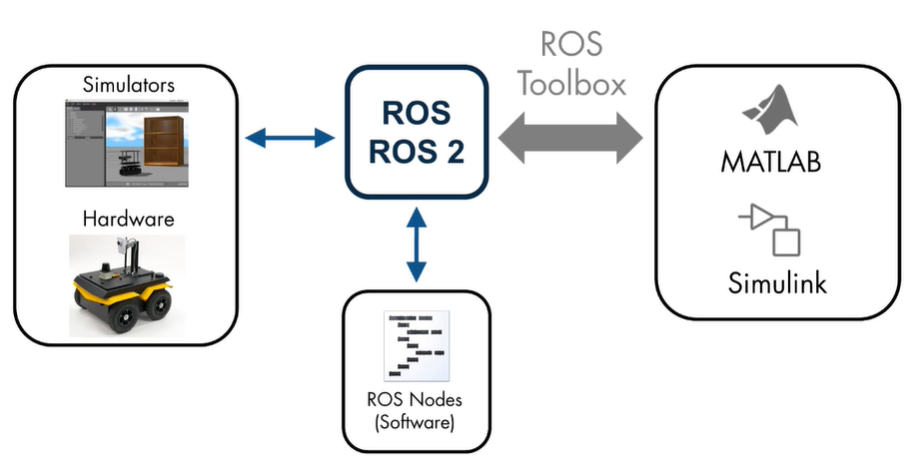
# [Matlab中的ROS](https://www.ncnynl.com/archives/201708/1936.html)

ROS（Robot Operating System）被广泛应用于机器人研究和教学当中，全球有400多所高校和科研单位在使用ROS。同时ROS也开始逐渐被工程师们采纳，应用在如无人驾驶汽车、无人驾驶飞机、水下搜索机器人，家用服务机器人和工业机器人等领域。ROS提供了标准的数据通信架构，常用的硬件驱动，以及成熟的机器人算法。然而，用户需要先学习Python或C++等语言，然后才能在ROS下编写、编译、运行和测试算法。很多机器人研究人员希望能够同时使用MATLAB/Simulink与ROS，最好能够在MATLAB/Simulink下编写和测试算法，并且不需要编译就能在支持ROS的机器人或者模拟器上验证算法。算法测试通过以后，直接从MATLAB/Simulink下生成ROS代码，然后发布到机器人上。

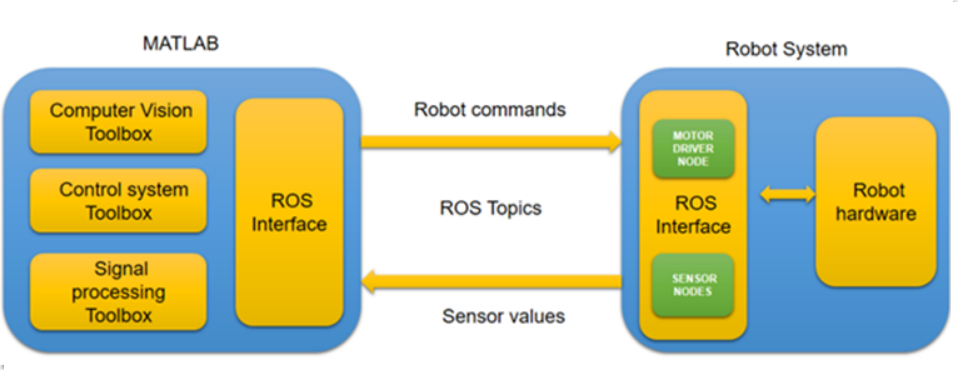
MathWorks机器人系统工具箱提供MATLAB/Simulink与ROS接口。通过此工具箱, 机器人研究人员可以在MATLAB/Simulink环境下开发和测试算法，然后在支持ROS的机器人模拟器上（如Gazebo和V-REP），或支持ROS的机器人上验证算法。

**ROS-MATLAB是什么**



**ROS工具箱**提供了一个在MATLAB 和Simulink 使用ROS操作系统（ROS和ROS2）的接口，可以创建ROS节点网络。工具箱包括MATLAB函数和Simulink模块，用于导入、分析和回放rosbag文件中记录的ROS数据。您还可以连接到实时ROS网络以访问ROS消息。

工具箱允许您通过桌面模拟和连接到外部机器人模拟器（如Gazebo）来验证ROS节点。ROS工具箱支持C++代码生成（与Simulink Coder）™), 使您能够从Simulink模型自动生成ROS节点并部署到模拟或物理硬件。支持Simulink外部模式，可以在模型在硬件上运行时查看消息和更改参数。



上边这张框图就是MATLAB和机器人系统的通信框架，可以看到，Matlab拥有强大而丰富的算法功能包（视觉处理、控制系统、信号处理等等），通过MATLAB ROS Toolbox可以获取机器人的数据，在处理之后再将控制指令发送到机器人。简而言之，**ROS Toolbox让Matlab成为了机器人强大的计算后台**。

需要注意的是，ROS-MATLAB在Matlab2013之后的版本才有，而且需要Matlab安装robotics system toolbox，可以登录官网链接下载安装。

# 在MATLAB中使用ROS

* ROS网络：包括机器人系统的不同部分（例如规划器或摄像机接口），这些部分通过ROS进行通信。网络可以分布在多台机器上。
* ROS master：协调ROS网络的不同部分。它由主URI（统一资源标识符）标识，该URI指定运行主URI的计算机的主机名或IP地址。
* ROS node：包含相关ROS功能（如发布者、订阅者和服务）的集合。一个ROS网络可以有许多ROS节点。
* 发布者、订阅者和服务是处理数据的不同类型的ROS实体。它们使用消息交换数据。

发布者将消息发送到特定主题（如“里程计”），而该主题的订阅者将接收这些消息。一个主题可以与多个发布者和订阅者关联。

### 1、初始化ROS Network

使用rosinit来初始化ROS。默认情况下，rosinit在MATLAB中创建一个ROS主节点，**并启动一个连接到主节点的全局节点**。全局节点由其他ROS函数自动使用。

rosinit

Launching ROS Core...

Done in 0.84716 seconds.

Initializing ROS master on http://192.168.0.10:49666.

