RTM と ROS を用いた物体操作システム

操作マニュアル

名城大学メカトロニクス工学科

ロボットシステムデザイン研究室

2020 年 12 月 14 日

目次

1. はじめに

1.1. 目的

本書の目的は，RTM と ROS を用いた物体操作システムの実行および操作方法を解説することである．

1.2. 本書を読むにあたって

本書は RT ミドルウェアに関する基礎知識を有した利用者を対象としている．また， 各 RTC の詳細な仕様等については同ファイル内の仕様解説マニュアルを参考にすること．

1.3. 動作環境

本 RTC の動作確認環境を以下に示す．

OS	Ubuntu 18.04
RT ミドルウェア	OpenRTM-aist-2.0
ROS	ROS melodic

1.4. 開発環境

本 RTC の開発環境を以下に示す．

OS	Ubuntu 18.04
言語	C++
RT ミドルウェア	OpenRTM-aist-2.0

2. シミュレータ，実機操作の準備

2.1. ROS で MOTOMAN-GP8 を利用するための環境構築

今回の物体操作システムでは安川電機の産業用ロボットである MOTOMAN-GP8 を利用した．MOTOMAN-GP8 を ROS で利用するため，以下のサイトを参考にして環境構築を行うこと．実機がなくシミュレーションでのみ検証の場合は「MoveIt!によるモーション作成」の項目まで行うこと．

―ROS で Motoman GP8 を利用するための環境構築と動作確認

http://www1.meijo-u.ac.jp/~kohara/cms/technicalreport/ros_motoman_gp8_setup

2.2. サンプルスクリプト(manip.py)の編集

ROS のマニピュレータを動作させるためにサンプルスクリプトを編集する必要がある．ダウンロードしたサンプルスクリプト(manip.py)を開き，14 行目にある”manip”の部分を動かしたいマニピュレータに応じて変更することで様々な ROS 対応のマニピュレータを動作させることが可能となる．

```
1  #! /usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import rospy
5  import math
6  import copy
7  import moveit_commander
8  import moveit_msgs.msg
9  from std_msgs.msg import Float32MultiArray
10 from sensor_msgs.msg import JointState
11 from geometry_msgs.msg import Quaternion, Pose, PoseStamped, Vector3
12
13 #grobal
14 manip = moveit_commander.MoveGroupCommander("manip")
```

図 1　サンプルスクリプト

今回の物体操作システムでは MOTOMAN-GP8 を利用したため，上記部分を”motoman_gp8”となるように書き換え，保存を行う．

```
manip = moveit_commander.MoveGroupCommander(“motoman_gp8”)
```

2.3. ROS のサンプルスクリプト配置

上記の ROS で MOTOMAN-GP8 を利用するための環境構築，サンプルスクリプトの編集が完了した後に，以下で作成する scripts というディレクトリ内にサンプルスクリプト(manip.py)を配置する．

```
~$ cd
~$ cd catkin_ws/src/motoman/motoman_gp8_moveit_config
~/catkin_ws/src/motoman/motoman_gp8_moveit_config$ mkdir scripts
~/catkin_ws/src/motoman/motoman_gp8_moveit_config$ cd scripts
~/catkin_ws/src/motoman/motoman_gp8_moveit_config/scripts$ (このディレクトリ内にサンプルスクリプトを配置する)
```

またサンプルスクリプト配置後に実行権限を与えるため，以下のコマンドを実行する．

```
~/catkin_ws/src/motoman/motoman_gp8_moveit_config/scripts$ chmod +x manip.py
```

2.4. RTC の設定ファイル(rtc.conf)の編集

RTMtoROS コンポーネントにおいて，RTC と ROS のノードを接続するためには RTC の設定ファイルを編集する必要がある．ダウンロードした RTC の設定ファイル(rtc.conf)を開き ros.roscore.host=XXX.XXX.XXX.XXX の X 部分を自身の PC の IP アドレスに書き換え，保存を行う．

```
1 manager.modules.load_path: /usr/local/lib/openrtm-2.0/transport/
2 manager.modules.preload: ROSTransport.so
3 manager.components.preconnect: RTMtoROS0.movePTPJointAbs?interface_type=ros&mar
shaling_type=ros:std_msgs/Float32MultiArray&ros.topic=movePTPJointAbs&ros.node.
name=RTMtoROS&ros.roscore.host=XXX.XXX.XXX.XXX&ros.roscore.port=11311,RTMtoROS0
.getHome?interface_type=ros&marshaling_type=ros:std_msgs/Float32MultiArray&ros.
topic=getHome&ros.node.name=RTMtoROS&ros.roscore.host=XXX.XXX.XXX.XXX&ros.rosco
re.port=11311
```

図 2　設定ファイル

自身の PC の IP アドレスを確認する方法として，ターミナルにて以下のコマンドを入力することで確認ができる．

```
~$ hostname -I
```

または

```
~$ ifconfig
```

2.5. ダウンロードしたRTCのビルド

ダウンロードしたRTC(ManipulatorControlSample，RTMtoROS)コンポーネントディレクトリの階層で，以下のコマンドを実行しビルドを行う．

```
~/$ mkdir build
~/$ cd build
~/$ cmake ..
~/$ make
```

