Tutorial 7 — SysML for Co-Simulation

Overview

This set of exercises will show you how to:

1. Create a simple CPS Design
2. Generate a Co-Simulation
3. Export a FMI Model Description
4. Import a FMI Model Description

Requirements

This tutorial requires the following tools from the INTO-CPS tool chain to be installed:

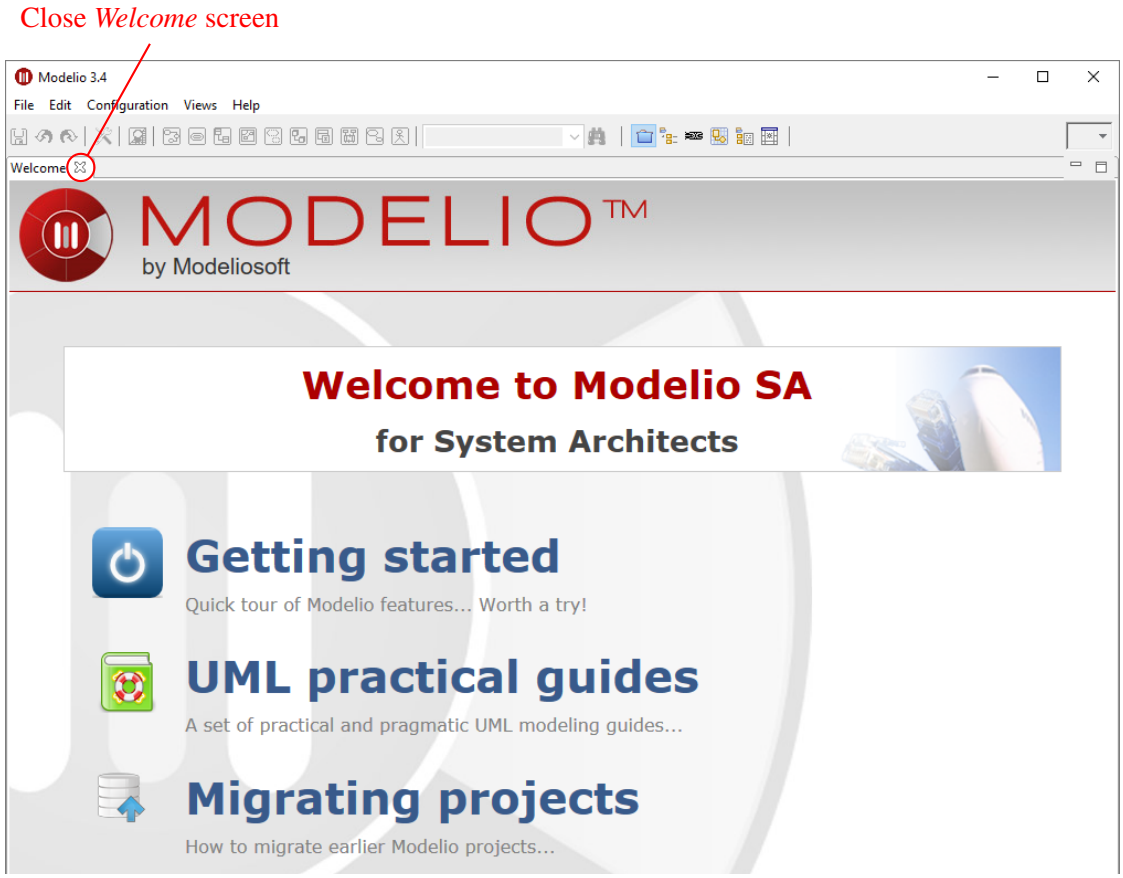
- Modelio v3.4.1

You may have been provided with tools on a USB drive at your training session. Otherwise follow Tutorial 0 with the guidelines to install the INTO-CPS Application. Tools can be downloaded from there through *Window > Show Download Manager* to your *into-cps-projects* install downloads directory. Please ask if you are unsure.

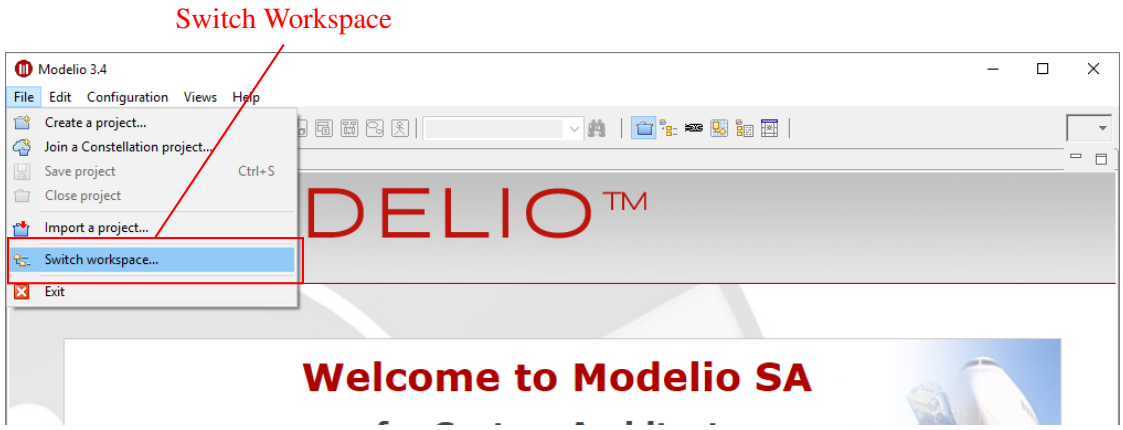
You may need to update the INTO-CPS extension for Modelio to utilise DSE components. The extension can be downloaded from `http://forge.modelio.org/projects/intocps-modelio34/files`. Version 1.2.05 or later is required for DSE modelling.