Initializing global node /matlab\_global\_node\_80471 with NodeURI http://bat800402glnxa64:37773/

**还可以使用rosinit命令连接到外部ROS主机**（例如在机器人或虚拟机上运行）。可以通过两种方式指定主机的地址：通过IP地址或运行主机的计算机的主机名。

每次调用rosinit之后，必须先调用rosshutdown，然后再使用不同的语法调用rosinit。为简洁起见，这些示例中省略了对rosshutdown的调用。

rosinit('192.168.1.1')

对rosinit的调用都假设主机在端口11311上接受网络连接，这是标准的ROS主机端口。如果主机在不同的端口上运行，可以将其指定为第二个参数。要连接到在主机名master\u host和端口12000上运行的ROS主机，请使用以下命令：

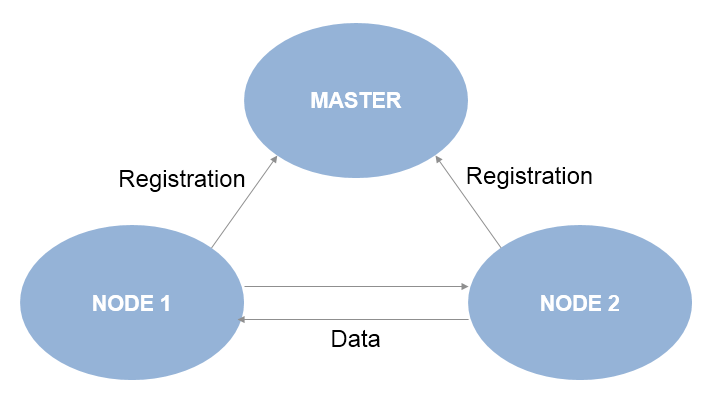
rosinit('192.168.1.1',12000)

或者

rosinit('http://192.168.1.1:12000')

ROS连接要正常工作，必须确保所有节点都可以与主节点以及彼此通信。各个节点必须与主节点通信以注册订户、发布者和服务。它们还必须能够相互通信以发送和接收数据。如果您的ROS网络设置不正确，则可能会发送数据而无法接收数据（反之亦然）。

这张图显示了一个ROS网络，其中有一个ROS主节点和两个不同的节点在主节点上注册。每个节点联系主节点以查找ROS网络中另一个节点的广告地址。一旦每个节点知道另一个节点的地址，就可以在没有主节点参与的情况下建立数据交换。



在MATLAB中创建一个节点：

* 使用robotics.ros.Node创建

node\_1 = robotics.ros.Node('node\_name', masterHost);

PS：新创建的节点将尝试连接到“localhost”上的ROS主机。如果您正在连接到外部主机，您必须使用它的IP地址或主机名。

**robotics.ros.Node命令可以创建一个独立于matlab\_node的新node。**即使在matlab中使用rosshutdown关闭了matlab\_node后，这些独立于matlab\_node的node不会受影响。

**在MATLAB中创建一个发布者或者订阅者节点**

MATLAB中可以使用**rospublisher函数**或通过调用**robotics.ros.Publisher**来创建Publisher对象：

* rospublisher只能使用rosinit生成的全局节点来发布消息。它不需要节点对象句柄作为参数。
* robotics.ros.Publisher使用robotics.ros.Node创建的其他节点。它需要一个节点对象句柄作为第一个参数。

需要注意的是，凡是运行rospublisher、rossubscriber等函数时，均默认使用当前的matlab\_node来发送或订阅消息；凡是用robotics.ros.Publisher、robotics.ros.Subscriber均使用指定的node来发送或订阅消息。

示例1：

segmentPub = rospublisher('/simple\_msgs', 'std\_msgs/Float64');

segmentMsg = rosmessage('std\_msgs/Float64');

segmentMsg.Data = 3.33;

% 注意，matlab中消息的字段都是大写开头的

send(segmentPub,segmentMsg);

sub = rossubscriber('/simple\_msgs');

pause(1) % Wait for message to updatesub.LatestMessage

示例2：

node1 = robotics.ros.Node('/test\_node\_1');

node2 = robotics.ros.Node('/test\_node\_2');

pub = robotics.ros.Publisher(node1,'/chatter','std\_msgs/String');

sub = robotics.ros.Subscriber(node2,'/chatter','std\_msgs/String');

msg = rosmessage('std\_msgs/String');

msg.Data = 'Message from Node 1';send(pub,msg) % Sent from node 1

pause(1) % Wait for message to updatesub.LatestMessage

Matlab的subscriber目前还存在着一些缺陷：  
(1). **无法设置 Matlab 中的 ROS 订阅者的callback函数的运行频率**。当用于接收数据时,会造成数据将丢失。这不是由回调函数的运行频率太低引起的。实际上，订阅者的回调函数的运行频率很高。  
(2). 大量消息的丢失是由于 matlab 中的callback函数很容易被挂起,因而 ROS 订阅者无法接收处理消息。Matlab 中的一个命令“**drawnow**”可以帮助解决这个问题。它可以更新图窗并处理任何被挂起的回调函数。  
(3). 这个工具箱的另一个缺陷是未提供在函数中接收实时的 ROS 消息的机制。ROS 消息是实时的,这意味着负责接收 ROS 消息的函数也应该被设计为实时在线的。否则,函数将立即完成,并会在 ROS 订阅者刚刚建立时终止。解决这个问题的方法是使用另一个数组来跟踪数据,使用 while 循环支持在线工作。此 while 循环有两部分:前台工作和后台工作。将已经接收到的数据写入另一个用于跟踪的数组中,这一工作在前台运行。而ROS 订阅者接收数据则在后台运行。只要确保 ROS 订阅者接收到的数据数量略大于复制到跟踪数组的数量,就可以保证一直在线接收。这个机制加上“drawnow”命令的效果很明显,数据的损失率降低到了1.2%。

**使用基本ROS消息类型**

* 创建消息

使用rosmessage来创建消息，例如：

twis = rosmessage('gemotry\_msgs/Twist')

twist =

ROS Twist message with properties:

MessageType: 'geometry\_msgs/Twist'

Linear: [1x1 Vector3]

Angular: [1x1 Vector3]

twist.Linear.Y = 5;

* 复制消息
* 可以创建引用副本，并且原始邮件共享相同的数据。

twistCopyRef = twist

twistCopyRef =

ROS Twist message with properties:

MessageType: 'geometry\_msgs/Twist'

Linear: [1x1 Vector3]

Angular: [1x1 Vector3]