3. シミュレータを用いたシステムの操作方法

3.1. roscoreの起動

ターミナルにてroscoreを実行する．

```
~$ roscore
```

3.2. シミュレーション(rviz)の起動

新規ターミナルまたは新規タブを開き，以下のコマンドを実行する．

```
~$ roslaunch motoman_gp8_moveit_config demo.launch
```

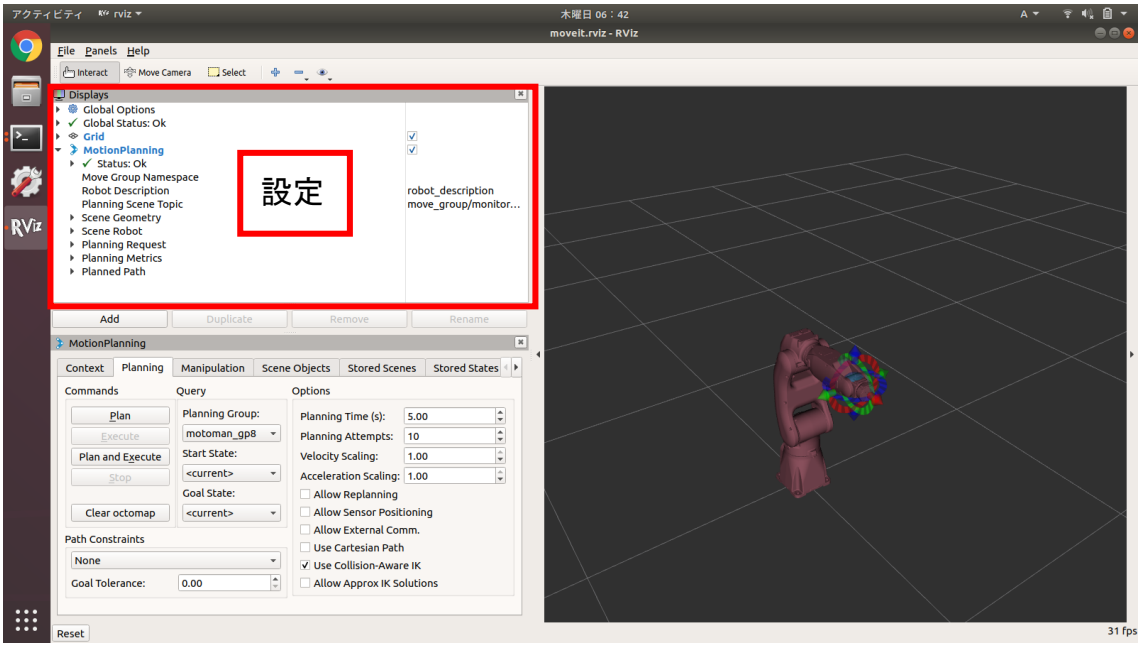

図3　rviz

このままシミュレーションを行うとロボットの動作が見えづらいため，rviz の設定を以下のように変更する．

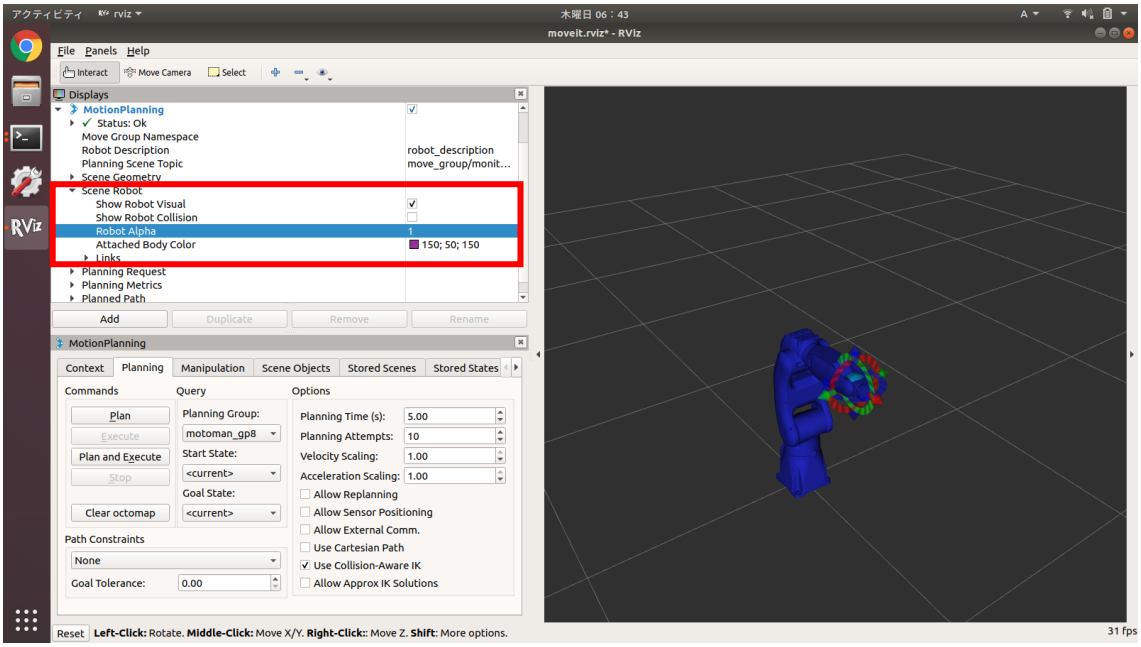

図 4　ロボットの透過設定

>Scene Robot >Robot Alpha を 0.5 から 1 に変更することで，ロボットが透過されなくなる．

