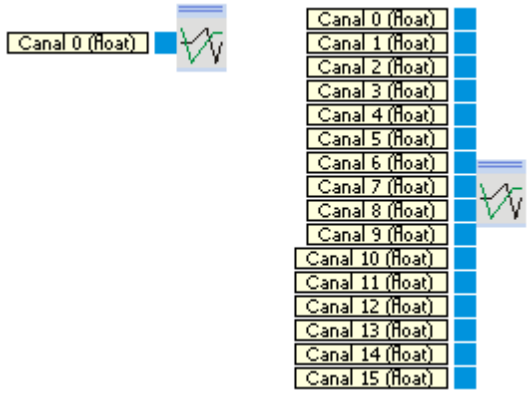
5.3.3.5.1 Exemple



5.4 Affichages

Cette catégorie contient des blocs de fonction pour la visualisation des données.

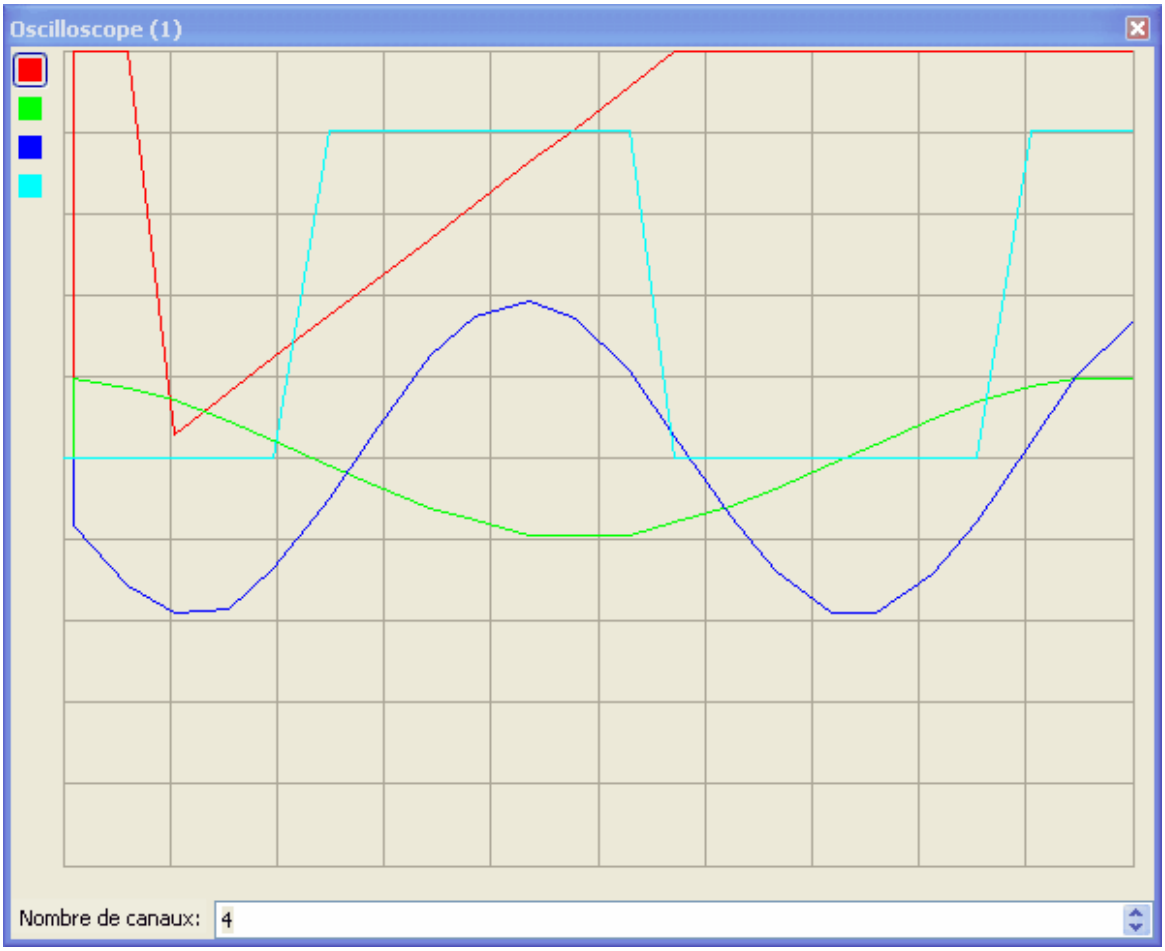
5.4.1 Oscilloscope



L'oscilloscope permet de visualiser jusqu'à 16 canaux.

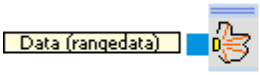
Entrées	Type	Standard	Description
Canal 0	float	0	
Canal 1	float	0	
...			
Canal 16	float	0	

5.4.1.1 Dialogue



La boîte de dialogue permet de visualiser les signaux appliqués aux canaux. Des paramètres, tels que le gain, peuvent être définis pour chaque canal. Il est également possible de désactiver certains canaux.

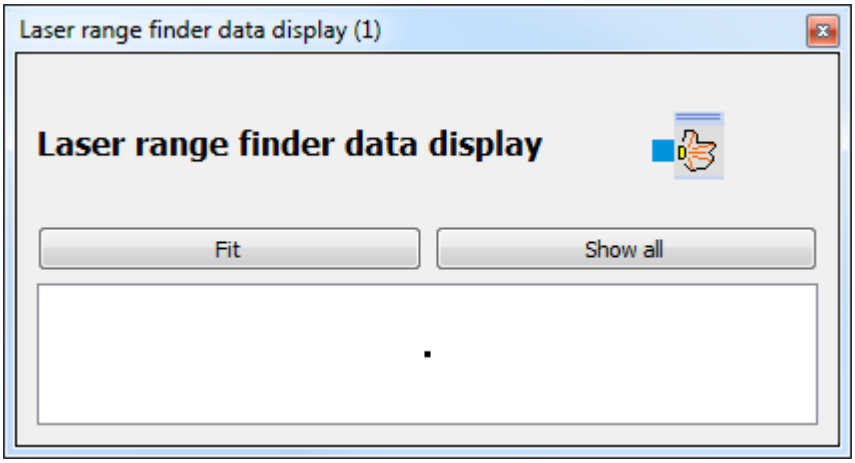
5.4.2 Affichage de données scanner laser



L'affichage de données scanner laser permet de visualiser les données d'un scanner laser.

Entrées	Type	Description
Données	laser range data	

5.4.2.1 Dialogue



5.5 Traitement d'images

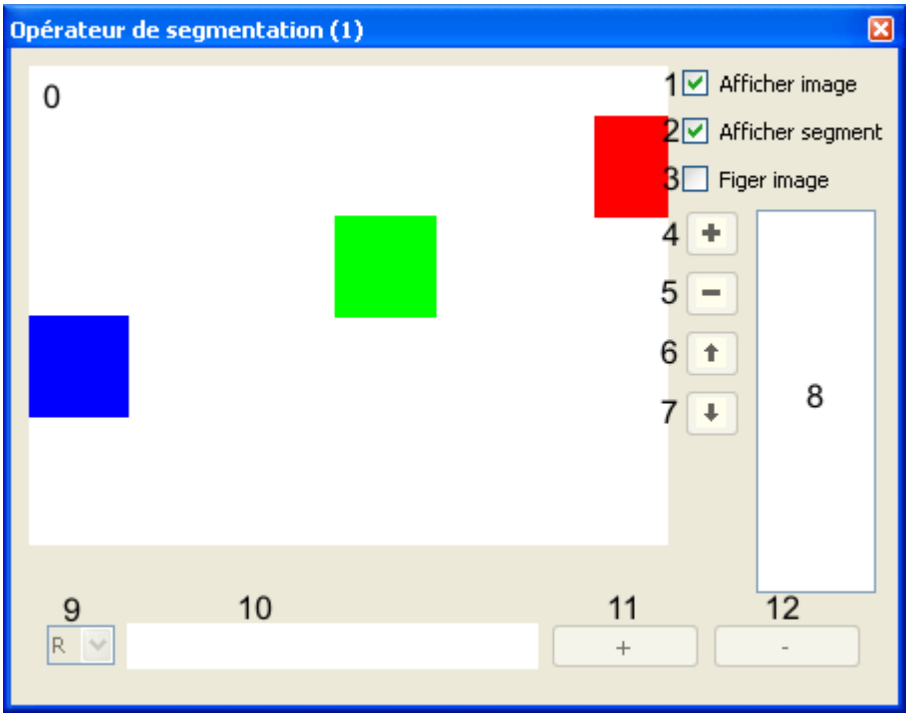
Cette catégorie contient des blocs de fonction de traitement d'images.

5.5.1 Opérateur de segmentation

L'opérateur de segmentation trouve dans une image les zones de même couleur.

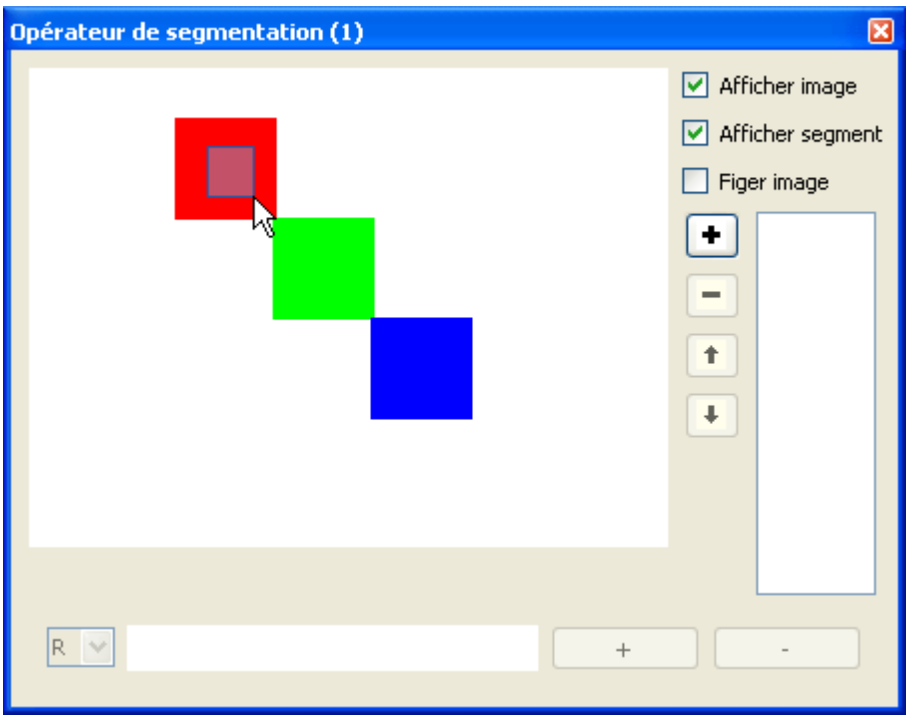
Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Sorties			
Sortie	image		Image segmentée

5.5.1.1 Dialogue

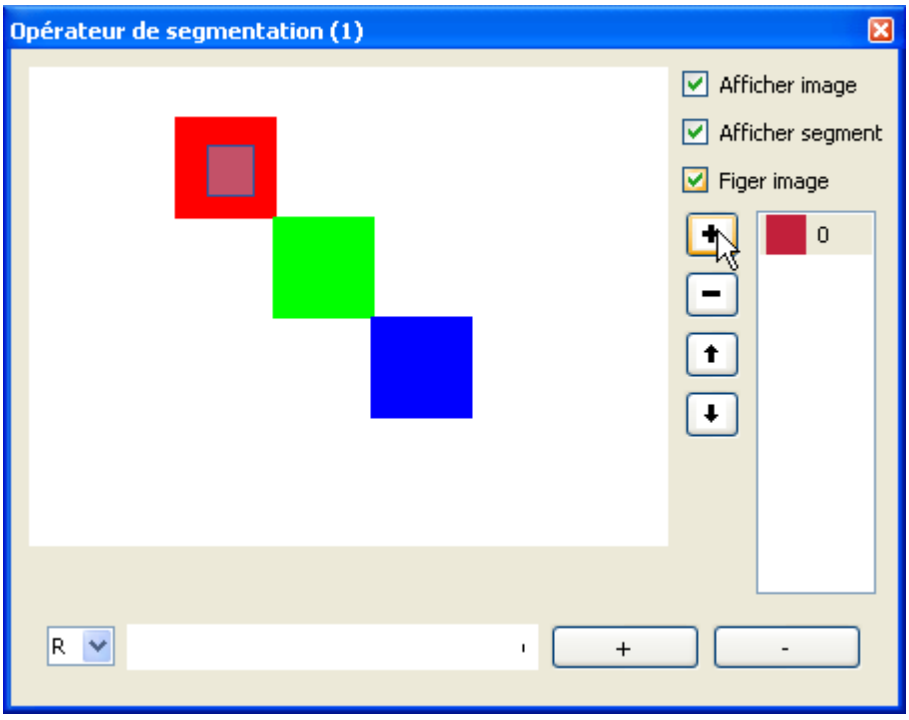


Bouton/affichage	Description
0	Affichage de l'image d'entrée et/ou des segments
1	S'il est activé, l'image d'entrée est affichée
2	S'il est activé, les segments sont affichés
3	Fige l'image d'entrée
4	Ajoute une sélection à l'image sous forme de segment
5	Supprime un segment de la liste des segments.
6	Décale un segment de la liste vers le haut
7	Décale un segment de la liste vers le bas
8	Liste des segments
9	Sélection du canal de couleur pour l'optimisation des segments
10	Affiche les valeurs d'un canal du segment sélectionné
11	Élargit les valeurs du canal voulu dans le segment sélectionné
12	Rétrécit les valeurs du canal voulu dans le segment sélectionné

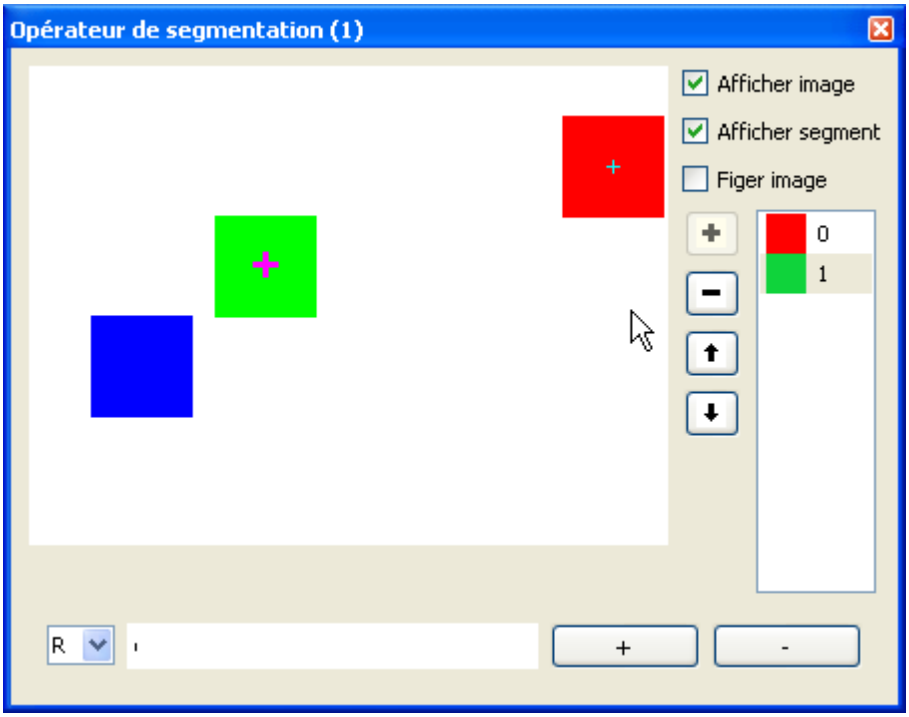
Pour identifier le carré comme surface d'un seul tenant, sélectionnez la zone au sein du carré à l'aide de la souris.



Cliquez sur + (bouton 4) pour enregistrer la sélection sous forme de segment.

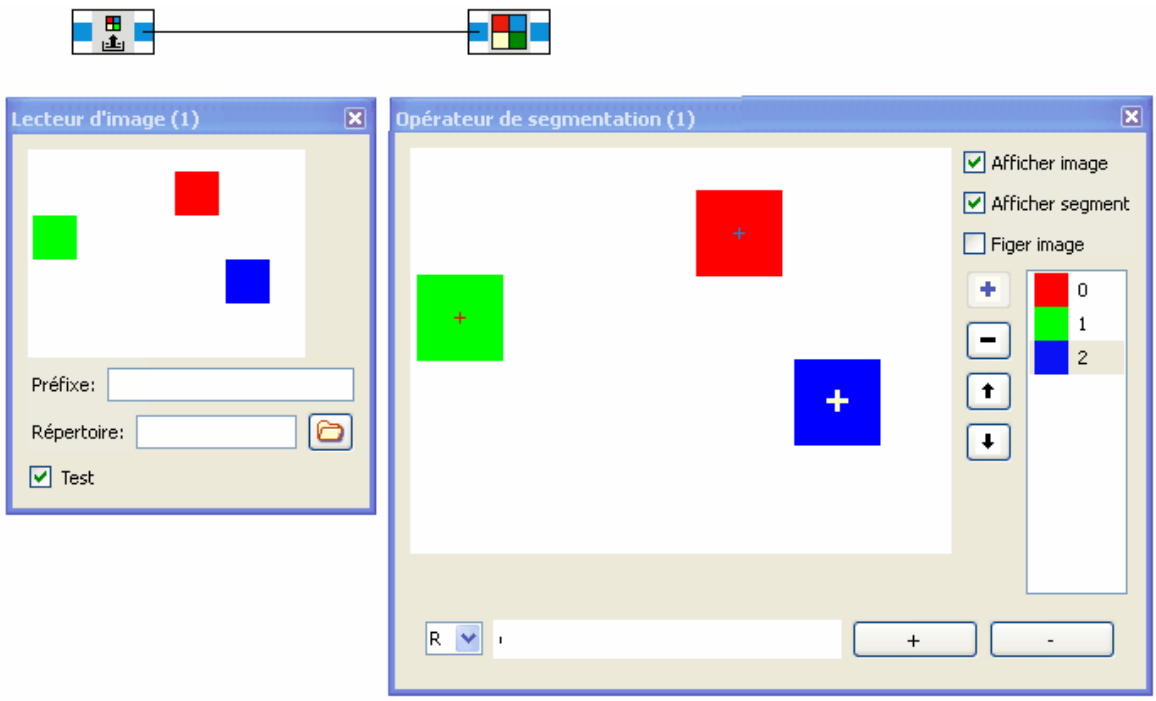


Le centre de gravité du segment sélectionné est représenté dans l'image sous forme de croix épaisse. Dans une image animée, la croix se déplace alors avec le carré rouge (désactiver au préalable l'arrêt sur image). Sélectionnez à présent une zone au sein du carré vert (désactivez à nouveau pour ce faire l'arrêt sur image) et ajoutez la sélection comme segment.



La liste contient maintenant deux segments. Le centre de gravité du segment vert est affiché sous forme de croix épaisse parce qu'à présent le segment vert est sélectionné.

5.5.1.2 Exemple



Le lecteur d'image est en mode test et génère une séquence d'images avec un carré animé rouge, vert et bleu. L'opérateur de segmentation a enregistré un segment pour chacune de ces couleurs. Le centre de gravité des carrés es affiché dans l'opérateur de segmentation.

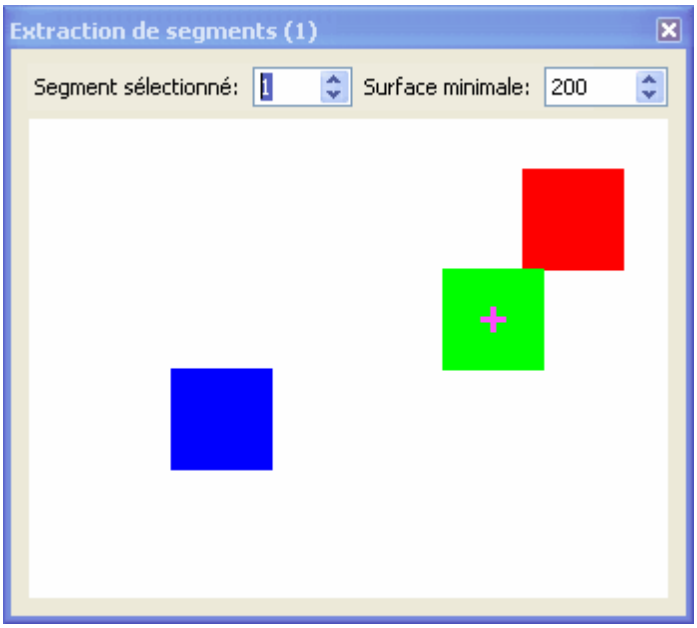
5.5.2 Extracteur de segment

Fournit la position et la taille d'un segment dans une image préalablement segmentée.

Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Segment sélectionné	int	0	Numéro du segment recherché.
Surface minimale	int	200	Le segment doit être constitué au moins du nombre indiqué de pixels pour que la sortie "Segment trouvé" passe à l'état vrai (true).
Sorties			
x	int		Coordonnées x du centre de gravité du segment trouvé

y	int		Coordonnées y du centre de gravité du segment trouvé
Surface	int		Nombre de pixels dont le segment trouvé est constitué
Segment trouvé	bool		Vrai (true) si le segment a été trouvé, sinon faux (false)

5.5.2.1 Dialogue

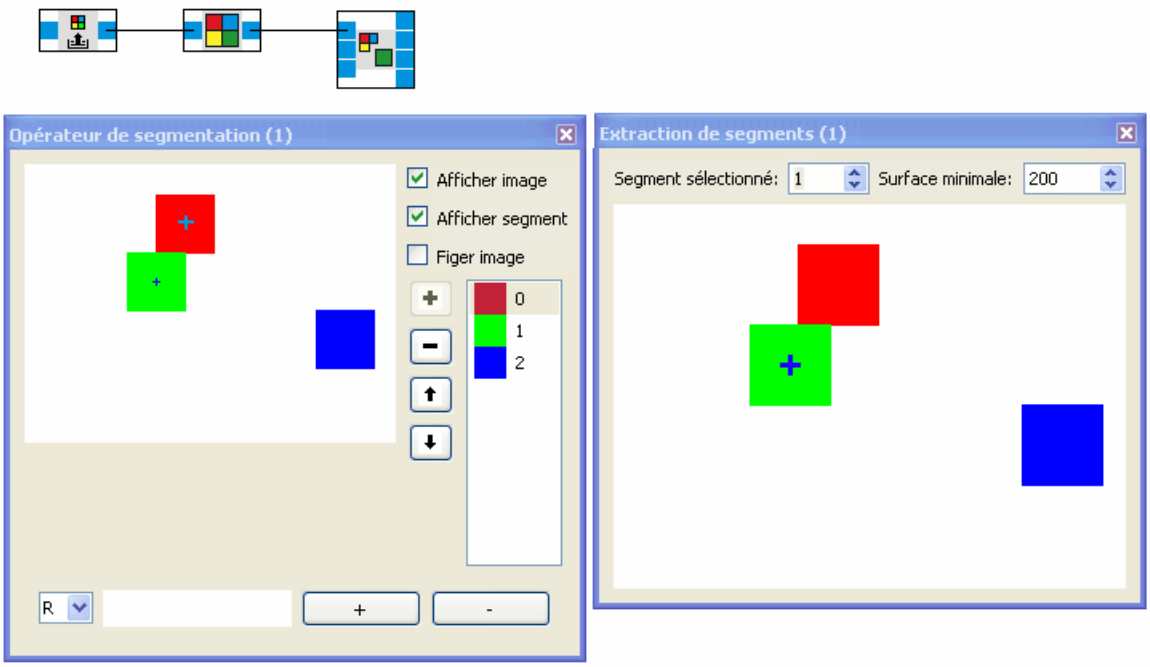


Segment sélectionné — Activé lorsque l'entrée du même nom n'est pas en circuit. Numéro du segment recherché.

Surface minimale — Activé lorsque l'entrée du même nom n'est pas en circuit. Le segment doit être constitué au moins du nombre indiqué de pixels pour que la sortie "Segment trouvé" passe à l'état vrai (true).

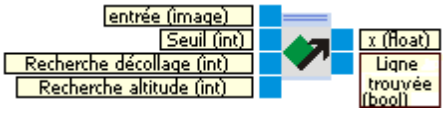
Affiche les segments présents dans l'image d'entrée. Le segment sélectionné est repéré (s'il a été trouvé) par un +.

5.5.2.2 Exemple



Le lecteur d'image génère une séquence de test avec trois carrés de couleur. L'opérateur de segmentation recherche dans l'image des surfaces rouges, vertes et bleues. L'extracteur de segment recherche le segment numéro 1 (le segment vert) et repère le centre de gravité par une croix.

5.5.3 Détection de ligne

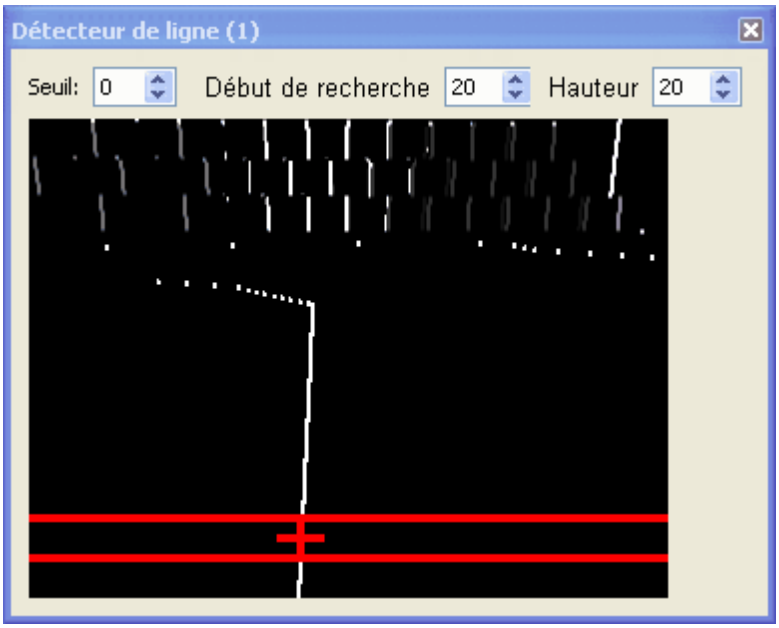


Trouve des lignes dans une image.

Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Seuil	int	0	Définit la sensibilité de l'algorithme au bruit dans l'image. Pour atténuer le bruit, il faut choisir une valeur de seuil plus élevée. Plage de valeurs : [0,255]
Début de recherche	int	20	La recherche d'une ligne débute au bord inférieur de l'image à la valeur indiquée.
Hauteur de recherche	int	20	L'image est analysée de bas en haut à la recherche de lignes. La hauteur de recherche définit le nombre de lignes d'image à prendre en compte pour une détection de ligne.

Sorties			
x	int		Position x de la ligne trouvée au sein de la fenêtre de recherche.
Ligne trouvée	bool		Vrai (true) si une ligne a été trouvée, sinon faux (false)

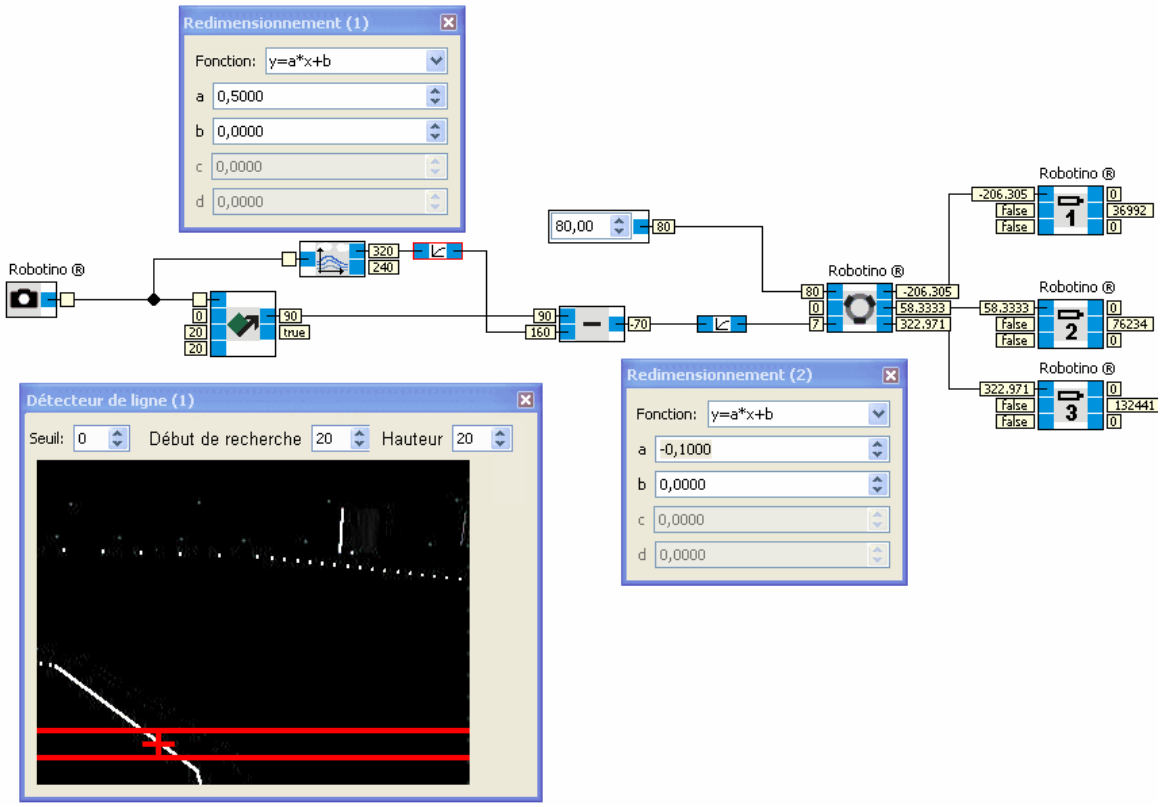
5.5.3.1 Dialogue



La zone dans laquelle une ligne est recherchée, est repérée par les deux lignes rouges horizontales. La ligne inférieure définit le début de la recherche. La ligne supérieure a pour coordonnées y la valeur Hauteur de l'image - "Début de la recherche" - Hauteur de recherche. La zone délimitée par les lignes constitue la fenêtre de recherche.

