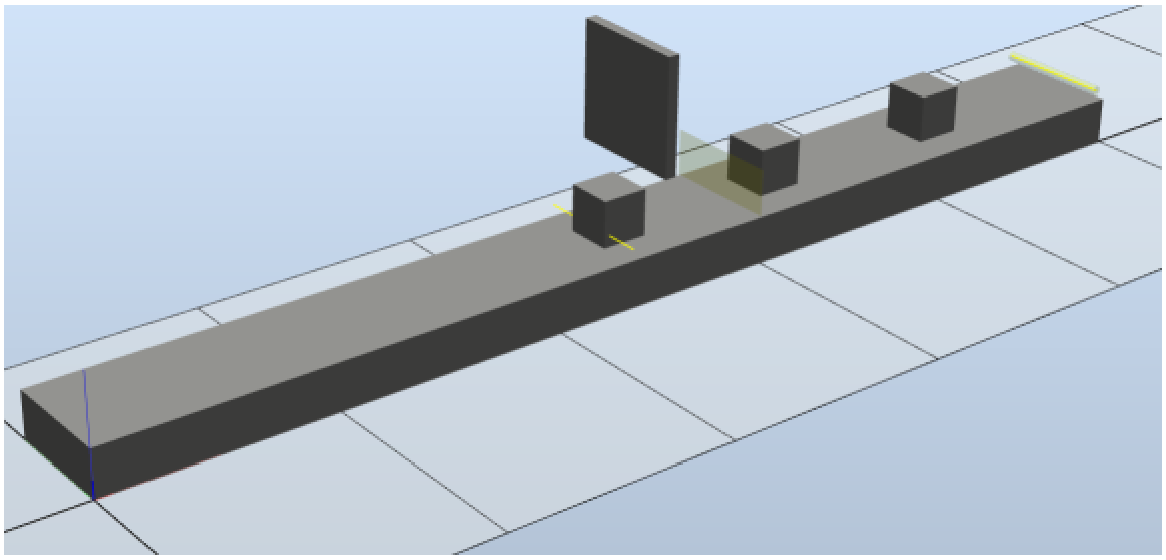
Guide for ABB RobotStudio

This guide explains the basic concepts of Mechanisms and Smart Components in RobotStudio



by

Michael Natapon Hansson

Department of Mechanical and Manufacturing Engineering, Aalborg University

January 2017

AALBORG UNIVERSITY
DENMARK

Content

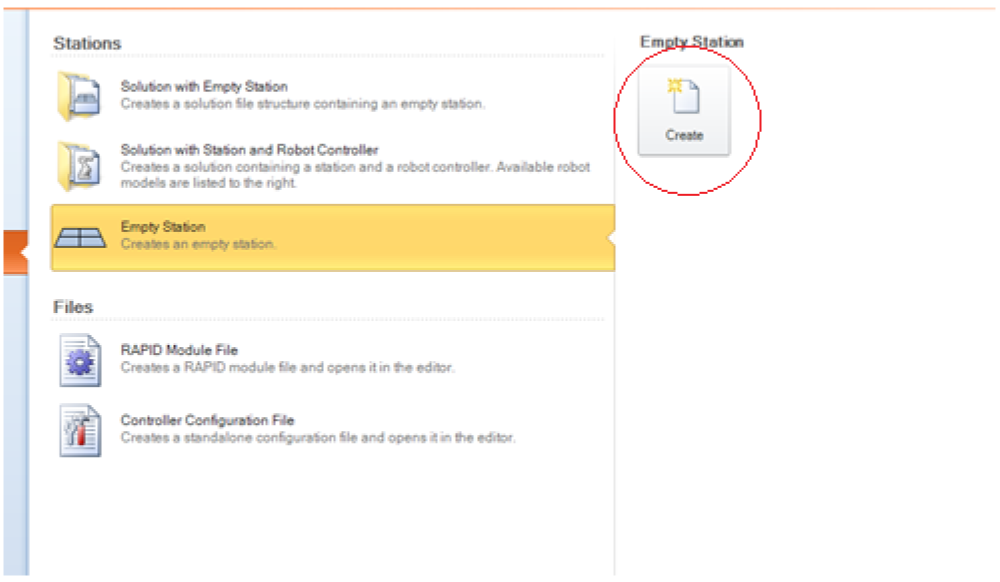
Creating a Device

Automating production always requires additional equipment besides a robot. This guide will show how functionalities such as “Mechanism” and “Smart Component” can be used to model such devices in RobotStudio. The specific outset for this guide is to create a conveyor with a gate, applying sensors, and setting up the logic for enabling the conveyor to transport parts.

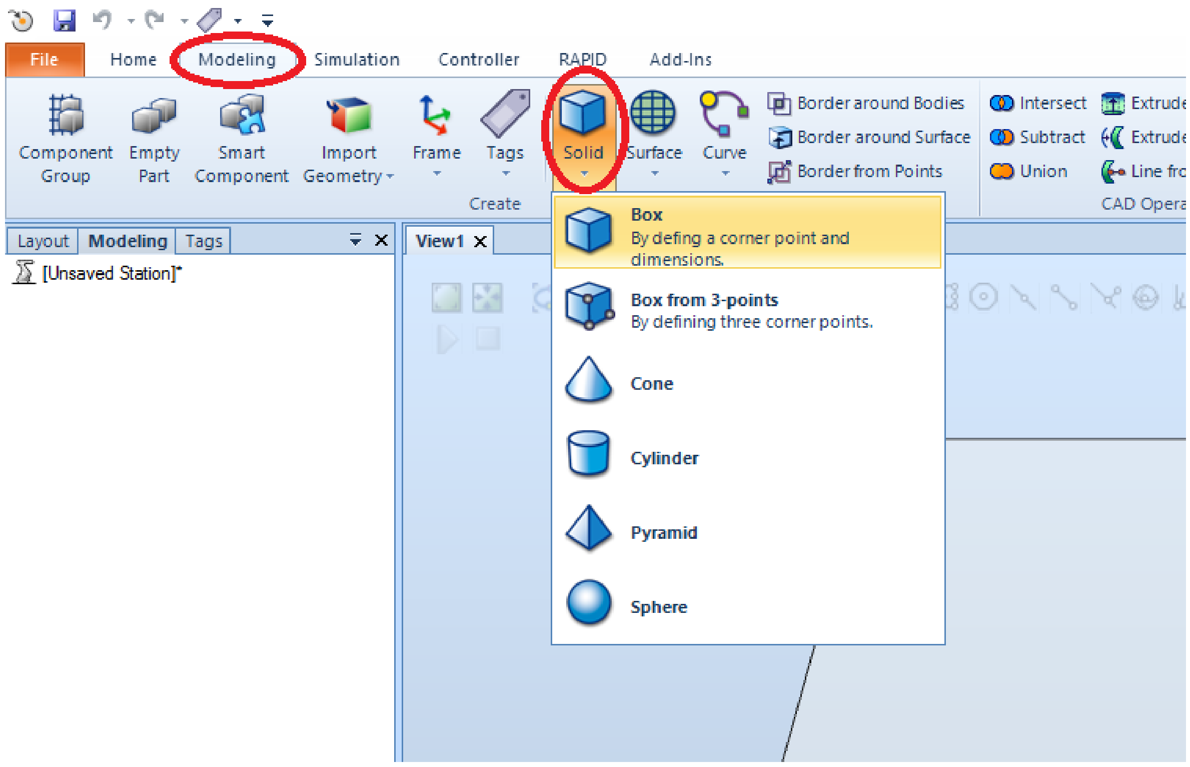
Modelling a conveyor and a gate

We will start by making the models for the conveyor, which in this example will be a very simple representation. You can choose to create any models in a CAD modelling software (e.g. SolidWorks) and import them into RobotStudio. IMPORTANT: The models should be saved in the *.SAT format, otherwise there may be problems importing the models in RobotStudio. For this example, we are going to create the models through RobotStudio.

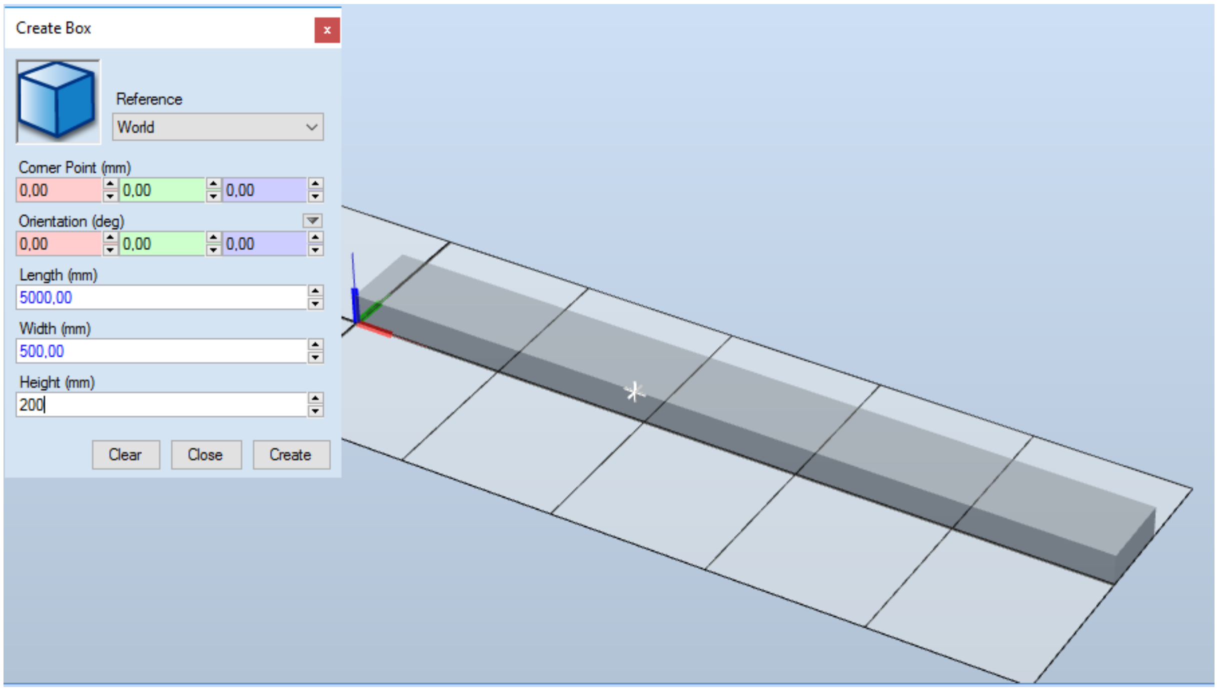
Start by creating a new station.


Stations
Solution with Empty Station
Creates a solution file structure containing an empty station.
Solution with Station and Robot Controller
Creates a solution containing a station and a robot controller. Available robot models are listed to the right.
Empty Station
Creates an empty station.
Files
RAPID Module File
Creates a RAPID module file and opens it in the editor.
Controller Configuration File
Creates a standalone configuration file and opens it in the editor.
Empty Station
Create

Next, go to “Modelling”, click on the dropdown of the “Solid” button and choose the “Box” option.



Specify the dimensions of the Box you wish to create. In this case we will set Length to 5000mm, Width to 500mm and Height to 200mm. When the dimensions have been specified, click “Create”.


Create Box

Reference: World

Corner Point (mm): 0,00 0,00 0,00

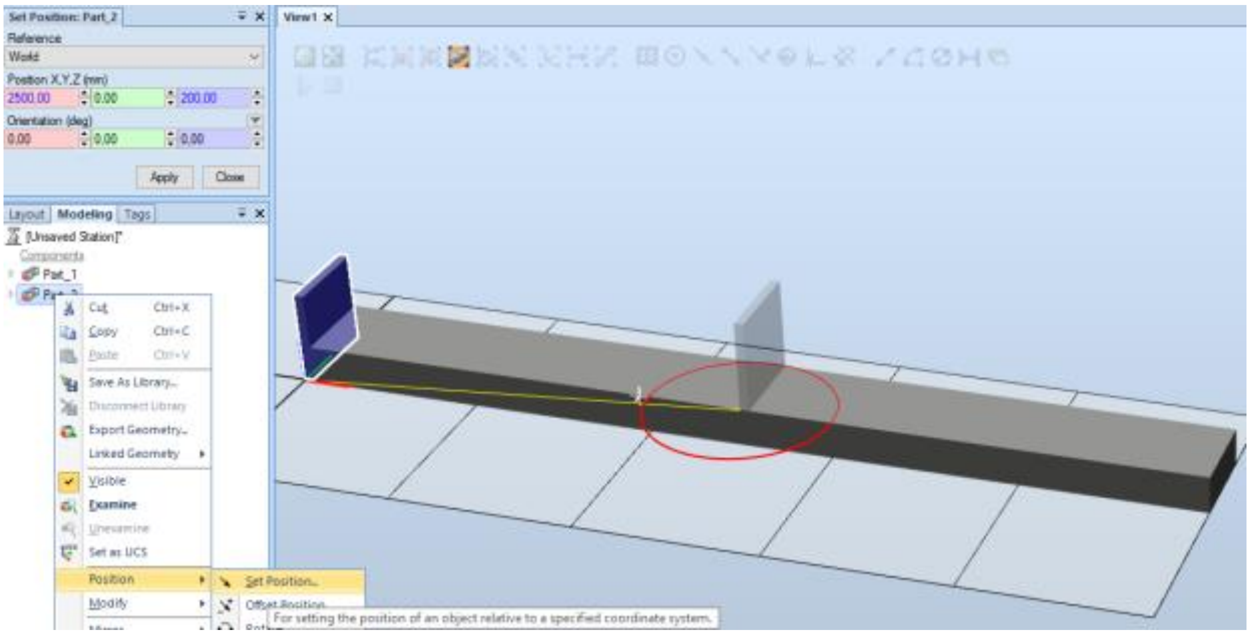
Orientation (deg): 0,00 0,00 0,00

Length (mm): 5000,00

Width (mm): 500,00

Height (mm): 200

Next, create another Box that will represent the gate with the dimensions L=50, W=500 and H=500, and position it at the middle of the conveyor. This can be done by right-clicking on the gate and choose “Set Position”.


Set Position: Part_2

Reference: World

Position X,Y,Z (mm): 2500.00 | 0.00 | 200.00

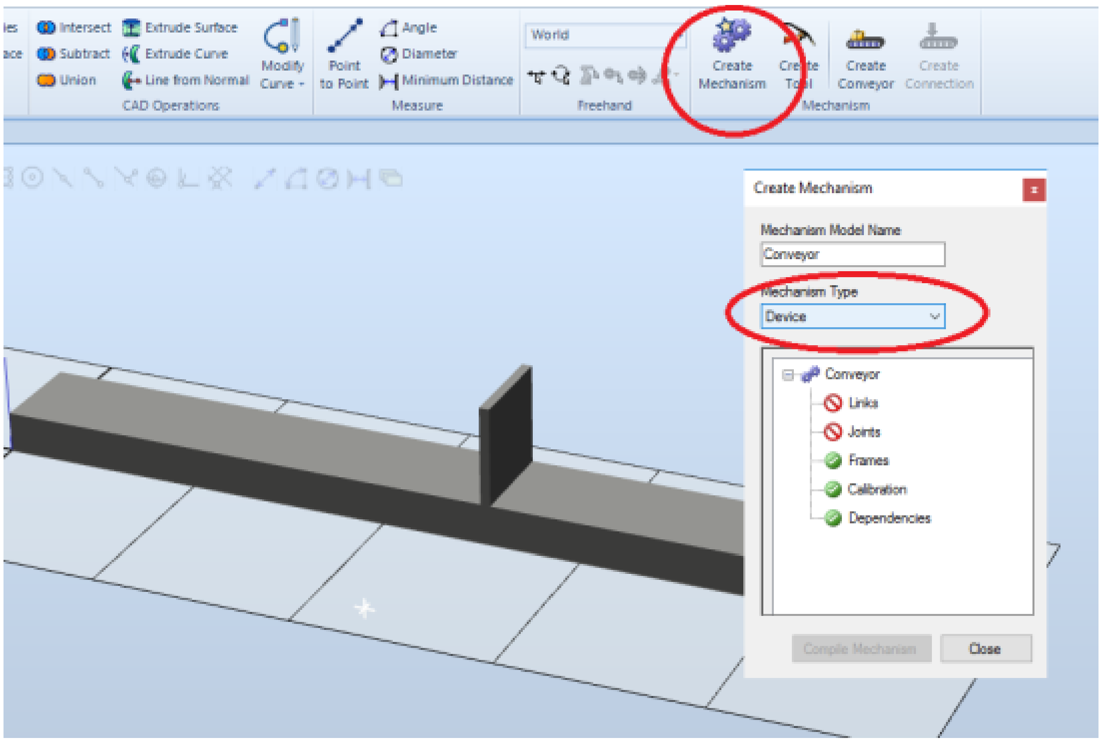
Orientation (deg): 0.00 | 0.00 | 0.00

For setting the position of an object relative to a specified coordinate system.

