ROS

Doosan Robot

M0609 | M0617 | M1013 | M1509

ROS Programming Manual

DOOSAN

1. Doosan ROS package 설치

1.1 Overview

- **Doosan robotics ROS package**

Doosan robotics ROS 패키지는 ROS 상에서 두산 협동로봇을 구동하기 위한 메타 패키지로 URDF 모델을 제공하고 Rviz, Gazebo 를 통하여 시뮬레이션이 가능하며, moveIt 이나 다양한 예제를 통하여 실제 로봇을 구동 시킬 수 있습니다.

1.2 제약 사항

- **System**

사용자 PC는 X86 시스템이어야 합니다.

원할한 시뮬레이션을 위해서는 워크스테이션급 PC를 권장합니다.

- **OS 및 ROS version**

Ubuntu 16.04(32/64bit) + ROS kinetic

Ubuntu 18.04(64bit) + ROS melodic

1.3 설치

- **소스 설치**

Doosan robotics Github 에서 소스를 다운 받아서 빌드 합니다

Github : **https://github.com/doosan-robotics/doosan-robot**

- **설치 방법**

```
$ mkdir -p /home/path/to/your/workspace/src
$ cd /home/path/to/your/workspace/src
$ catkin_init_workspace
$ git clone https://github.com/doosan-robotics/doosan-robot
$ rosdep install --from-paths doosan-robot --ignore-src --rosdistro kinetic -r -y
$ catkin_make
$ source ./devel/setup.bash
```

2. 로봇 제어기와 연동

2.1 Emulator mode

▪ 기능

- 실제 로봇제어기가 없는 경우, 두산 로봇 애뮬레이터(DRCF)를 통해서 로봇을 시뮬레이션 가능합니다.
- 로봇 한대당 하나의 애뮬레이터가 필요합니다.
- 다중로봇 제어 시, 로봇 개수만큼 애뮬레이터를 실행 시켜야하며 각각 다른 포트를 사용해야 합니다.
- 애뮬레이터 실행에는 **반드시 root 권한이 필요**합니다.

▪ 애뮬레이터(DRCF) 실행

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
  or
$ sudo ./DRCF32 <port>     ## 32bits OS
```

▪ 다중 로봇(2 대) 사용 시

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 12345
새로운 콘솔 창에서
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 12346
```

그림 2.1 DRCF 실행

2.2 실제 로봇 연결

▪ 기능

- 실제 로봇 제어기와 연결을 합니다.
- 로봇 제어기의 디폴트 IP는 192.168.127.100, port는 12345 입니다.