1 Opening a Project

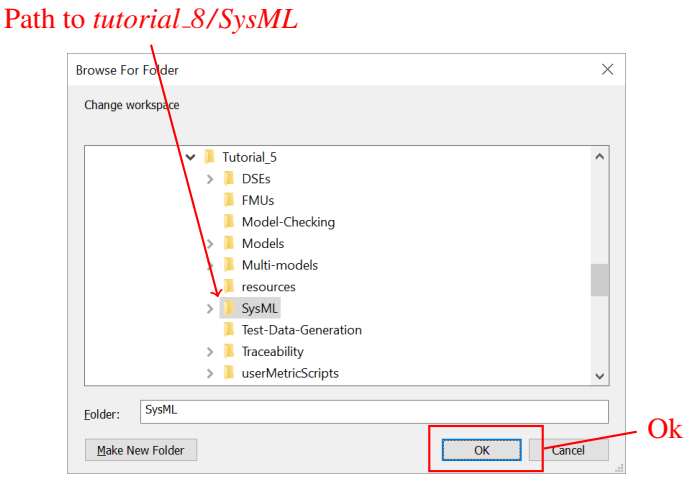
Step 1. Launch *Modelio*. On first loading, you may have to close the *Welcome* screen (you can bring it back with 'Help > Welcome' if you need)



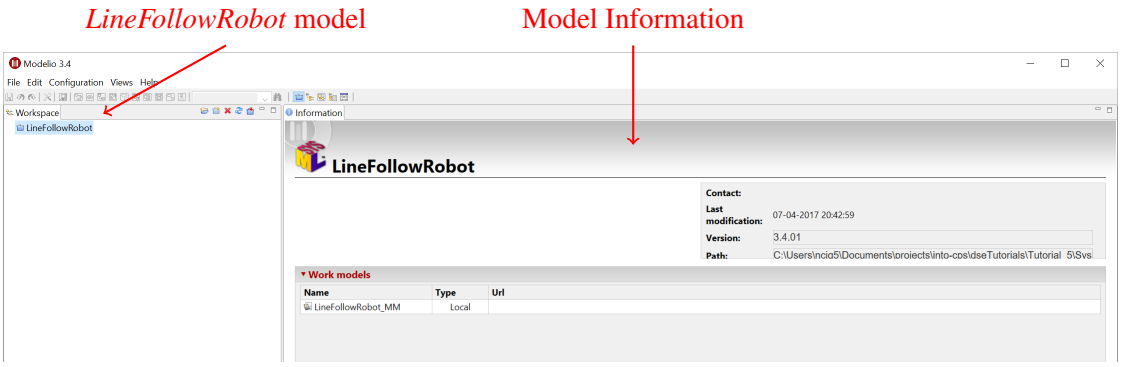
Step 2. A workspace must be chosen, select 'File > Switch Workspace'.



Step 3. Set the *Workspace* to the location of *tutorial_7/SysML* and click *Ok*.



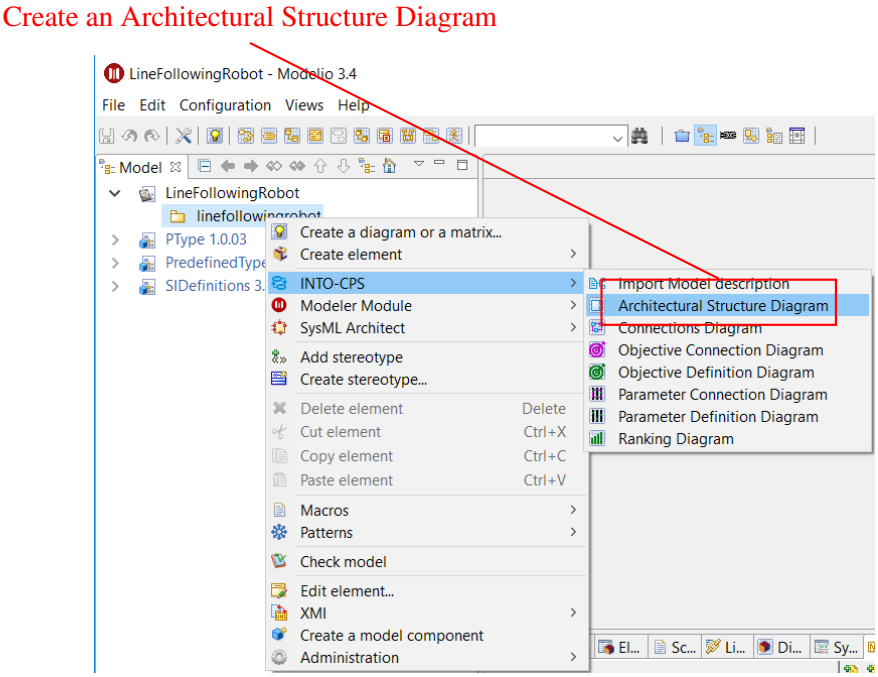
Step 4. Left-click on the *LineFollowRobot* model once on the left to see details of the model. Double-click the *LineFollowRobot* model to open the model.



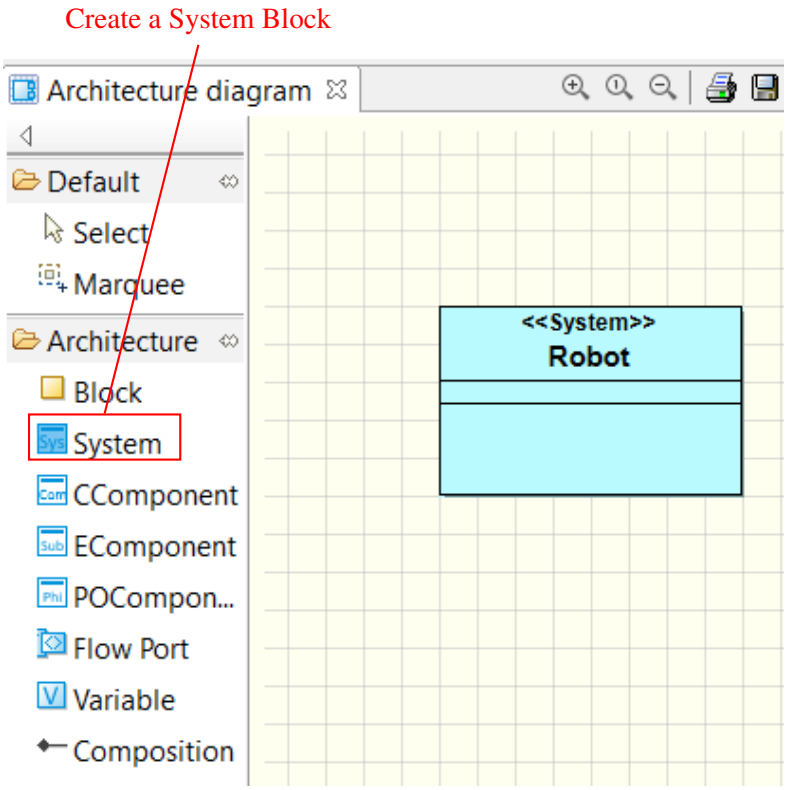
2 Define an Architecture

This section describes how to the design of the architecture of your system architecture. CPS Architecture is mainly composed of an unique System Block and a set of CComponent.