Le + rouge repère le bord sombre-clair de la ligne, venant de la gauche.

5.5.3.2 Exemple



La caméra de Robotino délivre une image servant d'entrée au détecteur de ligne. La position x de la ligne dans l'image est représentée, à l'aide de l'information d'image, non pas dans la zone [0;largeur d'image] mais dans la zone [-largeur d'image/2; largeur d'image/2]. Après changement de signe et mise à l'échelle, on obtient une valeur directement utilisable comme valeur d'entrée de la vitesse angulaire des moteurs de Robotino. Ceci permet de faire tourner Robotino à gauche lorsque la ligne se trouve dans la moitié gauche de l'image et vers la droite lorsque la ligne se trouve dans la moitié droite de l'image. La vitesse d'avance étant constante, Robotino se déplace le long de la ligne.

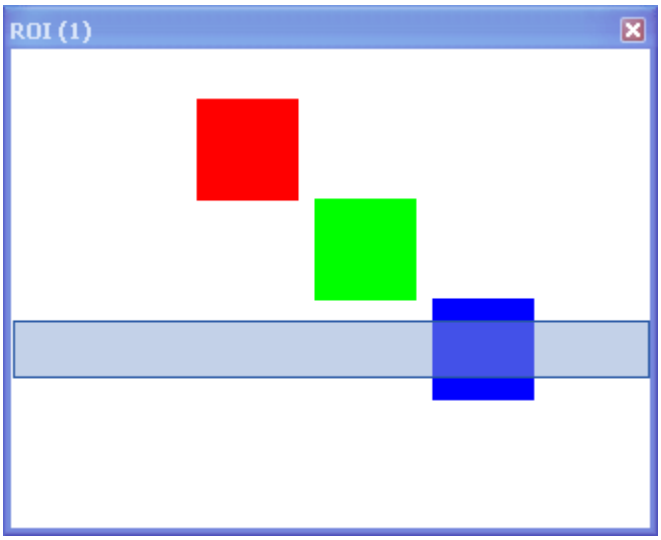
5.5.4 ROI

Sélectionne une région d'intérêt dans l'image (Region Of Interest, ROI).

Entrées	**Type**	**Standard**	**Description**
Entrée	image		Image d'entrée
Sorties			

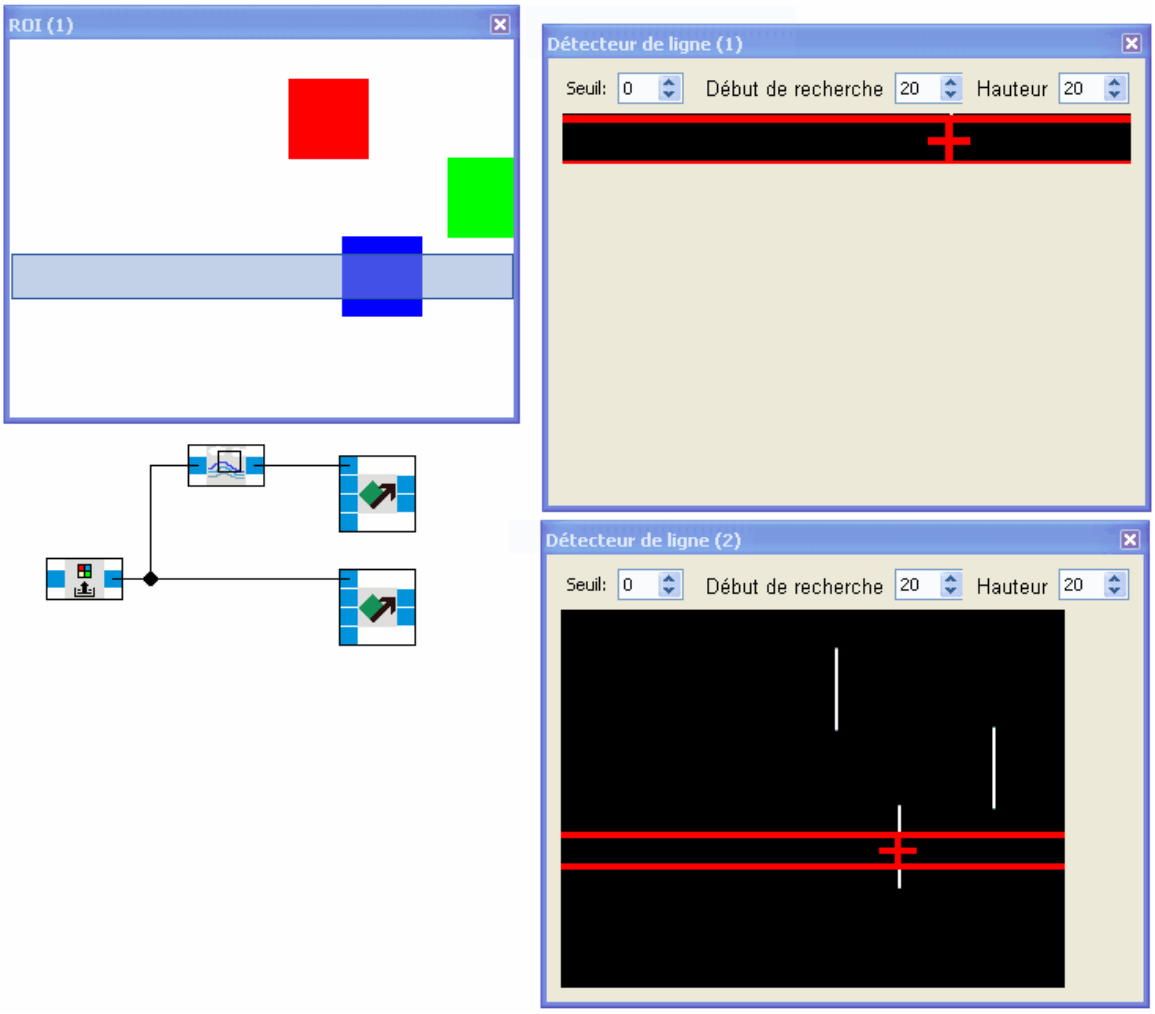
Sortie	image		L'image d'entrée complétée par la ROI. Les opérations de traitement d'image ci-après s'appliquent à la région sélectionnée.

5.5.4.1 Dialogue



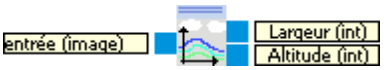
Dans l'image d'entrée affichée, on peut sélectionner une région d'intérêt avec la souris.

5.5.4.2 Exemple



Le lecteur d'image génère une séquence d'images de test. Le détecteur de ligne inférieur opère sur l'image complète générée par le lecteur d'image. Le détecteur de ligne supérieur opère uniquement sur la zone intéressante définie par la ROI.

5.5.5 Information d'image

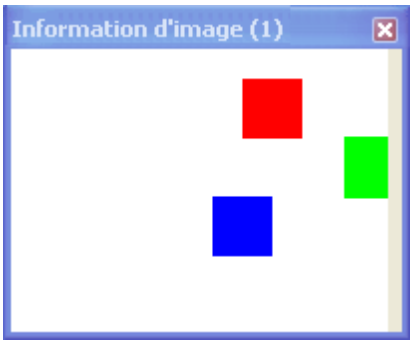


Indique la largeur et la hauteur d'une image.

Entrées	Type	Standard	Description
Entrée	image		Image d'entrée

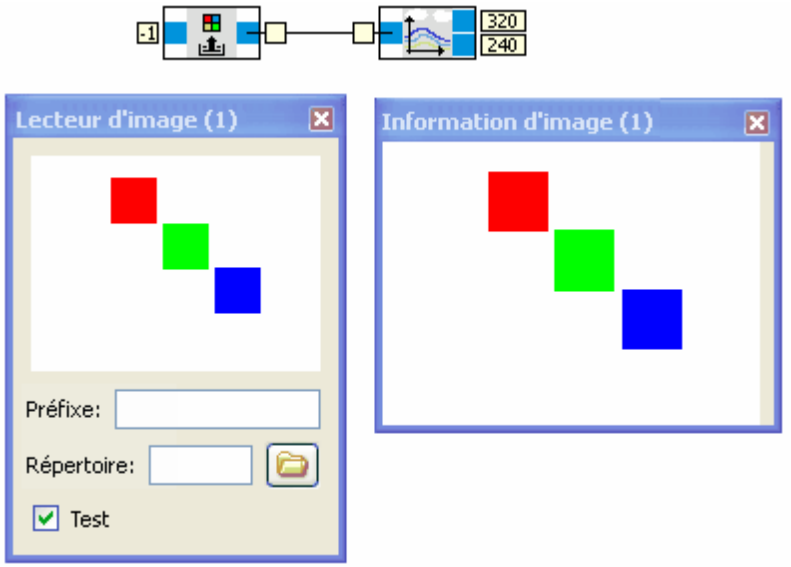
Sorties			
Largeur	int		La largeur de l'image en pixels
Hauteur	int		La hauteur de l'image en pixels

5.5.5.1 Dialogue



La boîte de dialogue affiche l'image d'entrée.

5.5.5.2 Exemple



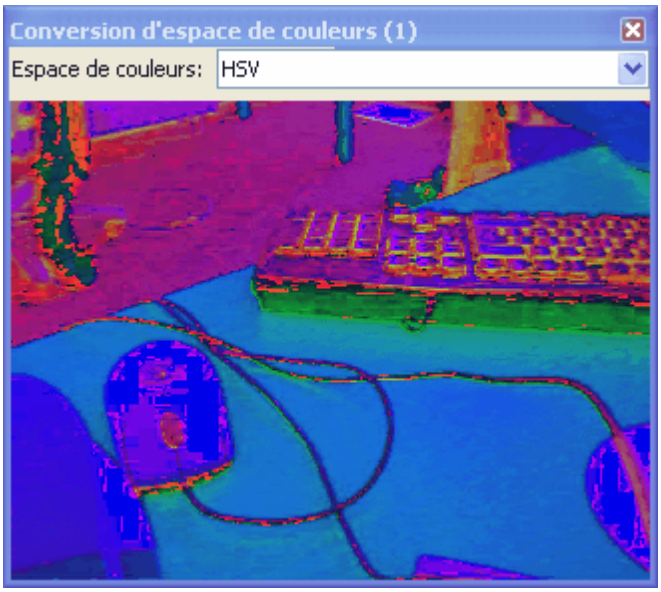
Les images de la séquence de test générées par le lecteur d'image ont une résolution de 320 x 240 pixels.

5.5.6 Conversion d'espace de couleur

Robotino transmet des images compressées JPEG qui sont décodées dans l'espace de couleur RVB. Dans l'espace de couleur RVB, les couleurs sont représentées selon la forme classique. Cet espace de couleur présente cependant l'inconvénient d'être très sensible aux influences de la lumière. L'espace de couleur YCbCr est en règle général bien plus stable en cas de variations de luminosité.

Entrées	**Type**	**Standard**	**Description**
Entrée	image		Image d'entrée
Sorties			
Sortie	image		Image convertie

5.5.6.1 Dialogue



La boîte de dialogue de conversion d'espace de couleur permet de définir l'espace de couleurs cible.

5.6 Générateurs

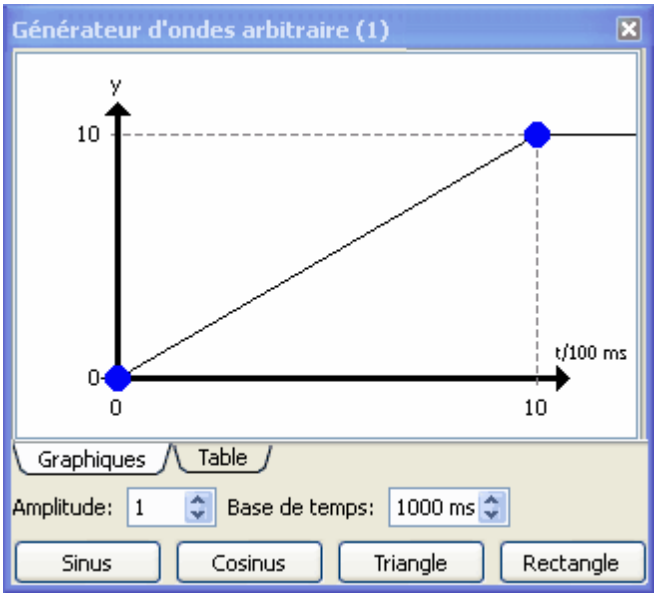
Cette catégorie contient des blocs de fonction permettant de générer des signaux.

5.6.1 générateur d'ondes arbitraire

Le générateur d'ondes arbitraire génère des signaux de sortie à ondes de forme voulue. Voir aussi http://en.wikipedia.org/wiki/Arbitrary_waveform_generator.

Entrées	Type	Standard	Description
Sorties			
Sortie	float		Le signal généré.

5.6.1.1 Dialogue



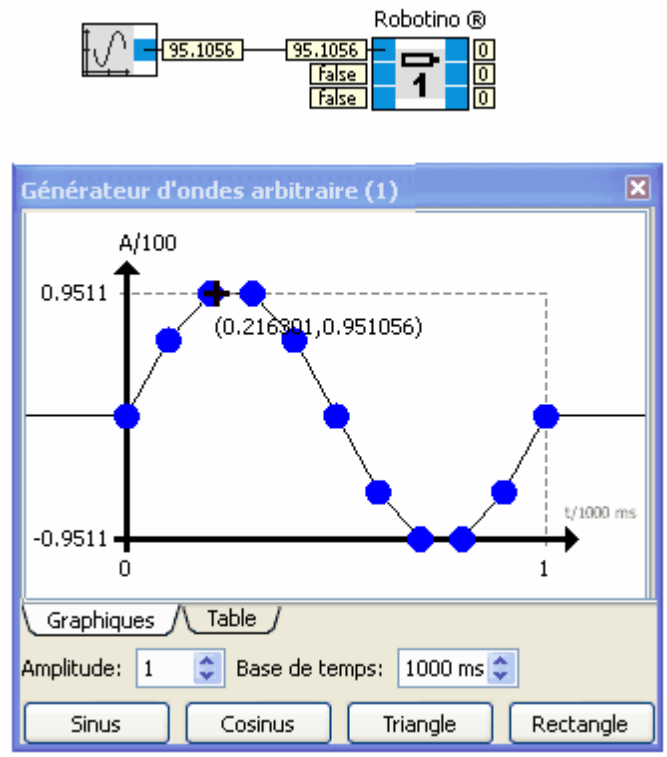
La partie supérieure de la boîte de dialogue est constituée, comme pour la fonction de transfert, 61 d'un graphique et d'une table permettant de définir la fonction voulue au moyen de points de contrôle.

Amplitude La valeur de sortie du générateur est multipliée par l'amplitude.

Base de temps Unité de l'axe x. Avec une base de temps de 100ms, la valeur 10 de l'axe x est atteinte au bout de 1 s.

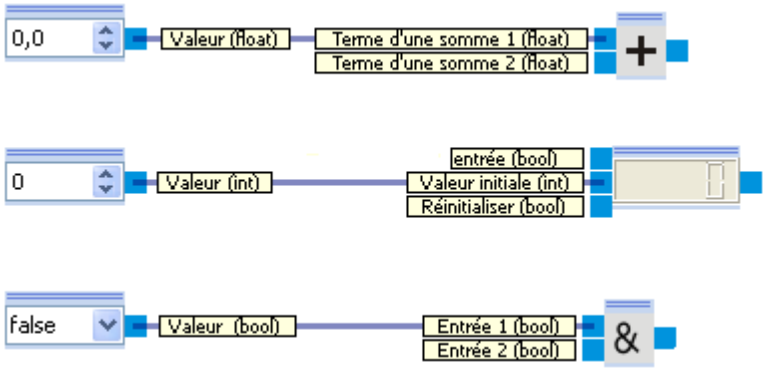
Sinus	Génère des points de contrôle constituant l'approximation d'une onde sinusoïdale.
Cosinus	Génère des points de contrôle constituant l'approximation d'une onde de type cosinus.
Triangle	Génère des points de contrôle constituant l'approximation d'une onde triangulaire.
Rectangle	Génère des points de contrôle constituant l'approximation d'une onde carrée.

5.6.1.2 Exemple



Le moteur 1 de Robotino, piloté par une sinusoïde, tourne en marche avant et en marche arrière.

5.6.2 Constante



Génère une valeur constante. Le type de la constante et par conséquent sa représentation graphique varient en fonction du type de données de l'entrée connectée.

La valeur des constantes est directement entrée dans le programme. Les entrées au clavier modifient directement la valeur.

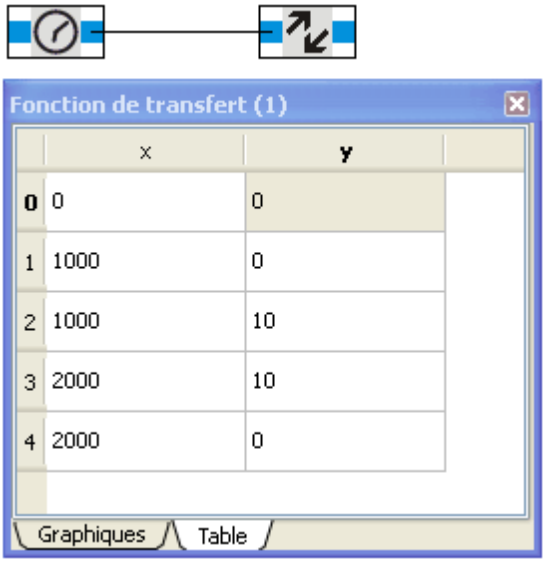
Entrées	**Type**	**Standard**	**Description**
Sorties			
Valeur	float, int, bool	0 / false	La valeur de la constante.

5.6.3 Bloc de temporisation

Mesure le temps en millisecondes écoulé depuis le démarrage du programme ou depuis une transition de vrai (true) à faux (false) de l'entrée de remise à zéro.

Entrées	**Type**	**Standard**	**Description**
Réinitialiser	bool	false	Si vrai, la sortie du bloc de temporisation délivre un 0. Si faux, le bloc de temporisation est activé.
Sorties			
Temps	float		Temps en millisecondes écoulé depuis le démarrage du programme ou depuis une transition de vrai (true) à faux de l'entrée de remise à zéro.

5.6.3.1 Exemple



Le bloc de temporisation génère, avec la fonction de transfert, [61] une impulsion de niveau 10 d'une durée de 1 s et ceci 1s après le démarrage du programme.

5.6.4 Générateur aléatoire

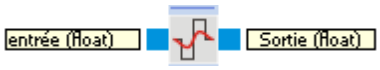
Le générateur aléatoire génère des nombres aléatoires dans une plage de valeurs définie.

Entrées	Type	Standard	Description
Maximum	float	1	Limite supérieure de la plage de valeurs.
Minimum	float	0	Limite inférieure de la plage de valeurs.
Sorties			
Valeur	float	0	Nombre aléatoire entre **minimum** et **maximum**.

5.7 Filtre

Cette catégorie contient des blocs de fonction de filtrage et de lissage des signaux.

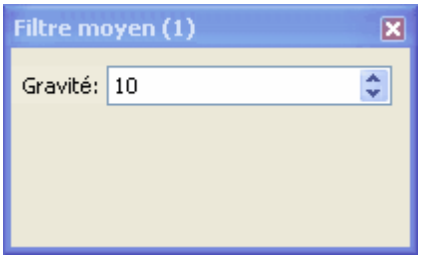
5.7.1 Filtre moyen



Forme la moyenne du signal d'entrée sur au maximum 1000 cycles.

Entrées	Type	Standard	Description
Entrée	float	0	Signal d’entrée
Sorties			
Sortie	float		Signal d'entrée lissé

5.7.1.1 Dialogue

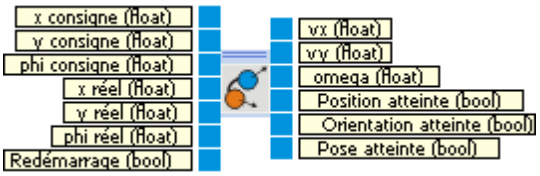


La profondeur détermine le nombre de cycles antérieurs servant à calculer la moyenne.

5.8 Navigation

Cette catégorie contient des blocs de fonction de navigation.

5.8.1 Parcoureur de positions



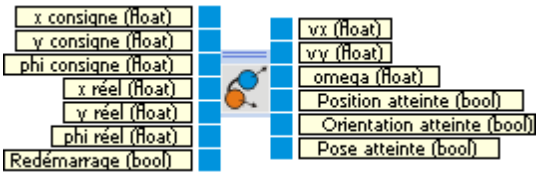
Mit dem Positionsfahrer können Positionen angefahren werden.

Der Positionsfahrer berechnet aus einer Soll- und einer Istposition Geschwindigkeit und Winkelgeschwindigkeit, so dass Robotino von der Ist- zur Sollposition fährt.

Eingänge	Typ	Einheit	Beschreibung
x Soll	float	mm	x Koordinate der Soll Position im Weltkoordinatensystem.
y Soll	float	mm	y Koordinate der Soll Position im Weltkoordinatensystem.
phi Soll	float	Grad	Winkel phi der Soll Position im Weltkoordinatensystem.
x Ist	float	mm	x Koordinate der Ist-Position im Weltkoordinatensystem.
y Ist	float	mm	y Koordinate der Ist-Position im Weltkoordinatensystem.
phi Ist	float	Grad	Winkel phi der Ist-Position im Weltkoordinatensystem.
Neustart	bool		Startet die Bewegung erneut
Ausgänge			
vx	float	mm/s	x Geschwindigkeit.
vy	float	mm/s	y Geschwindigkeit.
omega	float	Grad/s	Winkelgeschwindigkeit.
Position erreicht	bool		Sobald die Ausgänge vx und vy 0 liefern, wird die Zielposition als erreicht angesehen und der Ausgang liefert wahr.
Orientierung erreicht	bool		Sobald der Ausgang omega 0 liefern, wird die Zielorientierung als erreicht angesehen und der Ausgang liefert wahr.
Pose erreicht	bool		Der Ausgang liefert wahr, wenn sowohl die Zielposition als auch die Zielorientierung erreicht sind. Ansonsten unwahr.

Siehe Bewegungen 100

----------OLD_TEXT----------



Le parcoureur de position permet de se rendre d'une position à l'autre.

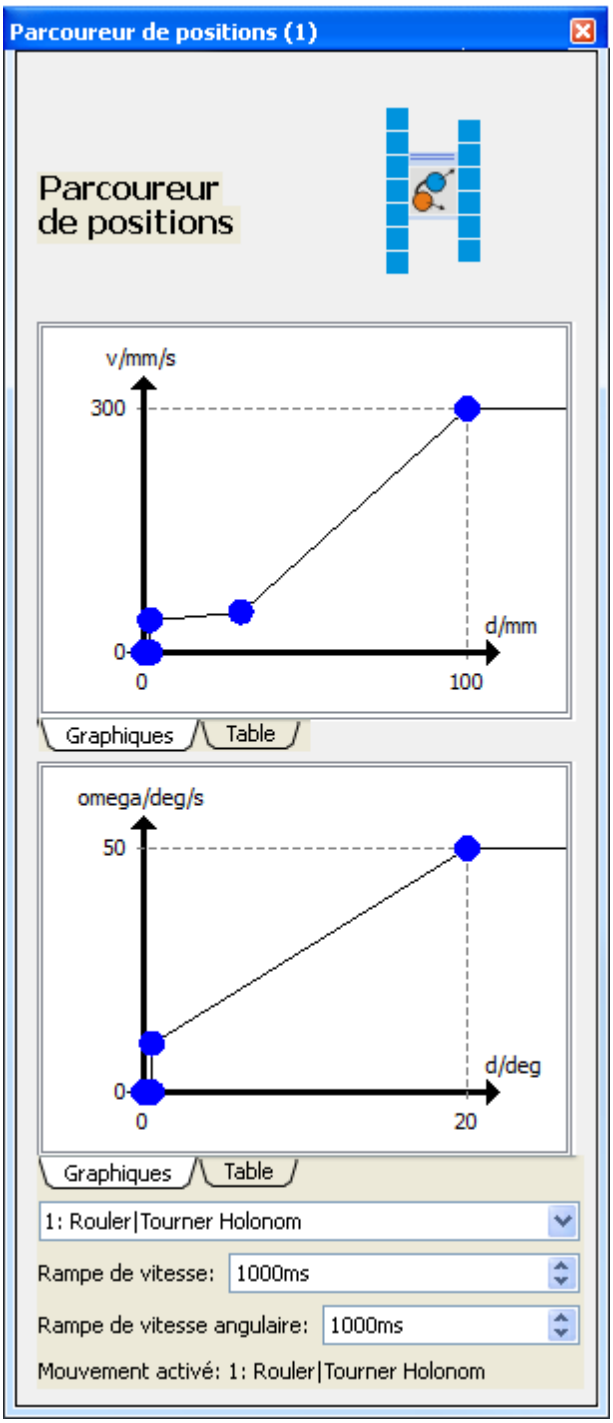
Le parcoureur de position calcule la vitesse et la vitesse angulaire en fonction de la position de consigne et de la position réelle de sorte que Robotino se rende de la position réelle à la position de consigne.

Entrées	Type	Unité	Description
x cons	float	mm	Coordonnées x de la position de consigne dans le système de coordonnées mondial.
y cons	float	mm	Coordonnées y de la position de consigne dans le système de coordonnées mondial.

phi cons	float	deg	Coordonnées phi de la position de consigne dans le système de coordonnées mondial.
x eff	float	mm	Coordonnées x de la position réelle dans le système de coordonnées mondial.
y eff	float	mm	Coordonnées y de la position réelle dans le système de coordonnées mondial.
phi eff	float	deg	Coordonnées phi de la position réelle dans le système de coordonnées mondial.
Redémarrage	bool		Redémarre le mouvement
Sorties			
vx	float	mm/s	Vitesse x.
vy	float	mm/s	Vitesse y.
omega	float	deg/s	Vitesse angulaire
Position atteinte	bool		Dès que les sorties vx et vy délivrent 0, la position de destination est considérée atteinte et la sortie délivre un signal vrai.
Orientation atteinte	bool		Dès que la sortie omega délivre 0, l'orientation de destination est considérée atteinte et la sortie délivre un signal vrai.
Pose atteinte	bool		La sortie délivre un signal vrai si à la fois la position et l'orientation de destination sont atteintes. Sinon faux.

Voir Mouvements 100

5.8.1.1 Dialogue

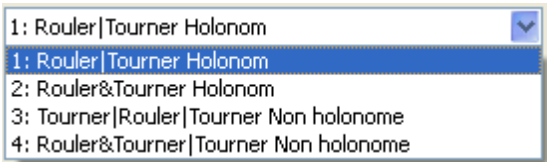


Le dialogue du parcoureur de positions se répartit en trois zones.

La zone supérieure indique la relation entre la distance jusqu'à la position de destination d (mesurée en mm) et la vitesse d'avance v (mesurée en mm/s).

La zone centrale indique la relation entre la distance angulaire à l'angle de destination d (mesurée en degrés) et la vitesse de rotation v (mesurée en degrés/s). La distance angulaire varie de 0° à 180°. Les rotations dans le sens horaire ou antihoraire sont traitées symétriquement. La rotation est exécutée dans le sens horaire ou antihoraire de sorte à minimiser la distance de rotation.

La zone de liste déroulante



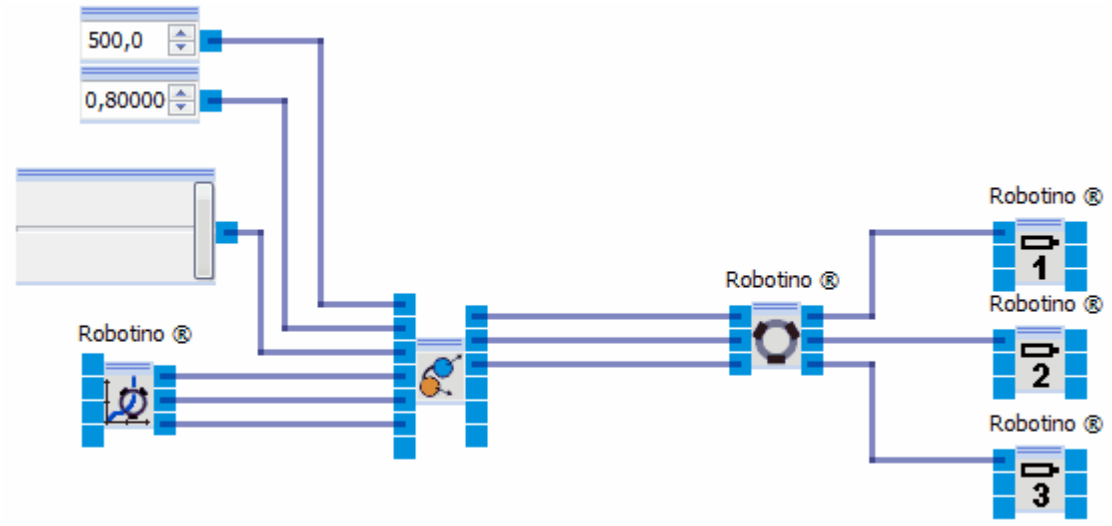
permet de définir le type de mouvement (voir Mouvements 100). Le mouvement sélectionné ici est le mouvement activé au

1. démarrage du programme
2. Si l'entrée Redémarrage est vraie

La rampe de vitesse est le temps en millisecondes au bout duquel 100% de la vitesse voulue sont atteints. On évite ainsi une accélération brutale au début du mouvement.

La rampe de vitesse angulaire est le temps en millisecondes au bout duquel 100% de la vitesse de rotation voulue sont atteints. Ceci permet d'amortir le mouvement en début de rotation.

