目次

[1. はじめに 1](#_Toc126151615)

[1.1. ROS Moveit!・Gazebo シミュレータ 1](#_Toc126151616)

[1.2. 本講座の構成 2](#_Toc126151617)

[1.3. 環境構築 3](#_Toc126151618)

[1.4. Docker Engine 5](#_Toc126151619)

[2. KHIDuaro/RS007N ロボットタイプ起動紹介 6](#_Toc126151620)

[2.1. KHIDuaro 6](#_Toc126151621)

[2.2. RS007N 7](#_Toc126151622)

[3. Moveit! Commander 8](#_Toc126151623)

[4. RS007N実機ロボット接続方法 9](#_Toc126151624)

[4.1. センサー・ハンドの紹介 9](#_Toc126151625)

[4.2. ドライバーの状態取得方法 9](#_Toc126151626)

[4.3. 教師データの取得方法 9](#_Toc126151627)

[5. 一連の作業動作のプログラミング 10](#_Toc126151628)

[付録A 1](#_Toc126151629)

[付録B 2](#_Toc126151630)

[Reference 1](#_Toc126151631)

# はじめに

　本講座は川崎重工株式会社が公開している**khi\_robot**のパッケージを用いて，GUIからロボットの操作・シミュレーション・プログラムによる操作の紹介をしていきます．実機のロボットを扱っていく際にリアルタイムカーネル(**RT**)の構築が必要になって行く為，その構築をする必要があります．本講座の内容をスムーズに行って頂きたい為，扱っていく環境をDockerEngineにまとめてあります．扱っていくROSバージョンはNoeticとなっており，本講座で作成したプログラムはPython3のバージョンで実行できるようになっています．最後に本講座はROSを扱った実機ロボットの操作を目的にしている為，ROSによる基本動作やプログラムの操作の紹介後，RS007Nタイプの実機ロボットの操作方法を紹介していきます．

　本講座で扱った環境の紹介を下記の一覧となっています．

|  |
| --- |
| * **計算機 仕様** * **OS : Ubuntu 20.04.5 LTS**   **(Ubuntu 20.04.5 ISOファイルダウンロード先)**  [**https://releases.ubuntu.com/20.04/ubuntu-20.04.5-desktop-amd64.iso**](https://releases.ubuntu.com/20.04/ubuntu-20.04.5-desktop-amd64.iso)   * **CPU :** * **GPU :** * **Kernel :** * **RealtimeKernel (RT) :** * **ROS** * **Version : Noetic** * **Catkin : 最新バージョン** * **Python: Version 3** * **Ros Package** * **khi\_robot (Github):** [**https://github.com/Kawasaki-Robotics/khi\_robot.git**](https://github.com/Kawasaki-Robotics/khi_robot.git) * **DockerEngine** * **Version:** * **Compose Version:** |

本講座で扱う環境構築スクリプト，実機ロボット操作パッケージやソースコードは，

(**Github**): <https://github.com/skrjtech/dockerRosKHIRobot.git>

に公開されています．こちらを参照しながら，コマンドのコピー＆ペーストを行うことで快適な操作が可能です．

## ROS Moveit!・Gazebo シミュレータ

　本講座で扱うROSシミュレータの種類の紹介をしていきます．

* + - * Ros シミュレータ

-　Moveit! : ロボット操作用モーションプランニング・ナビゲーション

-　Gazebo : ロボットの姿勢や動作の可視化・仮想空間で疑似障害物の構築

　Moveit!では，GUIの操作を行うことで，実機ロボットによるモーション動作や移動などのプランニングを行うことができ，また，Gazeboと連携させることで仮想空間でロボットのモーションの確認を行うことができます．

## 本講座の構成

本講座では実機ロボットの操作を紹介していく為に，環境構築からシミュレータの扱い方，Moveit! Commanderを使用したプログラミングを行います．

　本講座の構成は下記の様になっています．

* + **はじめに**
  + **KHIDuaro/RS007N ロボットタイプの起動紹介**
  + **Moveit! Commander**
  + **RS007N実機ロボット接続方法**
  + **教師データ取得方法**
  + **一連の作業動作のプログラミング**
  + **付録A新規パッケージ作成**
  + **付録B独自プログラミング解説**
  + **Reference**