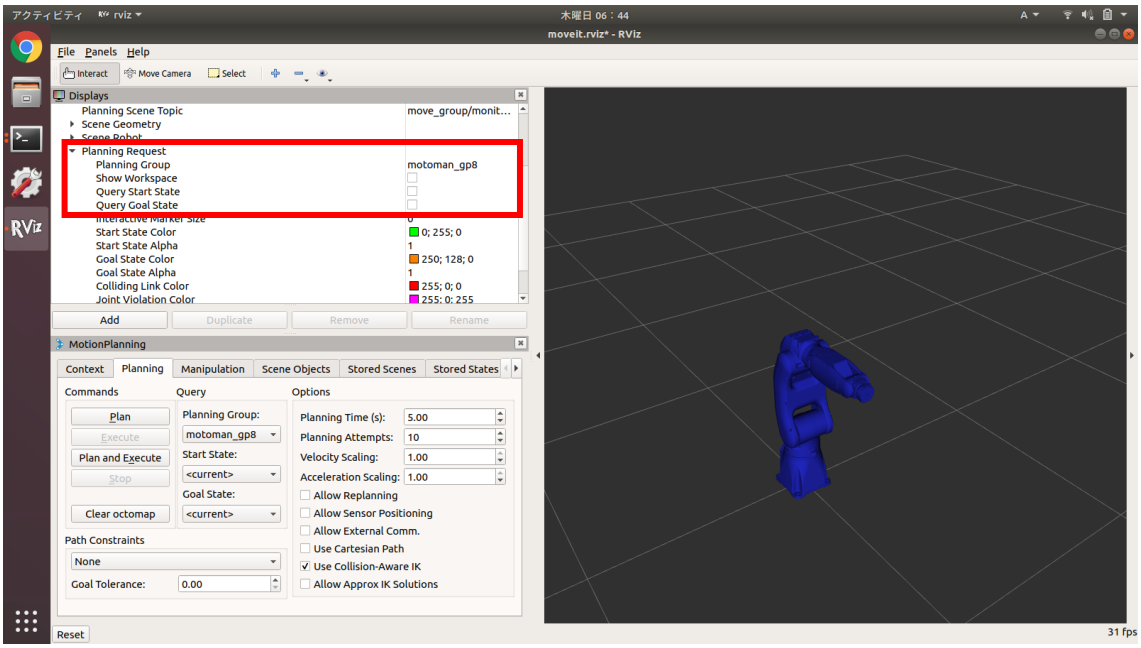

図 5　インタラクティブマーカー表示設定

>Planning Request>Query Goal State のチェックを外すことで，インタラクティブマーカーを非表示にできる．

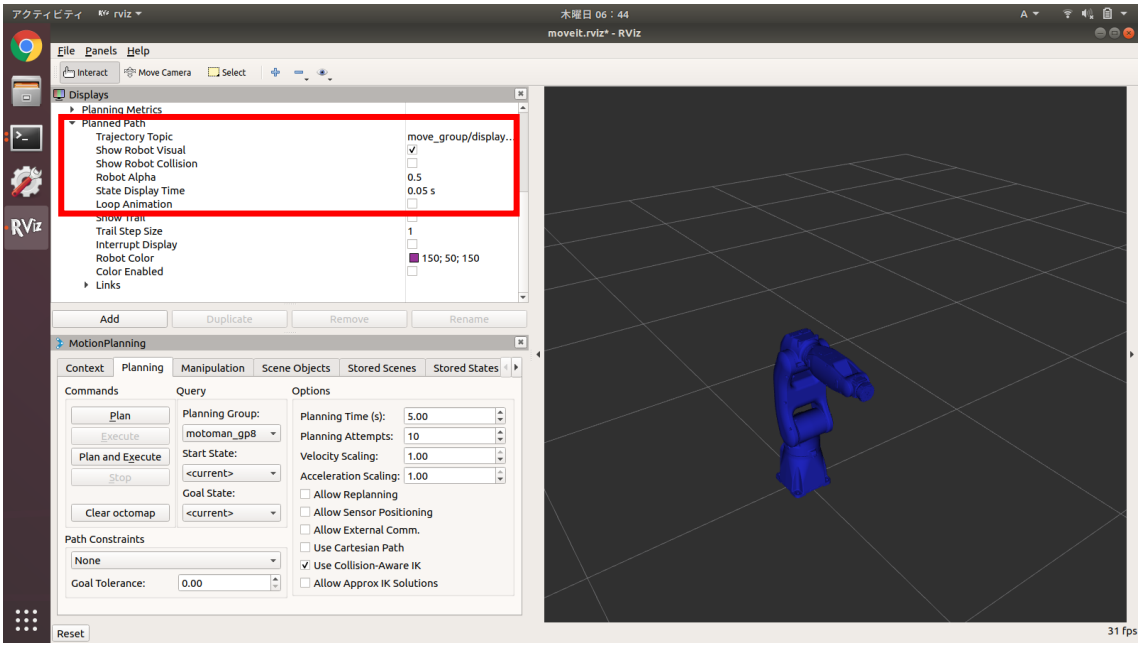

図 6　軌道アニメーションループ設定

>Planned Path>Loop Animation のチェックを外すことで，軌道アニメーションがループしなくなる．

3. 3. サンプルスクリプトの実行

新規ターミナルまたは新規タブを開き，以下のコマンドを実行する．

```
~$ rosrun motoman_gp8_moveit_config manip.py
```