▪ 제어기 연결

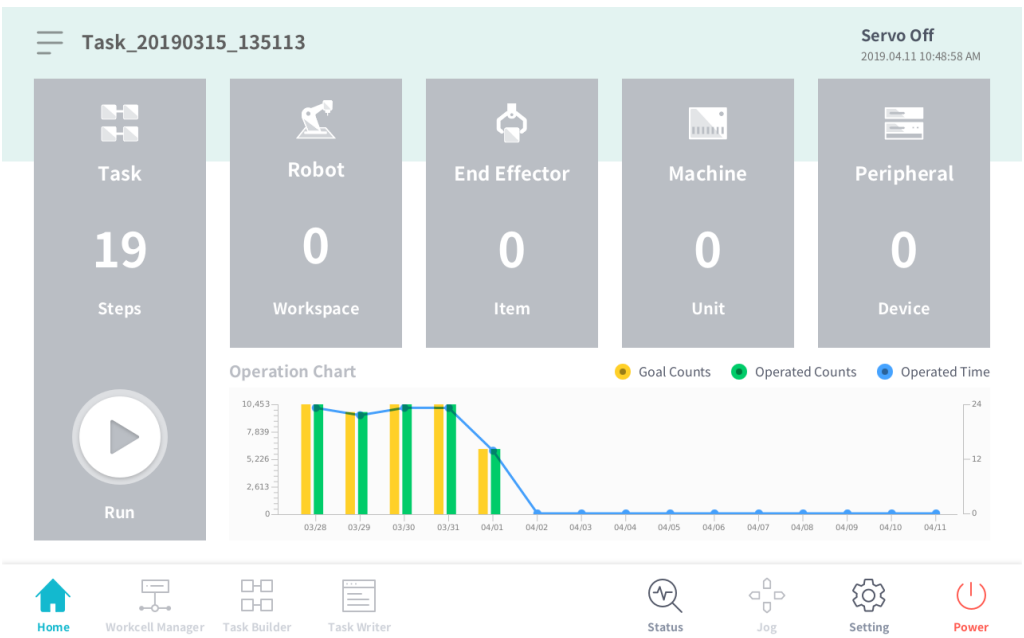


그림 2.2 TP 화면

- 사용자는 TP 화면의 Setting -> Network에서 고정 IP를 설정할 수 있습니다.

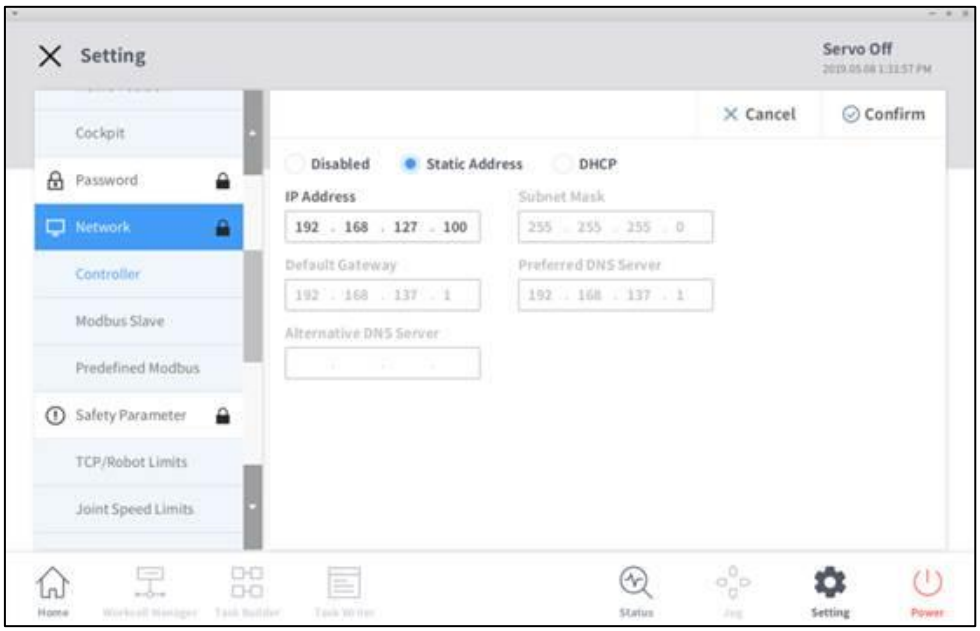


그림 2.3 TP 상에서 제어기 IP 확인

- Network 탭에서 설정된 제어기의 IP를 확인하고, 이 IP를 ROS 상에서 host := ROBOT_IP 로 사용합니다.
- ROS Control Node가 올바르게 실행되었다면, TP의 제어권이 ROS로 넘어갑니다.
- 제어권이 넘어간 TP 화면에는 아래와 같은 팝업이 출력됩니다.

