OpenModelica 1.14.1 (Official Release) – Ubuntu 18.04 LTS
Manual de Instalação e Guia de Utilização

Lucas Gabriel Cosmo Morais
Maio, 2020.

OpenModelica é um ambiente open-source de modelagem e simulação baseado na linguagem Modelica, para pesquisa, ensino e uso industrial. Deve-se atentar que há distinção entre o tipo de linguagem e o ambiente utilizado. Existe outros ambientes, tais como: SimulationX, Dymola e WolframSystemModeler que utilizam o mesmo tipo de linguagem. Mais informações sobre a linguagem em: https://www.modelica.org/.

A versão utilizada do OpenModelica durante o desenvolvimento do projeto foi a 1.14.1 Release, outras versões estão disponíveis com mais atualizações e em desenvolvimento contínuo, bem como para outros sistemas operacionais. Pode-se encontrar na aba de downloads da página: https://openmodelica.org/

Índice

Instalação do OpenModelica 1.14.1....2
Guia de Utilização Rápida do OMEdit....3
1. Interface e Workflow....3
2. Adicionando um modelo....5
3. Rodando uma simulação (configuração de parâmetros)....5
4. Visualizando variáveis....6
5. Mais detalhes do OpenModelica....6
6. Biblioteca MODEST....7
7. Simulação em Tempo Real (de uma Esteira)....8
Guia de Utilização Rápida do OMShell....10
Exemplo de uma simulação com OMShell (Motor DC)....11

Instalação do OpenModelica 1.14.1

O OpenModelica é disponibilizado em pacotes .deb compilados em Ubuntu e Debian: **jessie, stretch, xenial e bionic.** Para instalar o OpenModelica 1.14.1 no Ubuntu 18.04 LTS vá até o link: **https://openmodelica.org/download/download-linux** onde encontram-se as versões.

1 – Passo: Copie o comando a seguir e cole em um novo terminal:

```
echo "$deb http://build.openmodelica.org/apt `lsb_release -cs` release"
```

2 – Passo: Em seguida, copie e cole o comando a seguir e dê enter:

```
sudo tee /etc/apt/sources.list.d/openmodelica.list
```

3 – Passo: Digite “deb”, copie e cole o resultado do “**1 – Passo”** e dê enter. Fica assim:

```
deb http://build.openmodelica.org/apt bionic release
```

4 – Passo: Agora digite Crtl + C e digite os comandos a seguir:

```
sudo apt update
sudo apt install openmodelica
```

5 – Passo: Após algumas confirmações de instalação o compilador de linguagem Modelica, o editor OMEdit, OMShell, OMNotebook e outros recursos estarão disponíveis para serem utilizados. Para acessar o OMedit procure-o digitando na barra de procura de aplicativos do Ubuntu.

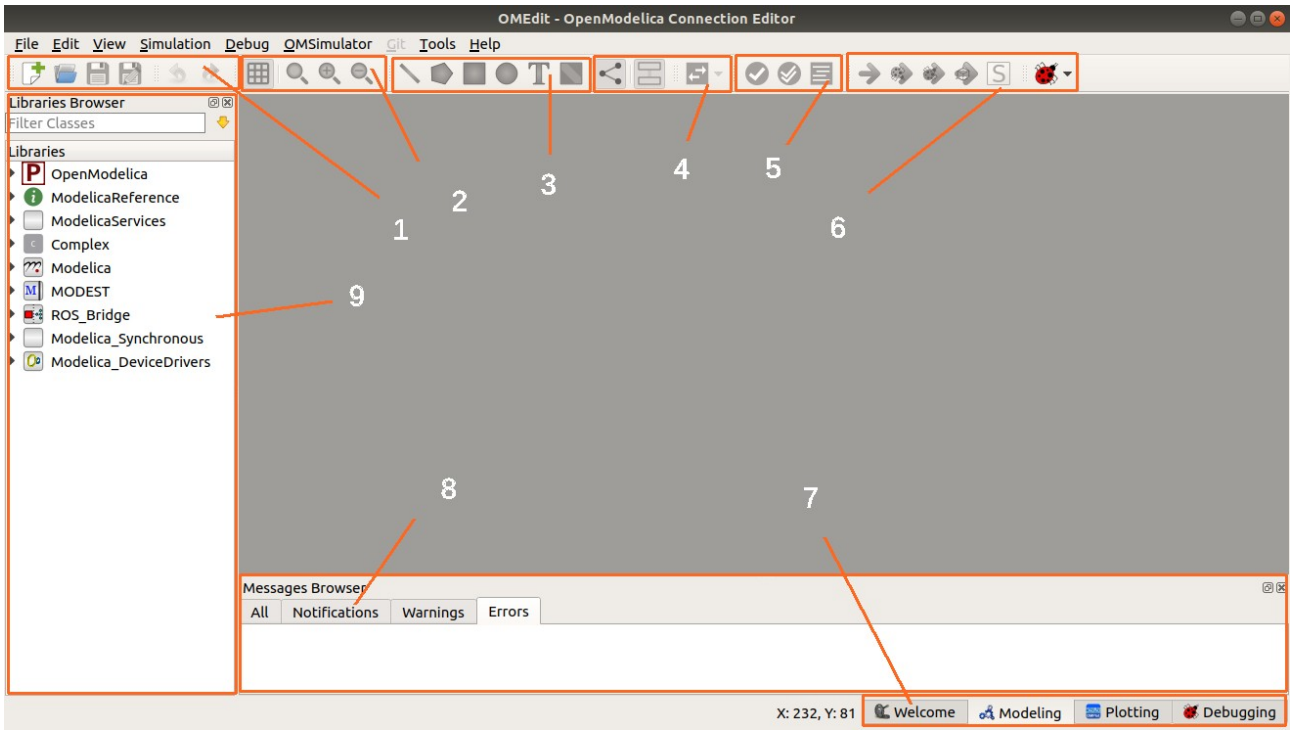
Problemas com a Instalação
Se ocorrer algum problema apague o arquivo da lista de fontes do Ubuntu.