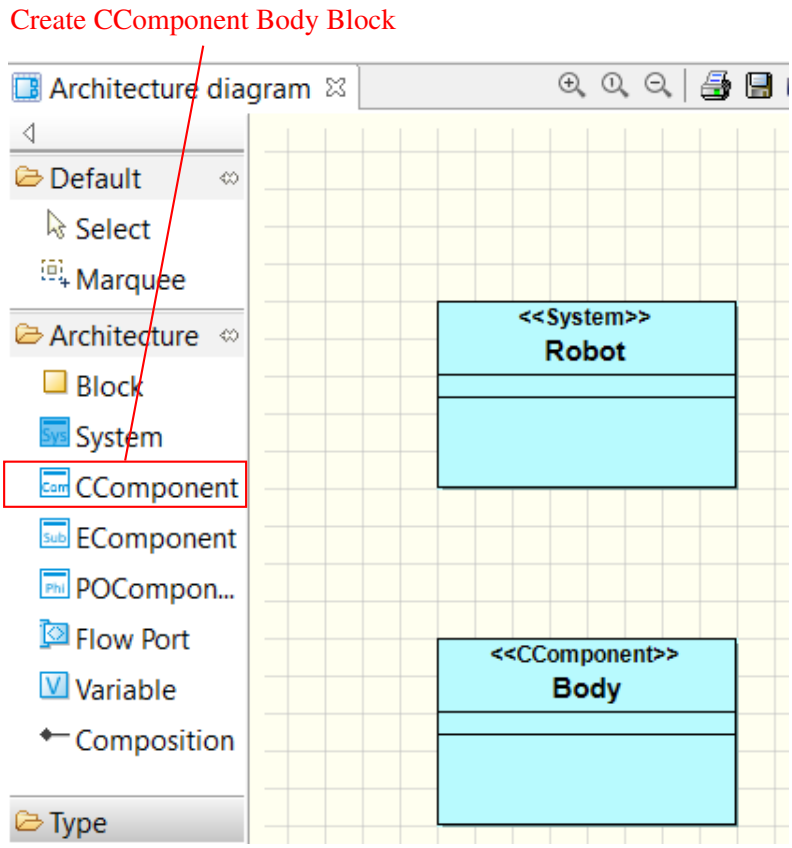
Step 5. Right click '$linefollowrobot\_mm$' in model outline. Select 'INTO-CPS > Architectural Structure Diagram'.



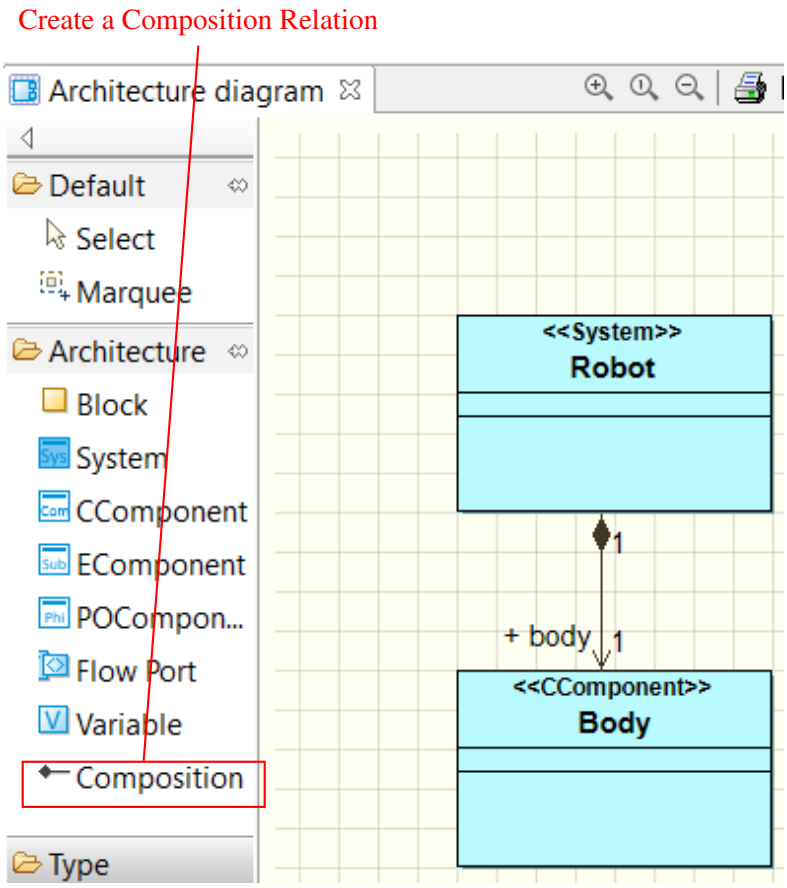
Step 6. From the *Toolbox*, select *System* and click on the empty diagram to create the System block. Double click on the created block and change its name to '*Robot*'.



