Workshop - C2: Cyber-physical System Modeling using Modelica for Smart and Sustainable Communities

Jing Wang[1], Jessica Stershic[1], Sen Huang[2], Wangda Zuo[1,*]

[1]University of Colorado Boulder

[2]Pacific Northwest National Laboratory

*Corresponding author: Wangda Zuo, wangda.zuo@colorado.edu

Tutorial 1: Smart and Connected Community Library

The objective is to understand what a smart and connected community (SCC) is and how different infrastructures are related. This will be accomplished through modeling in the equation-based object-oriented language Modelica. The following open source libraries will be used in this tutorial:

Modelica Buildings Library (version 5.0.0)[1]

Smart and Connected Community Library (version 1.1)[2].

At the end of this workshop, you will be more familiar with the SCC library through an energy system example, and an energy and transportation system example. This guide has step-by-step instructions to accomplish these goals. Dymola 2020 in Linux system was used to create this tutorial.

Step-by-Step Guide

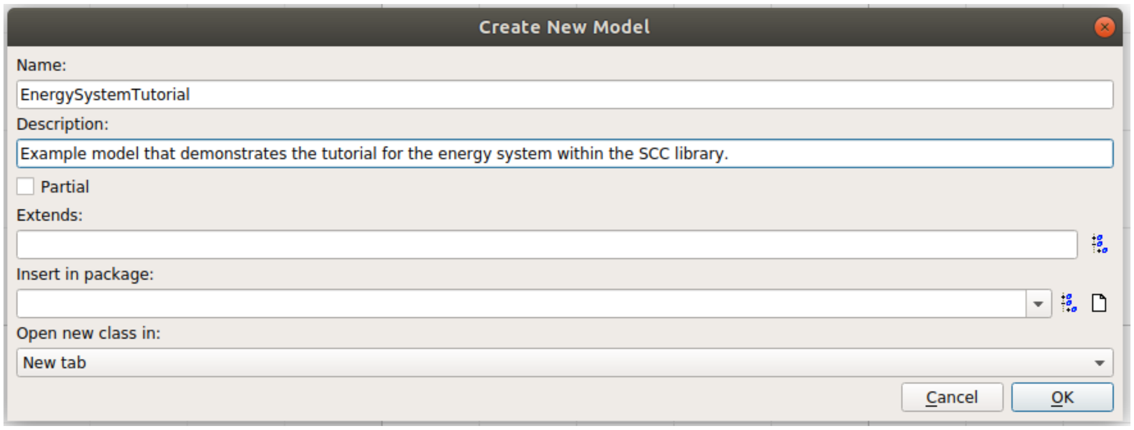
Example 1 will demonstrate how to use the energy system model within the SCC library.

1. Load the SCC library wherever it is stored on your system.

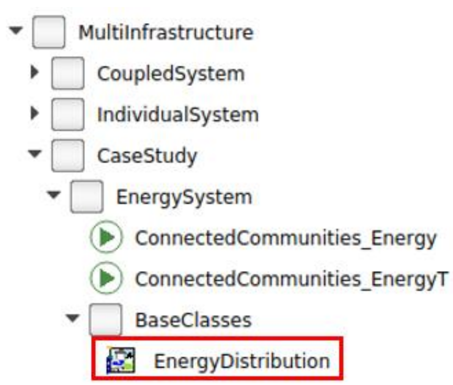
2. Create a new model File > New > Model or Crtl+Shift+M.

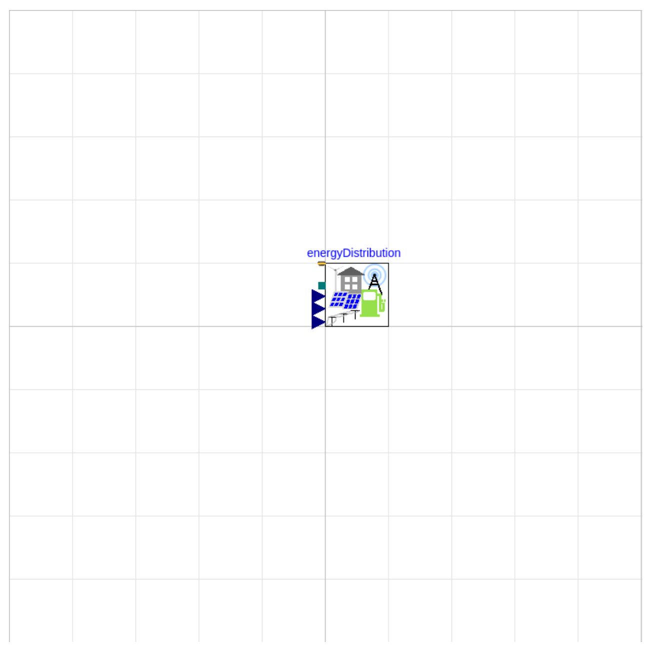
[1] https://simulationresearch.lbl.gov/modelica/downloads/archive/modelica-buildings.html
[2] https://www.colorado.edu/lab/sbs/scc-library

3. Fill in model information and click OK.

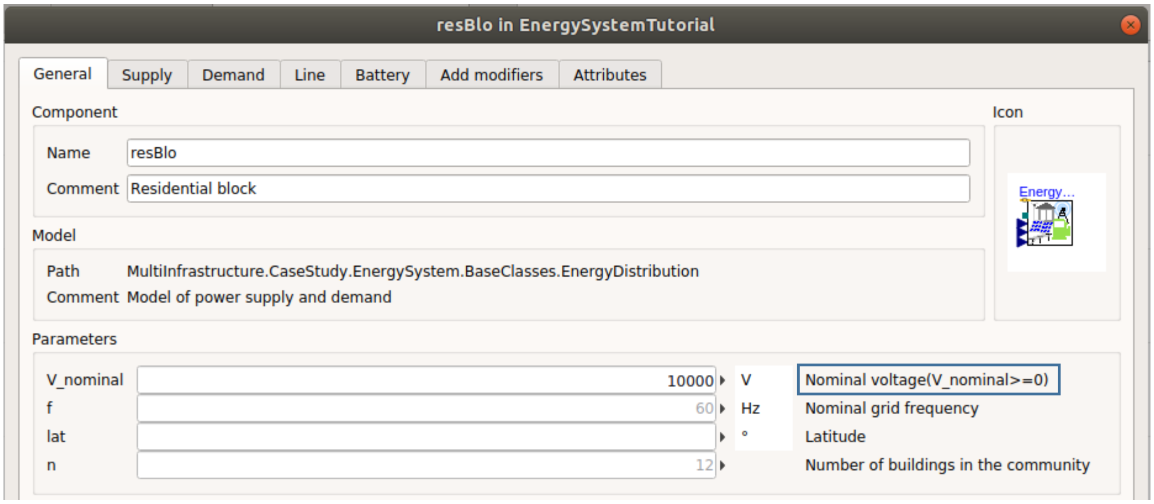


4. Insert EnergyDistribution Block from the Package browser.
   Click on EnergyDistribution and drag into the diagram area. Your model should now look like this.

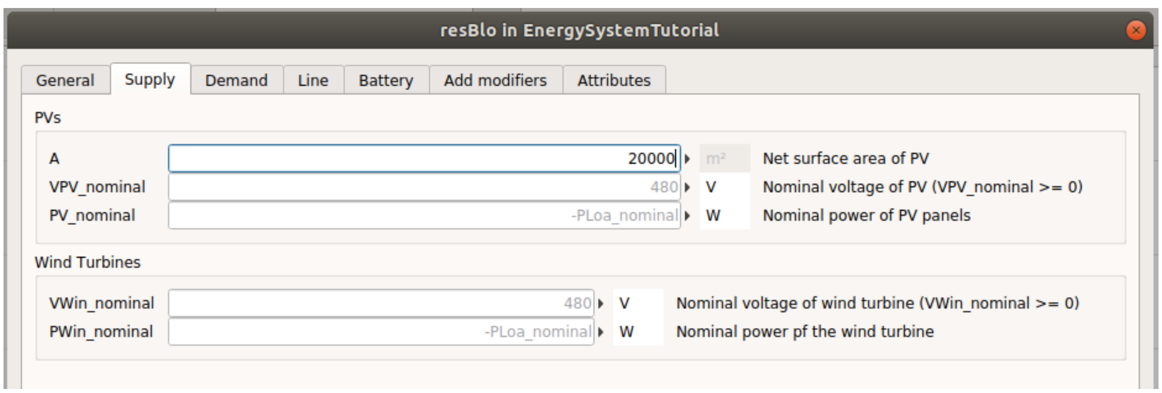


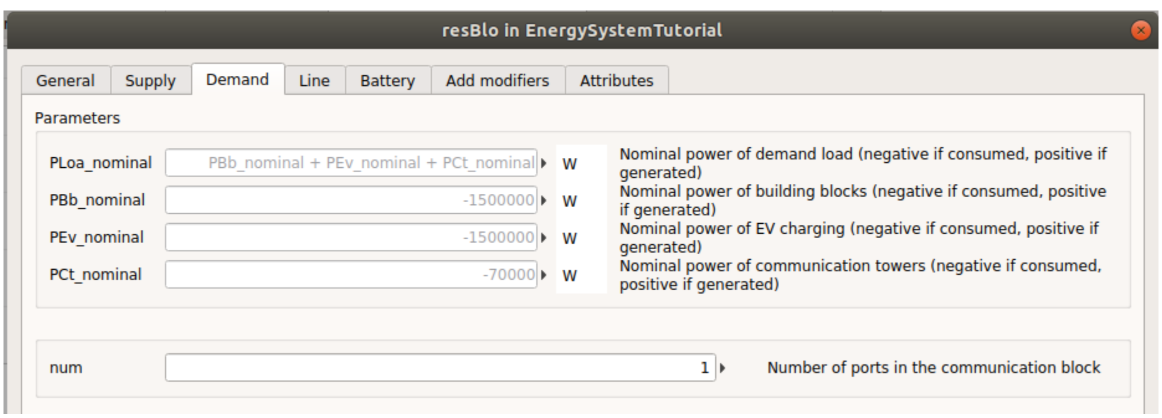


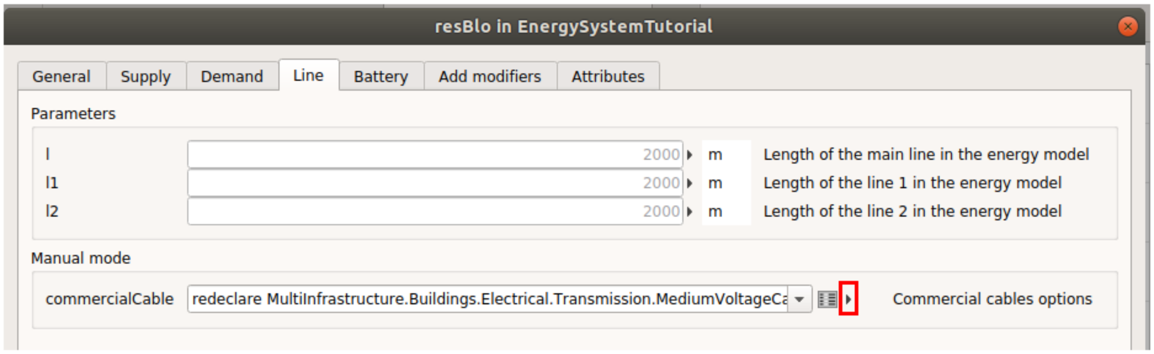
5. Rename energyDistribution block: Double click on energyDistribution block and fill in information like below.



It is a good idea to add comments for future users to know what your component is. The grey values are default parameters that you can change.

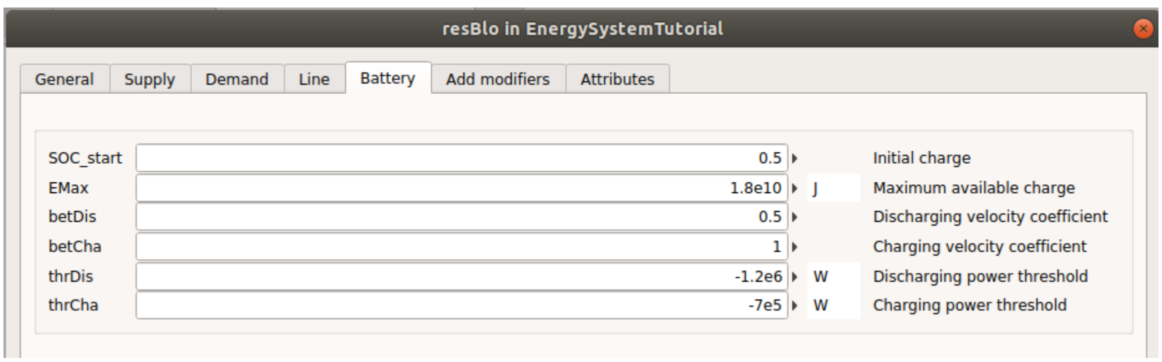






Click the arrow, Edit Text, then enter the following:

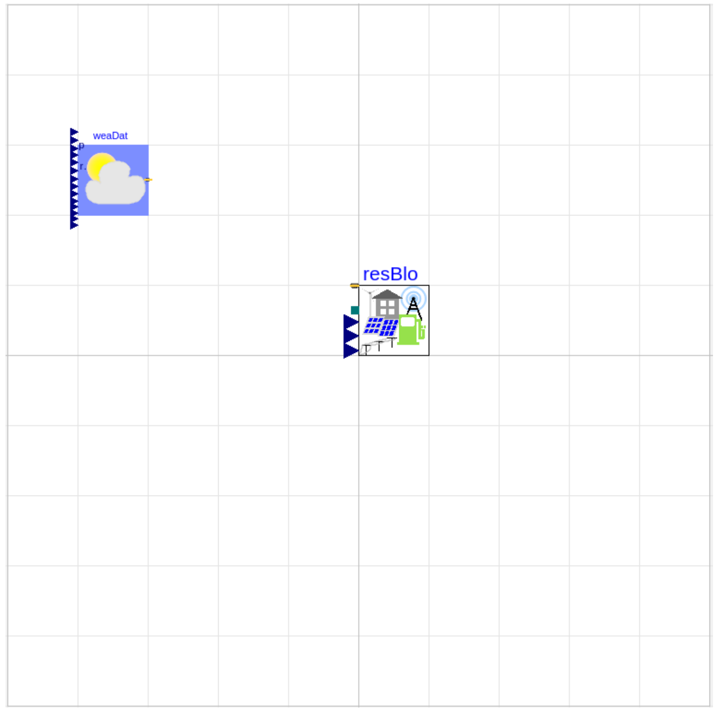
redeclare
MultiInfrastructure.Buildings.Electrical.Transmission.MediumVoltageCables.Annealed_
Al_10 commercialCable



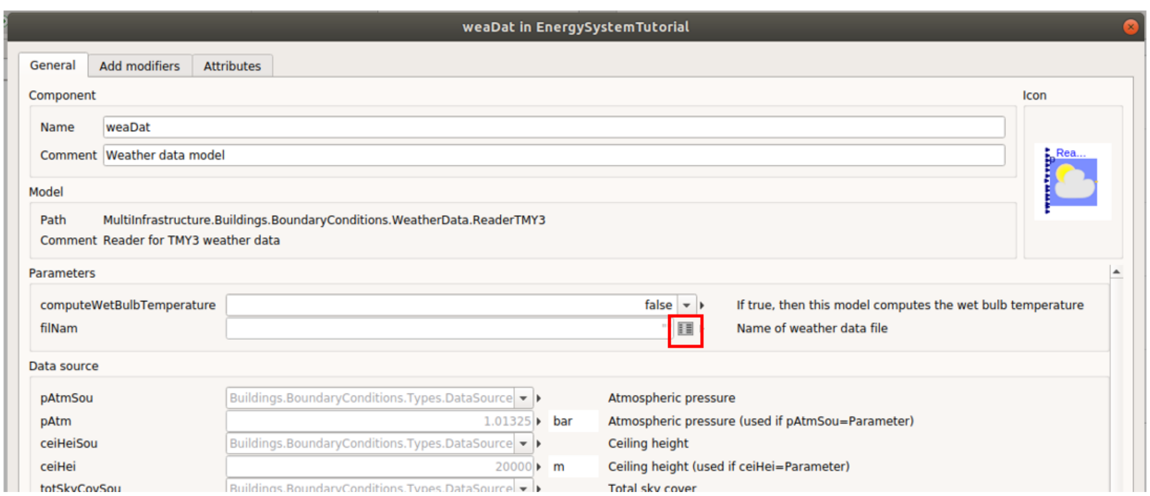
Click OK at the bottom once all of the parameters have been changed.

