

# ROS体験コース

---

「USB一本でインストールなしにROS環境を体験」

NEDOプロジェクトを核とした人材育成、産学連携等の総合的展開  
「システム・インテグレーションを加速するロボット共通ソフトウェア技術を維持  
・普及・発展させていくための人材の育成・交流・研究の活性化に係る特別講座」

\*なお、本資料は、NEDOロボット活用型市場化技術開発プロジェクト(以下、市場化プロジェクト)で開発されたLive USBを用いたROS環境であるROS/Kinetic USBを対象としております

2021年1月29日公開

# 本体験コースの対象となる方

- ROSって聞いたことがある
- すごく大まかだけどどんなものかは知っている
  - まったく知らない人は、ネット検索をして頂くか、入門書が多数ありますので、まずは、そちらをご参照ください
  - 本講座のHPにも非常に簡単ですが資料があります
    - ・ [https://robo-marc.github.io/moveit\\_tutorial/](https://robo-marc.github.io/moveit_tutorial/)
- インストールは面倒だけど、とりあえずさわってみたい
  - ・ どのような計算機でもOKというわけではありませんが、Windowsが起動するPCがあれば、とりあえずインストール不要でROS環境+NEDO市場化プロジェクトの成果を体験できます
  - ・ 最低でもメモリは4GB. できれば8GB以上ある機種でお願いします
  - ・ 一時的でよろしければ、制約は多いですが、VMWareやMacのParallels上でも動作します

# 本ROS体験コースで試せること

- 体験コースで用いるUSB環境
  - ROS環境をインストー済みのUbuntu Linux 16.04の環境
  - 本USBを用いてOSを立ち上げることで、インストール、環境設定を行わずにNEDO市場化プロジェクトで開発したロボットプラットフォームソフトウェアのデモンストレーションが実行可能です
- 本体験コースのビデオで紹介するのは、MoveIt! \*<sup>1</sup>を用いたロボットのオフラインティーチングデモ
- 対象とするロボット: 川崎重工業社製 duAro
  - 他にも、本USB環境には、富士ソフト (株) FSRobo, THK (株) SEED-NoId, カワダロボティクス (株) NEXTAGEを対象としたデモ実行環境が含まれております

\*<sup>1</sup> MoveIt!: ROS環境で動作するオフラインティーチングツールの名称

# 体験版USBの入手方法

---

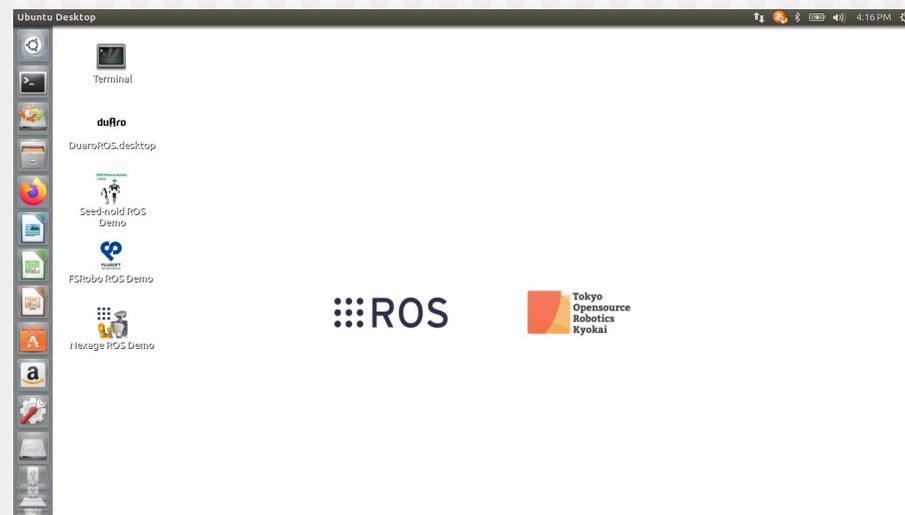
- 郵送による体験USBの入手（在庫がなくなり次第、終了します）
- USBメモリーイメージのダウンロードとUSBの作成
  - [「https://robo-marc.github.io/tutorials/si2020」](https://robo-marc.github.io/tutorials/si2020)を参照
- USBメモリーイメージなどを使用してVMWare等の仮想環境でも起動が可能です
  - 一部、機能制限や不具合がありますが、即座に試せるメリットがあります
- 詳細は、別途配布のPDF資料をご覧ください

# 体験USB環境の起動(終了)方法

- Windows PC上で直接起動する場合
  - 電源OFFの状態ですべてUSBを計算機のUSBコネクタに挿入し、その後、電源を投入して起動します
  - これで起動できない場合は、お持ちの計算機の外部メディアからの起動方法を調べて、それに従ってください
- OSが起動すると自動ログインが行われて以下のような画面になります

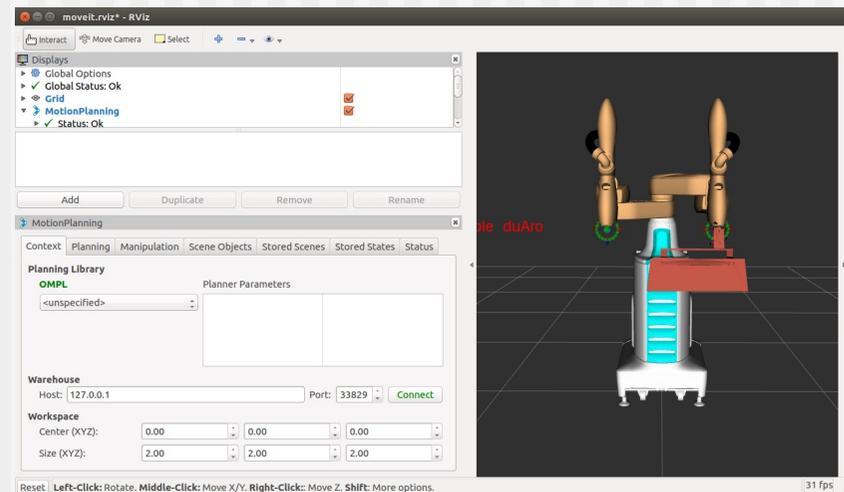
各アイコンをクリックして頂くことで、それぞれのデモンストレーションが実行できます