Use showdetails to show the contents of the message

twistCopyRef.Linear.Z = 7;

twist.Linear

* 可以创建深度副本。深度副本，其中副本和原始消息各有自己的数据。

twistCopyDeep = copy(twist)

**发布数据**

[pub = rospublisher(topicname)](https://ww2.mathworks.cn/help/ros/ref/publisher.html?searchHighlight=rospublisher&s_tid=srchtitle#d123e27830)

[pub = rospublisher(topicname,msgtype)](https://ww2.mathworks.cn/help/ros/ref/publisher.html?searchHighlight=rospublisher&s_tid=srchtitle#d123e27845)

例如：

chatpub = rospublisher("/chatter","std\_msgs/String","DataFormat","struct");

%创建消息，并发送

msg = rosmessage(chatpub);

msg.Data = 'test phrase';

msg.Data = 'test phrase';

**订阅数据**

[msg = receive(sub)](https://ww2.mathworks.cn/help/ros/ref/receive.html#d123e8503)

[msg = receive(sub,timeout)](https://ww2.mathworks.cn/help/ros/ref/receive.html#d123e8524)

除了使用receive获取数据外，可以指定接收新消息时要调用的回调函数。使用回调函数允许在订阅者等待新消息时执行其他MATLAB代码。如果您想使用多个订阅者，回调是必不可少的。

基本语法形式：

[sub = rossubscriber(topicname,subCallback)](https://ww2.mathworks.cn/help/ros/ref/subscriber.html?searchHighlight=rossubscriber&s_tid=srchtitle#d123e30733)

function subCallback(src,msg)

**例如：**

robotpose = rossubscriber('/pose',@exampleHelperROSPoseCallback)

**调用服务：**

MATLAB中ROS 支持两种主要的通讯机制: topics 和services。MATLAB中使用rossvcserver命令来创建服务的服务器端，使用rossvcclient来创建服务的客户端。

语法形式：

server = rossvcserver(**'/test**', 'std\_srvs/Empty', @serviceCallback)

function response = serviceCallback(src,reqMsg,defaultRespMsg)

创建服务客户端

client = rossvcclient(servicename)

具体用法：

client = rossvcclient('/test');

成功创建服务的服务器端和客户端之后，创建服务请求，并发送之。

testreq = rosmessage(testclient);

testresp = call(testclient,testreq,'Timeout',3);

示例：

sumserver = rossvcserver('/sum', 'roscpp\_tutorials/TwoInts', @exampleHelperROSSumCallback)

sumclient = rossvcclient('/sum')

sumclient = rossvcclient('/sum')

function resp = exampleHelperROSSumCallback(~,req,resp)

%exampleHelperROSSumCallback Callback function for a service to add two integers

% RESP = exampleHelperROSSumCallback(~,REQ,RESP) adds the two integers A

% and B in the service request message REQ and returns them to the client

% in the response message RESP.

%

% And empty response message is given as an argument to this function and

% after assigning data to it, it has to be returned as a single output

% argument. This output will be sent to the service client.

%

% See also ROSServicesExample.

% Copyright 2014-2015 The MathWorks, Inc.

resp.Sum = req.A + req.B;

end

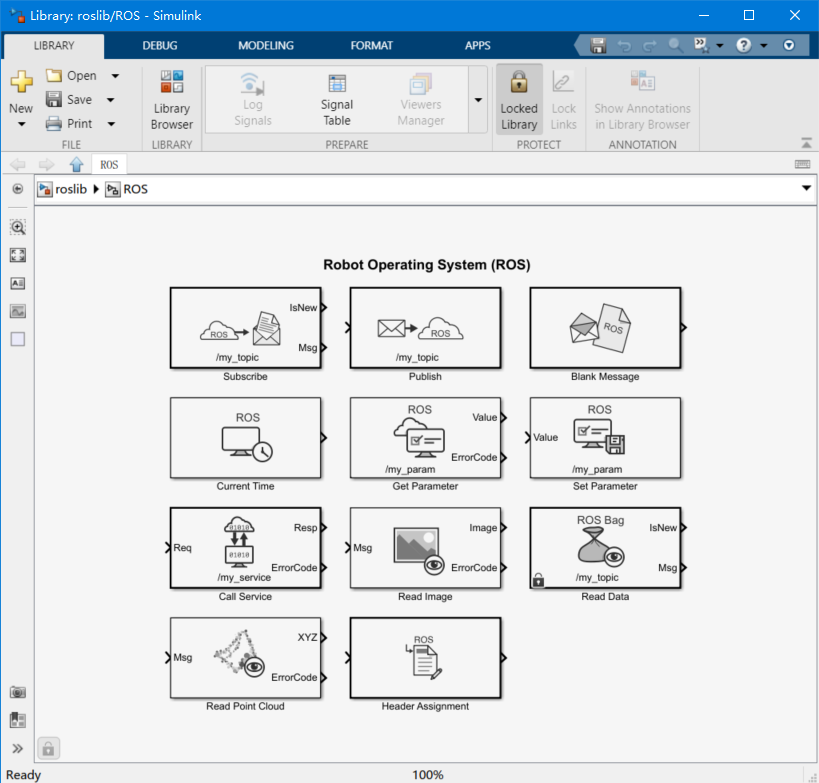
**在Simulink中使用ROS**

Simulink支持机器人操作系统（ROS），使得用户可用Simulink创建模型在ROS网络中使用。ROS是一个通信布局，允许机器人系统的不同部分以消息的形式交换信息。一个组件通过发布消息到一个话题，例如“/odometry”，其它组件通过订阅该话题接收。  
  
Simulink支持ROS包括Simulink库用于发送和接收指定话题的消息，当用户仿真模型时，Simulink连接到ROS网络，该网络可以运行在Simulink上的机器人，也可以是远程的系统。一旦建立连接，Simulink一直与ROS网络交换消息直到仿真结束。如果安装了嵌入式代码，用户还可以从Simulink模型为单独的ROS组件或者节点生成C++代码。