6. Add in weather data.

Again, drag and drop ReaderTMY3 into your model. Your screen will now look like this.



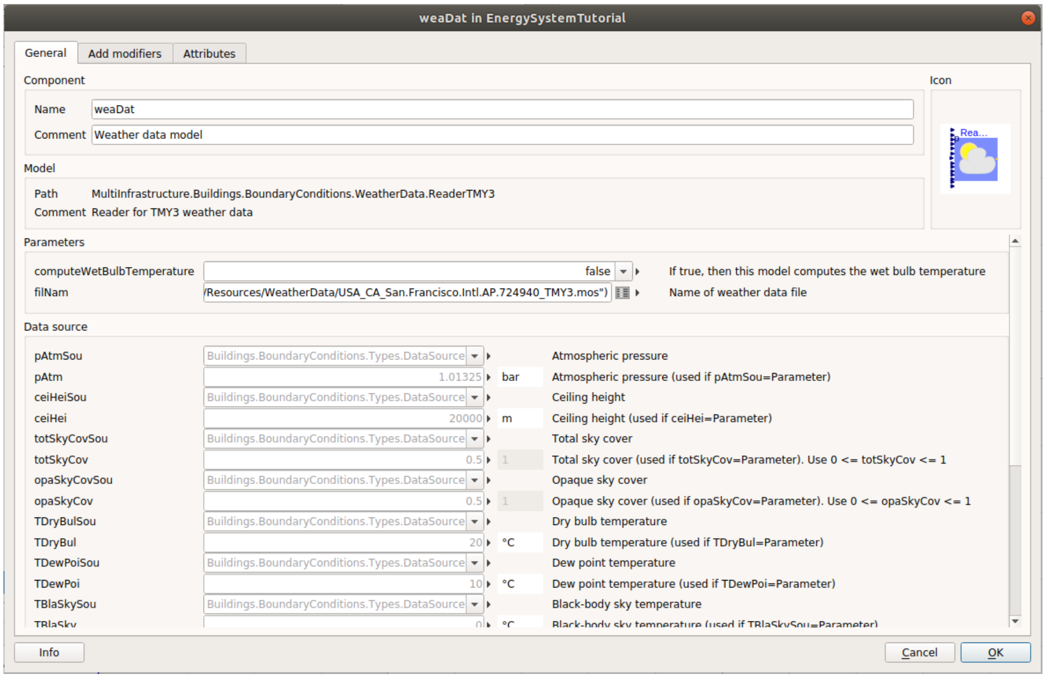
Double click on weaDat and change the information as seen below.



To load the weather file, click on the grey table icon called "edit". This will pull up a screen where you can choose the weather file.

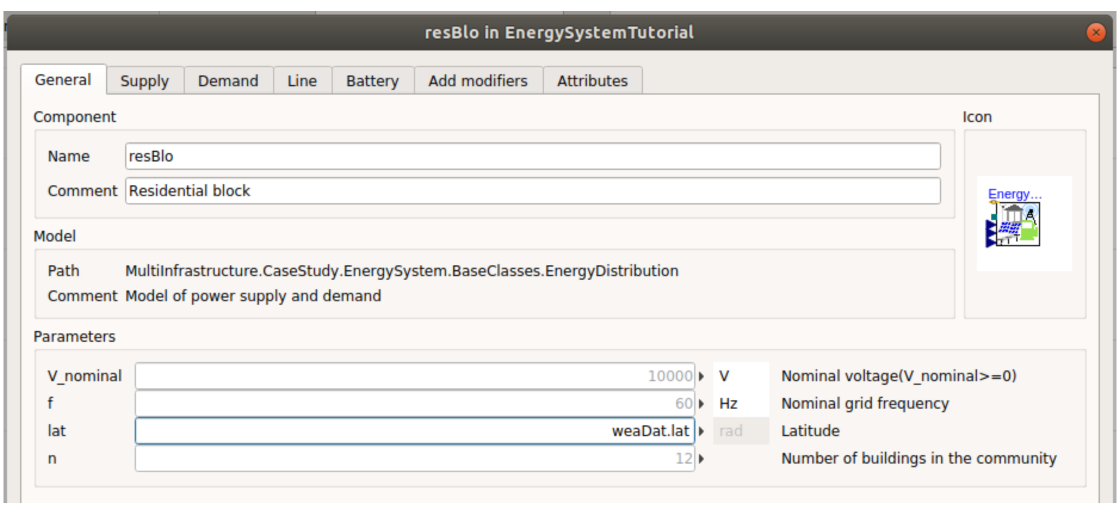
Resources > WeatherData > USA_CA_San.Francisco.Intl.AP.724940_TMY3.mos

Click open and now your screen should look like this.

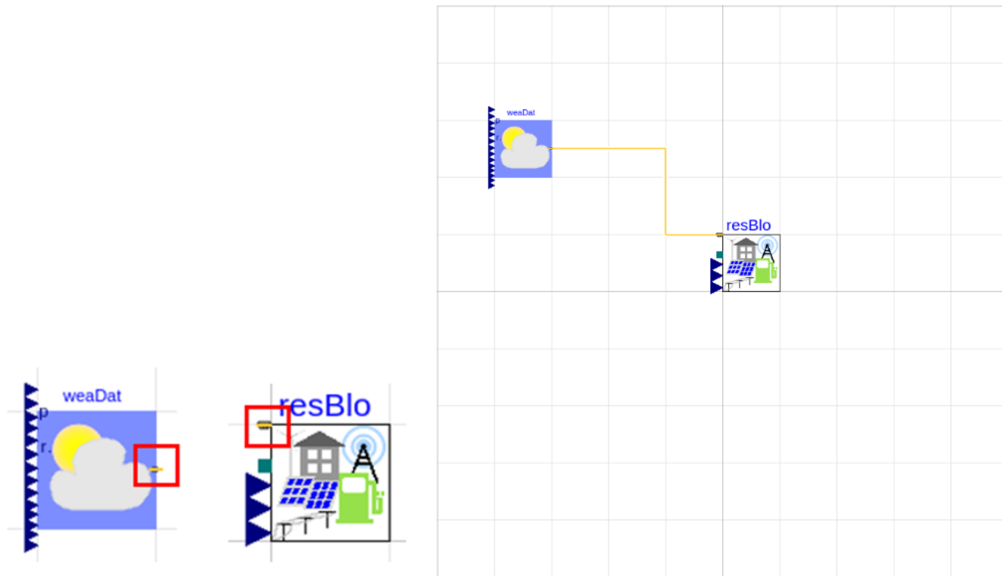


Click OK.

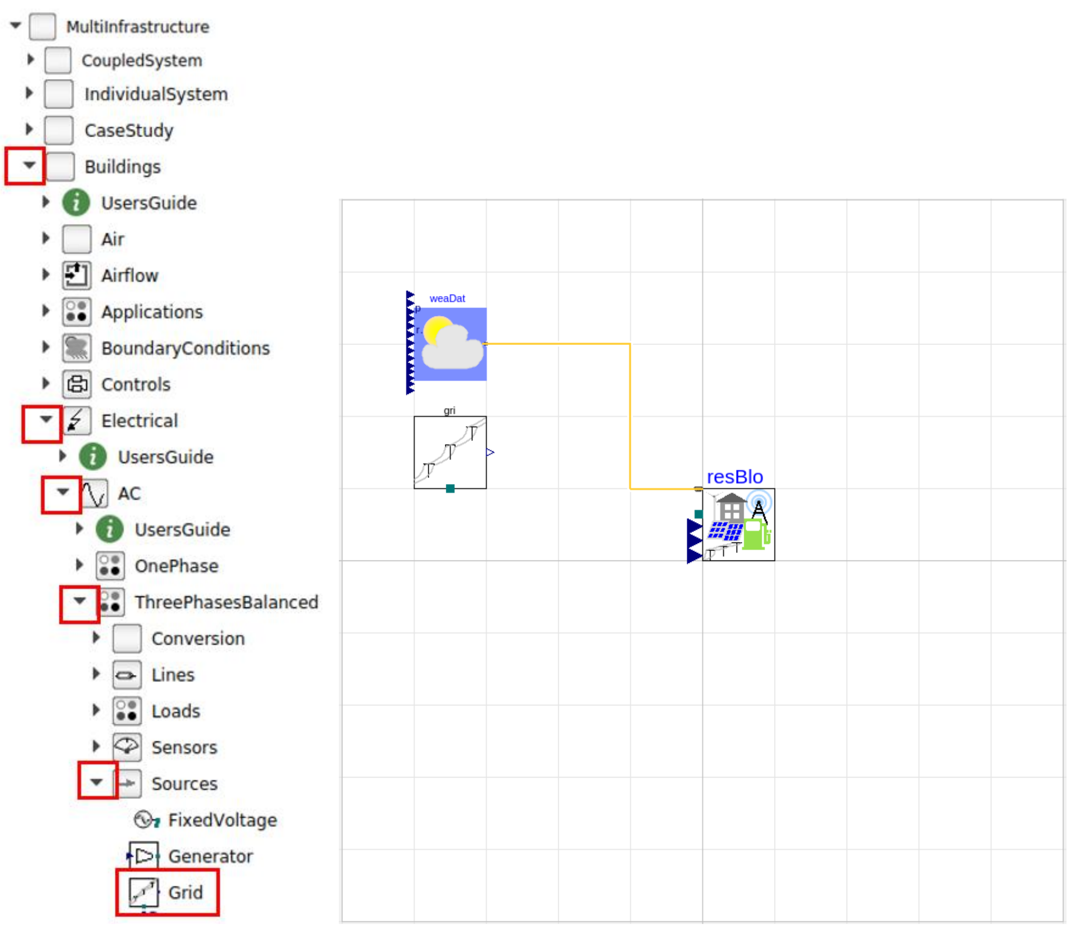
7. Add latitude to resBlo: Double click on resBlo. Now that we have added in the weather data, we can add more information to the resBlo *lat* (Latitude) parameter. Type weaDat.lat into the *lat* parameter box and click OK.



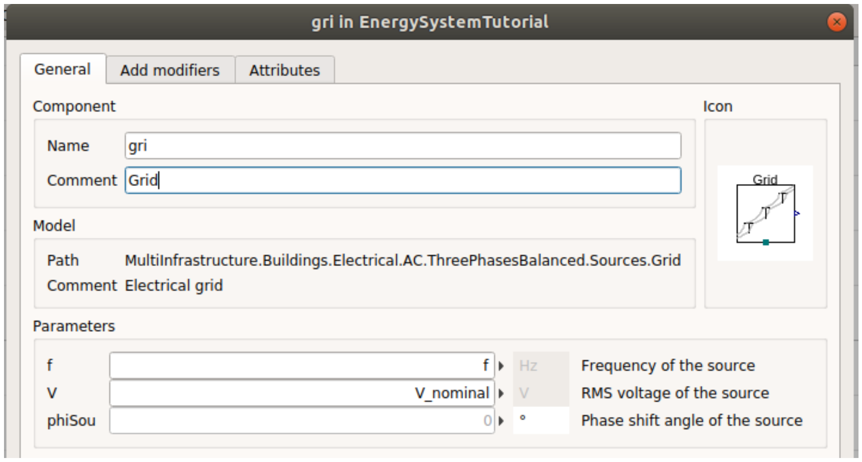
8. Connect weaDat and resBlo: You can see that weaDat and resBlo each have a yellow bus which can be connected. Click on the weaDat bus and drag it to connect with the resBlo bus. The location of the connection line is not that important, however, it is good practice to have the connection lines align with the grid lines.



9. Add the grid for electricity. Click on the Grid component and drag it into your model. Your screen should now look like this.

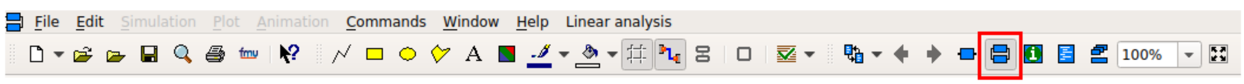


Double click on gri and edit the parameters *f* and *V*.

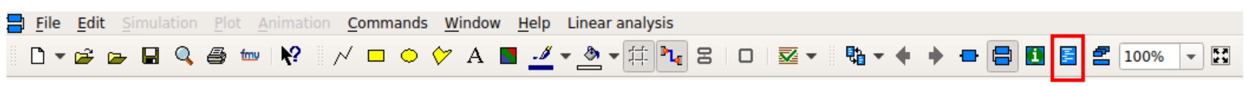


Now we need to define *f* and *V*. Since these parameters are also used in other components at the system level, they are considered system-level parameters.

This whole time we have been editing our model in the graphical representation called **Diagram**.



Models can also be edited in text mode called **Modelica Text**.



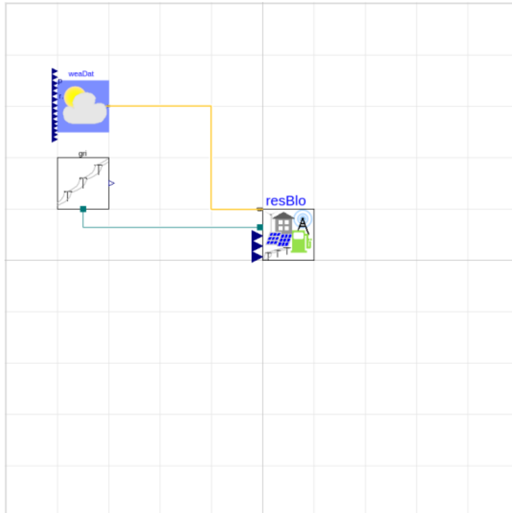
Click on **Modelica Text**. The beginning lines of the text should be changed to the following.

```
model EnergySystemTutorial
  "Example model that demonstrates the energy system tutorial within the SCC library"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Voltage V_nominal = 10000 "Nominal grid voltage";
  parameter Modelica.SIunits.Frequency f = 60 "Nominal grid frequency";
```

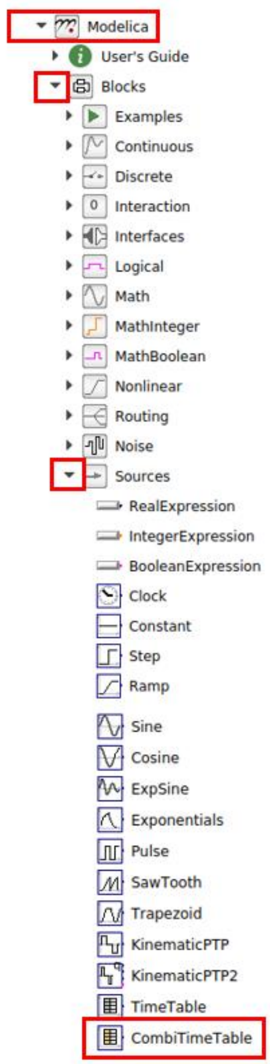
These lines give the model an icon in the left panel and also define V_nominal and f.

extends Modelica.Icons.Example;
parameter Modelica.SIunits.Voltage V_nominal = 10000 “Nominal grid voltage” ;
parameter Modelica.SIunits.Frequency f = 60 “Nominal grid frequency”;

Switch back to **Diagram**. Again we see that the Grid (gri) and resBlo have matching connectors (electrical terminals), both green. Now, click on the green connector on gri and connect it to resBlo. Your model now should look like this.