終了する場合は、右上の電源マークをクリックするとShutdownメニューがでますので、それを選択して下さい



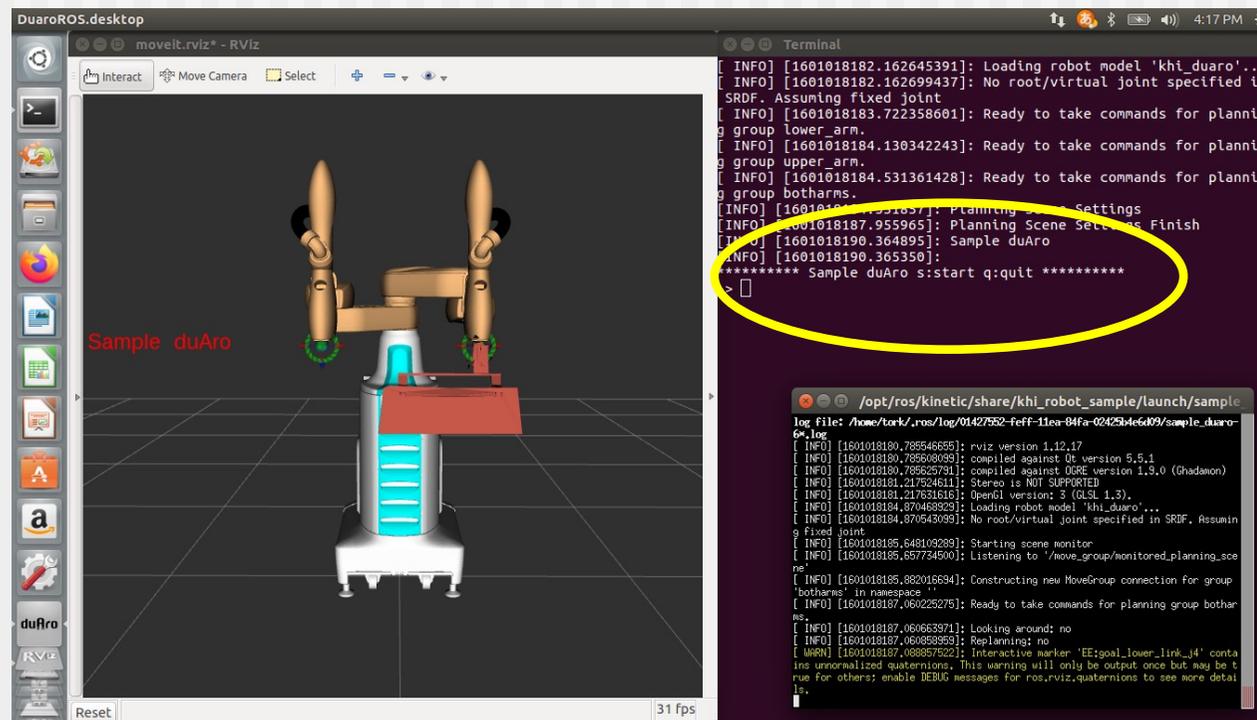
# duAro用環境の立ち上げ

- デスクトップ上にあるロボットの名称のついたアイコンをダブルクリック
  - これで、それぞれのロボット用の環境が立ち上がります
- 例として川崎重工業社製duAroの環境を立ち上げてみます



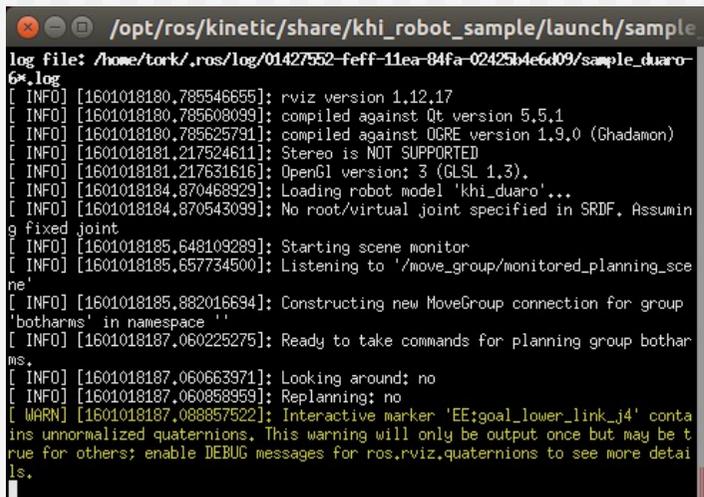
# 川崎重工業社製duAroのROS環境

- 下記の図にあるように、「Sample duAro s:start q:quit」と出力されたら起動が完了しています
  - メッセージの通り、「s」と入力すると自動で起動するデモンストレーションを見ることができます

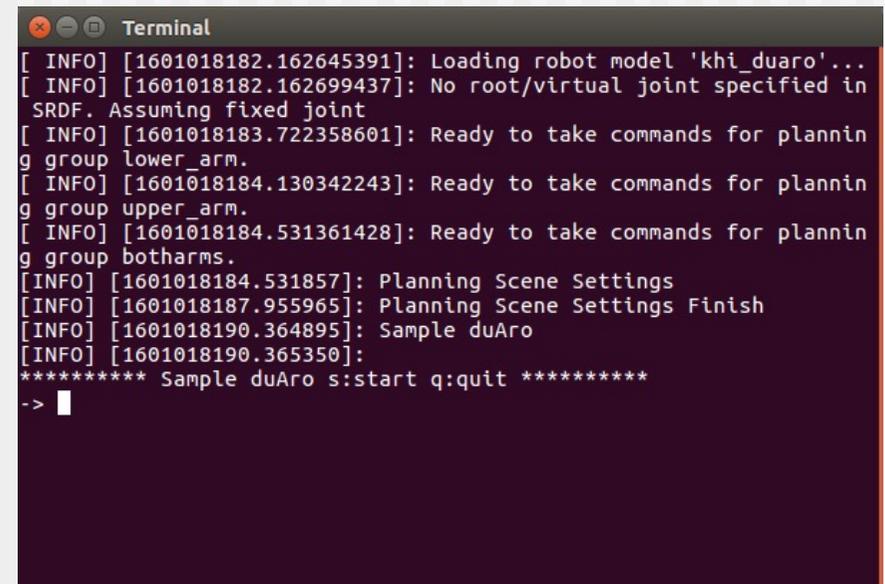


# 川崎重工業社製duAroのROS環境

- 関連するソフトウェアを実行しているターミナル(自動で起動, 触らなくて良い)
- 自動起動するターミナル:これが, ROS環境にコマンドを与えるユーザーI/Fになる