Creating a Mechanism

In RobotStuido, a “Mechanism” can be used to define how a collection of parts may be moved according to each other. In this example we want to specify the movements of the gate according to the conveyor.

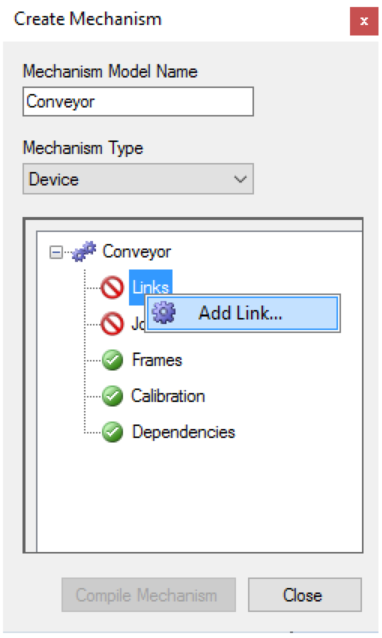
Select “Create Mechanism” under the “Modelling” tab. Name the mechanism, and specify that we want to create a “Device”.



Mechanism Model Name: Conveyor

Mechanism Type: Device

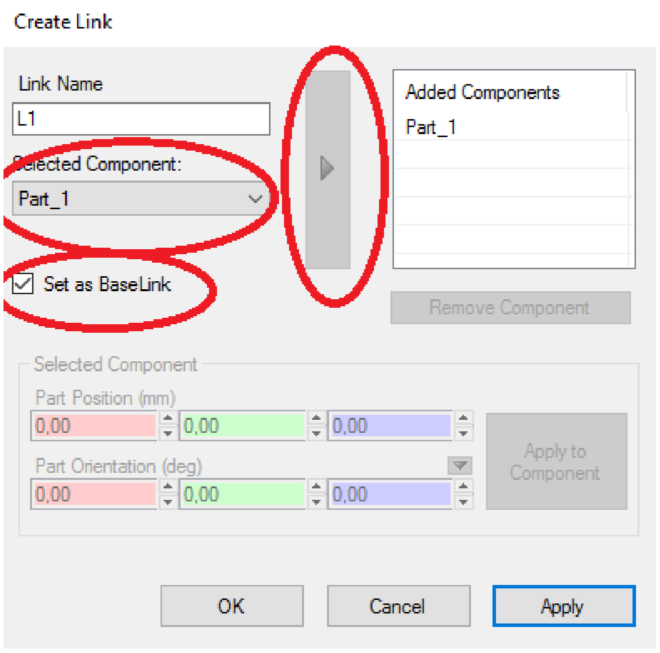
Next, we need to define the parts of interest. Right-click on “Links” and select “Add Links”



Mechanism Model Name: Conveyor

Mechanism Type: Device

Name the link if you wish, select the part that represents the conveyor, set is as Baselink, press the arrow, and press apply.


Create Link

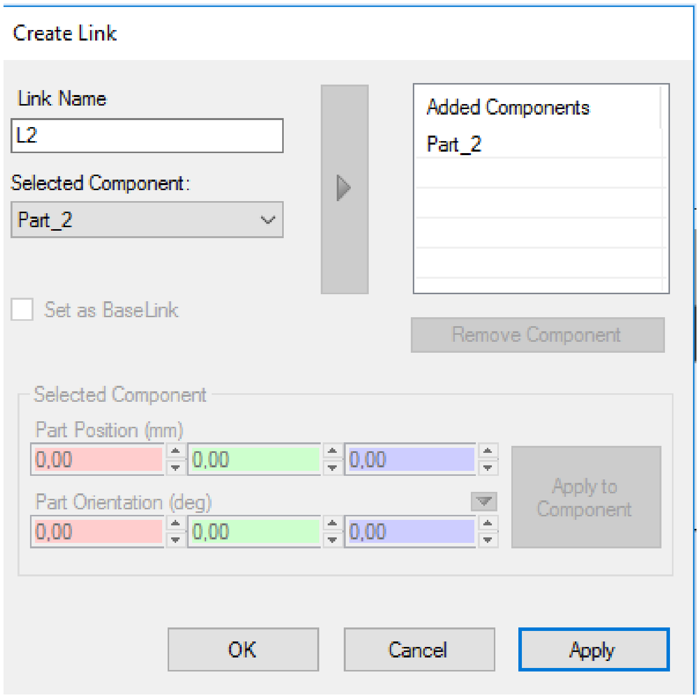
Link Name: L1

Selected Component: Part_1

☑ Set as BaseLink

Added Components: Part_1

Continue to do the same for the gate. You will see that you cannot set it as Baselink, but then again, you are not supposed to.


Create Link

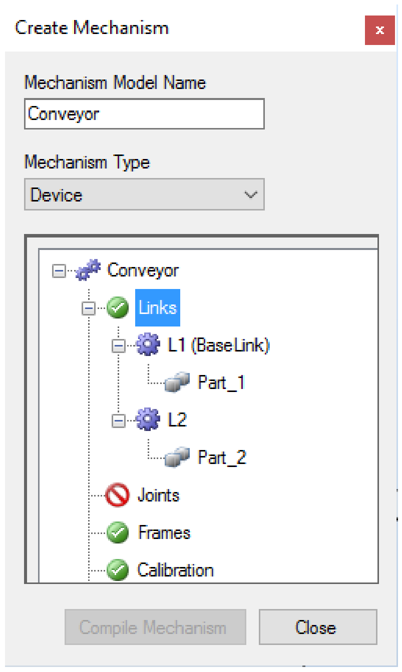
Link Name: L2

Selected Component: Part_2

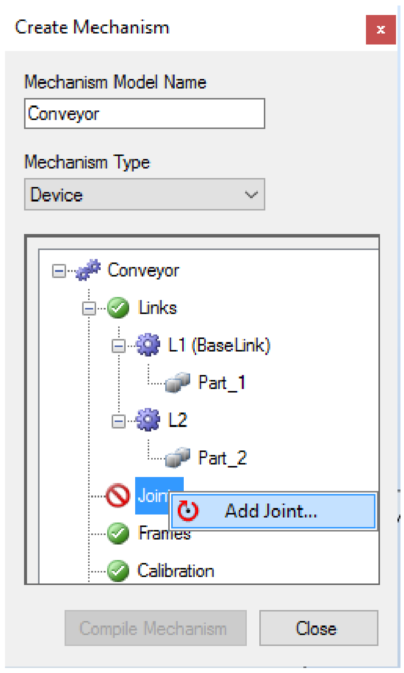
☐ Set as BaseLink

Added Components: Part_2

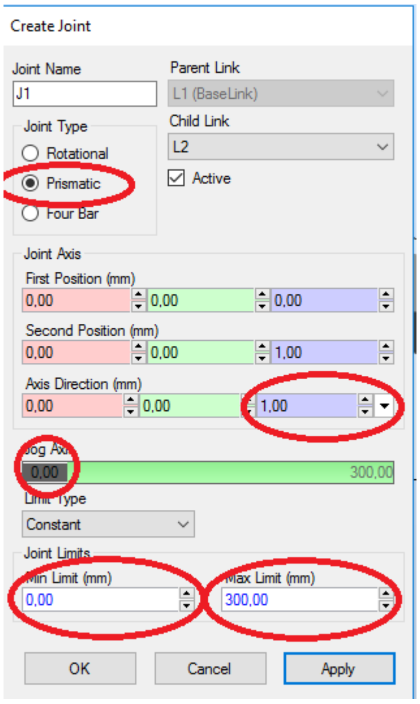
Your links should now be defined, and it should look like this.



Next, right-click “Joints” and select “Add Joint”



Name the joint if you wish, select prismatic, since we want the gate to move up and down. Type "1" in Z "Axis direction", set the limits to "0" and "300", and press "Apply".


Create Joint

Joint Name: J1

Parent Link: L1 (BaseLink)

Child Link: L2

[✓] Active

Joint Type: ○ Rotational [✓ ● Prismatic] ○ Four Bar

Joint Axis

First Position (mm): 0,00 0,00 0,00

Second Position (mm): 0,00 0,00 1,00

Axis Direction (mm): 0,00 0,00 1,00

Jog Axis: 0,00 300,00

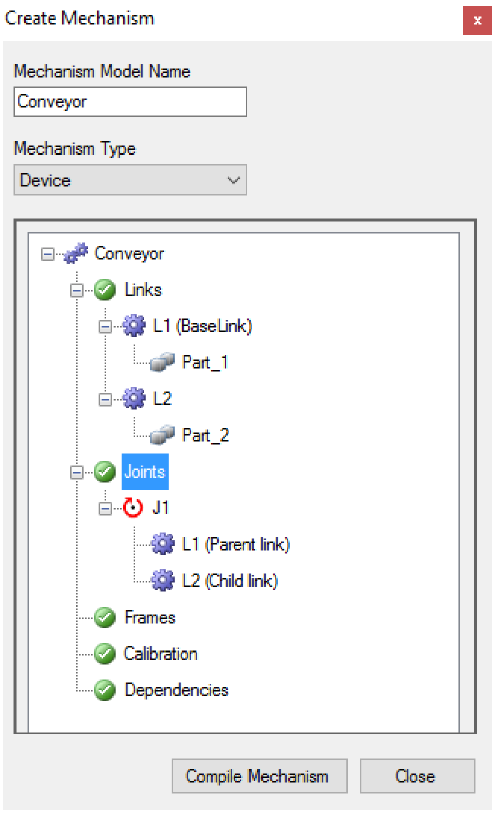
Limit Type: Constant

Joint Limits

Min Limit (mm): 0,00

Max Limit (mm): 300,00

The joints should now be defined, and it should look like this.

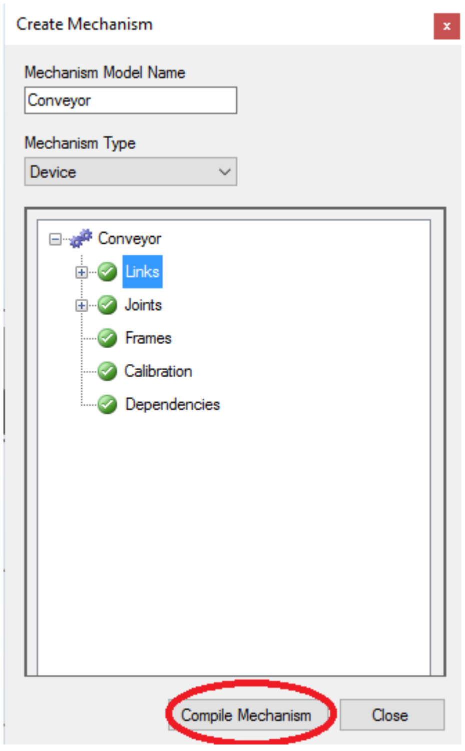

Create Mechanism

Mechanism Model Name: Conveyor

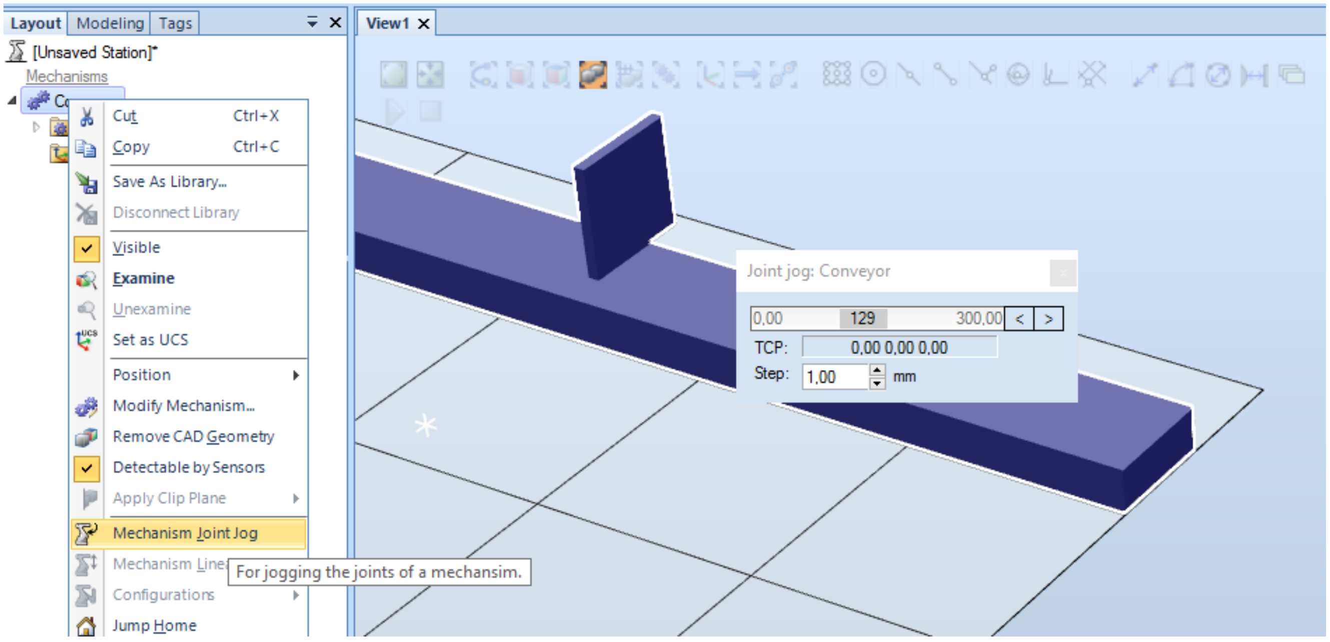
Mechanism Type: Device

```
Conveyor
  Links
    L1 (BaseLink)
      Part_1
    L2
      Part_2
  Joints
    J1
      L1 (Parent link)
      L2 (Child link)
  Frames
  Calibration
  Dependencies
```

Click “Compile Mechanism”, and afterwards click close.



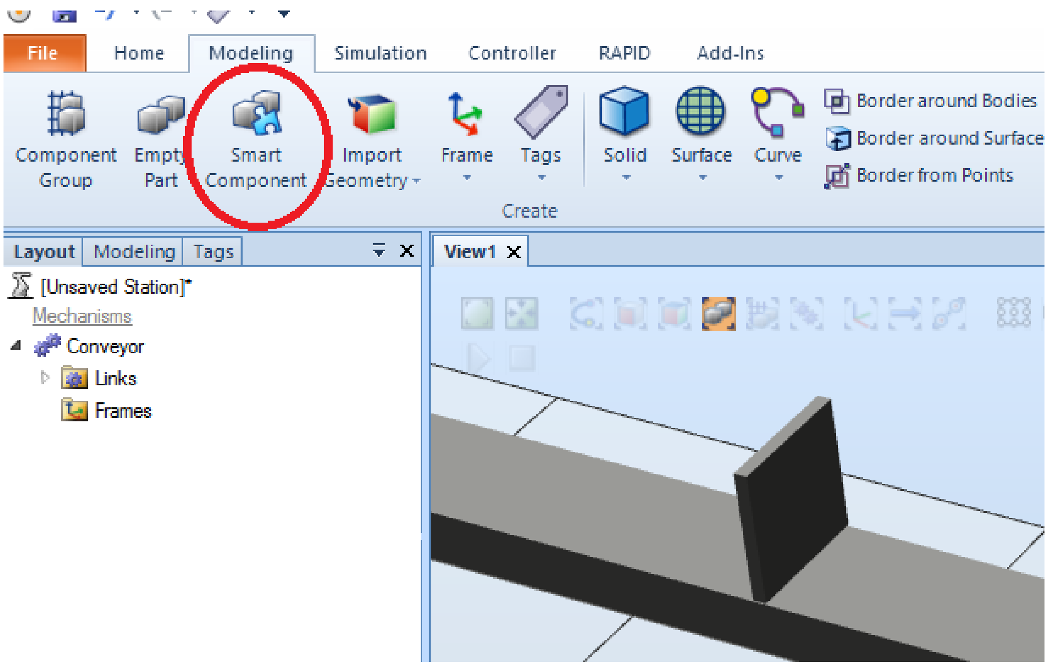
You have now created a conveyor-gate mechanism, feel free to play with it.