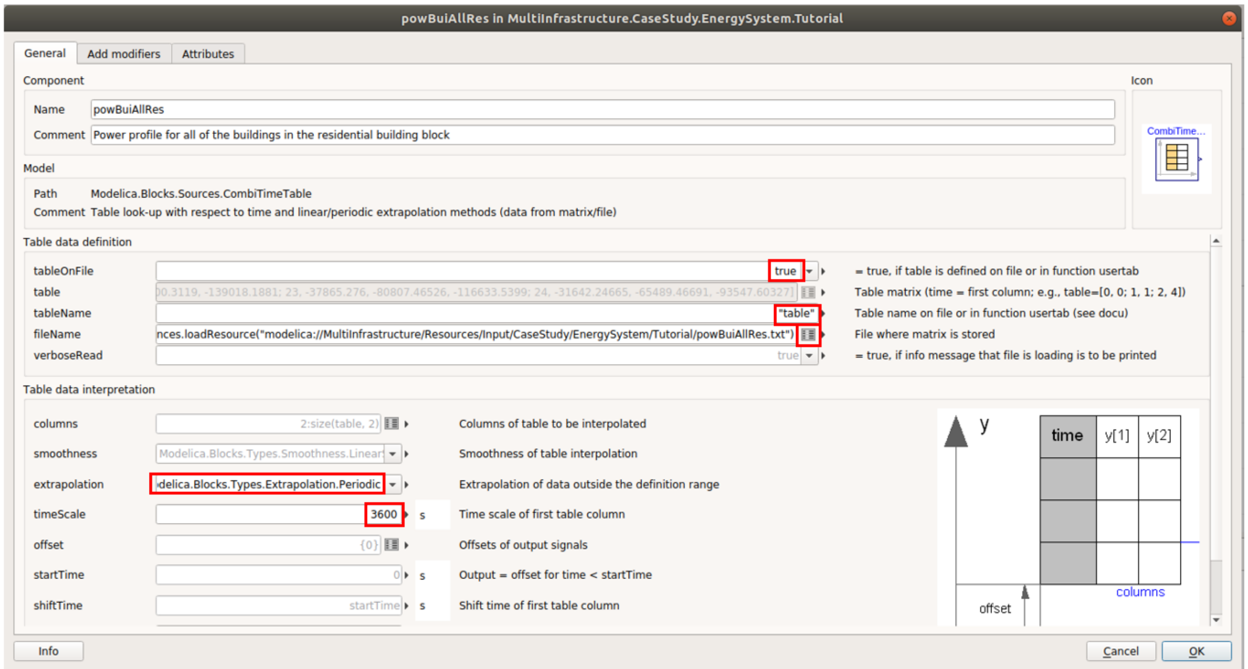


10. Add in building demand profiles for resBlo. Add in the CombiTimeTable to your model by clicking and dragging.



Double click on combiTimeTable to change some of the information.

Now we will load the building profile information. This data comes from the DOE Commercial Reference Buildings Models and Building America House Simulation Protocols. Fill out *tableOnFile* and *tableName* as seen below. To access the *fileName*, click the Edit button at the end of the line for table.



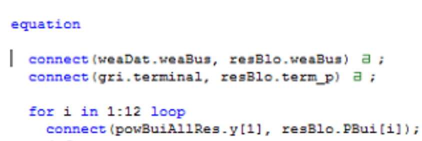
To open the proper txt file in *fileName*, go to:

Resources>Input>CaseStudy>EnergySystem>Tutorial>powBuiAllRes.txt

Then also fill in *extrapolation* and *timeScale* as seen above. Click OK.

11. Connect powBuiAllRes to resBlo using a script. Models in lower-level packages within the Energy System require 12 building loads connected to the resBlo. Instead of drawing 12 connections and picking the correct columns of data to use for each component, we will write a script to do that for us.

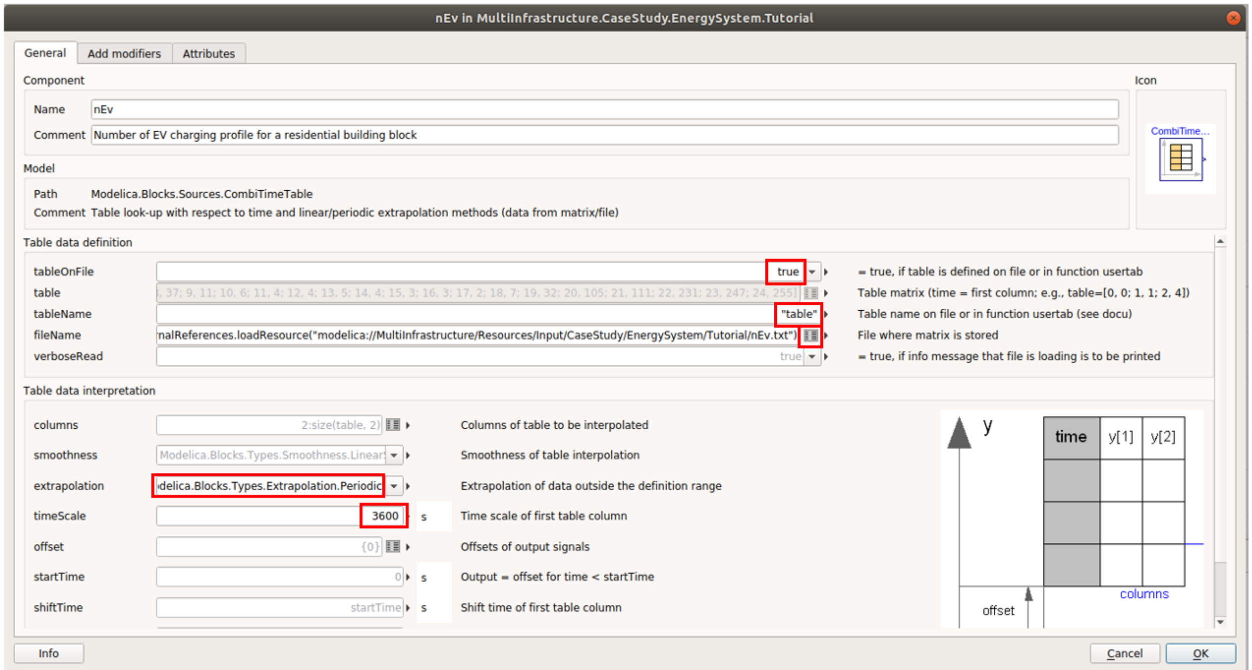
Click on **Modelica Text**. Add the following to the equation section.




```
for i in 1:12 loop
  connect(powBuiAllRes.y[1], resBlo.PBui[i]);
end for;
```

Now, even if the connections are not shown via a line, they are connected via equations.

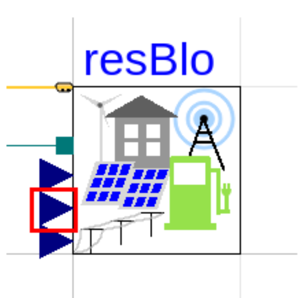
12. Add in electric vehicle (EV) charging profile. Drag another CombiTimeTable into your model like seen in step 10. The model path is Modelica.Blocks.Sources.CombiTimeTable. Double click on the table and fill in your information like so:



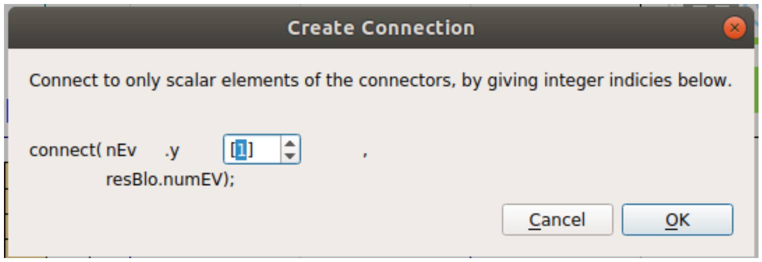
Double click the Edit button at the end of the *fileName* parameter. Go to: Resources>Input>CaseStudy>EnergySystem>Tutorial>nEV.txt

Now the number of EVs charging should be loaded. Click OK. You should now be back to your model screen.

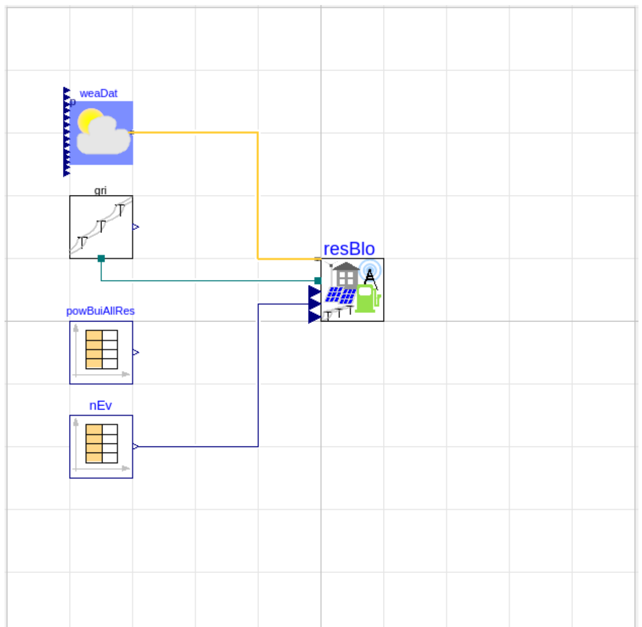
13. Connect nEv to resBlo



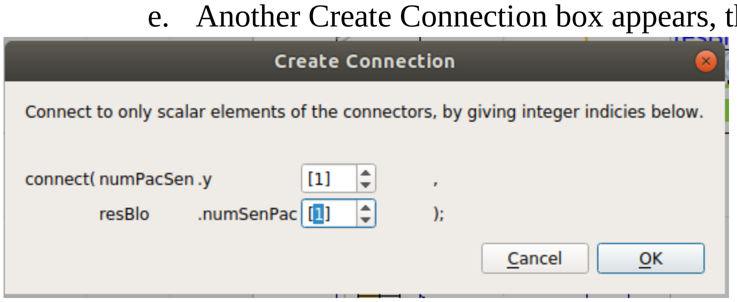
If you toggle your mouse over the blue triangles or inputs to resBlo you will see text descriptors appear which tell you what the input should be. After toggling, it is clear the second triangle is the one we need to connect nEv to. After connecting to the second input, this Create Connection screen appears. It is asking you what data you would like to connect with the input. The nEv data only has one column and thus put 1 in the box. Click OK.



At this point, your model is looking something like this.

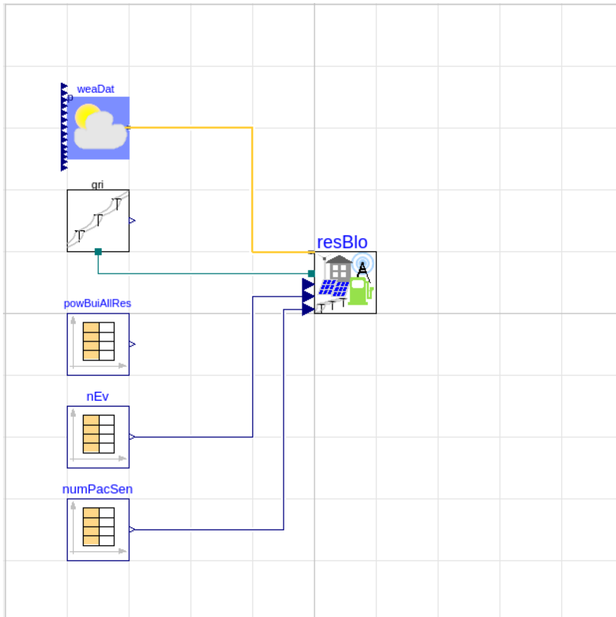


14. Add in communication and connect to resBlo following these steps:
    a. Add a new CombiTimeTable to your model. The *Name* is numPacSen. The *Comment* is “Number of packages sent for a residential building block”.
    b. Fill in the *tableOnFile*, *tableName*, *extrapolation* and *timeScale* parameters the same way as the past two tables.
    c. Click the Edit button at the end of *fileName*. Open SCC_Workshop_Data and by going to Resources>Input>CaseStudy>EnergySystem>Tutorial>numPacSen.txt
    d. Get back to your model. Determine which resBlo input numPacSen should be connected to and then connect them.
    e. Another Create Connection box appears, this time with two options.

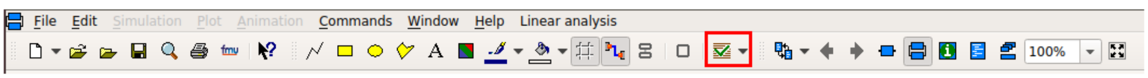


Again, numSenPac data has one column and resBlo.numSenPac has one column, so each should have a 1.

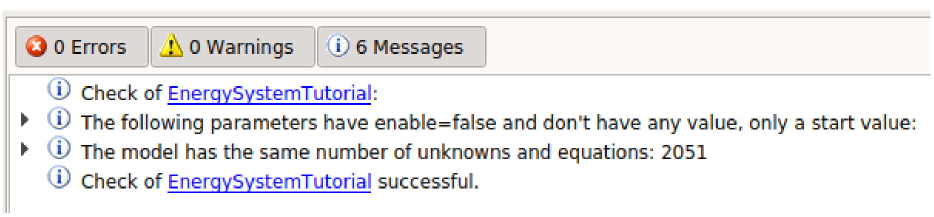
f. Return to your model. Your model should now look something like this.



15. Check the model to see if the model can be translated. Click the check box as seen below to check.



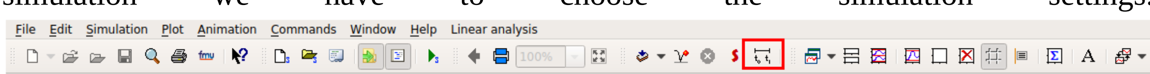
The log window will look something like this:



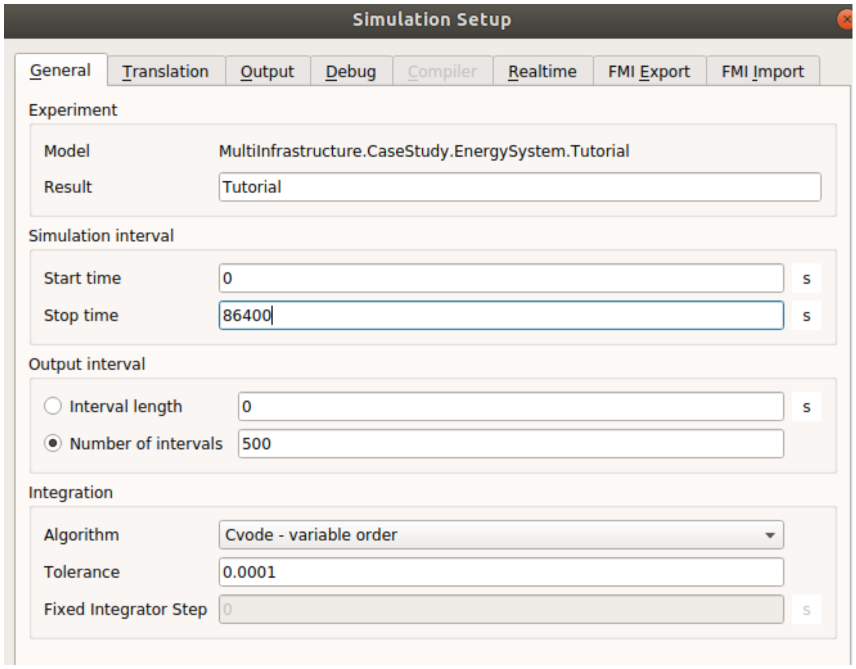
16. Switch to Simulation mode.

This entire time, we have been in Modeling mode. Now switch to Simulation.

17. Simulation Setup settings: There is now a new ribbon at the top. Before running the simulation we have to choose the simulation settings.



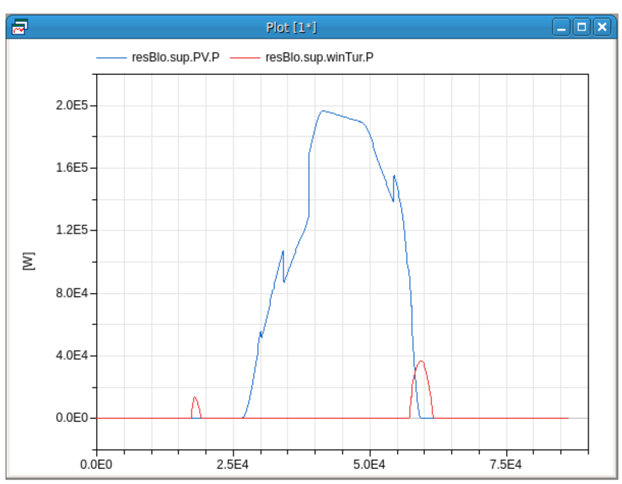
Change the Simulation Setup box to look like below. We will first run this example for 86,400 seconds (one day). Click OK.



18. Click Simulate.

19. Evaluate results:
    a. Go to resBlo.sup.PV.P and click the box next to it.
    b. Go to resBlo.sup.winTur.P and click the box next to it.

These two items together show you how much renewable energy was produced that day. This is just a short overview of the results. Feel free to look at some of the other results. This concludes the first tutorial with the SCC Library.