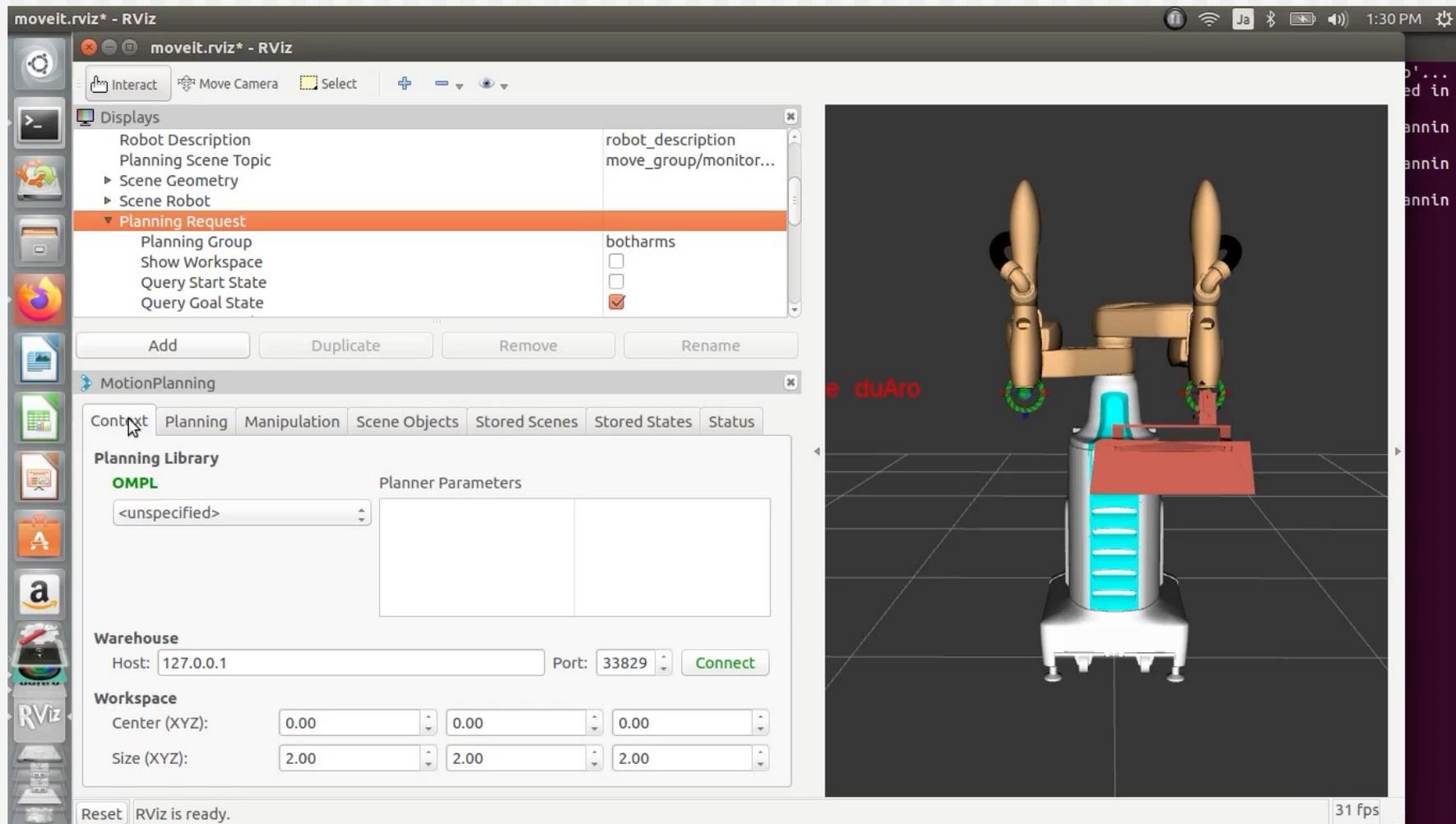
```
/opt/ros/kinetic/share/khi_robot_sample/launch/sample
log file: /home/tork/.ros/log/01427952-feff-11ea-84fa-02425b46d09/sample_duaro-6*.log
[ INFO] [1601018180.785546695]: rviz version 1.12.17
[ INFO] [1601018180.785608099]: compiled against Qt version 5.5.1
[ INFO] [1601018180.785625791]: compiled against OGRE version 1.9.0 (Ghadamon)
[ INFO] [1601018181.217524611]: Stereo is NOT SUPPORTED
[ INFO] [1601018181.217631616]: OpenGL version: 3 (GLSL 1.3).
[ INFO] [1601018184.870468929]: Loading robot model 'khi_duaro'...
[ INFO] [1601018184.870543099]: No root/virtual joint specified in SRDF. Assuming fixed joint
[ INFO] [1601018185.648109289]: Starting scene monitor
[ INFO] [1601018185.657734500]: Listening to '/move_group/monitored_planning_scene'
[ INFO] [1601018185.882016694]: Constructing new MoveGroup connection for group 'botharms' in namespace ''
[ INFO] [1601018187.060225275]: Ready to take commands for planning group botharms.
[ INFO] [1601018187.060663971]: Looking around: no
[ INFO] [1601018187.060858959]: Replanning: no
[ WARN] [1601018187.088857522]: Interactive marker 'EE:goal_lower_link_j4' contains unnormalized quaternions. This warning will only be output once but may be true for others; enable DEBUG messages for ros.rviz.quaternions to see more details.
```



```
Terminal
[ INFO] [1601018182.162645391]: Loading robot model 'khi_duaro'...
[ INFO] [1601018182.162699437]: No root/virtual joint specified in SRDF. Assuming fixed joint
[ INFO] [1601018183.722358601]: Ready to take commands for planning group lower_arm.
[ INFO] [1601018184.130342243]: Ready to take commands for planning group upper_arm.
[ INFO] [1601018184.531361428]: Ready to take commands for planning group botharms.
[INFO] [1601018184.531857]: Planning Scene Settings
[INFO] [1601018187.955965]: Planning Scene Settings Finish
[INFO] [1601018190.364895]: Sample duAro
[INFO] [1601018190.365350]:
***** Sample duAro s:start q:quit *****
-> █
```

デモ環境を終了したい場合は, このコマンドI/Fのターミナルをクローズしてください

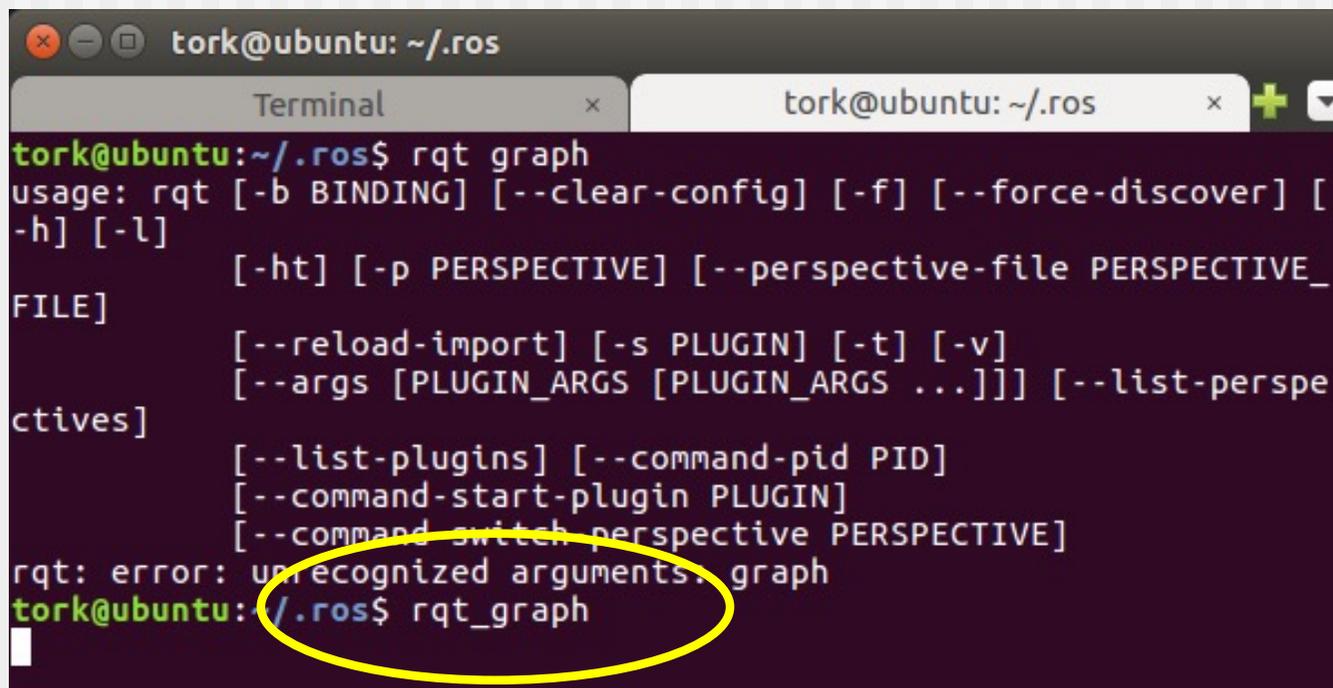
# duAroを題材にした簡単なオフラインティーチング(動画)