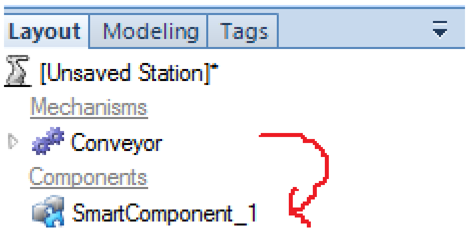
Creating a Smart Component

So now we have successfully created a conveyor-gate mechanism, but at the moment, there is nothing that defines the behaviour of the mechanism to be a conveyor. To accomplish this, we will need to add some logic. In RobotStudio, this is done by creating a Smart Component, where it will be possible to add sensors, logic and actions, to among others, mechanisms.

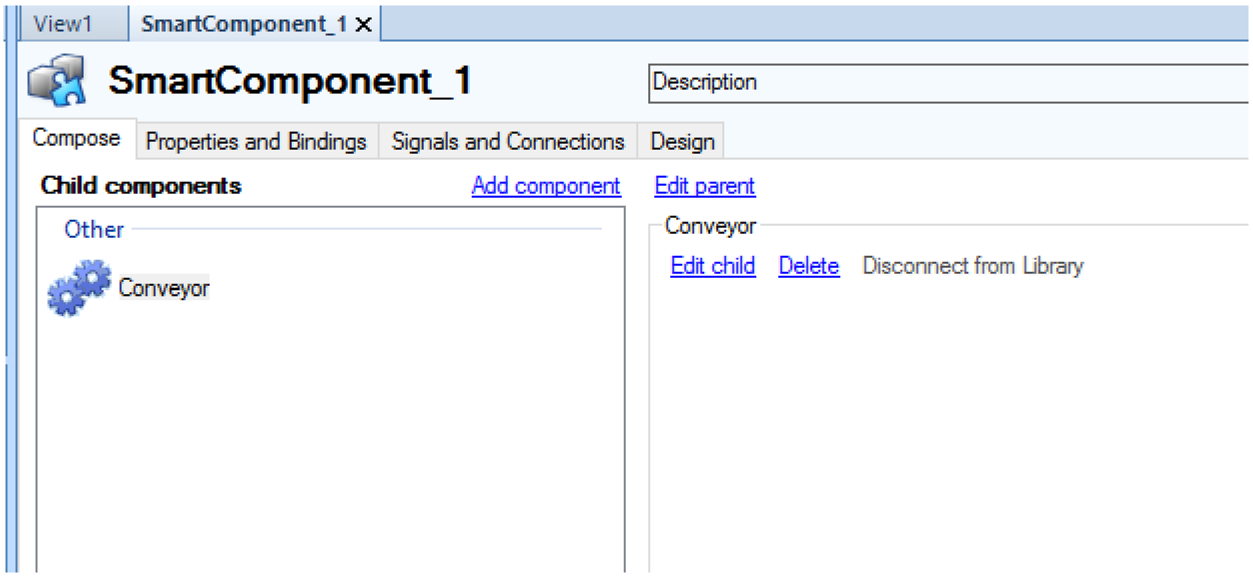
Go to “Modelling” and click “Smart Component”



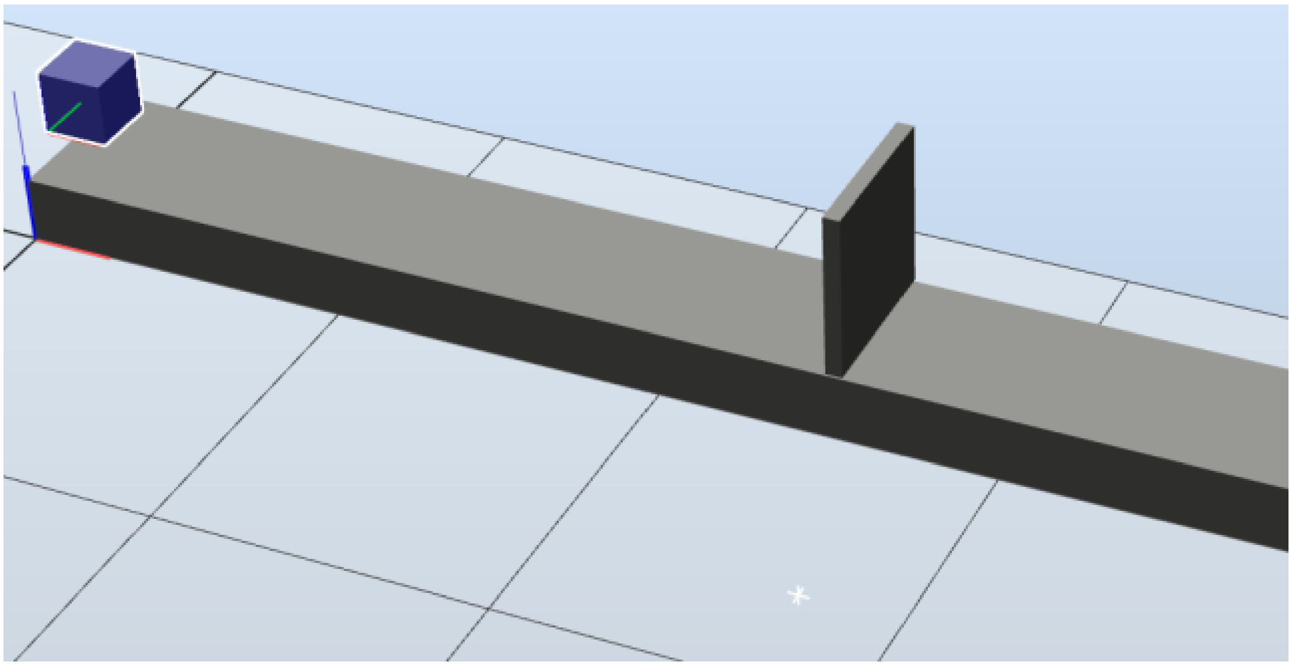
Drag your conveyor-gate mechanism to the created smart component.



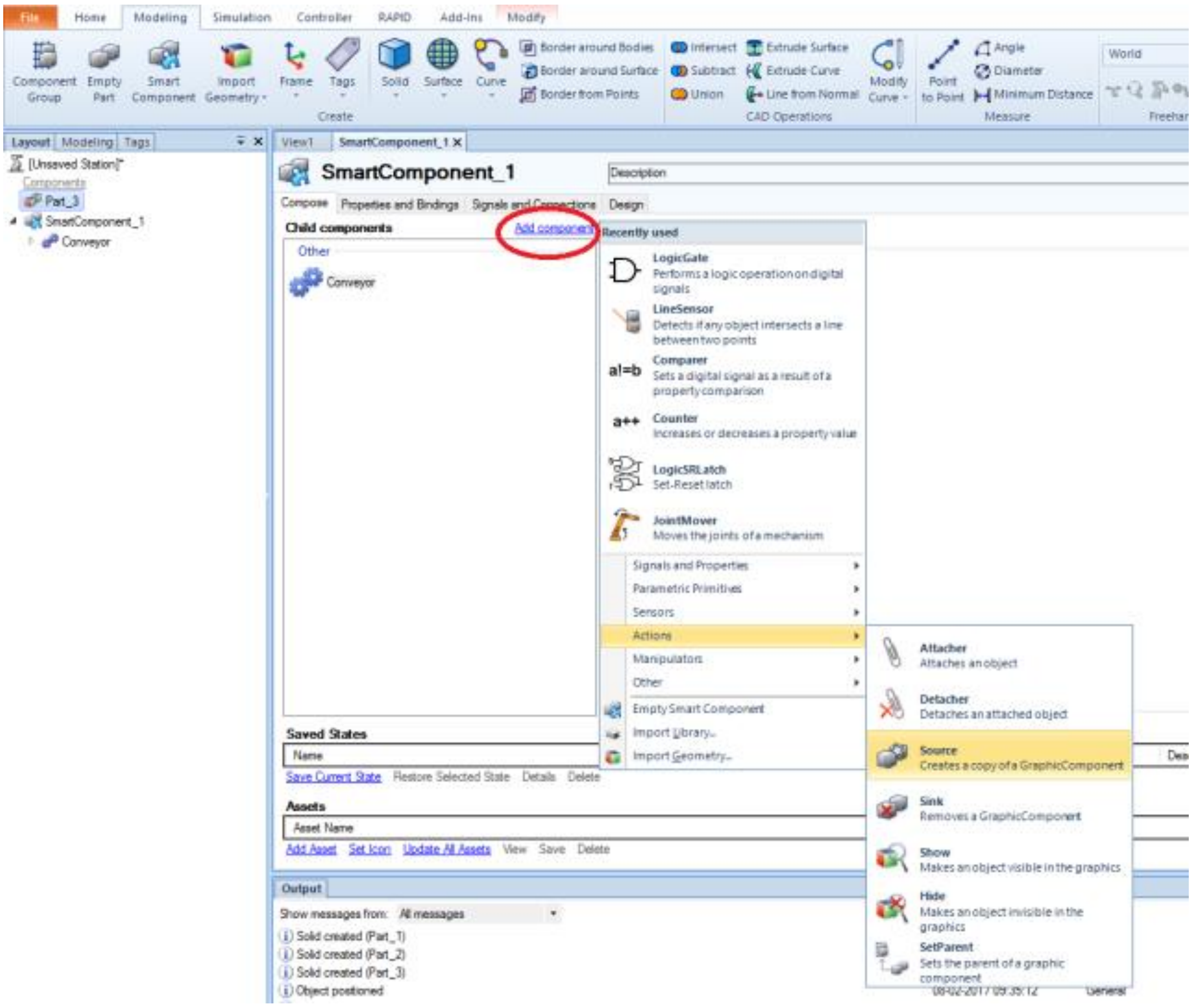
The smart component will then look like this.



Next, we want to create an object that can travel along the conveyor. In this example, we will just create a "Box" with the dimensions 200x200x200.

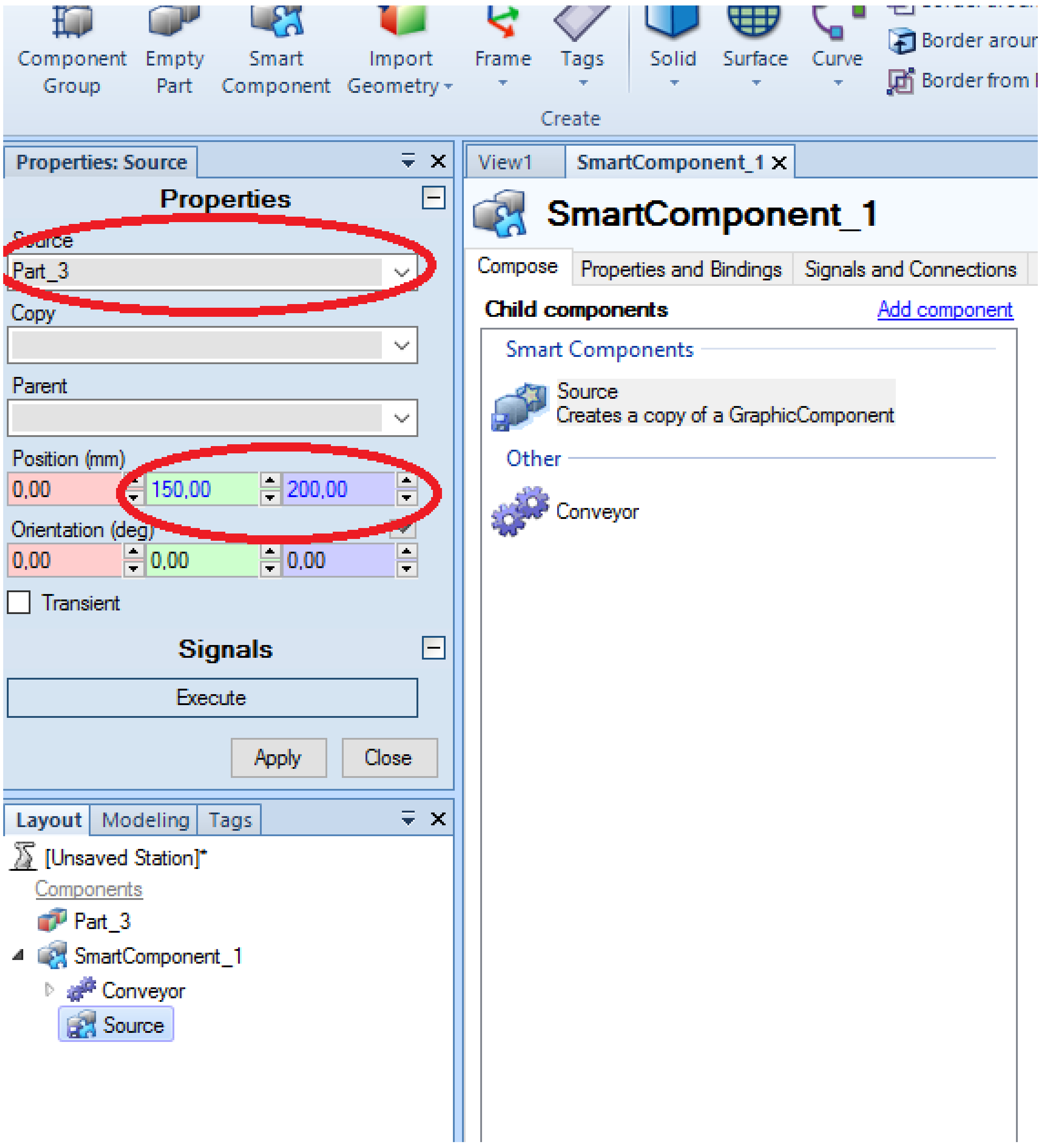


Go to the smart component view, click “Add component” and select “Source”. A source is component that can create an object, in this case, it will be used to create boxes.