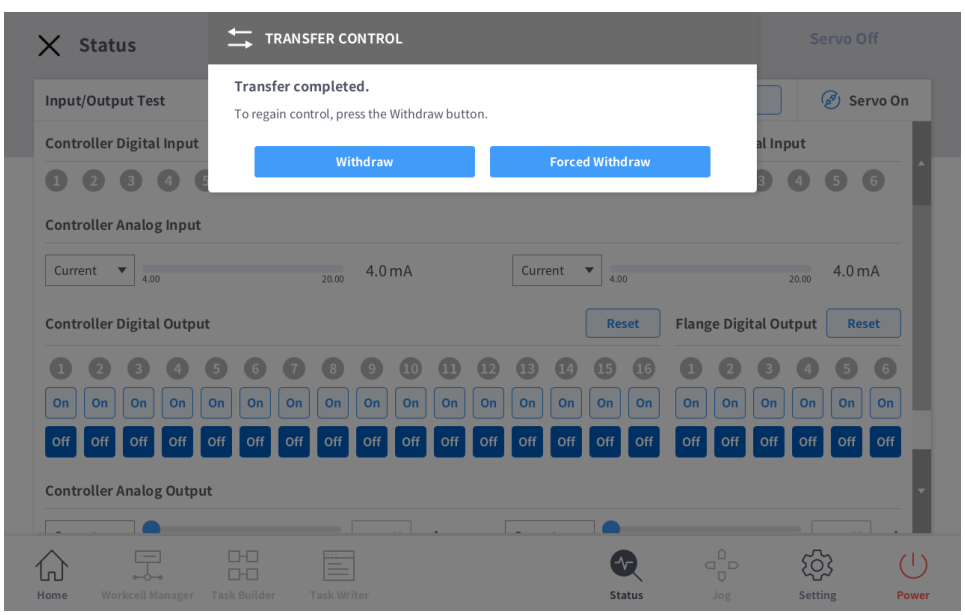


그림 2.4 제어권 이전 시 발생 팝업

3. dsr_description

3.1 dsr_description <robot_model>.launch

▪ 기능

- Rviz 상에 로봇 모델을 띄우고, Joint_state_publisher 를 로딩합니다.
- 로딩된 Joint_state_publisher를 통하여 로봇을 움직입니다.

▪ 파라미터

인수명	자료형	기본값	설명
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509
color	-	white	로봇 컬러 . white or blue
gripper	-	none	gripper 사용 유무 . none : gripper 미 사용 . robotiq_2f : robotiq two finger 장착

▪ 예제

```
$ roslaunch dsr_description m0609.launch
$ roslaunch dsr_description m1013.launch color:=blue # Change Color
$ roslaunch dsr_description m1509.launch gripper:=robotiq_2f # insert robotiq
gripper
$ roslaunch dsr_description m0617.launch color:=blue gripper:=robotiq_2f
```

하기 3.1 그림과 같이 Rviz 상에 로봇과 Joint_state_publisher 가 로딩 됨.
Joint_state_publisher 를 통하여 로봇을 구동 시킴.

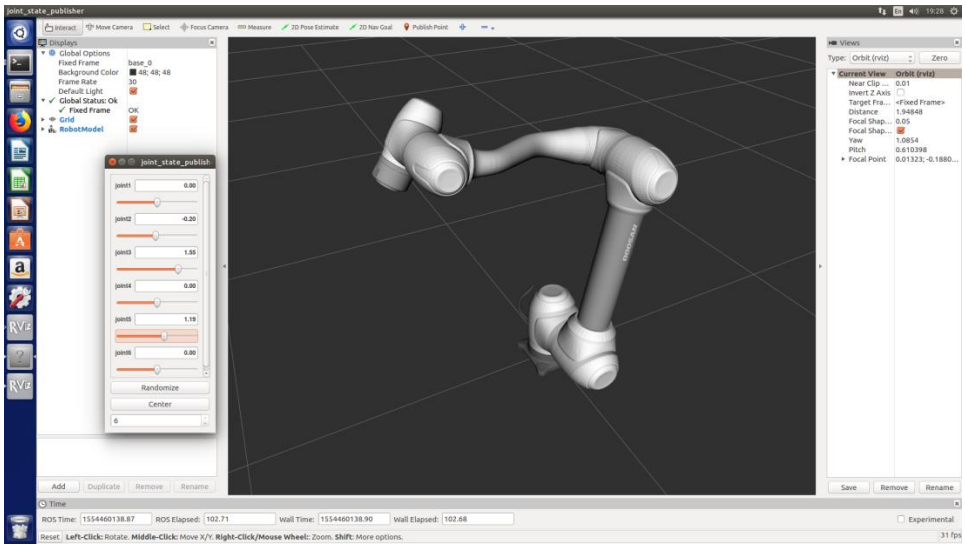


그림 3.1 Rviz 상에 로봇

4. dsr_moveit_config

4.1 dsr_moveit_config

▪ 기능

- Rviz 상에 로봇 모델을 띄우고 moveit을 통하여 로봇을 제어합니다.
- 시뮬레이션 모드로만 동작합니다.

