

# Modelica Bridge – ROS Melodic – Ubuntu 18.04 LTS

## Manual de Instalação

Lucas Gabriel Cosmo Moraes  
Dezembro, 2019.

O `modelica_bridge` é um pacote ROS para realizar comunicação via socket TCP/IP entre ROS com a simulação do OpenModelica, cuja linguagem de modelagem é Modelica. A comunicação é realizada entre um nó `modelica_bridge` (que atua como servidor) no ROS e um bloco `ROS_bridge` (que atua como cliente) em um pacote Modelica. `Modelica_bridge` foi desenvolvido por Shashank Swaminathan e o código fonte está disponível em: <https://github.com/ModROS>

A ponte de comunicação encontra-se atualmente disponível para instalação direta com ROS Kinetic, contando com um comando para a instalação direta nesta versão (`sudo apt-get install ros-kinetic-modelica-bridge`).

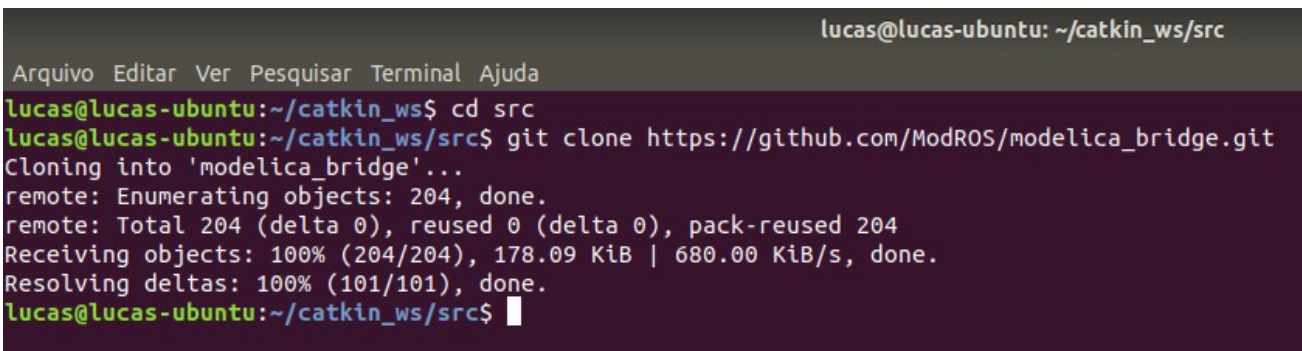
Mais informações em: [http://wiki.ros.org/modelica\\_bridge/Tutorials/Introduction%20to%20modelica\\_bridge](http://wiki.ros.org/modelica_bridge/Tutorials/Introduction%20to%20modelica_bridge)

A adaptação para ROS Melodic se utiliza de pequenas alterações para fazer funcionar na versão Melodic.

**1 – Passo:** Vá até a página ([http://wiki.ros.org/modelica\\_bridge/Tutorials/Introduction%20to%20modelica\\_bridge](http://wiki.ros.org/modelica_bridge/Tutorials/Introduction%20to%20modelica_bridge)) de tutorial de instalação e copie o local do código fonte:

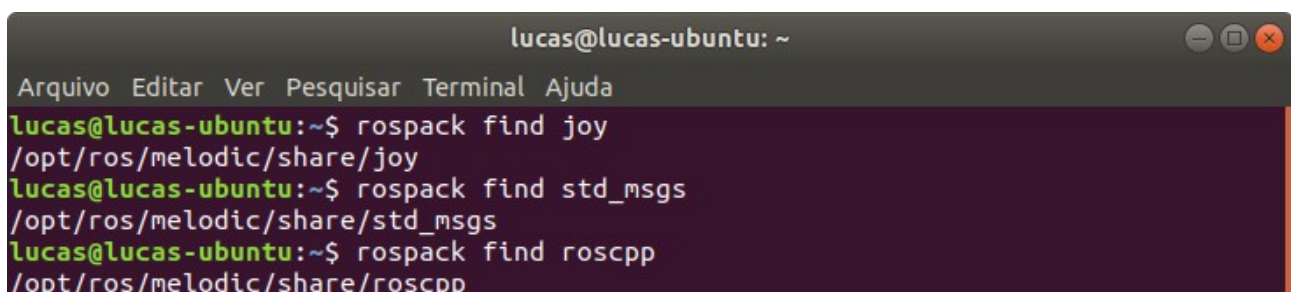
```
git clone https://github.com/ModROS/modelica_bridge.git
```

