Tutorial 4 — FMU Export (Overture)

Overview

This third INTO-CPS tutorial will show you how to:

1. Generate a new controller FMU in Overture
   (a) Import a model description into Overture
   (b) Complete the skeleton model to produce a working controller
   (c) Export the controller FMU
2. Associate the new controller FMU with a multi-model configuration
3. Execute a co-simulation using the new controller

Requirements

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

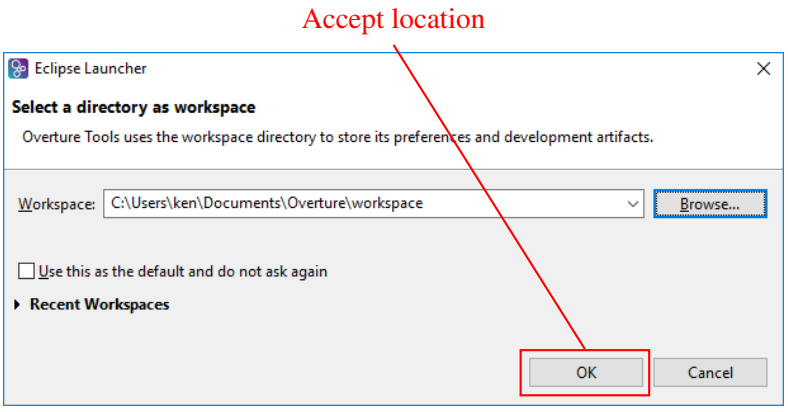
- INTO-CPS Application
- COE (Co-simulation Orchestration Engine) accessible to the Application
- Overture and the FMU plug-in extension

You may have been provided with tools on a USB drive at your training session. Otherwise the INTO-CPS Application can be downloaded from `https://into-cps.github.io/download/` and 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.

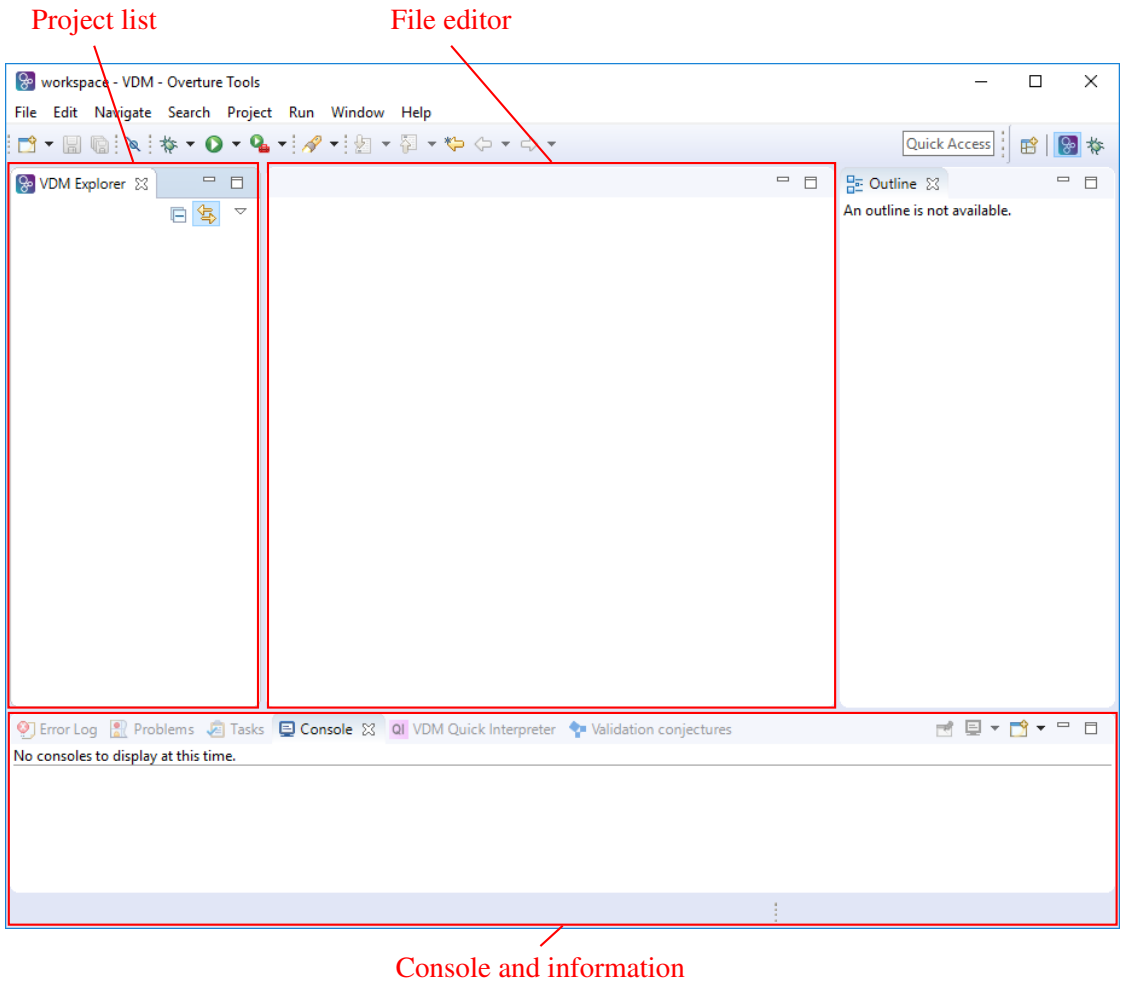
1 Creating a Project in Overture

The example in this tutorial is a small line-following robot with two infrared sensors. We will generate a controller FMU that reads these sensors and controls the wheels to follow the line. First we will create a project.

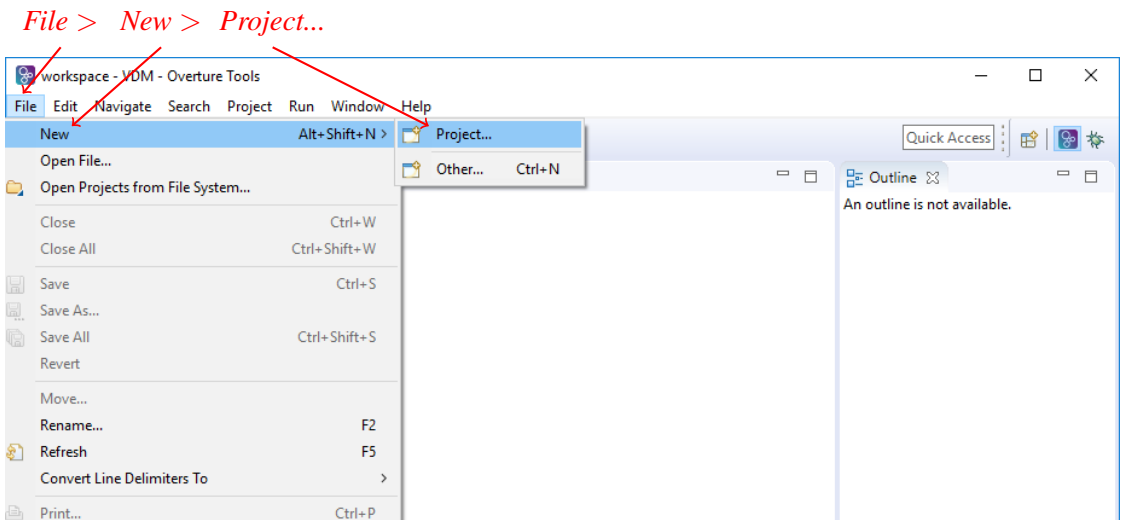
Step 1. Open *Overture*. It will prompt you to select a location for its workspace. You may accept the default location by pressing *OK*, or press *Browse...* to select a different location. If you do not want to be prompted in future, check *Use this as the default and do not ask again*.