以下の図のように Waiting for jointPos or carPos ...と表示されずエラー等が出る場合は，roscore やシミュレーション(rviz)が正常に起動できているか，そしてサンプルスクリプトが正しく編集されているか確認を行う．

図 7　サンプルスクリプト実行時

3.4. eclipse の実行

① 新規ターミナルを開き，ネーミングサービスを実行する．

```
~$ rtm-naming
```

② eclipse を起動する．

③ openrtp が立ち上がったら，右上の「パースペクティブを開く」から RT System Editor を選択する．（図 8~10）

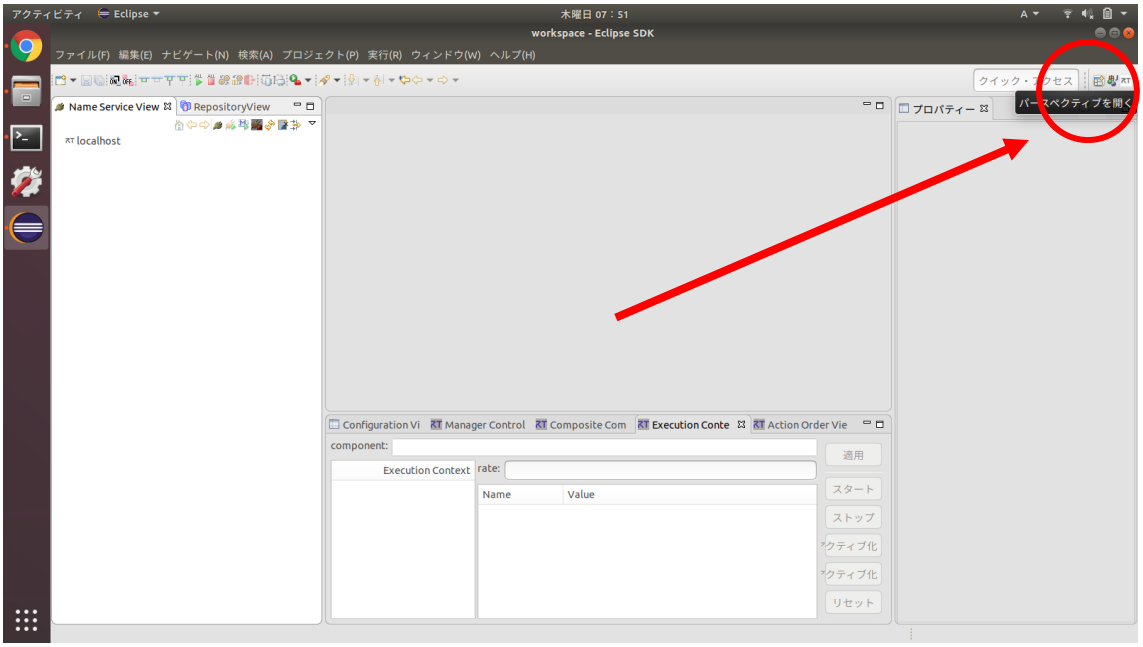

図 8　eclipse 起動画面

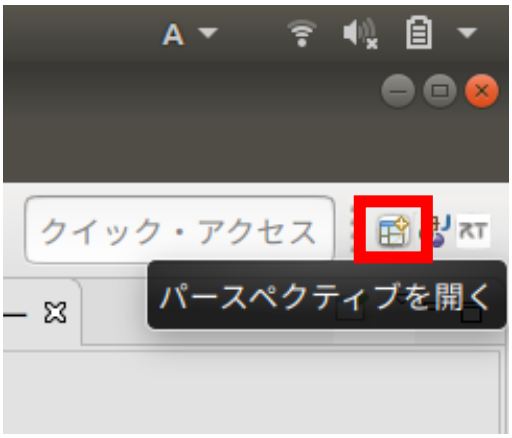

図 9　パースペクティブを開く

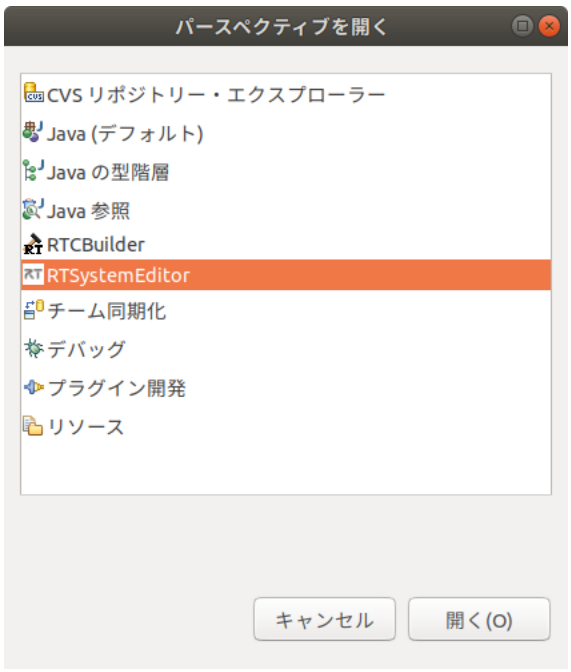

図 10　RT System Editor

④ ネームサービスに何もない場合，図 11 のように左上のネームサーバーの追加から localhost を追加する．（このとき失敗する場合はネーミングサービスが立ち上がっていないため①の rtm-naming を実行する．）