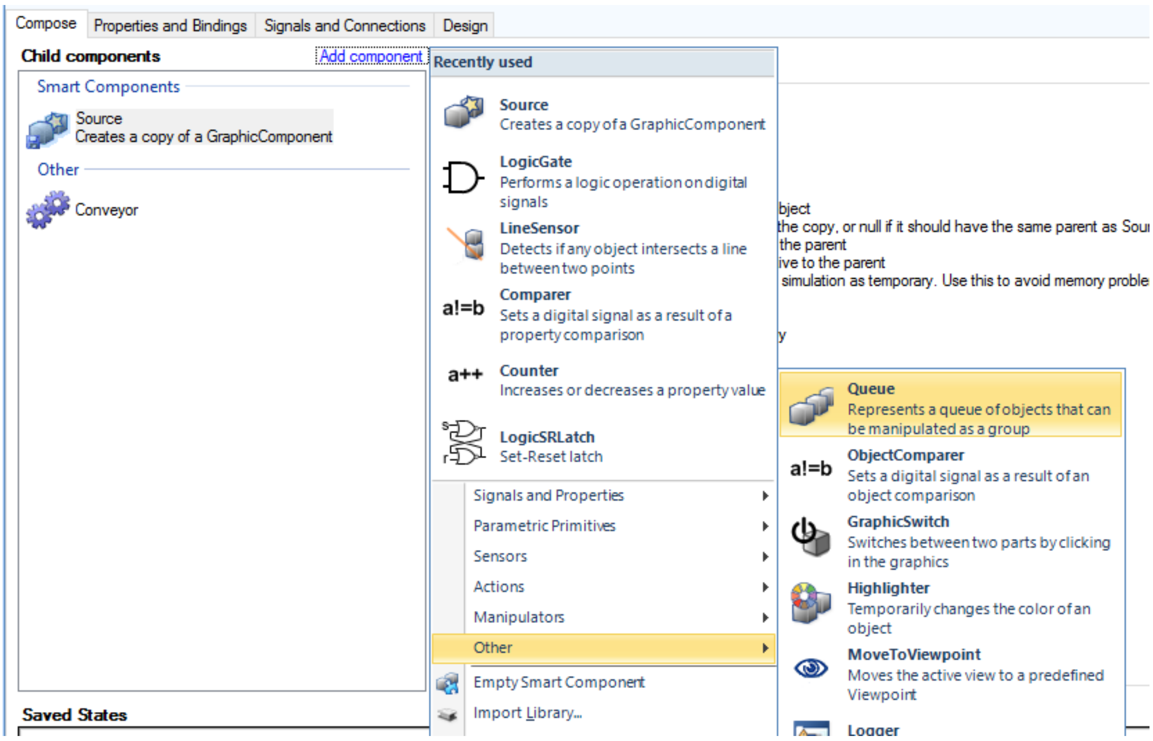


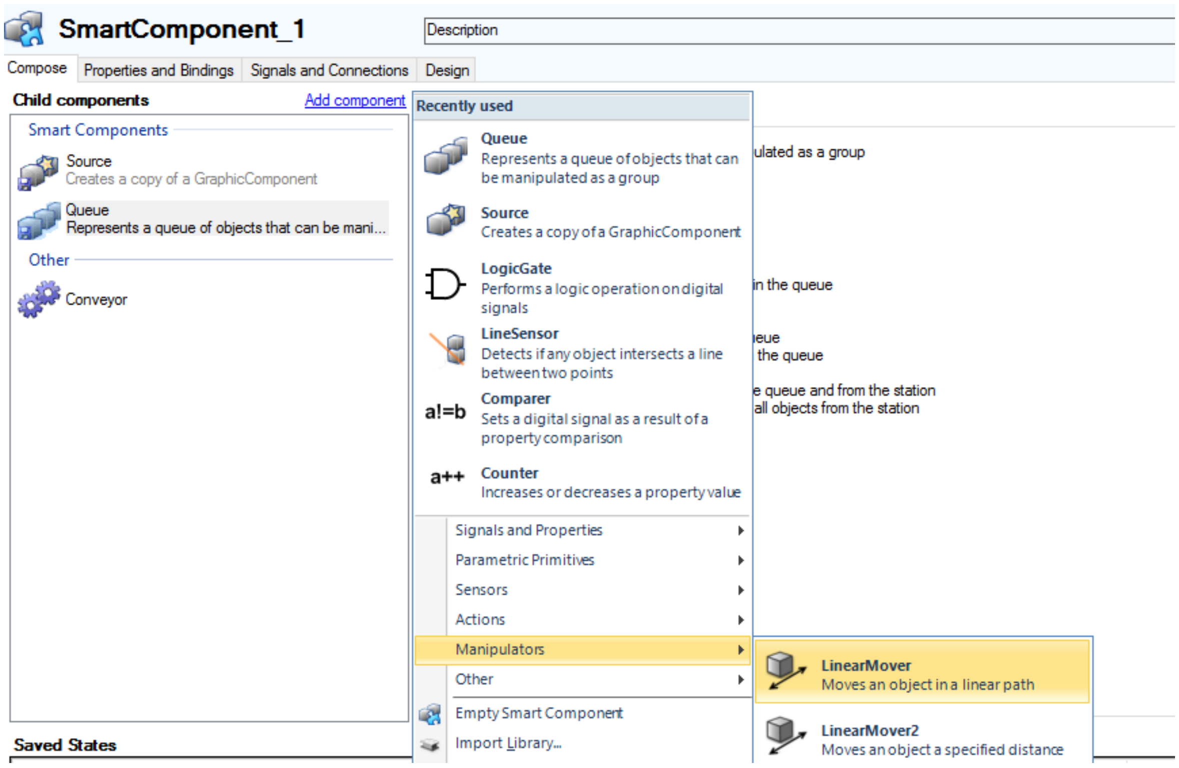
Solid created (Part_1)
Solid created (Part_2)
Solid created (Part_3)
Object positioned
08-02-2017 09:35:12 General

For the properties of the created source component, we will specify that it should create the box object when it is triggered, and it should place them at a specific position. Press “Apply” when the properties have been specified. After the properties have been applied to can press the execute button to check if the source functions as intended.

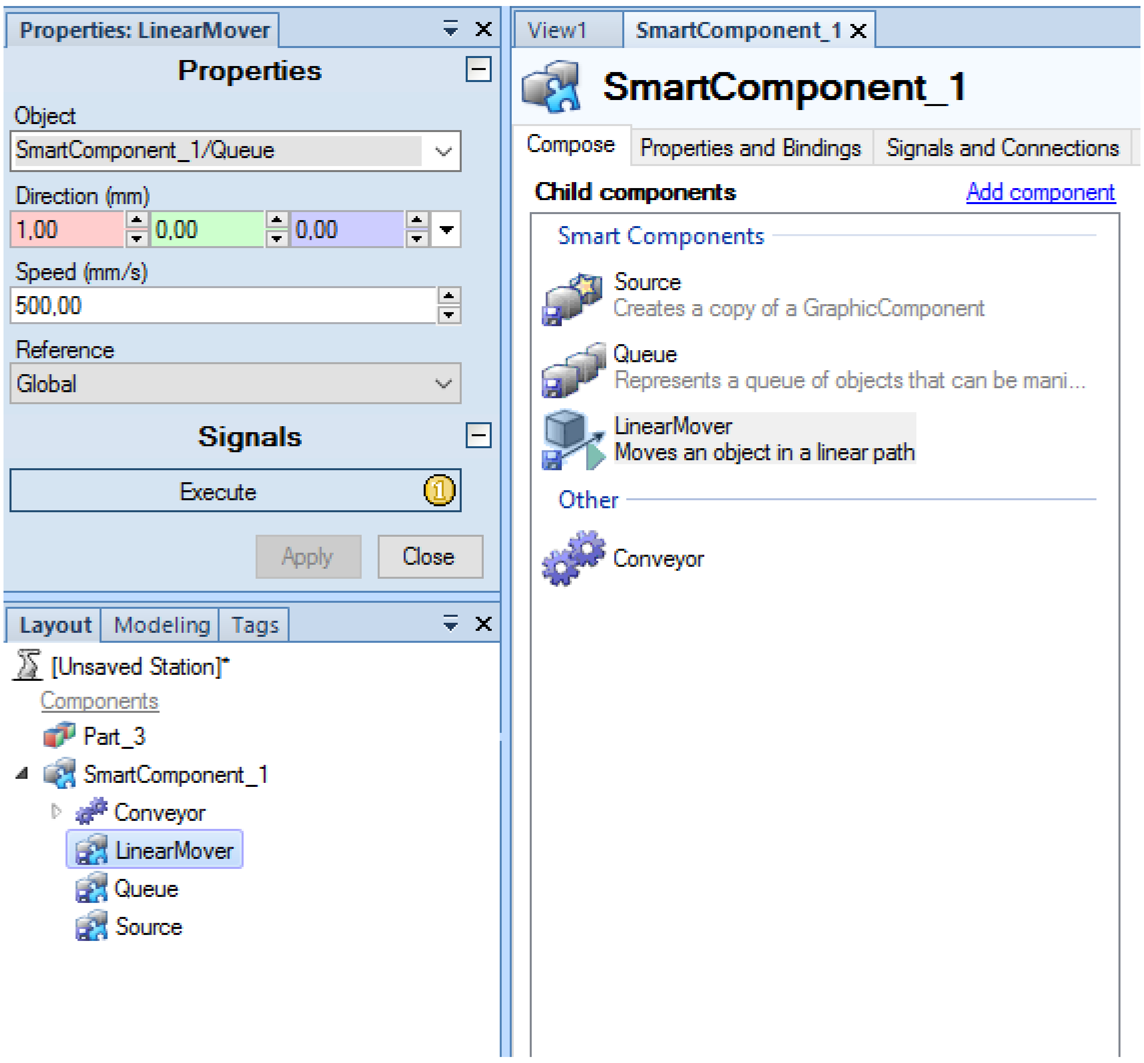


Next, create two more components, a “Queue” and a “LinearMover”.

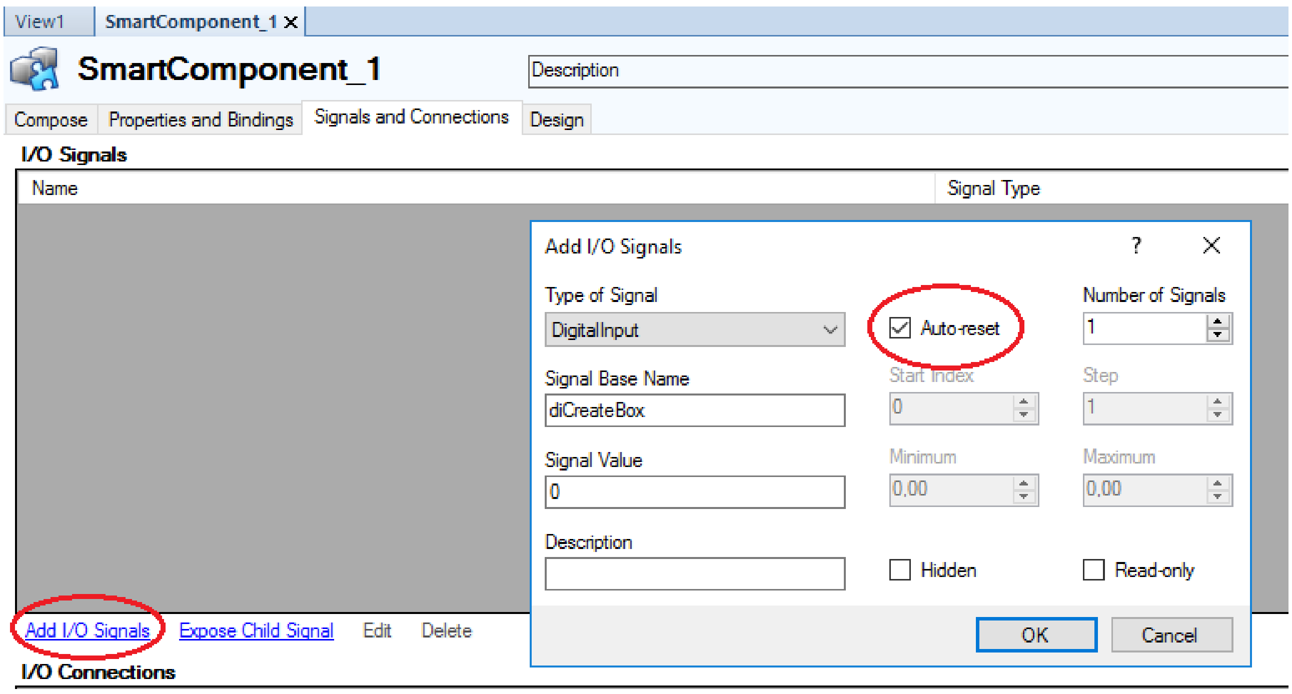

Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Other
Conveyor
Saved States
Recently used
Source
Creates a copy of a GraphicComponent
LogicGate
Performs a logic operation on digital signals
LineSensor
Detects if any object intersects a line between two points
Comparer
Sets a digital signal as a result of a property comparison
Counter
Increases or decreases a property value
LogicSRLatch
Set-Reset latch
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Queue
Represents a queue of objects that can be manipulated as a group
ObjectComparer
Sets a digital signal as a result of an object comparison
GraphicSwitch
Switches between two parts by clicking in the graphics
Highlighter
Temporarily changes the color of an object
MoveToViewpoint
Moves the active view to a predefined Viewpoint
Logger


SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
Other
Conveyor
Saved States
Recently used
Queue
Represents a queue of objects that can be manipulated as a group
Source
Creates a copy of a GraphicComponent
LogicGate
Performs a logic operation on digital signals
LineSensor
Detects if any object intersects a line between two points
Comparer
Sets a digital signal as a result of a property comparison
Counter
Increases or decreases a property value
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
LinearMover
Moves an object in a linear path
LinearMover2
Moves an object a specified distance

We do not need to specify any of the properties for the "Queue" component, but we will need to specify it for the "LinearMover" component. Specify that the component should move all objects in the "Queue", it should move it in the x-direction, and set the speed to "500". Remember to click "Apply" and make sure that "Execute" is active.



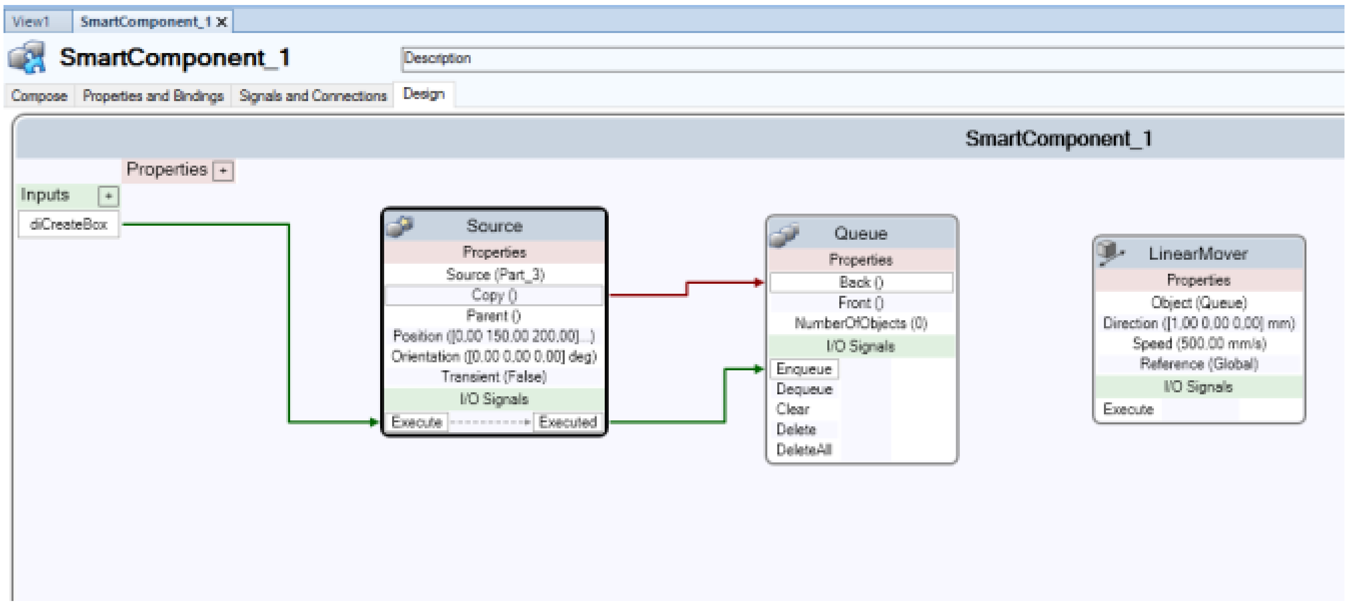
Next, we will create an I/O signals for our smart component to control the “Source” component. Go to “Signals and Connections” in the smart component view and click “Add I/O Signals”. Specify that it should be a “DigitalInput”, give it a name, and check that it should “Auto-reset” the signal.