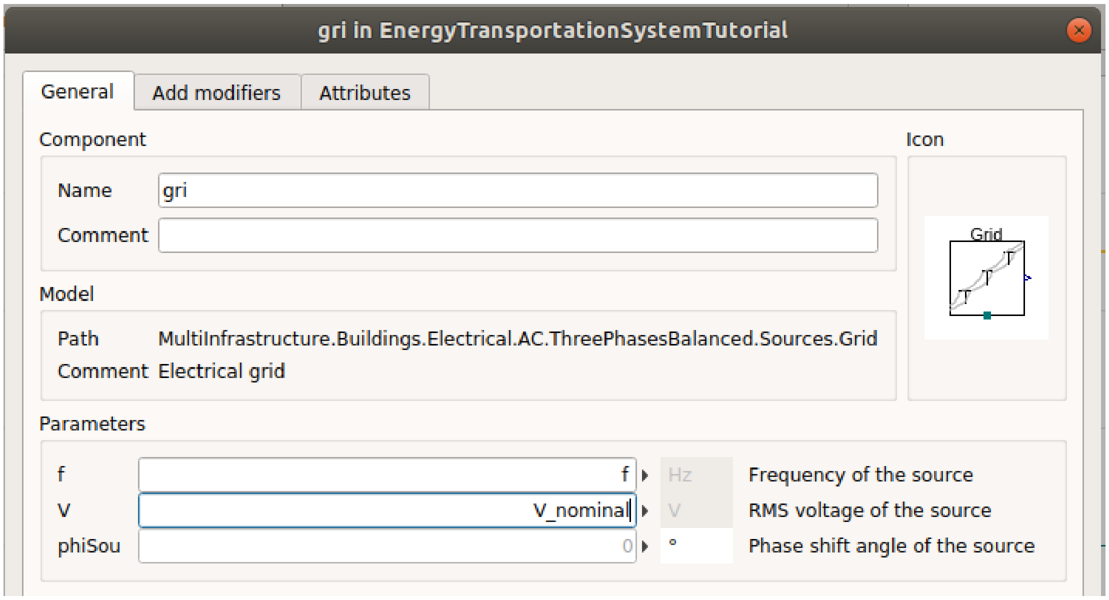


Example 2 will show how to use the energy and transportation systems coupled in the SCC library.

1. New model

Open a new model and call it EnergyTransportationSystemTutorial. Description can be: Example model that demonstrates the tutorial for the energy and transportation systems within the SCC library.

2. Insert the TMY3 reader and load the weather file: See step 6 in the first tutorial. MultiInfrastructure.Buildings.BoundaryConditions.WeatherData.ReaderTMY3
3. Add the grid for electricity: See step 9 in the first tutorial. MultiInfrastructure.Buildings.Electrical.AC.ThreePhasesBalanced.Sources.Grid



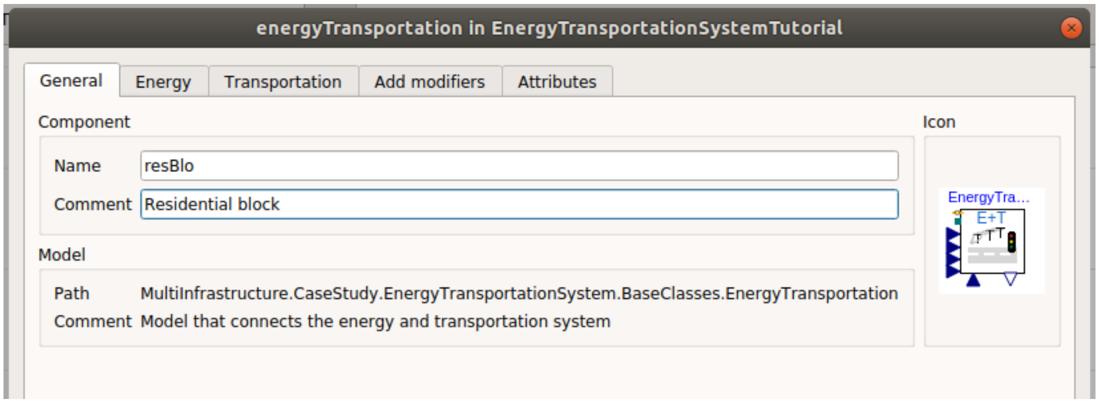
4. Add in grid frequency, grid voltage, and the example icon in **Modelica Text**. Switch to **Modelica Text** and add the following lines:

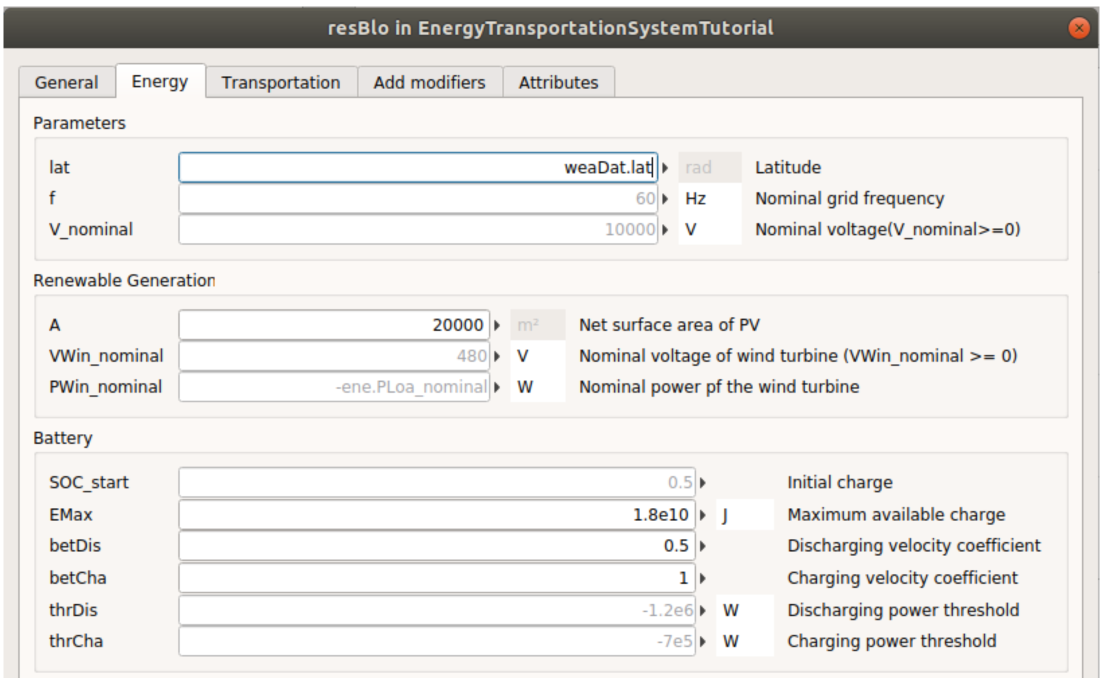
```
extends Modelica.Icons.Example;
parameter Modelica.SIunits.Voltage V_nominal = 10000 "Nominal grid voltage";
parameter Modelica.SIunits.Frequency f = 60 "Nominal grid frequency";
```

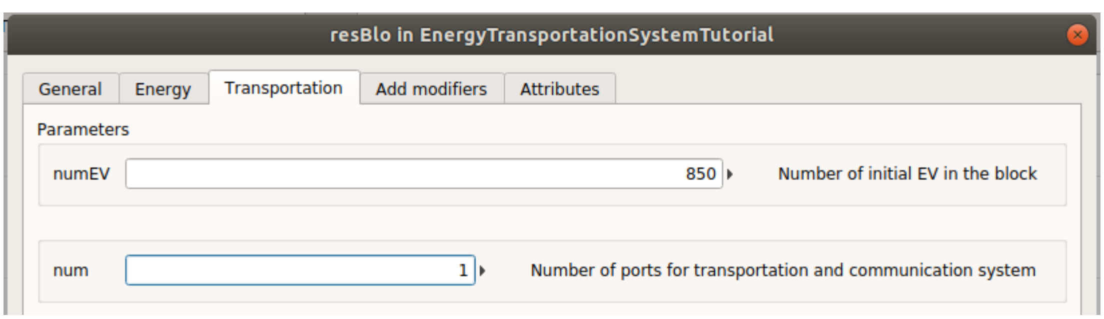
5. Insert residential and commercial blocks: Insert two EnergyTransportation blocks: MultiInfrastructure.CaseStudy.EnergyTransportationSystem.BaseClasses.EnergyTransportation

Notice that these blocks have one more set of input and output ports than the ones used in the last example.

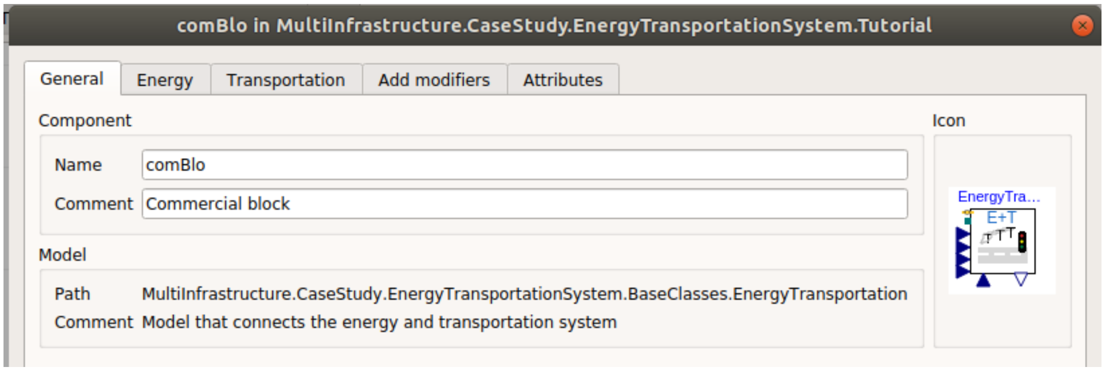
6. Change the parameters on the first energyTransportation block to match the following images.

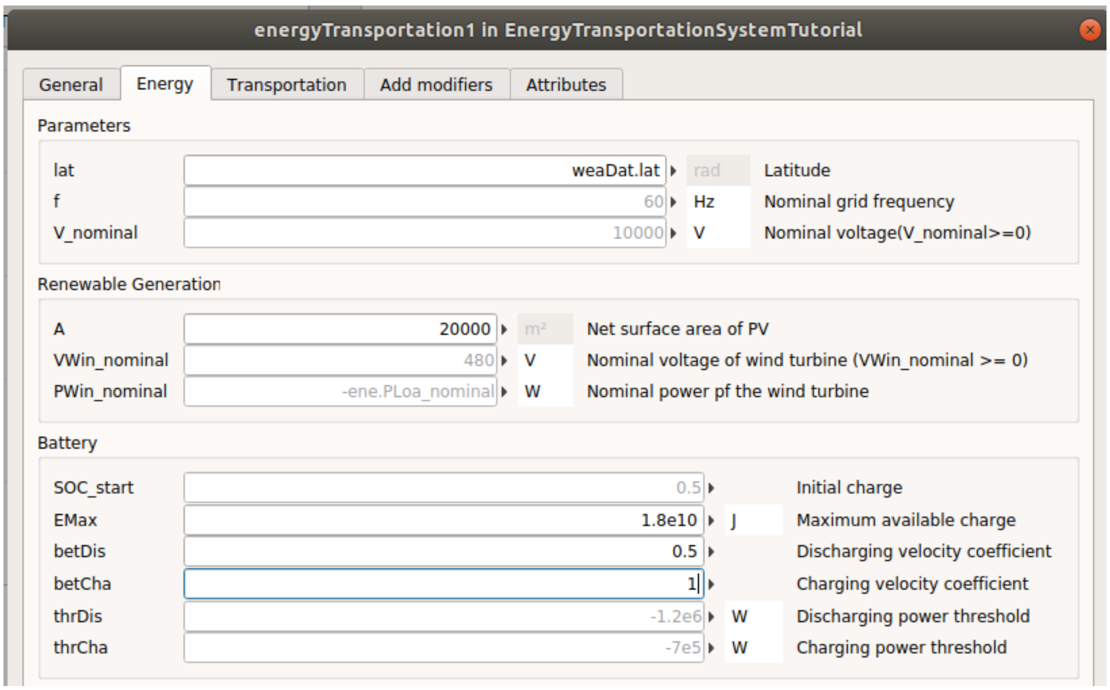


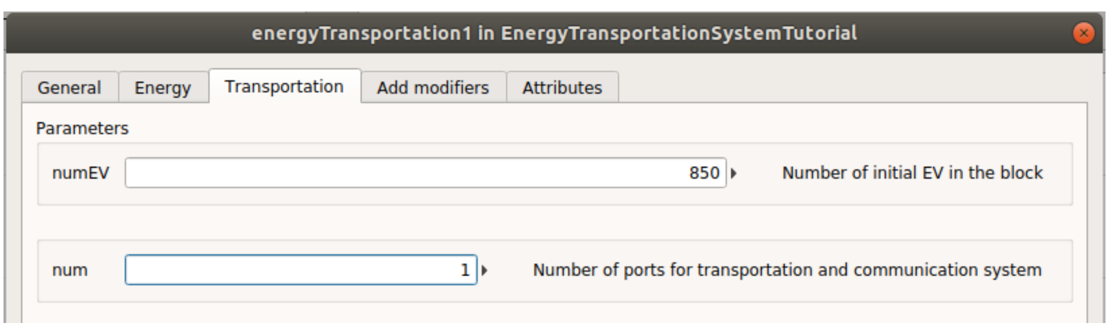




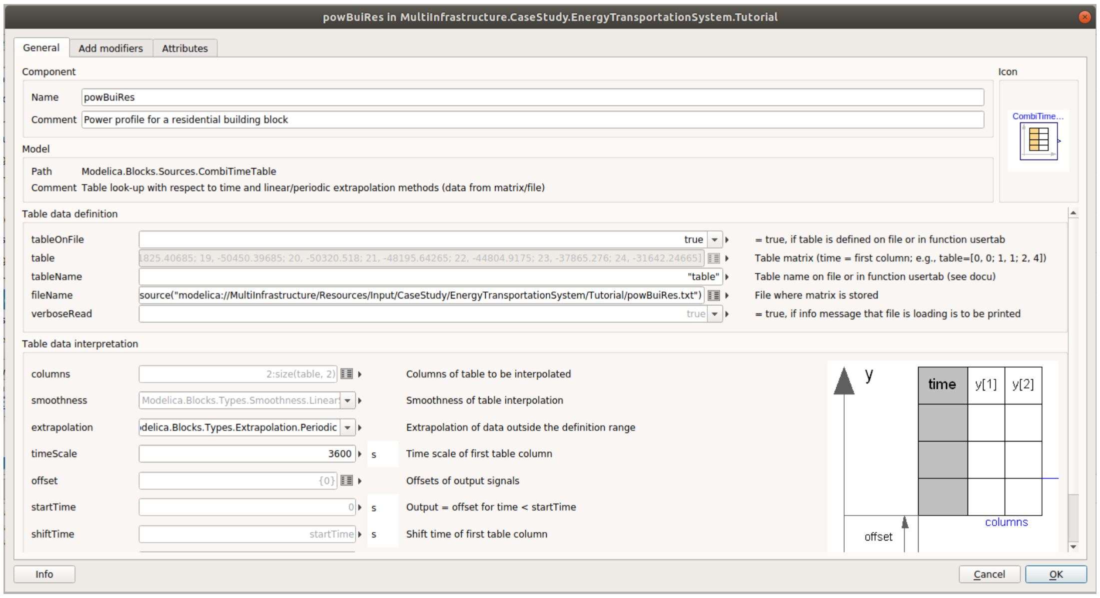
7. Change the second energyTransportation block to match the following parameters.