▪ 파라미터

인수명	자료형	기본값	설명
color	-	white	로봇 컬러 . white or blue

▪ 예제

```
$ roslaunch moveit_config_m0609 m0609.launch
$ roslaunch moveit_config_m0617 m0617.launch
$ roslaunch moveit_config_m1013 m1013.launch color:=blue
$ roslaunch moveit_config_m1509 m1509.launch
```

하기 4.1 그림과 같이 Rviz 상에 로봇과 MotionPlanning 창이 로딩 됨.
MotionPlanning 를 통하여 로봇을 가상으로 구동 시킴.

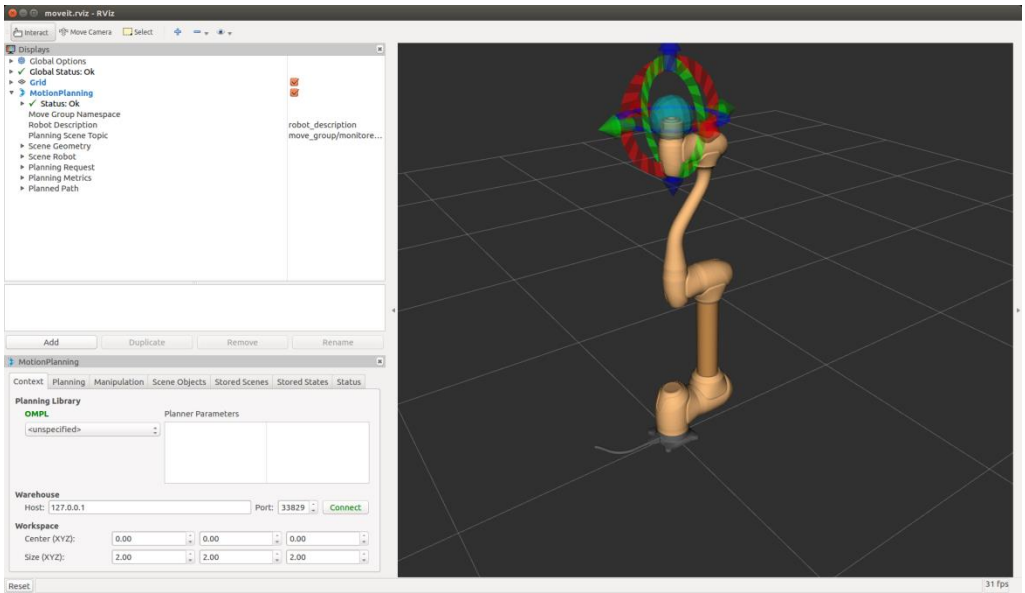


그림 4.1 Rviz + MoveIt

4.2 dsr_control

▪ 기능

- Rviz 상에 로봇 모델을 띄우고 moveit을 통하여 로봇을 제어합니다.
- 애뮬레이터 모드나 실제 로봇과 연결하여 동작합니다.
- 애뮬레이터 모드는 virtual mode만 작동합니다.

▪ 파라미터

인수명	자료형	기본값	설명
host	-	127.0.0.1	로봇 제어기 IP . 애뮬레이터 : 127.0.0.1 . 실제 로봇제어기 : 192.168.127.100
Port	-	12345	서비스 port
mode	-	virtual	로봇 동작 모드 - virtual : 가상 동작 - real : 실제 동작
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509
color	-	white	로봇 컬러 . white or blue
gripper	-	none	gripper 사용 유무 . none : gripper 미 사용 . robotiq_2f : robotiq two finger 장착

▪ 예제

<애뮬레이터 실행>

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
```

or

```
$ sudo ./DRCF32 <port>     ## 32bits OS
$ roslaunch dsr_control dsr_moveit.launch model:=m0609
$ roslaunch dsr_control dsr_moveit.launch model:=m0617 color:=blue
$ roslaunch dsr_control dsr_moveit.launch model:=m1013
$ roslaunch dsr_control dsr_moveit.launch model:=m1509 gripper:=robotiq_2f
```

<실제 로봇과 연동 모드>
로봇제어기 IP defalut = 192.168.127.100, port = 12345

