

Tutorial : Control of a S-Model Gripper Using the Modbus TCP Protocol

Description: This tutorial explains how to use the "robotiq_s_model_control" and "robotiq_modbus_tcp" packages to control a S-Model Gripper configured with the Modbus TCP protocol. The S-Model Gripper has 3 fingers. Please visit the website for more information on the [Robotiq Adaptive Robot Grippers](#).

Tutorial level : Beginner

1 - Prerequisites

This tutorial assumes that you have a S-Model Gripper configured with the Modbus TCP protocol. The Gripper should be connected to a network, which has been properly configured (you can validate the communication with the Gripper using the Windows Robotiq User Interface). For more information on the Gripper installation please look at the [Robotiq S-Model user manual](#). Then, make sure that the external dependency for the package "robotiq_modbus_tcp" has been installed. The dependency is the python package pyModbus. On Ubuntu Precise (12.04), it is simply installed using:

```
$ rosdep install robotiq_modbus_tcp
```

On other systems, it can be installed using:

```
$ easy_install -U pymodbus
```

It is possible that you are required to have administrator rights. If so, simply add "sudo" at the beginning of the command.

1.1 - Network configuration

Make sure that the Ethernet card in your PC to which your Gripper is connected has the following static IP network configuration :



Figure 1- Network Configuration

The static IP address of the PC (192.168.1.2) can be any IP address different from the hand's IP address (whose default is 192.168.1.11). To verify that your network is properly configured, you can ping the Gripper's IP. For example, if you are using the standard Gripper IP address, the command would be "ping 192.168.1.11".

2 - ROS Nodes to Control the Gripper

2.1 - Run the S-Model Driver Node

After launching the "roscore" command in a terminal, the driver node can be started. The Gripper is driven by the node "SModelTcpNode.py" contained in the package "robotiq_s_model_control". The IP address of the Gripper has to be provided as an argument.

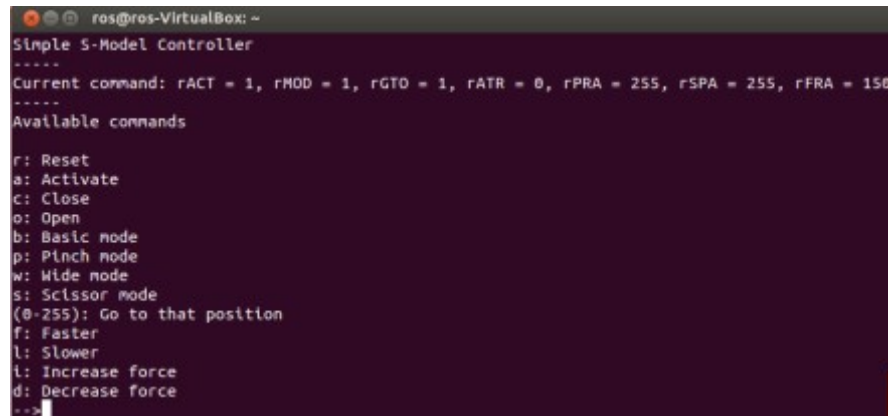
For example, the driver for controlling an S-Model Gripper with the IP address 192.168.1.11 is launched using the following command: "roslaunch robotiq_s_model_control SModelTcpNode.py 192.168.1.11".

```
$ roslaunch robotiq_s_model_control SModelTcpNode.py 192.168.1.11
```

2.2 - Run the S-Model Simple Controller Node

The driver listens for messages on "SModelRobotOutput" using the "SModel_robot_output" msg type. The messages are interpreted and commands are sent to the Gripper accordingly. A simple controller node is provided which can be run (at another terminal) using "roslaunch robotiq_s_model_control SModelSimpleController.py".

```
$ roslaunch robotiq_s_model_control SModelSimpleController.py
```

A screenshot of a terminal window titled "ros@res-VirtualBox: ~". The terminal displays the output of the "SModelSimpleController.py" node. It shows the current command values: rACT = 1, rMOD = 1, rGTO = 1, rATR = 0, rPRA = 255, rSPA = 255, rFRA = 150. Below this, it lists available commands: r: Reset, a: Activate, c: Close, o: Open, b: Basic mode, p: Pinch mode, w: Wide mode, s: Scissor mode, (0-255): Go to that position, f: Faster, l: Slower, i: Increase force, d: Decrease force. The prompt "-->" is visible at the bottom.

```
ros@res-VirtualBox: ~
Simple S-Model Controller
-----
Current command: rACT = 1, rMOD = 1, rGTO = 1, rATR = 0, rPRA = 255, rSPA = 255, rFRA = 150
-----
Available commands
r: Reset
a: Activate
c: Close
o: Open
b: Basic mode
p: Pinch mode
w: Wide mode
s: Scissor mode
(0-255): Go to that position
f: Faster
l: Slower
i: Increase force
d: Decrease force
-->
```

Figure 2 - Simple Controller Node

The "SModel_robot_output" msg type is simply composed of the robot output variables described in the [Robotiq S-Model user manual](#). The simple controller node can therefore be modified to send custom commands to the Gripper.

2.3 - Run the S-Model Status Listener Node

In the package "robotiq_s_model_control", there is also a node for listening to and interpreting the status of the Gripper. The driver publishes the status of the Gripper on "SModelRobotInput" using the "SModel_robot_input" msg type. The msg type is composed of the robot input variables described in the [Robotiq S-Model user manual](#). The status listener node can be run (at another terminal) using the following command: "roslaunch robotiq_s_model_control SModelStatusListener.py".

```
$ roslaunch robotiq_s_model_control SModelStatusListener.py
```

```
res@ros-VirtualBox: ~
-----
S-Model status interpreter
-----
gACT = 1: Gripper activation
gMOD = 1: Pinch Mode
gGTO = 1: Go to Position Request
gIMC = 3: Activation and mode change are completed
gSTA = 1: Gripper is stopped. One or two fingers stopped before requested position
gDTA = 2: Finger A has stopped due to a contact while closing
gDTB = 3: Finger B is at requested position
gDTC = 2: Finger C has stopped due to a contact while closing
gDTS = 3: Scissor is at requested position
gFLT = 0: No Fault
gPRA = 255: Echo of the requested position for the Gripper (or finger A in individual mode): 255/255
gPOA = 108: Position of Finger A: 108/255
gCUA = 0: Current of Finger A: 0 mA
gPRB = 0: Echo of the requested position for finger B: 0/255
gPOB = 114: Position of Finger B: 114/255
gCUB = 0: Current of Finger B: 0 mA
gPRC = 0: Echo of the requested position for finger C: 0/255
gPOC = 108: Position of Finger C: 108/255
gCUC = 0: Current of Finger C: 0 mA
gPRS = 0: Echo of the requested position for the scissor axis: 0/255
gPOS = 220: Position of the scissor axis: 220/255
gCUS = 0: Current of the scissor axis: 0 mA
```

Figure 3 - Status Listener Node