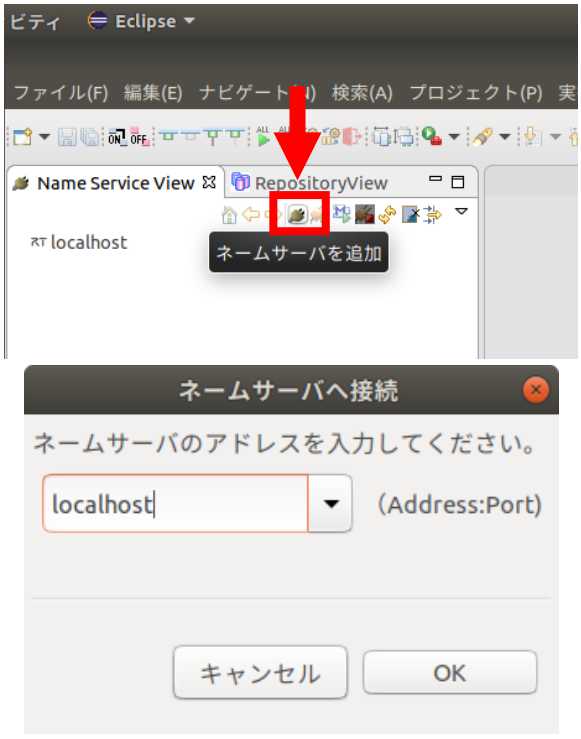

図 11　ネームサバーの追加

⑤ 左上の RT System Editor のシステムダイアグラムを ON にする.

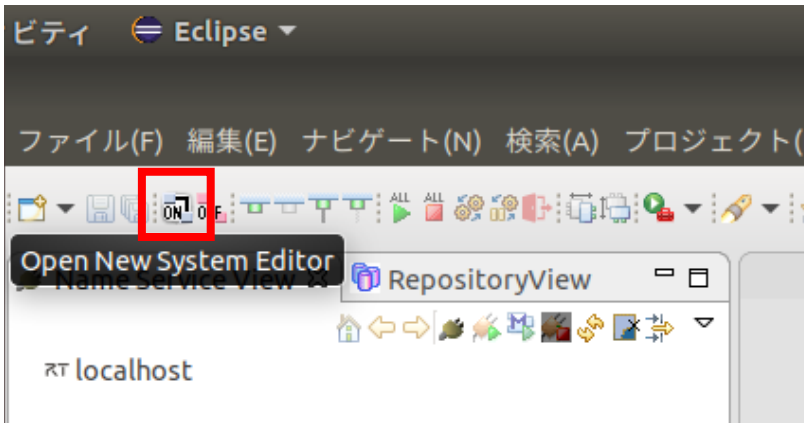

図 12　システムダイアグラムの ON

3.5. コンポーネントの実行

① ManipulatorControlSample の実行

ビルドを行った ManipulatorControlSample を実行する. ターミナルで以下のコマンドを実行する.

```
~$ cd ManipulatorControlSample/build/src
~/ManipulatorControlSample/build/src$ ./ManipulatorControlSampleComp
```

② RTMtoROS の実行

ビルドを行った RTMtoROS を実行する. ターミナルで以下のコマンドを実行する.

```
~$ cd RTMtoROS
~/RTMtoROS/$ build/src/RTMtoROSComp -f rtc.conf
```