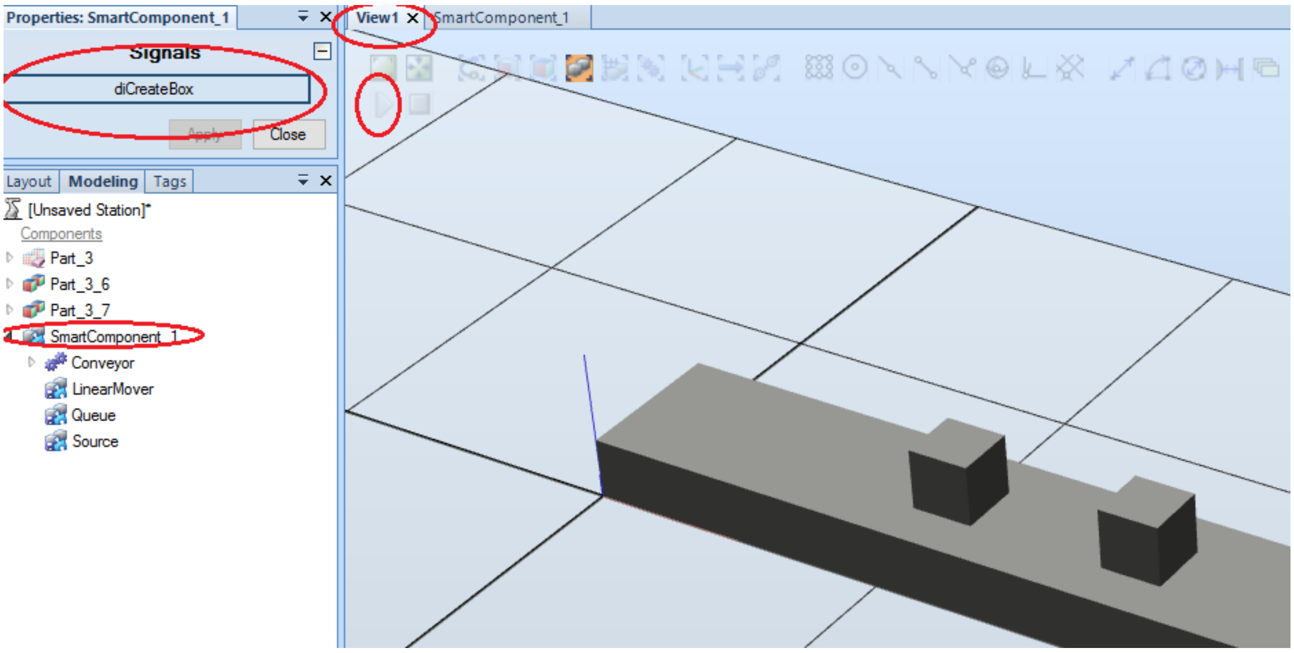
Add I/O Signals

Type of Signal: DigitalInput	[✓] Auto-reset	Number of Signals: 1
Signal Base Name: diCreateBox	Start Index: 0	Step: 1
Signal Value: 0	Minimum: 0,00	Maximum: 0,00
Description:	[ ] Hidden	[ ] Read-only

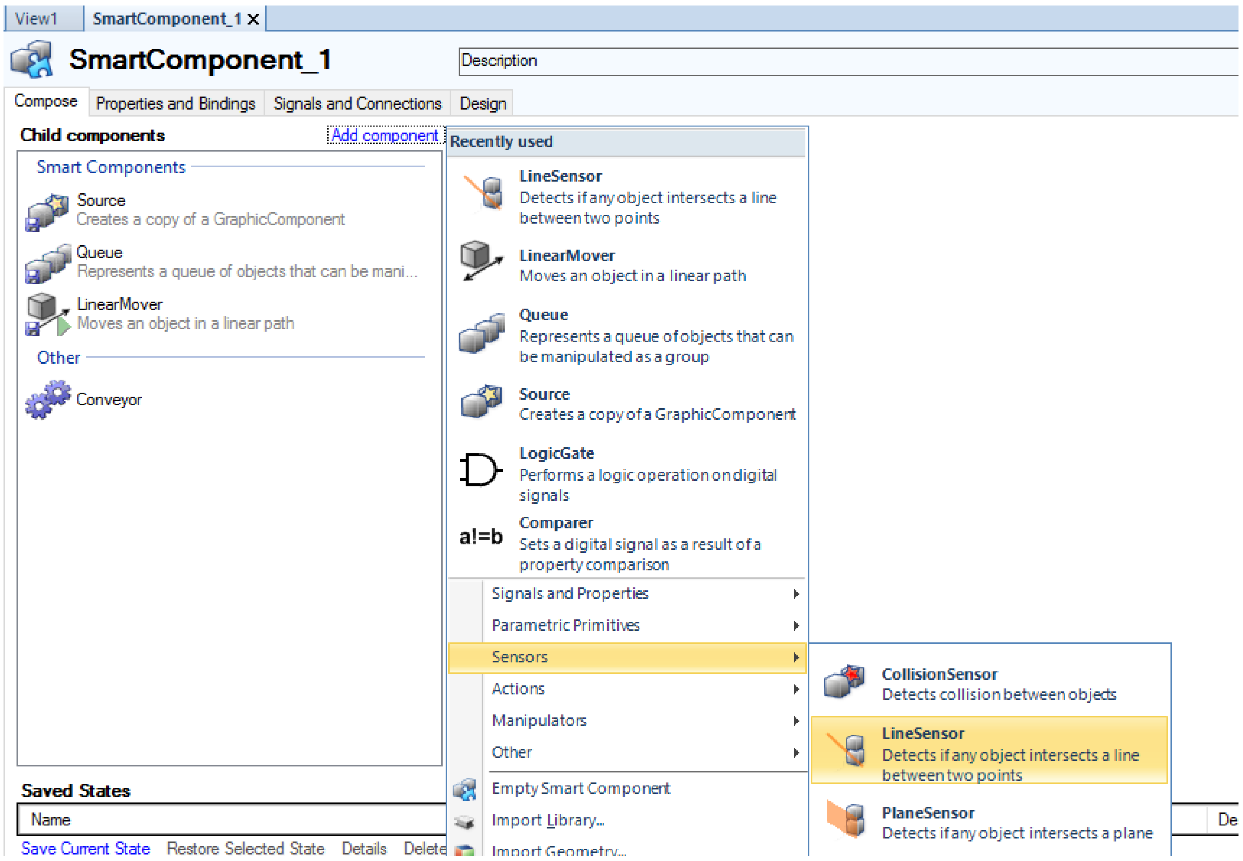
Next, we are going to “wire” the logic together to specify that the created I/O signal will trigger the source to create a box, and as soon the source have completed the task, the box is inserted in the queue. Remember that the LinearMover is specified to move all objects that’s in the queue. This is the logic needed to acquire a conveyor behaviour. Go to “Design” in the smart component view and drag the signals to achieve this configuration.



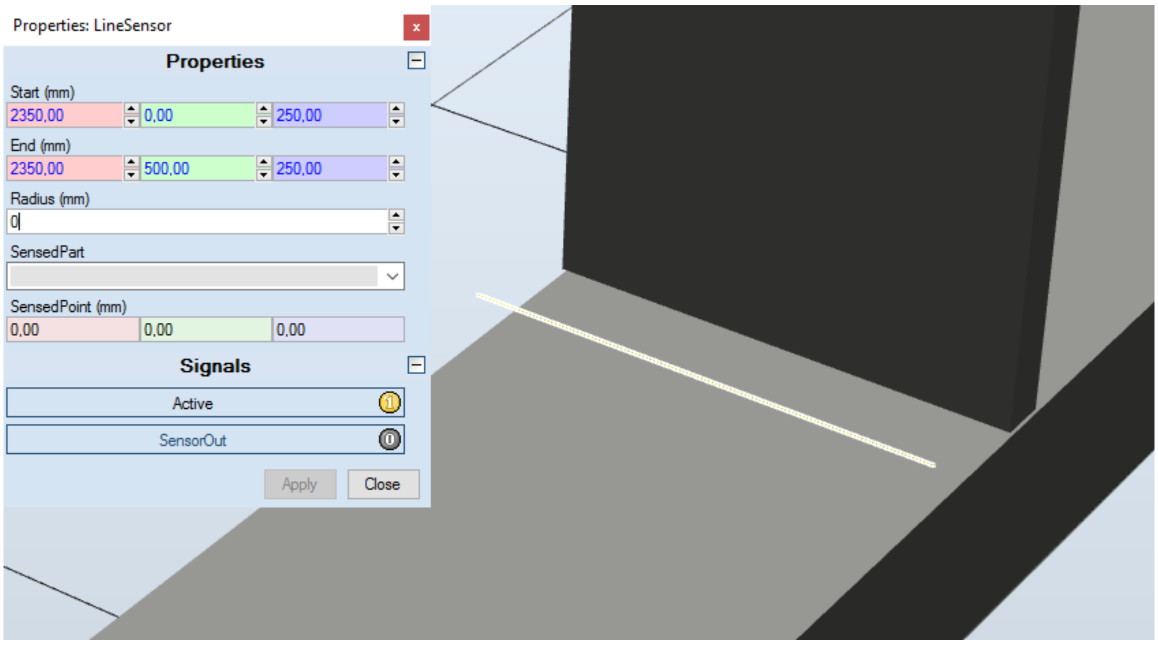
Feel free to test if the logic works. Go back to the view so you can see the environment. Click on the created smart component, start the simulation and trigger the I/O signal.



Next, we want to add a sensor that can detect an object before it arrives at the gate. Add a "LineSensor" component.


View1
SmartComponent_1
SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
LinearMover
Moves an object in a linear path
Other
Conveyor
Recently used
LineSensor
Detects if any object intersects a line between two points
LinearMover
Moves an object in a linear path
Queue
Represents a queue of objects that can be manipulated as a group
Source
Creates a copy of a GraphicComponent
LogicGate
Performs a logic operation on digital signals
Comparer
Sets a digital signal as a result of a property comparison
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
CollisionSensor
Detects collision between objects
LineSensor
Detects if any object intersects a line between two points
PlaneSensor
Detects if any object intersects a plane
Saved States
Name
Save Current State
Restore Selected State
Details
Delete

Specify the properties of the LineSensor like this.



Properties: LineSensor

Properties

Start (mm): 2350,00 | 0,00 | 250,00

End (mm): 2350,00 | 500,00 | 250,00

Radius (mm): 0

SensedPart:

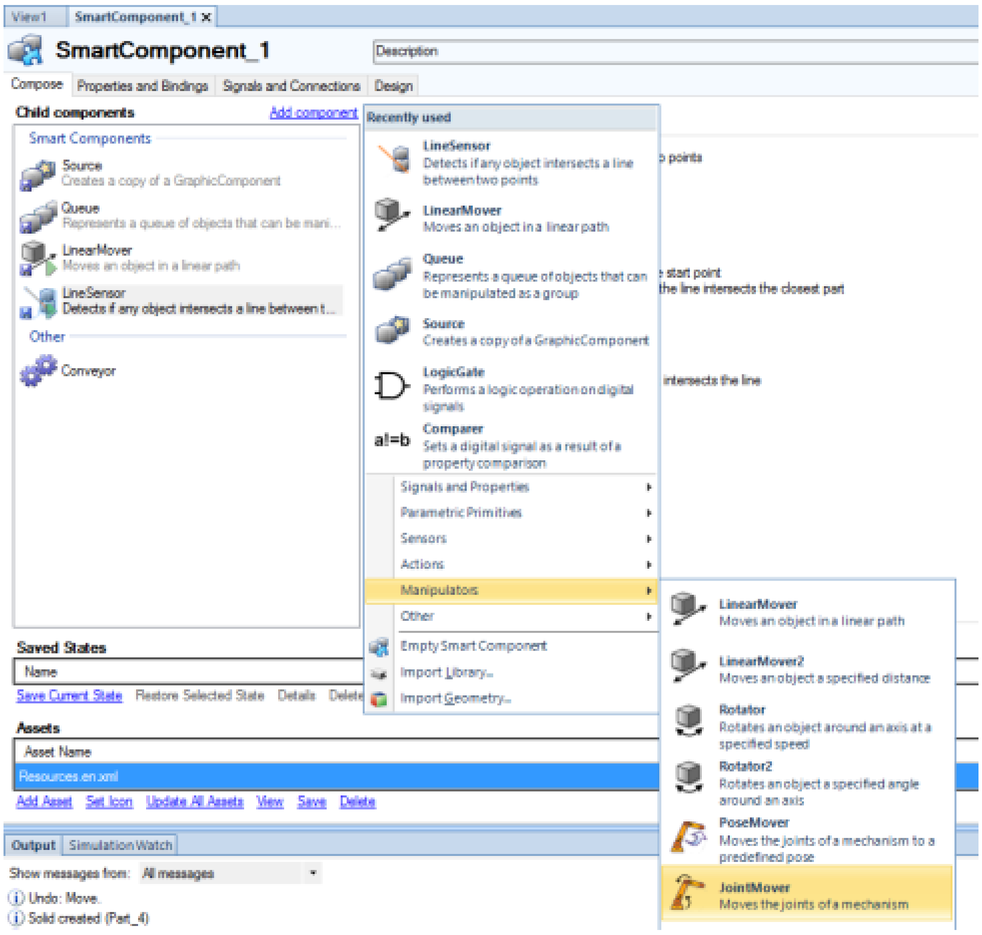
SensedPoint (mm): 0,00 | 0,00 | 0,00

Signals

Active: 1

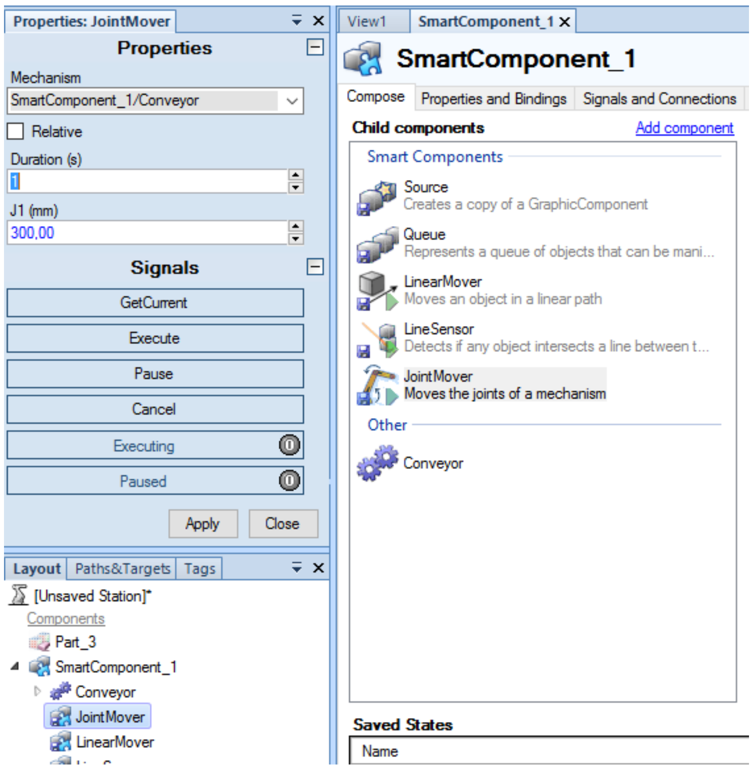
SensorOut: 0

Next, add a “JointMover” component



View1
SmartComponent_1
SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
LinearMover
Moves an object in a linear path
LineSensor
Detects if any object intersects a line between t...
Other
Conveyor
Recently used
LineSensor
Detects if any object intersects a line between two points
LinearMover
Moves an object in a linear path
Queue
Represents a queue of objects that can be manipulated as a group
Source
Creates a copy of a GraphicComponent
LogicGate
Performs a logic operation on digital signals
Comparer
Sets a digital signal as a result of a property comparison
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
LinearMover
Moves an object in a linear path
LinearMover2
Moves an object a specified distance
Rotator
Rotates an object around an axis at a specified speed
Rotator2
Rotates an object a specified angle around an axis
PoseMover
Moves the joints of a mechanism to a predefined pose
JointMover
Moves the joints of a mechanism
Saved States
Name
Save Current State
Restore Selected State
Details
Delete
Assets
Asset Name
Resources.en.xml
Add Asset
Set Icon
Update All Assets
View
Save
Delete
Output
Simulation Watch
Show messages from:
All messages
Undo: Move.
Solid created (Part_4)

Specify the properties of the JointMover like this.



Properties: JointMover

Properties

Mechanism: SmartComponent_1/Conveyor

Relative: unchecked

Duration (s): 1

J1 (mm): 300,00

Signals

GetCurrent

Execute

Pause

Cancel

Executing 0

Paused 0

Apply Close

Layout | Paths&Targets | Tags

[Unsaved Station]*
- Components
  - Part_3
  - SmartComponent_1
    - Conveyor
    - JointMover
    - LinearMover

View1 | SmartComponent_1

SmartComponent_1

Compose | Properties and Bindings | Signals and Connections

Child components — Add component

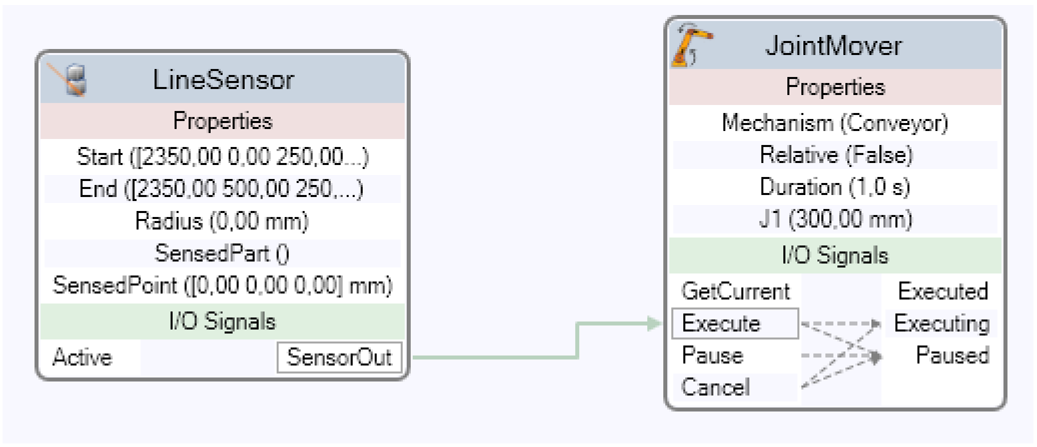
Smart Components
- Source — Creates a copy of a GraphicComponent
- Queue — Represents a queue of objects that can be mani...
- LinearMover — Moves an object in a linear path
- LineSensor — Detects if any object intersects a line between t...
- JointMover — Moves the joints of a mechanism

Other
- Conveyor

Saved States

Name

Wire the LineSensor and the JointMover like this. Specifying that when the sensor detects an object, start opening the gate.