```
$ roslaunch dsr_control dsr_moveit.launch model:=m0609 host:=192.168.127.100
mode:=virtual
$ roslaunch dsr_control dsr_moveit.launch model:=m0617 host:=192.168.127.100
mode:=real
$ roslaunch dsr_control dsr_moveit.launch model:=m1013 host:=192.168.127.100
mode:=virtual color:=blue
$ roslaunch dsr_control dsr_moveit.launch model:=m1509 host:=192.168.127.100
mode:=virtual color:=blue gripper:=robotiq_2f
```

하기 4.2 그림과 같이 Rviz 상에 로봇과 MotionPlanning 창이 로딩 됨.
MotionPlanning 를 통하여 로봇을 구동 시킴.

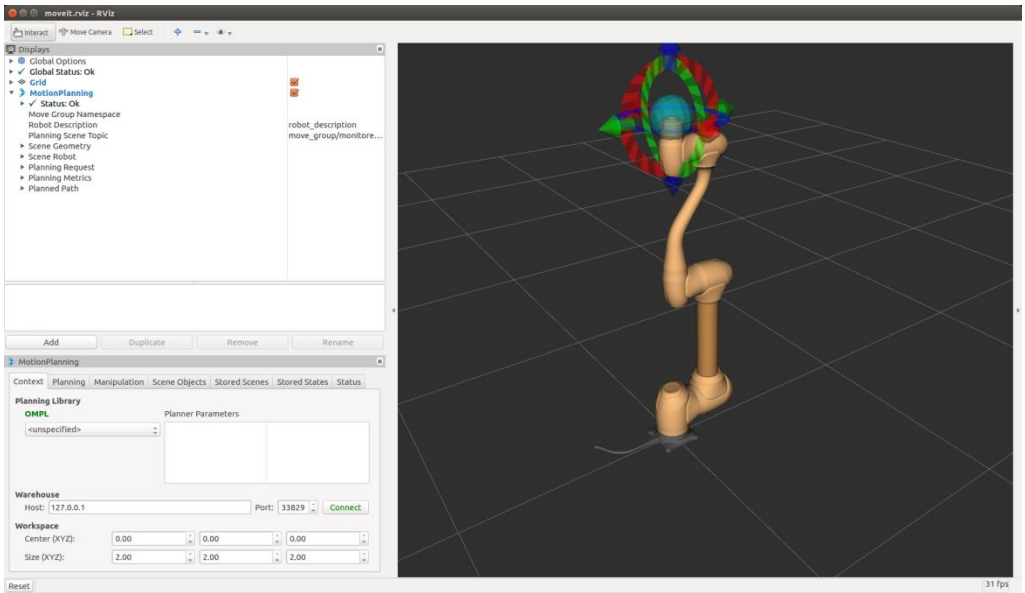


그림 4.2 Rviz + dsr_control

5. dsr_launcher

5.1 dsr_launcher

▪ 기능

- dsr_launcher 를 통하여 다양한 로봇 환경을 구성합니다.
- 파라미터에 따라 Single Robot, Multi Robot, Gripper, mobile 환경을 구축할 수 있습니다.
- dsr_launcher 로딩 후, 각 환경에 맞는 dsr_example 을 rosrun 으로 구동시킵니다.
  (상세 내역은 6장. dsr_example 을 참조하시기 바랍니다.)

▪ 파라미터

인수명	자료형	기본값	설명
host	-	127.0.0.1	로봇 제어기 IP . 애뮬레이터 : 127.0.0.1 . 실제 로봇제어기 : 192.168.127.100
port	-	12345	서비스 port
mode	-		로봇 동작 모드 - virtual : 가상 동작 - real : 실제 동작 . 애뮬레이터 : only . 실제 로봇제어기 : 192.168.127.100
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509
color	-	white	로봇 컬러 . white or blue
gripper	-	none	gripper 사용 유무 . none : gripper 미 사용 . robotiq_2f : robotiq two finger 장착
mobile	-	none	Mobile robot 사용 유무

인수명	자료형	기본값	설명
			. none : 미 사용 . husky : husky 모바일 로봇 사용

- **예제: 애뮬레이터 모드**

<애뮬레이터 실행>

$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF

$ sudo ./DRCF64 <port> ## 64bits OS , defult port = 12345

or

$ sudo ./DRCF32 <port> ## 32bits OS

<single robot>

$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013

$ roslaunch dsr_launcher single_robot_gazebo.launch color:=bule

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch

<single robot + gripper>

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch gripper:=robotiq_2f

<single robot + gripper + mobile>

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch gripper:=robotiq_2f mobile:=husky

<multi robot>

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch

< multi robot + gripper>

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch gripper:=robotiq_2f

< multi robot + gripper + mobile>

$ roslaunch dsr_launcher multi _robot_rviz_gazebo.launch gripper:=robotiq_2f mobile:=husky

- **예제: 실제 로봇 모드**

<실제 로봇 연결>

<single robot>

$ roslaunch dsr_launcher single_robot_rviz.launch host:=192.168.127.100 port:= 12345 mode:=real model:=m1013

$ roslaunch dsr_launcher single_robot_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real color:=bule

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real

<single robot + gripper>

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real gripper:=robotiq_2f

<single robot + gripper + mobile>

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real gripper:=robotiq_2f mobile:=husky

<multi robot>

$ roslaunch dsr_launcher multi_robot_rviz.launch host:=192.168.127.100 port:= 12345 mode:=real model:=m1013

$ roslaunch dsr_launcher multi_robot_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real color:=bule

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real

< multi robot + gripper>

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real gripper:=robotiq_2f

< multi robot + gripper + mobile>

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real gripper:=robotiq_2f mobile:=husky

6. dsr_example

6.1 dsr_example

▪ 기능

- dsr_launcher 를 통하여 구성된 로봇 환경에 따른 로봇 구동 예제를 제공합니다.
  (자세한 로봇 환경 구성은 5장. dsr_launcher 을 참조하시기 바랍니다.)
- 예제 파일은 python 으로 작성 되어 있습니다.
  - python 소스 위치: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

▪ 파라미터

인수명	자료형	기본값	설명
Robot ID	-	127.0.0.1	ROBOT ID . single robot : dsr01 . multi robot : dsr01 부터 순차적으로 dsr02, dsr03, dsr04 …
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509

▪ 예제 : 애뮬레이터 모드

1. 애뮬레이터 실행

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
  or
$ sudo ./DRCF32 <port>     ## 32bits OS
```