3.6. コンポーネントの接続

コンポーネントをシステムダイアグラムに表示し，以下の図 13 のようにポートが接続されていることを確認する．

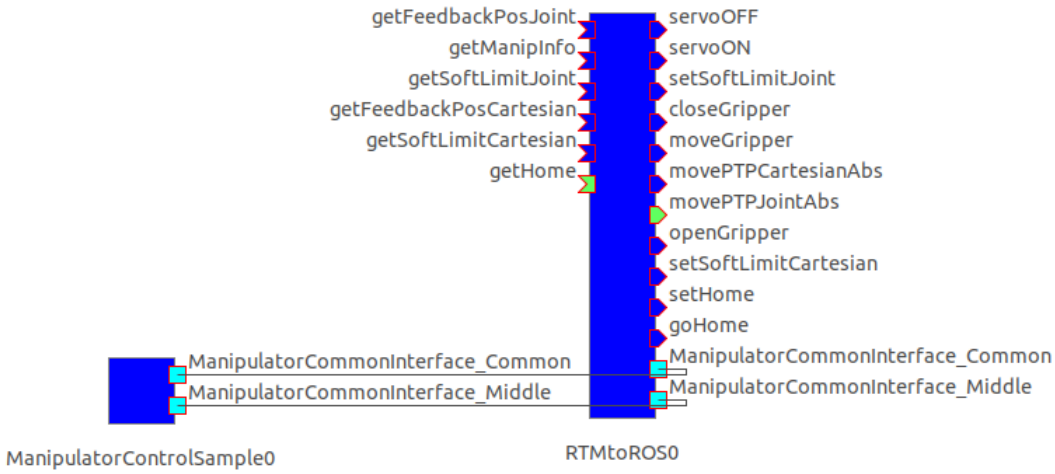

図 13　コンポーネントの接続図

3.7. RTM と ROS の接続確認

システムを Activate する前に RTM と ROS の接続確認を行う．

① 新規ターミナルにて以下のコマンドを実行する．

```
~$ rqt_graph
```

新規ウィンドウが立ち上がり，rqt_graph が表示される．

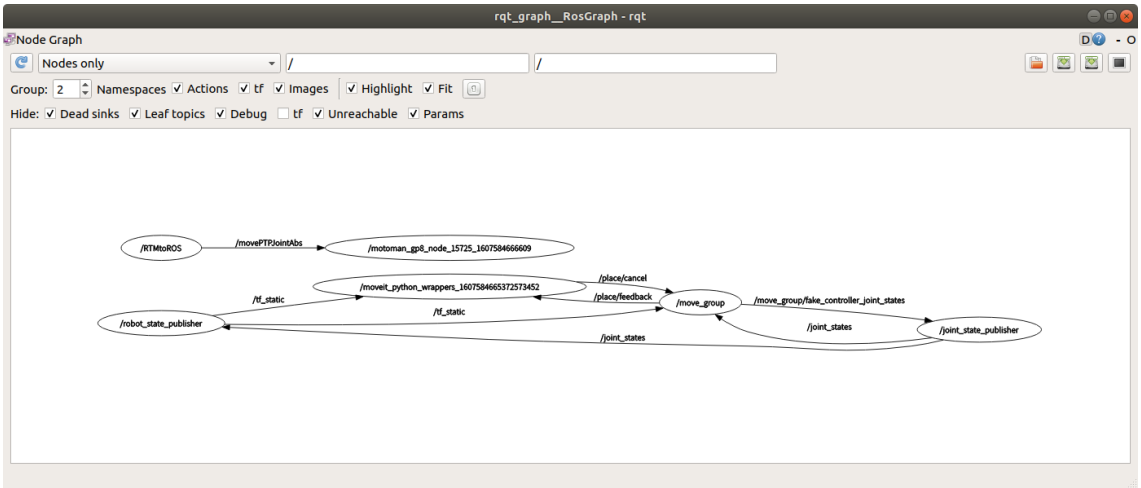

図 14　rqt_graph

② rqt_graph にすべてのノード/トピックが表示されるよう変更する.

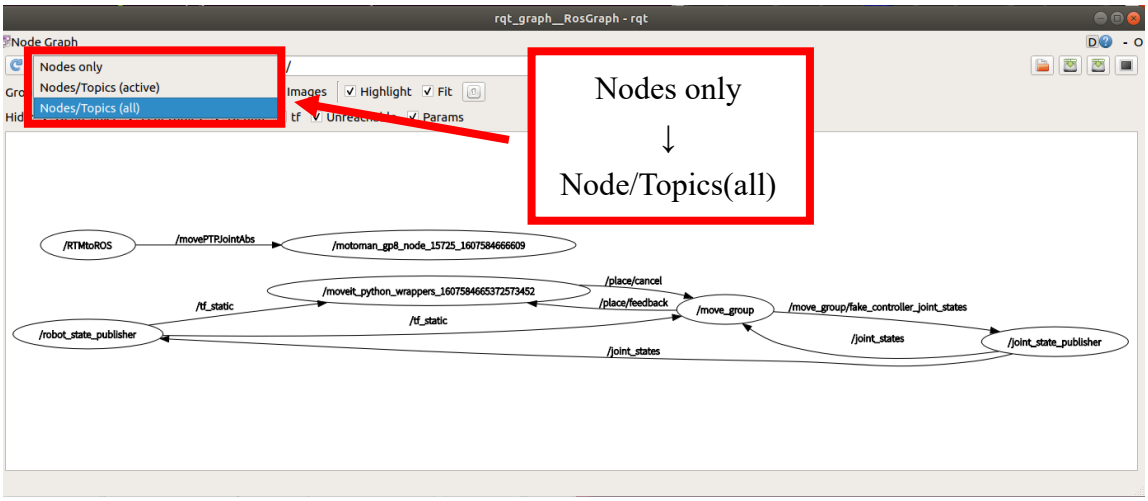

図 15　rqt_graph の表示設定変更

③ rqt_graph に以下の図のように RTMtoROS ノードと motoman ノードが接続されていることを確認する.