## 環境構築

　環境構築には本講座で用意したリポジトリをGithubからダウンロード，もしくはGitでCloneを行うことにより，本講座同様の環境を実現することができます．新規ターミナルを起動して下記のリンクから，

(Github): <https://github.com/skrjtech/dockerRosKHIRobot.git>

必要なリポジトリをCloneし，dockerRosKHIRobotのディレクトリに移動します．

|  |
| --- |
| $ git clone <https://github.com/skrjtech/dockerRosKHIRobot.git>  $ cd dockerRosKHIRobot |

**dockerRosKHIRobot**の階層について下記の一覧の様になっています．

* dockerRosKHIRobot
  + .devcontainer (VSCodeで起動するコンテナ情報格納ディレクトリ **本講座では使用しない**)
  + .docker
    - noetic　(本講座で使用するDockerのコンテナ構築情報)
      * docker-compose-realtime.yml (リアルタイム制御用コンテナ起動情報ファイル)
      * docker-compose.yml (シミュレーション用コンテナ起動情報ファイル)
      * Dockerfile (Dockerコンテナ環境構築用情報ファイル)
  + archives (本講座で作成した独自パッケージとソースコード)
    - khiRobotPack (独自パッケージ格納ディレクトリ)
      * rs007n\_type (RS007N用独自パッケージ 教師データを含めたソースコード)
    - khirobotenv.sh (khi\_robotパッケージの環境構築用スクリプト)
  + config (リアルタイムカーネル情報ファイル格納ディレクトリ)
  + docs (本講座内容公開ディレクトリ)
  + install
    - docker\_install\_khirobot\_setup.sh (DockerEngineのインストールとNoeticのインストール済み環境イメージのダウンロードのスクリプト)
    - realtime\_kernel\_install.sh (計算機のリアルタイムカーネルの構築スクリプト)
  + README.md (本講座概要)
  + build.sh (メイン：環境構築用スクリプト)

以上でdockerRosKHIRobotの内容となります，メインとなるbuild.shのスクリプトを起動することで，一連のインストールが開始され環境構築を行うことができます．build.shの実行の前に**「Software & Update」**の画面を開き,

|  |
| --- |
|  |

**Source Code**のチェック欄を有効にし，Closeをクリックします．次の画面が出現しますので，**Reload**をクリックし，画面を閉じます．

|  |
| --- |
|  |

build,shを実行して環境構築を開始します．

|  |
| --- |
| $ . build.sh |

途中でパスワード入力が求められるので，御使用のアカウントのパスワードを入力し，先に進みます．リアルタイムカーネル(RT)のインストール時にも下記の様にパスワードが求められるので，こちらもパスワードを入力して行きます．

|  |
| --- |
|  |

すべてのインストールが終了しますと，自動的に再起動が実行されます．

再起動されましたらリアルタイムカーネル(RT)で起動するように画面一覧から，下記のように選択します．

* Ubuntu 起動メニュー
* Ubuntu
* Advanced options for Ubuntu <-- こちらを選択します．
  + Ubuntu, with Linux 5.15.0-57-generic
  + Ubuntu, with Linux 5.15.0-57-generic (recovery mode)
  + Ubuntu, with Linux 5.15.0-56-generic
  + Ubuntu, with Linux 5.15.0-56-generic (recovery mode)
  + Ubuntu, with Linux 5.xxx-xxx-rtxxx <-- こちらを選択しRTで起動
  + Ubuntu, with Linux 5.xxx-xxx-rtxxx (recovery mode)
* Memory Test (memtest86+)
* Memory Test (memtest86+, serial consol 115200)
* ...