1 – Passo: Navegue pelo terminal até a pasta raiz **/etc/apt/sources.list.d**
2 – Passo: Visualize os documentos da pasta com **ls -l**
3 – Passo: Apague o arquivo **openmodelica.list** digitando **sudo rm openmodelica.list**
4 – Passo: Reinicie o processo de instalação. O arquivo openmodelica.list correto deve conter o que está escrito no passo 3 (da instalação).

Guia de Utilização Rápida do OMEdit

1. Interface e Workflow

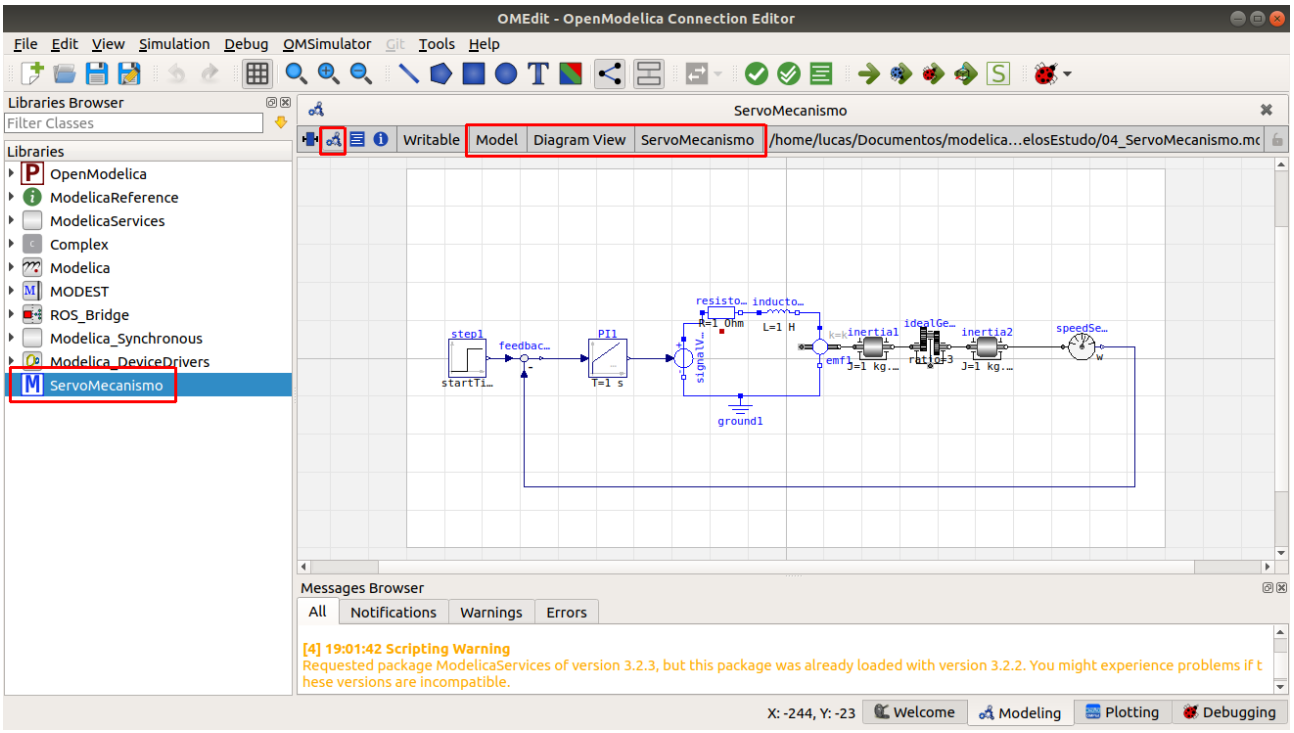
A interface do OMEdit se parece com o mostrado na figura a seguir, onde pode-se evidenciar que há no geral perspectivas (7) para se trabalhar, configurar os parâmetros das simulações (6), verificar messagens de erros ou warnings (8) e observar e plotar as variáveis simuladas (7).



1 – Criação de novas classes, abetura de modelos e bibliotecas, opções de salvar.
2 – Visualização de malha quadriculada, Zoom In e Zoom Out (Diagram View).
3 – Inserção de formas, desenhos, textos e imagens nas classes (Icon View).
4 – Alternar entre classes e modelos, modo de seleção.
5 – Checagem de equacionamento de modelo e instanciação.
6 – Opções de simulação e configuração da simulação.
7 – Perspectivas: Área de boas vindas, Modelgame, Plotagem e Debugging.
8 – Mensagens de erro, notificação e warnings.
9 – Modelos e Bibliotecas disponíveis (carregadas para o OpenModelica).