# duAroを題材にROSの機能体験

## ■ rqt-graphの表示

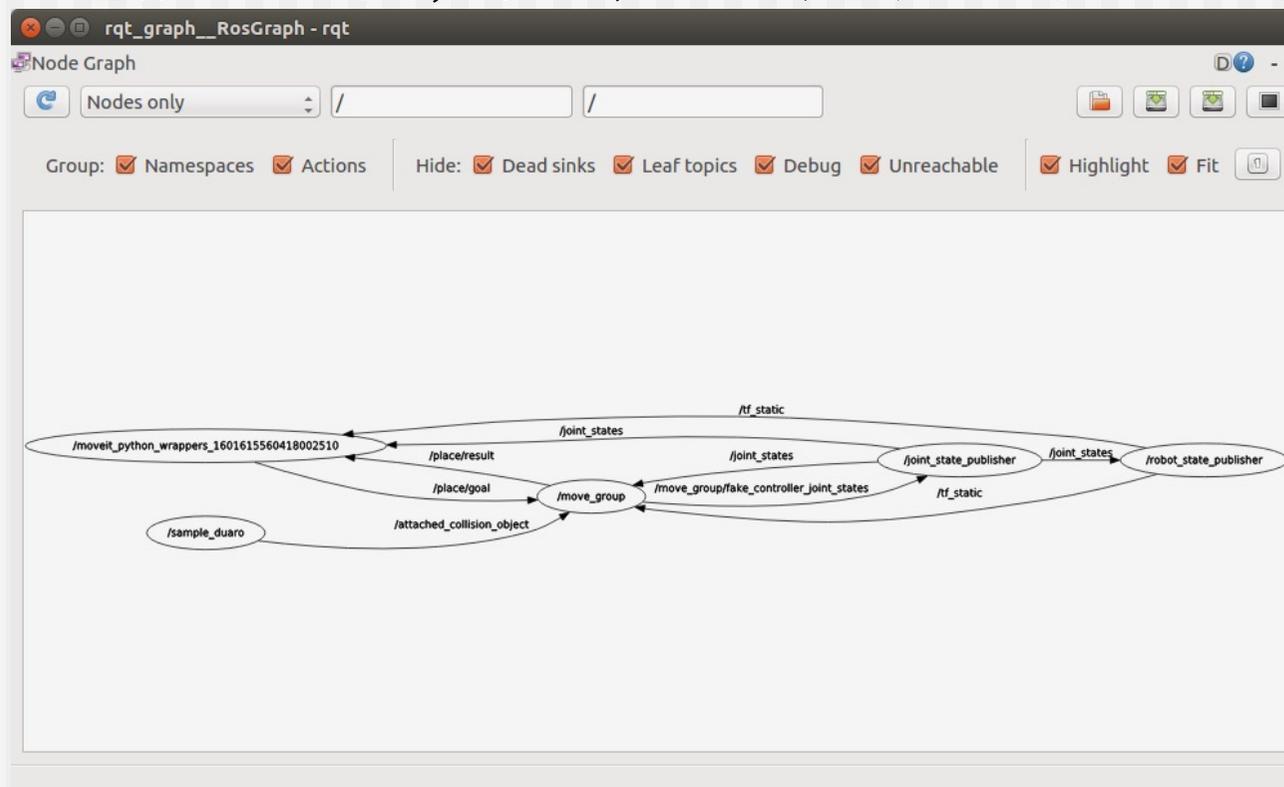
- ctrl+shift+tでカレントディレクトリ (/ros) で新規タブ作成し，以下のコマンド入力



```
tork@ubuntu: ~/.ros
Terminal x tork@ubuntu: ~/.ros x
tork@ubuntu:~/.ros$ rqt graph
usage: rqt [-b BINDING] [--clear-config] [-f] [--force-discover] [-h] [-l]
          [-ht] [-p PERSPECTIVE] [--perspective-file PERSPECTIVE_
FILE]
          [--reload-import] [-s PLUGIN] [-t] [-v]
          [--args [PLUGIN_ARGS [PLUGIN_ARGS ...]]] [--list-perspe
ctives]
          [--list-plugins] [--command-pid PID]
          [--command-start-plugin PLUGIN]
          [--command-switch perspective PERSPECTIVE]
rqt: error: unrecognized arguments: graph
tork@ubuntu:~/.ros$ rqt_graph
```