**2 – Passo:** Abra um terminal e vá para o espaço de trabalho catkin e, em seguida mude de diretório para o sub diretório `src`. Cole o código copiado e dê enter no comando.



```
lucas@lucas-ubuntu: ~/catkin_ws/src
Arquivo Editar Ver Pesquisar Terminal Ajuda
lucas@lucas-ubuntu:~/catkin_ws$ cd src
lucas@lucas-ubuntu:~/catkin_ws/src$ git clone https://github.com/ModROS/modelica_bridge.git
Cloning into 'modelica_bridge'...
remote: Enumerating objects: 204, done.
remote: Total 204 (delta 0), reused 0 (delta 0), pack-reused 204
Receiving objects: 100% (204/204), 178.09 KiB | 680.00 KiB/s, done.
Resolving deltas: 100% (101/101), done.
lucas@lucas-ubuntu:~/catkin_ws/src$
```

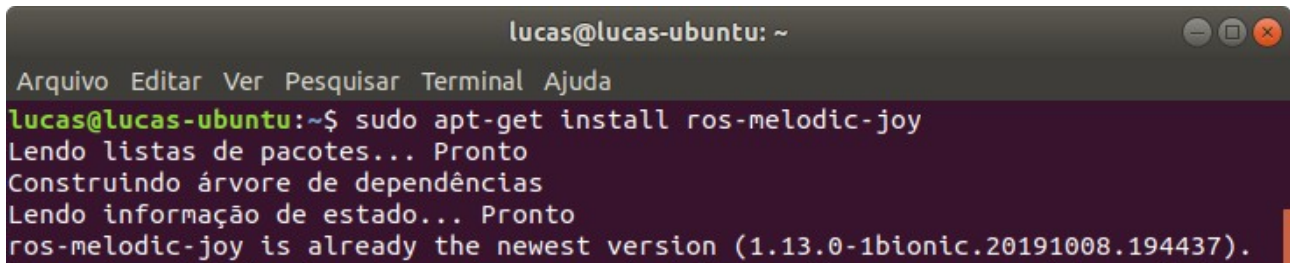
**3 – Passo:** O pacote `modelica_bridge` se utiliza de outros pacotes para o pleno funcionamento. Geralmente os erros que aparecerão estão associados a estas dependências. Tenha certeza que os seguintes pacotes estão instalados por padrão no seu ROS. (`joy`, `roscpp`, `std_msgs`).



```
lucas@lucas-ubuntu: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
lucas@lucas-ubuntu:~$ rospack find joy
/opt/ros/melodic/share/joy
lucas@lucas-ubuntu:~$ rospack find std_msgs
/opt/ros/melodic/share/std_msgs
lucas@lucas-ubuntu:~$ rospack find roscpp
/opt/ros/melodic/share/roscpp
```

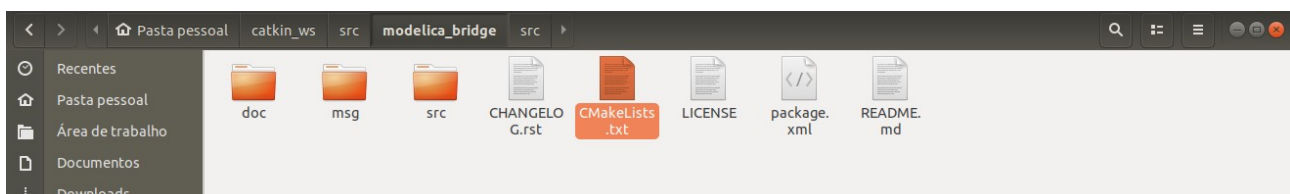
Se algum dos pacotes não estiver disponível você deve instalar o pacote que falta por meio do comando a seguir:

```
sudo apt-get install ros-melodic-'nome do pacote'
```