## Docker Engine

　本講座では，DockerEngineをROS環境の土台として使用していきます．Docker Engineを使用するのは，独自パッケージの作成の操作不良や，何かしらの不具合で動作が出来なくなってしまった時にDocker Enginのコンテナの作り直しをすることで，環境の再構築することなく元の環境を呼び出せるメリットとして使用していきます．前節で解説した環境構築でDockerEngineがインストールされると同時に本講座で扱う**コンテナイメージファイル**がダウンロードされます．dockerRosKHIRobotディレクトリの下で新規ターミナルを起動し，次のコマンド入力でイメージファイルが存在しているかを確認します．

|  |
| --- |
| $ docker images |

上記のコマンドで下記のように表示されます．

|  |
| --- |
|  |

**skrjtech/khi\_robot:noetic**のイメージファイルの確認が取れましたら，コンテナの起動をしていきます．

|  |
| --- |
| $ docker compose -f .docker/noetic/docker-compose.yml up -d |

コマンド実行後下記のように表示されます．

|  |
| --- |
|  |

次のコマンドで，コンテナ一覧を表示し実際にコンテナが起動されてかを確認します．

|  |
| --- |
| $ docker ps -a |

次のように表示されましたら，コンテナが起動されている状態となります．

|  |
| --- |
|  |

次にコンテナ内に入るには，

|  |
| --- |
| $ docker exec -it ros\_noetic bash |

のコマンドで中に入ることができ下記のようになります．

|  |
| --- |
|  |

コンテナから抜けるには，exitと入力し，元のターミナルに戻ります．

|  |
| --- |
|  |

最後にコンテナの削除や環境のリセットを行う際には，コンテナの削除動作をします．下記のコマンドで，

|  |
| --- |
| $ docker rm -f ros\_noetic |

コンテナの削除を行うことができ，もう一度コンテナ起動を行うと初期状態のコンテナを起動することができます．

# KHIDuaro/RS007N ロボットタイプ起動紹介

　Githubで公開されているkhi\_robotのパッケージ内には，複数のロボットタイプが存在します，今回はその中でKHIDuaroと実機制御として使用するRS007Nのロボットタイプを紹介していきます．また，紹介の際に，GUIによる操作でGazeboとRvizのツールを使用して，Gazeboで起動したロボットタイプの動作制御をRvizで行っていく紹介をしていきます．

## KHIDuaro

KHIDuaroのシミュレーションを行って行く前に，Dockerコンテナとホストとなる御使用の計算機間でGUIの通信を行えるように設定をしていきます．dockerRosKHIRobotのディレクトリの下で新規ターミナルを開き，

|  |
| --- |
| $ xhost +local: |

のコマンドを入力します．次に，コンテナを起動していきます．

|  |
| --- |
| $ docker compose -f .docker/noetic/docker-compose.yml up -d |

コンテナの起動を行ったらコンテナ内に入ります．

|  |
| --- |
| ＄docker exec -it ros\_noetic bash |

DockerよるGUIの起動を確認します．次のコマンドで，

|  |
| --- |
| $ xeyes |

xeyesのアプリケーションが起動され下記の画面が出現します．

|  |
| --- |
|  |

画面を閉じるときは，画面から閉じるか，もしくは，ターミナルから強制停止の(Ctrl + C)でアプリケーションを閉じることができます．

　ここまで正常に動作が行えたら，KHIDuaroを立ち上げていきます．複数のターミナルを開いていく為，tmuxのツールを使用していきます．次のコマンドで，

|  |
| --- |
| $ tmux |

次の画面に切り替わります．

|  |
| --- |
|  |

切り替わった画面でKHIRobotのノードを起動します．

|  |
| --- |
| $ roslaunch khi\_robot\_bringup duaro\_bringup.launch |

下記のようにノードが起動されていきます．

|  |
| --- |
|  |

別のターミナルを開いて，Gazeboの起動をします．ターミナル上で「Ctrl + b」を押した後に「C」を押すと新規タブのターミナルが開いていきます．

|  |
| --- |
|  |

次のコマンドで，

|  |
| --- |
| $ roslaunch khi\_duaro\_gazebo duaro\_world.launch |

ロボット情報が含めたGazebo環境が開きます．

|  |
| --- |
|  |

次にRvizを起動し，GUIのよる動作制御をしていきます．次のコマンドで，

|  |
| --- |
| $ roslaunch khi\_duaro\_moveit\_config moveit\_planning\_execution.launch |