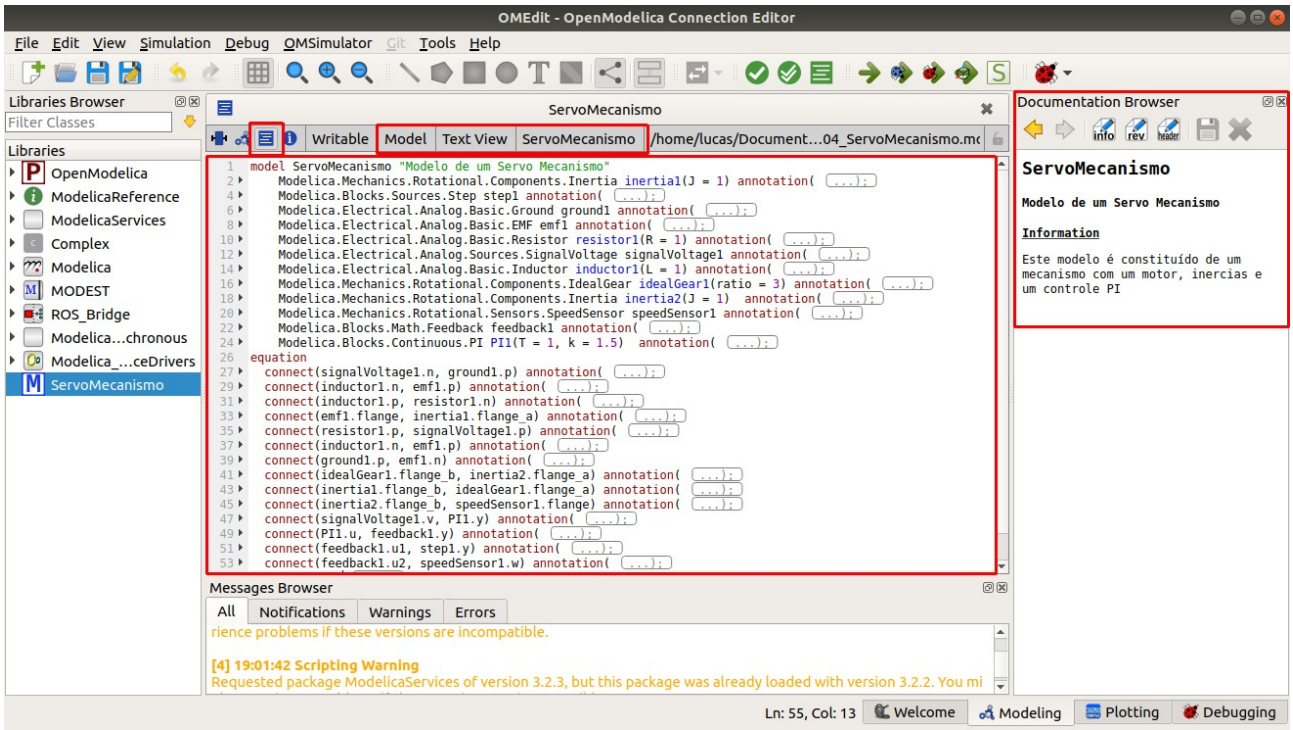
Um **workflow básico** pode ser descrito como: Criar ou carregar novas classes em linguagem Modelica, arrastar e soltar objetos dentro do Diagram View (ou escrever código no Text View), configurar os parâmetros de simulação (Simulation Setup), Simular o modelo e observar os resultados na perspectiva Plotting.

É possível trabalhar com diferentes visualizações de uma mesma classe (ou modelo). Com o exemplo a seguir observa-se o modelo (Diagram View) de um sistema de servo-mecanismo.



Do modelo do servo-mecanismo (da imagem anterior) observa-se que cada objeto e suas conexões estão descritas em linguagem Modelica e podem ser acessados pela visualização Text View. As informações disponibilizadas sobre o modelo ficam disponíveis no Documentation View.

A seleção de Icon, Diagram, Text e Documentation View pode ser feita alternando entre os 4 ícones na borda superior à esquerda.



2. Adicionando um modelo

Pode-se adicionar (carregar para o ambiente OpenModelica) modelos ou bibliotecas indo até o ícone “Pasta” no canto superior esquerdo ou digitando Crtl + O. Entre os arquivos do seu computador, busque por aqueles com extensão .mo, por exemplo: servomecanismo.mo