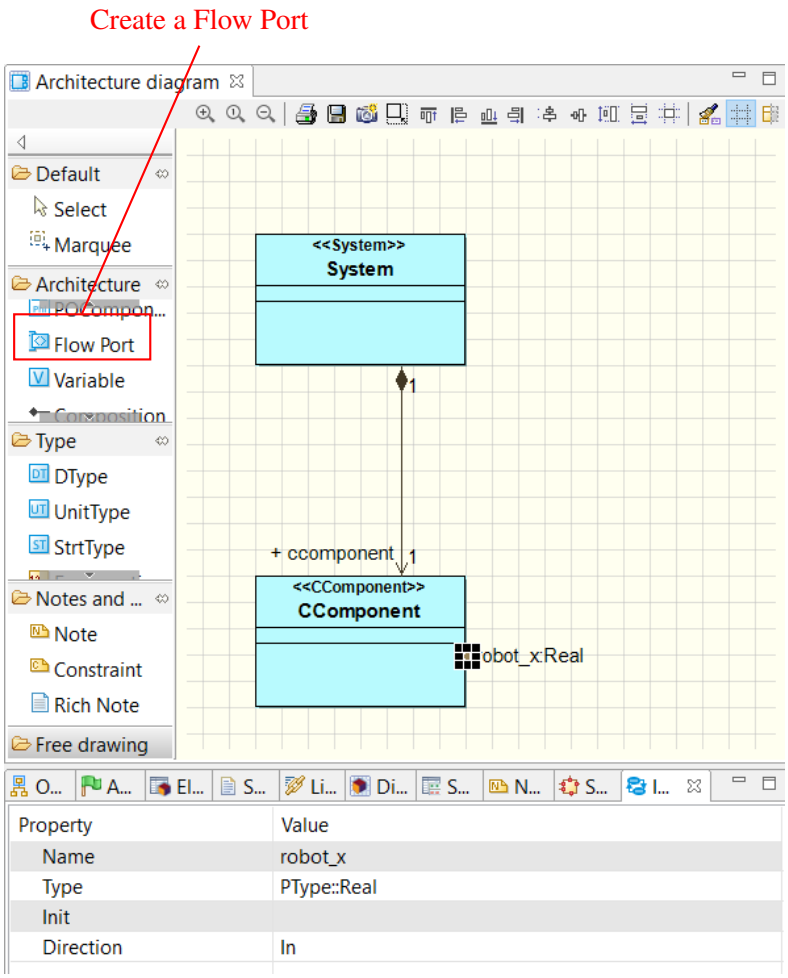
Step 7. From the *Toolbox*, select *CComponent* and add it to the diagram. Double click on the created block and change its name to '*Body*'.



Step 8. To associate the '*Body*' block with the Robot, select the *Composition* relation from the *Toolbox*, and click first on the '*Robot*' and then on the '*Body*' block.



Step 9. From the *Toolbox*, select *Flow Port* and add it to the '$Body$' CComponent. Select the block that this creates and F2 to change its name to '$robot\_x$'. On the *INTO-CPS* property view , change *Type* to '$PType :: Real$', and *Direction* to 'Out'.



Step 10. Repeat Step 9. to create the following flow ports to the '$Body$' CComponent:

- Name: '$robot\_y$', Type: '$PType :: Real$', Direction: 'Out';
- Name: '$robot\_z$', Type: '$PType :: Real$', Direction: 'Out';
- Name: '$robot\_theta$', Type: '$PType :: Real$', Direction: 'Out';
- Name: '$servo\_right\_input$', Type: '$PType :: Real$', Direction: 'In';
- Name: '$servo\_left\_input$', Type: '$PType :: Real$', Direction: 'In';

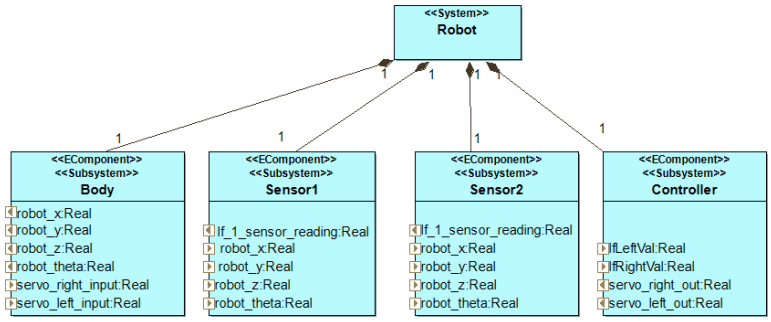
Step 11. Repeat from Step 7. to Step 9. to create the following ccomponent and their corresponding ports:

- Name: '$Sensor1$';
  - Name: '$robot\_x$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$robot\_y$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$robot\_z$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$robot\_theta$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$lf\_1\_sensor\_reading$', Type: '$PType :: Real$', Direction: 'Out';
- Name: '$Sensor2$';
  - Name: '$robot\_x$', Type: '$PType :: Real$', Direction: 'In';

  - Name: '$robot\_y$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$robot\_z$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$robot\_theta$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$lf\_1\_sensor\_reading$', Type: '$PType :: Real$', Direction: 'Out';

- Name: '$Controller$';
  - Name: '$ILeftVal$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$IRightVal$', Type: '$PType :: Real$', Direction: 'In';
  - Name: '$servo\_right\_out$', Type: '$PType :: Real$', Direction: 'Out';
  - Name: '$servo\_left\_out$', Type: '$PType :: Real$', Direction: 'Out';

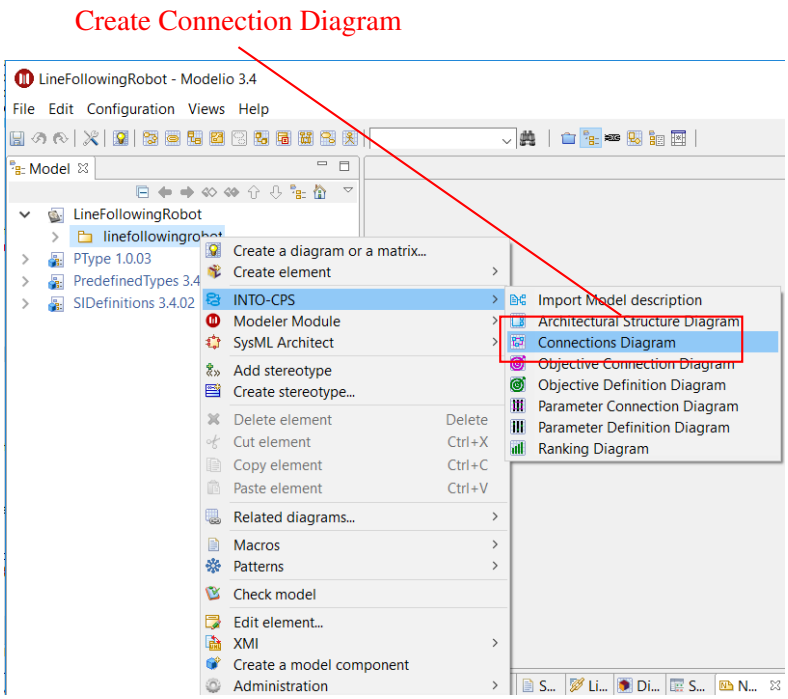
The model should now look like this:



3 Design a Connection Diagram

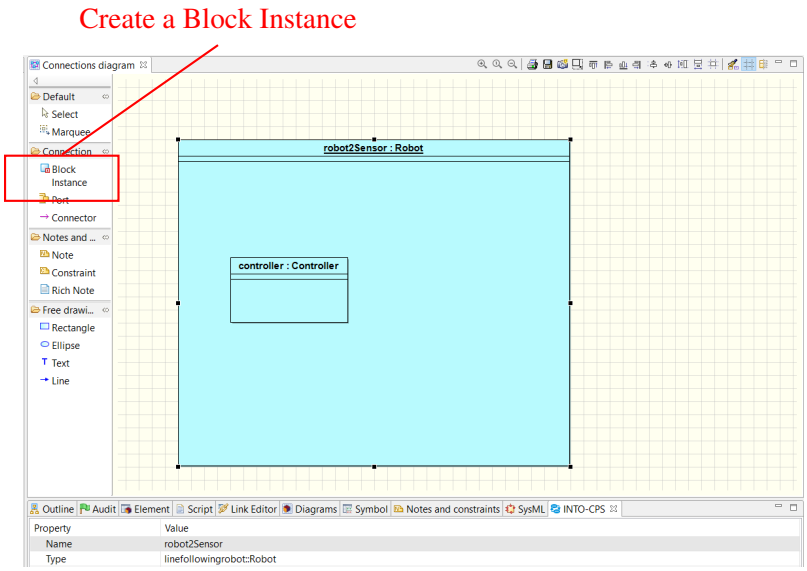
This section will outline the steps necessary to create a Connection Diagram under Modelio.

Step 12. Right click '$linefollowrobot\_mm$' in model outline. Select 'INTO-CPS > Connection Diagram'.

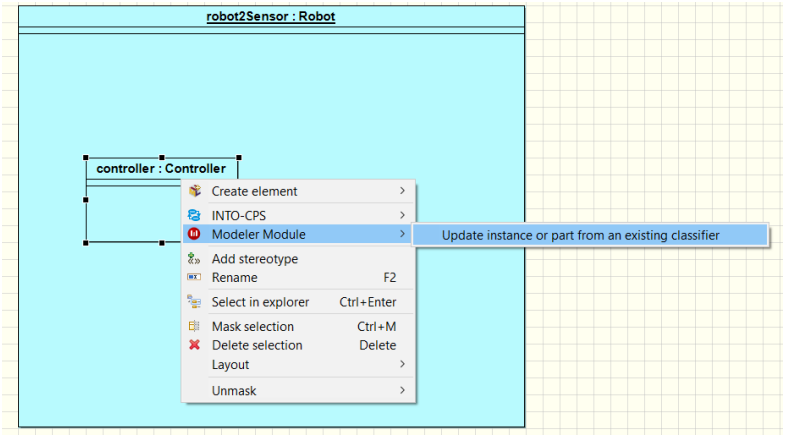


Step 13. Select the Block Instance automatically created with the Connection Diagram. On the *INTO-CPS* property view , change *Name* to '$robot2Sensor$', and its *Type* to '$Robot$'.

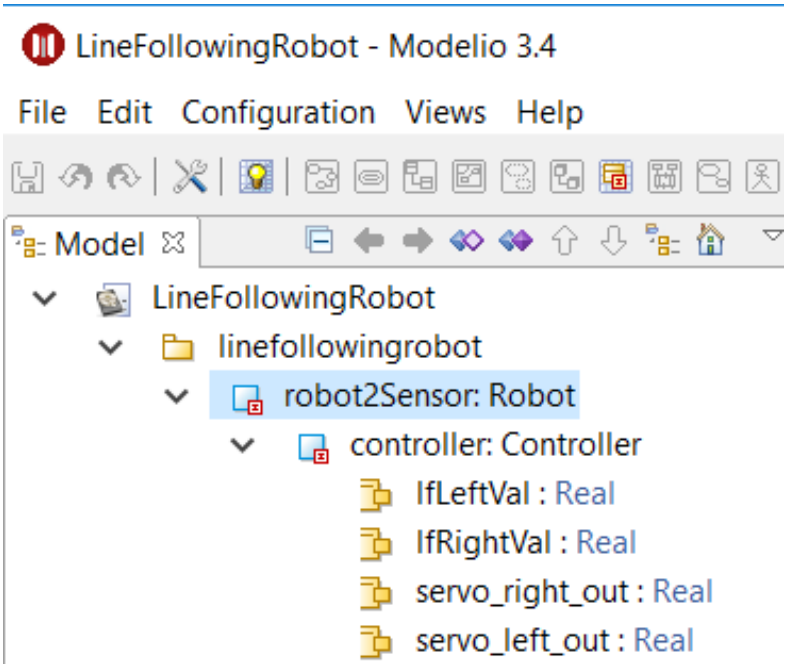
Step 14. From the *Toolbox*, select *BlockInstance* and click on the instance element available on the created diagram to create the a block instance. Select the Block Instance created and on the *INTO-CPS* property view , change its *Name* to '$controller$', and its *Type* to '$Controller$'