5.8.1.2 Exemple



5.8.1.3 Mouvements

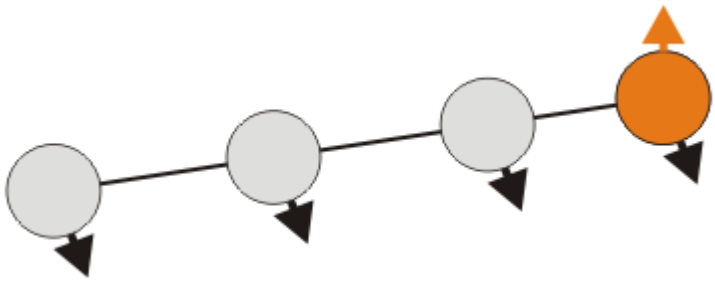
Il existe quatre types de mouvements. Deux conviennent aux véhicules holonomes, deux aux véhicules non holonomes. Robotino possédant un entraînement holonome - les trois degrés de liberté dans un plan peuvent être modifiés indépendamment l'un de l'autre - Robotino peu effectuer les quatre mouvements. Dans le cas des mouvements non holonomes, la sortie vy est constamment égale à 0.

Les mouvements débutent au démarrage du programme ou si l'entrée Redémarrage passe à vrai. Le mouvement ne débute effectivement dans le deuxième cas que lorsque l'entrée Redémarrage retourne à faux.

Mouvement 1 - rouler, tourner - (holonome)

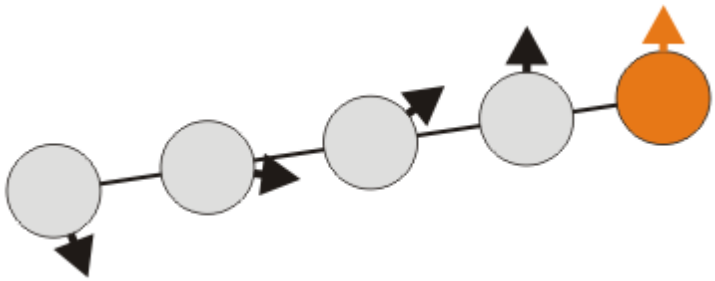
Étape 1 : Rouler vers la position de destination en conservant l'orientation de la position de départ

Étape 2 : Lorsque la position de destination est atteinte tourner jusqu'à l'orientation de destination



Mouvement 2 - rouler & tourner - (holonome)

Étape 1 : Rouler et en même temps tourner jusqu'à l'orientation de destination

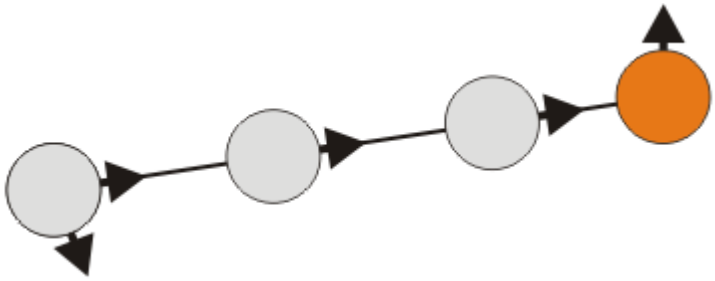


Mouvement 3 - tourner, rouler, tourner - (non holonome)

Étape 1 : Tourner jusqu'en direction de roulage

Étape 2 : Rouler jusqu'à la position de destination

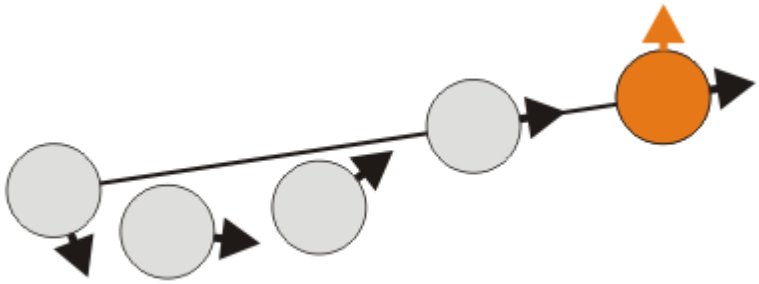
Étape 3 : Lorsque la position de destination est atteinte tourner jusqu'à l'orientation de destination



Mouvement 4 - rouler & tourner, tourner - (non holonome)

Étape 1 : Rouler et tourner en direction de roulage

Étape 2 : Lorsque la position de destination est atteinte tourner jusqu'à l'orientation de destination



5.8.2 Pose constante

Ce champ de saisie permet de définir la pose. Les coordonnées sont séparées par une espace.

Entrée	Pose résultante
x y phi	(x, y, phi)
x y	(x, y, non valide)
x	Pose non valide
	Pose non valide

L'orientation phi est indiquée dans le champ de saisie en degrés.

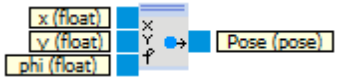
Exemple :

10.5 20 120

soit x=10.5 y=20 et l'orientation=120°

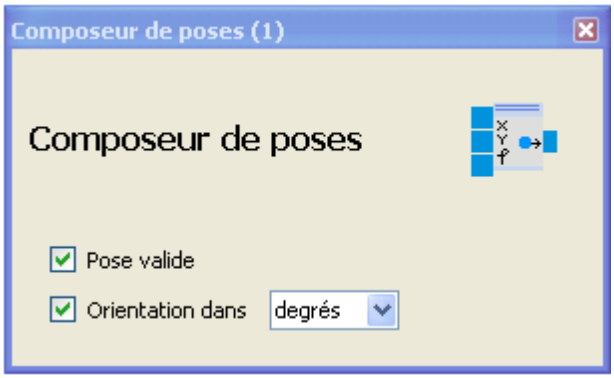
Entrées	Type	Standard	Description
Sorties			
Pose	pose	Pose non valide	La valeur de la pose constante. La valeur de l'orientation est représentée en sortie en radians.

5.8.3 Composeur de poses



Entrées	Type	Unité	Standard	Description
x	float		0	La composante x de la pose.
y	float		0	La composante y de la pose.
phi	float	Degré	0	Orientation de la pose en degrés. Dans la boîte de dialogue, 103 l'unité peut être changée en radian.
Sorties				
Pose	pose		(0, 0, 0)	La pose obtenue par la combinaison des valeurs (x, y, phi). La valeur de l'orientation est représentée en sortie en radians.

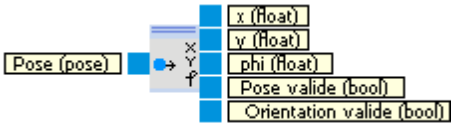
5.8.3.1 Dialogue



Pose valide	Détermine la validité de la pose. Les poses non valides sont ignorées dans l'itinéraire.

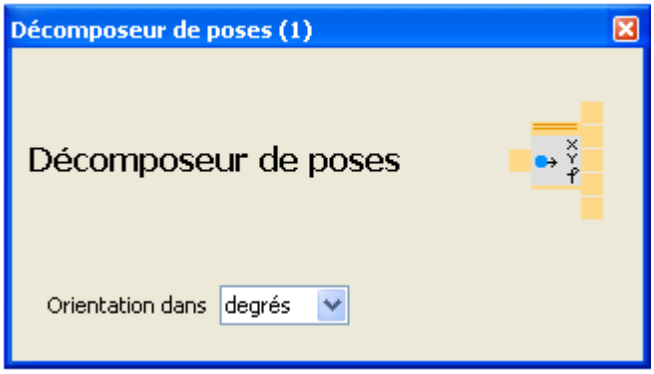
Orientation	Détermine la validité de l'orientation ainsi que l'unité (degré ou radian).

5.8.4 Décomposeur de poses



Entrées	Type	Unité	Standard	Description
Pose	pose		(0, 0, 0)	La pose à décomposer
Sorties				
x	float		0	La composante x de la pose.
y	float		0	La composante y de la pose.
phi	float	Degré	0	Orientation de la pose en degrés. Dans la boîte de dialogue, 104 l'unité peut être changée en radian.
Pose valide	bool		false	Indique si la pose est valide ou non.
Orientation valide	bool		false	Indique si l'orientation enregistrée dans la pose est valide ou non.

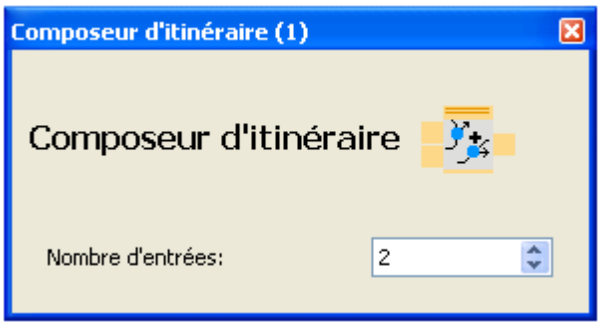
5.8.4.1 Dialogue



5.8.5 Composeur d'itinéraire

Entrées	Type	Standard	Description
Itinéraire 1	path	Itinéraire vide	Premier élément d'itinéraire. Il est possible de créer ici une pose car pose est convertible en path. Voir Transtypage 19.
...			
Itinéraire 20	path	Itinéraire vide	Dernier élément d'itinéraire. Il est possible de créer ici une pose car pose est convertible en path. Voir Transtypage 19.
Sorties			
Itinéraire	path	Itinéraire vide	L'itinéraire composé d'éléments d'itinéraire Itinéraire 1 + ... + Itinéraire 20

5.8.5.1 Dialogue



5.8.6 Décomposeur d'itinéraire

Découpe un élément d'itinéraire dans un itinéraire. Un itinéraire est constitué d'une liste de poses.

Indice	Pose
1	p1
2	p2
...	

N	pN

Les entrées **Démarrer** et **Longueur** déterminent la pose de départ et la longueur de l'itinéraire décomposé. **Démarrer** doit être situé dans la plage de valeurs [1;N]. Si **Démarrer** ‹1, la valeur 1 est utilisée en interne. Si **Démarrer** › la longueur de l'itinéraire, un itinéraire vide est délivré en sortie. **Longueur** doit être située dans la plage de valeurs [0;N-**Démarrer**+1]. En cas de **Longueur** ‹=0 un itinéraire vide est délivré en sortie. Si la longueur ›N-**Démarrer**+1, l'itinéraire débutant par l'indice **Démarrer** est délivré en sortie.

Exemples :

Itinéraire = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Démarrer = 3
Longueur = 5
Élément d'itinéraire = p3, p4, p5, p6, p7

Itinéraire = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Démarrer = 0
Longueur = 1
Élément d'itinéraire = p1

Itinéraire = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Démarrer = 11
Longueur = 1
Élément d'itinéraire = itinéraire vide

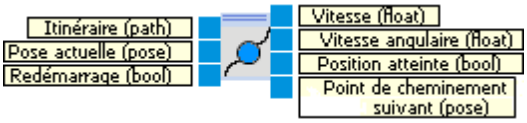
Itinéraire = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Démarrer = 1
Longueur = 0
Élément d'itinéraire = itinéraire vide

Itinéraire = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Démarrer = 2
Longueur = 20
Élément d'itinéraire = p2, p3, p4, p5, p6, p7, p8, p9, p10

Entrées	**Type**	**Standard**	**Description**
Itinéraire	path	Itinéraire vide	L'itinéraire à décomposer

Démarrer	int	1	La pose à l'indice **Démarrer** de l'itinéraire à décomposer devient la première pose de l'itinéraire décomposé.
Longueur	int	1	L'itinéraire décomposé est constitué de **Longueur** Poses en commençant par la pose de l'indice **Démarrer** de l'itinéraire à décomposer.
Sorties			
Élément d'itinéraire	path	Itinéraire vide	L'itinéraire en sortie commence par la pose de l'indice **Démarrer** et comprend **Longueur** Poses.

5.8.7 Parcoureur d'itinéraire

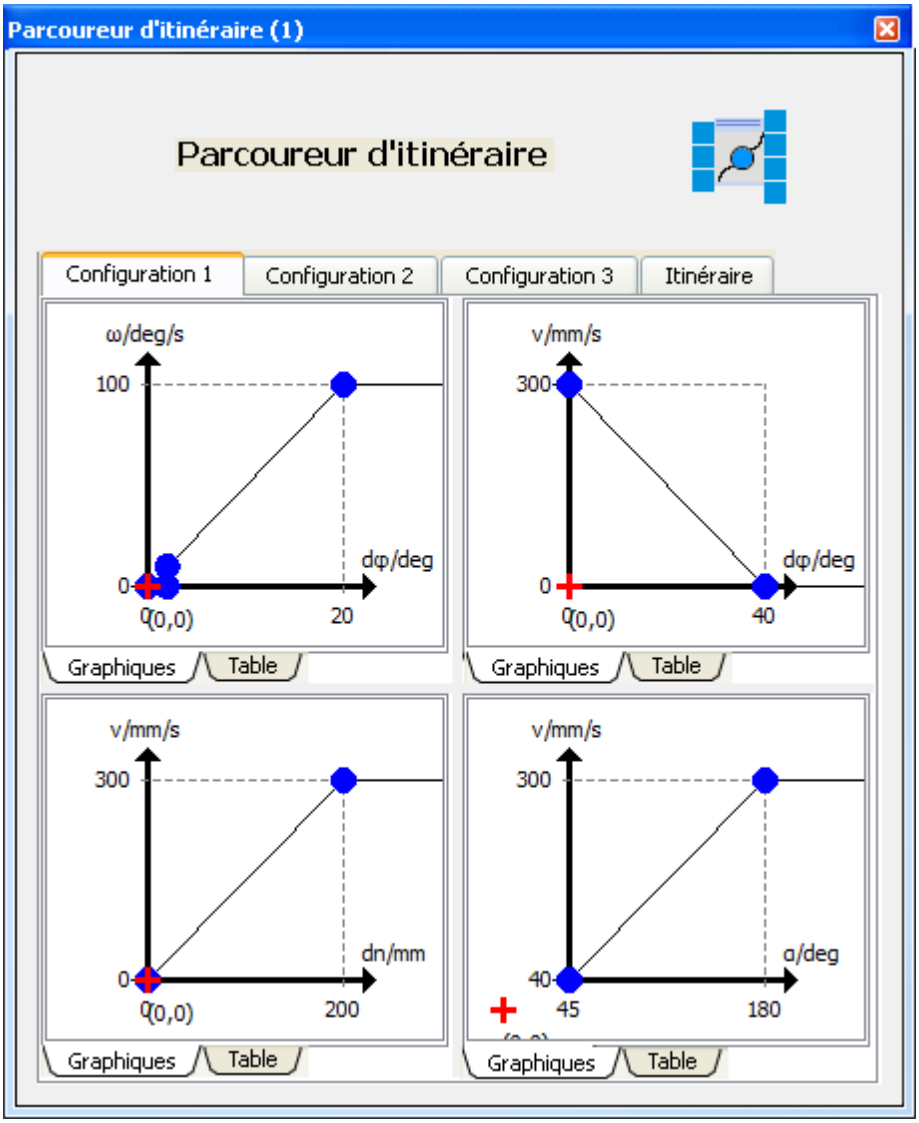


Le parcoureur d'itinéraire permet de rouler en suivant un itinéraire défini.

La vitesse d'avance et la vitesse angulaire sont calculées à partir de l'itinéraire et de la pose actuelle de sorte que Robotino exécute des trajectoires rectilignes entre les différentes poses.

Entrées	**Type**	**Unité**	**Standard**	**Description**
Course	path		Itinéraire vide	L'itinéraire à parcourir.
Pose actuelle	Pose		(0, 0, 0)	La pose actuelle déterminée par odométrie ou SLAM.
Redémarrage	bool		false	Redémarre le mouvement
Sorties				
Vitesse	float	mm/s		Vitesse d'avance.
Vitesse angulaire	float	deg/s		Vitesse angulaire
Position atteinte	bool			En cas d'itinéraire vide, la sortie retourne un signal vrai. Sinon la sortie délivre un signal vrai si le point virtuel se trouve sur le dernier tronçon d'itinéraire et si v(d) = 0.
Point de cheminement suivant	Pose			Point de cheminement suivant à parcourir.

5.8.7.1 Dialogue de configuration 1



En haut à gauche

Relation entre vitesse de rotation et erreur angulaire d□.

En haut à droite

Relation entre vitesse d'avance et erreur angulaire d□.

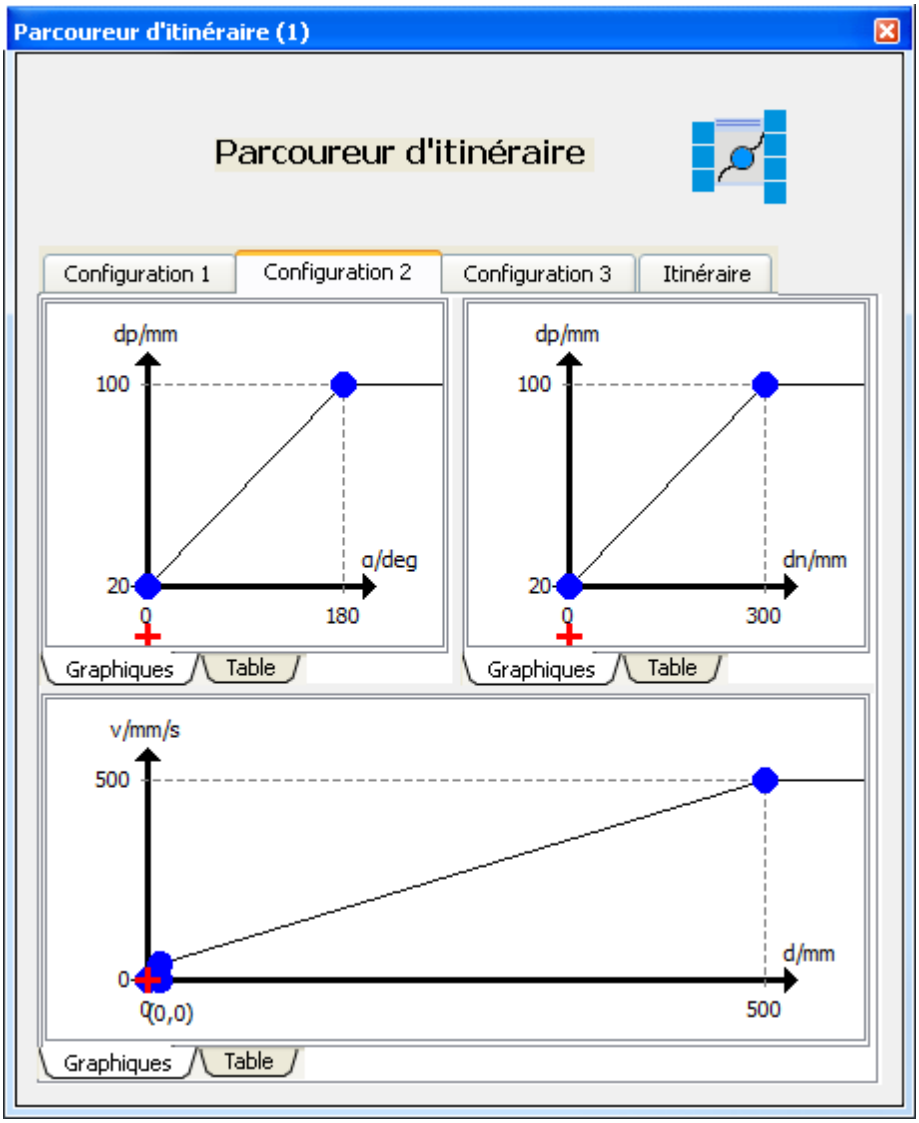
En bas à gauche

Relation entre vitesse d'avance et distance du point de cheminement suivant.

En bas à droite

Relation entre vitesse d'avance et angle du prochain tronçon d'itinéraire.

5.8.7.2 Dialogue de configuration 2



En haut à gauche

Relation entre distance du robot au point de cheminement virtuel et angle du prochain tronçon d'itinéraire.

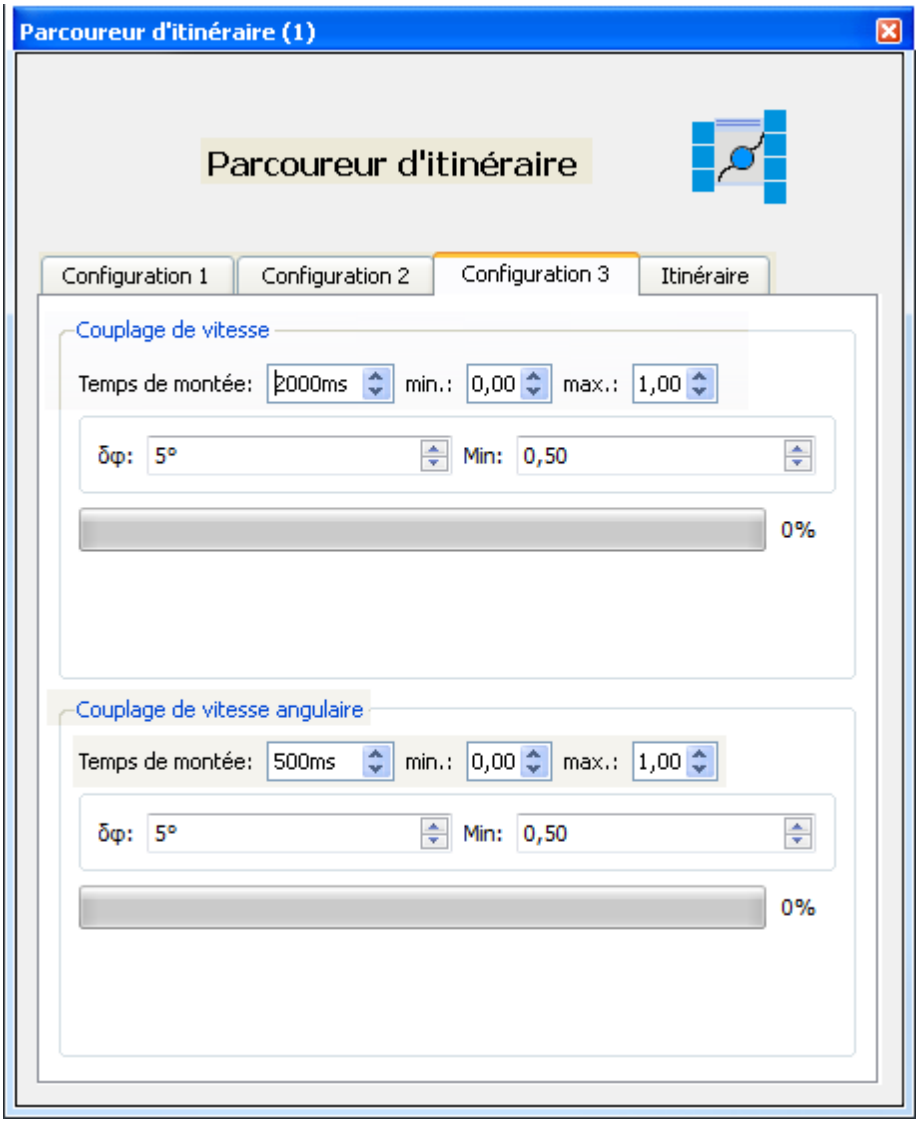
En haut à droite

Relation entre distance du robot au point de cheminement virtuel et distance du point de cheminement suivant.

En bas

Relation entre vitesse d'avance et distance de la fin de l'itinéraire.

5.8.7.3 Dialogue de configuration 3



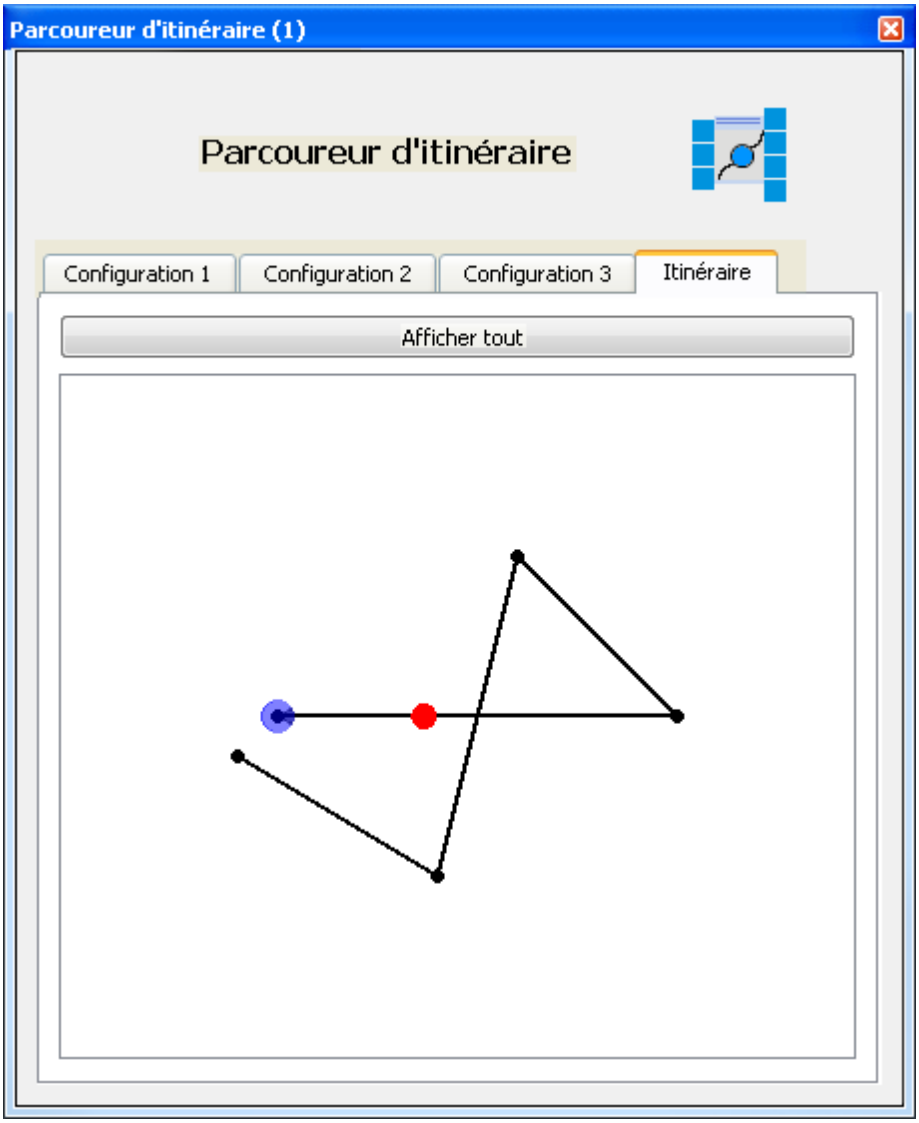
En haut

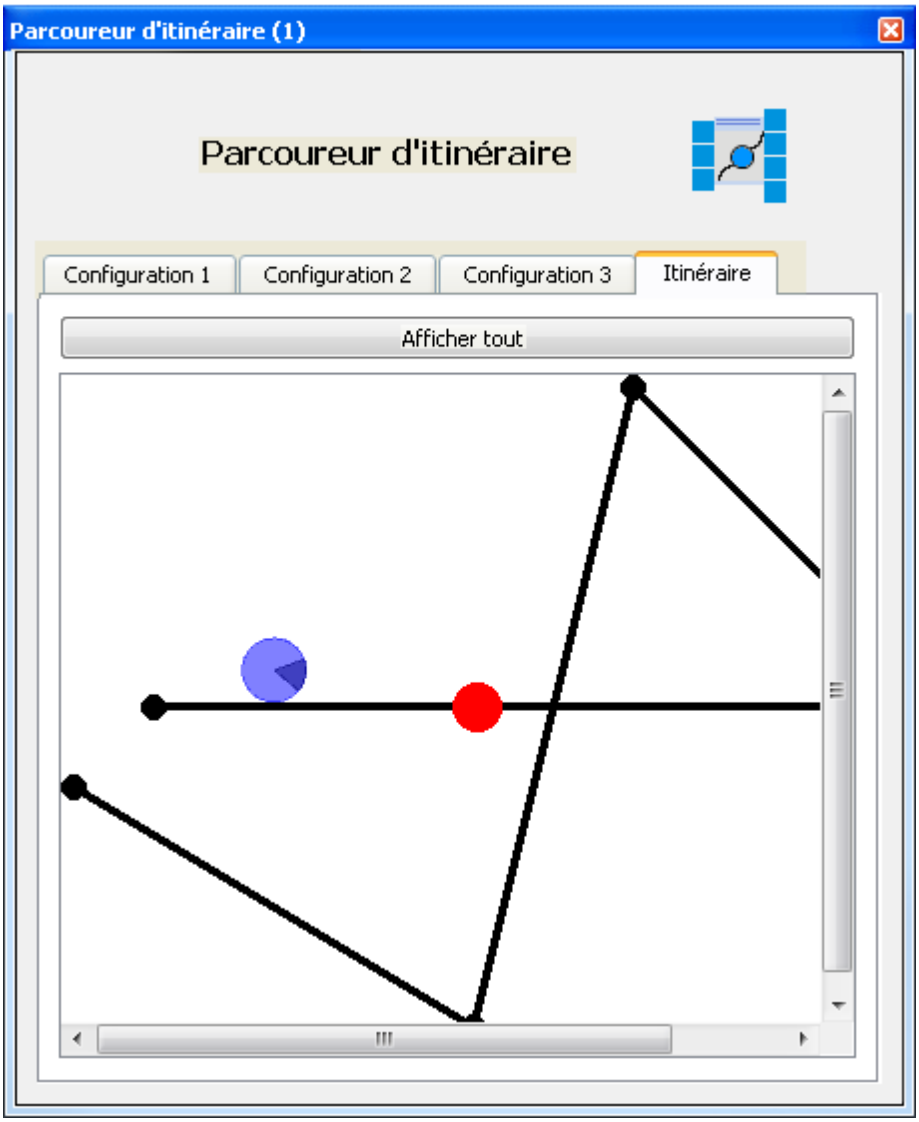
Adaptation du facteur de couplage entre la vitesse calculée en fonction de la configuration des dialogues 1 et 2 et de la vitesse effective en sortie.