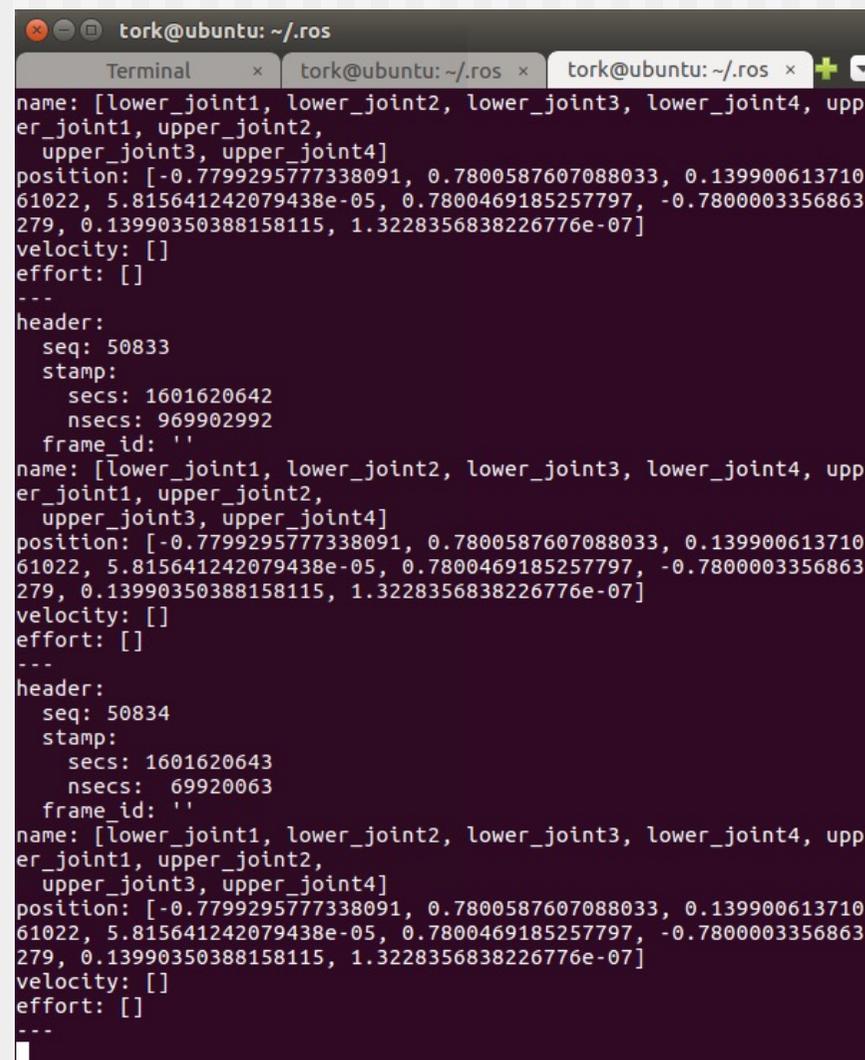
# 起動している関連ソフトウェア

- ソフトウェア群の関係を表すグラフが表示される
  - 丸で囲まれたものがノード，矢印がノード間の通信の方向，矢印に付随するものが通信の名前



# さらに深堀

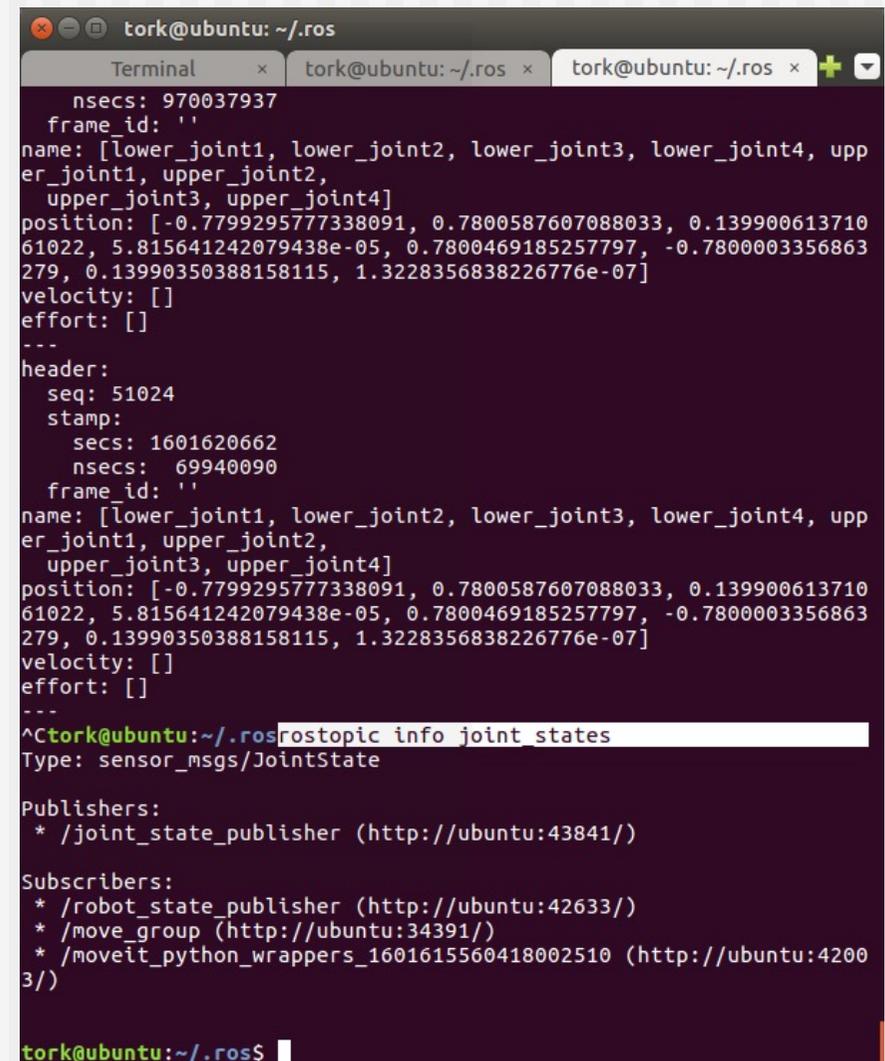
- トピック通信名「joint\_state」で扱っているデータについて調べてみる
- `$ rostopic echo joint_states`
- を入力すると通信しているデータをターミナルに出力することができる
- 内容は、データの名称通り、関節角度値



```
tork@ubuntu: ~/ros
Terminal x tork@ubuntu: ~/ros x tork@ubuntu: ~/ros x +
name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1, upper_joint2, upper_joint3, upper_joint4]
position: [-0.7799295777338091, 0.7800587607088033, 0.13990061371061022, 5.815641242079438e-05, 0.7800469185257797, -0.7800003356863279, 0.13990350388158115, 1.3228356838226776e-07]
velocity: []
effort: []
---
header:
  seq: 50833
  stamp:
    secs: 1601620642
    nsecs: 969902992
  frame_id: ''
name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1, upper_joint2, upper_joint3, upper_joint4]
position: [-0.7799295777338091, 0.7800587607088033, 0.13990061371061022, 5.815641242079438e-05, 0.7800469185257797, -0.7800003356863279, 0.13990350388158115, 1.3228356838226776e-07]
velocity: []
effort: []
---
header:
  seq: 50834
  stamp:
    secs: 1601620643
    nsecs: 69920063
  frame_id: ''
name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1, upper_joint2, upper_joint3, upper_joint4]
position: [-0.7799295777338091, 0.7800587607088033, 0.13990061371061022, 5.815641242079438e-05, 0.7800469185257797, -0.7800003356863279, 0.13990350388158115, 1.3228356838226776e-07]
velocity: []
effort: []
---
```

# トピック通信に関する情報の表示

- `$ rostopic info joint_states`
- を入力するとトピック通信に関する様々な情報を出力することができる(通信のデータの型, publisher(データの送信側), subscriber(データの受信側))
- 今回は, `sensor_msgs`という型の「JointState」という名前のデータを扱っていることがわかる



```
tork@ubuntu: ~/.ros
Terminal x tork@ubuntu: ~/.ros x tork@ubuntu: ~/.ros x +
nsecs: 970037937
frame_id: ''
name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1, upper_joint2, upper_joint3, upper_joint4]
position: [-0.7799295777338091, 0.7800587607088033, 0.13990061371061022, 5.815641242079438e-05, 0.7800469185257797, -0.7800003356863279, 0.13990350388158115, 1.3228356838226776e-07]
velocity: []
effort: []
---
header:
  seq: 51024
  stamp:
    secs: 1601620662
    nsecs: 69940090
  frame_id: ''
name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1, upper_joint2, upper_joint3, upper_joint4]
position: [-0.7799295777338091, 0.7800587607088033, 0.13990061371061022, 5.815641242079438e-05, 0.7800469185257797, -0.7800003356863279, 0.13990350388158115, 1.3228356838226776e-07]
velocity: []
effort: []
---
^Ctork@ubuntu: ~/.ros$ rostopic info joint_states
Type: sensor_msgs/JointState

Publishers:
* /joint_state_publisher (http://ubuntu:43841/)

Subscribers:
* /robot_state_publisher (http://ubuntu:42633/)
* /move_group (http://ubuntu:34391/)
* /moveit_python_wrappers_1601615560418002510 (http://ubuntu:42003/)

tork@ubuntu: ~/.ros$
```