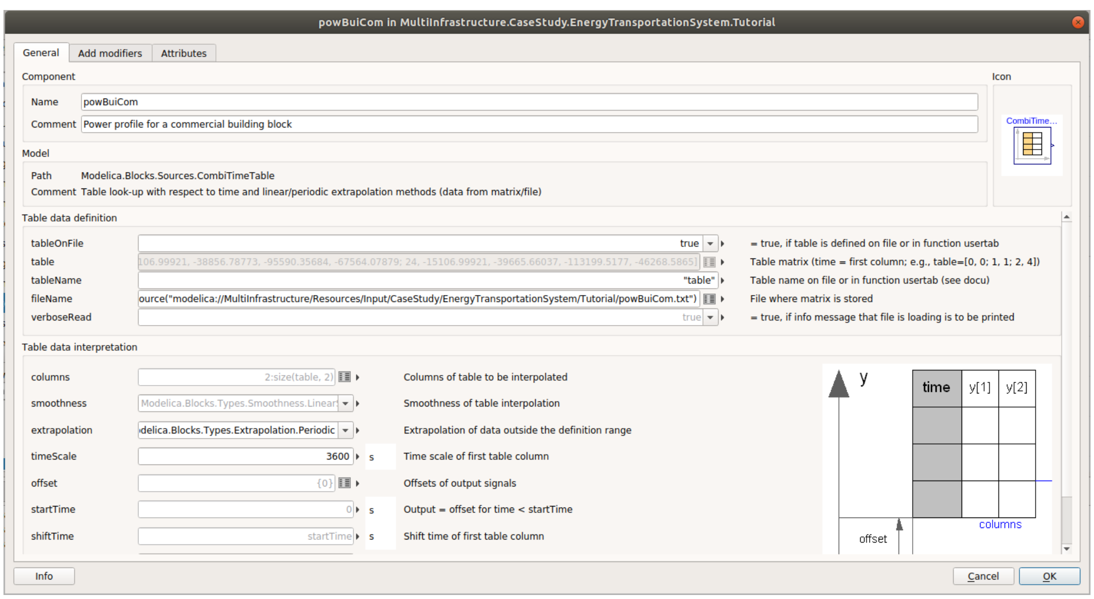


8. Connect weaDat to the resBlo and the comBlo.
9. Connect gri to the resBlo and the comBlo.
10. Add the building load data to resBlo and comBlo: Add two instances of Modelica.Blocks.Sources.CombiTimeTable to your model. On the first one, change the parameters to match the following image:



The *fileName* can be loaded from:
Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>powBuiRes.txt

The second instance of the CombiTimeTable can be adjusted like:



The table can be loaded from Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>powBuiCom.txt

11. Connect powBuiRes and resBlo: Similar to the last tutorial we are going to write a script to perform the connections for us. Switch to **Modelica Text** and write the following script.

```
for i in 1:12 loop
  connect(powBuiRes.y[1], resBlo.PBui[i]);
end for;
```

You can copy and paste the below script if you like.

```
for i in 1:12 loop
  connect(powBuiRes.y[1], resBlo.PBui[i]);
end for;
```

12. Connect powBuiCom to comBlo.

Similar to the previous step, we will write a script that connects powBuiCom to comBlo. If you remember, powBuiCom has multiple columns with different data. This data corresponds to different building types. In this tutorial, we will only model one type of building. You could emulate the following script to test out different buildings or different columns of data.

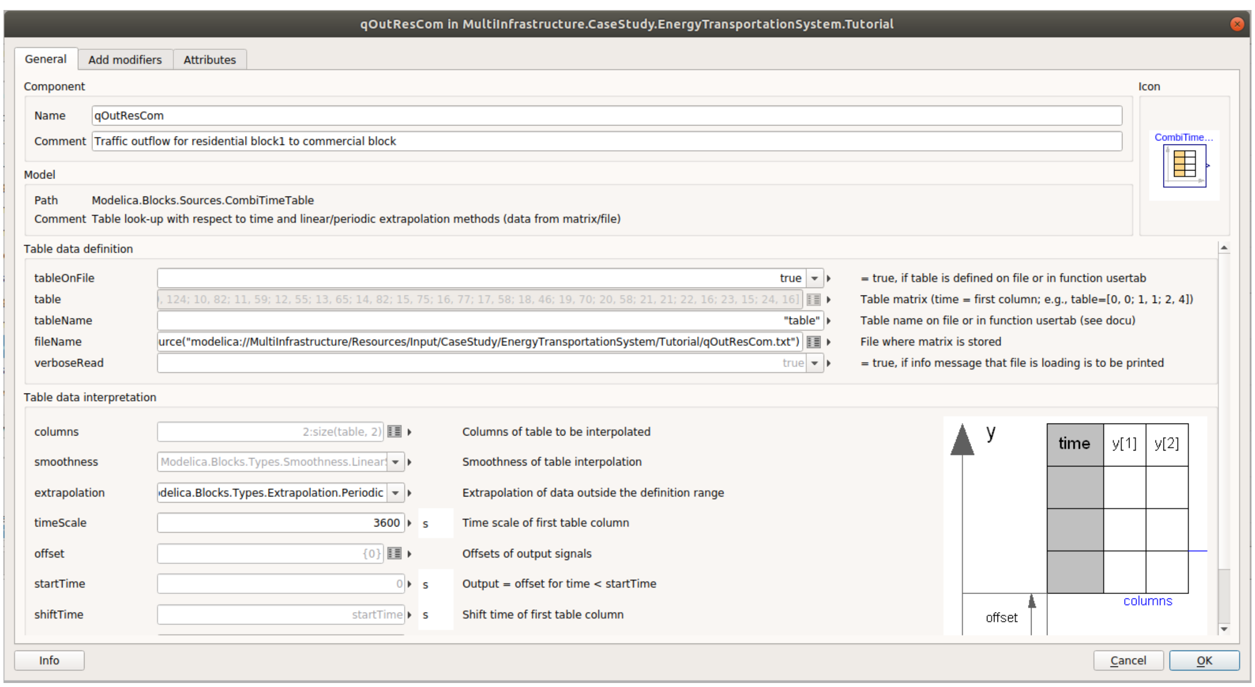
```
for i in 1:12 loop
  connect(powBuiCom.y[1], comBlo.PBui[i]);
end for;
```

Feel free to copy and paste the following text:

```
for i in 1:12 loop
  connect(powBuiCom.y[1], comBlo.PBui[i]);
end for;
```

13. Add in the traffic data.

Add in two more instances of the CombiTimeTable to your model. Change the table to have the following name, description, and parameters.

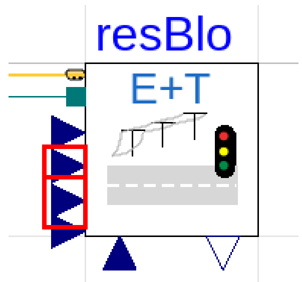


Next, hit the Edit button at the end of the *fileName* line to add in the matrix data. The table can be loaded from:
Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>qOutResCom.txt

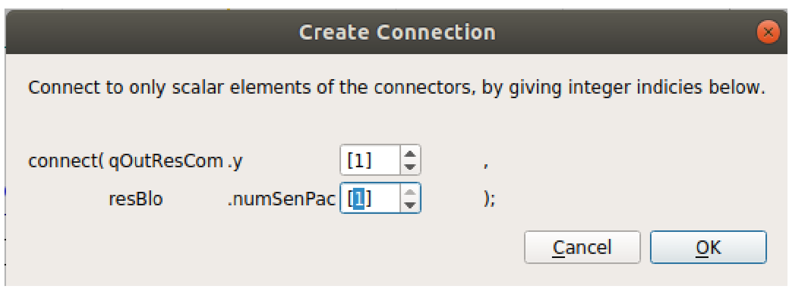
Hit OK and then you should return to your model.

14. Connect qOutResCom to resBlo.

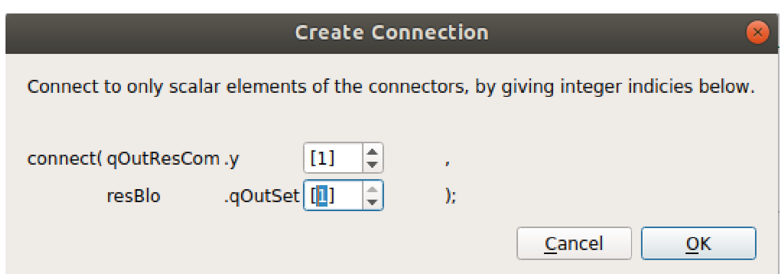
When connecting qOutResCom to resBlo, it will have to be connected twice to the inputs in the second and third position as shown below.



Connect qOutResCom to the second input and a Create Connection box will pop up. Put 1 in each of the choices as shown below.

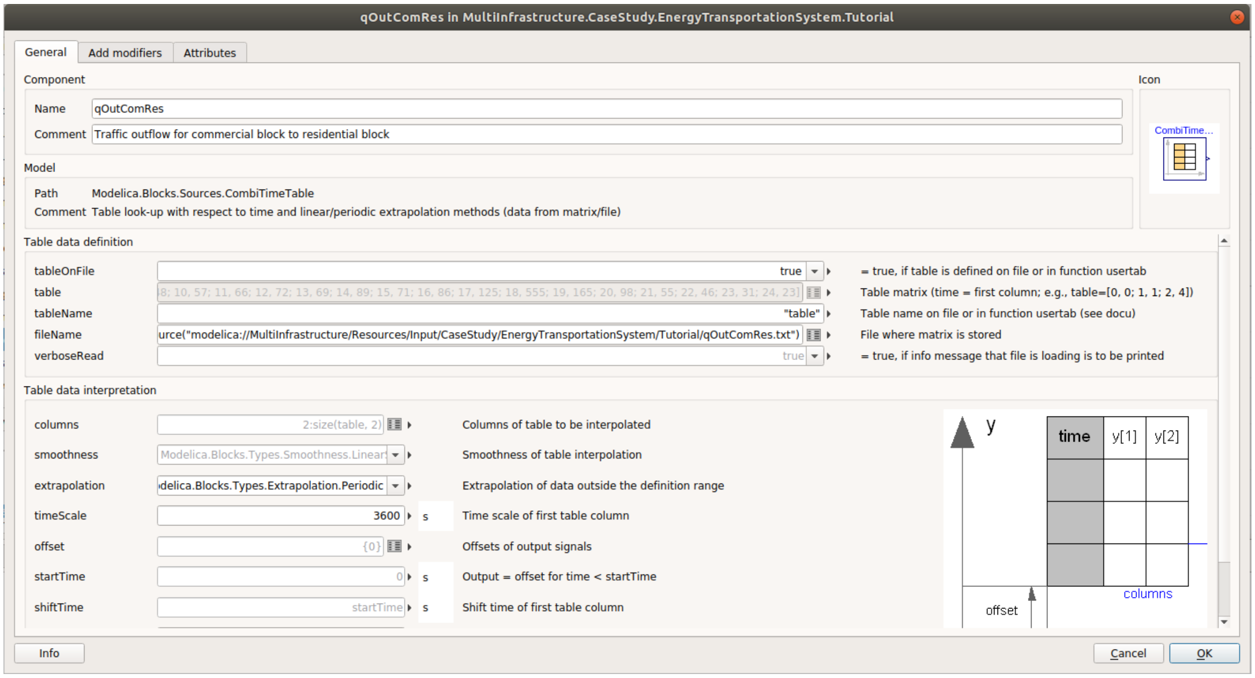


Now connect qOutResCom to the third input of resBlo. When the Create Connection box comes up again, also put 1 in each option as shown below.



15. Create qOutComRes and connect to comBlo.

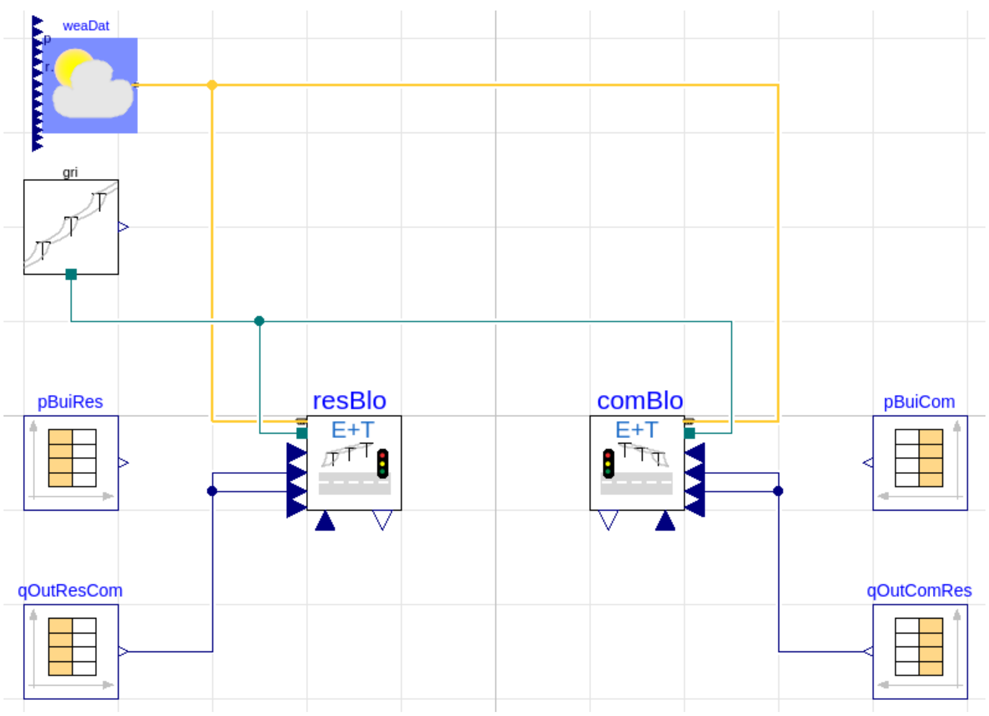
On that second instance of the CombiTimeTable, change the parameters to look like the following.



Import the traffic data from: Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>qOutComRes.txt.

Connect qOutComRes to the second and third inputs of comBlo, similar to what we did in the previous step. Fill in all of the Create Connection options with a 1.

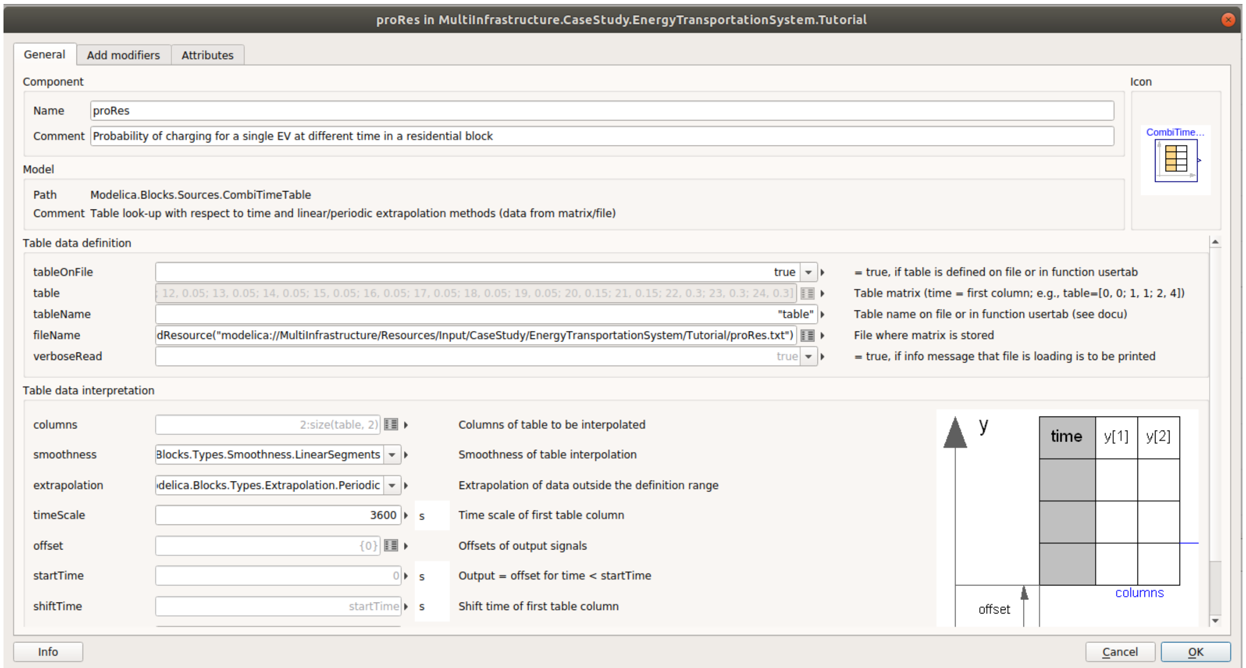
Take a brief moment to check that your model looks something like the following. It is ok if all of the connections or placements of the blocks are different, but they should all be there.



16. Add in the probability of an EV charging to both resBlo and comBlo.

Similar to the previous steps, we will now add in two more instances of CombiTimeTable, add a name, description, parameters, data, and then connect to the respective block.

For the first instance, change the parameters to match the following.