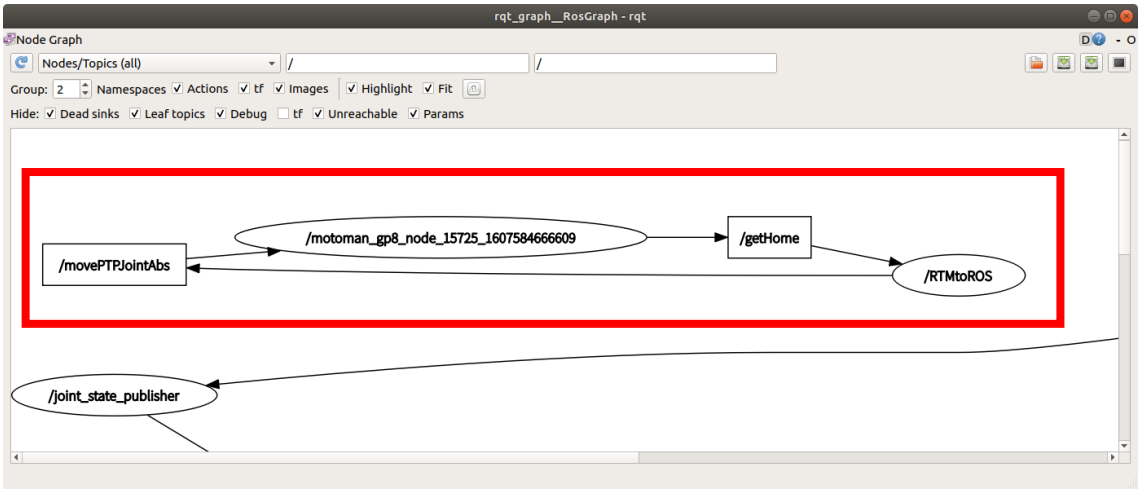

図 16　RTM と ROS の接続確認

RTM と ROS が接続されてない場合，eclipse にてシステムを All Exit(図 15)し，以下の項目を確認する.

1) RTC の設定ファイル(rtc.conf)の IP アドレスを自身の PC のものに変更している.
2) ROS→RTM の順にスクリプトやコンポーネントを実行している.

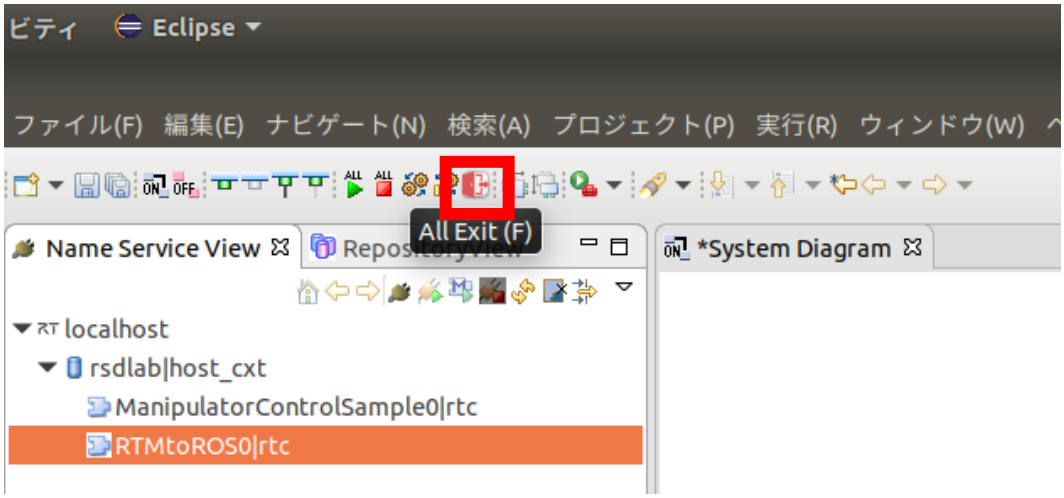

図 17　All Exit

3.8. システムの Activate・Deactivate

上記の rqt_graph による RTM と ROS の接続が確認できたら図 18 のように Activate で実行する．このとき，ManipulatorControlSample を先に Activate するとエラーとなるため，Activate Systems するか，RTMtoROS を先に Activate してから ManipulatorControlSample の順に Activate する．

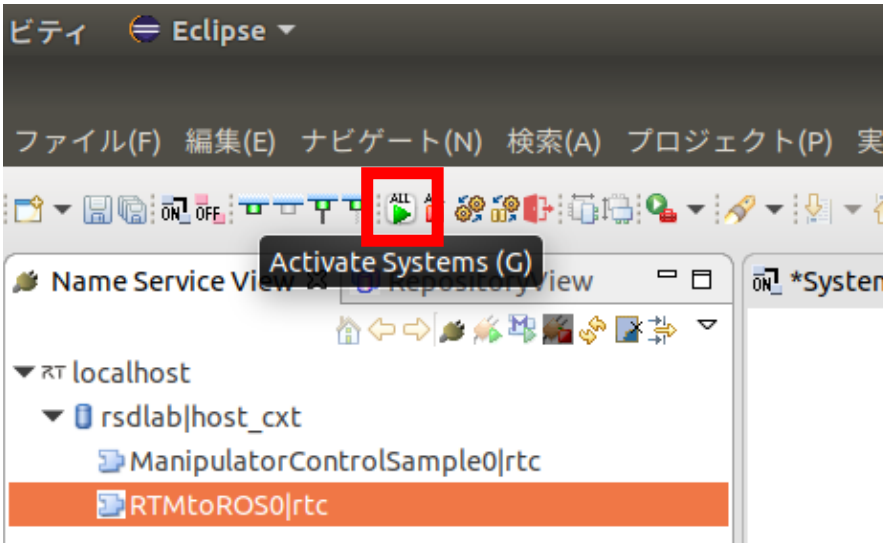

図 18　Activate Systems

また，システムを Deactivate する際は，ManipulatorControlSample にて「3：終了」を選択した後に，図 19 のように Deactivate する．