Rvizが起動されます．

|  |
| --- |
|  |

ロボットの右アーム先端の水色球体をマウス操作で移動させ，**赤枠内**の「Plan&Execute」をクリックすると動かした方向にアームが移動されます．

|  |
| --- |
|  |

最後に起動されたGUIとノードを閉じます．新規タブのターミナルを起動し，

|  |
| --- |
| $ tmux kill-server |

のコマンドを入力することにより，tmux上で立ち上げたすべてのターミナルが強制的に閉じていきます．

以上でKHIDuaroの紹介を終わります．

## RS007N

KHIDuaroの起動方法と同様にRS007Nの起動をしていきます．RS007Nでは今後実機制御の解説をしていく為，単純な起動方法を紹介していきます．はじめに，tmuxを開いて，RS007Nのノードを起動します．

|  |
| --- |
| $ tmux |

新規タブのターミナルでノードの起動，

|  |
| --- |
| $ roslaunch khi\_roboto\_bringup rs007n\_bringup.launch |

起動後に新規タブを立ち上げGazeboとRvizを起動していきます．

|  |
| --- |
| $ roslaunch khi\_rs\_gazebo rs007n\_world.launch |

新規タブでRvizを起動．

|  |
| --- |
| $ roslaunch khi\_rs007n\_moveit\_config moveit\_planning\_execution.launch |

# Moveit! Commander

紹介したロボットタイプの制御にはGUIによる制御の他にプログラムによる制御ができます．Moveit！のAPIを使用することでロボットの各関節の制御や位置制御，姿勢制御が行うことができ，また，プランニングによる動作計画を決めることで一連の作業を行うことができます．

次のリンク先で，

<https://robo-marc.github.io/moveit_tutorial/>

NEDO特別講座としてMoveit!のチュートリアルが詳細に解説されていますので，詳しく知りたい方はこちらをご参照ください．本講座では実機として動かすRS007Nの単純なXYZ座標の移動や目的の位置に向かう制御方法のプログラムを紹介していきます．

RS007Nの関節制御をしていく為に，マスター情報の起動をしていきます．tmuxを起動し，

|  |
| --- |
| $ tmux |

roslaunchでRS007Nマスターを起動します．

|  |
| --- |
| $ roslaunch khi\_robot\_bringup rs007n\_bringup.launch |

新規タブを開き，Gazeboを起動します．

|  |
| --- |
| $ roslaunch khi\_rs\_gazebo rs007n\_world.launch |

最後に各関節の情報を取得する為にRvizを起動します．

|  |
| --- |
| $ roslaunch khi\_rs007n\_moveit\_config moveit\_planning\_execution.launch |

対話的なプログラムを行っていく為，Pythonでよく使われるJupyter Labを使っていきます．新規タブを開き次のコマンドを入力していきます．

|  |
| --- |
| $ roscd  $ jupyterlab --allow-root --NotebookApp.token=’’ |

JupyterLabの起動を行うと自動的にブラウザが起動します．もし自動的に起動しない場合は，ブラウザを起動させ，検索欄で「http://localhost:8888/lab」と検索します．

|  |
| --- |
|  |

# RS007N実機ロボット接続方法

　本講座の目的である実機との接続方法と各関節制御の解説をしていきます．下記のイメージは，

|  |
| --- |
|  |

本講座で使用したRS007Ｎの環境で， Gazeboのシミュレーション上にはなかった**ハンド**が取り付けられています．またその周辺にはセンサー類が設置されており，センサーの状態を取得する関数がkhi\_robotのパッケージに用意され，関数にコマンドと取得したしシグナル値を指定することでシグナルに接続されているセンサの状態を取得することが出来ます．下記のイメージは周辺装置となっています．

|  |
| --- |
|  |

ロボット周辺には，在荷センサーや色識別センサー，供給ユニット，ピックする物体があります．これらの操作方法を順に解説していきます．

## RS007N周辺装置の紹介

## ドライバーの状態取得方法

## 教師データの取得方法

# 一連の作業動作のプログラミング

# 付録A

# 付録B

# Reference