2. launch

- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 color:=white
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 color:=blue
```

- single robot in rviz + gazebo

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 color:=white

3. run application node

$ rosrun dsr_example_py single_robot_simple.py dsr01 m1013

---

1. 애뮬레이터 실행

$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF

$ sudo ./DRCF64 <port> ## 64bits OS , defult port = 12345

or

$ sudo ./DRCF32 <port> ## 32bits OS

Multi robot인 경우 로봇 개수에 맞게 각각 다른 port 로 DRCF를 실행합니다.

$ sudo ./DRCF64 12345

새 콘솔 창에서

$ sudo ./DRCF64 12346

2. launch : multi robot

- launch 파일 수정

. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch

. multi_robot_*.launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013

- multi robot in gazebo

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule gripper:=robotiq_2f

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch gripper:=robotiq_2f mobile:=husky

<run application node >

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

```
                robot_id2 = "dsr02"; robot_model2 = "m1013"

$ rosrun dsr_example_py mulit_robot_simple.py
```

- **예제 : 실제 로봇 모드**

1. 로봇 제어기 연결
- IP : 192.168.127.100 , port = 12345
- multi robot 인 경우 각 로봇 제어기의 IP를 다르게 설정합니다.

2. launch
- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 color:=white
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 color:=blue
```

- single robot in rviz + gazebo

```
$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013
color:=white
```

3. run application node

```
$ rosrun dsr_example_py single_robot_simple.py dsr01 m1013
```

---

1. 로봇 제어기 연결
- IP : 192.168.127.100 , port = 12345
- multi robot 인 경우 각 로봇 제어기의 IP를 다르게 설정합니다.