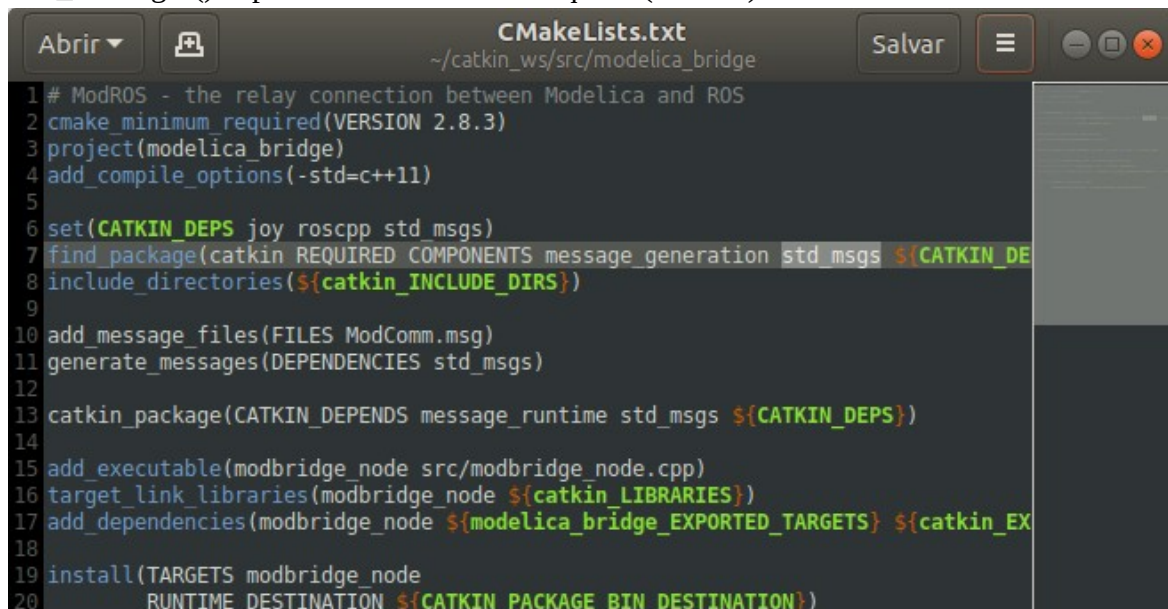


```
lucas@lucas-ubuntu: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
lucas@lucas-ubuntu:~$ sudo apt-get install ros-melodic-joy  
Lendo listas de pacotes... Pronto  
Construindo árvore de dependências  
Lendo informação de estado... Pronto  
ros-melodic-joy is already the newest version (1.13.0-1bionic.20191008.194437).
```

**4 – Passo:** Agora vá ao diretório *modelica\_bridge* (dentro de *catkin\_ws/src*) e procure pelo arquivo *CMakeLists.txt* e o abra.



**5 – Passo:** Você deve **completar** as informações **conforme a imagem a seguir**. Serão feitas alterações incluindo-se algumas vezes *'std\_msgs'* e um *'DEPENDENCIES std\_msgs'* no *generate\_messages()*. Após concluído salve o arquivo (`ctrl + S`).



```
CMakeLists.txt  
~/catkin_ws/src/modelica_bridge  
Abrir Salvar  
1 # ModROS - the relay connection between Modelica and ROS  
2 cmake_minimum_required(VERSION 2.8.3)  
3 project(modelica_bridge)  
4 add_compile_options(-std=c++11)  
5  
6 set(CATKIN_DEPS joy roscpp std_msgs)  
7 find_package(catkin REQUIRED COMPONENTS message_generation std_msgs ${CATKIN_DE  
8 include_directories(${catkin_INCLUDE_DIRS})  
9  
10 add_message_files(FILES ModComm.msg)  
11 generate_messages(DEPENDENCIES std_msgs)  
12  
13 catkin_package(CATKIN_DEPENDS message_runtime std_msgs ${CATKIN_DEPS})  
14  
15 add_executable(modbridge_node src/modbridge_node.cpp)  
16 target_link_libraries(modbridge_node ${catkin_LIBRARIES})  
17 add_dependencies(modbridge_node ${modelica_bridge_EXPORTED_TARGETS} ${catkin_EX  
18  
19 install(TARGETS modbridge_node  
20     RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
```

(O arquivo *CMakeLists.txt* é a entrada para o sistema de criação do CMake para a construção de pacotes de software. Qualquer pacote compatível com CMake contém um ou mais arquivos *CMakeLists.txt* que descrevem como criar o código e onde instalá-lo.

O arquivo foi alterado para ser igual ao estabelecido em: <http://wiki.ros.org/catkin/CmakeLists.txt>).