LineSensor

Properties
- Start ([2350,00 0,00 250,00...)
- End ([2350,00 500,00 250,...)
- Radius (0,00 mm)
- SensedPart ()
- SensedPoint ([0,00 0,00 0,00] mm)

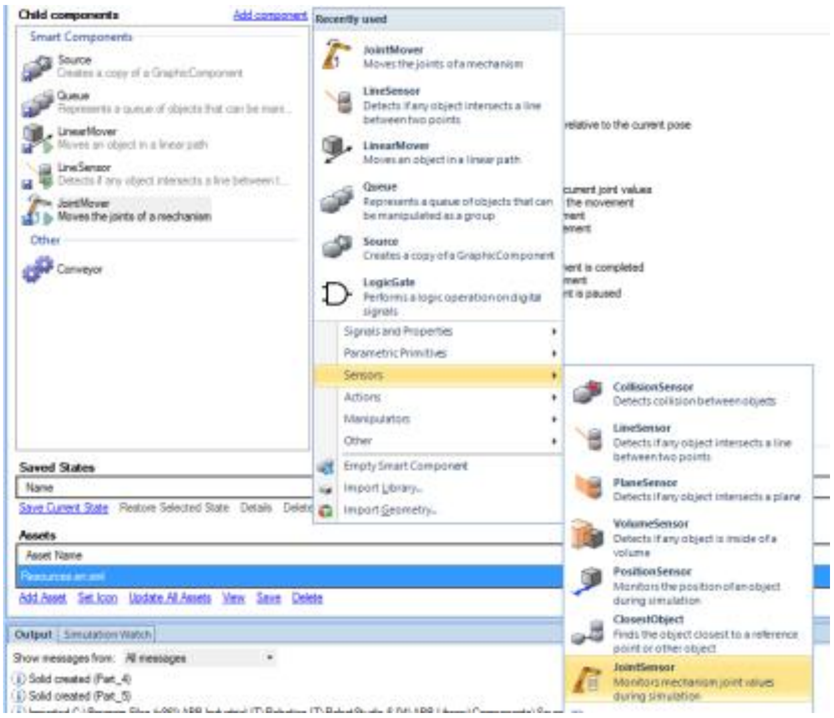
I/O Signals
- Active
- SensorOut

JointMover

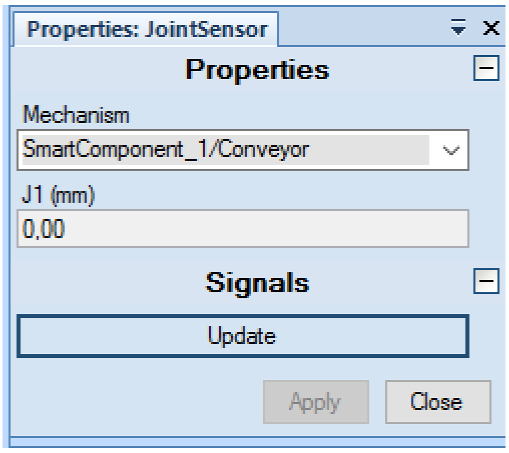
Properties
- Mechanism (Conveyor)
- Relative (False)
- Duration (1,0 s)
- J1 (300,00 mm)

I/O Signals
- GetCurrent
- Execute
- Pause
- Cancel
- Executed
- Executing
- Paused

We need to make sure that when the LineSensor detects an object, the conveyor will not move until the gate is completely open. Otherwise, there may be a risk that objects "collides" with the gate. To insure this, we need to add a few more components. The first is an "JointSensor" component, to read the position of the gate.



Specify the properties of the JointSensor like this.


Properties: JointSensor

Properties

Mechanism
SmartComponent_1/Conveyor

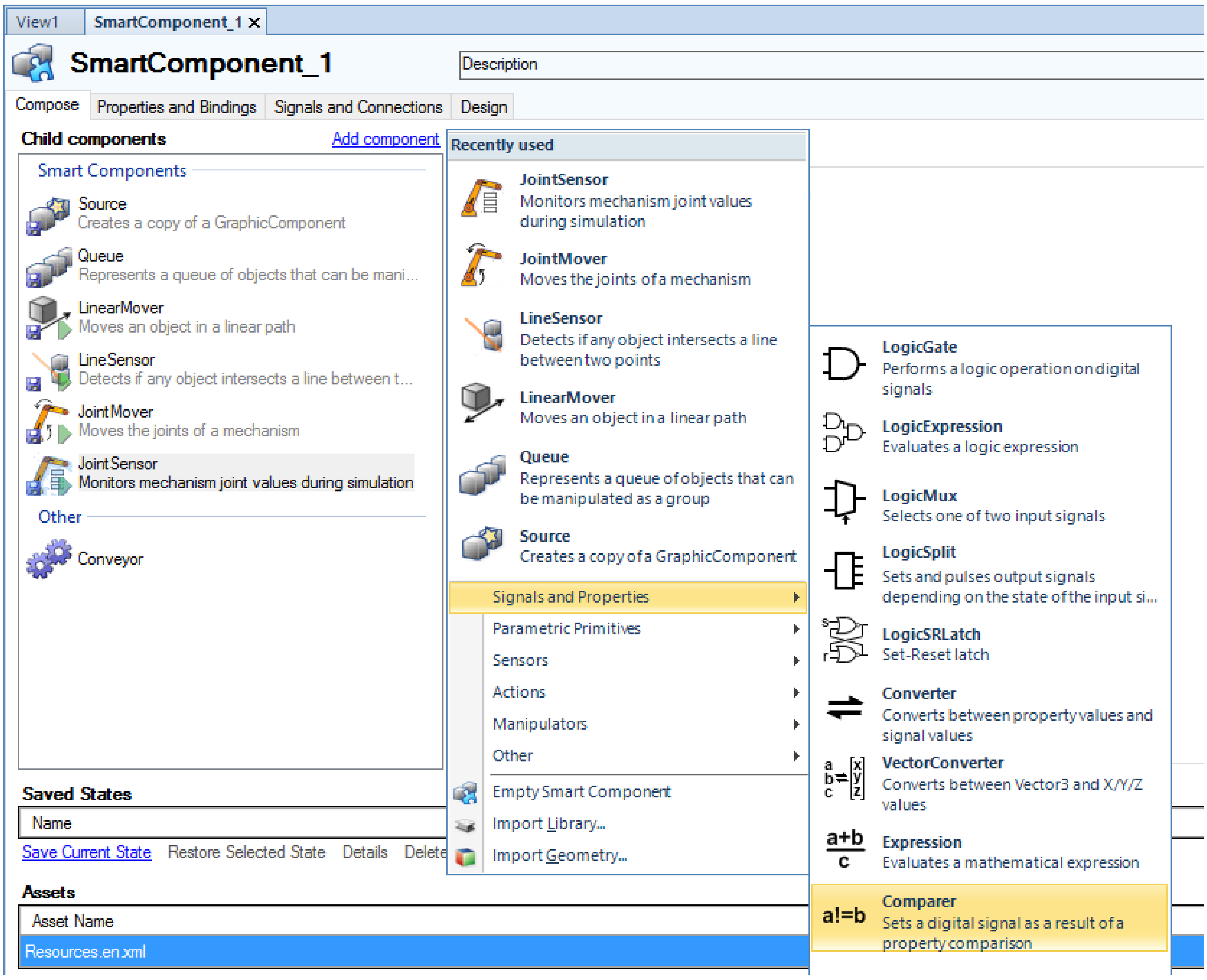
J1 (mm)
0,00

Signals

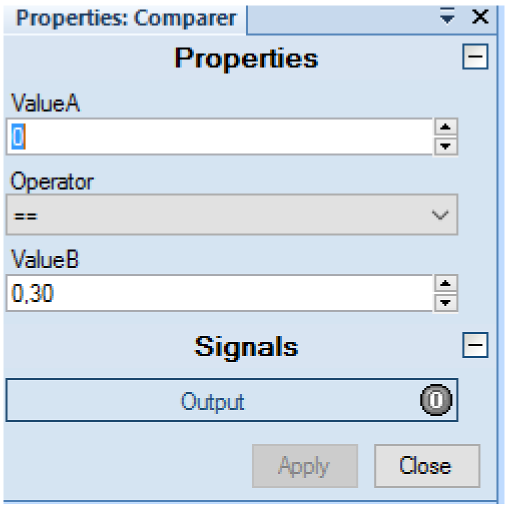
Update

Apply Close

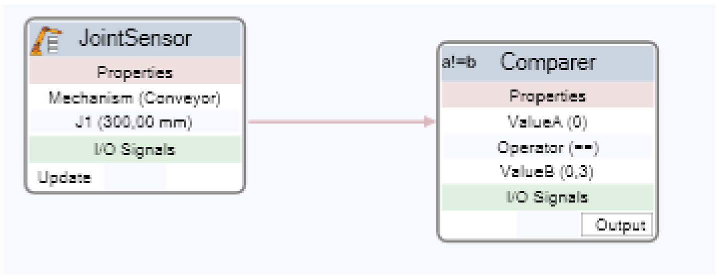
Next, add a “Comparer” component.


View1
SmartComponent_1
SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
LinearMover
Moves an object in a linear path
LineSensor
Detects if any object intersects a line between t...
JointMover
Moves the joints of a mechanism
JointSensor
Monitors mechanism joint values during simulation
Other
Conveyor
Recently used
JointSensor
Monitors mechanism joint values during simulation
JointMover
Moves the joints of a mechanism
LineSensor
Detects if any object intersects a line between two points
LinearMover
Moves an object in a linear path
Queue
Represents a queue of objects that can be manipulated as a group
Source
Creates a copy of a GraphicComponent
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
LogicGate
Performs a logic operation on digital signals
LogicExpression
Evaluates a logic expression
LogicMux
Selects one of two input signals
LogicSplit
Sets and pulses output signals depending on the state of the input si...
LogicSRLatch
Set-Reset latch
Converter
Converts between property values and signal values
VectorConverter
Converts between Vector3 and X/Y/Z values
Expression
Evaluates a mathematical expression
Comparer
Sets a digital signal as a result of a property comparison
Saved States
Name
Save Current State
Restore Selected State
Details
Delete
Assets
Asset Name
Resources.en.xml

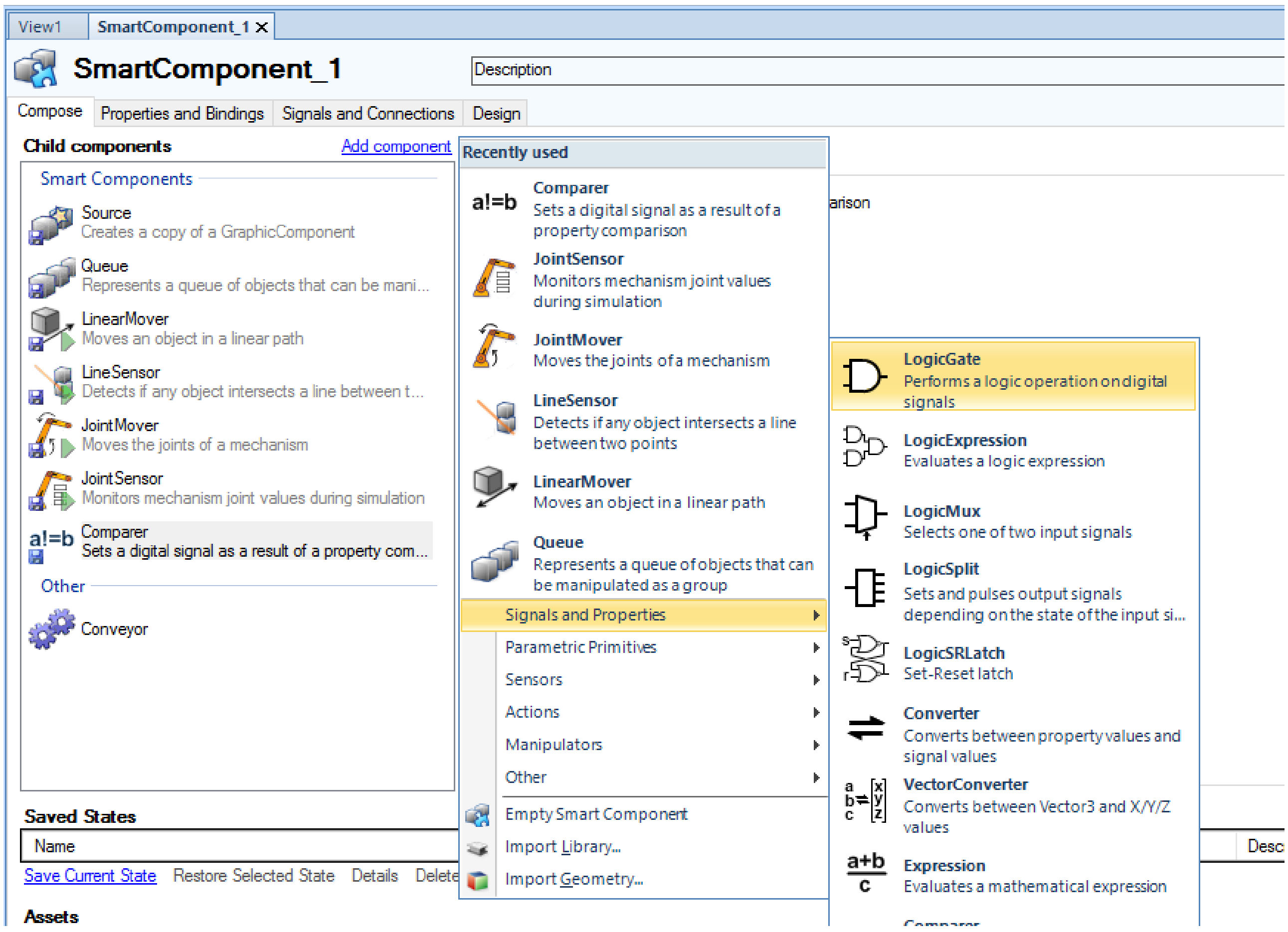
Specify the properties of the Comparer like this.