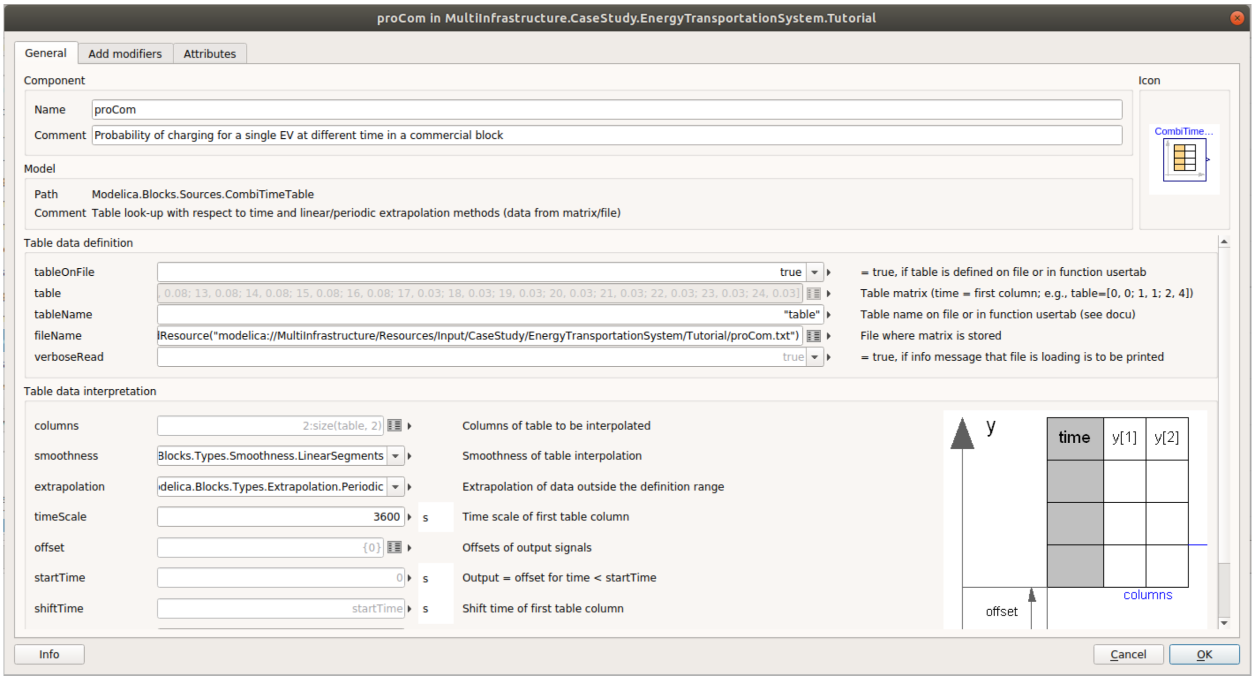


Add in the data from:
Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>proRes.txt.

Connect proRes to the fourth input on the left as shown below.

Put 1 in the Create Connection box that pops up.

For the second instance of the CombiTimeTable input the following parameters and copy the data from Resources>Scripts>CaseStudy>EnergyTransportationSystem>Tutorial>proCom.txt .

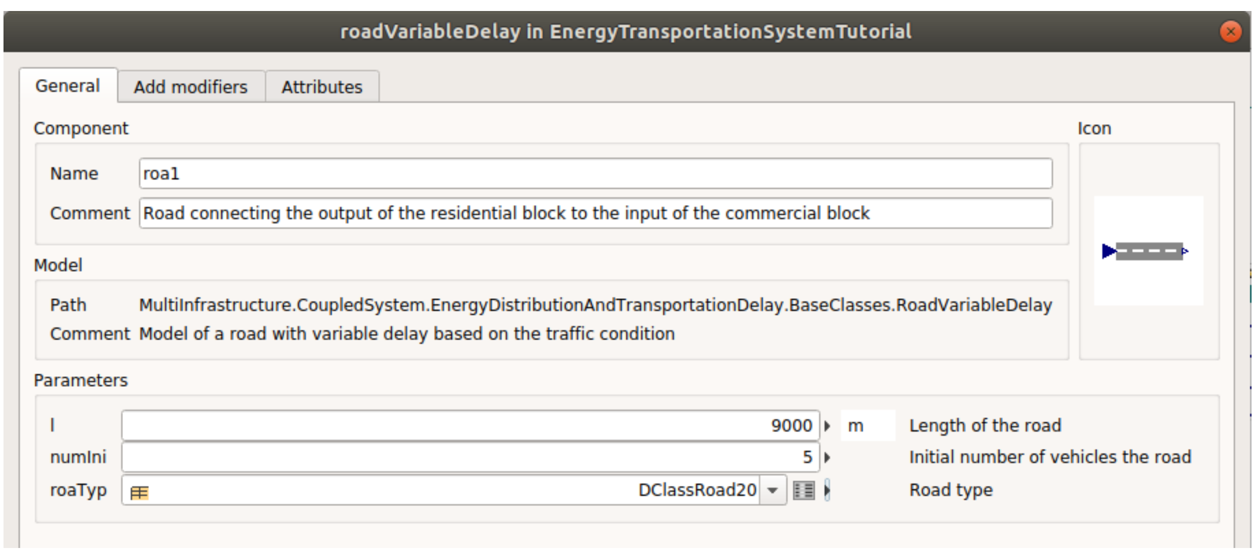


Connect proCom to the fourth input on comBlo and input 1 in the Create Connection box.

17. Add a road between the resBlo and the comBlo.

Add two instances of the road from:
MultiIinfrastructure.CoupledSystem.EnergyDistributionAndTransportationDelay.BaseClasses.RoadVariableDelay

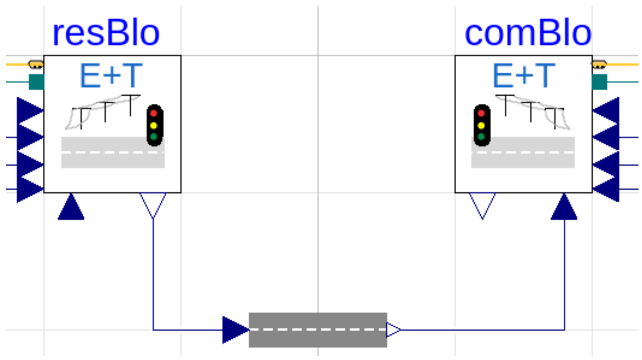
Change the parameters of one of the roads to look like the following.



From the drop down menu of roaTyp make sure to pick DClassRoad20.

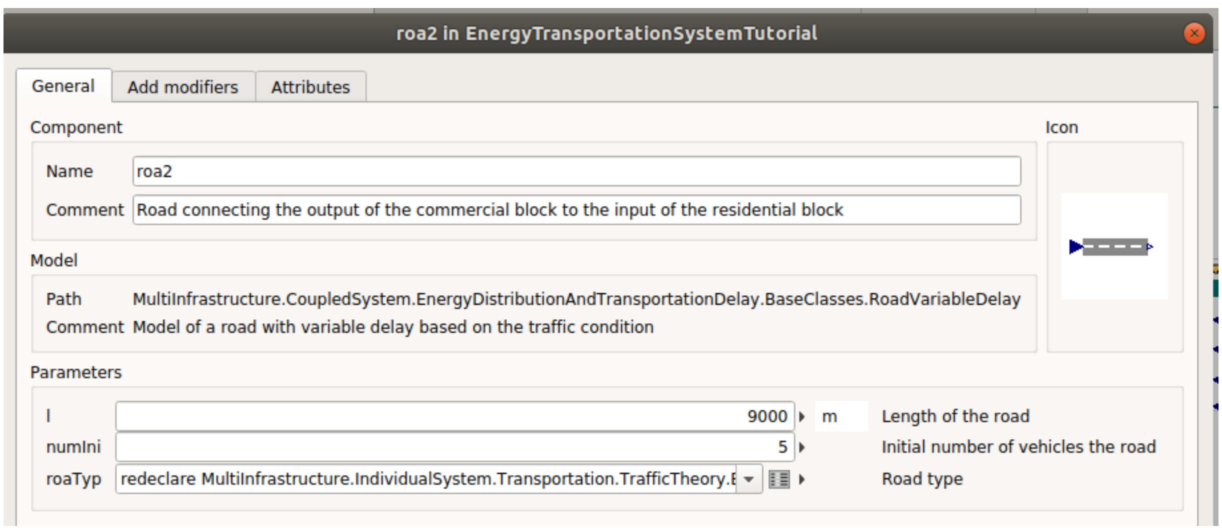
Connect the output of resBlo to the input of roa1. Enter 1 for the Create Connection box.

Connect the output of roa1 to the input of comBlo. Enter 1 for the Create Connection box. It should look something like this.



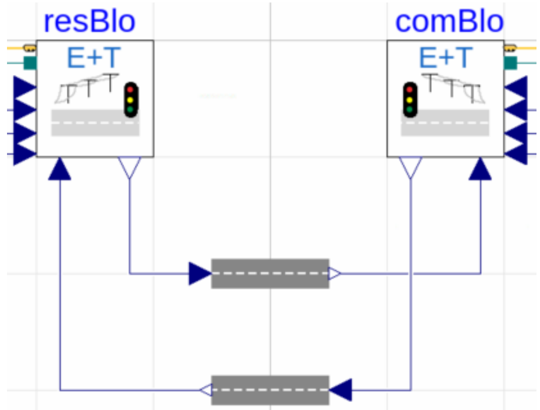
18. Add a road between comBlo and resBlo

For the second road instance, edit the parameters to look like the following.

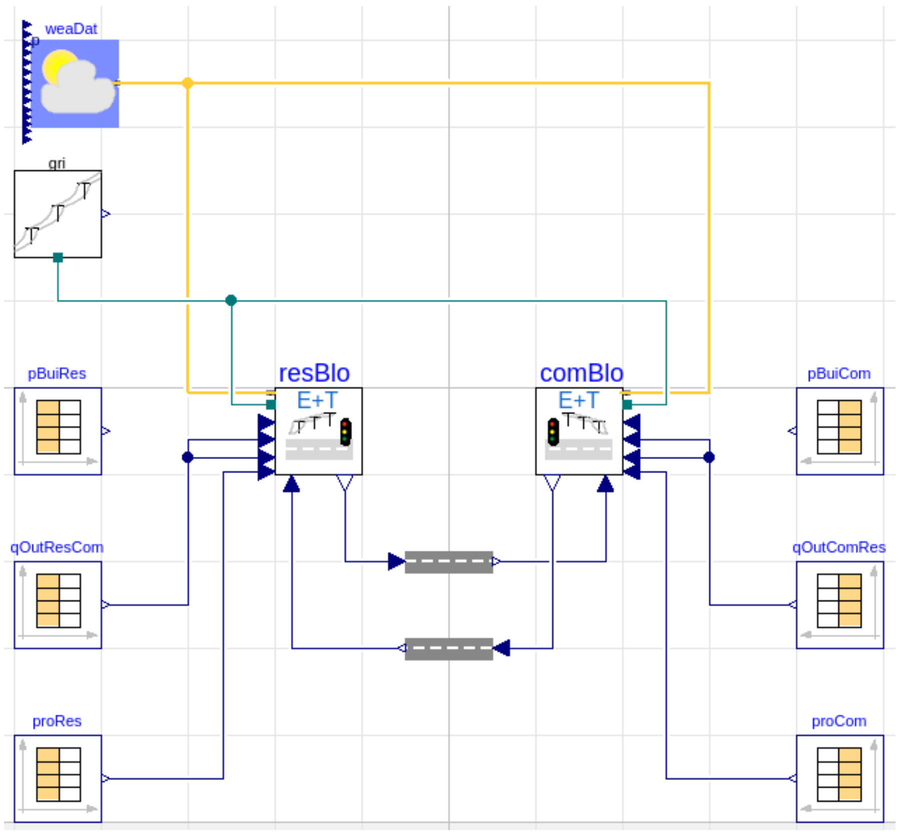


Again, choose RoadDClass20 for the roaTyp. Hit ok and connect the output of comBlo to the input of roa2. Enter 1 into the Create Connection box.

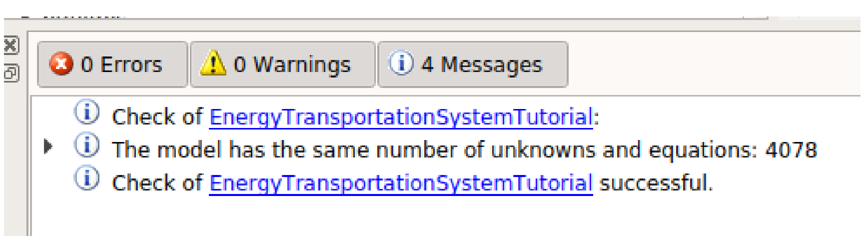
Connect the output of roa2 to the input of resBlo. Enter 1 into the Create Connection box. It should look like this now.



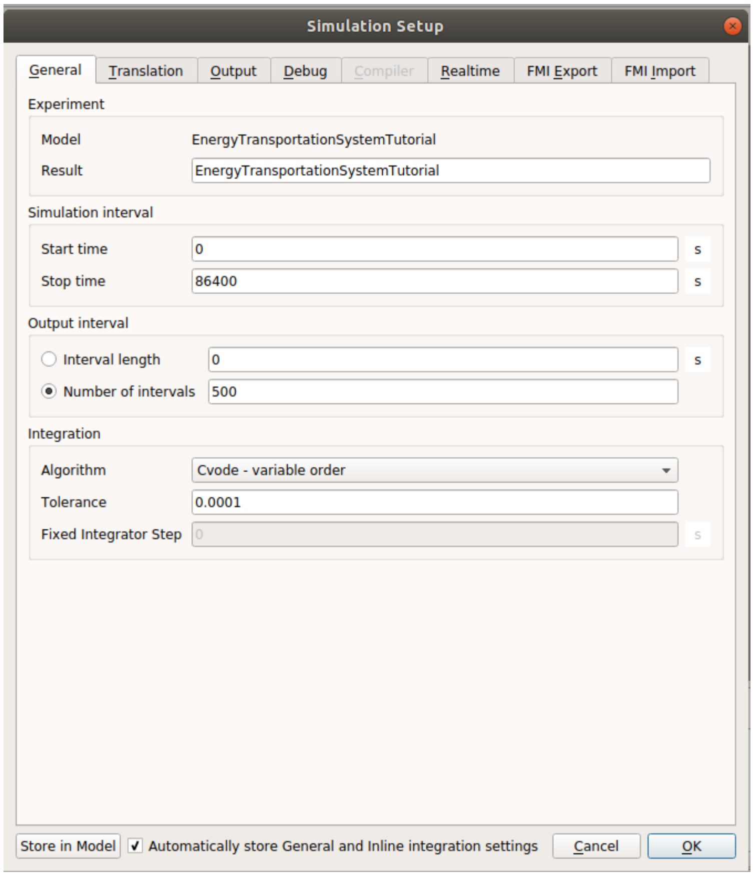
Overall your model should now look something like this.



19. Check the model: Click the checkmark at the top of the ribbon to compile the model. It should look like this.



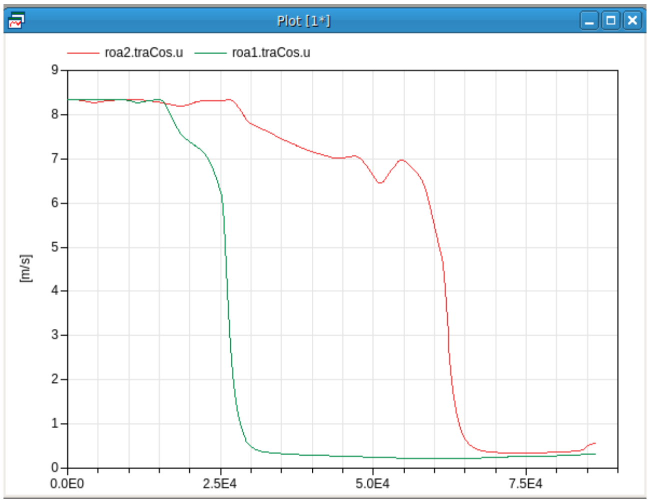
20. Simulate the model: Switch to simulation mode and enter in the following parameters.



Click Ok and click the Simulate button.

21. Evaluate results.

Here is one example of the results. The green and the red each represent the traffic flow on one of the roads.



There are many other interesting correlations to look at for the results. Take your time to explore. This concludes the second SCC tutorial.

Tutorial 2: Net-Zero Energy Community Library

This tutorial aims to provide a detailed step-by-step guide on building a heat pump system with control in Modelica. The following libraries will be used in this tutorial:

Modelica Buildings Library (version 5.0.1)[3]
Net-Zero Energy Community Library (version 1.1)[4].