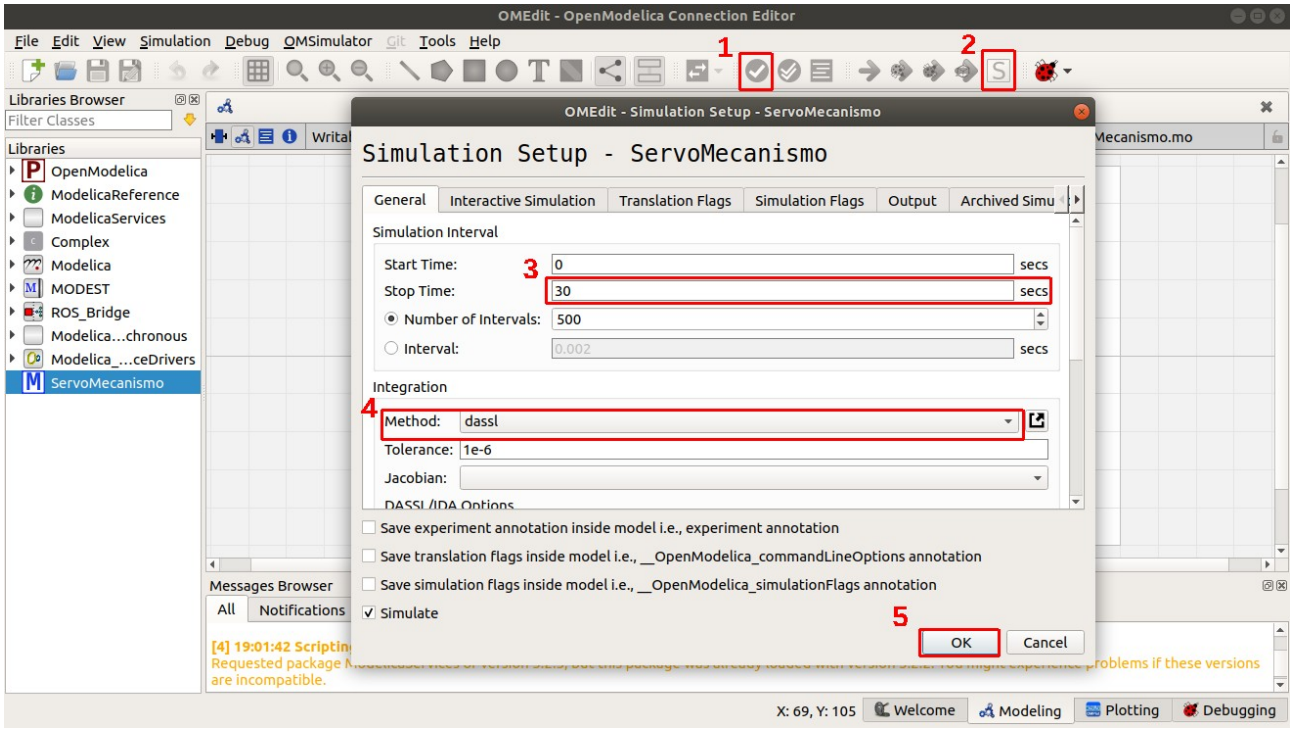
Bibliotecas são adicionadas indo até a pasta da biblioteca e adicionando o arquivo “package.mo” da mesma.

Observação: O modelo do Servo-Mecanismo é desenvolvido em um tutorial de outro ambiente - WolframSystemModeler (https://reference.wolfram.com/system-modeler/GettingStarted/MultidomainAServoMechanism.html) e pode ser montado a partir da biblioteca padrão do Modelica.

3. Rodando uma simulação (configuração de parâmetros)

Antes de rodar uma simulação **você deve checar se o número de equações que descrevem o modelo é o mesmo do número de variáveis**. Cheque isto clicando no ícone “check model”. Modelos desenvolvidos pelo próprio usuário podem conter menos equações que o número de variáveis. (Se isto ocorrer cheque se todas as conexões do diagrama estão certas. Se sim e o problema persistir, cheque o equacionamento matemático do modelo ou blocos utilizados no modelo).

Após isto, para configurar a simulação clique no ícone em formato de S, *Simulation Setup*. Digite *30*, para 30 segundos de simulação. No ítem 4 da imagem a seguir pode-se deixar o **método de integração da simulação** com “**dassl**” (Método padrão do OpenModelica). Depois é só clicar em “Ok” e esperar a compilação e simulação do modelo.