**6 – Passo:** Concluída toda a alteração, vá até o diretório `catkin_ws` e compile com `catkin_make`.

```
lucas@lucas-ubuntu:~/catkin_ws$ catkin_make
Base path: /home/lucas/catkin_ws
Source space: /home/lucas/catkin_ws/src
Build space: /home/lucas/catkin_ws/build
Devel space: /home/lucas/catkin_ws/devel
Install space: /home/lucas/catkin_ws/install
####
#### Running command: "cmake /home/lucas/catkin_ws/src -DCATKIN_DEVEL_P
PREFIX=/home/lucas/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/lucas/ca
tkin_ws/install -G Unix Makefiles" in "/home/lucas/catkin_ws/build"
####

[ 77%] Built target modelica_bridge_generate_messages
[ 88%] Building CXX object modelica_bridge/CMakeFiles/modbridge_node.dir/src/modbridge_node.cpp.o
[100%] Linking CXX executable /home/lucas/catkin_ws/devel/lib/modelica_
bridge/modbridge_node
[100%] Built target modbridge_node
lucas@lucas-ubuntu:~/catkin_ws$
```

**7 – Passo:** Dê um `roslaunch` no pacote (com o `roscore` já iniciado em outro terminal):

```
roslaunch modelica_bridge modbridge_node
```

(Lembre-se que para todo novo terminal que será aberto deve-se dar ‘`source`’ a um `setup.bash` específico. Isto serve como controle de versões e workspace ativos. Veja em: [https://answers.ros.org/question/229365/do-i-really-need-to-source-catkin\\_wsdevelsetupbash/](https://answers.ros.org/question/229365/do-i-really-need-to-source-catkin_wsdevelsetupbash/)).

**8 – Passo:** Agora, para visualizar os tópicos ativos, abra um novo terminal dê um ‘`source`’ ao `setup.bash` do `catkin_ws/devel`. E execute o seguinte comando para visualizar os tópicos `/model_values` e `/control_values`:

```
rostopic list
```

```
lucas@lucas-ubuntu: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
lucas@lucas-ubuntu:~$ source ~/catkin_ws/devel/setup.bash
lucas@lucas-ubuntu:~$ rostopic list
/control_values
/model_values
/rosout
/rosout_agg
```

**Pronto!**

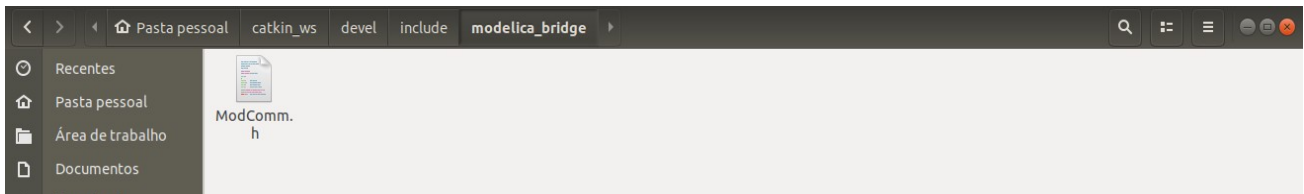
## PROBLEMAS COM A INSTALAÇÃO

Um problema comum está com a não identificação da biblioteca ModComm.h.

Este problema ocorre devido a compilação da biblioteca que gera o formato de mensagem. Para uma melhor configuração do arquivo de mensagem e solução do problema deve-se informar no CMakeLists que compile a biblioteca da mensagem antes de compilar programas que dependam desta.

**Solução:** [http://docs.ros.org/latest/api/catkin/html/howto/format2/building\\_msgs.html](http://docs.ros.org/latest/api/catkin/html/howto/format2/building_msgs.html)

ModComm é uma biblioteca gerada automaticamente durante a compilação. Recebe este nome pois a mensagem personalizada (criada pelo autor do pacote modelica\_bridge) tem nome de ModComm.msg.



```

//-----
//Biblioteca Modcomm.h

// Generated by gencpp from file modelica_bridge/ModComm.msg
// DO NOT EDIT!

#ifndef MODELICA_BRIDGE_MESSAGE_MODCOMM_H
#define MODELICA_BRIDGE_MESSAGE_MODCOMM_H

#include <string>
#include <vector>
#include <map>
#include <ros/types.h>
#include <ros/serialization.h>
#include <ros/builtin_message_traits.h>
#include <ros/message_operations.h>

namespace modelica_bridge
{
template <class ContainerAllocator>
struct ModComm_
{
typedef ModComm_<ContainerAllocator> Type;

ModComm_()
: data()
, size(0) {
}
ModComm_(const ContainerAllocator& _alloc)
: data(_alloc)
, size(0) {
(void)_alloc;
}

typedef std::vector<double, typename ContainerAllocator::template rebind<double>::other > _data_type;
_data_type data;

typedef uint16_t _size_type;
_size_type size;

typedef boost::shared_ptr< ::modelica_bridge::ModComm_<ContainerAllocator> > Ptr;
typedef boost::shared_ptr< ::modelica_bridge::ModComm_<ContainerAllocator> const> ConstPtr;

}; // struct ModComm_

typedef ::modelica_bridge::ModComm_<std::allocator<void> > ModComm;

typedef boost::shared_ptr< ::modelica_bridge::ModComm > ModCommPtr;
typedef boost::shared_ptr< ::modelica_bridge::ModComm const> ModCommConstPtr;

// constants requiring out of line definition

template<typename ContainerAllocator>
std::ostream& operator<<(std::ostream& s, const ::modelica_bridge::ModComm_<ContainerAllocator> & v)
{
ros::message_operations::Printer< ::modelica_bridge::ModComm_<ContainerAllocator> >::stream(s, "", v);
return s;
}

} // namespace modelica_bridge

namespace ros
{
namespace message_traits
{

// BOOLTRAITS {'IsFixedSize': False, 'IsMessage': True, 'HasHeader': False}
// {'modelica_bridge': ['/tmp/binarydeb/ros-kinetic-modelica-bridge-0.1.1/msg']}

// !!!!!!! ['__class__', '__delattr__', '__dict__', '__doc__', '__eq__', '__format__', '__getattribute__', '__hash__', '__init__', '__module__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_parsed_fields',
'constants', 'fields', 'full_name', 'has_header', 'header_present', 'names', 'package', 'parsed_fields', 'short_name', 'text', 'types']

template <class ContainerAllocator>
struct IsFixedSize< ::modelica_bridge::ModComm_<ContainerAllocator> >
: FalseType
{
};