En bas

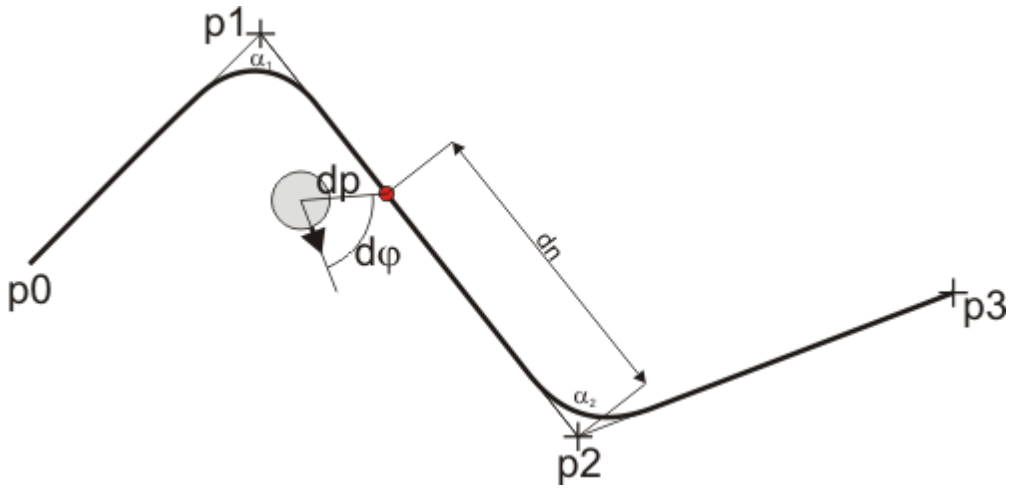
Adaptation du facteur de couplage entre la vitesse calculée en fonction de la configuration des dialogues 1 et 2 et de la vitesse de rotation effective en sortie.

5.8.7.4 Affichage de l'itinéraire





5.8.7.5 Stratégie



Le bloc de fonction Parcoureur d'itinéraire génère un itinéraire qui, dans un premier temps, relie les points de cheminement par des droites.

Le robot est guidé au moyen d'un point de cheminement virtuel (représenté ici par un point rouge). Partant de la position actuelle du robot, le point de cheminement virtuel est placé sur l'itinéraire de sorte que la distance entre le robot et le point de cheminement virtuel soit égale à dp (distance virtual point). Le point de cheminement virtuel ne peut se déplacer sur l'itinéraire qu'en direction de la fin de l'itinéraire, c.-à-d. que si le robot s'éloigne du point de cheminement virtuel, ce dernier reste inchangé. La régulation en fonction du point virtuel se traduit par un lissage de l'itinéraire. Ce lissage est d'autant plus important que la distance dp est grande.

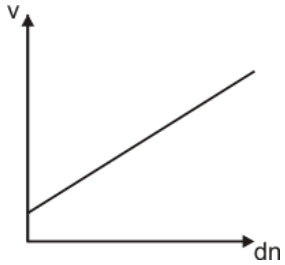
Paramétrage de la vitesse angulaire

Vitesse angulaire ω(d□) est spécifié en fonction de l'erreur angulaire d□ dans le dialogue du bloc de fonction. d□ est l'angle formé par l'orientation actuelle du robot et la ligne reliant le centre du robot au point de cheminement virtuel.

Paramétrage de la vitesse

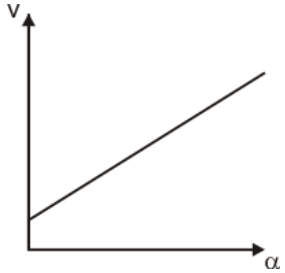
La vitesse est également spécifiée en fonction de d□, à savoir sous v(d□). Ceci permet de réduire la vitesse du robot lorsque le robot n'est pas correctement orienté.

Pour pouvoir réduire la vitesse lorsque l'itinéraire forme un coude, on spécifie également la vitesse en tant que fonction v(dn) de la distance du point virtuel au point de cheminement suivant. L'allure typique de v(dn) est



En d'autres termes, la vitesse doit diminuer au fur et à mesure que le robot se rapproche du point de cheminement (et donc du prochain virage).

On souhaite cependant freiner le robot en fonction de l'angle $\square_n$. $\square_n$ est l'angle formé par le tronçon d'itinéraire actuel et le suivant. Si $\square_n$ est égal à 180° (c.-à-d. si l'itinéraire passe tout droit par le point de cheminement), la vitesse ne doit pas être ralentie. Si $\square_n$ tend vers 0° (une épingle à cheveux), le robot doit être fortement ralenti. C'est la raison pour laquelle on a besoin de la fonction v($\square_n$). L'allure typique de v($\square_n$se présente comme suit :



C.-à-d. que plus □$_n$ est faible, plus la vitesse d'avance sera réduite.

Les trois profils de vitesse v(d□), v(dn) et v(□) sont combinés en une vitesse d'itinéraire globale V(d□,dn, □) :

Vp(d□,dn,□) = min(v(d□), max(v(dn), v(□)))

Accostage du dernier point de cheminement

Pour freiner en fin d'itinéraire, la vitesse est spécifiée en fonction de l'itinéraire qui reste à parcourir, sous la désignation v(d). La destination est supposée atteinte lorsque la vitesse en fonction de l'itinéraire restant à parcourir est nulle.

La vitesse non lissée se calcule selon la formule :

V(d,d□,dn,□) = min(v(d), Vp(d□,dn,□))

Lissage de la vitesse et vitesse angulaire

Il existe deux paramètres supplémentaires pour le lissage du mouvement.

Le **couplage de la vitesse** désigne le temps, en millisecondes, nécessaire pour que le couplage vCC de la vitesse calculée Vp(d□,dn,□) et la vitesse en sortie velocity atteigne la valeur 1.

Le **couplage de vitesse angulaire** désigne le temps, en millisecondes, nécessaire pour que le couplage omegaCC entre la vitesse angulaire calculée ω(d□) et la vitesse angulaire en sortie omega atteigne la valeur 1.

$dv = vCC * (Vp_t - Vp_{t-1})$

$velocity = Vp_{t-1} + dv$

$domega = omegaCC * (\omega(d□)_t - \omega(d□)_{t-1})$

$velocity = \omega(d□)_{t-1} + domega$

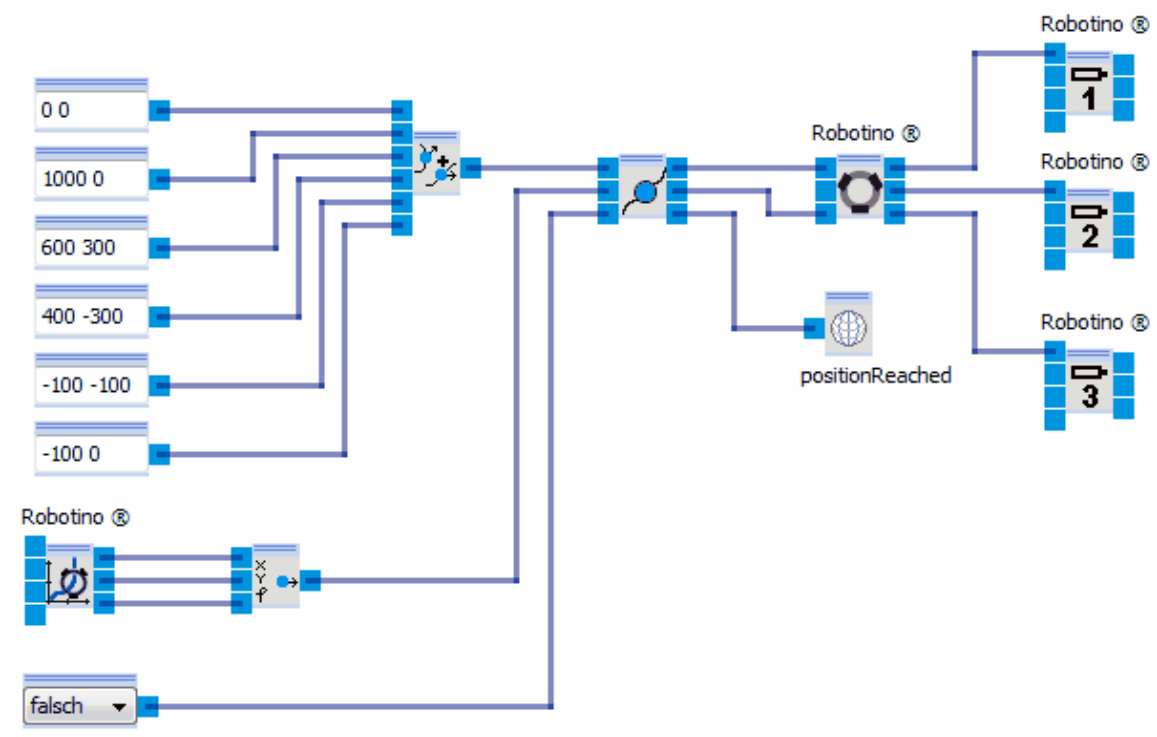
Le t en indice désigne la valeur à l'instant t. t-1 désigne la valeur à un cycle avant t.

Au redémarrage, vCC est mis à 0 et passe à la valeur 1 dans l'espace de temps spécifié par le **couplage de la vitesse** .

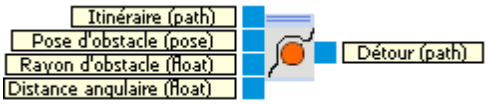
Au redémarrage, omegaCC est mis à 0 et passe à la valeur 1 dans l'espace de temps spécifié par le **couplage de vitesse angulaire** .

vCC et omegaCC sont également remis à 0 lorsque le point virtuel passe à un nouveau tronçon d'itinéraire.

5.8.7.6 Exemple



5.8.8 Évitement d'obstacle

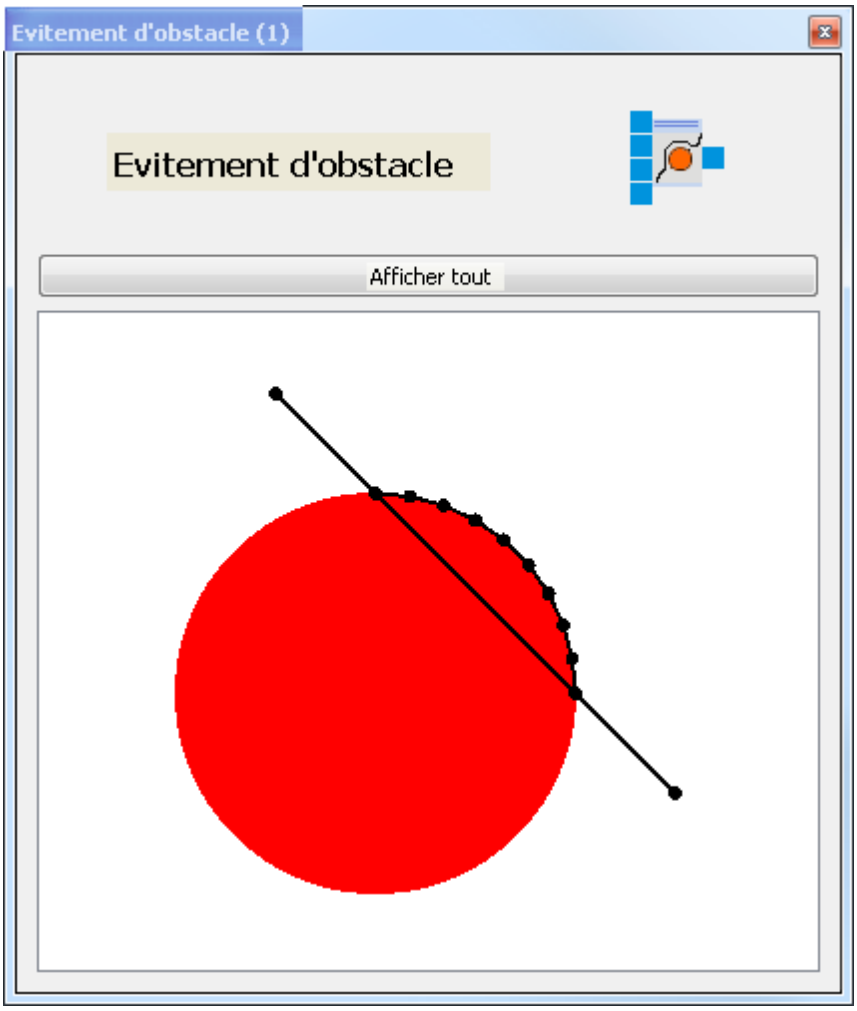


Le module d'évitement d'obstacle calcule, pour un itinéraire donné, une déviation autour d'un obstacle circulaire.

Entrées	Type	Unité	Standard	Description
Course	path		Itinéraire vide	L'itinéraire à parcourir.
Pose de l'obstacle	pose		(0, 0, 0)	La position de l'obstacle circulaire.
Le rayon de l'obstacle	float	mm	100	Le rayon de l'obstacle circulaire.
Distance angulaire	float	Degré	10	La distance angulaire maximale entre deux points de contrôle du détour autour de l'obstacle.

Sorties				
Détour	path		Itinéraire vide	Détour autour de l'obstacle.

5.8.8.1 Dialogue



Ce dialogue affiche l'itinéraire initial, ainsi que l'obstacle et le détour.

5.9 Périphériques d'entrée

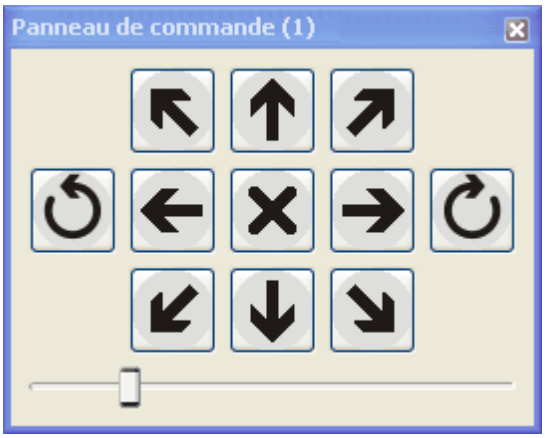
Cette catégorie contient des blocs de fonction assurant l'interactivité avec l'utilisateur.

5.9.1 Panneau de commande

Ce bloc de fonction représente un panneau de commande utilisable avec la souris.

Sorties	Type	Description
vx	float	Vitesse sur l'axe x
vy	float	Vitesse sur l'axe y
omega	float	Vitesse de rotation

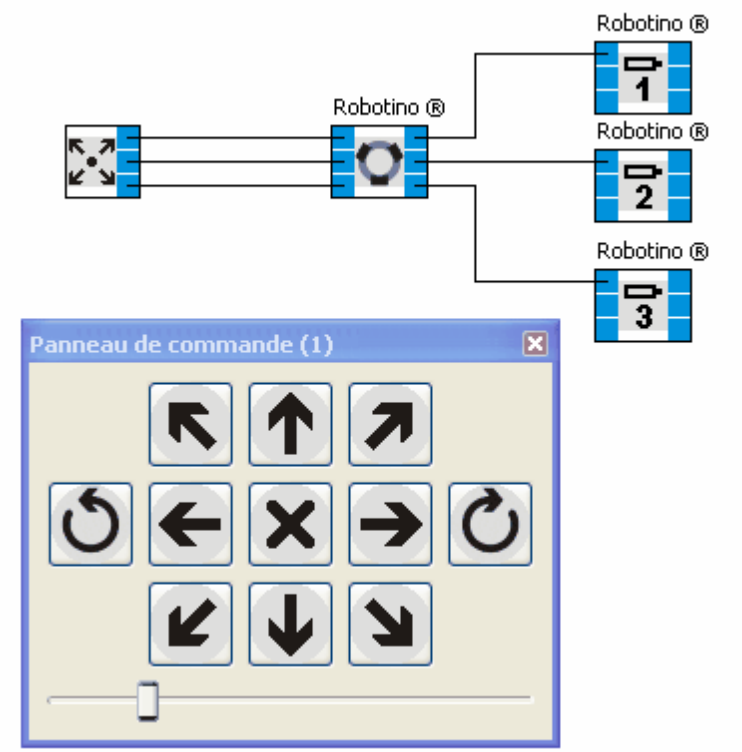
5.9.1.1 Dialogue



Le panneau de commande peut être utilisé comme suit :

- Lorsque vous cliquez sur l'un des boutons, le robot se déplace dans la direction de la flèche du bouton.
- Un clic de souris sur l'un des deux boutons à flèche circulaire déclenche une rotation dans le sens de la flèche.
- Un clic sur le bouton central arrête le mouvement.
- La réglette permet de régler la vitesse du mouvement.

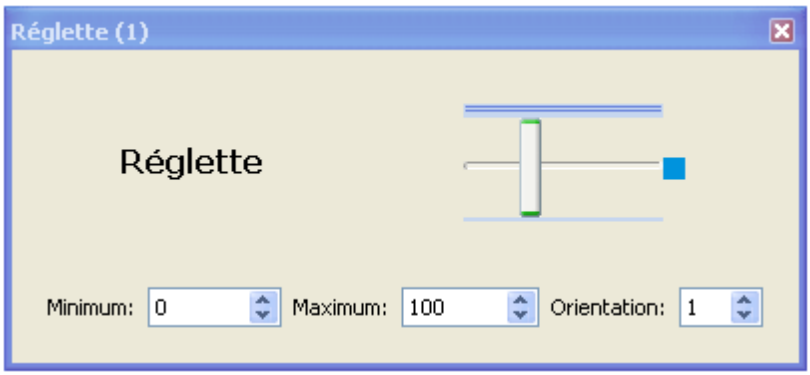
5.9.1.2 Exemple



5.9.2 Réglette

La réglette permet de générer des valeurs entières quelconques dans une plage de valeurs définie.

5.9.2.1 Dialogue



Ce dialogue permet de définir la plage de valeurs ainsi que l'orientation (1 = horizontale, 0 = verticale) de la réglette.

5.10 Échange de données

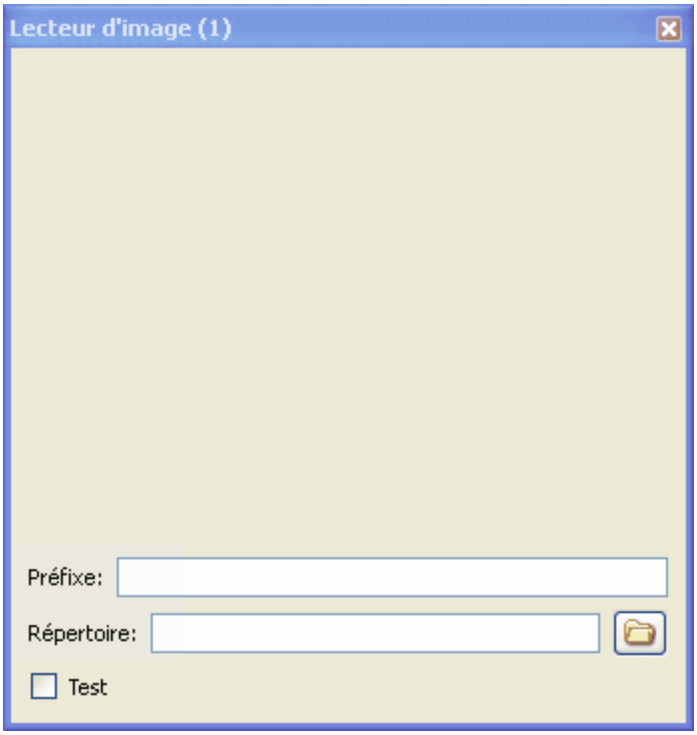
Cette catégorie contient des blocs de fonction permettant d'échanger des données au sein de Robotino® View ou avec des applications externes.

5.10.1 Lecteur d'image

Le lecteur d'image lit les images JPEG d'une séquence d'images enregistrée dans le système de fichiers. Le chemin et le préfixe peuvent être spécifiés dans une boîte de dialogue 120.

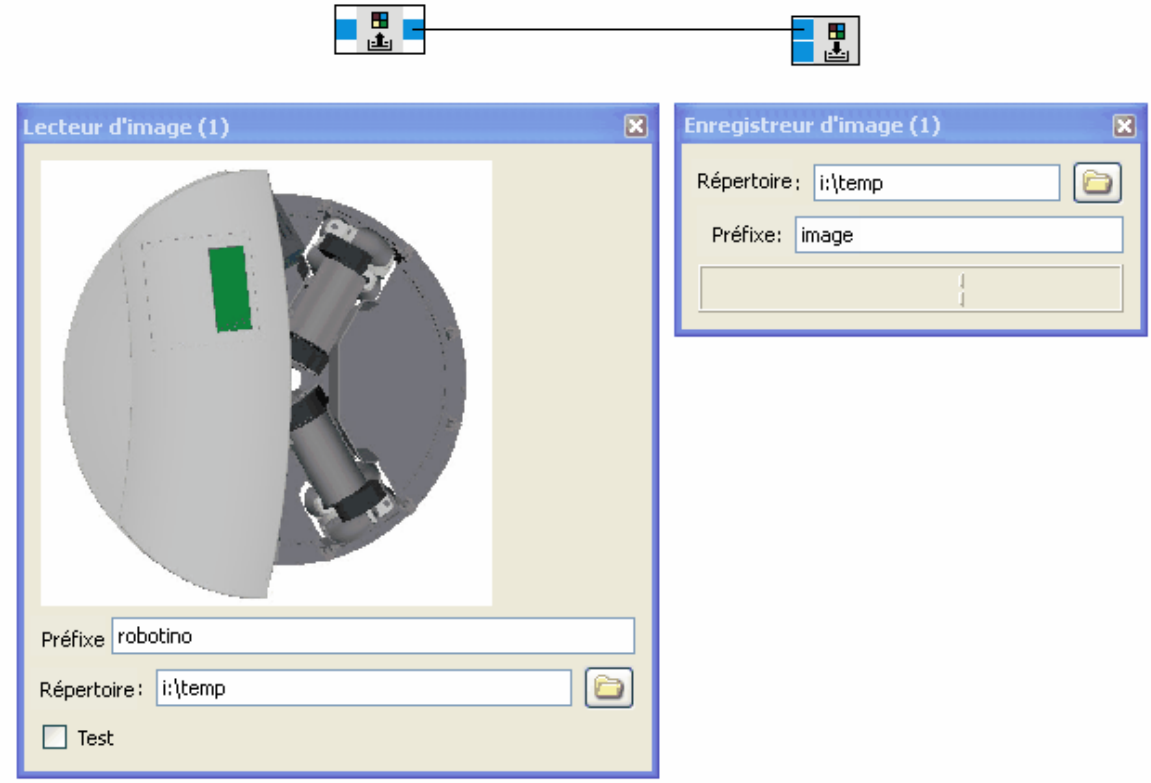
Entrées	Type	Standard	Description
Numéro	int16	-1	Numéro de l'image voulue de la séquence. Si le numéro = -1, la numérotation débute par 0 et est automatiquement incrémentée de 1 à chaque étape.
Sorties			
Sortie	image		Image JPEG du fichier "‹chemin›/‹préfixe›‹numéro›.jpg" ou "‹chemin›/‹préfixe›_‹numéro›.jpg". Si le fichier est introuvable, le numéro sera complété à gauche par des zéros (numéro de 4 chiffres max.), jusqu'à ce qu'un fichier d'image soit trouvé.

5.10.1.1 Dialogue

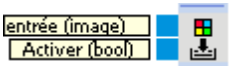


Ce dialogue permet de spécifier le chemin et le préfixe de la séquence d'images à lire.

5.10.1.2 Exemple



5.10.2 Enregistreur d'image

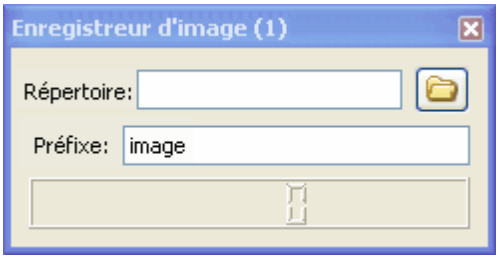


L'enregistreur d'image enregistre une séquence d'images JPEG dans le système de fichiers. Le chemin et le préfixe peuvent être spécifiés dans une boîte de dialogue 122. La numérotation des images débute par 0 et est incrémentée de 1 à chaque étape.

Chaque image est enregistrée dans le chemin "‹chemin›/‹préfixe›_‹numéro›.jpg", le numéro comptant, avec les zéros à gauche, au moins 4 chiffres.

Entrées	Type	Standard	Description
Entrée	image		Image suivante de la séquence
Actif	bool	true	L'enregistreur d'image est actif.

5.10.2.1 Dialogue



5.10.2.2 Exemple

Voir Exemple à propos du lecteur d'image 121.

5.11 Variables

Les variables globales occupent une position particulière. Des blocs de fonction d'écriture et de lecture sont disponibles pour toutes les variables globales et dans chaque sous-programme. Ces blocs de fonction indiquent toujours dans les sous-programmes le nom de variable et ne peuvent donc pas être renommés.

Les variables globales peuvent être créées, supprimées et dotées d'une valeur initiale dans le gestionnaire de variables (affichage programme principal) 15.

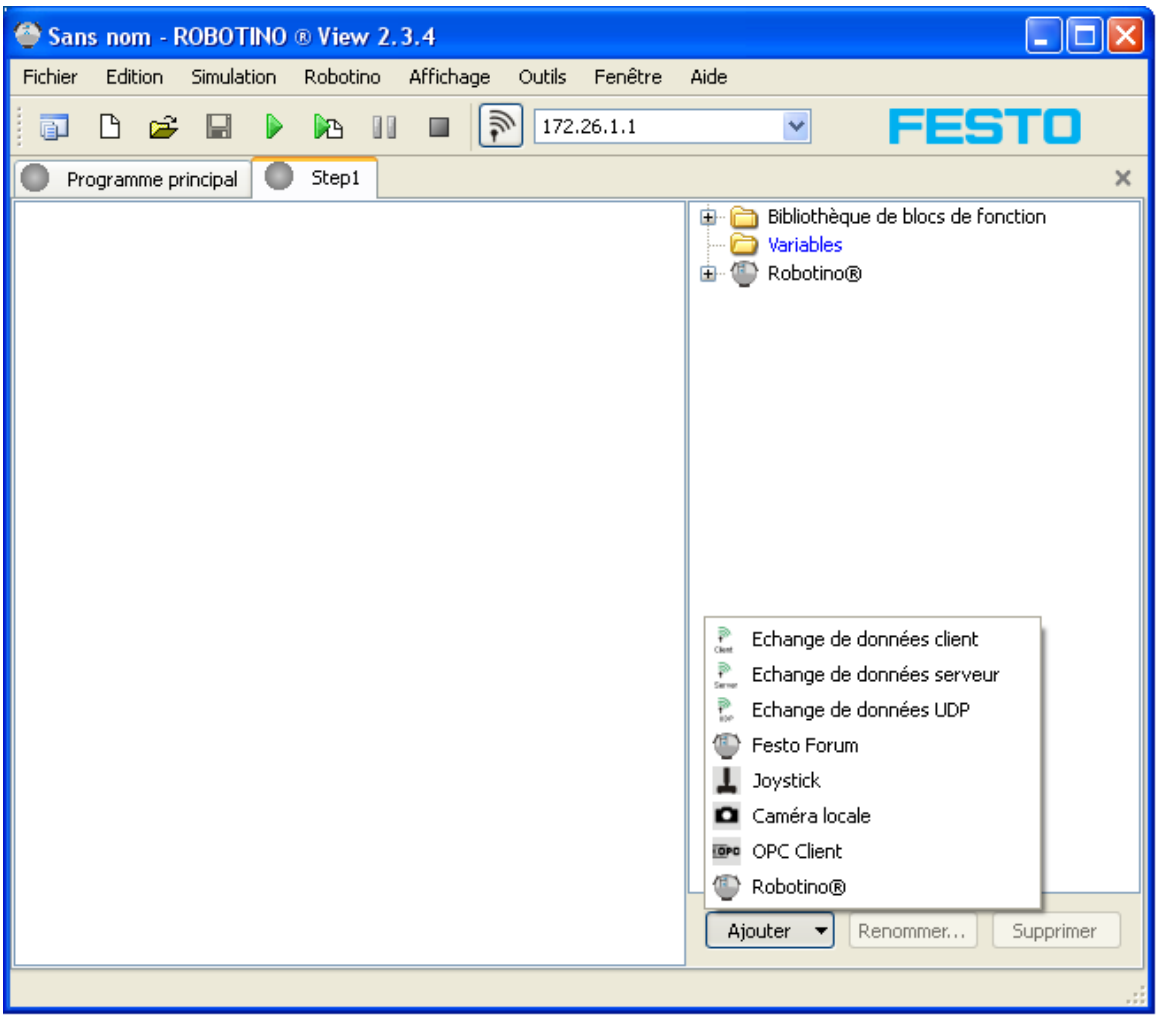
Il est également possible de créer, de supprimer et de renommer des variables globales dans la bibliothèque de blocs de fonction par un clic avec le bouton droit de la souris sur le dispositif "Variables" et sélection de la commande "Ajouter" ou par un clic sur le lecteur ou enregistreur d'une variable globale et sélection de la commande "Supprimer" ou "Renommer".

6 Dispositifs

Ils établissent la liaison entre Robotino View et l'environnement. Le dispositif "Robotino" peut communiquer avec un Robotino réel ou simulé. Le dispositif "Joystick" permet de lire la position des axes d'une manette connectée à l'ordinateur.

6.1 Créer et éditer

Un "Robotino" est présent dans tout nouveau projet. Pour créer d'autres dispositifs, il faut passer dans un sous-programme.



Sous la bibliothèque de blocs de fonction figure le bouton "Ajouter" qui permet d'ajouter de nouveaux dispositifs. Le dispositif sélectionné apparaît alors sous Robotino dans la bibliothèque de blocs de fonction.