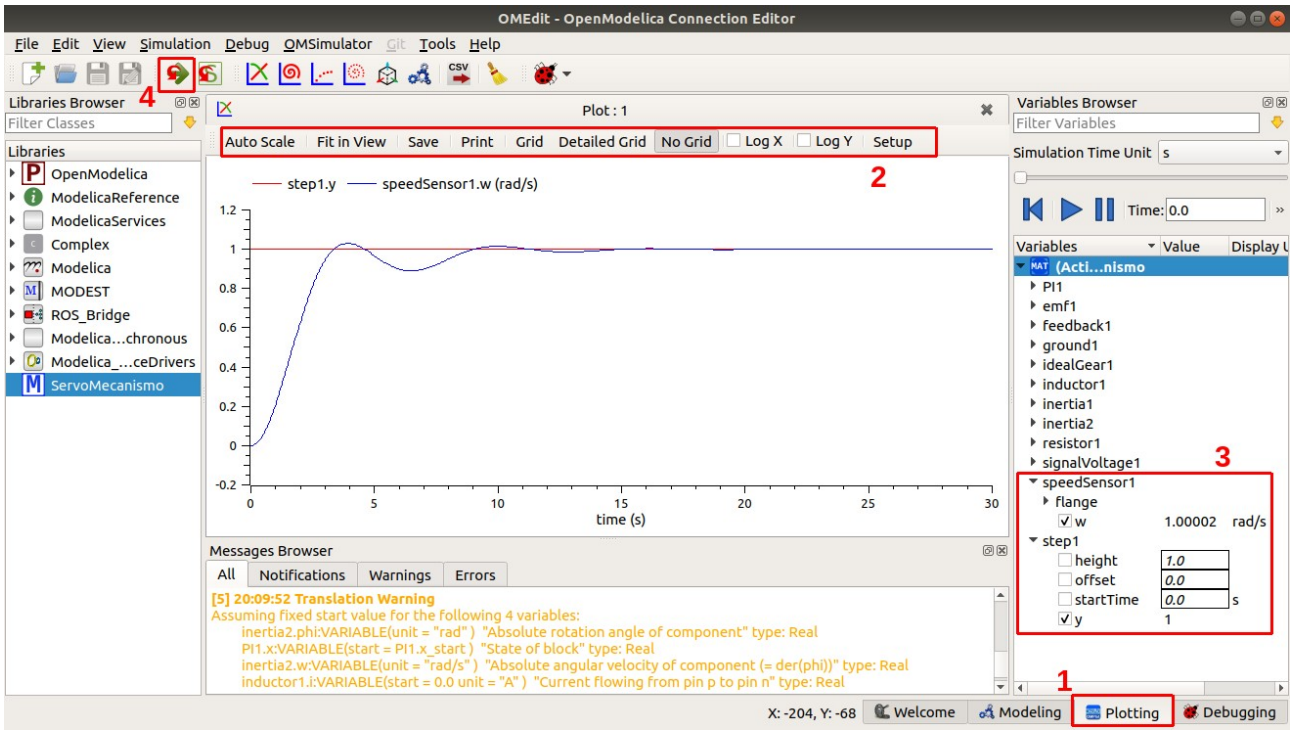


4. Visualizando variáveis

Geralmente pós simulação a perspectiva muda para *plotting* (1). Nesta perspectiva selecione as variáveis que deseja obsevar como resultado da simulação. No exemplo a seguir foram obsevadas o setpoint *step1.y* e a resposta medida pelo sensor de velocidade *speedSensor1.flange.w*.

Na perspectiva *plotting* pode-se alterar as propriedades do plot na aba acima da imagem (2), escolhendo-se um grid para imagem, redimensionando a visualização, salvando ou selecionando plote logarítimico. As variáveis podem ser selecionadas na aba da esquerda (3).

Para mudar algum parâmetro e simular novamente de maneira rápida: Altere os parâmetros das variáveis (3) e clique em re-simulate. Por exemplo, em step1 mude para height igual a 2.0. Depois clique no ícone em (4) *re-simulate*.



5. Mais detalhes do OpenModelica

Mais detalhes sobre utilização dos recursos do OpenModelica como: flags de simulação, visualização de simulações 3D entre outros, podem ser obtidos no Manual do Usuário oficial que está disponibilizado na página: https://openmodelica.org/useresresources/userdocumentation

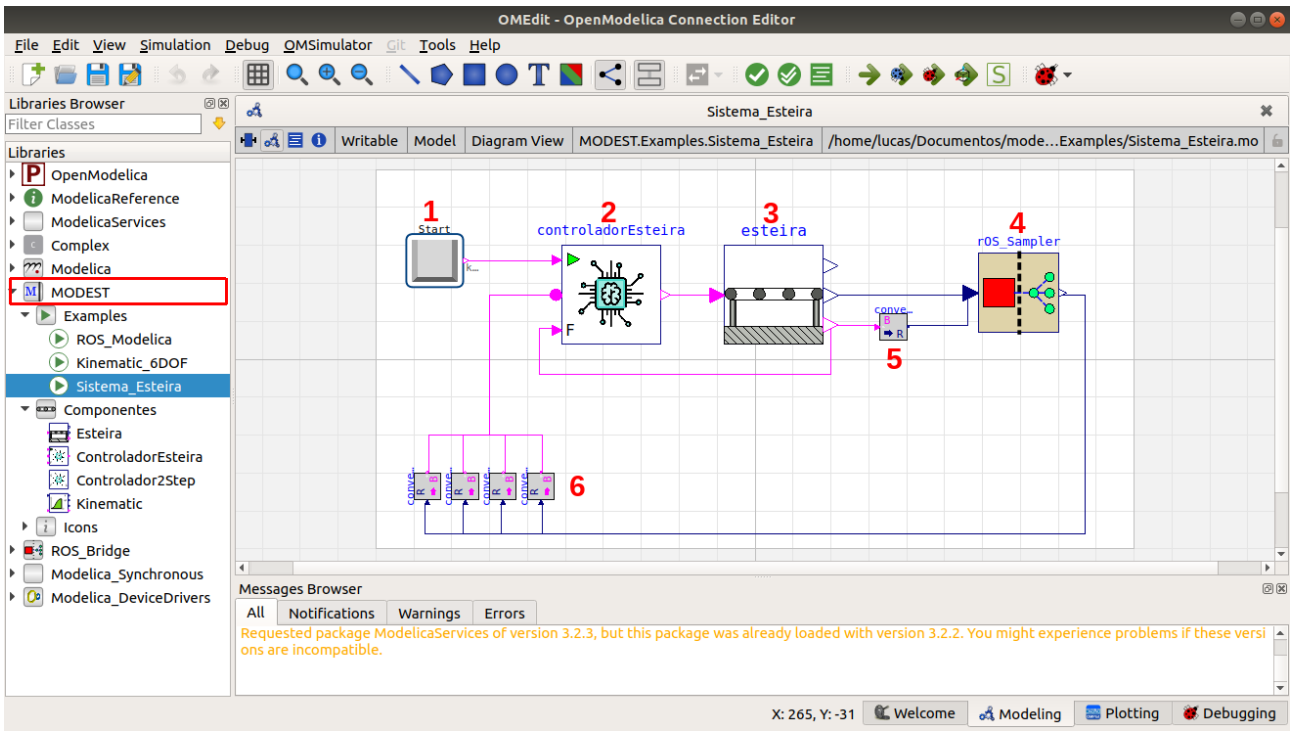
6. Biblioteca MODEST

MODEST é um pacote com equacionamento cinemático de uma esteira com comunicação via Robot Operation System – ROS. Foi desenvolvido com o intuito de modelar o comportamento de uma esteira para integração e representação em um modelo 3D.

Este pacote depende de outros 3 pacotes (ou bibliotecas) que devem ser baixados e adicionados ao ambiente para seu pleno funcionamento. A Biblioteca 4 é uma dependência para correto funcionamento da 3 (para algumas funções são requisitadas operações de sincronização e tempo).