```

```

template <class ContainerAllocator>
struct IsFixedSize< ::modelica_bridge::ModComm_<ContainerAllocator> const>
: FalseType
{ };

template <class ContainerAllocator>
struct IsMessage< ::modelica_bridge::ModComm_<ContainerAllocator> >
: TrueType
{ };

template <class ContainerAllocator>
struct IsMessage< ::modelica_bridge::ModComm_<ContainerAllocator> const>
: TrueType
{ };

template <class ContainerAllocator>
struct HasHeader< ::modelica_bridge::ModComm_<ContainerAllocator> >
: FalseType
{ };

template <class ContainerAllocator>
struct HasHeader< ::modelica_bridge::ModComm_<ContainerAllocator> const>
: FalseType
{ };

template<class ContainerAllocator>
struct MD5Sum< ::modelica_bridge::ModComm_<ContainerAllocator> >
{
    static const char* value()
    {
        return "dce2d314d75c473202b101e56f1e5ee9";
    }

    static const char* value(const ::modelica_bridge::ModComm_<ContainerAllocator>&) { return value(); }
    static const uint64_t static_value1 = 0xdce2d314d75c4732ULL;
    static const uint64_t static_value2 = 0x02b101e56f1e5ee9ULL;
};

template<class ContainerAllocator>
struct DataType< ::modelica_bridge::ModComm_<ContainerAllocator> >
{
    static const char* value()
    {
        return "modelica_bridge/ModComm";
    }

    static const char* value(const ::modelica_bridge::ModComm_<ContainerAllocator>&) { return value(); }
};

template<class ContainerAllocator>
struct Definition< ::modelica_bridge::ModComm_<ContainerAllocator> >
{
    static const char* value()
    {
        return "float64[] data\n\
uint16 size\n\
";
    }

    static const char* value(const ::modelica_bridge::ModComm_<ContainerAllocator>&) { return value(); }
};

} // namespace message_traits
} // namespace ros

namespace ros
{
namespace serialization
{

template<class ContainerAllocator> struct Serializer< ::modelica_bridge::ModComm_<ContainerAllocator> >
{
    template<typename Stream, typename T> inline static void allInOne(Stream& stream, T m)
    {
        stream.next(m.data);
        stream.next(m.size);
    }
}

```

```

    ROS_DECLARE_ALLINONE_SERIALIZER
}; // struct ModComm_

} // namespace serialization
} // namespace ros

namespace ros
{
    namespace message_operations
    {

template<class ContainerAllocator>
struct Printer< ::modelica_bridge::ModComm_<ContainerAllocator> >
{
    template<typename Stream> static void stream(Stream& s, const std::string& indent, const ::modelica_bridge::ModComm_<ContainerAllocator>&
v)
    {
        s << indent << "data[]" << std::endl;
        for (size_t i = 0; i < v.data.size(); ++i)
        {
            s << indent << "  data[" << i << "]: ";
            Printer<double>::stream(s, indent + " ", v.data[i]);
        }
        s << indent << "size: ";
        Printer<uint16_t>::stream(s, indent + " ", v.size);
    }
};

} // namespace message_operations
} // namespace ros

#endif // MODELICA_BRIDGE_MESSAGE_MODCOMM_H

```