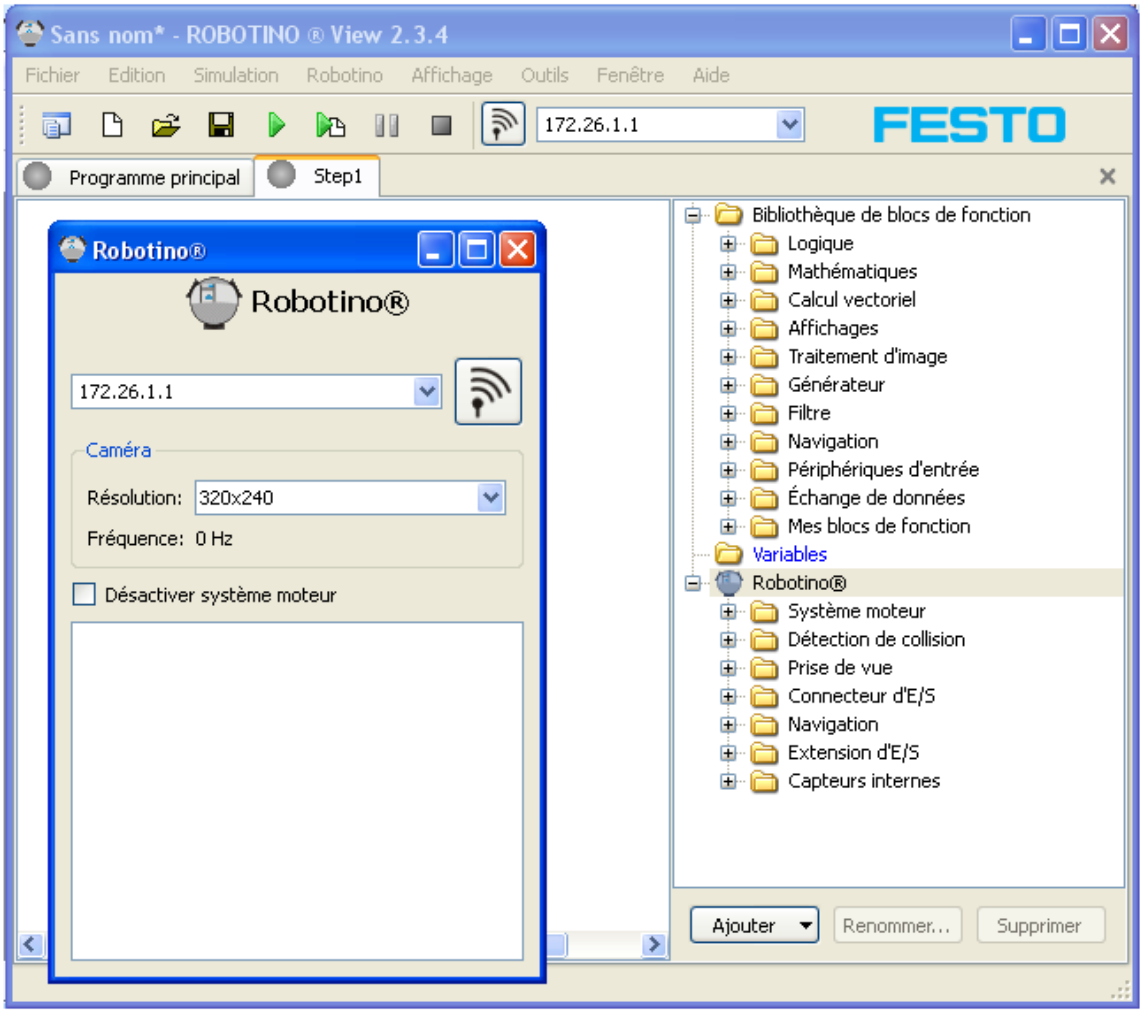
Un nom unique est affecté aux nouveaux dispositifs. Ce nom peut être modifié à l'aide du bouton "Renommer" à condition d'avoir sélectionné au préalable le dispositif en question dans la bibliothèque de blocs de fonction.

Le bouton "Supprimer" permet de supprimer des dispositifs de la bibliothèque de blocs de fonction. Cette fonction est uniquement disponible si aucun bloc de fonction du dispositif n'est utilisé dans le projet.

6.2 Ouvrir une boîte de dialogue

Il existe, pour chaque dispositif, une boîte de dialogue qui permet d'effectuer des paramétrages.

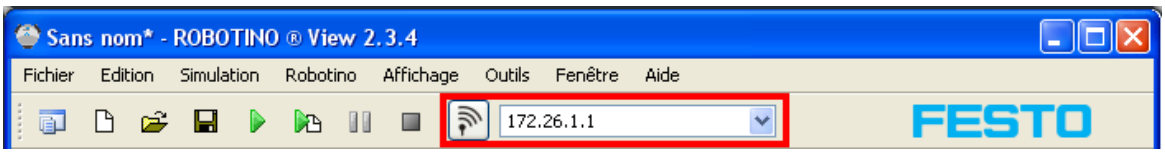
Pour afficher une boîte de dialogue, il suffit de double cliquer sur le dispositif voulu dans la bibliothèque de blocs de fonction.



6.3 Robotino

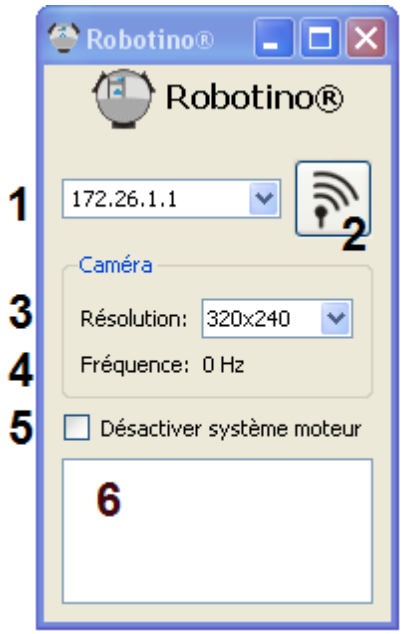
Le dispositif "Robotino" permet d'accéder aux capteurs et actionneurs d'un système robotique Robotino®.

6.3.1 Barre d'outils



La barre d'outils de Robotino® View affiche l'adresse IP et le bouton de connexion du dispositif Robotino® figurant en premier dans le gestionnaire de dispositifs (zone encadrée de rouge). La fonctionnalité de la saisie d'adresse IP et du bouton de connexion est la même que celle de la boîte de dialogue du gestionnaire de dispositifs.

6.3.2 Dialogue



La boîte de dialogue du dispositif Robotino s'affiche après un double clic sur Robotino.

1	Entrée de l'adresse IP	L'adresse par défaut de Robotino est 172.26.1.1. Dans le cas d'une simulation locale de Robotino l'adresse I est 127.0.0.1:8080, dans laquelle 8080 désigne le numéro de port. En présence de plusieurs Robotino dans une simulation, le numéro de port peut être plus élevé.
2	Bouton de connexion	Appuyez sur ce bouton pour établir ou couper une connexion à Robotino.
3	Résolution	Il s'agit de la résolution des images enregistrées par Robotino.
4	Fréquence	Il s'agit de la fréquence de réception des images.
5	Désactiver système moteur	Cochez cette case pour désactiver les moteurs de Robotino.
6	Fenêtre de message	Champ de texte d'affichage de messages.

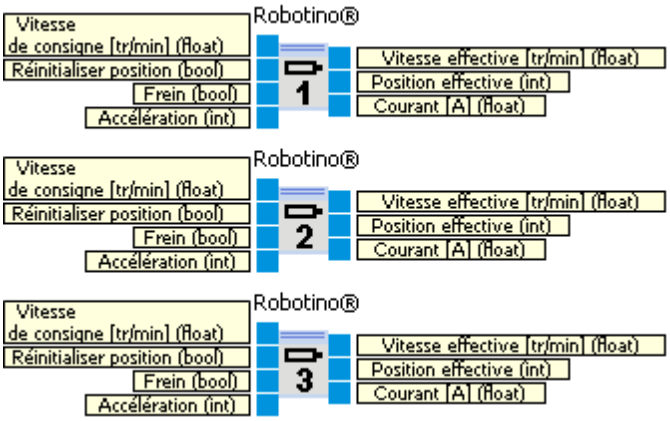
6.3.3 Blocs de fonction

Les blocs de fonction permettent d'utiliser le dispositif Robotino dans un sous-programme.

6.3.3.1 Système moteur

Ce dossier contient des blocs de fonction pour la commande d'entraînement de Robotino.

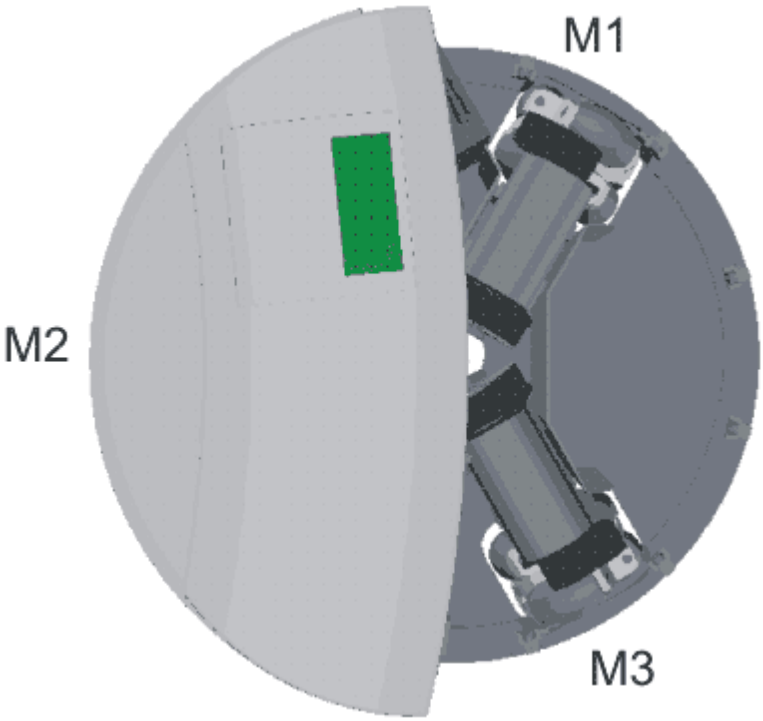
6.3.3.1.1 Moteur



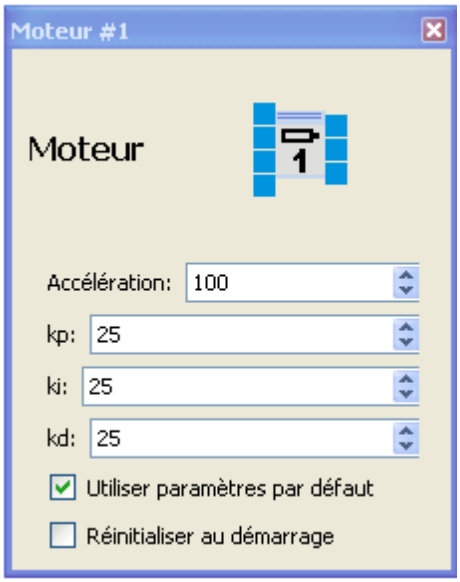
Ce bloc de fonction représente l'un des trois moteurs de Robotino. Le numéro du moteur figure dans le symbole graphique du bloc de fonction.

.

Entrées	**Type**	**Unité**	**Standard**	**Description**
Vitesse de consigne	float	tr/min	0	Vitesse de consigne du moteur en tours par minute (tr/min). Veuillez noter qu'un réducteur 16:1 est monté entre le moteur et la roue.
Mise à 0 de la position	bool		false	Si vrai (true), le compteur d'impulsions du codeur du moteur est mis à 0.
Frein	bool		false	Si vrai (true), le moteur est arrêté.
Accélération	int		100	Couplage de la vitesse de consigne à l'entrée et de la vitesse de consigne effectivement transmise (voir aussi Dialogue 127).
Sorties				
Vitesse	float	tr/min		La vitesse réelle du moteur
Position	int			Le nombre d'impulsions comptées du codeur implanté sur l'axe du moteur depuis la mise en marche de Robotino ou depuis la dernière transition de l'entrée "Mise à zéro de la position" de vrai (true) à faux (false). Le codeur génère 2000 impulsions par tour de moteur. Peut être utilisée pour obtenir la position réelle de la roue motrice.
Courant	float	A		Le courant absorbé par le moteur en A.



6.3.3.1.1 Dialogue



Paramètre	Description
Accélération	Facteur d'accélération ou de décélération. La valeur maximale 100 transmet directement la vitesse de consigne au moteur. Les valeurs inférieures se traduisent par une rampe d'accélération. Ceci permet d'obtenir des mouvements moins saccadés.

kp	Composante proportionnelle du régulateur PID monté en amont du moteur
ki	Composante intégrale du régulateur PID monté en amont du moteur
kd	Composante différentielle du régulateur PID monté en amont du moteur
Utiliser paramètres par défaut	Utilise les valeurs de kp, ki et kd enregistrées dans le firmware de la carte d'E/S de Robotino. Les paramètres par défaut sont également utilisés si kp=ki=kd=255.
Réinitialiser au démarrage	Mise à 0 de la valeur de position à chaque démarrage du programme

La régulation de vitesse de chaque moteur est assurée par un régulateur PID.

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

Les paramètres sont :

K_p

$K_i = 1/T_n$

K_d

Les valeurs paramétrables dans la boîte de dialogue sont converties en paramètres utilisables par le régulateur :

$K_p = kp / 2$

$K_i = ki / 1024$

$K_d = kd / 2$

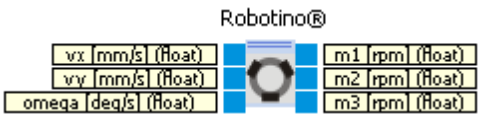
Les valeurs par défaut sont :

kp = 25

ki = 25

kd = 25

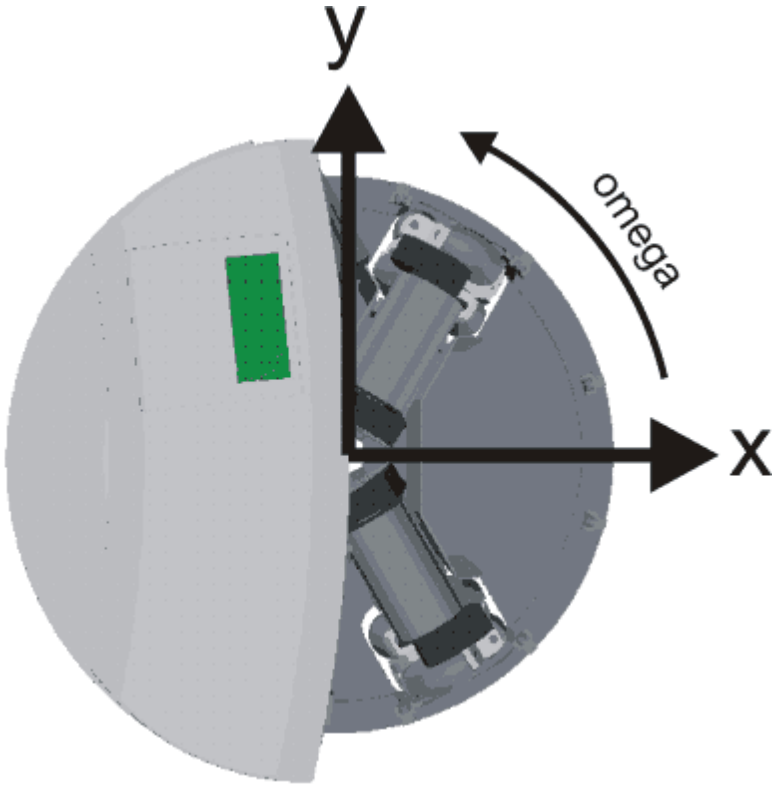
6.3.3.1.2 Entraînement omnidirectionnel



Calcule la vitesse de consigne des moteurs 1,2 et 3 sur la base de la consigne de vitesse dans l'axe x et dans l'axe y ainsi que d'une consigne de vitesse de rotation.

Entrées	Type	Unité	Standard	Description
vx	float	mm/s	0	Vitesse de consigne dans l'axe x.
Réinitialiser vy	float	mm/s	0	Vitesse de consigne dans l'axe y
omega	float	deg/s	0	Vitesse de consigne en rotation
Sorties				
m1	float	tr/min		Vitesse de consigne du moteur 1
m2	float	tr/min		Vitesse de consigne du moteur 2
m3	float	tr/min		Vitesse de consigne du moteur 3

Le bloc de fonction "Entraînement omnidirectionnel (inverse)" calcule vx, vy et omega à partir des vitesses de rotation des moteurs.



L'illustration montre le système de coordonnées local de Robotino. Une valeur positive de la vitesse de rotation oméga génère, vu de dessus, une rotation dans le sens horaire.

6.3.3.2 Détection de collision

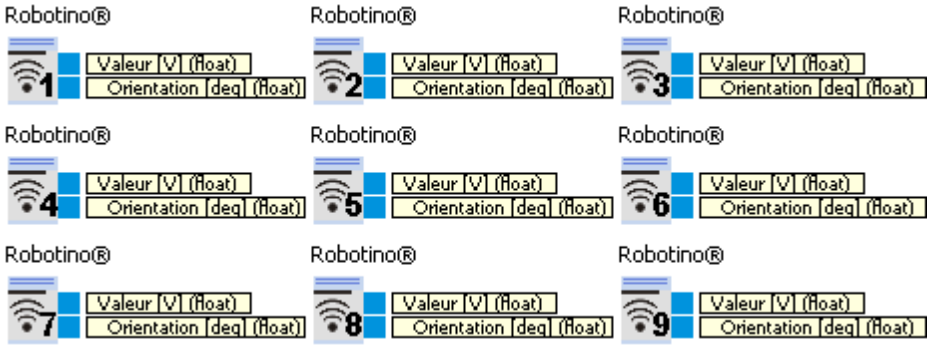
Ce dossier contient des blocs de fonction qui permettent de détecter des objets.

6.3.3.2.1 Parechoc

Un capteur de contact est intégré au parechoc. Le capteur délivre un signal en cas de contact.

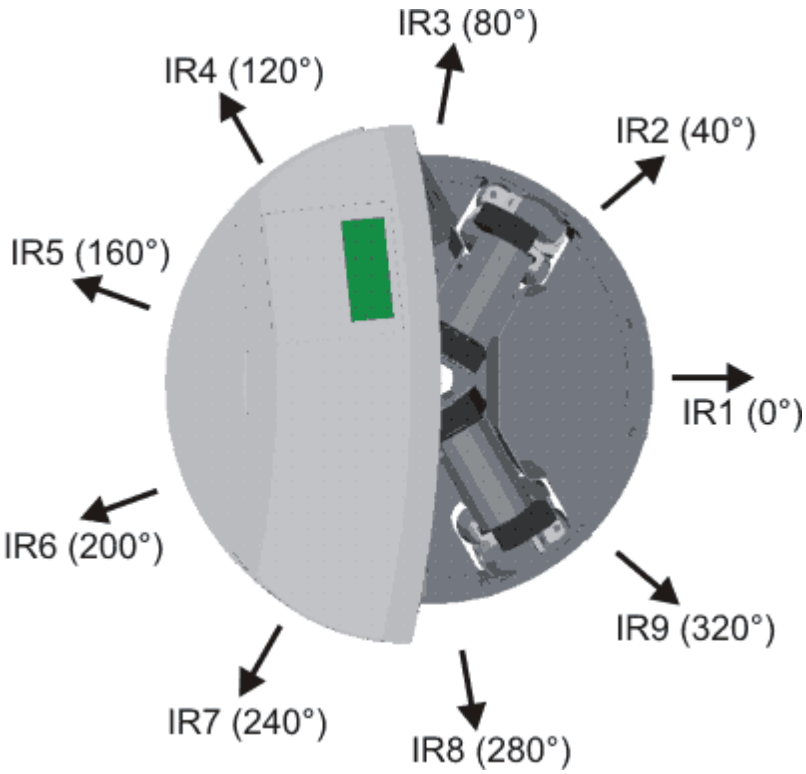
Entrées	Type	Standard	Description
Sorties			
Valeur	bool		Vrai (true) lorsqu'un objet exerce une pression sur le parechoc. Sinon faux (false).

6.3.3.2.2 Capteurs de distance



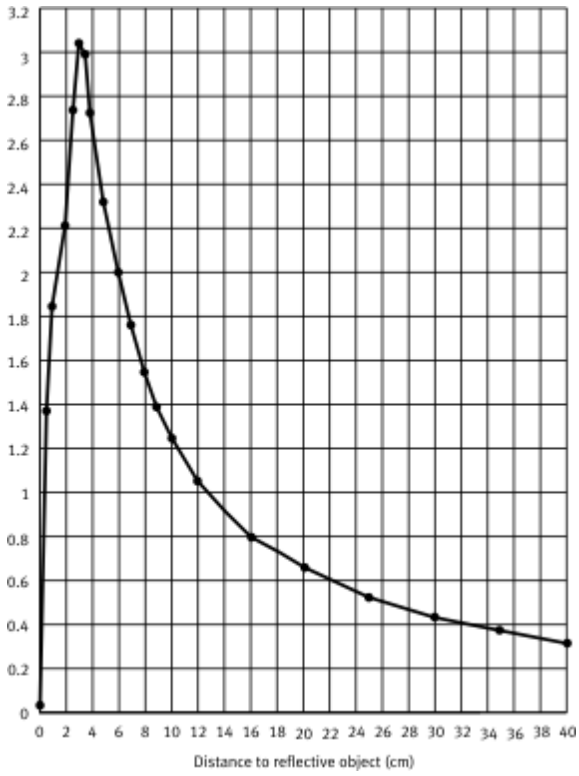
Fournit la valeur du capteur de distance.

Entrées	Type	Unité	Standard	Description
Sorties				
Valeur	float	Volt		Indique, en V, la tension analogique délivrée par le capteur. La mise à l'échelle et la conversion de ces valeurs en une valeur de distance ayant la dimension d'une longueur doivent être réalisées par l'utilisateur.
Orientation	float	Degré		Orientation du capteur dans le système de coordonnées local de Robotino. La valeur se calcule avec le numéro du capteur de distance selon la formule orientation = 40° x (numéro - 1)



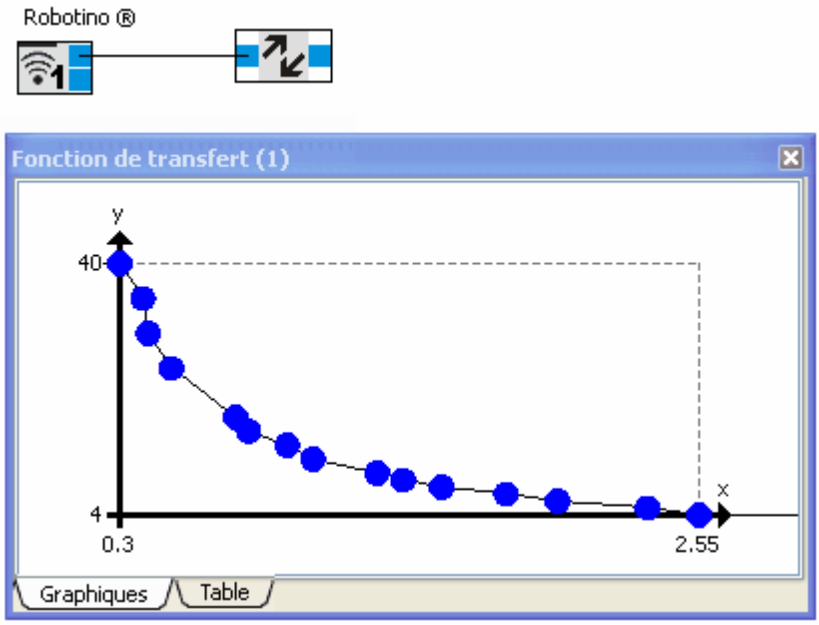
6.3.3.2.2 Exemple

Dans la fiche technique du capteur de distance (un Sharp GP2D120), on trouve la courbe ci-après illustrant la relation entre la distance de l'objet en cm et le signal de sortie analogique en volts.



Cette courbe est facile à reproduire avec une fonction de transfert[61] ce qui permet d'obtenir la conversion voulue de la tension analogique en distance de l'objet en cm. Veuillez noter à ce propos que la fonction de transfert représente la fonction inverse et que la tension analogique est représentée sur l'axe x en fonction de la distance en cm sur l'axe y. Pour que la fonction soit univoque, on ne prend en compte que les valeurs de distance supérieures à 4 cm. Pour les distances inférieures, la tension analogique ne permet pas de déterminer si l'objet se trouve plus ou moins loin (distance ‹ 4 cm).

Notez aussi que le convertisseur AN auquel le capteur est connecté, ne mesure que des tensions jusqu'à 2,55 V. Étant donné que le rapport entre la tension analogique et la distance varie aussi en fonction du matériau de l'objet réfléchissant, on vérifiera de préférence soi-même en effectuant une mesure et en utilisant les valeurs ainsi mesurées pour la conversion.



Les valeurs de la fonction de transfert 61 sont indiquées en bas. Vous pouvez les copier et les entrer dans votre fonction de transfert 61 personnelle.

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 Prise de vue

Ce dossier contient des blocs de fonction qui permettent d'utiliser la caméra de Robotino.

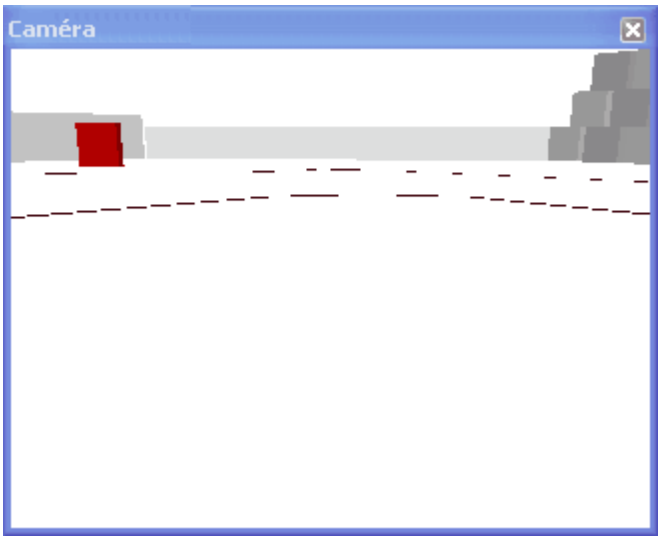
6.3.3.3.1 Caméra

Fournit l'image en direct de la caméra montée sur Robotino.

Entrées	Type	Standard	Description
Sorties			
Image	image		Image en direct de la caméra sur Robotino.

Les paramètres de profondeur de la couleur et de résolution sont définis dans la boîte de dialogue 125 de Robotino.

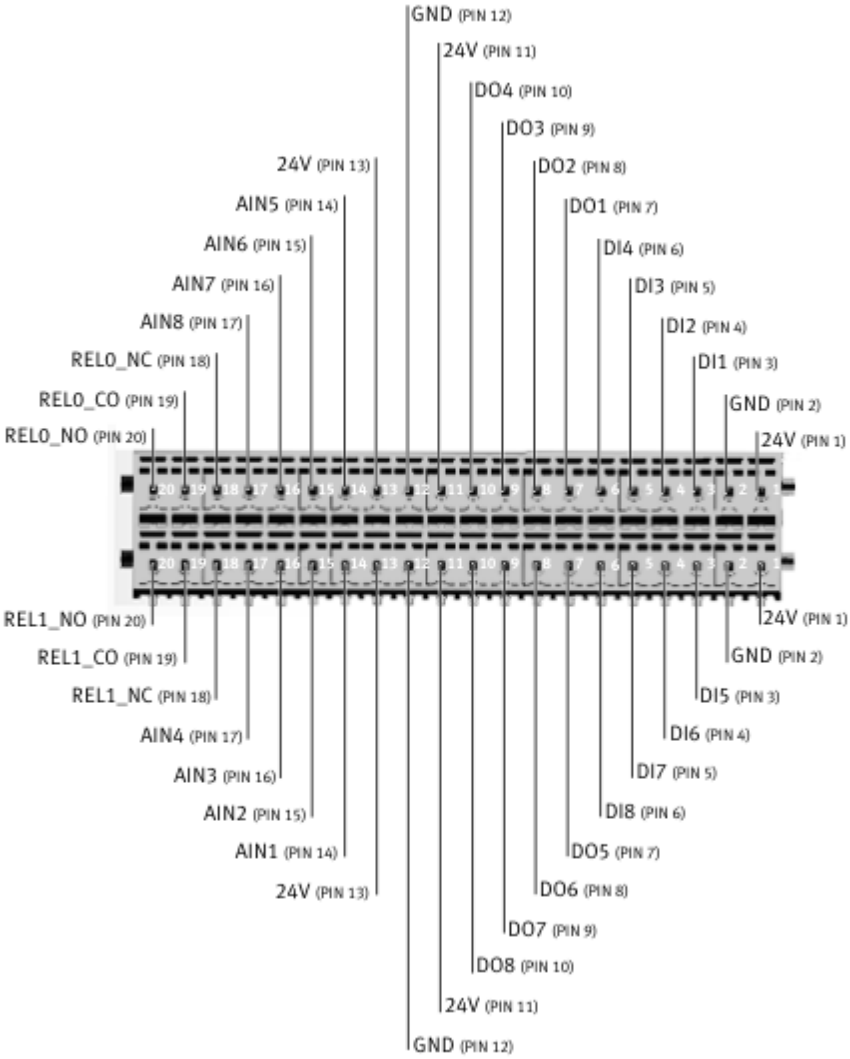
6.3.3.3.1 Dialogue



La boîte de dialogue de la caméra affiche l'image actuelle de la caméra. Pour modifier les paramètres d'image voir boîte de dialogue 125.

6.3.3.4 Connecteur d'E/S

Ce dossier contient des blocs de fonction qui permettent d'utiliser le connecteur d'E/S de Robotino.