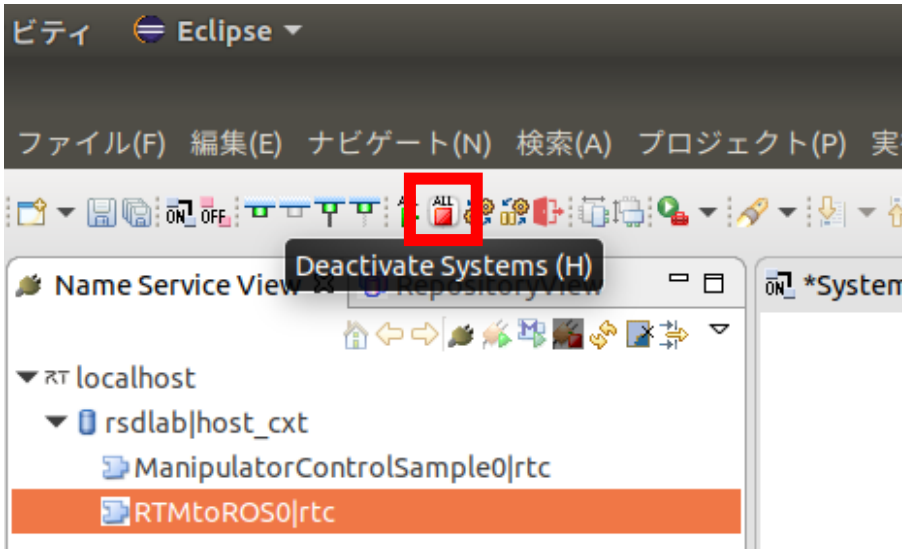

図 19　Deactivate Systems

3.9. ManipulatorControlSample のコマンド解説

システムの Activate に成功すると，ManipulatorControlSample を実行しているターミナルにて以下の図 20 のように表示される．

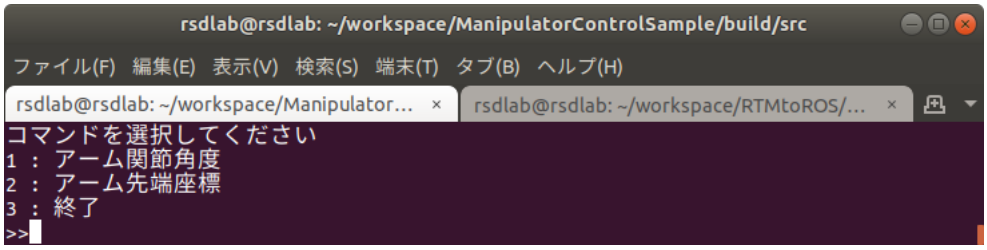

図 20　ManipulatorControlSample 実行画面

コマンドはそれぞれ以下の機能を持つ．

1：アーム関節角度

MOTOMAN-GP8 の各関節(第 1~6 関節)の角度[rad]を指定して回転させる．

2：アーム先端座標

絶対座標で入力した座標にアームの先端を移動させる．単位は[mm]

3：終了

コンポーネントを終了する．このコマンドを実行した後に，eclipse にて Deactivate を行う(3.9. システムの Activate・Deactivate 参照)．

4. 実機を用いたシステムの操作方法

実機でのシステム操作のためには以下のサイトの環境環境構築をすべて行う必要がある．そしてこれからの説明は上記サイトを参考に実機の MOTOMAN-GP8 と接続されいるものとする．

―ROS で Motoman GP8 を利用するための環境構築と動作確認

http://www1.meijo-u.ac.jp/~kohara/cms/technicalreport/ros_motoman_gp8_setup

「3. シミュレーションを用いたシステムの操作方法」での，「3.2. シミュレーション(rviz)の起動」にて実行したコマンドを以下のように変更する．

```
~$roslaunch motoman_gp8_moveit_config moveit_planning_execution.launch
sim:=false robot_ip:=192.168.255.1 controller:=yrc1000
```

以降の手順に関しては，「3.3. サンプルスクリプトの実行」からシミュレーション時と同様に進める．

ただし，「3.9. システムの Activate・Deactivate」を行う前に， MOTOMAN-GP8 をサーボオンにする必要があるため以下のコマンドを実行する．

```
~$ rosservice call /robot_enable
```

上記コマンド実行後に，サーボオンがされる(カチッと音が鳴る)．

サーボオンを行った後に「3.9. システムの Activate・Deactivate」を行う．