To implement the proposed heat pump system, three examples will be introduced: (1) RC room model with heat transfer through the envelope; (2) room model coupled with a heat pump system; (3) heat pump system model with speed control. The latter examples are built upon the former examples. After this tutorial, the users will be more familiar with the modeling of energy systems coupled with room models as well as the control implementations. The remainder of this tutorial introduces the three examples step-by-step. Dymola 2020 in Windows system was used to create this tutorial.

Figure of final system:

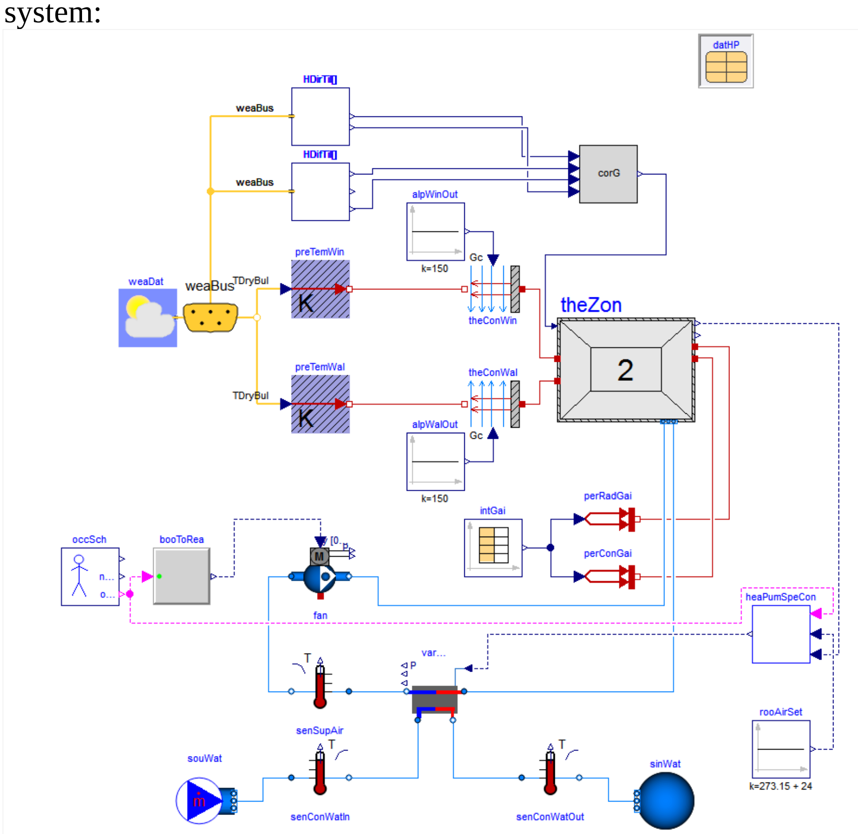


Figure 1 Modelica Diagram of the Heat Pump System with Control

[3] https://simulationresearch.lbl.gov/modelica/downloads/archive/modelica-buildings.html
[4] https://www.colorado.edu/lab/sbs/nzec-library

Example 1 – RC Room Model

This example builds an RC room model with heat convection and radiation from the room envelope (*Figure 2*).

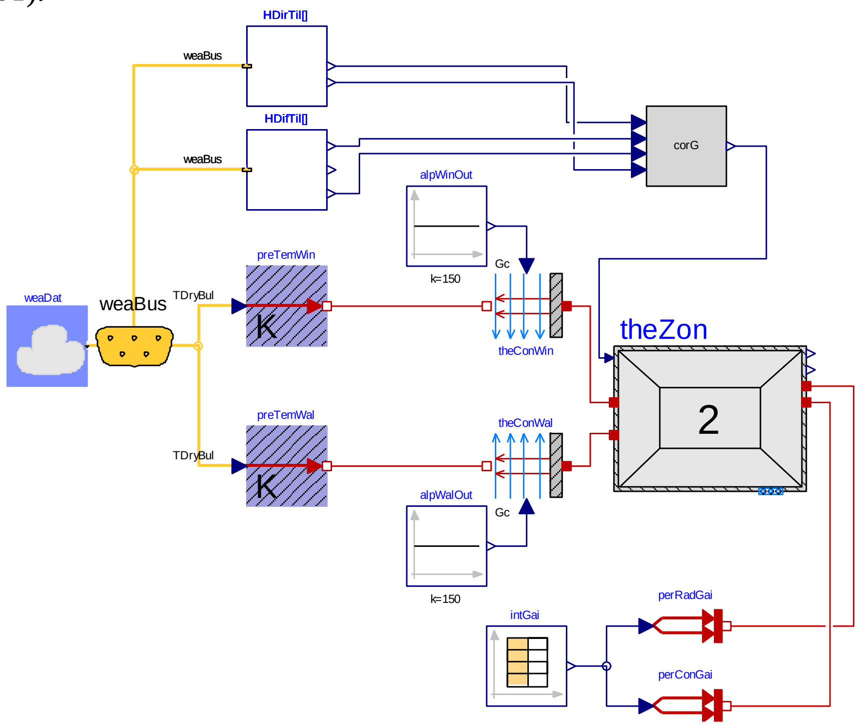


Figure 2 Modelica Diagram of Example 1: RC Room Model

This example is adapted from:
Buildings.ThermalZones.ReducedOrder.Examples.SimpleRoomTwoElements.

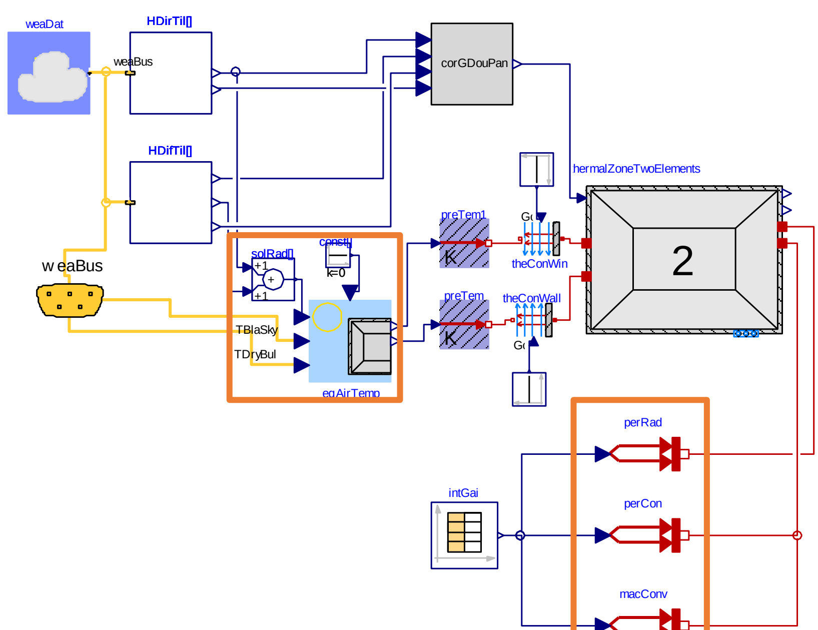


Figure 3 Modelica Diagram of Buildings.ThermalZones.ReducedOrder.Examples.SimpleRoomTwoElements

We reused this model with the following changes:

(1) Ignoring the computation of the equivalent air temperature and using drybulb temperature as the boundary condition;
(2) Ignoring the convective heat gain from machines.

The differences have been marked with orange rectangular in *Figure 3*. Additionally, we made the following changes to the parameters:

Name	Comment	Value
weaDat.filNam	Weather data	ModelicaServices.ExternalReferences.loadResource("modelica://GreenVillage/Resources/Weather/USA_FL_Tampa.Intl.AP.722110_TMY3.mos")
alpWinOut.k	Outdoor coefficient of heat transfer for windows	150
alpWalOut.k	Outdoor coefficient of heat transfer for walls	150
intGai.fileName	Internal heat gain	ModelicaServices.ExternalReferences.loadResource("modelica://GreenVillage/Resources/Input/Examples/HeatPump/InternalGain.txt")
intGai.columns	Columns of table to be interpolated	2:3
theZon.Medium	Medium of air	Buildings.Media.Air
theZon.VAir	Room air volume	rooLen*rooWid*rooHei
rooLen	Room length	8
rooWid	Room width	8
rooHei	Room height	10
theZon.nOrientations	Number of orientations	nOrientations
nOrientations	Number of orientations	2
theZon.AWin	Vector of areas of windows by orientations	{4,4}
theZon.ATransparent	Vector of areas of transparent elements by orientations	{4,4}
theZon.gWin	Total energy transmittance of windows	0.6
theZon.AExt	Vector of areas of exterior walls by orientations	{rooLen*rooHei,rooWid*rooHei}
theZon.RExt	Vector of resistance of exterior walls, from inside to outside	{0.33142}
theZon.AInt	Area of interior walls	2*rooHei*rooLen
theZon.RInt	Vector of resistance of	{0.0168895}

Name	Comment	Value
	interior walls, from port to center	

All values are in SI units. The parameters not mentioned are the same as in the Buildings library example.

With the following simulation setup, you can get the results shown in *Figure 5*.

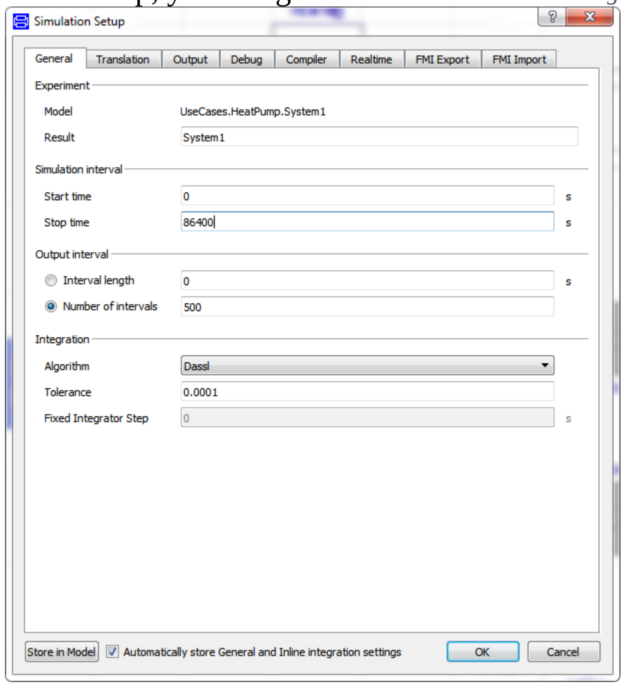


Figure 4 Simulation Setup for Example 1

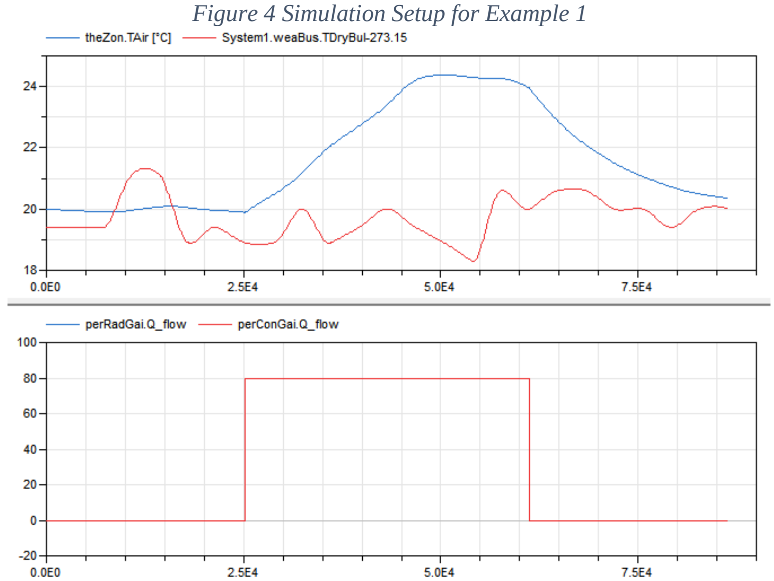


Figure 5 Simulation Results for Example 1

If you go to "Commands" and click "Simulate and Plot". This will call a script that we prepared and automatically simulate and plot the model with exactly the same setup above.

Example 2 – Heat Pump Model

This example adds a heat pump model on top of the RC room model we just built . The water cooled variable-speed heat pump is modeled as an instance of the class Buildings.Fluid.HeatExchangers.DXCoils.WaterCooled.VariableSpeed. The fan is modeled as an instance of the class Buildings.Fluid.Movers.SpeedControlled_y.

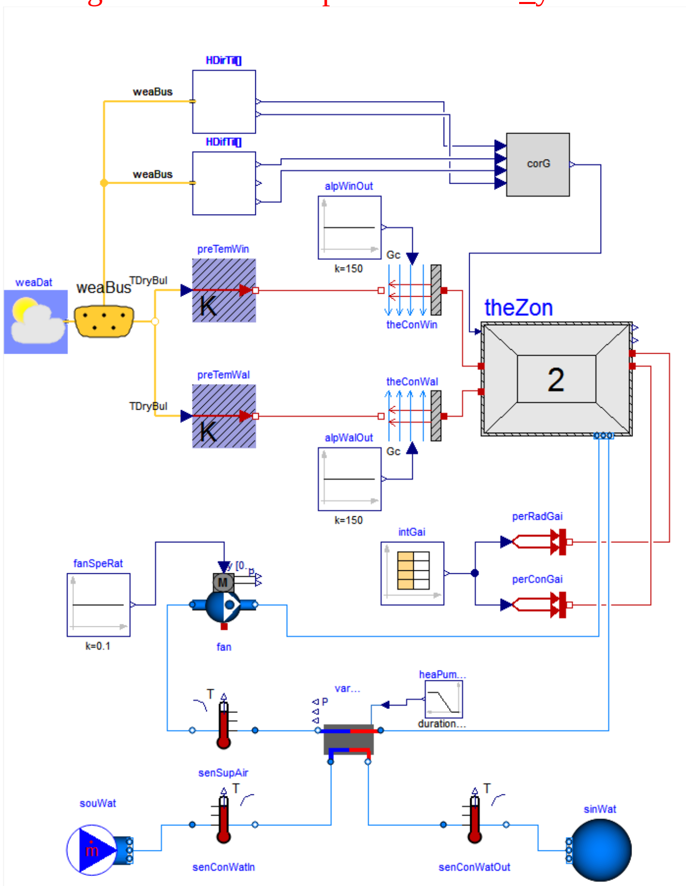


Figure 6 Modelica Diagram of Example 2: RC Room Model Coupled with Heat Pump

1. To define the condenser-side medium as water, we added the declaration in **Text** mode:

```
replaceable package Medium2 = Buildings.Media.Water
  "Medium model at the condenser side";
```

This will allow the users to change the medium to other refrigerants later. Then, we also define the system-level parameters that are related to the heat pump. Declaring parameters at the top-level allows to propagate them to other models, and to easily change them at one location when revising the model.

```
  replaceable parameter GreenVillage.Subsystems.HeatPump.WSHPPhysics.Data.HeatPump1 datHP
    "Performance data for heat pump";
parameter Boolean computeReevaporation=true
"Set to true to compute reevaporation of water that accumulated on coil";
parameter Real minSpeRat= 0.05 "Minimum speed ratio";
parameter Modelica.SIunits.PressureDifference dpEva_nominal=1000
"Pressure difference over evaporator at nominal flow rate";
parameter Modelica.SIunits.PressureDifference dpCon_nominal=40000
"Pressure difference over condenser at nominal flow rate";
```

Now, we instantiate Buildings.Fluid.HeatExchangers.DXCoils.WaterCooled.VariableSpeed and name it as *varSpeHeaPum*. Redeclare the heat pump parameters using the system-level parameters.

```
// Heat pump related parameters
Buildings.Fluid.HeatExchangers.DXCoils.WaterCooled.VariableSpeed varSpeHeaPum(
redeclare final package MediumEva = Medium1,
redeclare package MediumCon = Medium2,
dpEva_nominal=dpEva_nominal,
dpCon_nominal=dpCon_nominal,
computeReevaporation=computeReevaporation,
minSpeRat=minSpeRat,
datCoi=datHP)
    "Variable speed heat pump";
```

2. To model the fan, we instantiate Buildings.Fluid.Movers.SpeedControlled_y and name it as *fan*. Then, we also define the fan related system-level parameters and redeclare the corresponding fan parameters using these system-level parameters.

```
  // fan related parameters
  replaceable parameter GreenVillage.Subsystems.HeatPump.WSHPPhysics.Data.FanHP1 perFan "Pe
rformance data for fan";
  parameter Boolean addPowerToMedium=false
    "Set to false to avoid any power (=heat and flow work) being added to medium (may give
simpler equations)";
Buildings.Fluid.Movers.SpeedControlled_y fan(
redeclare package Medium = Medium1, addPowerToMedium=addPowerToMedium,
per=perFan.per);
```

3. To monitor the supply air temperature and the condenser inlet/outlet water temperature, we instantiate Buildings.Fluid.Sensors.TemperatureTwoPort three times and name them as *senSupAir*, *senConWatIn* and *senConWatOut*, respectively. Define the sensor related system-level parameters and apply them to all three sensor models.

```
  // sensor related parameters
  parameter Modelica.SIunits.MassFlowRate m_flow_nominal = datHP.sta[datHP.nSta].nomVal.mCo
n_flow_nominal
    "Nominal mass flow rate";
  Buildings.Fluid.Sensors.TemperatureTwoPort senSupAir(redeclare final package
      Medium = Medium1, final m_flow_nominal=m_flow_nominal)
    "Supply air temperature sensor";
  Buildings.Fluid.Sensors.TemperatureTwoPort senConWatIn(redeclare final
package Medium = Medium2, final m_flow_nominal=m_flow_nominal)
    "Inlet condenser water temperature";
  Buildings.Fluid.Sensors.TemperatureTwoPort senConWatOut(redeclare final
package Medium = Medium2, final m_flow_nominal=m_flow_nominal)
"Inlet condenser water temperature";
```

4. For the model to run, we still need a water source and a water sink, which are instances of Buildings.Fluid.Sources.MassFlowSource_T and Buildings.Fluid.Sources.Boundary_pT. We name them as *souWat* and *sinWat*, respectively. Configure the two models using the following parameters in **Text** mode.

```
  Buildings.Fluid.Sources.MassFlowSource_T souWat(
    redeclare package Medium = Medium2,
    use_T_in=false,
    use_m_flow_in=false,
    m_flow=m_flow_nominal,
    T=298.15,
    nPorts=1) "Source on water side";
  Buildings.Fluid.Sources.Boundary_pT sinWat(
    redeclare package Medium = Medium2,
    nPorts=1,
    p(displayUnit="Pa")) "Sink on water side";
```

Connect the water loop and air loop. Positive design flow direction is from port_a to port_b.