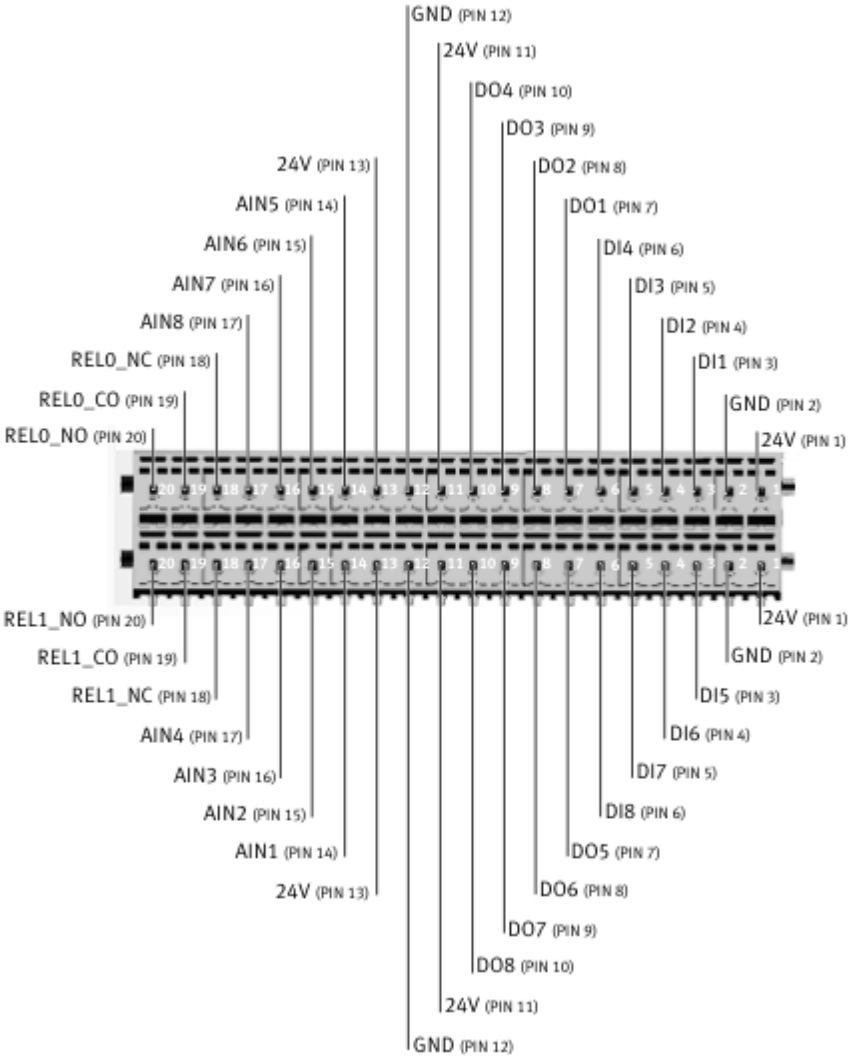


6.3.3.4.1 Relais

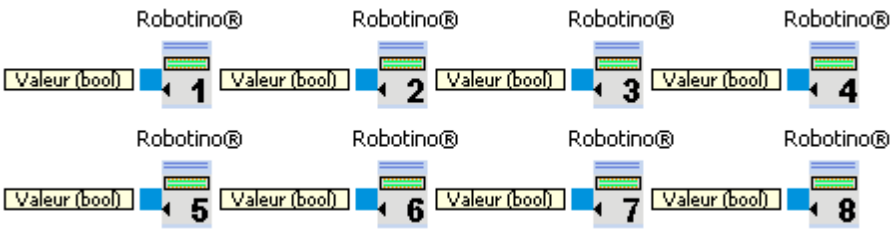
Fait commuter les relais 1 et 2.

Entrées	Type	Standard	Description
Valeur	bool	false	Met le relais sélectionné au repos si faux (false), sinon il est mis au travail.

Les contacts de relais sont pour le relais 1 REL1_NO, REL1_INV et REL1_NF.
Les contacts de relais sont pour le relais 2 REL2_NO, REL2_INV et REL2_NF.



6.3.3.4.2 Sortie TOR



Permet d'activer une sortie TOR.

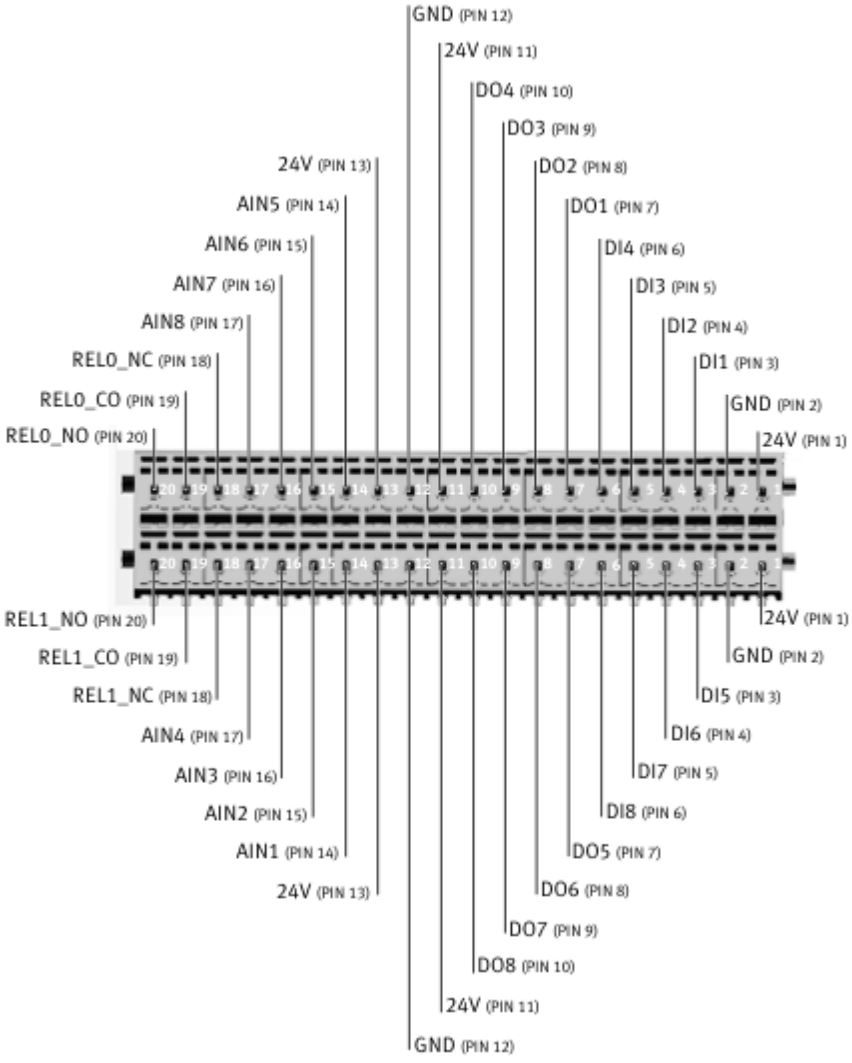
Entrées	Type	Standard	Description
Valeur	bool	false	Si vrai (true), la sortie du connecteur d'E/S de Robotino délivre +10 V. Si faux (false), la sortie du connecteur d'E/S de Robotino délivre 0 V.

La connexion de la sortie TOR 1 est DO1.

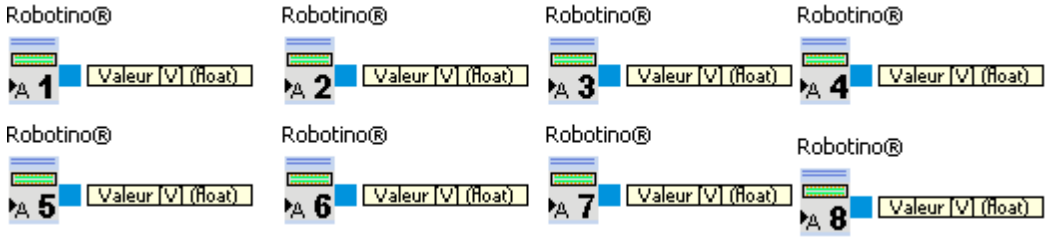
La connexion de la sortie TOR 2 est DO2.

et ainsi de suite jusqu'à

La connexion de la sortie TOR 8 est DO8.



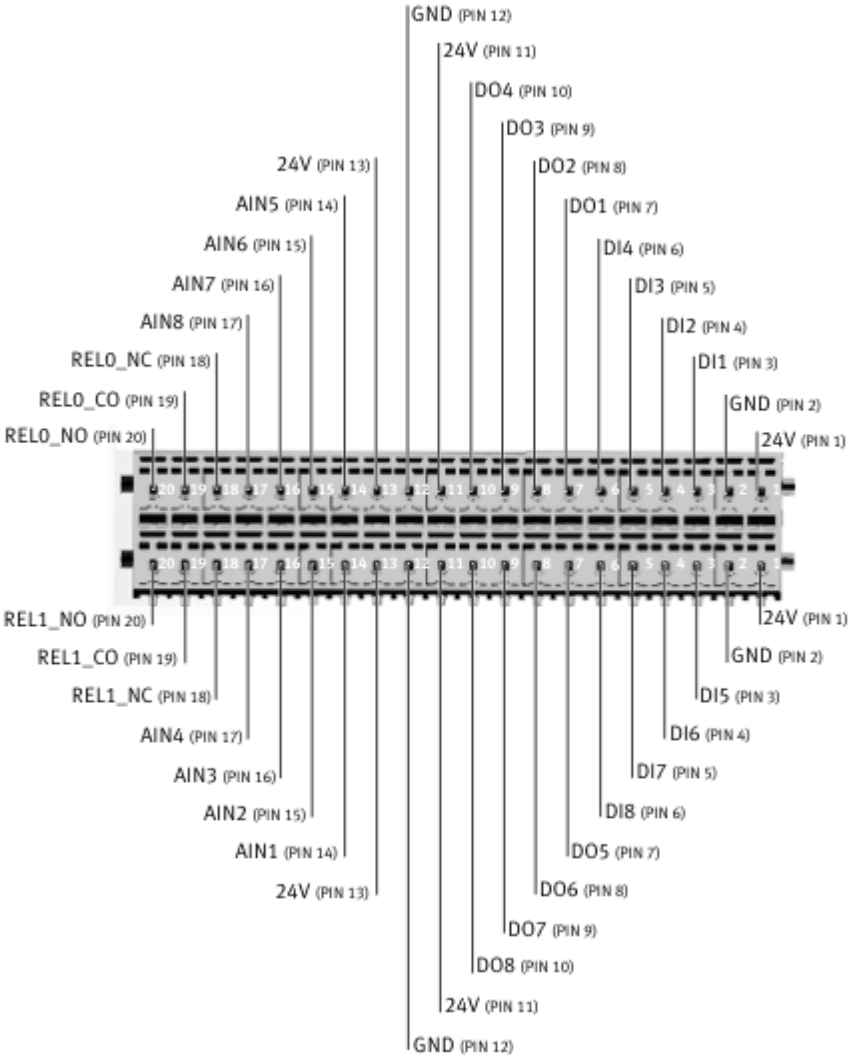
6.3.3.4.3 Entrée analogique



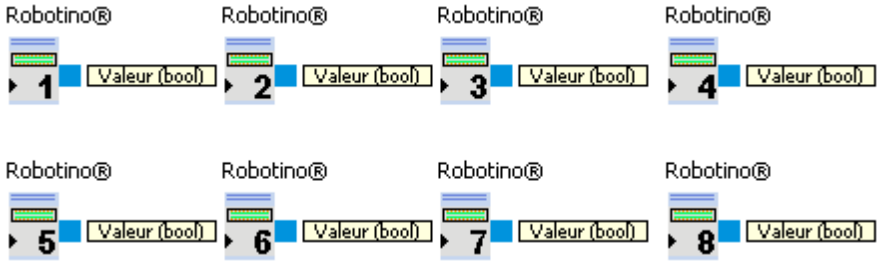
Fournit la valeur d'une entrée analogique.

Sorties	Type	Unité	Description
Valeur	float	Volt	La tension de 0 V à 10 V mesurée au connecteur d'E/S de Robotino.

La connexion de l'entrée analogique 1 est AIN1.
La connexion de l'entrée analogique 2 est AIN2.
et ainsi de suite jusqu'à
La connexion de l'entrée analogique 8 est AIN8.



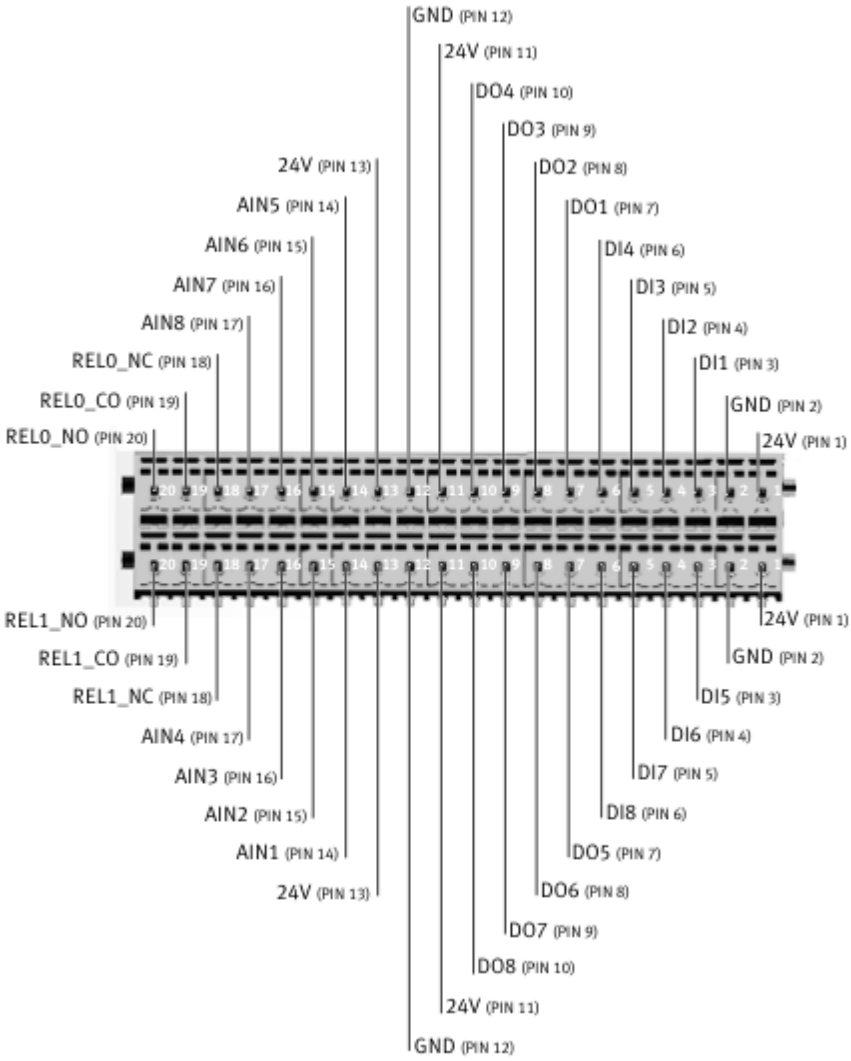
6.3.3.4.4 Entrée TOR



Fournit la valeur d'une entrée TOR.

Sorties	Type	Description
Valeur	bool	La valeur appliquée au connecteur d'E/S de Robotino. Une tension inférieure à 5,75 V équivaut à faux (false). Une tension supérieure à à 8,6 V équivaut à vrai (true) Si la tension à l'entrée se situe entre 5,75 V et 8,6 V, la valeur reste inchangée.

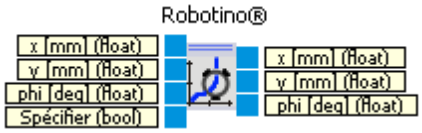
La connexion de l'entrée TOR 1 est DIN1.
La connexion de l'entrée TOR 2 est DIN2.
et ainsi de suite jusqu'à
La connexion de l'entrée TOR 8 est DIN8.



6.3.3.5 Navigation

Ce dossier contient des blocs de fonction pour la localisation de Robotino.

6.3.3.5.1 Odométrie



Cette fonction nécessite la carte mémoire Compact Flash 1 Go de Robotino (V 1.7 ou suivante).

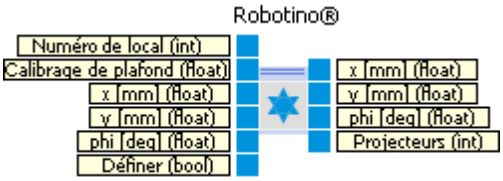
(ne fonctionne pas avec la carte mémoire de 256 Mo, version <=1.6)

L'odométrie désigne le calcul de la position momentanée d'un véhicule en fonction des rotations effectuées par les roues. Le mot odométrie vient des mots grecs "hodos" (qui signifie "voyage") et "metron" (qui signifie "mesure"). Voir aussi http://fr.wikipedia.org/wiki/Odom%C3%A9trie.

La rotation des roues est mesurée à une fréquence si possible élevée. La distance parcourue est calculée en fonction de la vitesse de la roue à chaque intervalle de temps. Les (très petites) distances parcourues au cours des intervalles de temps sont additionnées ce qui donne la position actuelle par rapport au point de départ. Cette méthode fournit localement de bons résultats. Sur de longs parcours ou dans des conditions difficiles (patinage des roues dû à la poussière au sol, dérive du robot en raison d'une direction préférentielle du tapis, etc.), les résultats obtenus avec cette méthode varient fortement. C'est la raison pour laquelle l'odométrie est presque toujours combinée à d'autres méthodes pour compenser les erreurs.

Entrées	**Type**	**Unité**	**Standard**	**Description**
x	float	mm	0	La nouvelle position x La valeur est reprise si l'entrée "Définir" est à l'état vrai (true).
y	float	mm	0	La nouvelle position y La valeur est reprise si l'entrée "Définir" est à l'état vrai (true).
phi	float	Degré	0	La nouvelle orientation. La valeur est reprise si l'entrée "Définir" est à l'état vrai (true).
Définir	bool		false	Si l'entrée est à l'état vrai (true), l'odométrie de Robotino recopie les valeurs des entrées x, y, phi. Pou mettre l'odométrie à (0,0,0), il suffit donc de mettre l'entrée à l'état vrai (true) pendant un cycle. Si l'entrée est à l'état faux (false), l'odométrie ne change que si les roues de Robotino bougent.
Sorties				
x	float	mm		La position x courante
y	float	mm		La position y courante
phi	float	Degré		L'orientation courante

6.3.3.5.2 North Star



North Star ® est un capteur qui détecte, à l'aide de projecteurs, la position absolue de Robotino®.

Entrées	**Type**	**Unité**	**Standard**	**Description**
Numéro de local	int		1	Le numéro du local dans lequel Robotino se trouve. Les locaux sont comptés en commençant par 1.
Calibrage du plafond	float	mm	1	Distance entre le détecteur et le plafond. En cas de hauteur de plafond de 3 m, la distance est d'environ 2800 mm.
x	float	mm	0	Position x de l'origine définie par l'entrée "Définir".
y	float	mm	0	Position y de l'origine définie par l'entrée "Définir".
phi	float	Degré	0	Orientation de l'origine définie par l'entrée "Définir".
Définir	bool		false	Si vrai (true) la pose (x,y,phi) est prise pour origine.
Sorties				
x	float	mm		La position x courante
y	float	mm		La position y courante
phi	float	Degré		L'orientation courante
Projecteurs	int			Nombre de projecteurs visibles.

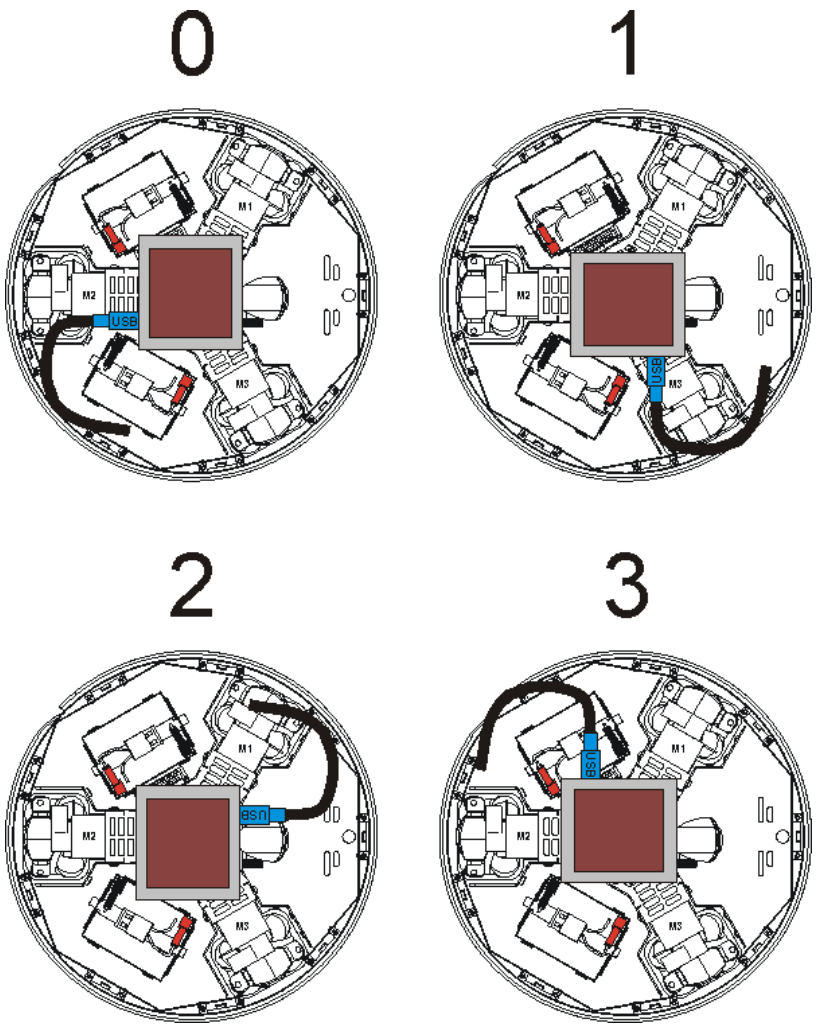
Room ID	Projector type	Switch-Setting	Frequency [Hz]
0	ProjectorKit NS1	Spot 1 SW2 = 0 Spot 2 SW2 = 8	1040 2350
1	ProjectorKit NS1	Spot 1 SW2 = 1 Spot 2 SW2 = 9	1150 2450
2	ProjectorKit NS1	Spot 1 SW2 = 2 Spot 2 SW2 = A	1250 2560
3	**Projector NS2-50Hz**	**1**	**Spot A 3025 Spot B 3925**
4	**Projector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


ProjectorKit NS1

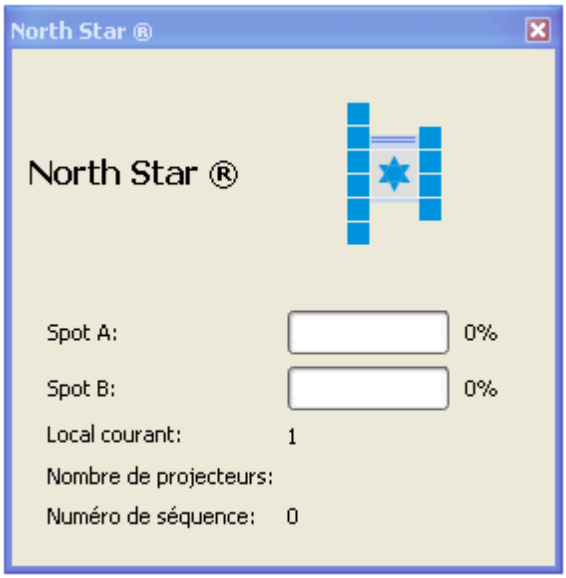

Projector NS2

Le détecteur Northstar peut être monté de différentes manières sur Robotino. Selon le montage, le fichier /etc/robotino/robotino.xml devra être adapté à Robotino dans un éditeur. La valeur de l'orientation doit être réglée en fonction du graphique ci-dessous.

```
<NorthStar>
        <!--The orientation of the northstar sensor. See www.openrobotino.org-->
       <Orientation value="1" />
</NorthStar>
```

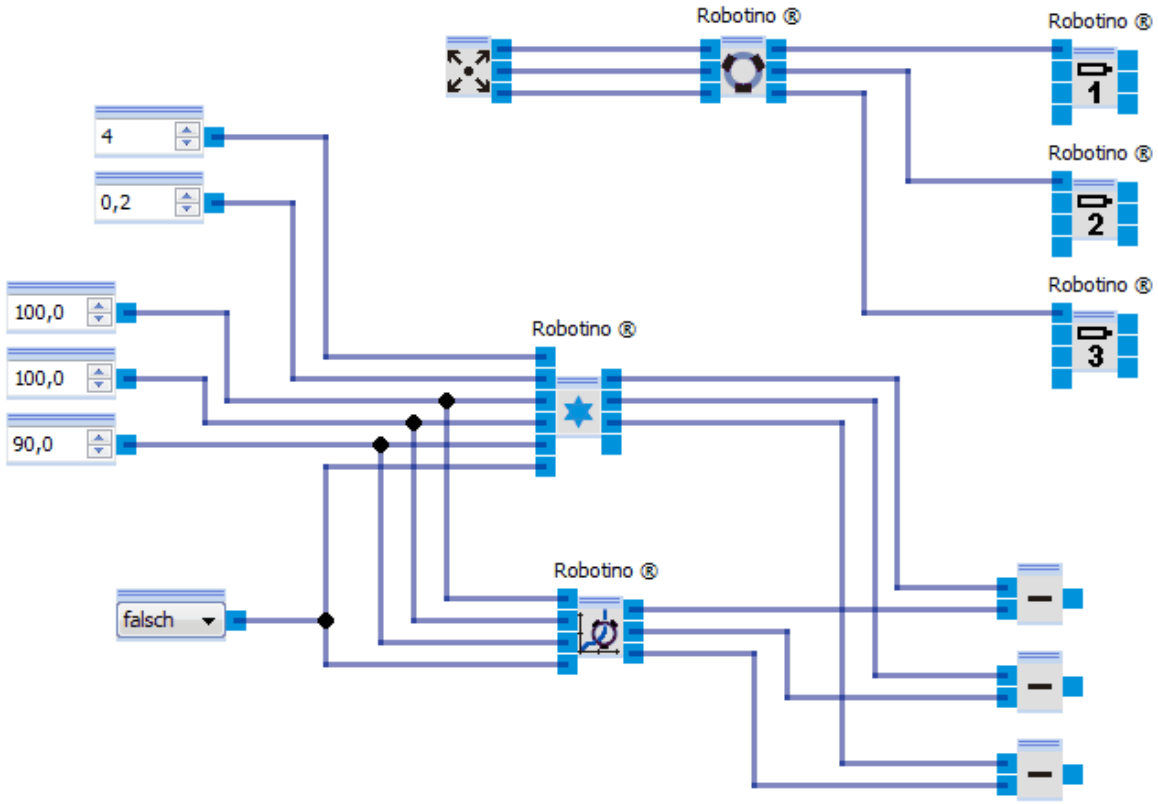


6.3.3.5.2 Dialogue



Spot A	Intensité du premier point lumineux émis par le projecteur.
Spot B	Intensité du deuxième point lumineux émis par le projecteur.
Local courant	Le local détecté par NorthStar.
Nombre de projecteurs	Le nombre de projecteurs visibles par le capteur NorthStar.
Numéro de séquence	Le numéro de séquence augmente de 1 à chaque fois le capteur NorthStar transmet une nouvelle estimation de position.

6.3.3.5.2 Exemple



Le panneau de commande est utilisé pour déplacer Robotino.

On veut que Northstar détecte le projecteur du local 4. Le nouveau projecteur NorthStar doit pour ce faire être réglé dans le local 1.

La constante booléenne (vrai/faix) permet de transformer les systèmes de coordonnées de Northstar et de l'odométrie, de sorte que la pose Northstar courante soit égale à la pose odométrique courante, à savoir (100,10,90).

La soustraction des composants Northstar et Odométrie permet de mettre en évidence l'erreur survenant entre Odométrie und Northstar.

6.3.3.6 Connecteur d'E/S

Ce dossier contient des blocs de fonction concernant d'autres interfaces matérielles de Robotino.

6.3.3.6.1 Entrée de codeur

Ce bloc permet d'accéder à l'entrée de codeur libre disponible sur Robotino. L'entrée de codeur exploite le signal d'un codeur angulaire numérique (canal A, B code de Gray). Elle prend en compte aussi bien les fronts montants que descendants de sorte que l'on profite d'une résolution quatre fois supérieure à la résolution nominale du codeur.

Exemple : Les codeurs des moteurs de Robotino possèdent une résolution de 500 par tour. On enregistre en fait 2000 impulsions de codeur par tour.

Entrées	Type	Unité	Standard	Description
Mise à 0 de la position	bool		0	Vrai (true) = mise à 0 de la position Faux (false) = pas de changement de position
Sorties				
Vitesse	int	Impulsions par seconde.		La vitesse du codeur.
Position	int	Impulsions		L'intégrale de toutes les impulsions mesurées depuis le démarrage de Robotino ou depuis "Mise à zéro de la position" = vrai

6.3.3.6.2 Sortie de puissance

Ce bloc permet d'accéder à la sortie de puissance de Robotino (anciennement moteur 4). La sortie de puissance pince ne peut être utilisée que si le sous-programme ne contient pas encore de pince 148.

Cette sortie est réalisée techniquement par un pont en H capable de délivrer jusqu'à 5 A. Le pont en H est piloté par un signal PWM à haute fréquence et un bit de direction. La valeur de consigne spécifiée à l'entrée positionne le bit de direction en fonction de son signe. La valeur de la consigne influence le signal PWM. Une consigne de 0 ne génère pas de signal PWM, c.-à-d. que le pont en H ne délivre pas de courant. Une valeur de consigne de 50 établit un rapport de 50% entre high et low du signal PWM. Une valeur de consigne de 100 se solde par un high constant, c.-à-d. que le pont en H délivre le courant maximal.

Entrées	Type	Unité	Standard	Description
Valeur de consigne	int		0	Positionne le bit de direction et le signal PWM. Plage de valeurs de -100 à 100. Les valeurs inférieures à -100 sont mises à -100. Les valeurs supérieures à 100 sont mises à 100.
Sorties				
Courant	float	A		Le courant passant par le pont H.

Le courant délivré à la sortie de puissance est limité par défaut. Pour adapter cette limitation ou la désactiver, il faut éditer sur Robotino le fichier /etc/robotino/robotino.xml. Les nouvelles valeurs sont lues et prises en compte par Robotino au bout de 2 s.

6.3.3.6.3 Pince

Utilisez ce bloc pour commander une pince Festo Robotino La pince ne peut être utilisée que si le sous-programme ne contient pas encore de sortie de puissance[147].