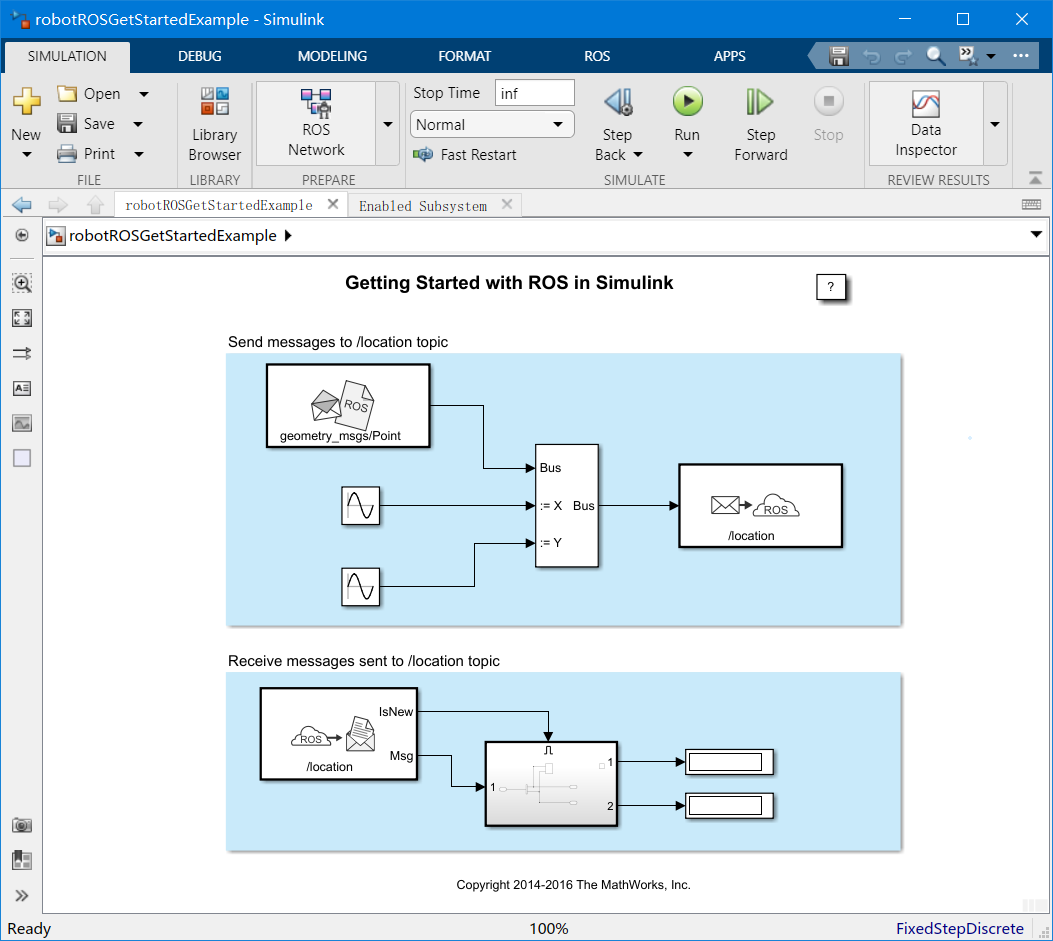
本小节将学习如何创建和运行Simulink模型以发送和接收ROS消息。将使用Simulink发布机器人的（x,y）位置坐标，还会订阅相同的定位话题并显示接收到的（x,y）坐标。

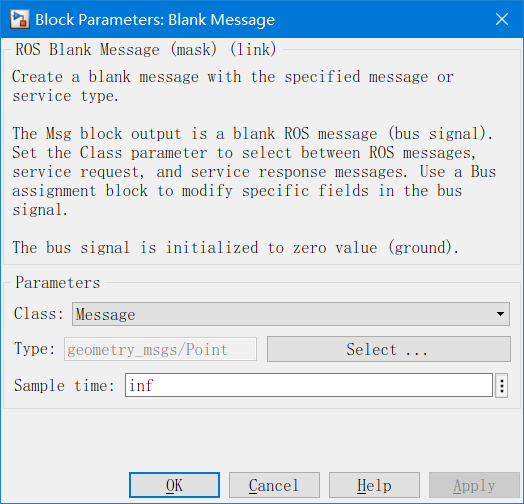
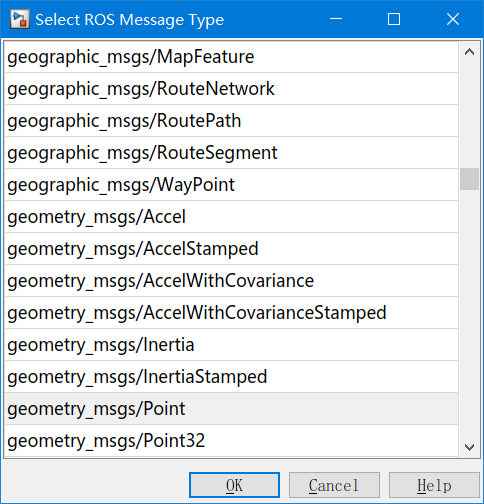
在MATLAB的命令窗口，输入：