1. **MODEST** (https://github.com/lucasgabriel11/MODEST)
2. ROS_Bridge (https://github.com/ModROS/ROS_Bridge)
3. Modelica_DeviceDrivers (https://github.com/modelica-3rdparty/Modelica_DeviceDrivers)
4. Modelica_Synchronous (https://github.com/modelica/Modelica_Synchronous)

Na imagem a seguir é possível visualizar o pacote MODEST já disponível para utilização. Este contém 3 pacotes: ***Examples, Componentes e Icons.*** Tem-se como componentes: ***Esteira, ControladorEsteira, Controlador2Step e Kinematic.*** Informações detalhadas sobre cada bloco estão disponibilizados em cada um destes (visualize Documentation View).



O modelo selecionado na imagem anterior (MODEST.Examples.Sistema_Esteira) retrata o objetivo principal no desenvolvimento do pacote: um ambiente, com uma esteira e um controlador associados, envia a velocidade da esteira enquanto recebe dados de sensores externos (que, neste caso, vêm do ambiente 3D - CoppeliaSim) via ROS.

1. **KeyboardKeyInput:** É dispositivo de entrada pelo usuário disponibilizado pela biblioteca *Modelica_DeviceDrivers*. Durante a simulação (em tempo real) o usuário aperta a tecla *S* para que a esteira dê inicio a sua movimentação.
2. **ControladorEsteira:** É o controlador (baseado em *stateGraphs*) que coordena as ações da esteira de acordo com os dados recebidos. Pode-se alterar o estado inicial começando a movimentação da esteira com a tecla *S* ou para a esteira de acordo com uma entrada *true* dos sensores externos.
3. **Esteira:** É a representação por meio de equacionamento cinemático do comportamento de uma esteira com ativação de movimento por meio de um *gatilho.* Descreve-se como parâmetros para a esteira: raio, comprimento, velocidade e aceleração máxima.
4. **ROS_Sampler:** É um amostrador que extrai dados da simulação e os publica para o Sistema ROS. Este bloco tem uma função externa associada a ele escrita em C (ROS_Socket_Call). A comunicação se dá por soquete TCP/IP com o servidor (um nó do sistema ROS que publica os dados no tópico /control_values). *ROS_Sampler* é disponibilizado pela Biblioteca *ROS_Bridge*. Mais informações em: http://wiki.ros.org/modelica_bridge
5. **BooleanToReal:** É um bloco da própria biblioteca padrão Modelica que converte Booleano para Real. (ROS_Sampler só receber Reais – Um Array de Reais).
6. **RealToBoolean:** É um bloco da própria biblioteca padrão Modelica que converte Real para Booleano. É necessário uma vez que a entrada de dados sensores no controlador da esteira é do tipo BooleanVector.

7. Simulação em Tempo Real (de uma Esteira)

Antes de rodar a simulação no OpenModelica **sempre inicie** o servidor do ROS, o nó ***modros_node:***

1 – Passo: Abra um terminal, digite ***roscore*** e dê enter.
2 – Passo: Em outro terminal, digite ***rosrun modelica_bridge modbridge_node*** e dê enter.

Para rodar o modelo MODEST.Examples.Sistema_Esteira anterior **em tempo real** podemos proceder de 3 maneiras (**com flag -rt=1 é a melhor maneira e com menos bugs**):

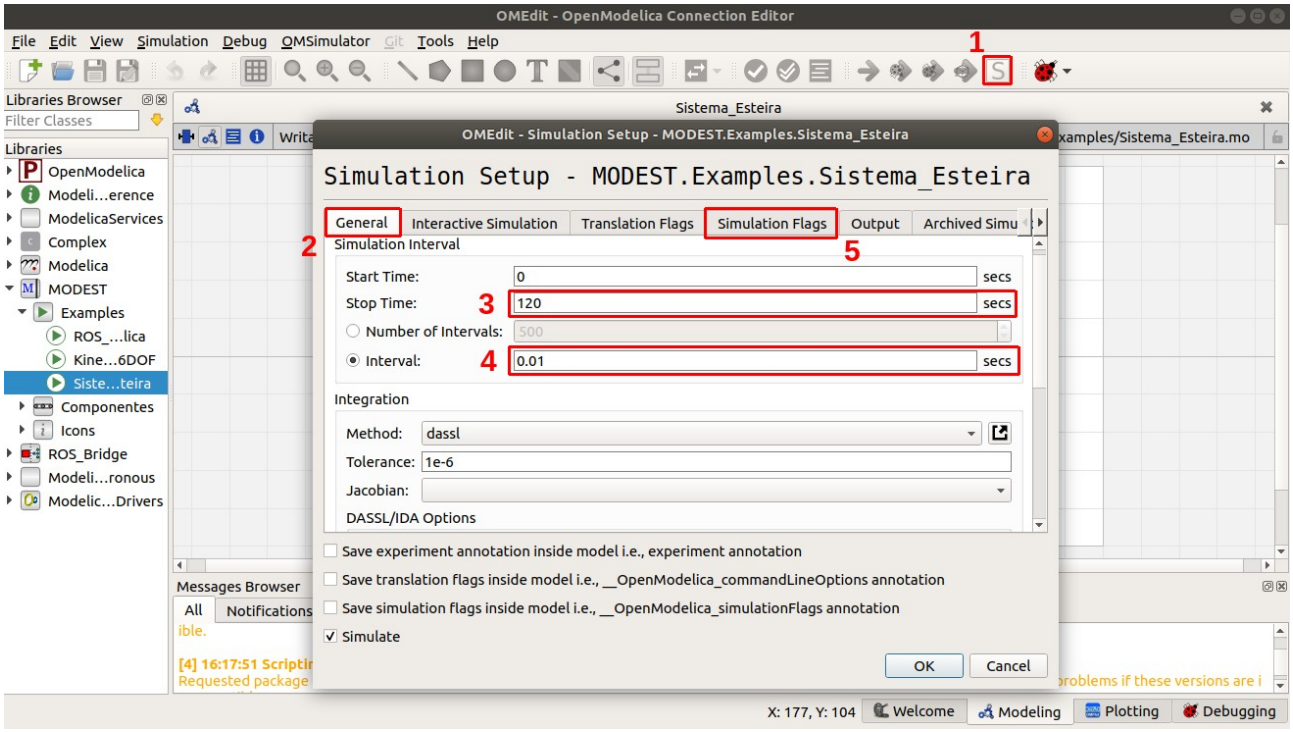
1. **flag de simulação *-rt=1* no Setup de Simulação [OpenModelica]**
2. Bloco *SynchronizeRealTime* da biblioteca *Modelica_DeviceDrivers*
3. *Interactive Simulation* [OpenModelica]