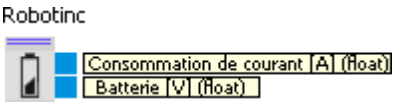
Entrées	Type	Standard	Description
Ouvrir	bool	false	Vrai (true) = Ouverture de la pince Faux (false) = Fermeture de la pince
Sorties			
Ouverte	bool		Vrai (true) = La pince est ouverte
Fermée	bool		Vrai (true) = La pince est fermée

La pince doit être connectée à la borne X15 de la carte située derrière l'accumulateur :
câble marron à gauche (+) / câble bleu à droite (-)

6.3.3.7 Capteurs internes

Enter topic text here.

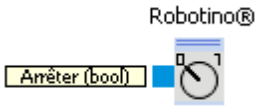
6.3.3.7.1 Surveillance de courant



Module de surveillance de l'alimentation de Robotino.

Entrées	Type	Unité	Standard	Description
Sorties				
Consommation de courant	float	A		Le courant momentanément prélevé sur la batterie.
Batterie	float	Volt		Tension des cellules des batteries 1 + 2

6.3.3.7.2 Arrêter



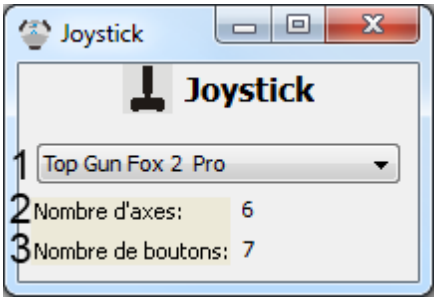
Arrêter Robotino.

Entrées	Type	Standard	Description
Arrêter	bool	false	Si vrai (true), Robotino s'arrête.

6.4 Manche à balai

Le dispositif "Joystick" permet d'accéder à une manette connectée localement.

6.4.1 Dialogue

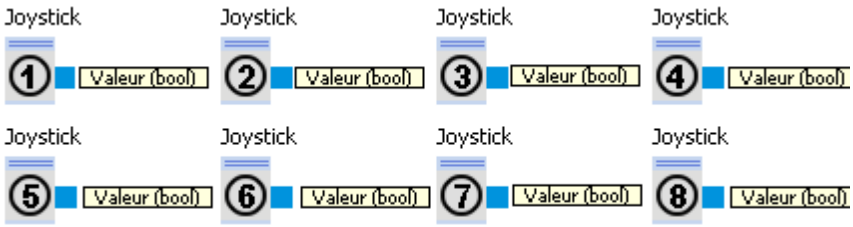


1	Liste des joysticks	Sont affichés ici tous les joysticks connectés au système. La liste est mise à jour dès qu'un joystick est connecté à l'ordinateur ou déconnecté. La sélection d'un joystick permet d'accéder à ses axes et boutons et aux blocs de fonction correspondants.
2	Nombre d'axes	Le nombre d'axes du joystick
3	Nombre de boutons	Le nombre de boutons du joystick

6.4.2 Blocs de fonction

Les blocs de fonction permettent d'utiliser le dispositif "Joystick" dans un sous-programme.

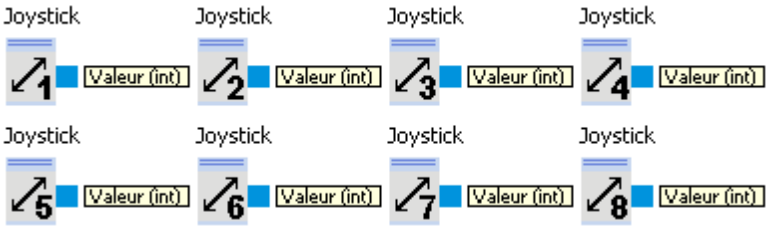
6.4.2.1 Bouton



Le bloc de fonction Bouton permet d'interroger l'état d'un bouton du joystick. Le numéro du bouton figure dans le symbole du bloc de fonction.

Sorties	Type	Description
Valeur	bool	Vrai (true), si le bouton est enfoncé. Sinon faux (false).

6.4.2.2 Axes



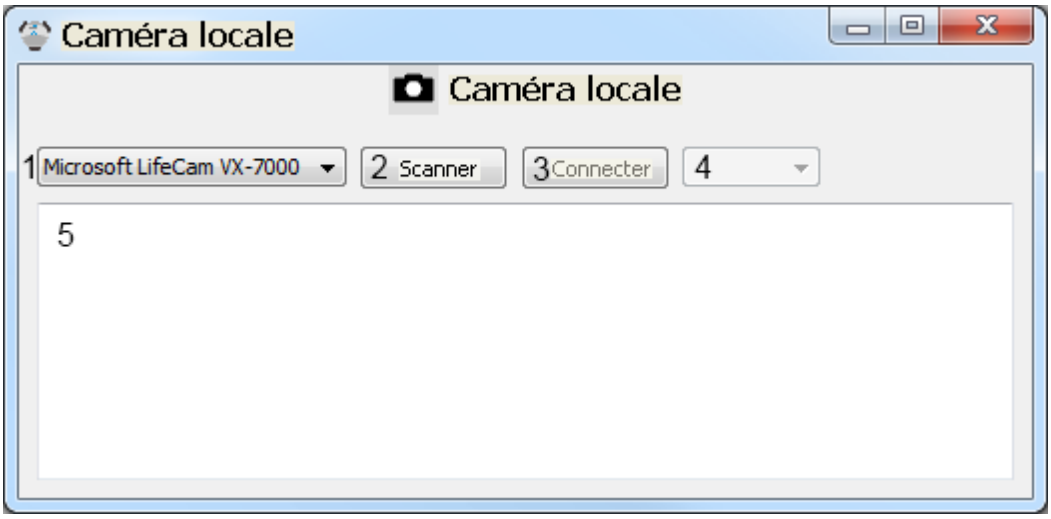
Le bloc de fonction Axes permet d'interroger l'état d'un axe du joystick. Le numéro de l'axe figure dans le symbole du bloc de fonction.

Sorties	Type	Description
Valeur	int	Plage de valeurs de -1000 à 1000.

6.5 Caméra locale

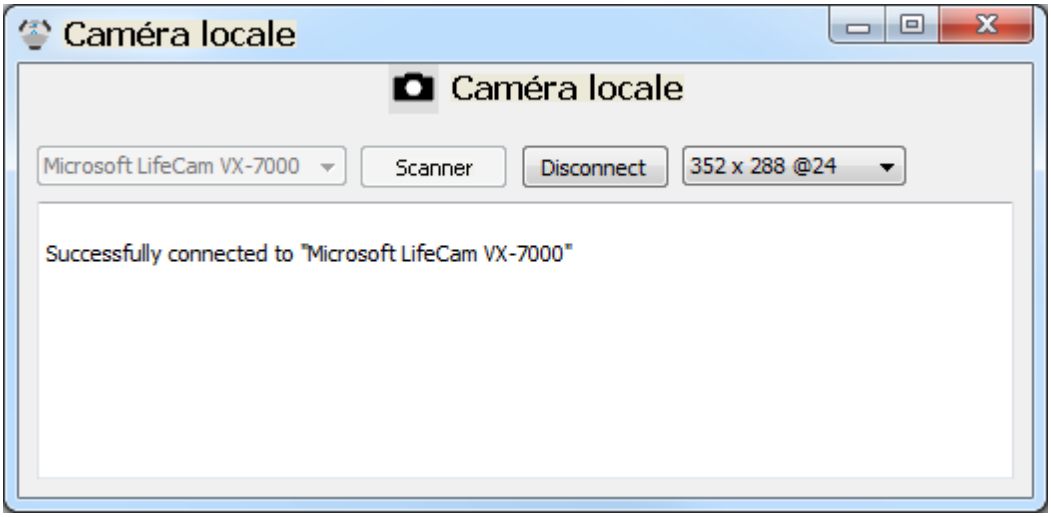
Le dispositif "Caméra locale" permet d'accéder à une caméra connectée localement à l'ordinateur (à une webcam p. ex.).

6.5.1 Dialogue



1	Liste des caméras	Sont affichés ici toutes les caméras connectées au système. La liste est mise à jour dès qu'une caméra est connectée à l'ordinateur ou déconnectée.
2	Scanner	Mise à jour de la liste des caméras
3	Connecter/déconnecter	Établissement/coupure d'une connexion à la caméra sélectionnée.
4	Résolution/profondeur de la couleur	Sélection de la résolution et de la profondeur de la couleur Toutes les résolutions prises en charge par la caméra sont disponibles.
5	Fenêtre de message	Affichage des messages d'état.

Après avoir sélectionné une caméra, il faut d'abord établir une connexion. Vous pouvez ensuite choisir la résolution voulue.



6.5.2 Blocs de fonction

Les blocs de fonction permettent d'utiliser le dispositif "Caméra locale" dans un sous-programme.

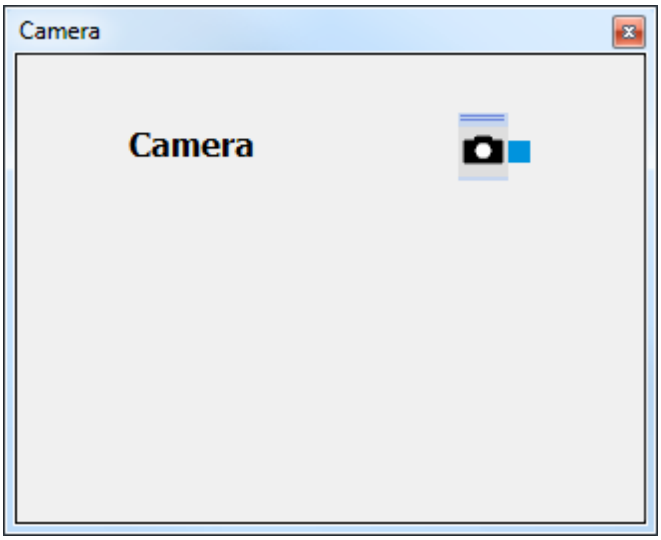
6.5.2.1 Caméra

Fournit l'image en direct de la caméra locale.

Entrées	Type	Standard	Description
Sorties			
Image	image		Image en direct de la caméra locale.

Les paramètres de profondeur de la couleur et de résolution sont définis dans la boîte de dialogue du dispositif 151.

6.5.2.1.1 Dialogue



La boîte de dialogue de la caméra affiche l'image actuelle de la caméra. Pour modifier les paramètres d'image voir boîte de dialogue du dispositif 151.

6.6 Client OPC

OPC est une interface standard entre divers programmes d'application et des pilotes de module (API p. ex.). Plusieurs clients OPC peuvent se connecter à un serveur OPC.
Les constructeurs d'API mettent souvent à disposition des serveurs OPC (spécifiques).

Dans l'exemple ci-dessous, un EasyPort Festo est connecté à RobotinoView par l'intermédiaire du serveur Festo gratuit EzOPC.

Le serveur EzOPC met les entrées/sorties à disposition, pour au maximum 4 EasyPorts, sous forme de "groupes" et de "variables" :

- L'appareil 1 forme le groupe "EasyPort1"
- La sortie 1 possède ainsi la variable "EasyPort1.OutputPort1"

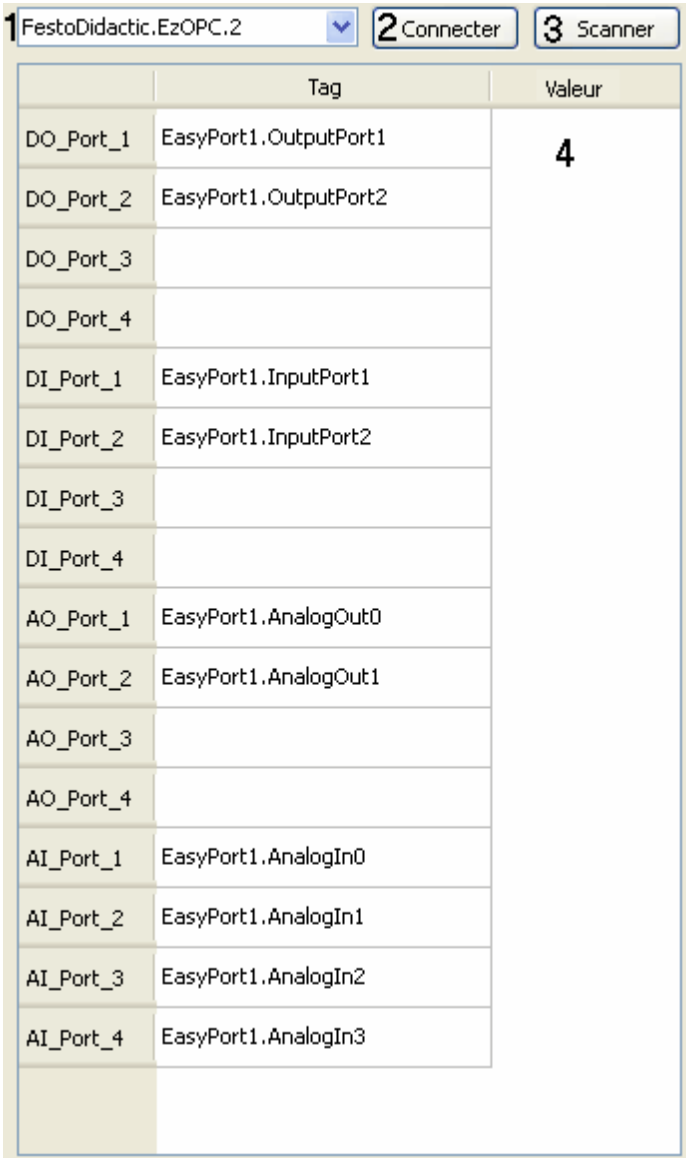
Marche à suivre :

1. Installez le serveur EzOPC
2. Démarrez le serveur EzOPC puis sélectionnez "Simulation de process ..." et "API via EasyPort".
3. Démarrez RobotinoView.
4. Ajoutez un dispositif "Client OPC".
5. Paramètres par défaut dans le menu contextuel de la boîte de dialogue du dispositif ▸ Sélectionnez "Festo EzOPC EasyPort". Les valeurs par défaut d'Easyport sont chargées.
6. Sélectionner "FestoDidactic.EzOPC.1", si nécessaire.
7. Établissez la connexion.
8. Utilisez les blocs OPC pour traiter les valeurs d'entrée/sortie du serveur OPC.

Astuce : Pour utiliser un API d'un autre constructeur, vous avez besoin d'un serveur OPC ou d'un client OPC de ce constructeur. Utilisez un client OPC pour savoir quels serveurs OPC et quelles variables sont disponibles sur votre PC.

Vous trouverez des fichiers à télécharger et des informations complémentaires p. ex. sous http://www.opcconnect.com/

6.6.1 Dialogue

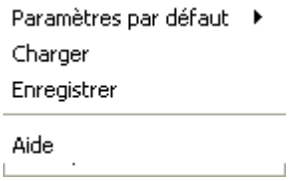


1	Sélection serveur OPC	La zone de liste déroulante affiche tous les serveurs OPC locaux actuellement activé.
2	Connecter	Appuyez sur ce bouton pour établir une connexion au serveur OPC sélectionné.
3	Recherche	Appuyez sur ce bouton pour mettre à jour la liste des serveurs OPC.
4	Liste d'affectations	Le tableau permet d'affecter les blocs de fonction aux "variables" OPC.

Nom de la ligne	**Bloc de fonction**

DO_Port_1	Sortie TOR 1
DO_Port_2	Sortie TOR 2
DO_Port_3	Sortie TOR 3
DO_Port_4	Sortie TOR 4
DI_Port_1	Entrée TOR 1
DI_Port_2	Entrée TOR 2
DI_Port_3	Entrée TOR 3
DI_Port_4	Entrée TOR 4
AO_Port_1	Sortie analogique 1
AO_Port_2	Sortie analogique 2
AO_Port_3	Sortie analogique 3
AO_Port_4	Sortie analogique 4
AI_Port_1	Entrée analogique 1
AI_Port_2	Entrée analogique 2
AI_Port_3	Entrée analogique 3
AI_Port_4	Entrée analogique 4

Le menu contextuel de la boîte de dialogue donne accès aux fonctions suivantes :



Paramètres par défaut ▶ Festo EzOPC VirtualPLC	Charge une affectation des blocs de fonction aux variables adaptée au VirtualPLC.

Paramètres par défaut ▶ Festo EzOPC EasyPort	Charge une affectation des blocs de fonction aux variables adaptée à l'EasyPort.
Charger	Charge une liste d'affectations à partir du disque dur.
Enregistrer	Enregistre la liste d'affectations actuelle sur le disque dur.
Aide	Affiche l'aide en ligne

6.6.2 Blocs de fonction

Les blocs de fonction permettent d'utiliser le dispositif "Client OPC" dans un sous-programme.

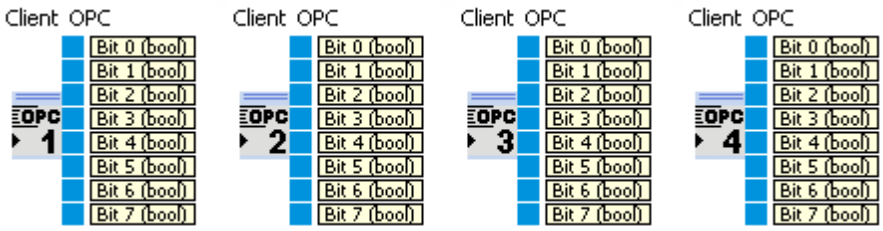
6.6.2.1 Entrées

6.6.2.1.1 Entrée analogique

Délivre la valeur de la "variable" affectée à la ligne AI_Port_x.

Sorties	Type	Description
Valeur	int	Plage de valeurs de 0 à 65535

6.6.2.1.2 Entrée TOR



Délivre les valeurs binaires de la "variable" affectée à la ligne DI_Port_x.

Sorties	Type	Description
Bit 0	bool	Vrai (true) si le bit 0 de la valeur délivrée par le serveur OPC est à 1.

...		
Bit 7	bool	Vrai (true) si le bit 7 de la valeur délivrée par le serveur OPC est à 1.

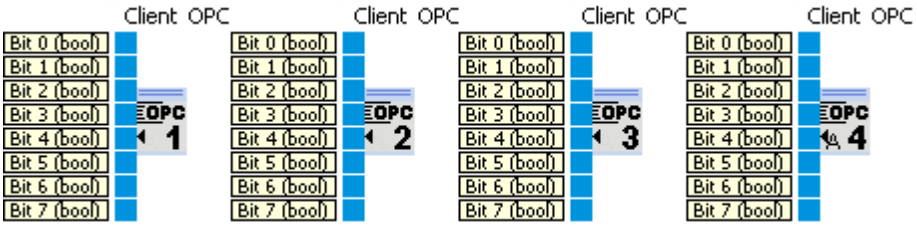
6.6.2.2 Sorties

6.6.2.2.1 Sortie analogique

Écrit la valeur d'entrée dans la "variable" affectée à la ligne AO_Port_x.

Entrées	Type	Description
Valeur	int	Plage de valeurs de 0 à 65535

6.6.2.2.2 Sortie TOR



Assemble les valeurs binaires en une valeur et l'écrit dans la "variable" affectée à la ligne DO_Port_x.

Entrées	Type	Description
Bit 0	bool	Si vrai (true), la valeur délivrée au serveur OPC est incrémenté de 2^0 = 1.
...		
Bit 7	bool	Si vrai (true), la valeur délivrée au serveur OPC est incrémenté de 2^7 = 128.

6.7 Échange de données

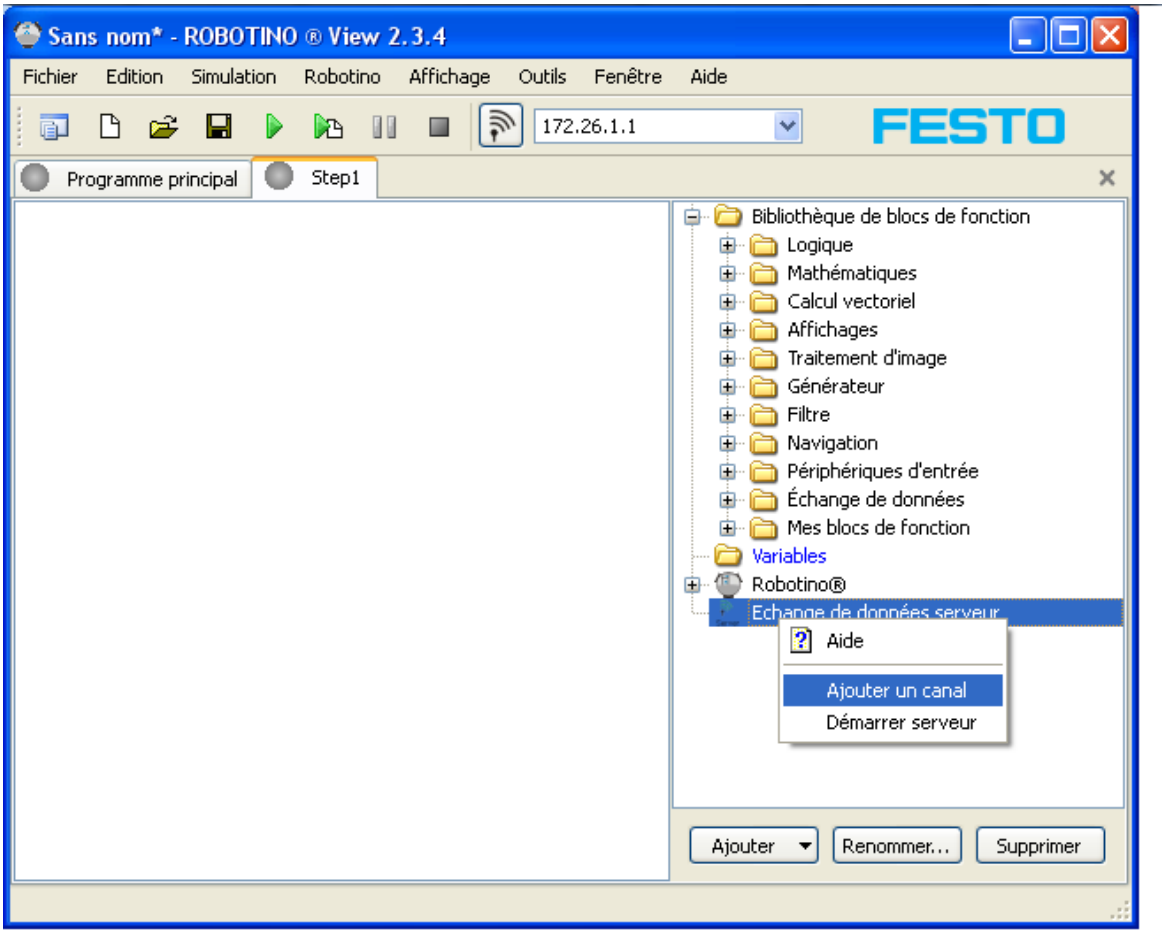
Les dispositifs de ce groupe permettent d'échanger via le réseau des données entre diverses instances de Robotino View.

6.7.1 Serveur

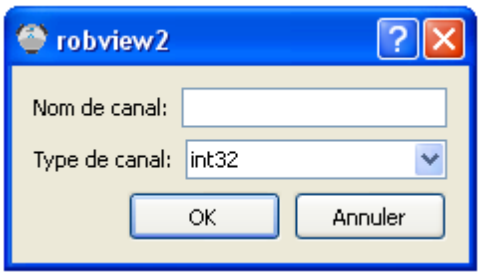
Le serveur d'échange de données peut être utilisé par un nombre illimité de clients pour échanger des données par un nombre illimité de canaux de communication. Les canaux de communication disponibles sont créés sur le serveur. Les canaux de communication disponibles sont communiqués aux clients. Les clients peuvent alors choisir les canaux via lesquels ils souhaitent communiquer avec le serveur.

Le serveur et les clients opèrent sur un pied d'égalité lors des échanges de données. Lorsqu'un client envoie des données sur un canal de communication, les données sont d'abord transmises au serveur puis de là, à tous les autres clients. Si plusieurs nœuds transmettent simultanément des données via un même canal, il n'est pas possible de prédire quelle donnée parviendra finalement à tous les nœuds.

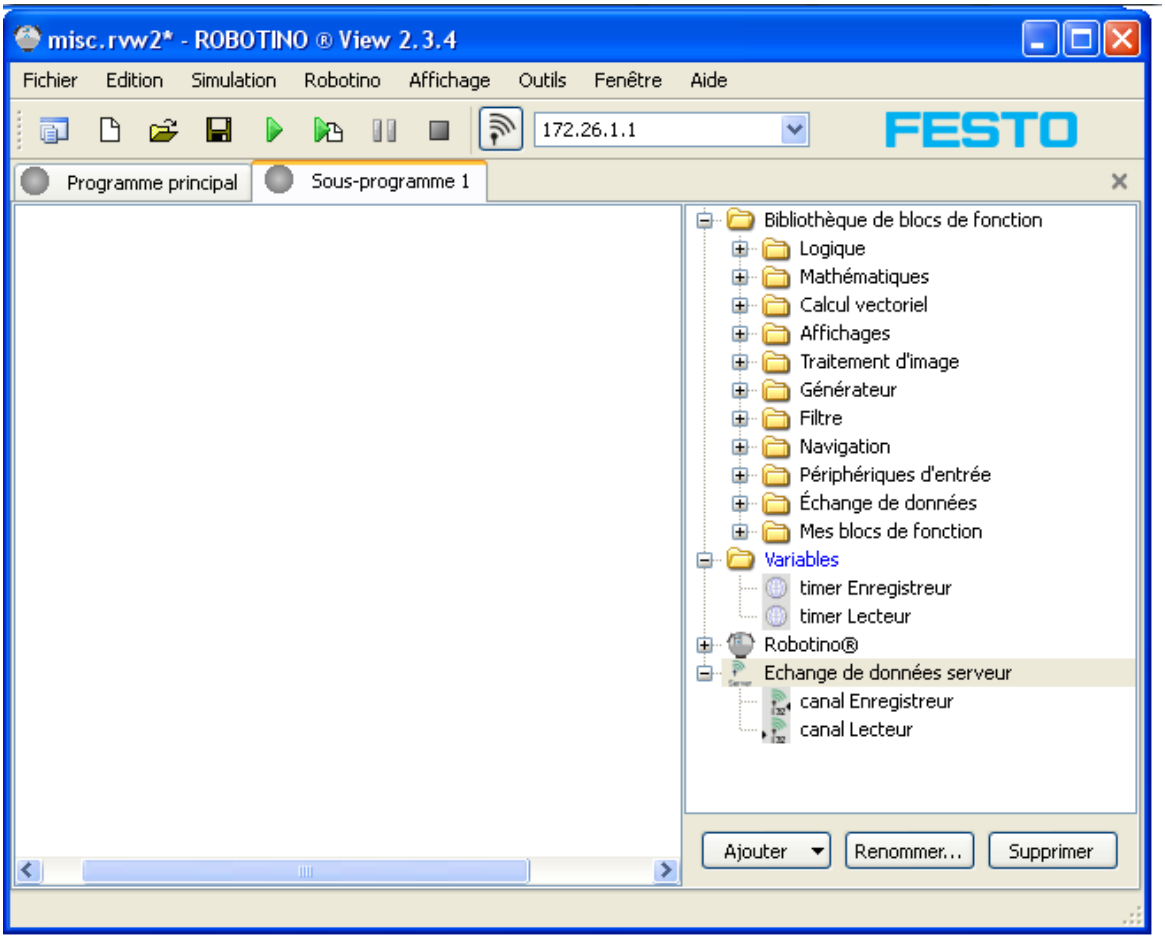
Une fois que le dispositif serveur d'échange de données a été ajouté à la bibliothèque de blocs de fonction, il est possible d'ajouter des canaux de communication à l'aide du menu contextuel du serveur d'échange de données.



La boîte de dialogue qui s'ouvre permet de spécifier le nom et le type du nouveau canal.



Le nom du canal doit être unique et ne comporter que des caractères ASCII sans "/". Le canal est créé après validation par OK. Dans la bibliothèque de blocs de fonction figurent alors les blocs de fonction "Nom de canal enregistreur" et "Nom de canal lecteur" pour l'écriture et la lecture via ce canal.



6.7.1.1 Dialogue

Le dialogue du serveur d'échange de données s'ouvre par un double clic sur la mention du dispositif dans la bibliothèque de blocs de fonction.

"Port du serveur" permet de définir le numéro de port TCP sur lequel le serveur reçoit les requêtes de connexion des clients.

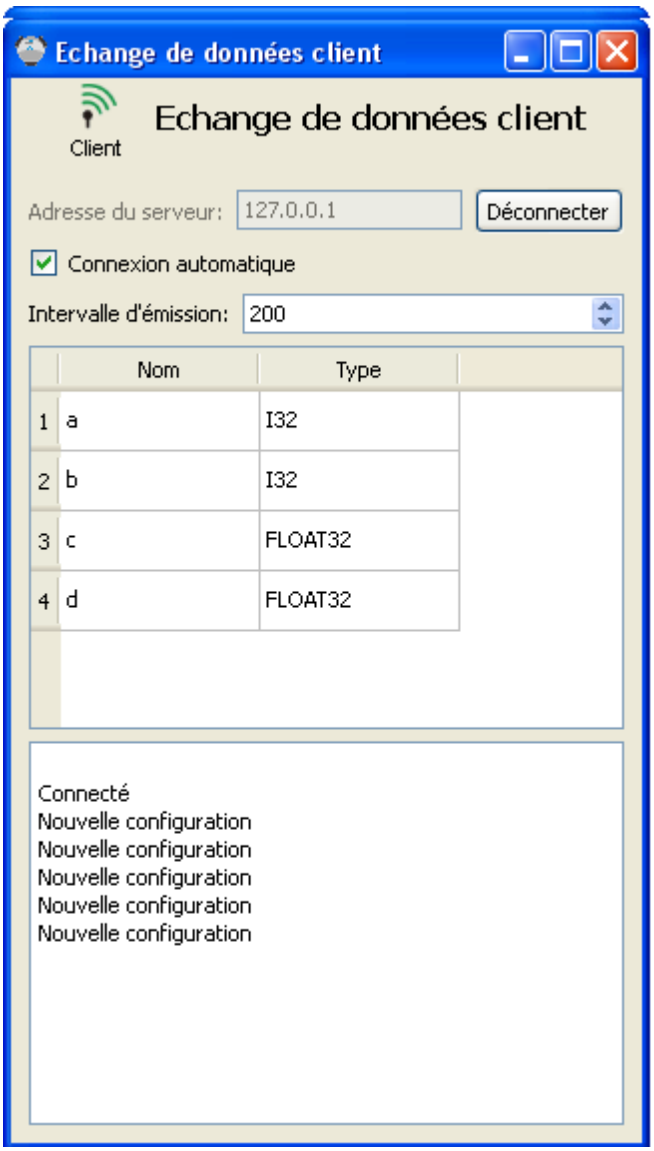
"Intervalle d'émission" définit le temps au bout duquel il est à nouveau possible de transmettre des données.