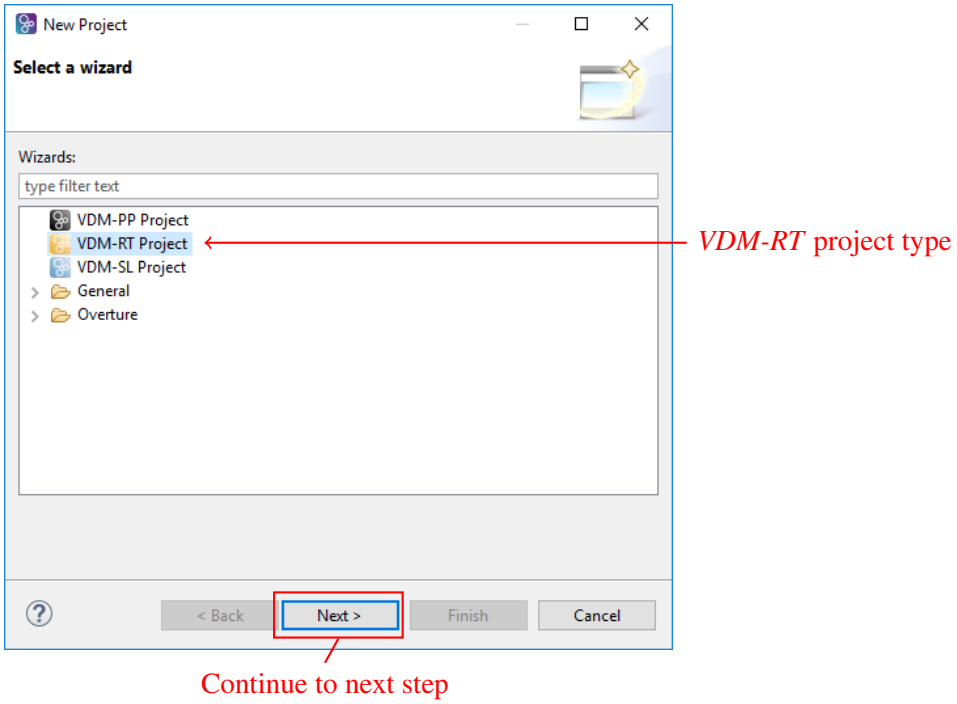
This is the *Overture* window, which includes a project list, file editor and a console.



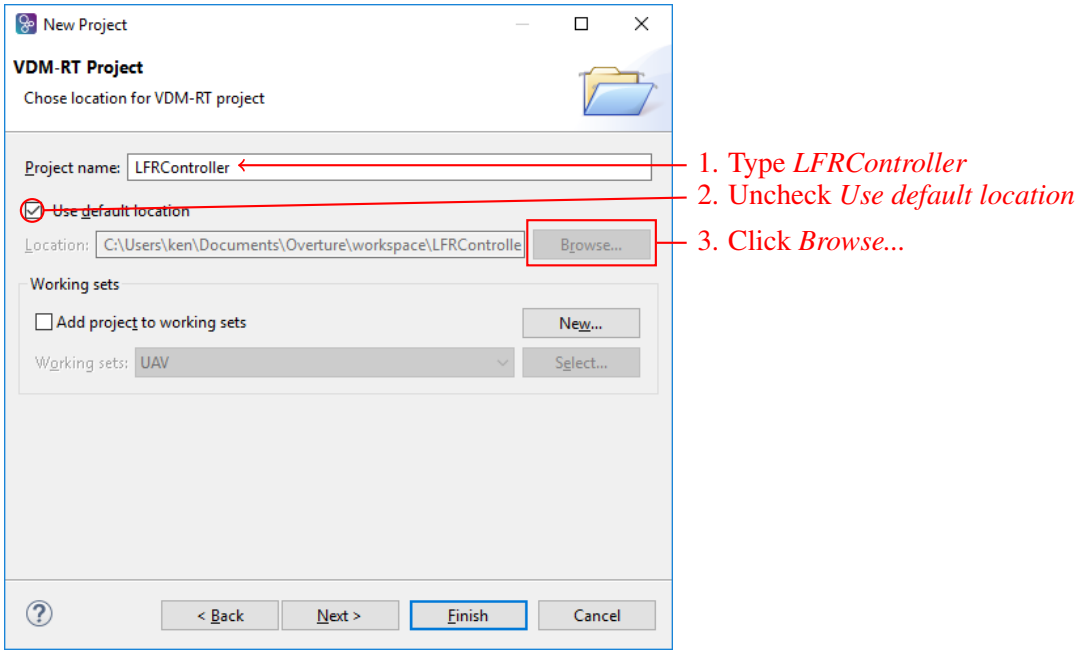
Step 2. First create a project that will hold the controller model. Select *File* > *New* > *Project...*.



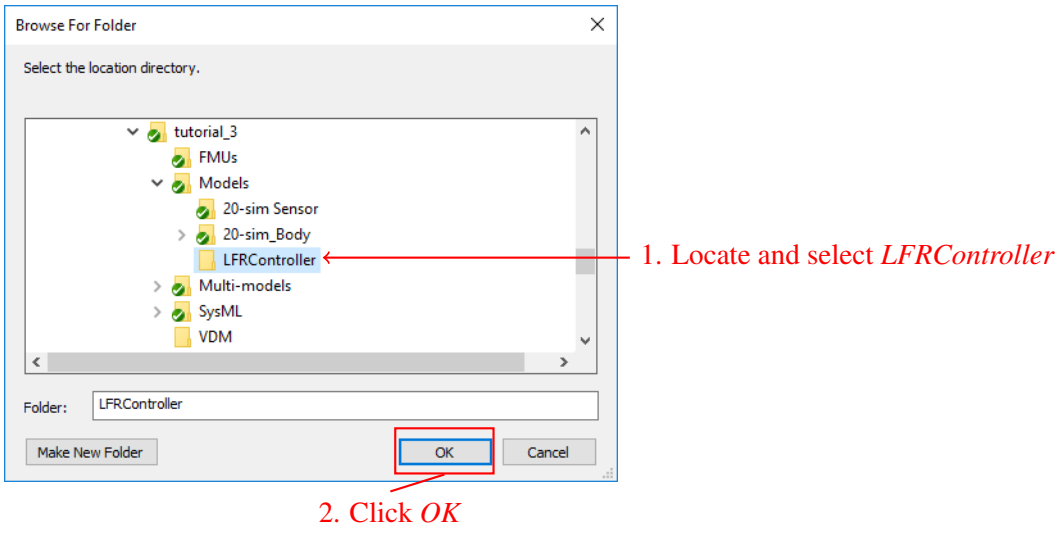
Step 3. In the *New Project* window, select *VDM-RT Project* and click *Next >* to go to the next step.



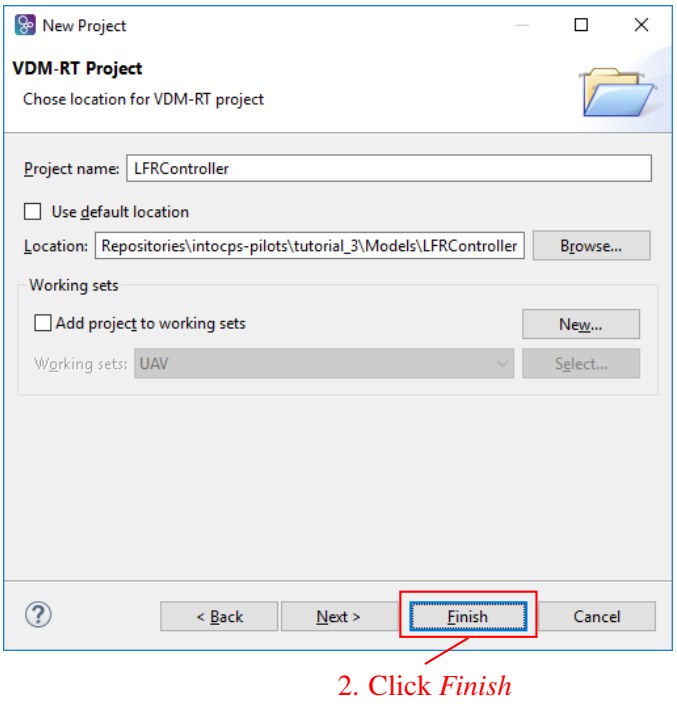
Step 4. The next screen asks for a name for the project. Call it *LFRController*. We will place the project in the *tutorial_3/Models* folder, so uncheck *Use default location* and click *Browse...*



Step 5. Locate and select the folder *tutorial_3/Models/LFRController* and click *OK*.