# センサ情報(関節角度)のグラフ表示

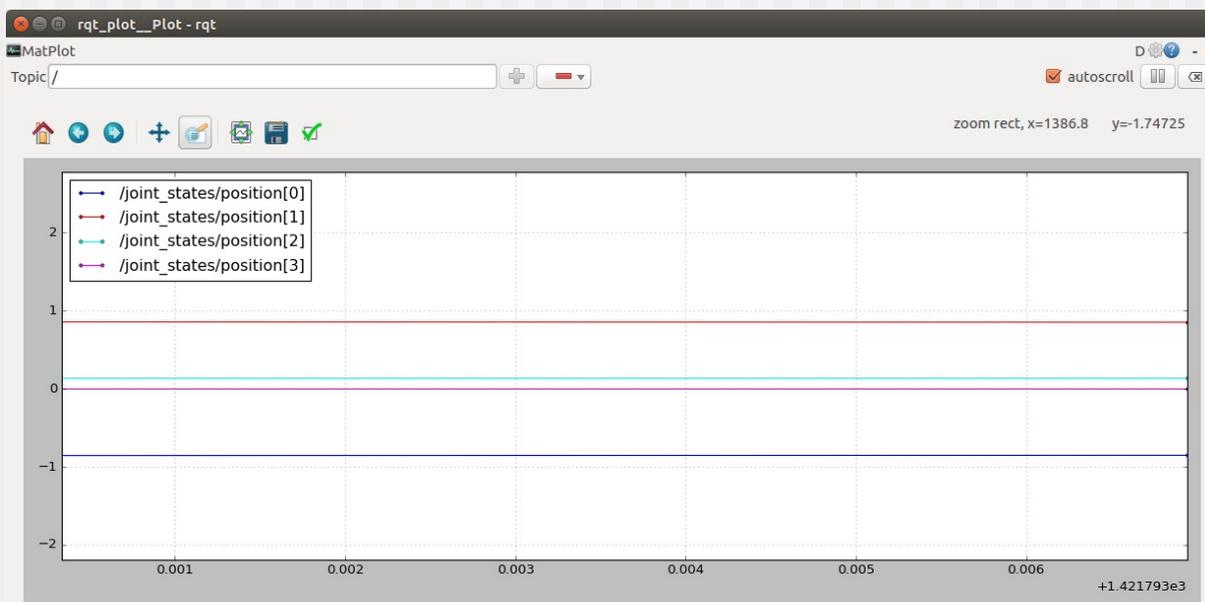
- トピック名とそのフィールドの名前がわかると、そこで配信している position(関節角度情報)のストリームをグラフにプロットすることができる
- 右図の白抜きで示したコマンドを入力してください
- `$rqt_plot`  
`joint_states/position[0]`  
`joint_states/position[1]`  
`joint_states/position[2]`  
`joint_states/position[3]`



```
tork@ubuntu: ~/ros
Terminal x tork@ubuntu: ~/ros x tork@ubuntu: ~/ros x +
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57851, stamp: , secs: 1601628456, nsecs: 644428014, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57852, stamp: , secs: 1601628456, nsecs: 744291067, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57852, stamp: , secs: 1601628456, nsecs: 744291067, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57853, stamp: , secs: 1601628456, nsecs: 844227075, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57853, stamp: , secs: 1601628456, nsecs: 844227075, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
PlotWidget.update_plot(): error in rosplot: [/joint_states/velocity[0]] index error for
: header: , seq: 57853, stamp: , secs: 1601628456, nsecs: 844227075, fram
e_id: ', name: [lower_joint1, lower_joint2, lower_joint3, lower_joint4, upper_joint1,
upper_joint2,, upper_joint3, upper_joint4], position: [-0.7799483878693078, 0.7799921
832771041, 0.1399073943829164, 7.824041894637048e-05, 0.780053481466556, -0.78005313106
46329, 0.13990184691431934, -7.580821523442864e-05], velocity: [], effort: []
tork@ubuntu:~/ros$ rqt_plot joint_states/position[0] joint_states/position[1] joint st
ates/position[2] joint_states/position[3]
tork@ubuntu:~/ros$
```

# rqt\_plotによる関節角度表示

- rqt\_plotを表示した状態で、さきほどのコマンドターミナル（Sample duAro s:start q:quitと出力されたターミナル）に戻り（タブをクリック）、sを入力してデモを実行すると、関節角度をライブプロットで見ることができる。このようなことを一切コードを読んだり、新たにコードを書かずに行うことができる