roslib

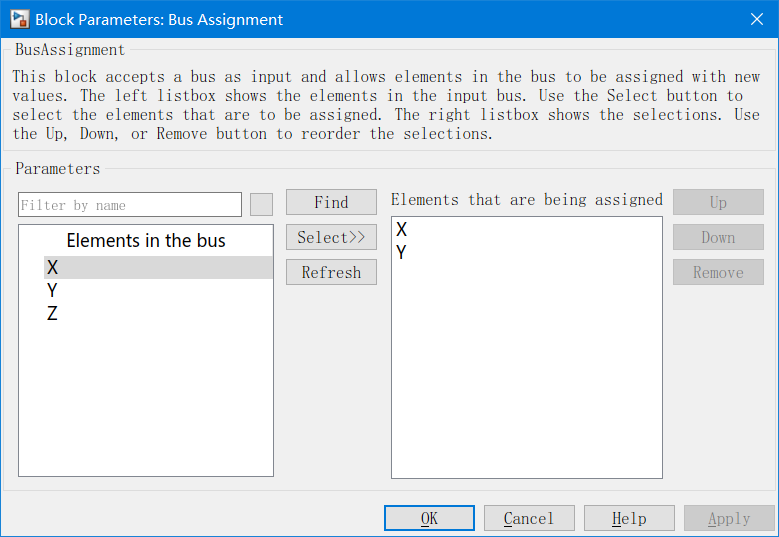


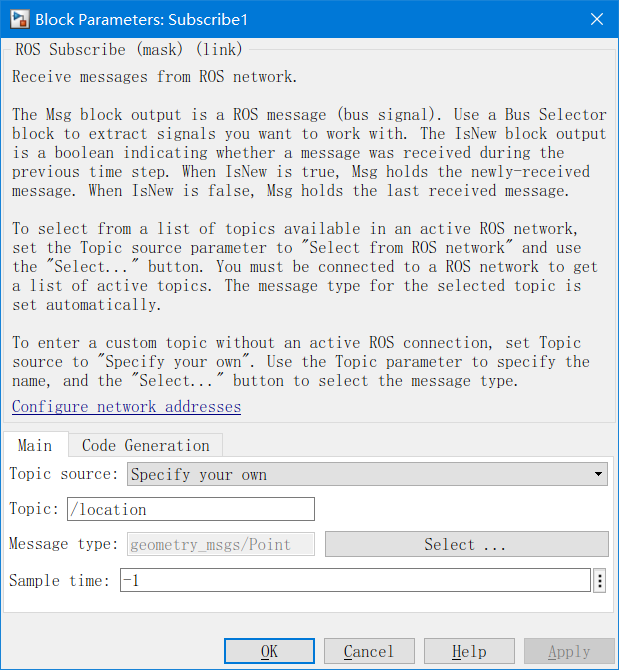
案例1：



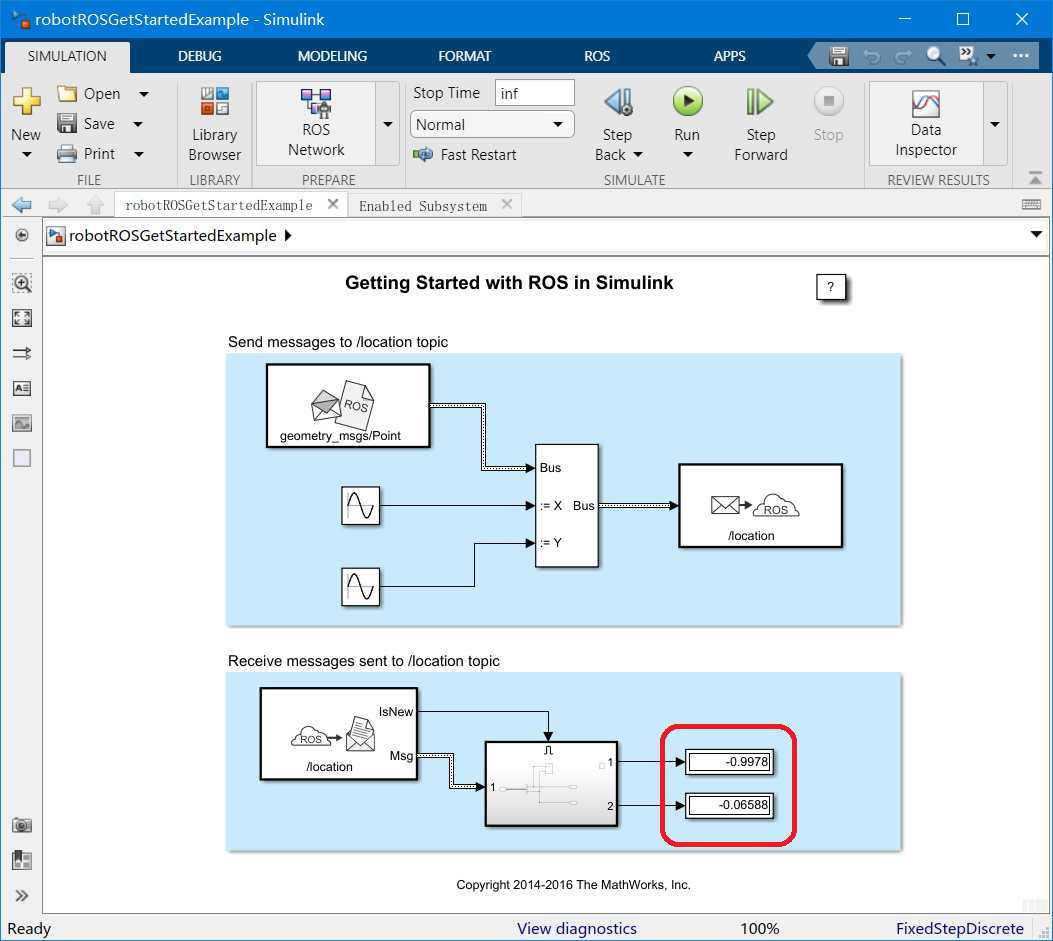
 

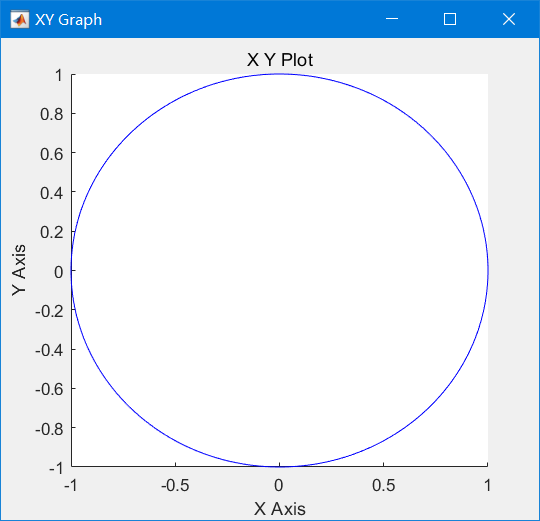
ROS消息在Simulink中被表述为“bus signal”，总线信号是Simulink信号的捆绑，可以包含其它总线信号。ROS“Blank Message”块输出一个与ROS消息有关的Simulink总线消息。

创建一个订阅者



在上面的模型中，“Subscribe”块在每个时间步进输出一个消息（总线信号），如果根本没有接收到消息，会输出空白消息（例如，消息的值为0）。因此，（X,Y）坐标初始化绘制在（0,0）。