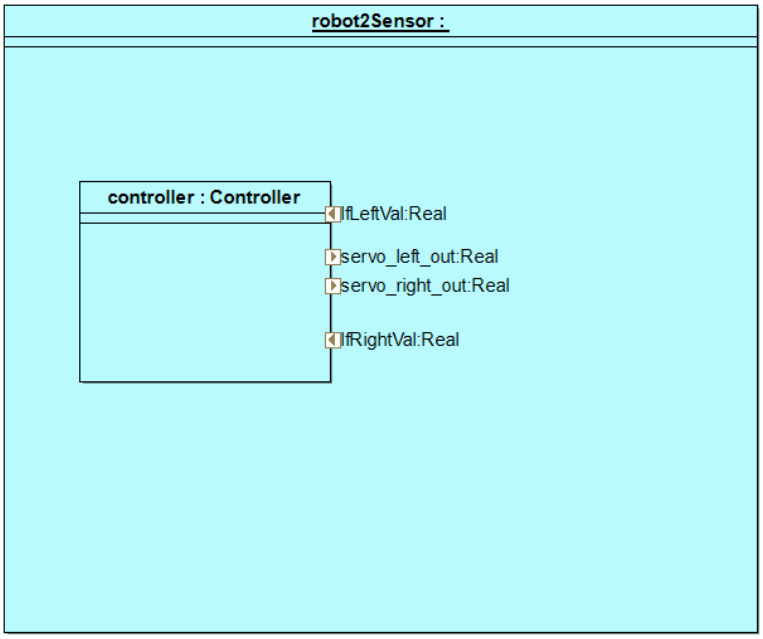
Step 15. Select '*controller*' block instance in the model outline and right click. Select 'Modeller Module > Update instance or part from an existing classifier'. Click *Update* then *OK*.



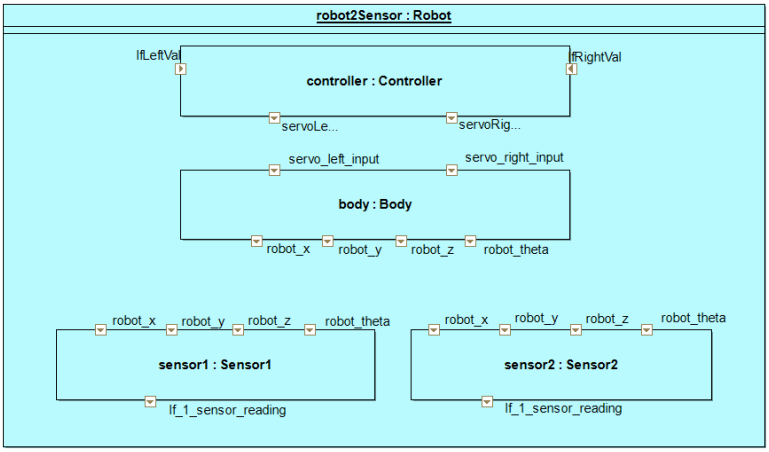
Step 16. Check the created ports inside the model explorer.



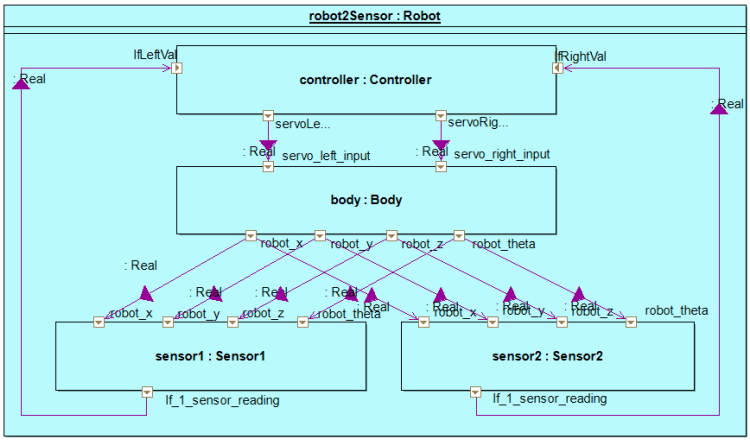
Step 17. Drag and Drop the ports from the model explore inside the diagram to unmask them.



Step 18. Repeat from Step 14. to Step 17. to create block instance of each CComponent.



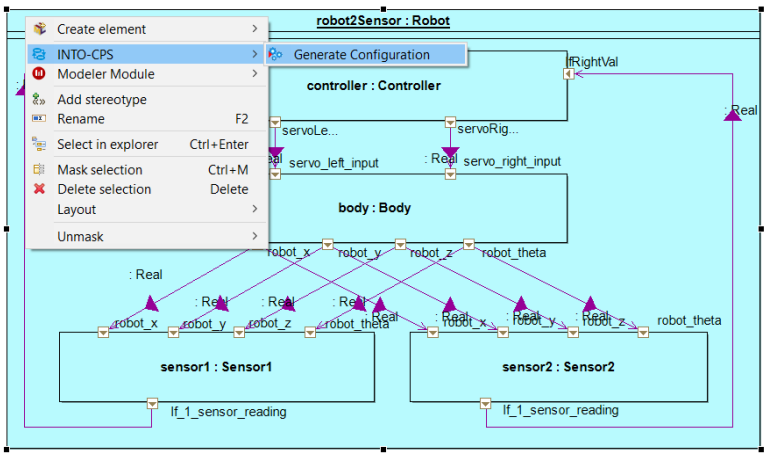
Step 19. From the *Toolbox*, select *Connector* and draw link from Output port to Input to design all connections.



4 Generating Co-Simulation Configuraton

This section will outline the steps necessary to generate a Co-simulation configuration in Modelio and include the file to the INTO-CPS app.

Step 20. Find any '$CComponent$' block in the model outline and right click. Select 'INTO-CPS > Generate Model Description'. Click *Export* then *OK*.