Existem bugs com a manipulação de eventos e/ou simulações longas por parte dos Solvers do OpenModelica que podem causar erros (e fazem a esteira não atualizar a próxima posição).

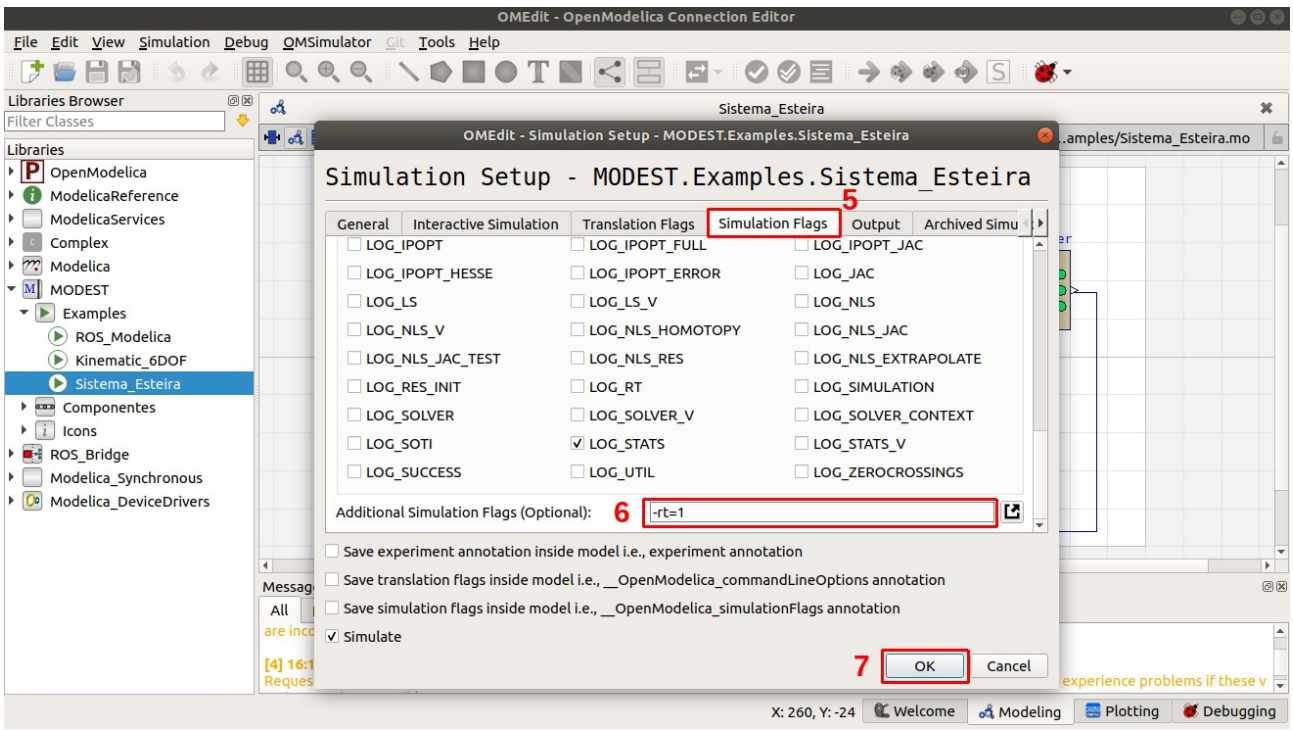
Para o **modo 1** vá até o *Simulation Setup* e siga os passos da sequência da imagem a seguir para uma simulação de 2 minutos, ou 120 segundos. **Imporante:** O número de intervalos não deve ultrapassar 220.000. Para o intervalo de 0.01 da imagem o número de intervalos é 12.000 (pois o StopTime é de 120 s).

Para o **modo 2** insira o bloco da biblioteca no ambiente e não coloque a flag anterior. Para o **modo 3** também sem a flag anterior selecione a opção de *Interactive Simulation* do *Simulation Setup*. O procedimento é descrito no guia do usuário (Esta opção está em desenvolvimento).

Primeira etapa de configuração do *Simulation Setup*



Segunda etapa de configuração do *Simulation Setup*



Guia de Utilização Rápida do OMShell

O OMShell é um manipulador de sessão interativo que analisa e interpreta comandos e expressões Modelica para avaliação, simulação, plotagem e etc. O manipulador de sessão também contém recursos simples de histórico e complemento de nomes de arquivos e determinados identificadores nos comandos (ativados com TAB).