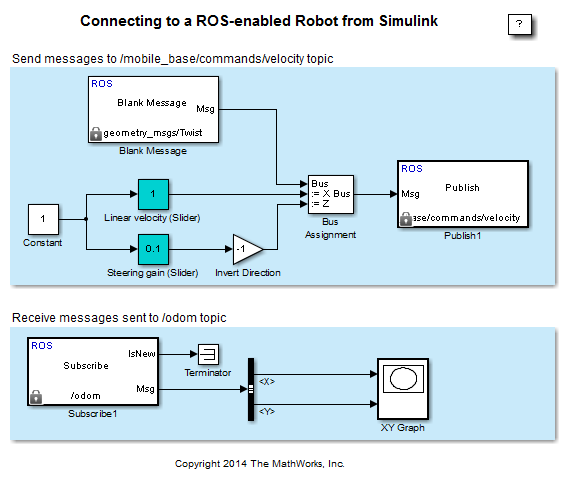




# [从Simulink连接到ROS使能的机器人](https://www.ncnynl.com/archives/201909/3298.html)

Simulink连接到一个物理机器人或者ROS使能的机器人仿真器，如Gazebo，

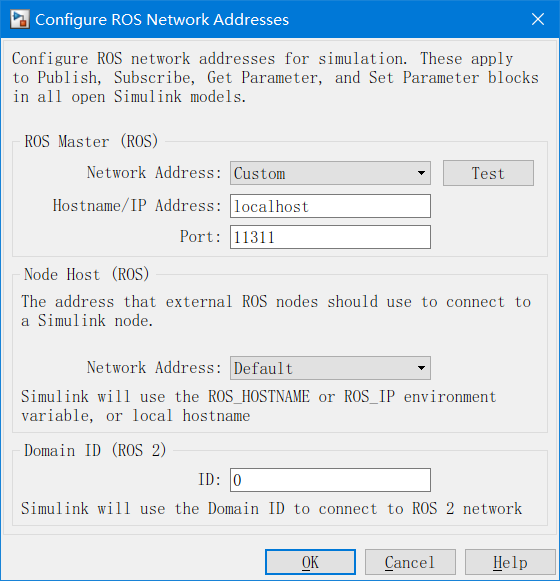
* 配置Simulink以连接到使用ROS的单独机器人仿真器；
* 发送速度命令到仿真机器人；
* 从仿真机器人接收位置信息。



1. 启动机器人仿真器

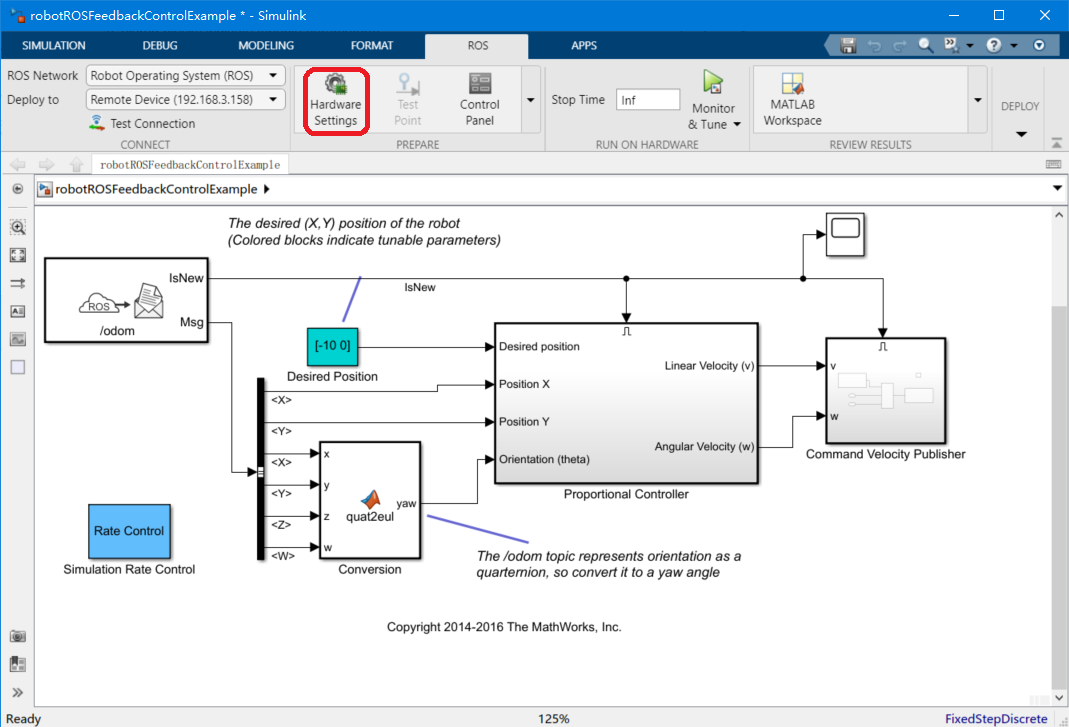
启动一个基于ROS的仿真差动机器人仿真器，该仿真器在“/mobile\_base/commands/velocity”主题上，接收“geometry\_msgs/Twist”类型的速度指令消息。以“nav\_msgs/Odometry”类型的消息，发送里程计的信息到“/odom”主题。