Step 6. Finally click *Finish* to create the project.



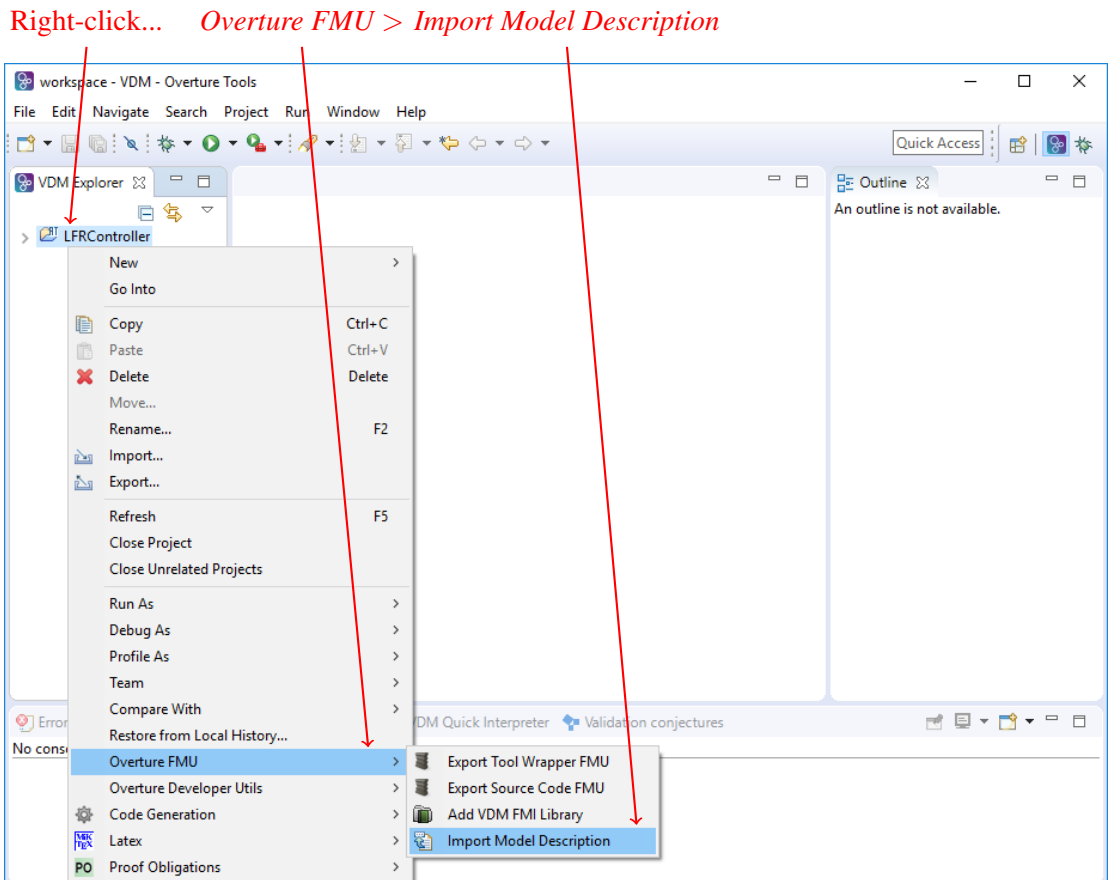
You should see the new project in the project list.



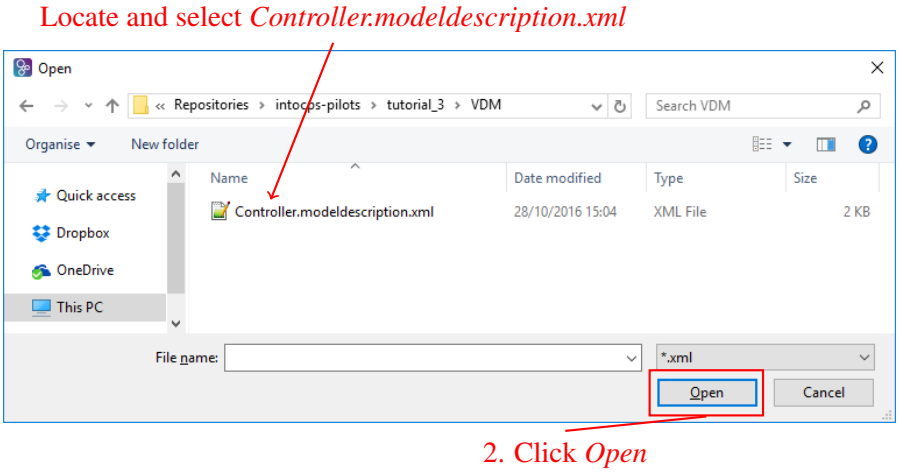
2 Importing a Model Description into Overture

Overture can import model description files to create a skeleton project with the correct input, output and parameter ports, as well as standard boilerplate elements needed in a VDM-RT model.

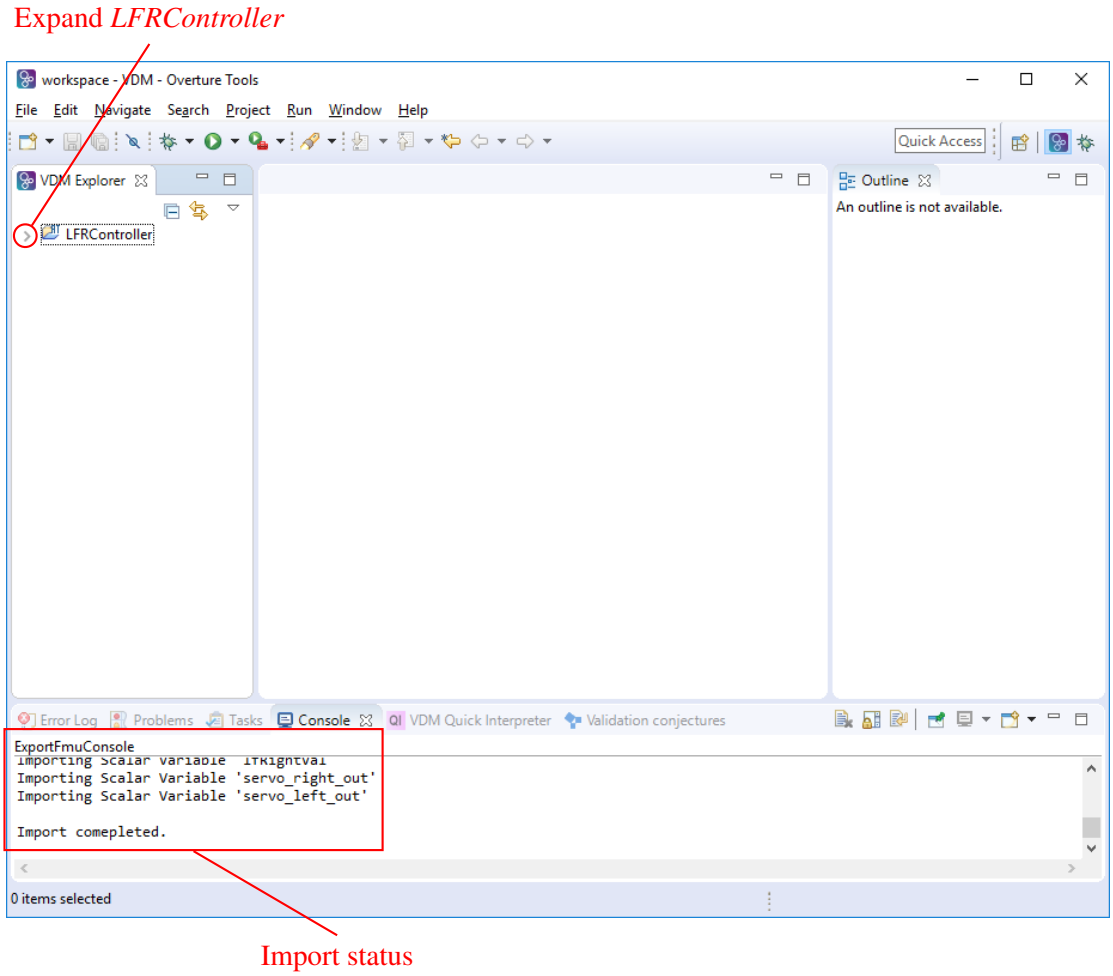
Step 7. To import a model description, right-click on the *LFRController* project and select *Overture FMU > Import Model Description*.