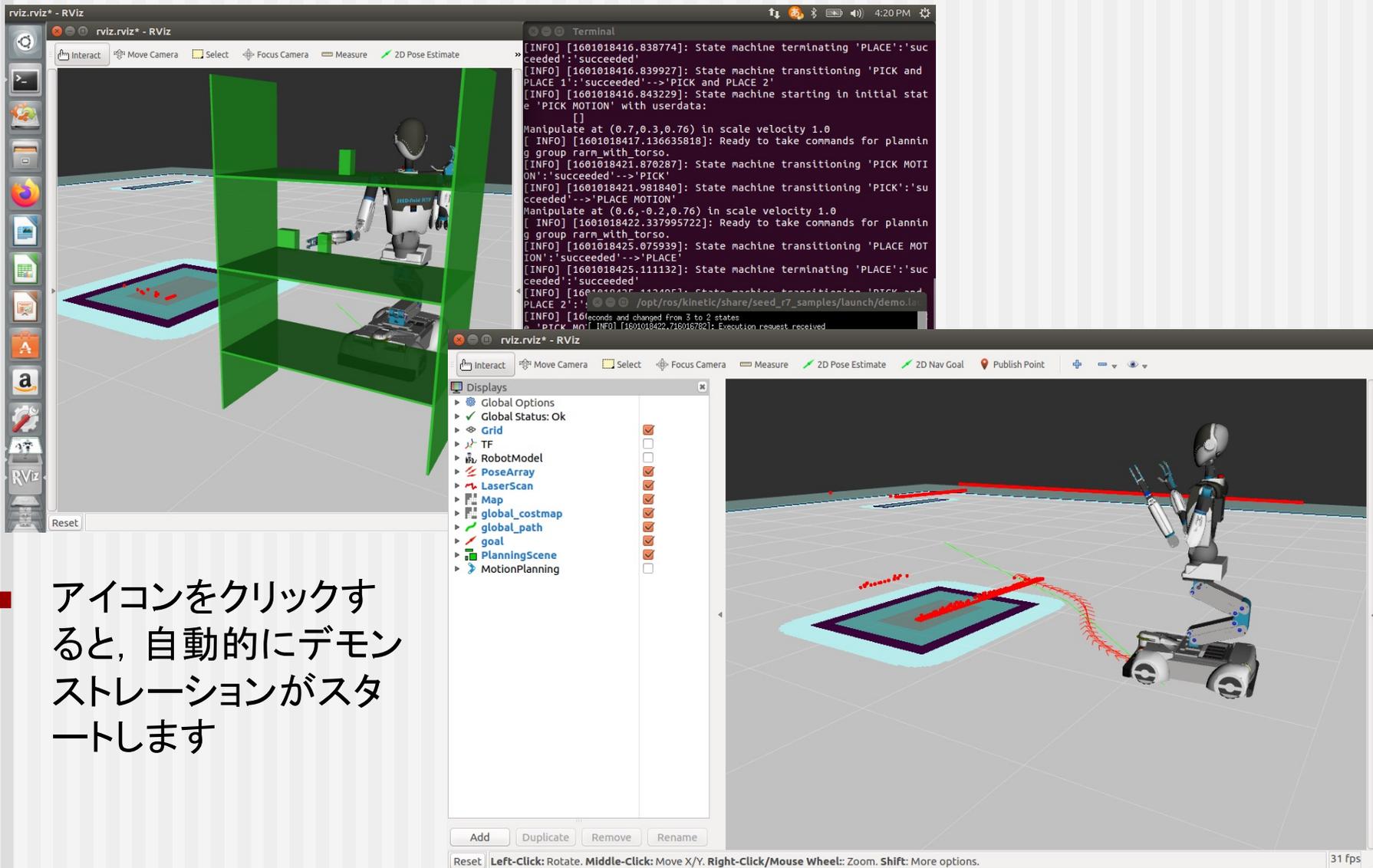


# Movelt!チュートリアル(日本語)

---

- さらに, コマンドを用いて, あるいはプログラムを用いてロボットの動作を定義することもできます
- 以下のURLのページにある「参考資料」  
[Tork Movelt! チュートリアル 0.0.10 for Kinetic](#)  
をご覧ください
  - <https://robo-marc.github.io/tutorials/si2020>
  - NEXTAGEを題材にしていますが, duAroの場合も記載されています

# SEED-Noïd (THK)



The screenshot displays the rviz.rviz\* - RViz interface. The main 3D view shows a robot in a room with a green shelf. The terminal window shows the following log messages:

```
[INFO] [1601018416.838774]: State machine terminating 'PLACE': 'succeeded': 'succeeded'  
[INFO] [1601018416.839927]: State machine transitioning 'PICK and PLACE 1': 'succeeded'-->'PICK and PLACE 2'  
[INFO] [1601018416.843229]: State machine starting in initial state 'PICK MOTION' with userdata:  
[]  
Manipulate at (0.7,0.3,0.76) in scale velocity 1.0  
[INFO] [1601018417.136635818]: Ready to take commands for planning group farm_with_torso.  
[INFO] [1601018421.870287]: State machine transitioning 'PICK MOTION': 'succeeded'-->'PICK'  
[INFO] [1601018421.981840]: State machine transitioning 'PICK': 'succeeded'-->'PLACE MOTION'  
Manipulate at (0.6,-0.2,0.76) in scale velocity 1.0  
[INFO] [1601018422.337995722]: Ready to take commands for planning group farm_with_torso.  
[INFO] [1601018425.075939]: State machine transitioning 'PLACE MOTION': 'succeeded'-->'PLACE'  
[INFO] [1601018425.111132]: State machine terminating 'PLACE': 'succeeded': 'succeeded'  
[INFO] [1601018425.112406]: State machine transitioning 'PICK and PLACE 2': 'succeeded'-->'PICK MOTION'  
[INFO] [1601018422.716016782]: Execution request received
```

The Displays panel shows the following options:

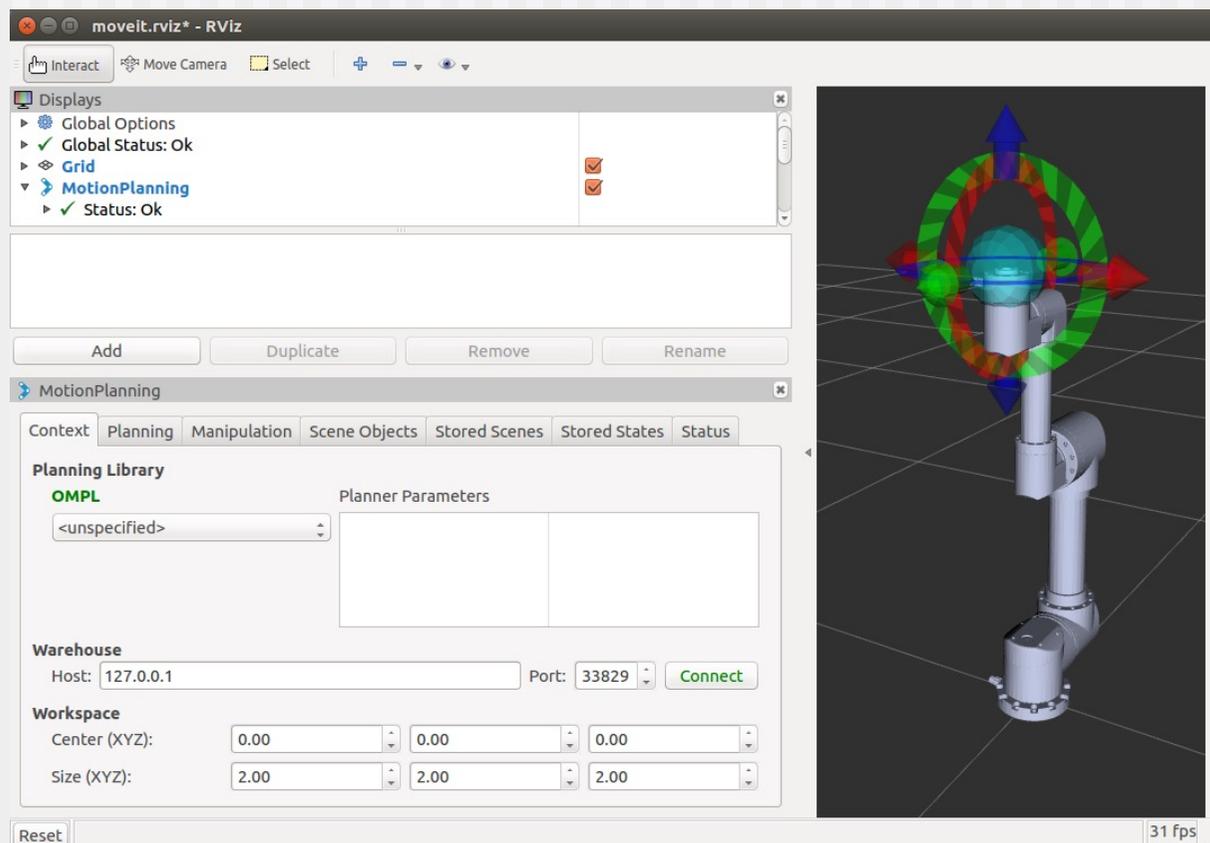
- Global Options
- Global Status: Ok
- Grid
- TF
- RobotModel
- PoseArray
- LaserScan
- Map
- global\_costmap
- global\_path
- goal
- PlanningScene
- MotionPlanning

The 3D view shows a robot in a room with a green shelf. The robot is positioned in the center of the room, facing the shelf. The shelf is a green structure with three shelves. The robot is holding a red object in its right hand. The floor is a light blue color. The walls are a light gray color. The ceiling is a light blue color. The robot is a white and blue humanoid robot with a head, torso, and arms. It has two legs and two feet. It is standing on a light blue floor. The robot is holding a red object in its right hand. The red object is a small, rectangular block. The robot is looking towards the shelf. The shelf is a green structure with three shelves. The robot is holding a red object in its right hand. The red object is a small, rectangular block. The robot is looking towards the shelf. The shelf is a green structure with three shelves. The robot is holding a red object in its right hand. The red object is a small, rectangular block. The robot is looking towards the shelf.