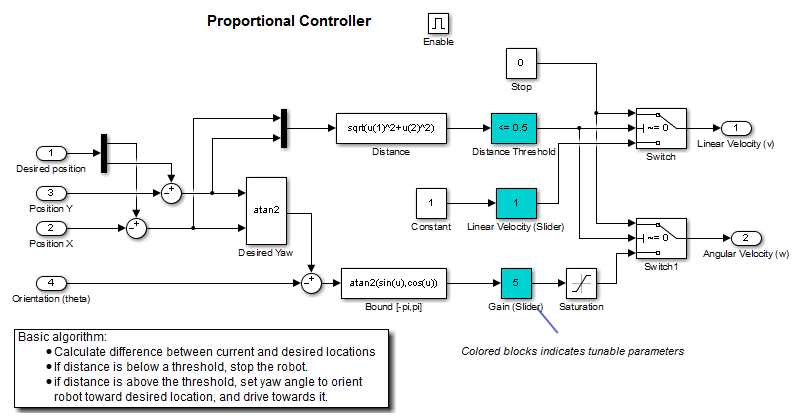
1. 配置Simulink以连接到ROS网络



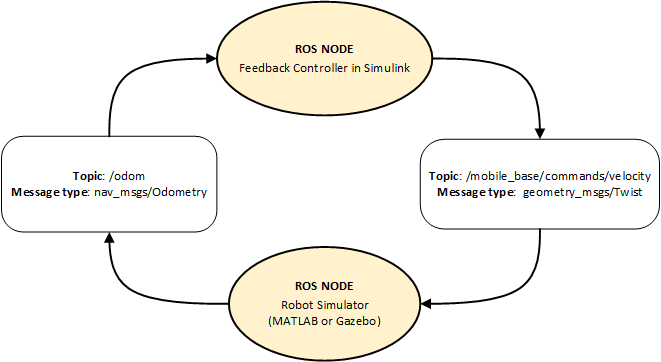
1. 运行仿真模型，发送速度指令到机器人，驱动机器人运行。

# [从Simulink中创建单独的ROS节点](https://www.ncnynl.com/archives/201909/3300.html)





注意，模型中总共有四个可调参数（由颜色块指示）。

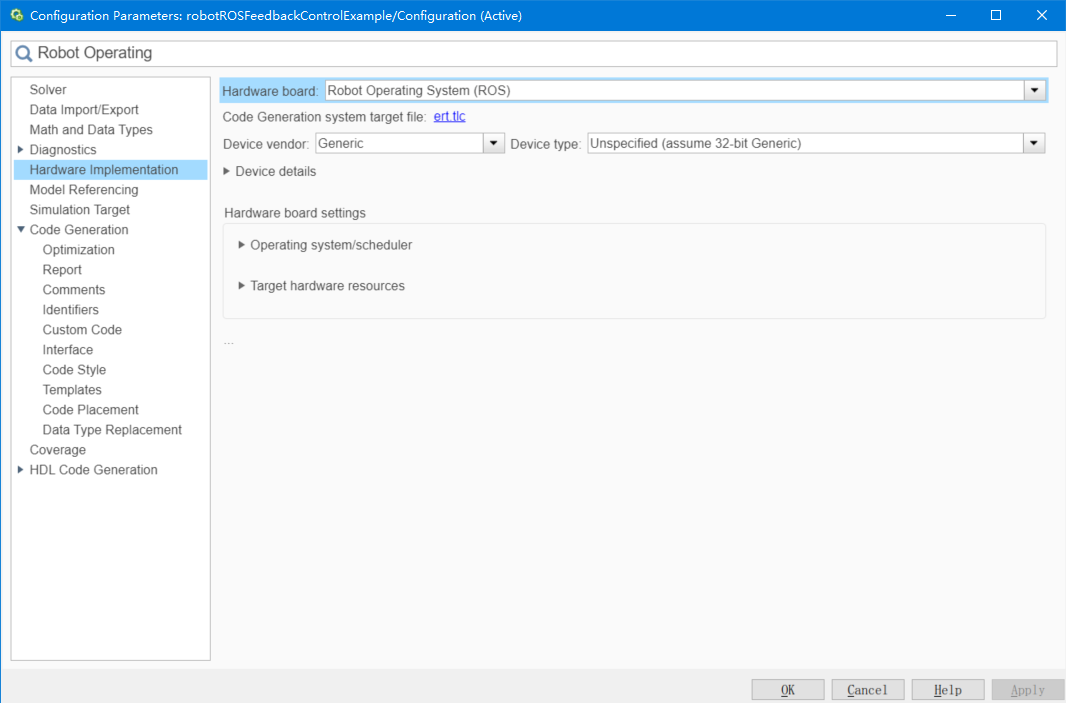


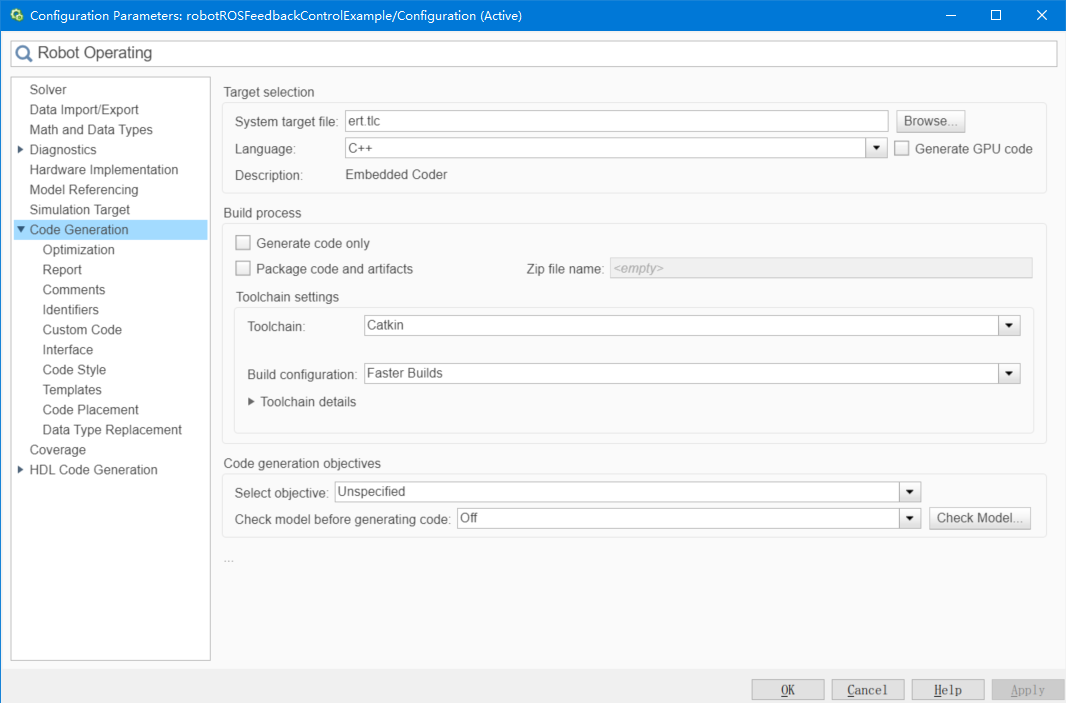
在仿真运行过程中，可以改变目标位置“Desired Position”模块和Proportional Controller子系统，双击“Linear Velocity(slider)”模块，通过调整滑块的值改变机器人的运动速度。