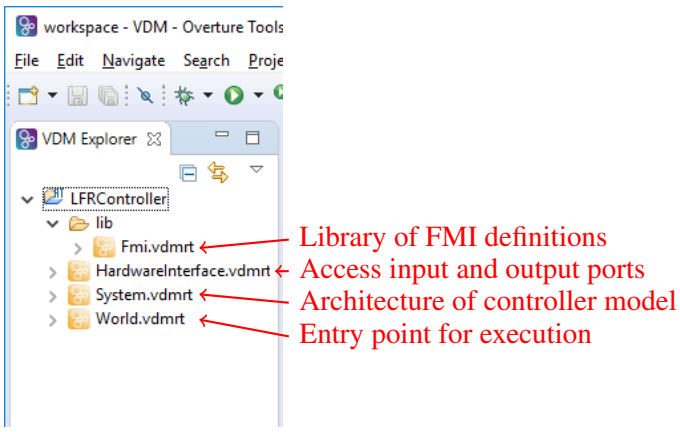
Step 8. Locate the file *tutorial_3/VDM/Controller.modeldescription.xml* that is included in the project and click *Open*.



Step 9. Overture will parse the file and populated the project. You can see status messages from the import in the *Console*. Expand the *LFRController* project to see what was imported.



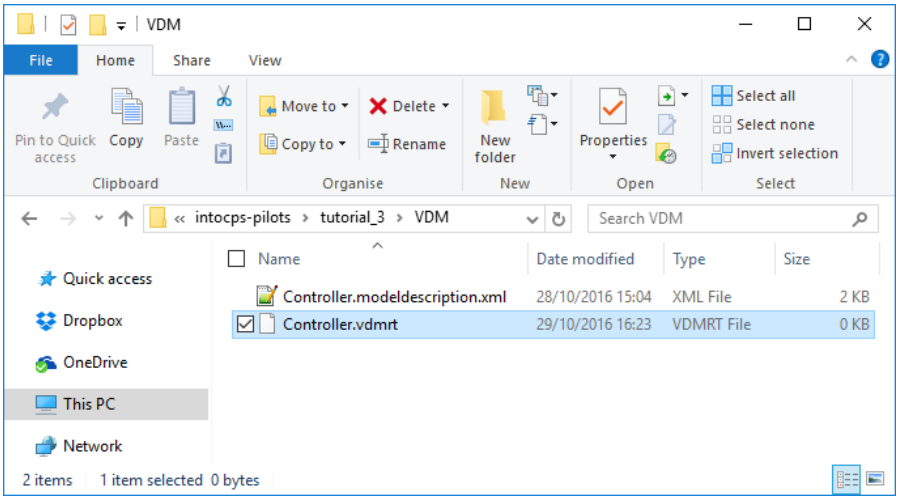
You should see the following structure:



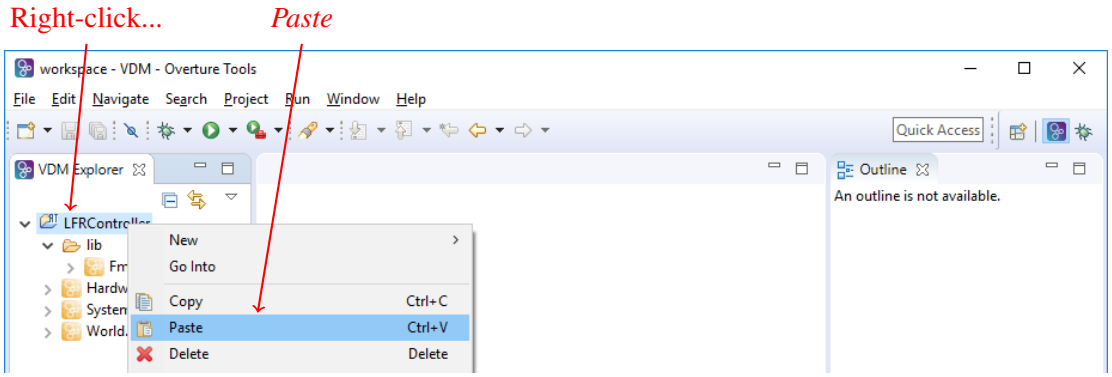
3 Adding a Controller Class

To make a functional controller, we will add a *Controller* class and instantiate it as an object in the *System* class, and set the *World* to start the controller thread. A basic controller class is included in the *tutorial_3* project.

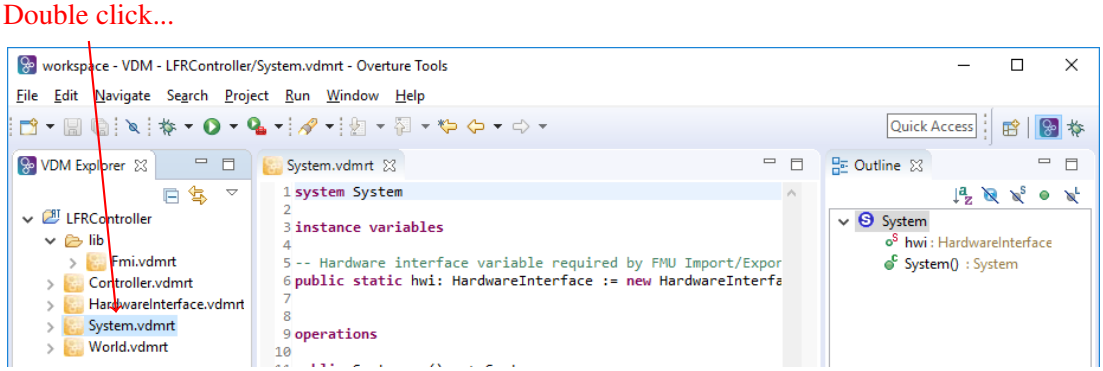
Step 10. Locate the file *tutorial_3/VDM/Controller.vdmrt* on on your file system and copy it.



Step 11. Right-click on right-click on the *LFRController* project and select *Paste*.



Step 12. Double-click *System.vdmrt* to open the `System` class.



Step 13. Add the highlighted lines to *System.vdmrt*. This will define a `controller` object of the `Controller` class and instantiate it.

```
system System

instance variables

-- Hardware interface variable required by FMU Import/Export
public static hwi: HardwareInterface := new HardwareInterface();

public static controller: Controller := new Controller(
  hwi.servoLeftVal, hwi.servoRightVal, hwi.lfRightVal, hwi.lfLeftVal);

cpu : CPU := new CPU(<FP>, 1E6);

operations

public System : () ==> System
System () ==
(
  cpu.deploy(controller);
);

end System
```

Step 14. Double-click *World.vdmrt* to open the `World` class. Uncomment the highlighted line to tell the controller thread to start at the beginning of co-simulation.

```
class World

operations

public run : () ==> ()
run() ==
 (
  start(System`controller);
  block();
 );

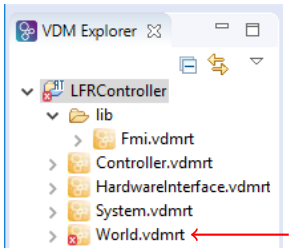
private block : () ==>()
block() ==
  skip;

sync

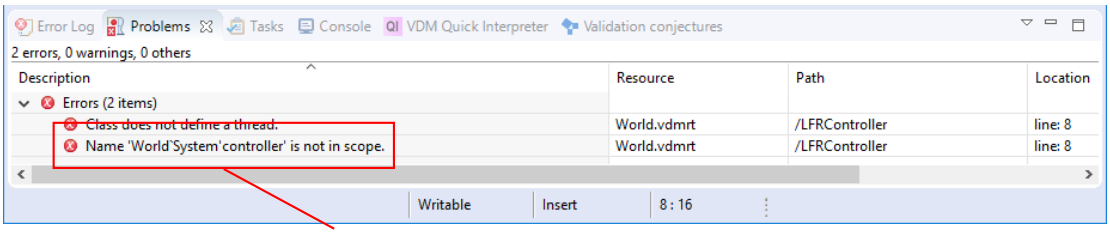
  per block => false;

end World
```

Step 15. Ensure that your model has no errors. If it does, a red cross will appear next to the file icon in the project browser. (You might have to refresh the project by right-clicking and selecting *Refresh* to see these.)