Le bouton "Démarrer serveur" démarre le serveur. Les clients peuvent alors établir une connexion au serveur.

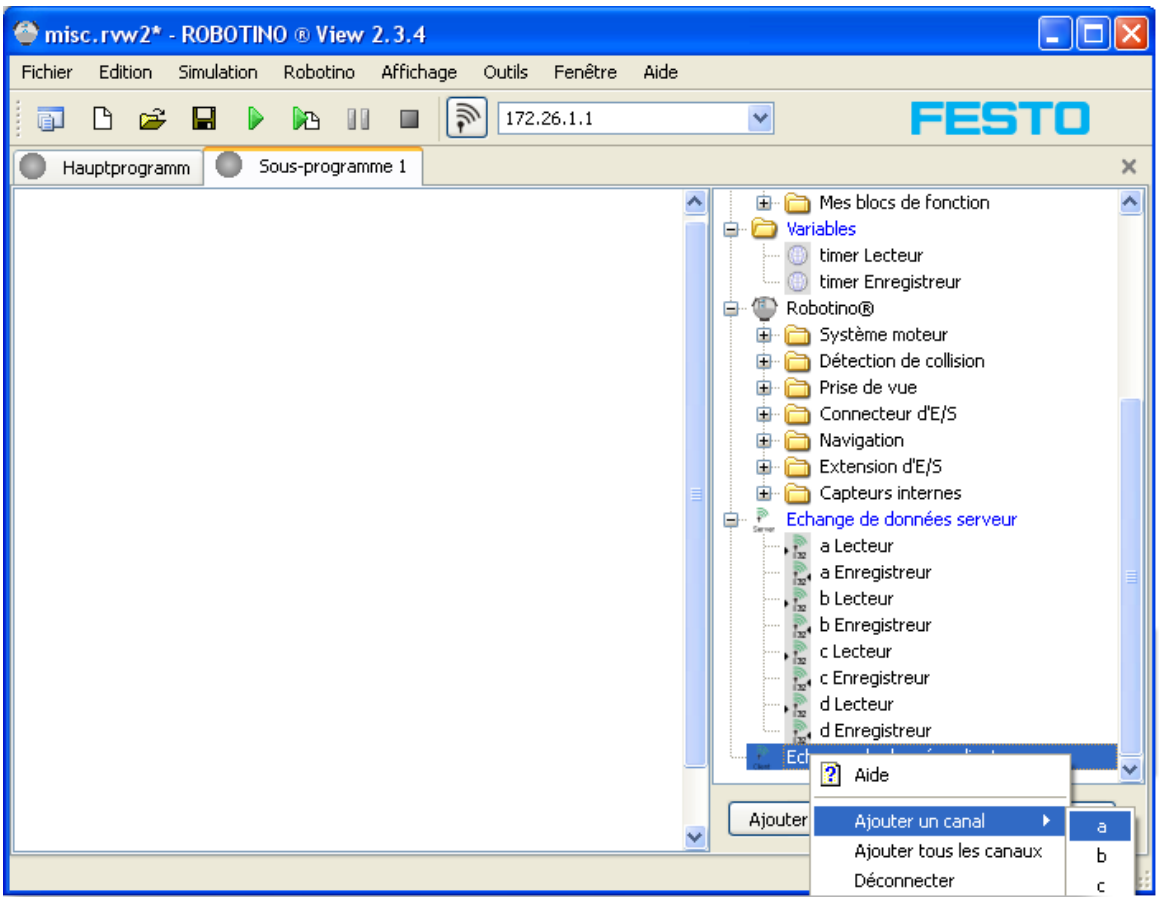
6.7.2 Client

Le client d'échange de données se connecte à un serveur d'échange de données 158 et permet d'échanger des données via des canaux de communication définis par le serveur.

La liste des canaux de communication est disponible dès que le client d'échange de données s'est connecté avec succès à un serveur d'échange de données 158.

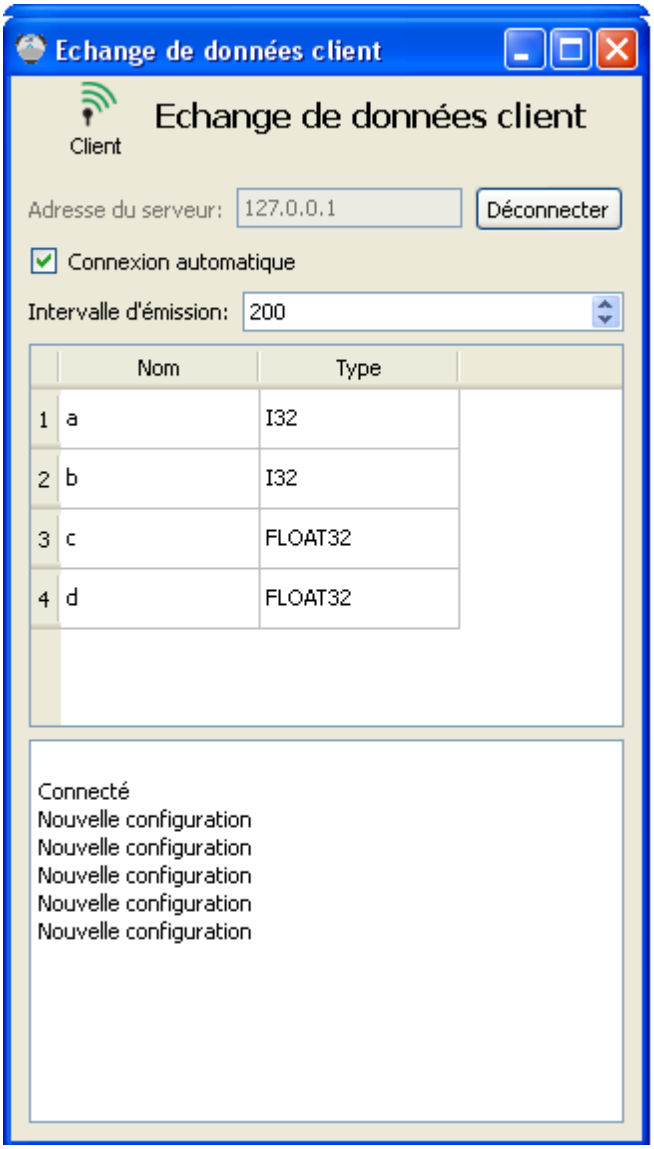


Les canaux de communication a,b de type I32 (entier de 32 bits) et c,d (nombre à virgule flottante de 32 bits) ont été créés sur le serveur. Ces canaux peuvent à présent être ajoutés au client dans la bibliothèque de blocs de fonction.



Comme pour le serveur d'échange de données[158] l'ajout d'un canal fait apparaître un bloc de fonction lecteur et un bloc de fonction enregistreur. Le menu contextuel permet d'ajouter les canaux individuellement ou tous en même temps. Le bouton "Connecter" établit la connexion au serveur sans qu'il soit nécessaire d'ouvrir le dialogue du client.

6.7.2.1 Dialogue



Adresse du serveur désigne l'adresse IP du serveur auquel le client doit se connecter. Si vous indiquez uniquement l'adresse IP, le port standard 9080 sera utilisé par défaut pour établir la connexion. Si le serveur écoute sur un autre port, il est possible de l'indiquer à la suite de l'adresse IP en le séparant de cette dernière par un ":".

Si le serveur est p. ex. actif sur l'ordinateur local sur le port 8000, l'adresse du serveur sera 127.0.0.1:8000.

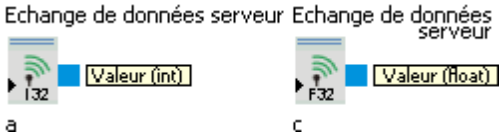
En cochant "Connexion automatique", vous spécifiez le rétablissement automatique par le client d'une connexion coupée.

"Intervalle d'émission" définit le temps au bout duquel il est à nouveau possible de transmettre des données.

6.7.3 Blocs de fonction

Les blocs de fonction permettent d'échanger des données avec les dispositifs.

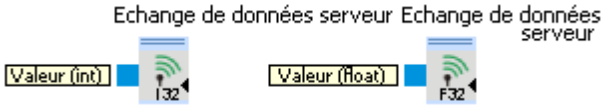
6.7.3.1 Lecteur



Le lecteur lit les données d'un canal de communication.

Sorties	Type	Description
Valeur	int, float, float array, laser range data	La valeur du canal de communication

6.7.3.2 Enregistreur



L'enregistreur écrit des données dans un canal de communication.

Entrées	Type	Standard	Description
Valeur	int, float, float array, laser range data	0	La valeur est envoyée au serveur et redistribuée par ce dernier à tous les clients.

6.8 Échange de données UDP

Le dispositif d'échange de données UDP permet d'échanger des données via UDP entre Robotino View et des applications externes.

6.8.1 Protocole

Spécification de la structure des données

Octet	Fonction
0	ID de message
1-2	Nombre d'octets du message N de type UINT16 166
3	Somme de contrôle (à initialiser par 0 lors de la constitution des paquets, voir Somme de contrôle) 165
N-1	Dernier octet du message

6.8.1.1 Somme de contrôle

Si le message compte moins de 100 octets, la somme s0 est formée par l'addition de tous les octets du paquet. Si le message compte 100 octets ou plus, la somme s0 est formée par l'addition des 50 premiers et 50 derniers octets.

Dans les deux cas, l'octet de la somme de contrôle doit être initialisé par 0. La somme de contrôle se calcule par la formule

Somme de contrôle = 0xff - s0

```
unsigned char checksum( const unsigned char* données_utiles, unsigned int Longueur_données_uti
{
	unsigned char s0 = 0;

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

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

Pour vérifier que le paquet a été correctement transmis, on additionne, pour les messages comptant moins de 100 octets, tous les octets du paquet pour former la somme s1. Si le message compte 100 octets ou plus, la somme s1 est formée par l'addition des 50 premiers et 50 derniers octets.

Le paquet est correct si

s1 = 0xFF

6.8.1.2 Types de données

Type	Largeur en octets	Description

UINT16	2	Byte0 : low
		Byte1 : high
		Dans un système petit-boutiste, il est possible de copier une valeur de données UINT16 directement dans les données utiles
		Exemple :
		//encoding
		`uint16 value = 9873;`
		`char Données_utiles[2];`
		`uint16* p = reinterpret_cast<uint16*>( Données_utiles );`
		`*p = value;`
		
		`//decoding`
		`value = *( reinterpret_cast<const uint16*>( Données_utiles ) );`

INT32	4	Byte0 : low
		Byte3 : high
		Dans un système petit-boutiste, il est possible de copier une valeur de données INT32 directement dans les données utiles
		Exemple :
		//encoding
		`int32 value = -3459873;`
		`char Données_utiles[4];`
		`int32* p = reinterpret_cast<int32*>( Données_utiles );`
		`*p = value;`
		
		`//decoding`
		`value = *( reinterpret_cast<const int32*>( Données_utiles ) );`

UINT32	4	Byte0 : low
		Byte3 : high
		Dans un système petit-boutiste, il est possible de copier une valeur de données UINT32 directement dans les données utiles
		Exemple :
		//encoding
		`uint32 value = 3459873;`
		`char Données_utiles[4];`
		`uint32* p = reinterpret_cast<uint32*>( Données_utiles );`
		`*p = value;`

```
//decoding
value = *( reinterpret_cast<const uint32*>( Données_utiles ) );
```

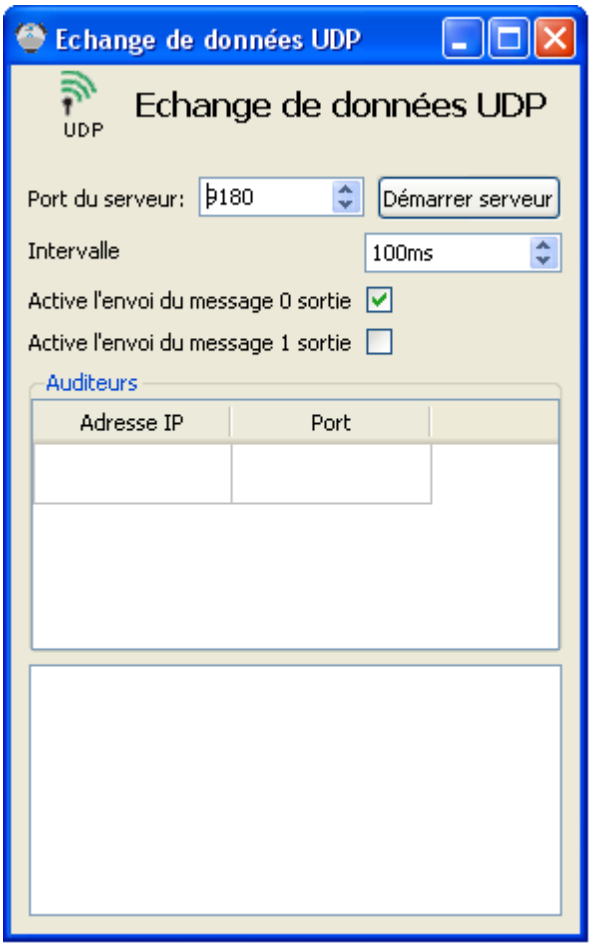
6.8.1.3 Message 0

Octet	Fonction
0	0
1	36
2	0
3	Somme de contrôle 165
4-7	INT0 de type INT32 166
8-11	INT1 de type INT32 166
12-15	INT2 de type INT32 166
16-19	INT3 de type INT32 166
20-23	INT4 de type INT32 166
24-27	INT5 de type INT32 166
28-31	INT6 de type INT32 166
32-35	INT7 de type INT32 166

6.8.1.4 Message 1

Octet	Fonction
0	1
1	36
2	0
3	Somme de contrôle 165
4-7	INT0 de type INT32 166
8-11	INT1 de type INT32 166
12-15	INT2 de type INT32 166
16-19	INT3 de type INT32 166
20-23	INT4 de type INT32 166
24-27	INT5 de type INT32 166
28-31	INT6 de type INT32 166
32-35	INT7 de type INT32 166

6.8.2 Dialogue



Le dialogue du dispositif d'échange de données UDP s'ouvre par un double clic sur la mention du dispositif dans la bibliothèque de blocs de fonction.

Il permet de configurer l'émission et la réception de paquets de données UDP :

"Port du serveur" permet de définir le numéro du port UDP qui reçoit et émet les datagrammes UDP.

Le bouton "Démarrer serveur" démarre le serveur. A compter de cet instant, les paquets de données UDP sont reçus, exploités et émis.

"Intervalle" définit le temps au bout duquel il est à nouveau possible de transmettre des données.

L'envoi du message 0 ou du message 1 peut être activé ou désactivé individuellement.

"Auditeurs" permet d'entrer l'adresse IP et le numéro de port des destinataires. Si aucun port n'est défini, le port 9180 est utilisé par défaut.

6.8.3 Blocs de fonction

Les blocs de fonction permettent d'échanger des données avec les dispositifs.

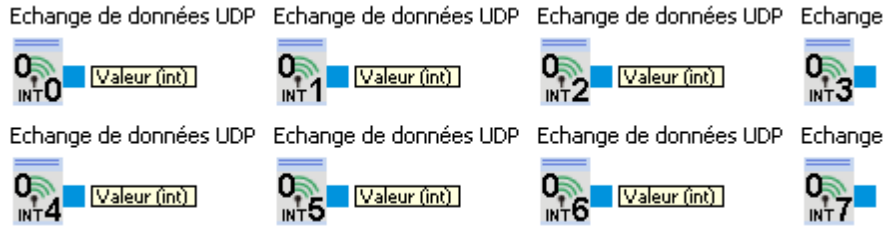
6.8.3.1 Message 0

Les blocs de fonction de la catégorie Message 0 permettent d'envoyer et de recevoir des données.

6.8.3.1.1 Entrée

Les entrées de Message 0 mettent à disposition des valeurs reçues.

6.8.3.1.1 Lecteur



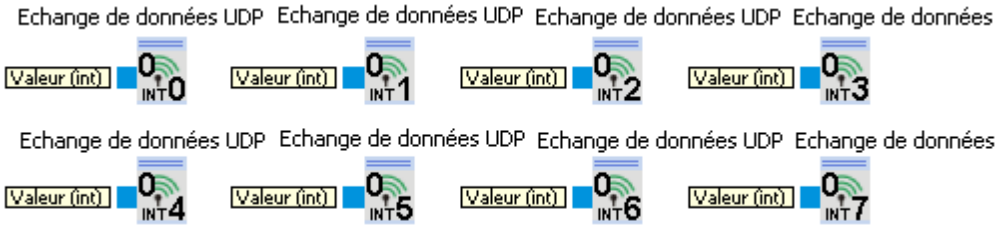
Le lecteur lit des données, met des données reçues à disposition. Il existe un lecteur pour respectivement INT0 à INT7.

Sorties	Type	Description
Valeur	int	La valeur reçue

6.8.3.1.2 Sortie

Les sorties permettent d'émettre des valeurs.

6.8.3.1.2 Enregistreur



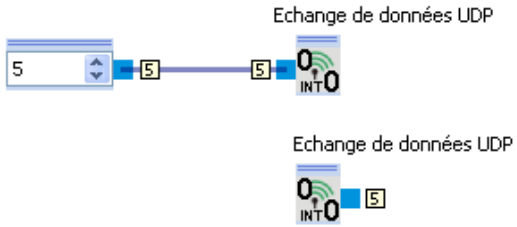
L'enregistreur reçoit les données à émettre et les remet au dispositif pour leur retransmission aux destinataires via UDP. Il existe un enregistreur pour respectivement INT0 à INT7.

Entrées	Type	Description
Valeur	int	La valeur à émettre

6.8.3.2 Message 1

Message 1 est identique à Message 0 169.

6.8.4 Exemple



7 Programmation

Pour compiler vos blocs de fonction ou dispositifs personnels, vous devez disposer de l'API Robotino® View 2.

7.1 Blocs de fonction personnels

Vous trouverez les exemples décrits ici sous

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

ou

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

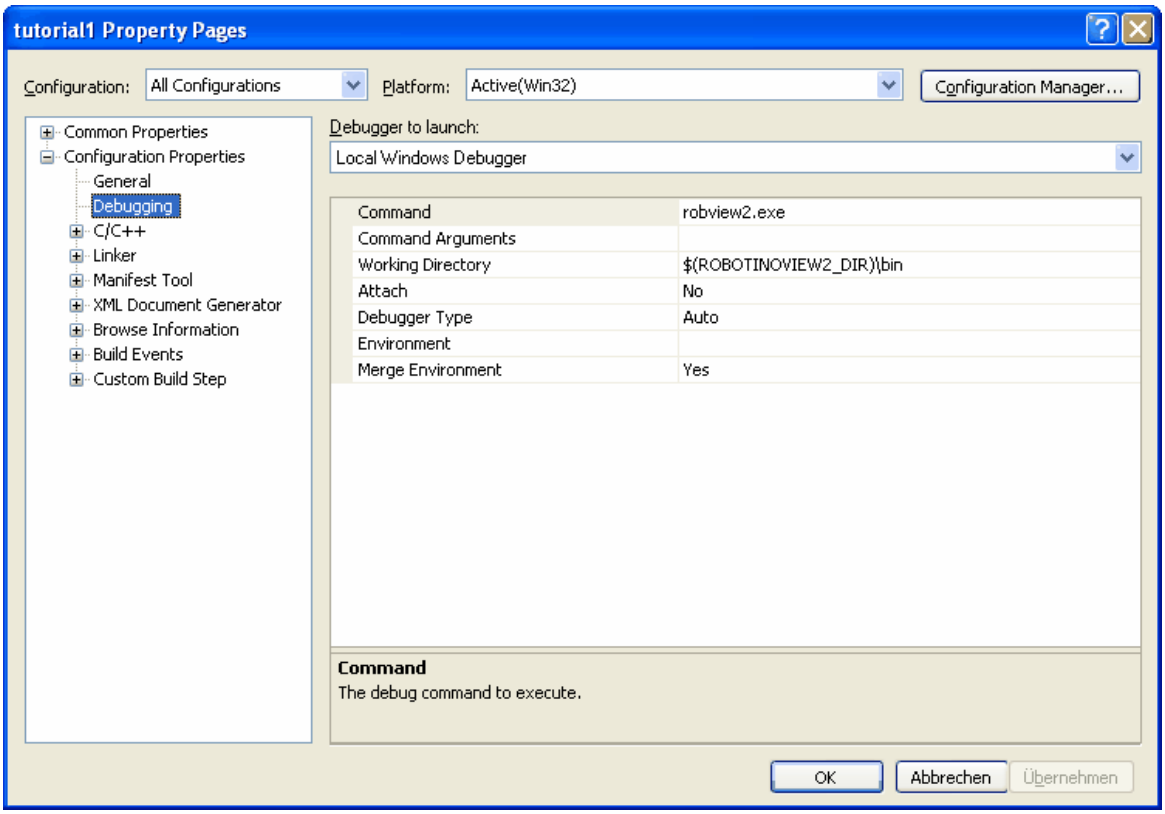
sur les système à 64 bits. La variable d'environnement %ProgramFiles% enregistre le chemin vers les programmes installés. Ce dernier est généralement "C:\Programmes" ou "C:\Program Files".

Avant d'ouvrir Visual Studio Solution tutorialx.sln, il faut exécuter le script

RUN THIS FIRST THEN START VS.cmd

dans le répertoire d'exercice correspondant. Ceci génère des paramètres personnalisés qui permettent entre autres de déboguer les blocs de fonction.

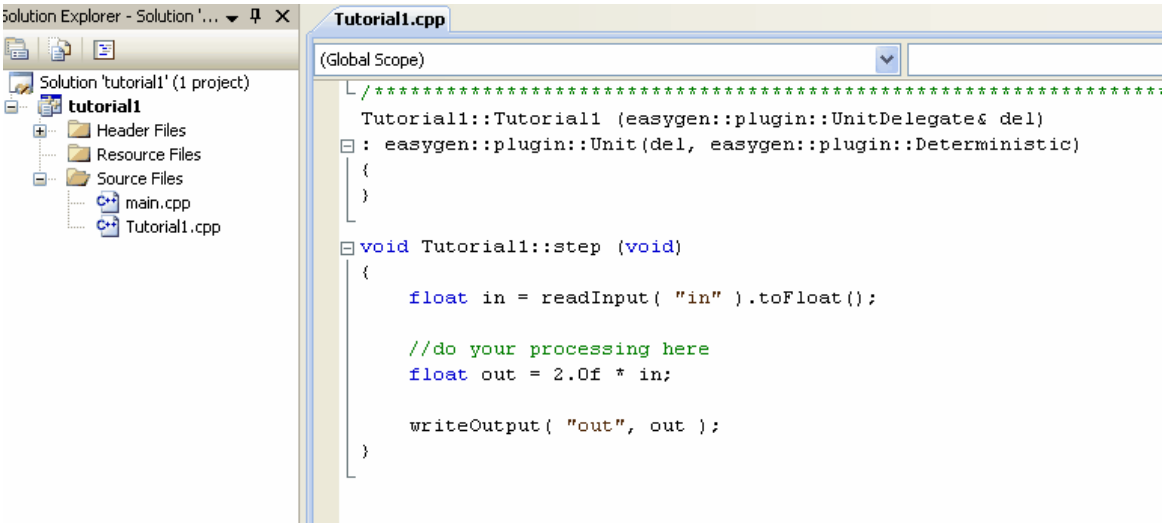
Pour que le débogage fonctionne, il faut spécifier, dans les propriétés du projet, Robotino View 2 comme fichier à exécuter ainsi que le répertoire de travail comme indiqué ci-dessous. Ces paramètres sont déjà correctement définis si vous avez déjà exécuté le script "RUN THIS FIRST THEN START VS.cmd" comme décrit ci-dessus.



7.1.1 Exercice 1

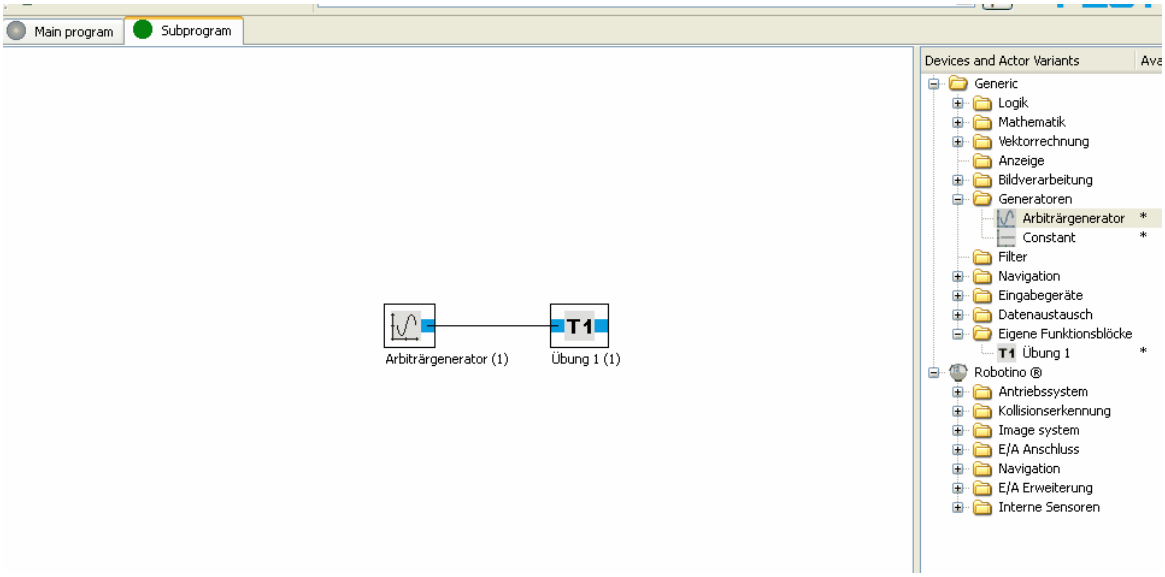
Répertoire : tutorial1.unit

Ce tutoriel montre comment créer un bloc de fonction personnel à une entrée et une sortie. Le code requis se trouve dans le fichier Tutorial1.cpp en méthod estep().

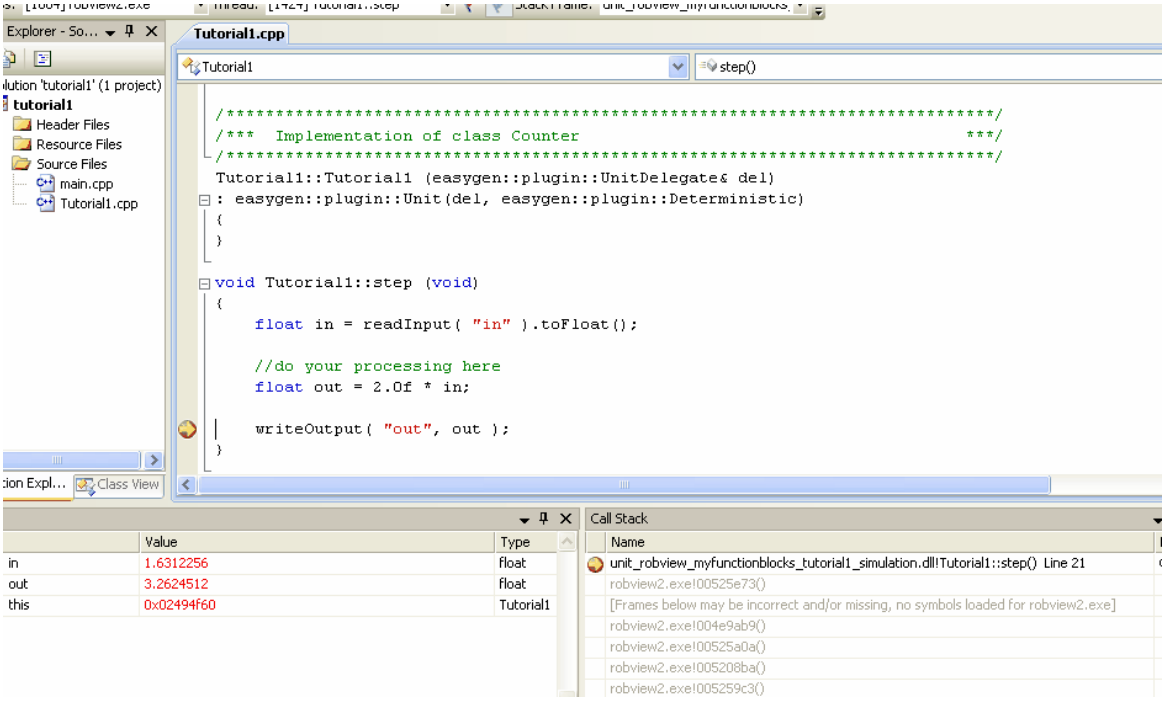


La valeur d'entrée "in" est multipliée par 2 puis émise en sortie. Toutes sortes de transformations peuvent être réalisées ici. Si vous voulez voir votre bloc de fonction en action, démarrez le débogage en appuyant sur la touche F5. Le bloc de fonction est compilé et Robotino View 2 démarré. L'écran affiche éventuellement une boîte de dialogue vous informant que robview2.exe ne contient pas d'informations de débogage. C'est bon. Valider en cliquant sur Yes.

Sous Robotino View 2, créez un sous-programme qui se présente p. ex. comme celui ci-dessous et démarrez-le.



Spécifiez dans VS un point d'interruption en méthode step().



L'exécution s'arrête au point d'interruption et vous pouvez faire afficher les valeurs. Supprimez le point d'interruption puis appuyez sur F5 pour poursuivre l'exécution du programme.

- O -

- P -

- R -

- S -

- T -

- U -

- V -

- X -