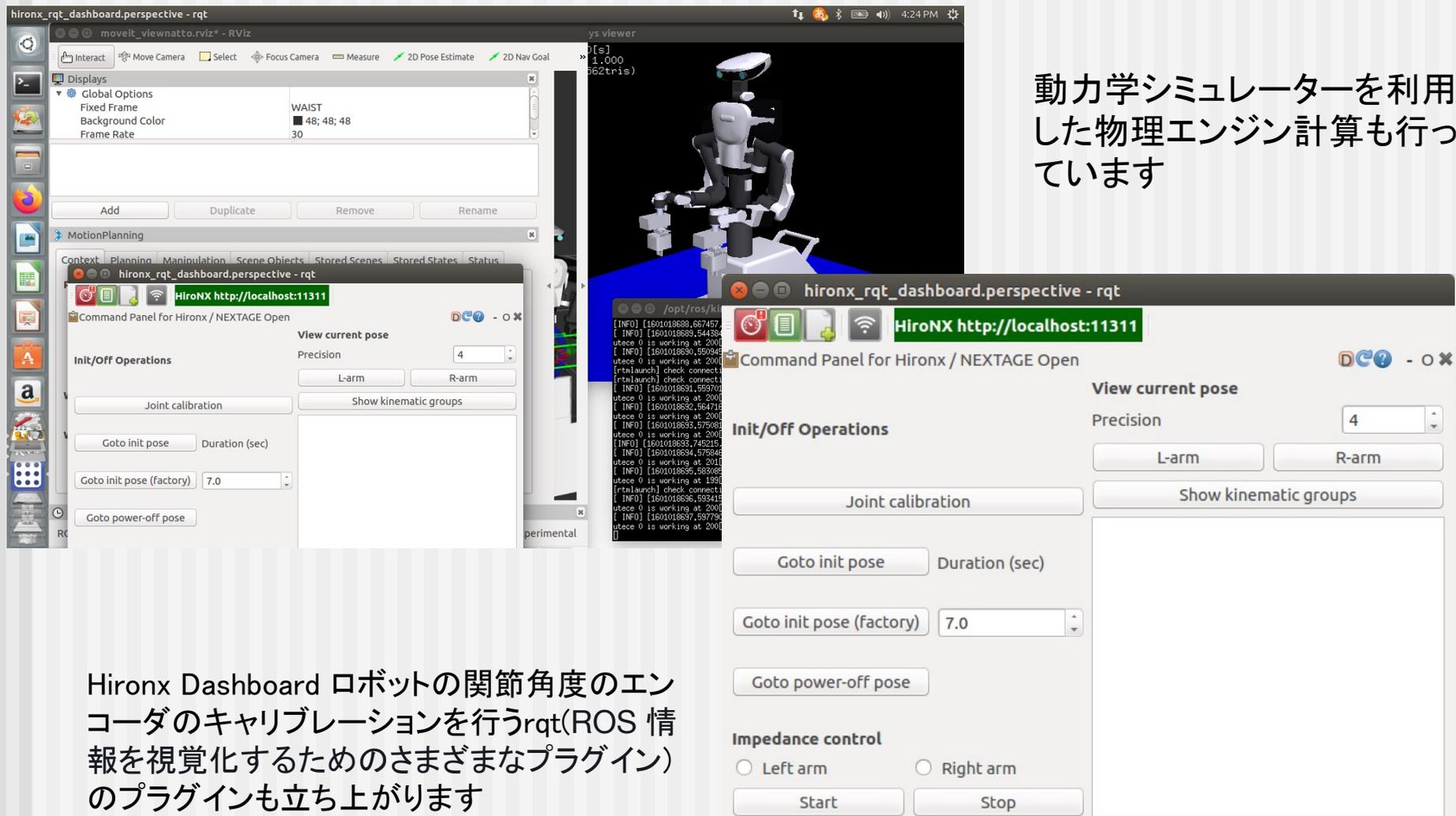
- アイコンをクリックすると、自動的にデモンストレーションがスタートします

# FR-Robot (富士ソフト)

- 一般的な6自由度ロボットです。duAroと同様にMoveItの基本的なI/Fで制御可能です。



# NEXTAGE(カワダロボティクス)



The image displays the Hironx Dashboard software interface, which is used for controlling the NEXTAGE robot. The interface is divided into several sections:

- Top Left:** A sidebar with various icons for system functions like 'Interact', 'Move Camera', 'Select', 'Focus Camera', 'Measure', '2D Pose Estimate', and '2D Nav Goal'.
- Top Center:** A 3D visualization window showing a white and black robot model in a simulated environment. The window title is 'hironx\_rqt\_dashboard.perspective - rqt'.
- Top Right:** A terminal window displaying system logs, including messages like '[INFO] [1601018638, 667457] [INFO] [1601018639, 544386] utacec 0 is working at 2001'.
- Bottom Left:** A 'Command Panel for Hironx / NEXTAGE Open' window. It features a 'View current pose' section with a 'Precision' dropdown set to '4', and buttons for 'L-arm' and 'R-arm'. Below this are 'Init/Off Operations' including 'Joint calibration', 'Show kinematic groups', 'Goto init pose' (with a 'Duration (sec)' field), 'Goto init pose (factory)' (set to '7.0'), and 'Goto power-off pose'.
- Bottom Right:** A similar 'Command Panel for Hironx / NEXTAGE Open' window, but with an 'Impedance control' section below the 'Init/Off Operations'. This section has radio buttons for 'Left arm' and 'Right arm', and 'Start' and 'Stop' buttons.

動力学シミュレーターを利用した物理エンジン計算も行っています

Hironx Dashboard ロボットの関節角度のエンコーダのキャリブレーションを行うrqt(ROS 情報を視覚化するためのさまざまなプラグイン)のプラグインも立ち上がります

# 最後に: より進んだ学習のために

---

- NEDO特別講座のHPへのリンク
  - <https://robo-marc.github.io/>
- 本講座では, より進んだ教育カリキュラムを提供予定(2021年1月現在)
  - ROS入門コース: ROS活用の基本的知識
  - OSS活用のガイドライン: オープンソフトウェア活用ためのライセンスについて
  - 市場化プロジェクト成果活用コース:
    - 移動ロボットのためのSLAM活用
    - モバイルマニピュレーション
    - 3Dビジョンセンサー活用
  - 産業用ロボット応用コース: OSSを用いた産業用ロボット制御
  - 画像処理・AI技術活用コース: 高度アルゴリズムの活用事例の紹介