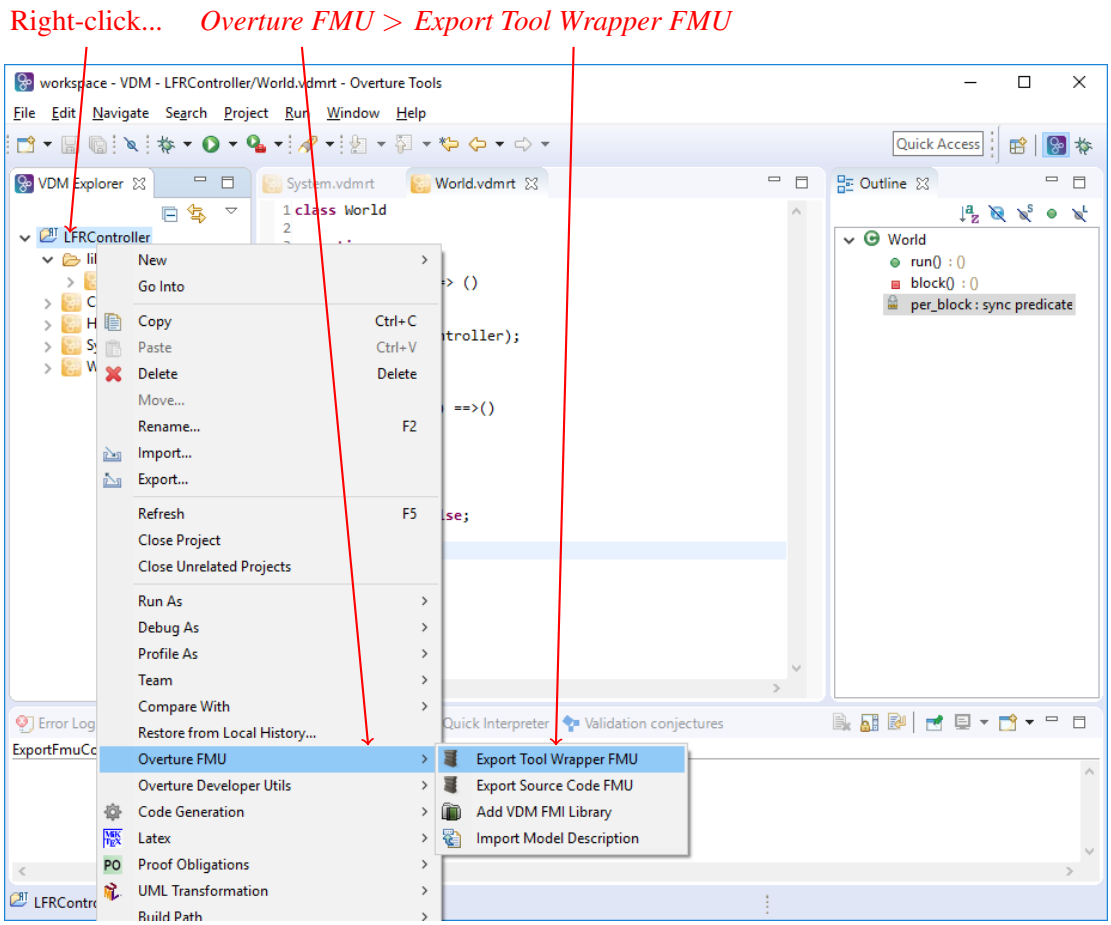
Check that you have correctly replicated the listings from Steps 13 and 14. Look at the *Problems* tab at the bottom for information, and double-click items to take you to the problem in the file editor.



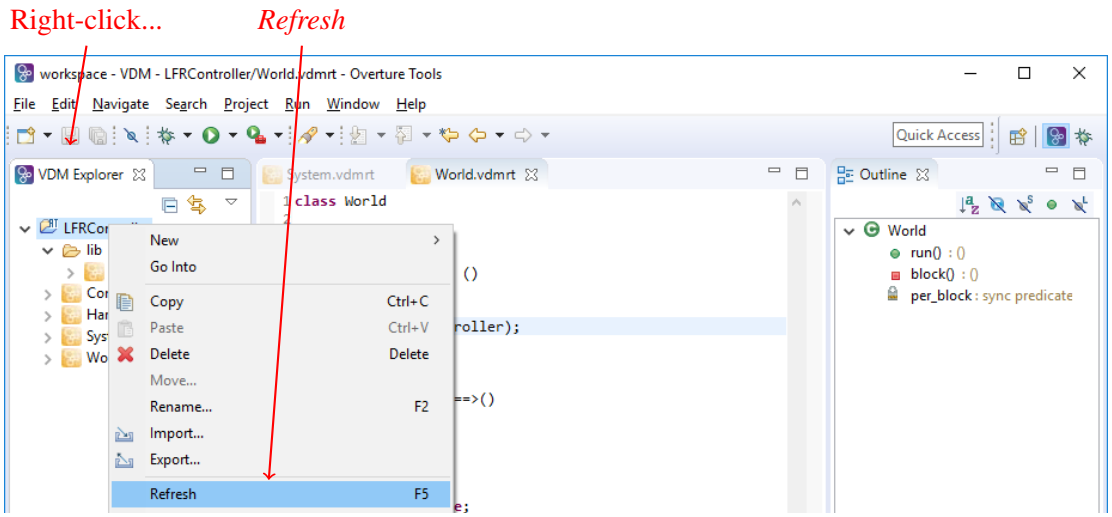
4 Exporting an FMU and Adding it to a Multi-model

Now that the controller model is complete, we can export an FMU and place it in the *tutorial_3* where the INTO-CPS Application can see it.

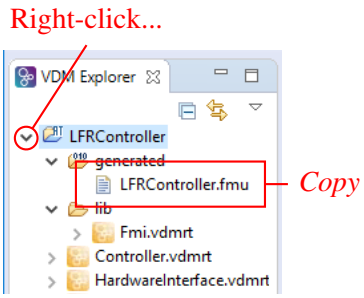
Step 16. To export an FMU, right-click on the *LFRController* project and select *Overture FMU > Export Tool Wrapper FMU*.



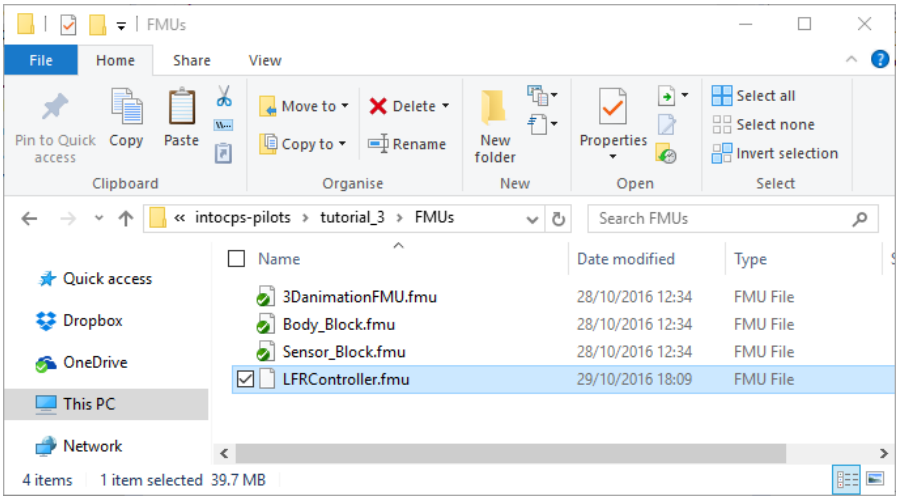
Step 17. Refresh the project so that the generated FMU appears. To do this, right-click on the project and select *Refresh*.