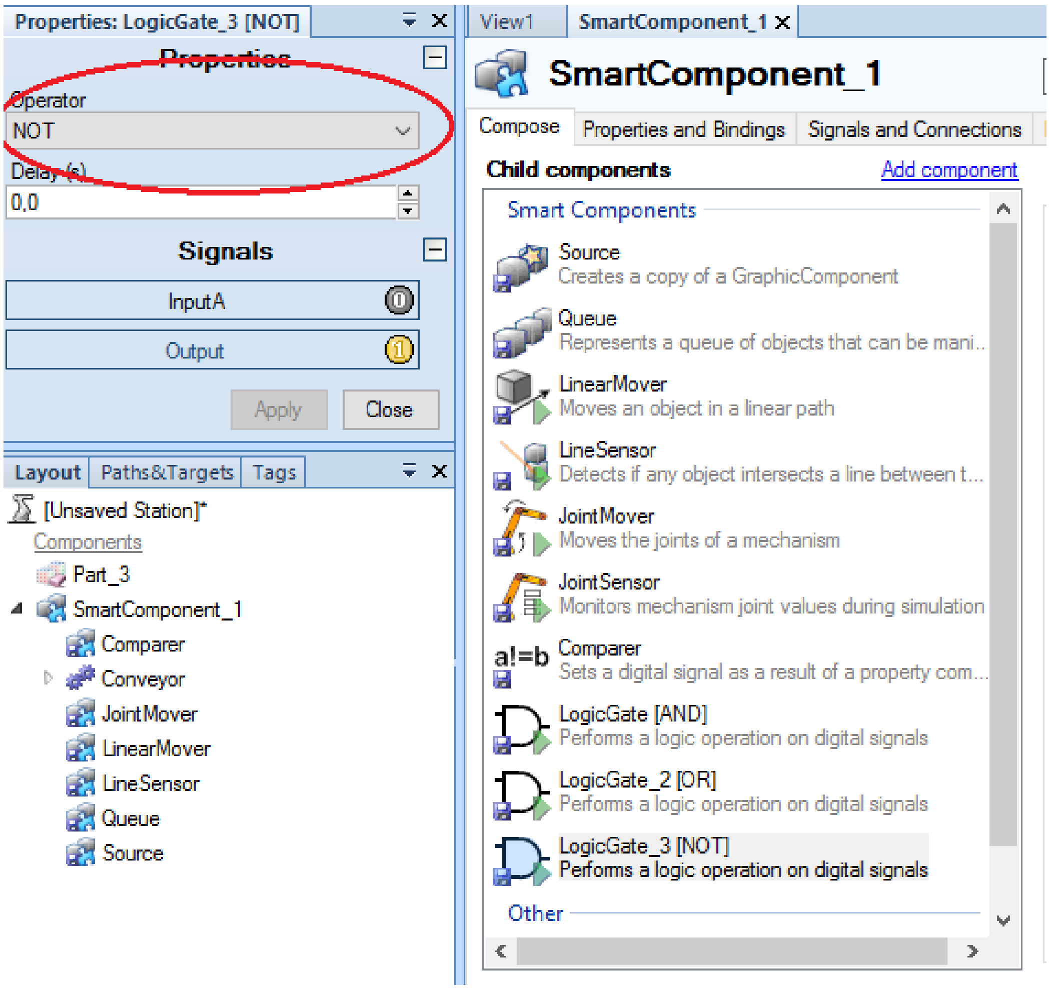
Wire the JointSensor and Comparer like this



Next, we need to create 2 logic gates, a “Not” logic gate, an “Or” logic gate and an “And” logic gate.


View1
SmartComponent_1
SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Source
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
LinearMover
Moves an object in a linear path
LineSensor
Detects if any object intersects a line between t...
JointMover
Moves the joints of a mechanism
JointSensor
Monitors mechanism joint values during simulation
Comparer
Sets a digital signal as a result of a property com...
Other
Conveyor
Recently used
Comparer
Sets a digital signal as a result of a property comparison
JointSensor
Monitors mechanism joint values during simulation
JointMover
Moves the joints of a mechanism
LineSensor
Detects if any object intersects a line between two points
LinearMover
Moves an object in a linear path
Queue
Represents a queue of objects that can be manipulated as a group
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
LogicGate
Performs a logic operation on digital signals
LogicExpression
Evaluates a logic expression
LogicMux
Selects one of two input signals
LogicSplit
Sets and pulses output signals depending on the state of the input si...
LogicSRLatch
Set-Reset latch
Converter
Converts between property values and signal values
VectorConverter
Converts between Vector3 and X/Y/Z values
Expression
Evaluates a mathematical expression
arison
Saved States
Name
Desc
Save Current State
Restore Selected State
Details
Delete
Assets

Specify the properties of the 3 different logic gates to be an “And”, an “Or” and a “Not”. Make sure to trigger “InputB” on the “And” logic gate.


Properties: LogicGate_3 [NOT]

Properties

Operator: NOT

Delay (s): 0,0

Signals

InputA 0

Output 1

SmartComponent_1

Child components

Smart Components

Source — Creates a copy of a GraphicComponent

Queue — Represents a queue of objects that can be mani...

LinearMover — Moves an object in a linear path

LineSensor — Detects if any object intersects a line between t...

JointMover — Moves the joints of a mechanism

JointSensor — Monitors mechanism joint values during simulation

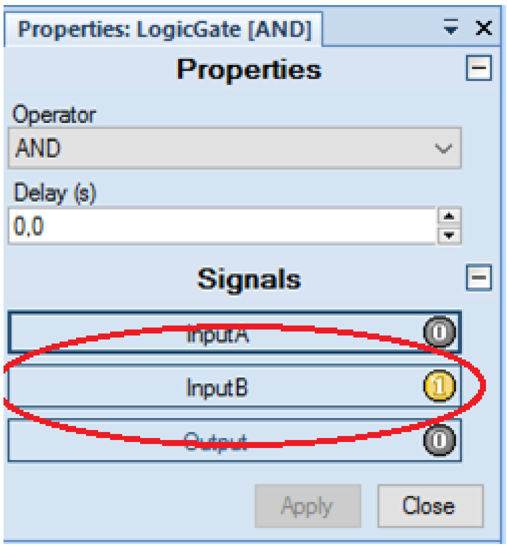
Comparer — Sets a digital signal as a result of a property com...

LogicGate [AND] — Performs a logic operation on digital signals

LogicGate_2 [OR] — Performs a logic operation on digital signals

LogicGate_3 [NOT] — Performs a logic operation on digital signals

Other


Properties: LogicGate [AND]

Properties

Operator: AND

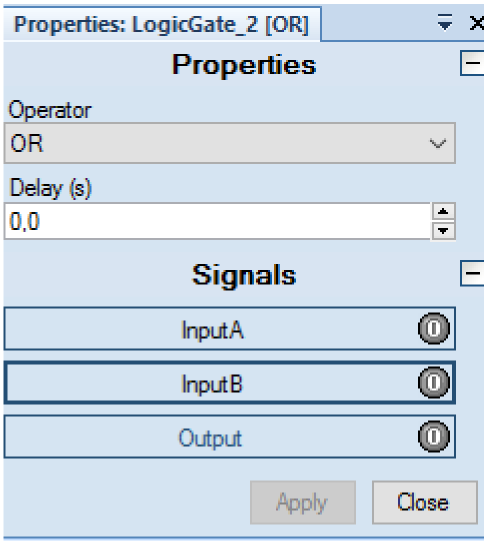
Delay (s): 0,0

Signals

InputA 0

InputB 1

Output 0


Properties: LogicGate_2 [OR]

Properties

Operator: OR

Delay (s): 0,0

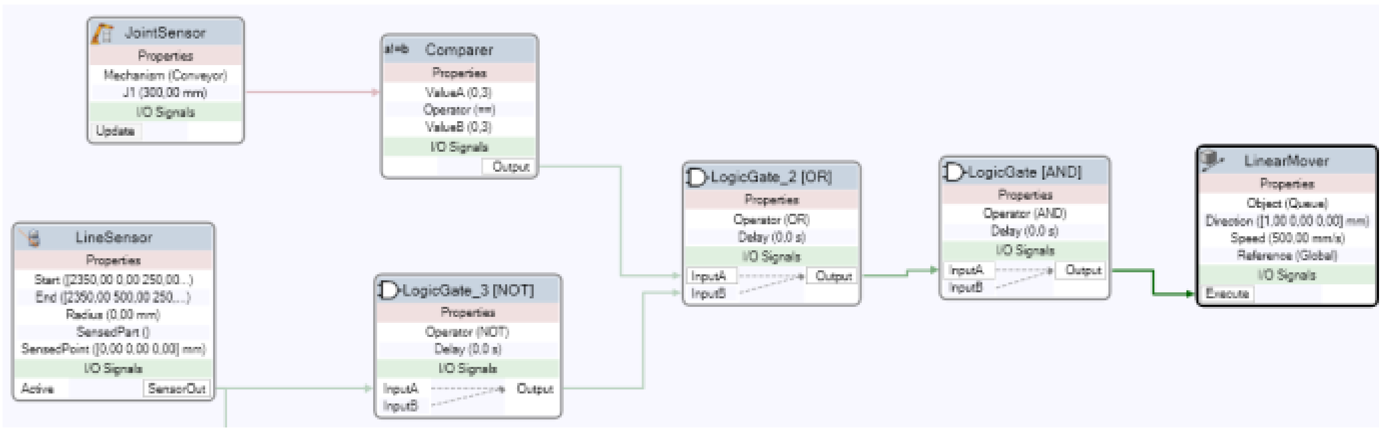
Signals

InputA 0

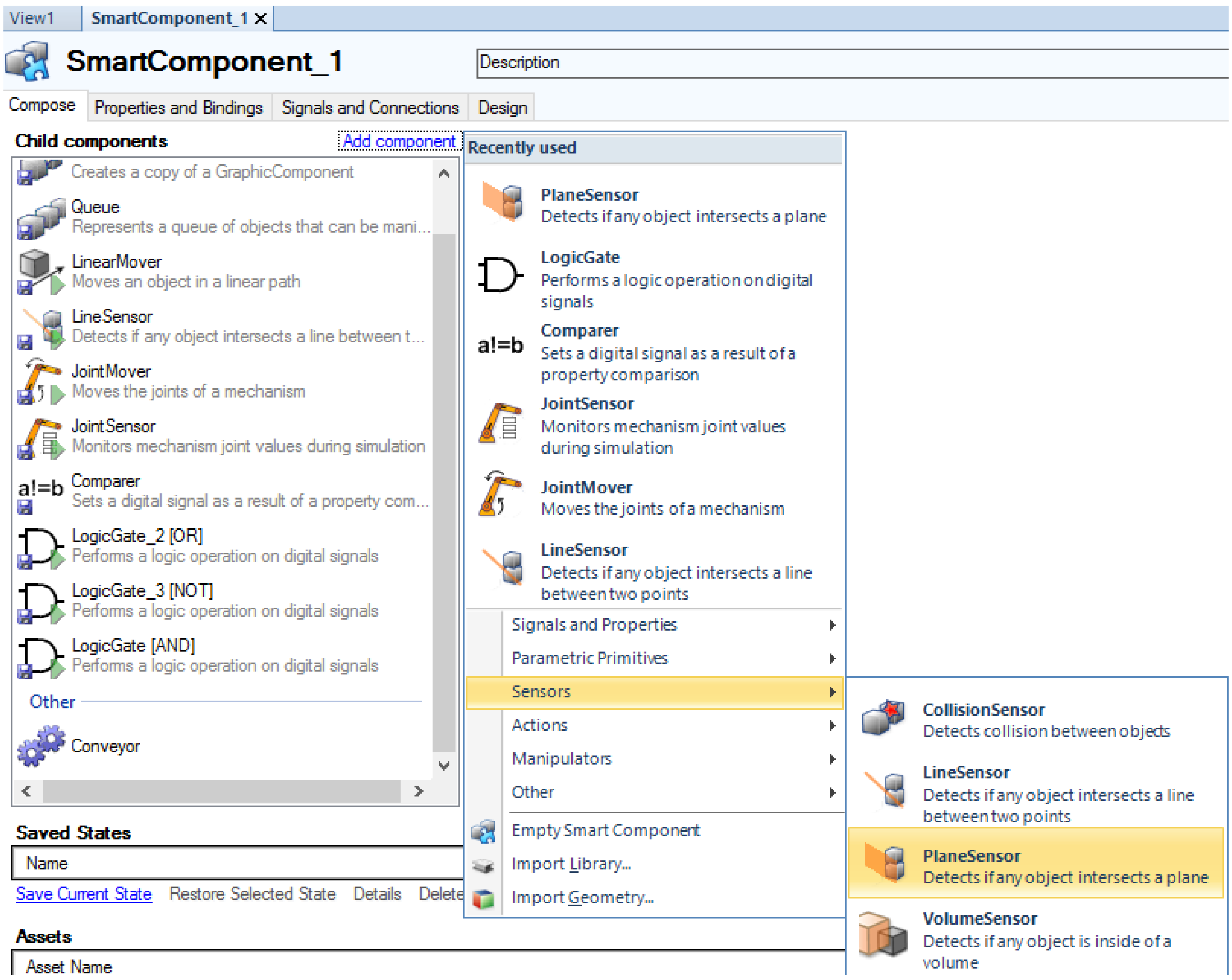
InputB 0

Output 0

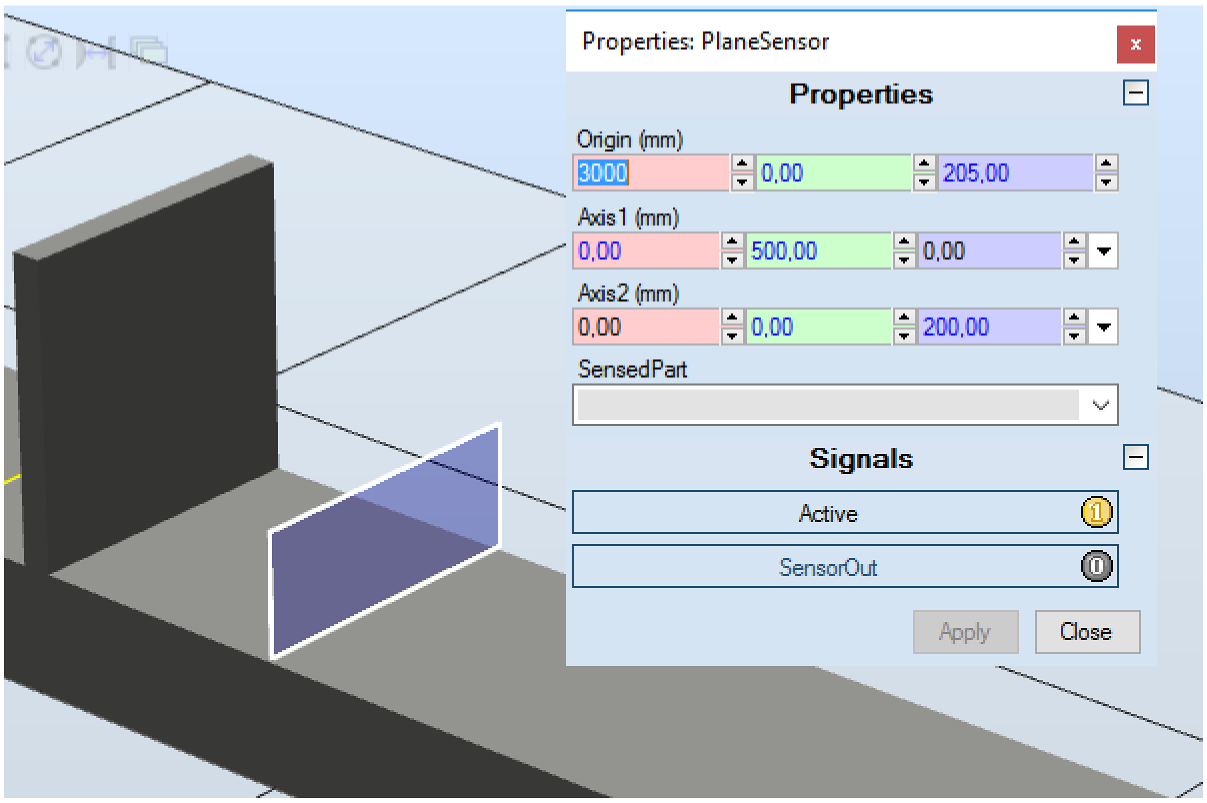
Wire the shown components like this



Next, we are going to add a sensor to detect when an object has moved passed the gate. Here we will use a “PlaneSenor”.


View1
SmartComponent_1 ×
SmartComponent_1
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Creates a copy of a GraphicComponent
Queue
Represents a queue of objects that can be mani...
LinearMover
Moves an object in a linear path
LineSensor
Detects if any object intersects a line between t...
JointMover
Moves the joints of a mechanism
JointSensor
Monitors mechanism joint values during simulation
Comparer
Sets a digital signal as a result of a property com...
LogicGate_2 [OR]
Performs a logic operation on digital signals
LogicGate_3 [NOT]
Performs a logic operation on digital signals
LogicGate [AND]
Performs a logic operation on digital signals
Other
Conveyor
Saved States
Name
Save Current State
Restore Selected State
Details
Delete
Assets
Asset Name
Recently used
PlaneSensor
Detects if any object intersects a plane
LogicGate
Performs a logic operation on digital signals
a!=b
Comparer
Sets a digital signal as a result of a property comparison
JointSensor
Monitors mechanism joint values during simulation
JointMover
Moves the joints of a mechanism
LineSensor
Detects if any object intersects a line between two points
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
CollisionSensor
Detects collision between objects
LineSensor
Detects if any object intersects a line between two points
PlaneSensor
Detects if any object intersects a plane
VolumeSensor
Detects if any object is inside of a volume

Specify the properties of the JointSensor like this.