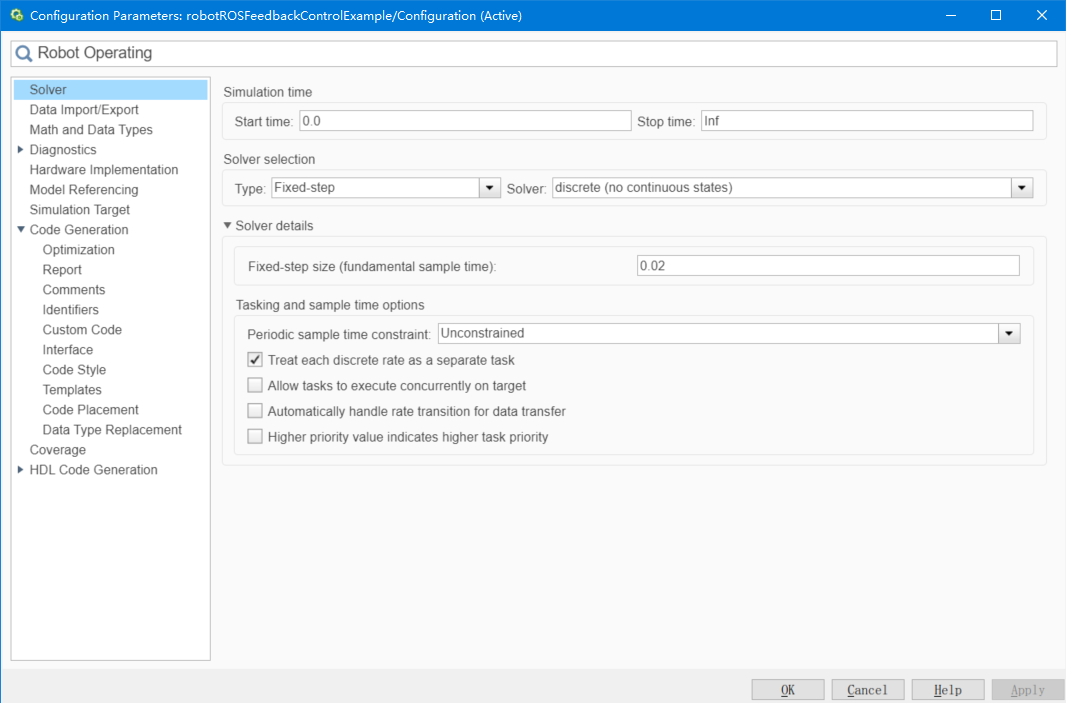
整个仿真系统具有4个可调节的参数：

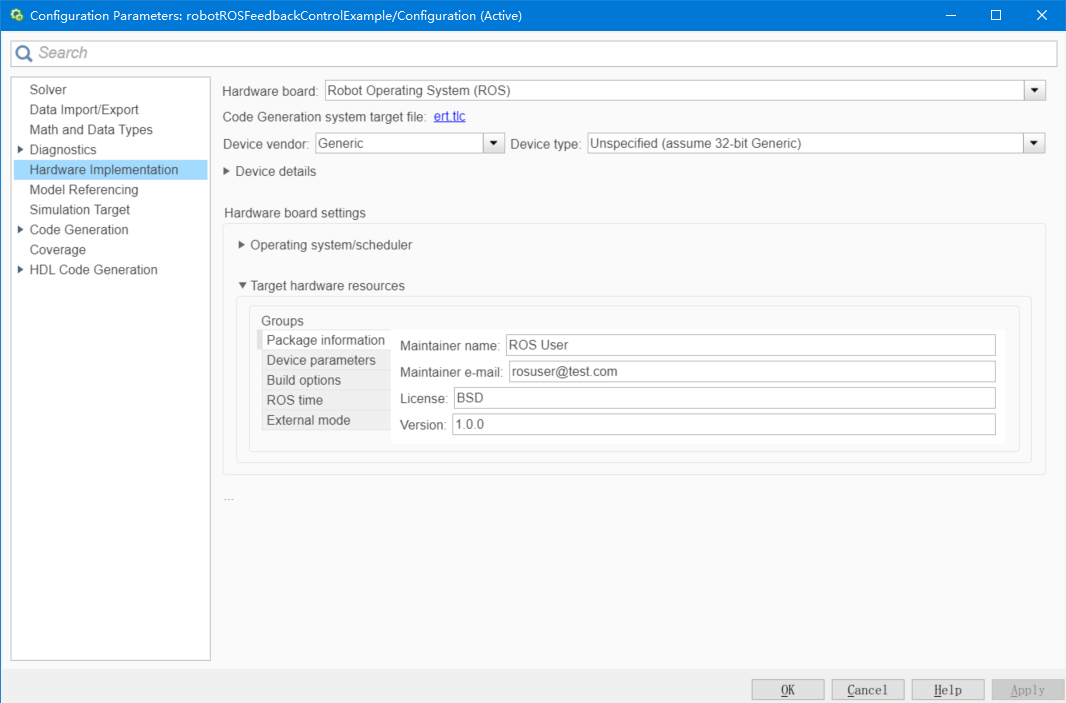
* 目标位置（模型的最高层）：目标位置的（X,Y）坐标；
* 距离阈值：当机器人距离目标位置的距离小于距离阈值，机器人则停止；
* 线性速度：机器人前进的线性速度；
* 增益：当纠正机器人方向是的比例增益。

1、发送速度指令到机器人



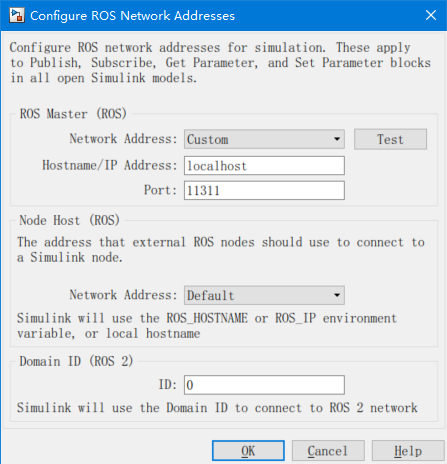






2 生成C++ROS节点

* 代码生成过程，首先准备为仿真准备模型，以确认所有的块都正确初始化。该准备工作需要已经启动有效的ROS主控节点。



* 在MATLAB命令行中输入“rosinit”，创建一个本地的ROS主控节点。
* 一旦编译完成，用户将会在当前文件夹看到两个文件：“robotROSConnectToRobotExample.tgz”（包含C++代码的归档文件）和“build\_ros\_model.sh”（用于提取和编译C++代码的shell脚本）。

3-转换文件到ROS系统

将生成的build\_ros\_model.sh和\*.tgz程序复制到机器人系统中(Ubuntu系统)的工作空间下。

mkdir -p ~/catkin\_ws\_simulink/src

cd ~/catkin\_ws\_simulink/src

catkin\_init\_workspace

cd ~

chmod +x build\_ros\_model.sh

./build\_ros\_model.sh RobotController.tgz <path\_to\_catkin\_ws>