Step 18. A new folder called *generated* will appear. Expand this to see *LFRController.fmu*. Select *LFRController.fmu* and copy it using *Ctrl+C* or right-clicking and selecting *Copy*.

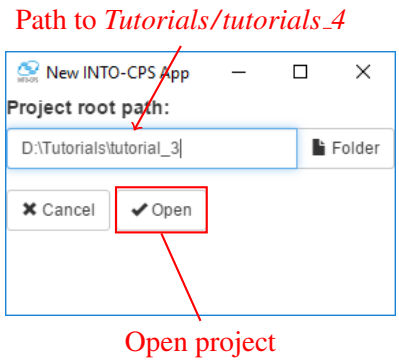


Step 19. Paste *LFRController.fmu* into the *tutorial_3/FMUs* folder on your file system.

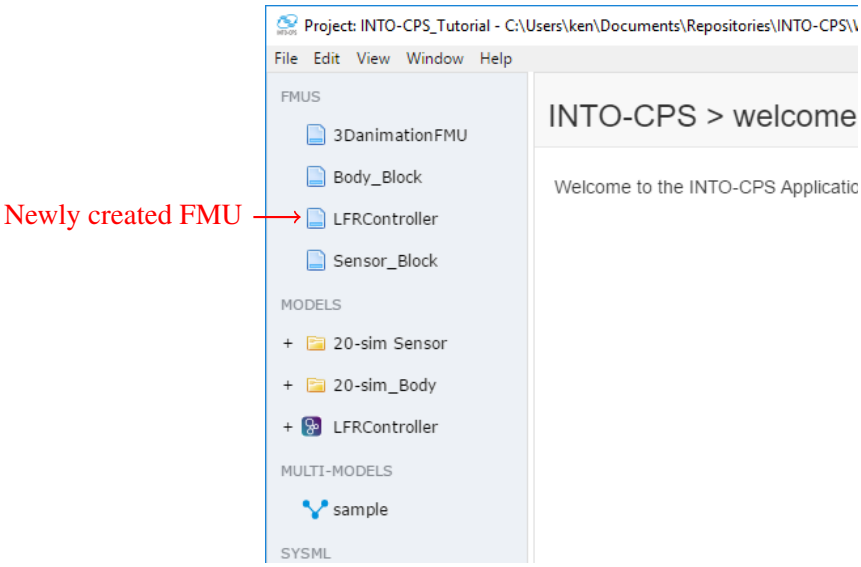


5 Co-simulating with the New Controller

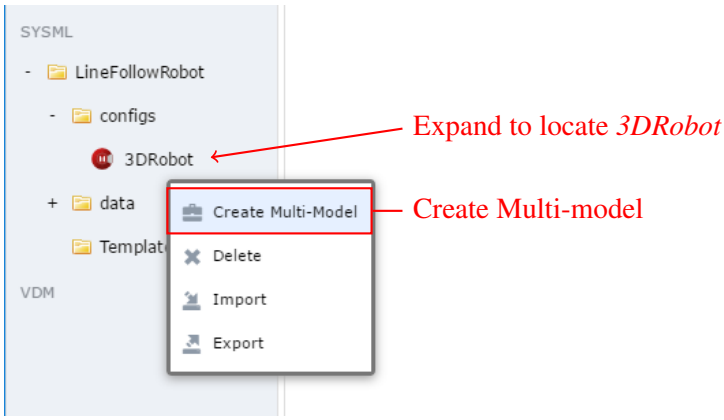
Step 20. Launch the *INTO-CPS Application* and select *File* > *Open Project*. Set the *Project root path* to the location of *Tutorials/tutorials_4* and click *Open*. You can browse using the *Folder* button.



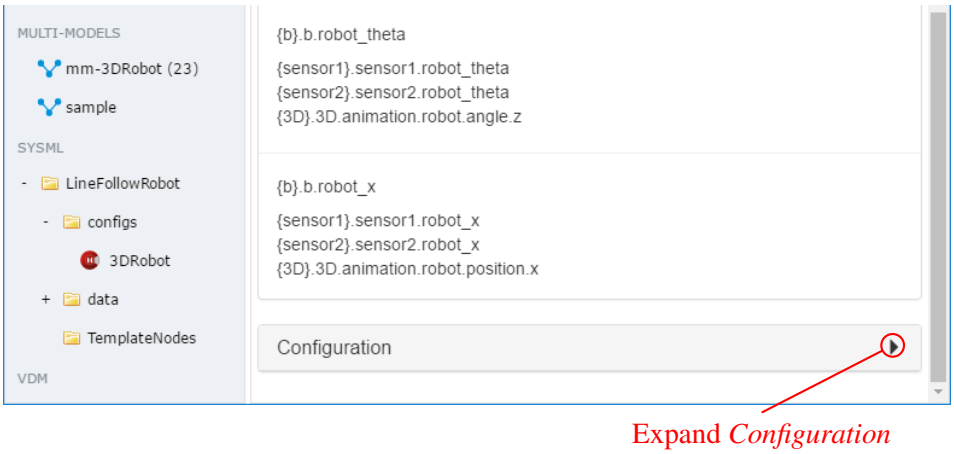
You should see the newly export *LFRController* FMU in the list.



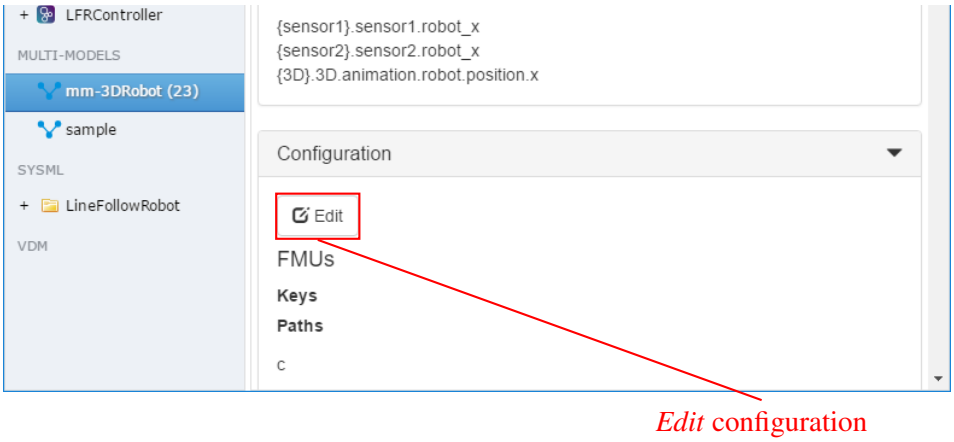
Step 21. In the SysML entry of the project browser, expand the *LineFollowRobot* folder, then *config* folders. Right-click on *3DRobot* and select *Create Multi-Model*.



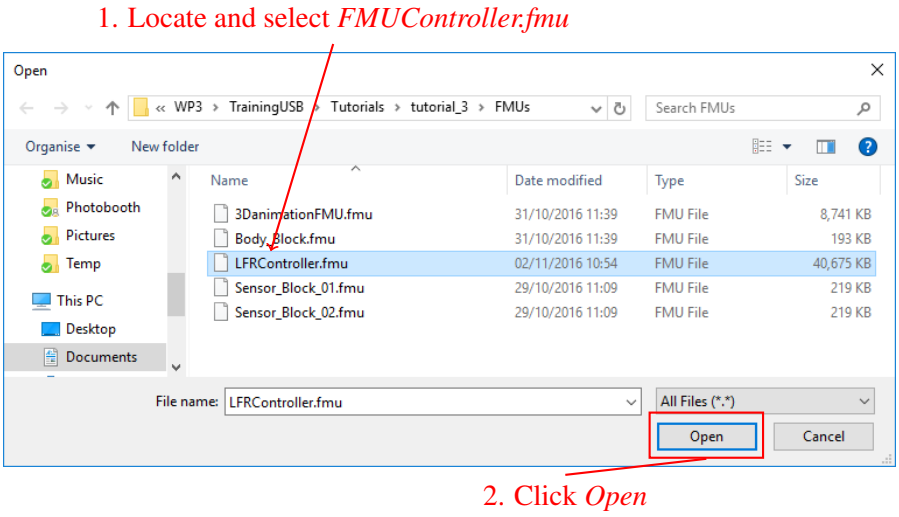
Step 22. We now need to associate FMUs to the multi-model as we did in *Tutorial 2*. Scroll down to find the *Configuration* panel and expand it by clicking the arrow.