Properties: PlaneSensor

Properties

Origin (mm)		
3000	0,00	205,00
Axis1 (mm)		
0,00	500,00	0,00
Axis2 (mm)		
0,00	0,00	200,00

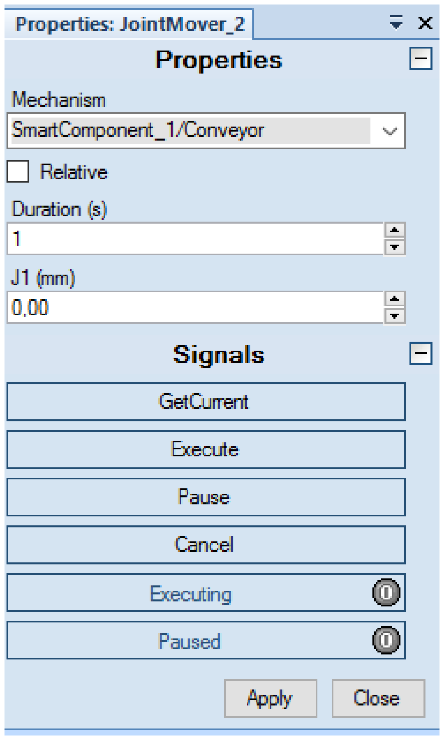
SensedPart

Signals

Active 1

SensorOut 0

Next, add another "JointMover" and specify the properties like this.



Properties: JointMover_2

Properties

Mechanism: SmartComponent_1/Conveyor

☐ Relative

Duration (s): 1

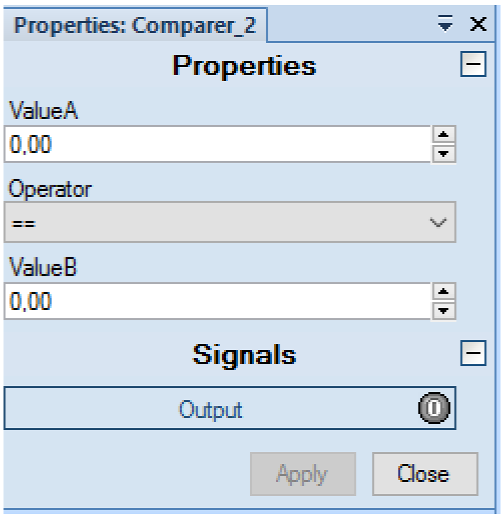
J1 (mm): 0,00

Signals

Executing 0

Paused 0

Add another "Comparer", but here you don't need to change the properties.


Properties: Comparer_2

Properties

ValueA: 0,00

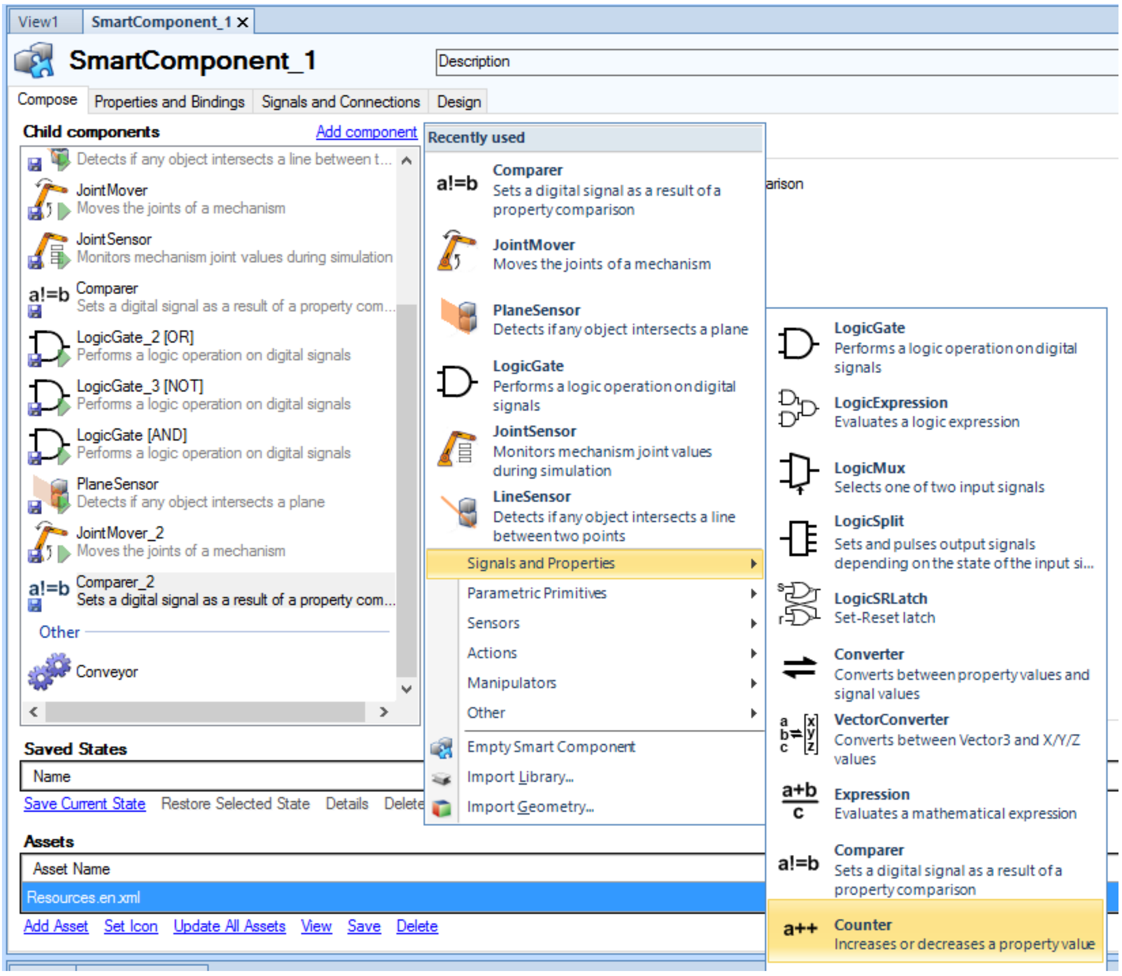
Operator: ==

ValueB: 0,00

Signals

Output: 0

Next, add a "Counter" Component

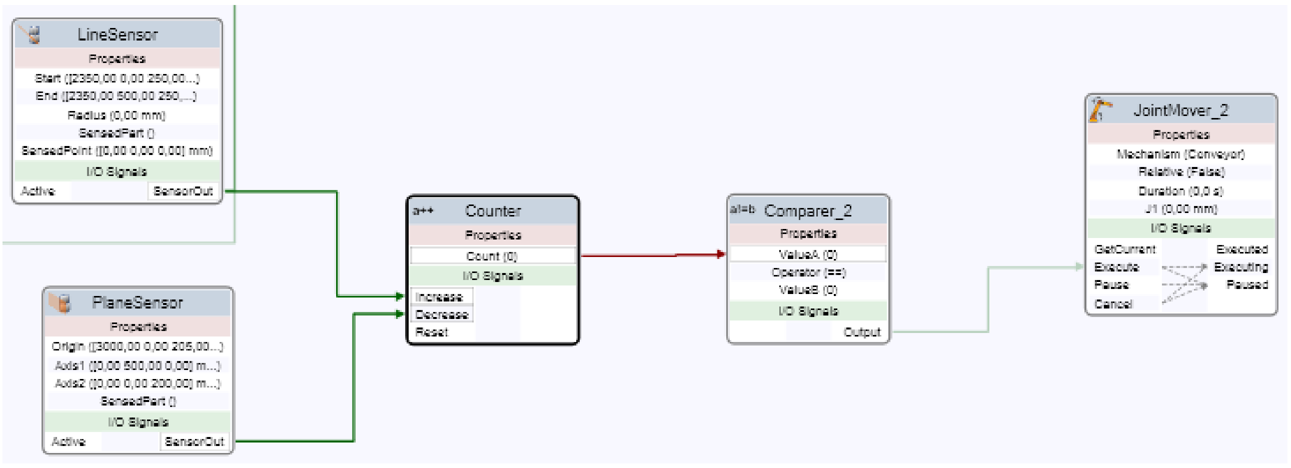


Detects if any object intersects a line between t...
JointMover — Moves the joints of a mechanism
JointSensor — Monitors mechanism joint values during simulation
Comparer — Sets a digital signal as a result of a property com...
LogicGate_2 [OR] — Performs a logic operation on digital signals
LogicGate_3 [NOT] — Performs a logic operation on digital signals
LogicGate [AND] — Performs a logic operation on digital signals
PlaneSensor — Detects if any object intersects a plane
JointMover_2 — Moves the joints of a mechanism
Comparer_2 — Sets a digital signal as a result of a property com...
Other
Conveyor

Recently used
Comparer — Sets a digital signal as a result of a property comparison
JointMover — Moves the joints of a mechanism
PlaneSensor — Detects if any object intersects a plane
LogicGate — Performs a logic operation on digital signals
JointSensor — Monitors mechanism joint values during simulation
LineSensor — Detects if any object intersects a line between two points

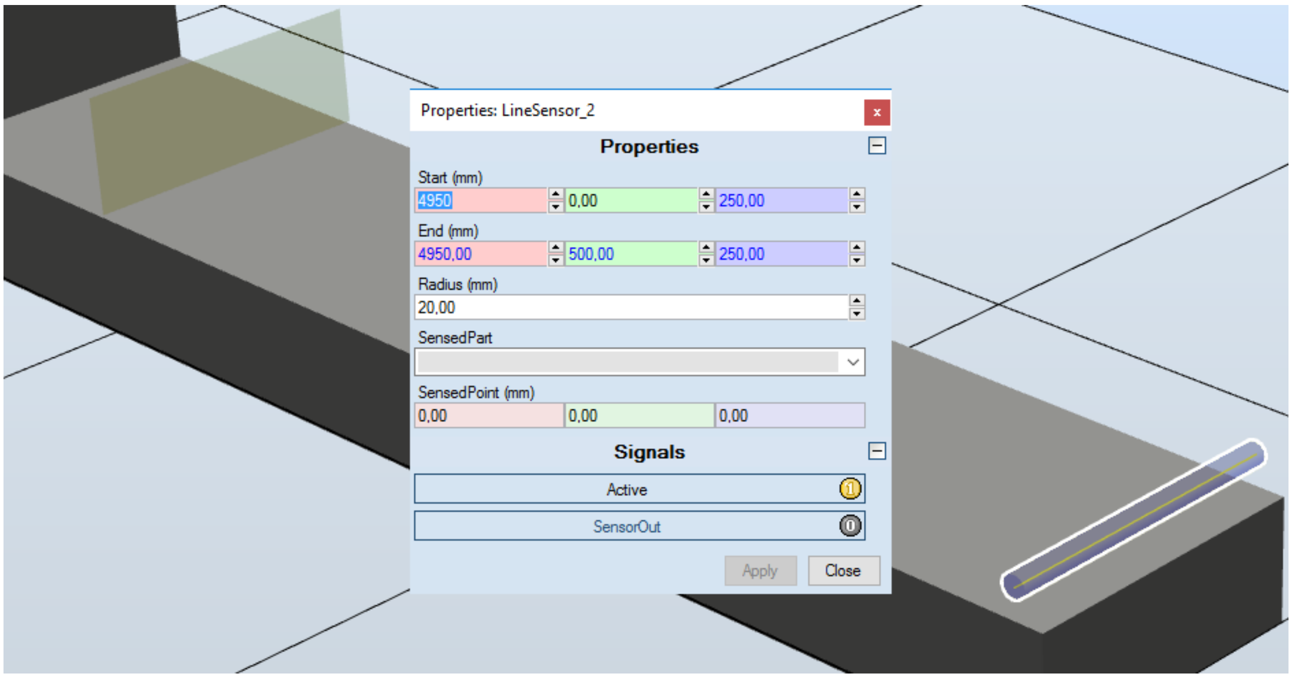
LogicGate — Performs a logic operation on digital signals
LogicExpression — Evaluates a logic expression
LogicMux — Selects one of two input signals
LogicSplit — Sets and pulses output signals depending on the state of the input si...
LogicSRLatch — Set-Reset latch
Converter — Converts between property values and signal values
VectorConverter — Converts between Vector3 and X/Y/Z values
Expression — Evaluates a mathematical expression
Comparer — Sets a digital signal as a result of a property comparison
Counter — Increases or decreases a property value

Wire the added components like this.



The reason why we needed a “Counter” and a “Comparer” is to make sure that the gate won’t close whenever an object passes the “PlaneSensor”, but only we are sure that a second object is not underneath the gate.

The last feature we want to create is to make the conveyor stop when an object has reached the end of the conveyor. To do this, we will add another “LineSensor”, and another “Not” logic gate. Specify the properties of the new “LineSensor” like this.



Properties: LineSensor_2

Properties

Start (mm): 4950 | 0,00 | 250,00

End (mm): 4950,00 | 500,00 | 250,00

Radius (mm): 20,00

SensedPart:

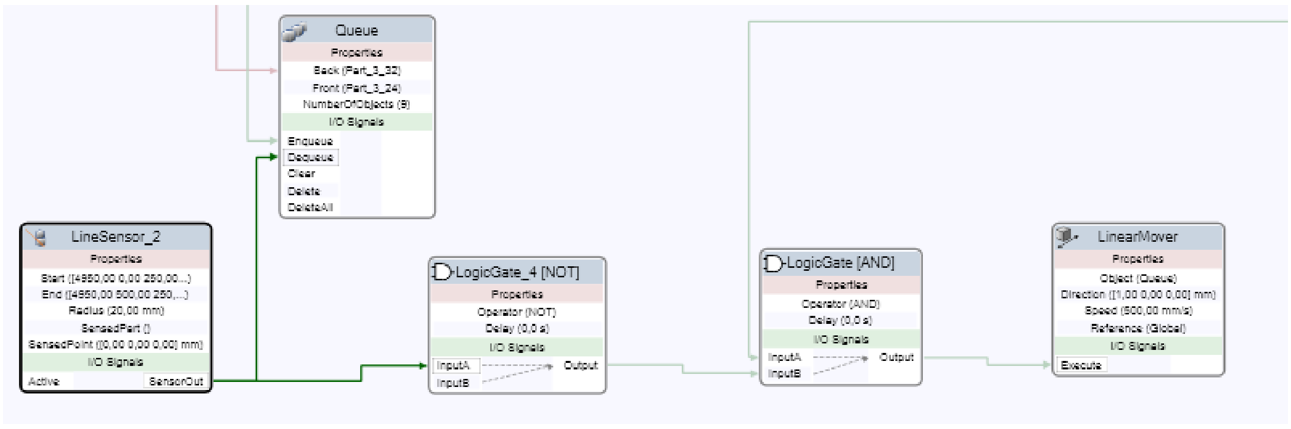
SensedPoint (mm): 0,00 | 0,00 | 0,00

Signals

Active

SensorOut

And now to the final wiring. Be aware that there is a connection from the new LineSensor to the Queue. As you remember the LinearMover moves everything that's in the Queue. This means that even though we are moving an object away from what we see as the conveyor, the logic dictates the LinearMover still effects an object if it's in the Queue.



Queue
Properties
Back (Part_3_32)
Front (Part_3_24)
NumberOfObjects (9)
I/O Signals
Enqueue
Dequeue
Clear
Delete
DeleteAll

LineSensor_2
Properties
Start ([4950,00 0,00 250,00...)
End ([4950,00 500,00 250,...)
Radius (20,00 mm)
SensedPart ()
SensedPoint ([0,00 0,00 0,00] mm)
I/O Signals
Active
SensorOut

LogicGate_4 [NOT]
Properties
Operator (NOT)
Delay (0,0 s)
I/O Signals
InputA
InputB
Output

LogicGate [AND]
Properties
Operator (AND)
Delay (0,0 s)
I/O Signals
InputA
InputB
Output

LinearMover
Properties
Object (Queue)
Direction ([1,00 0,00 0,00] mm)
Speed (900,00 mm/s)
Reference (Global)
I/O Signals
Execute

fin ...