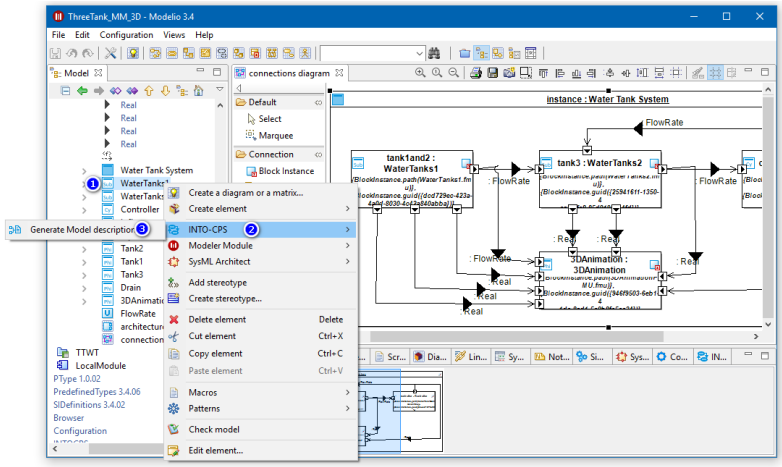


In the INTO-CPS app, this configuration file can be found in 'SysML > LineFollowerRobot > config'. Right click the configuration file and select 'Create Multi model'.

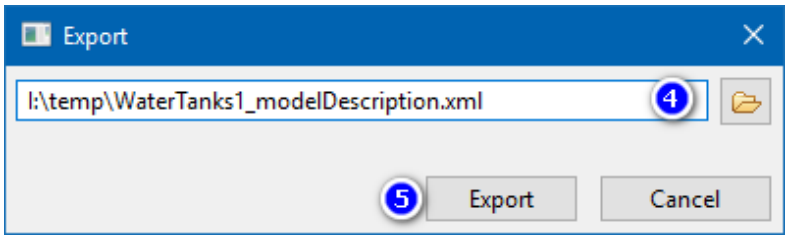
5 Exporting Model Description

This section will outline the steps necessary to export a CComponent as a FMI Model Description.

Step 21. Select any CComponent block like the '$Controller$' block in the model outline and right click. Select 'INTO-CPS > Generate Model Description'. Click *Export* then *OK*.



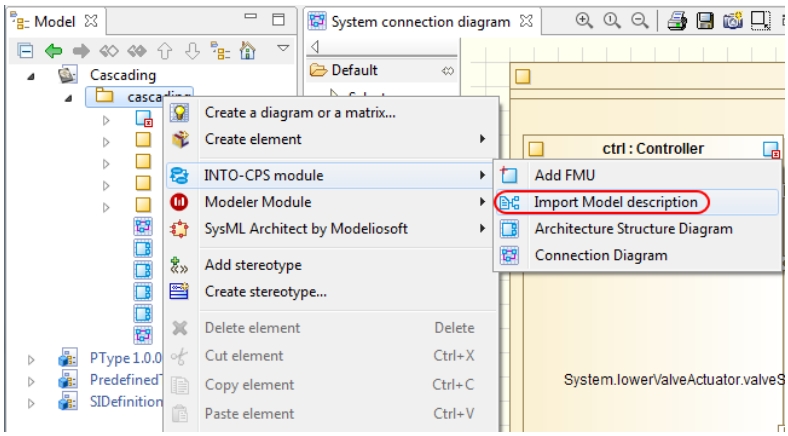
Step 22. Choose a file name destination.



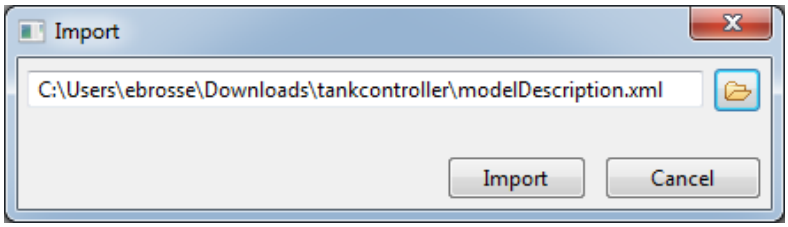
6 Importing Model Description

This section will outline the steps necessary to import an existing Model Description in Modelio.

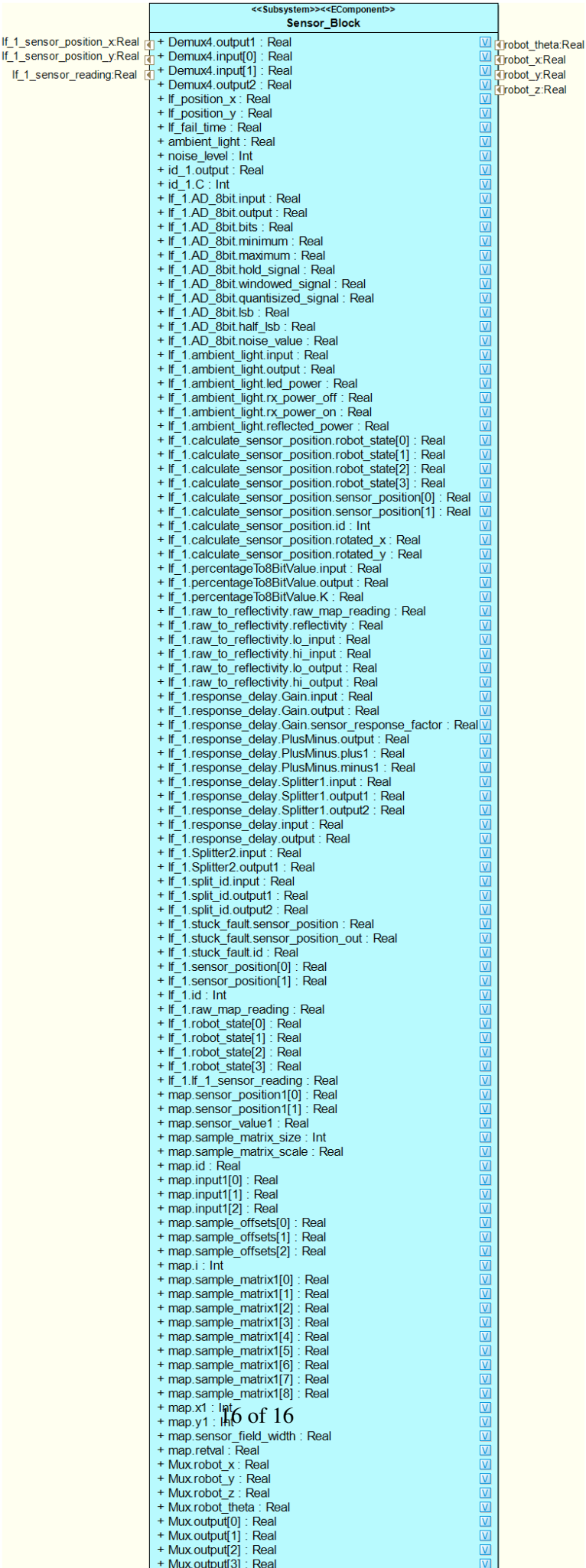
Step 23. Right click in the Modelio model browser on any *Package* element, then select 'INTO-CPS > Import Model Description'.