5. Lastly, we need to provide inputs for heat pump speed ratio as well as the fan speed ratio. Here, we use the instance *fanSpeRat* of Modelica.Blocks.Sources.Constant to model a constant fan speed ratio. Similarly, we use the instance *heaPumSpeRat* of Modelica.Blocks.Sources.Ramp to model a ramped change in the heat pump speed ratio. The following screenshots show the parameter configuration of the two inputs.

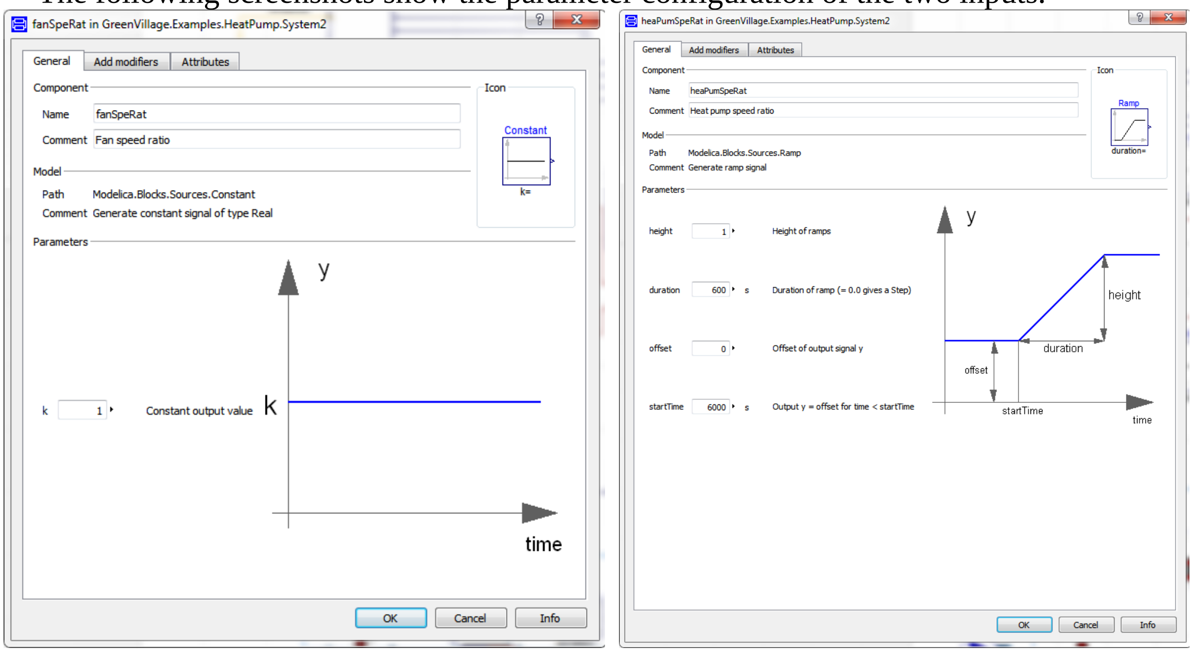
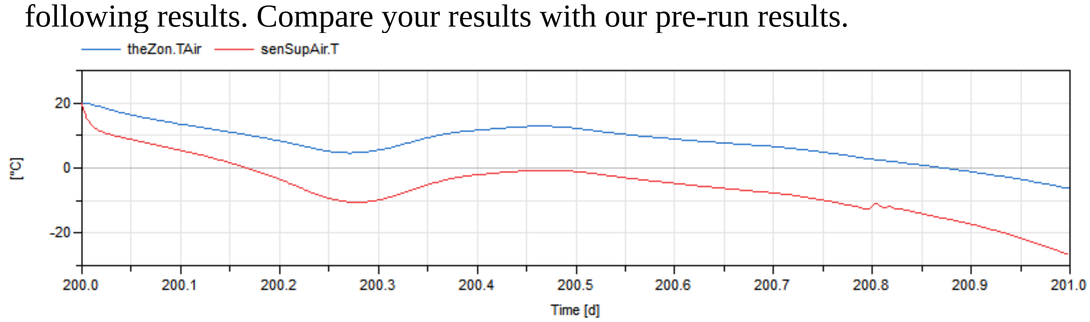
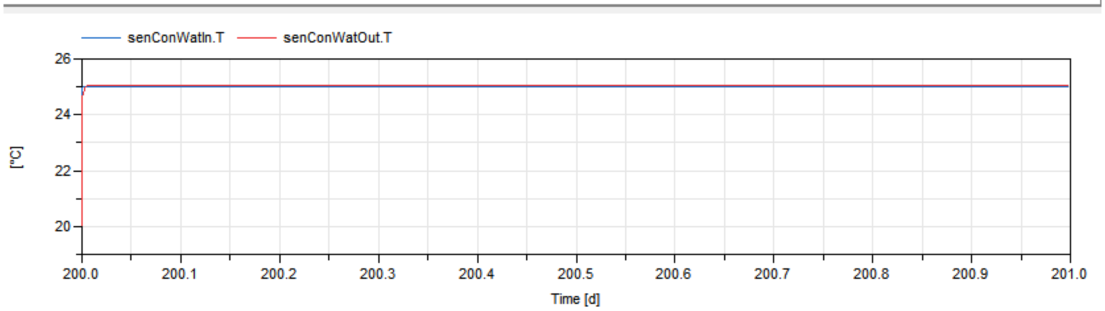


Figure 7 Parameters for fanSpeRat and heaPumSpeRat

6. Run the simulation from Day 200 to Day 201 (summer condition). You will see the following results. Compare your results with our pre-run results.





Parameters (use default value if parameter is not listed in table)

Name	Comment	Value
varSpeHeaPum.MediumEva	Heat pump evaporation side medium	Buildings.Media.Air
varSpeHeaPum.MediumCon	Heat pump condensation side medium	Buildings.Media.Water
varSpeHeaPum.dpEva_nominal	Heat pump evaporation side pressure drop	1000
varSpeHeaPum.dpCon_nominal	Heat pump condensation side pressure drop	40000
varSpeHeaPum.computeReevaporation	Whether to compute reevaporation in heat pump	true
varSpeHeaPum.minSpeRat	Minimum speed ratio of heat pump	0.05
varSpeHeaPum.datCoi	Heat pump data	datHP
datHP	Heat pump data	replaceable parameter GreenVillage.Subsystems.HeatPump.WSHPPhysics.Data.HeatPump1
fan.Medium	Fan medium	Buildings.Media.Air
fan.addPowerToMedium	Whether to add power to medium in fan simulation	false
fan.per	Fan performance data	perFan.per
perFan	Fan performance data	replaceable parameter GreenVillage.Subsystems.HeatPump.WSHPPhysics.Data.FanHP1
senSupAir.Medium	Supply air temperature sensor medium	Buildings.Media.Air
senSupAir.m_flow_nominal	Supply air temperature sensor mass flow rate	m_flow_nominal
senConWatIn.Medium	Water inlet temperature sensor medium	Buildings.Media.Water
senConWatIn.m_flow_nominal	Water inlet temperature sensor mass flow rate	m_flow_nominal
senConWatOut.Medium	Water outlet temperature sensor medium	Buildings.Media.Water
senConWatOut.m_flow_nominal	Water outlet temperature sensor mass flow rate	m_flow_nominal
m_flow_nominal	System mass flow rate	datHP.sta[datHP.nSta].nomVal.mCon_flow_nominal
souWat.Medium	Water source medium	Buildings.Media.Water

Name	Comment	Value
souWat.use_T_in	Whether to use temperature input in water source	false
souWat.use_m_flow_in	Whether to use mass flow rate input in water source	false
souWat.m_flow	Water source mass flow rate	m_flow_nominal
souWat.T	Water source temperature	298.15
sinWat.Medium	Water sink medium	Buildings.Media.Water
heaPumSpeRat.duration	Heat pump speed ratio input duration	600
heaPumSpeRat.startTime	Heat pump speed ratio input start time	6000
heaPumSpeRat.height	Heat pump speed ratio input height	0.1
heaPumSpeRat.offset	Heat pump speed ratio input offset	0
fanSpeRat.k	Fan speed ratio input value	0.1

Example 3 – Heat Pump System with Control

This example adds fan on/off control and heat pump speed control on top of Example 2. The control logic we will implement here is as follows.

1. The fan and heat pump will only be on when the room is occupied.
2. The heat pump speed ratio is controlled with a PI controller based on the room air temperature setpoint and the measured room air temperature.
3. The room air setpoint will be constant for simplicity reasons.

The final diagram of the model will look like the following figure.

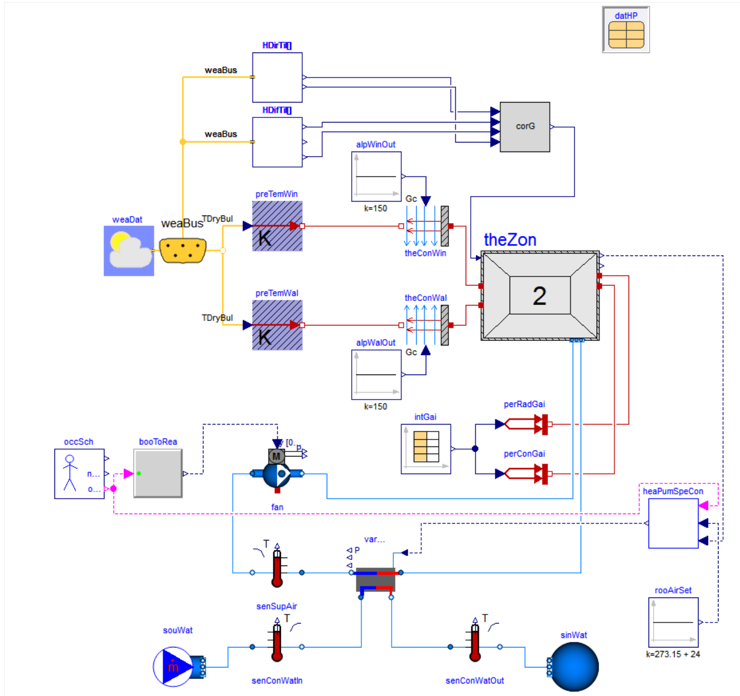


Figure 8 Modelica Diagram of Example 3: Heat Pump System with Control

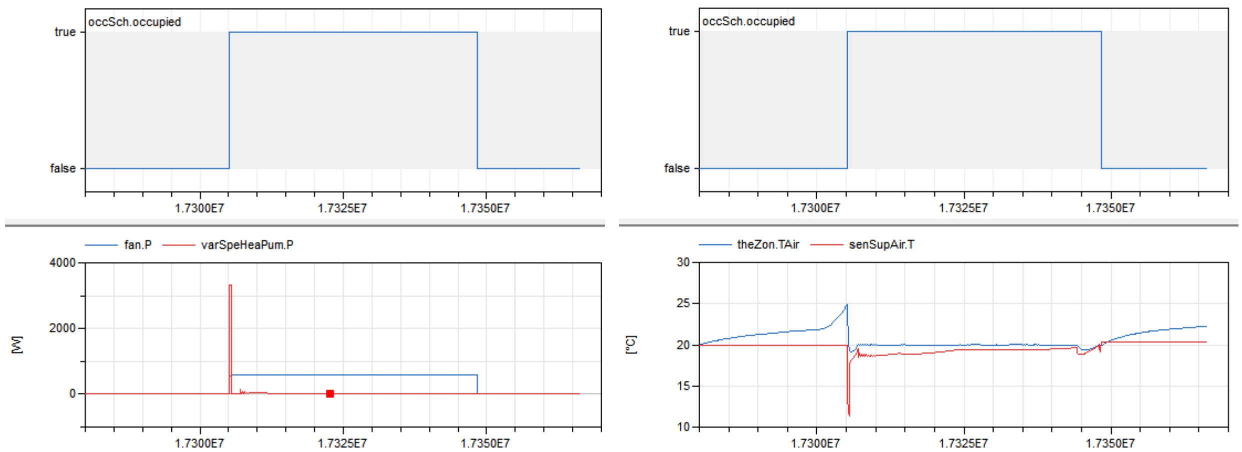
Step 1: We model the occupancy schedule *occSch* as an instance of Buildings.Controls.SetPoints.OccupancySchedule. Because the first entry of the occupancy indicates a change from unoccupied to occupied, we will choose true for the *firstEntryOccupied*.

Here, we also need to use an instance *booToRea* of Buildings.Controls.OBC.CDL.Conversions.BooleanToReal to convert the occupancy from boolean signal to real signal. Connect *booToRea.y* to *fan.y*.

Step 2: The control block we will use here is an instance of GreenVillage.Subsystems.HeatPump.WSHPPhysics.BaseModels.FanSpeedControl. This controller can do P, PI, PD, and PID control as the user prefers. Choose the *controllerType* as PI controller and set *Ti* to 60 s, *reverseAction* to true for cooling.

Step 3: We model the constant room air setpoint as an instance of Buildings.Controls.OBC.CDL.Continuous.Sources.Constant. Set the value of k to 273.15+20 to represent a constant room air temperature setpoint at 20 °C.

Step 4: Make the rest connections as indicated in *Figure 8*. Run the simulation from Day 200 to 201 with default settings and compare the results to our pre-run results by clicking "Commands-Simulate and Plot". The pre-run results will look like in the following figures.



Parameters (use default value if parameter is not listed in table)

Name	Comment	Value
occSch.firstEntryOccupied	Set to false as first entry denotes a change from occupied to unoccupied.	true
heaPumSpeCon.controllerType	Type of controller	Modelica.Blocks.Types.SimpleController.PI
heaPumSpeCon.Ti	Time constant of Integrator block	60 s
heaPumSpeCon.reverseAction	Set to true for cooling	true
rooAirSet.k	Constant output value	273.15+20

Acknowledgement

This workshop is supported by the following projects:

- U.S. Department of Energy, Energy Efficiency and Renewable Energy, Building Technologies Office, under Contract No. DE-AC05-76RL01830.
- National Science Foundation under Award IIS-1802017.

Special thanks to Dassault Systèmes and Barcroft Technology for providing free Dymola licenses during the workshop!