Step 23. Scroll down and click *Edit*.



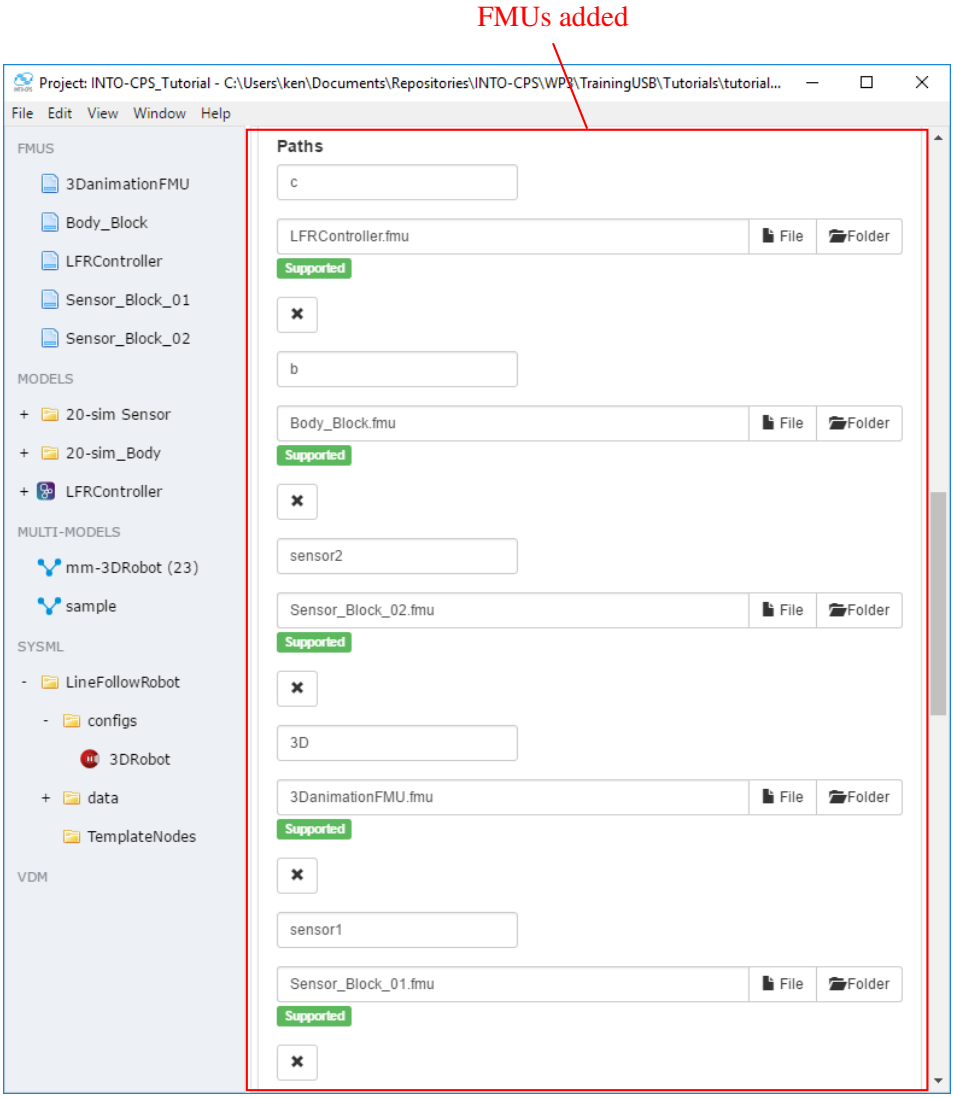
Step 24. As in *Tutorial 2*, in the FMUs section press *File* next to the Controller element, *c*. A file browser window will open and show five FMUs (if the file browser does not show the FMUs, navigate to *tutorials_4/FMUs*). Select *FMUController.fmu* and click *Open*.



Step 25. Repeat this for the remaining elements:

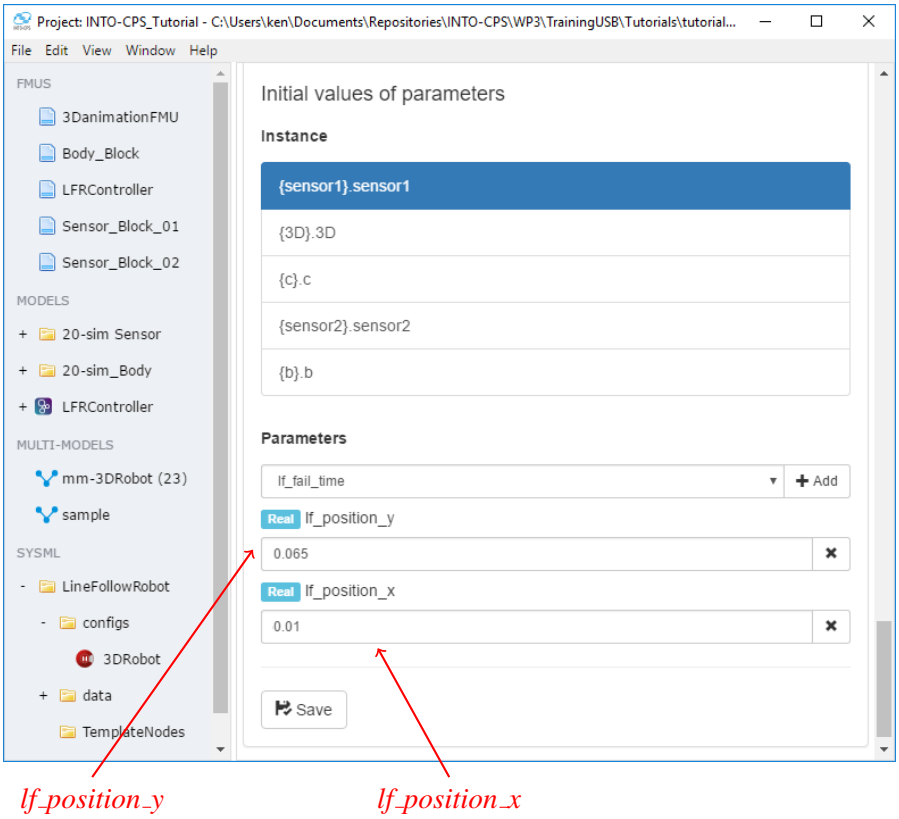
- *b* : *Body_Block.fmu*
- *3D* : *3DanimationFMU.fmu*
- *sensor1* : *Sensor_Block_01.fmu*
- *sensor2* : *Sensor_Block_02.fmu*

The complete set of FMUs will look like this:



Step 26. Scroll down to the *Initial values of parameters section*, and click {*sensor1*}.*sensor1*. In the *Parameters* section, enter the following values:

- *lf_position_y* = 0.065
- *lf_position_x* = 0.01



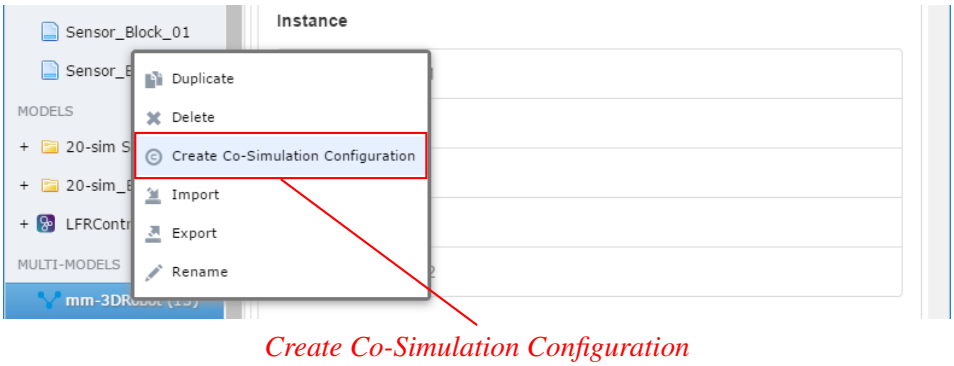
Step 27. Repeat the previous step for the second sensor, {*sensor2*}.*sensor2*, with the following values:

- *lf_position_x* = -0.01
- *lf_position_y* = 0.065

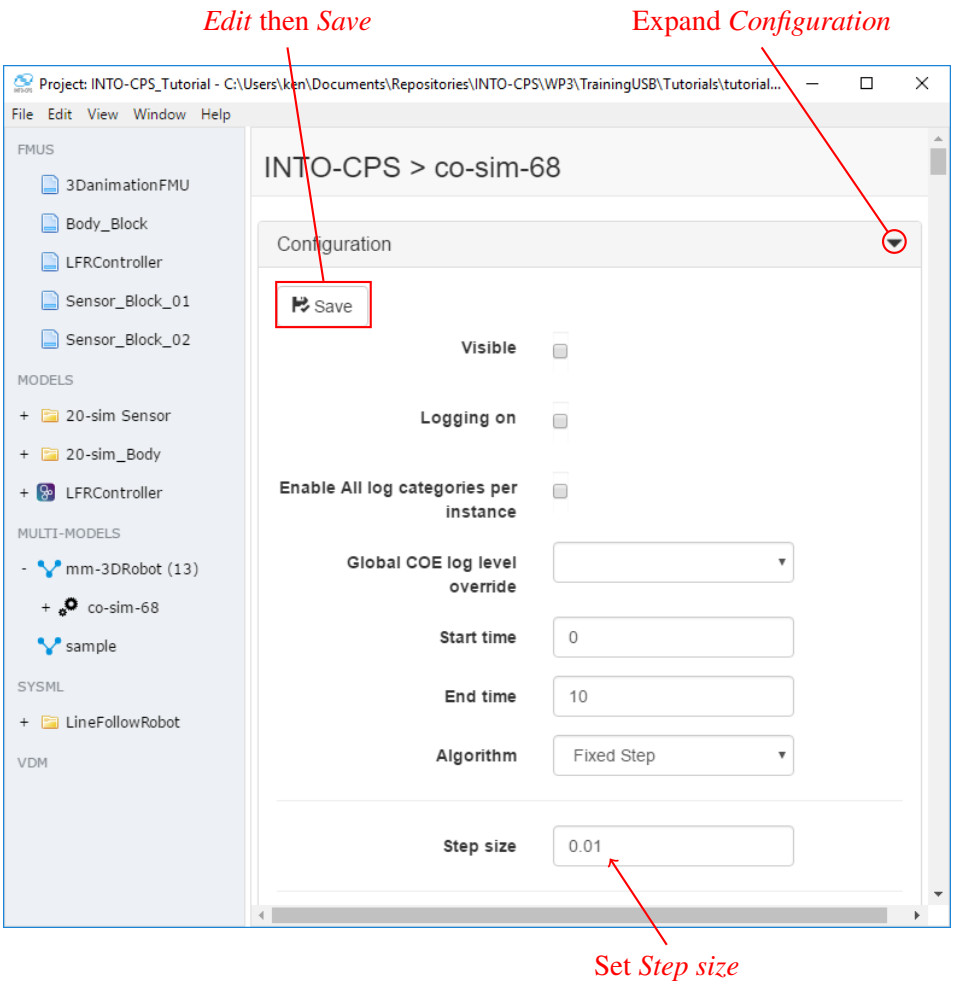
Step 28. *Save* the *Configuration*.



Step 29. Right-click on the new multi-model configuration and select *Create Co-simulation Configuration*.



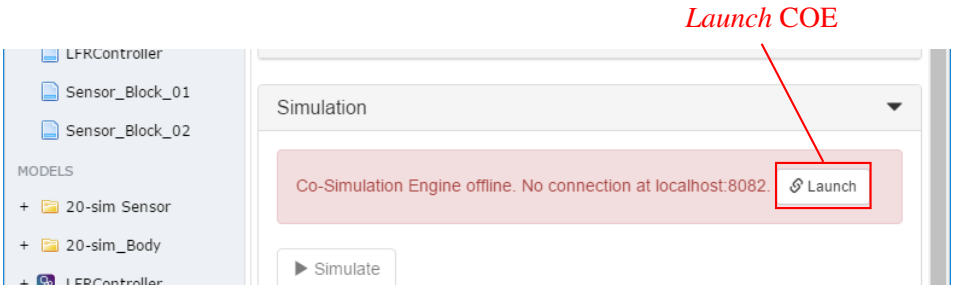
Step 30. Set the *Step size* to 0.01. Don't forget to press *Edit* then *Save*.



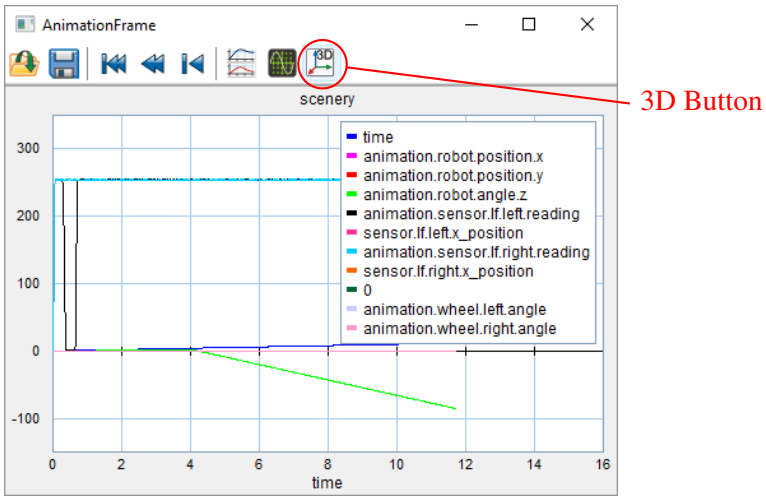
Step 31. Check *lf_1_sensor_reading* from {*sensor1*}.*sensor1* and {*sensor2*}.*sensor2* to see the sensor values appear in the *Livestream Configuration*.



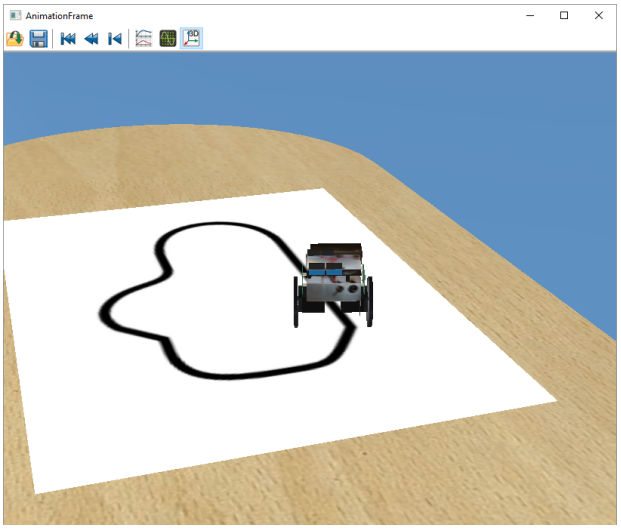
Step 32. Launch the COE if necessary (see *Tutorial 1 — First Co-simulation* for a reminder if needed).



Step 33. When the COE is running, click the *Simulate* button. The *Animation Frame* should appear. You can click the *3D* button to see the 3D visualisation of the robot.



Step 34. If everything went well, the robot should follow the line as in *Tutorial 2 — Adding FMUs*.



You can go back to *Overture* and look at the logic in `Controller.vdmrt`, and try to make some changes. Just repeat Step 16. to Step 19. to regenerate and copy the FMU, then press *Simulate*.