Step 24. Select the desired target file in your computer and click on *Import*.



This import command creates an Architecture Structure Diagram describing the interface of an INTO-CPS *CComponent* corresponding to the `modelDescription.xml` file imported.


<<Subsystem>><<EComponent>>
Sensor_Block
lf_1_sensor_position_x:Real
lf_1_sensor_position_y:Real
lf_1_sensor_reading:Real
+ Demux4.output1 : Real
+ Demux4.input[0] : Real
+ Demux4.input[1] : Real
+ Demux4.output2 : Real
+ lf_position_x : Real
+ lf_position_y : Real
+ lf_fail_time : Real
+ ambient_light : Real
+ noise_level : Int
+ id_1.output : Real
+ id_1.C : Int
+ lf_1.AD_8bit.input : Real
+ lf_1.AD_8bit.output : Real
+ lf_1.AD_8bit.bits : Real
+ lf_1.AD_8bit.minimum : Real
+ lf_1.AD_8bit.maximum : Real
+ lf_1.AD_8bit.hold_signal : Real
+ lf_1.AD_8bit.windowed_signal : Real
+ lf_1.AD_8bit.quantisized_signal : Real
+ lf_1.AD_8bit.lsb : Real
+ lf_1.AD_8bit.half_lsb : Real
+ lf_1.AD_8bit.noise_value : Real
+ lf_1.ambient_light.input : Real
+ lf_1.ambient_light.output : Real
+ lf_1.ambient_light.led_power : Real
+ lf_1.ambient_light.rx_power_off : Real
+ lf_1.ambient_light.rx_power_on : Real
+ lf_1.ambient_light.reflected_power : Real
+ lf_1.calculate_sensor_position.robot_state[0] : Real
+ lf_1.calculate_sensor_position.robot_state[1] : Real
+ lf_1.calculate_sensor_position.robot_state[2] : Real
+ lf_1.calculate_sensor_position.robot_state[3] : Real
+ lf_1.calculate_sensor_position.sensor_position[0] : Real
+ lf_1.calculate_sensor_position.sensor_position[1] : Real
+ lf_1.calculate_sensor_position.id : Int
+ lf_1.calculate_sensor_position.rotated_x : Real
+ lf_1.calculate_sensor_position.rotated_y : Real
+ lf_1.percentageTo8BitValue.input : Real
+ lf_1.percentageTo8BitValue.output : Real
+ lf_1.percentageTo8BitValue.K : Real
+ lf_1.raw_to_reflectivity.raw_map_reading : Real
+ lf_1.raw_to_reflectivity.reflectivity : Real
+ lf_1.raw_to_reflectivity.lo_input : Real
+ lf_1.raw_to_reflectivity.hi_input : Real
+ lf_1.raw_to_reflectivity.lo_output : Real
+ lf_1.raw_to_reflectivity.hi_output : Real
+ lf_1.response_delay.Gain.input : Real
+ lf_1.response_delay.Gain.output : Real
+ lf_1.response_delay.Gain.sensor_response_factor : Real
+ lf_1.response_delay.PlusMinus.output : Real
+ lf_1.response_delay.PlusMinus.plus1 : Real
+ lf_1.response_delay.PlusMinus.minus1 : Real
+ lf_1.response_delay.Splitter1.input : Real
+ lf_1.response_delay.Splitter1.output1 : Real
+ lf_1.response_delay.Splitter1.output2 : Real
+ lf_1.response_delay.input : Real
+ lf_1.response_delay.output : Real
+ lf_1.Splitter2.input : Real
+ lf_1.Splitter2.output1 : Real
+ lf_1.split_id.input : Real
+ lf_1.split_id.output1 : Real
+ lf_1.split_id.output2 : Real
+ lf_1.stuck_fault.sensor_position : Real
+ lf_1.stuck_fault.sensor_position_out : Real
+ lf_1.stuck_fault.id : Real
+ lf_1.sensor_position[0] : Real
+ lf_1.sensor_position[1] : Real
+ lf_1.id : Int
+ lf_1.raw_map_reading : Real
+ lf_1.robot_state[0] : Real
+ lf_1.robot_state[1] : Real
+ lf_1.robot_state[2] : Real
+ lf_1.robot_state[3] : Real
+ lf_1.lf_1_sensor_reading : Real
+ map.sensor_position1[0] : Real
+ map.sensor_position1[1] : Real
+ map.sensor_value1 : Real
+ map.sample_matrix_size : Int
+ map.sample_matrix_scale : Real
+ map.id : Real
+ map.input1[0] : Real
+ map.input1[1] : Real
+ map.input1[2] : Real
+ map.sample_offsets[0] : Real
+ map.sample_offsets[1] : Real
+ map.sample_offsets[2] : Real
+ map.i : Int
+ map.sample_matrix1[0] : Real
+ map.sample_matrix1[1] : Real
+ map.sample_matrix1[2] : Real
+ map.sample_matrix1[3] : Real
+ map.sample_matrix1[4] : Real
+ map.sample_matrix1[5] : Real
+ map.sample_matrix1[6] : Real
+ map.sample_matrix1[7] : Real
+ map.sample_matrix1[8] : Real
+ map.x1 : Int
+ map.y1 : Int
+ map.sensor_field_width : Real
+ map.retval : Real
+ Mux.robot_x : Real
+ Mux.robot_y : Real
+ Mux.robot_z : Real
+ Mux.robot_theta : Real
+ Mux.output[0] : Real
+ Mux.output[1] : Real
+ Mux.output[2] : Real
+ Mux.output[3] : Real
robot_theta:Real
robot_x:Real
robot_y:Real
robot_z:Real