A lista de comandos completa encontra-se em:
https://build.openmodelica.org/Documentation/OpenModelica.Scripting.html

1. **cd( ):** Exibe o diretório atual.
2. **cd("/home/lucas/example/"):** Muda para o diretório da indicado (com aspas).
3. **loadFile("/home/lucas/example/nome_do_arquivo.mo"):** Carrega os modelos e arquivos para o ambiente (com aspas).
   loadFile ("nome_do_arquivo.mo") também inclue biblioteca de terceiros, geralmente encapsuladas em um package.mo no diretório da biblioteca.
4. **list(Nome_do_Modelo):** Exibe o código Modelica do modelo carregado.
5. **simulate(Nome_do_Modelo, …):** Simula o arquivo com as opções desejadas [tempo de início, tempo de parada, método de resolução, formato do arquivo de saída]. (Após digitar *simulate* aperte TAB para completar o comando e preencha os campos como desejar).
   (**Arquivos de simulação são armazenados no diretório atual do ambiente**).
6. **readSimulationResultVars("nome_do_arquivo_res.mat"):** Exibe as variáveis armazenadas em um arquivo de simulação.
7. **plot(variavel):** Plota a variável (nome da variavel) do modelo da última simulação realizada. A função plot para mais de uma variável é indicada como **plot ({var1,var2}).** O plot pode ser personalizado de acordo com os comandos passados a função (ver lista completa) ou personalizado na própria janela de plot. **plot (var1, true)** plota a variável em uma nova aba sem fechar a anterior.

Exemplo de uma simulação com OMShell (Motor DC)

Digite os comandos a seguir em um terminal do OMShell

```
>> cd()
```

```
"/tmp/OpenModelica"
```

```
>> cd("/home/lucas/Documentos/modelica/ModelosEstudo")
```

```
"/home/lucas/Documentos/modelica/ModelosEstudo"
```

```
>> loadFile(“01_DCMotor.mo”)
```

```
true
```

```
>> list(DCMotor)
```

```
“model DCMotor

      Modelica.Electrical.Analog.Sources.SignalVoltage signalVoltage1;
      Modelica.Electrical.Analog.Basic.Resistor resistor1(R = 10);
      Modelica.Electrical.Analog.Basic.Inductor inductor1(L = 0.1);
      Modelica.Electrical.Analog.Basic.EMF emf1;
      Modelica.Mechanics.Rotational.Components.Inertia inertia1(J = 0.3);
      Modelica.Electrical.Analog.Basic.Ground ground1;
      Modelica.Blocks.Sources.Step step1;
equation
      connect(inductor1.p, resistor1.n);
      connect(resistor1.p, signalVoltage1.p);
      connect(inductor1.n, emf1.p);
      connect(signalVoltage1.n, ground1.p);
      connect(ground1.p, emf1.n);
      connect(emf1.flange, inertia1.flange_a);
      connect(signalVoltage1.v, step1.y);
      connect(inductor1.n, emf1.p);
end DCMotor;"
```

```
>> simulate(DCMotor, startTime=0, stopTime=20, numberOfIntervals=500, tolerance=1e-4, method="dassl", outputFormat="mat")
```

```
record SimulationResult

      resultFile = "/home/lucas/Documentos/modelica/ModelosEstudo/DCMotor_res.mat",
      simulationOptions = "startTime = 0.0, stopTime = 20.0,     numberOfIntervals = 500,
tolerance = 0.0001, method = 'dassl',  fileNamePrefix = 'DCMotor', options = '',
outputFormat = 'mat',     variableFilter = '.*', cflags = '', simflags = ''",
      messages = "LOG_SUCCESS | info | The initialization finished     successfully
without homotopy method.
LOG_SUCCESS | info | The simulation finished successfully.
",     timeFrontend = 0.353113875,
      timeBackend = 0.018912698,
      timeSimCode = 0.004587216,
      timeTtemplates = 0.142822901,
      timeCompile = 12.524740986,
      timeSimulation = 0.231085814,
      timeTotal = 13.275427634
end SimulationResult;
```

```
>> readSimulationResultVars("DCMotor_res.mat")
```

```
{"der(emf1.phi)","der(inductor1.i)","der(inertia1.phi)","der(inertia1.w)","emf1.
flange.phi","emf1.flange.tau","emf1.i","emf1.k","emf1.n.i","emf1.n.v","emf1.p.i"
,"emf1.p.v","emf1.phi","emf1.tau","emf1.tauElectrical","emf1.v","emf1.w","ground
1.p.i","ground1.p.v","inductor1.L","inductor1.i","inductor1.n.i","inductor1.n.v"
,"inductor1.p.i","inductor1.p.v","inductor1.v","inertia1.J","inertia1.a","inerti
a1.flange_a.phi","inertia1.flange_a.tau","inertia1.flange_b.phi","inertia1.flang
e_b.tau","inertia1.phi","inertia1.w","resistor1.LossPower","resistor1.R","resist
or1.R_actual","resistor1.T","resistor1.T_heatPort","resistor1.T_ref","resistor1.
alpha","resistor1.i","resistor1.n.i","resistor1.n.v","resistor1.p.i","resistor1.
p.v","resistor1.v","signalVoltage1.i","signalVoltage1.n.i","signalVoltage1.n.v",
"signalVoltage1.p.i","signalVoltage1.p.v","signalVoltage1.v","step1.height","ste
p1.offset","step1.startTime","step1.y","time"}
```

```
>> plot({inertia1.flange_a.phi,step1.y})
```

```
false
```