2. launch
- launch 파일 수정
. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch
. multi_robot_*.launch 파일을 각 상황에 맞게 수정합니다.
.. host 와 port 를 올바르게 수정해야 합니다.
- multi robot in rviz

```
$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013
```

- multi robot in gazebo

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule gripper:=robotiq_2f

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch gripper:=robotiq_2f mobile:=husky

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

$ rosrun dsr_example_py mulit_robot_simple.py

6.2 Single Robot

▪ 기능

- Single Robot 구동 예제를 제공합니다.

  (자세한 로봇 환경 구성은 5장. dsr_launcher 을 참조하시기 바랍니다.)

- 예제 파일은 python 으로 작성 되어 있습니다.

  - python 소스 위치: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

▪ 파라미터

인수명	자료형	기본값	설명
Robot ID	-	127.0.0.1	ROBOT ID . single robot : dsr01 . multi robot: dsr01 부터 순차적으로 dsr02, dsr03, dsr04 …
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509

▪ 예제 : 애뮬레이터 모드

1. 애뮬레이터 실행

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
  or
$ sudo ./DRCF32 <port>     ## 32bits OS
```

2. launch

- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 color:=white
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 color:=blue
```

- single robot in rviz + gazebo

```
$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013
```

color:=white

3. run application node

$ rosrun dsr_example_py single_robot_simple.py dsr01 m1013

- **예제: 실제 로봇 모드**

1. 로봇 제어기 연결

- IP : 192.168.127.100 , port = 12345

2. launch

$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 color:=white

- single robot in gazebo

$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 color:=blue

- single robot in rviz + gazebo

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 color:=white

3. run application node

$ rosrun dsr_example_py single_robot_simple.py dsr01 m1013

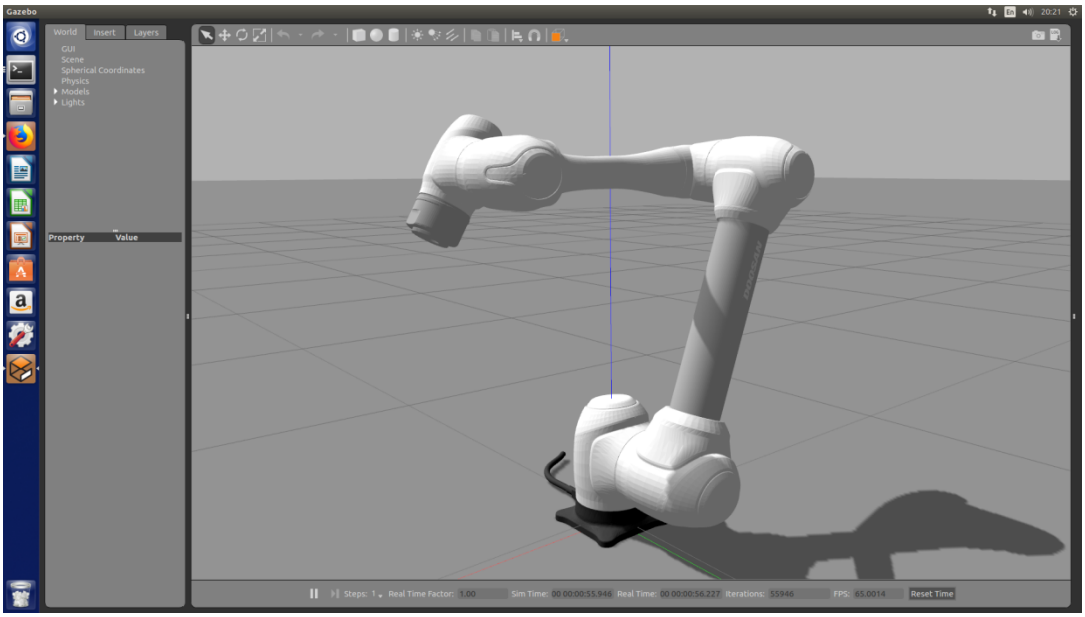


그림 6.2 single robot

6.3 Multi Robot

▪ 기능

- Multi Robot 구동 예제를 제공합니다.

  (자세한 로봇 환경 구성은 5장. dsr_launcher 을 참조하시기 바랍니다.)

- 예제 파일은 python 으로 작성 되어 있습니다.

  - python 소스 위치: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

▪ 파라미터

인수명	자료형	기본값	설명
Robot ID	-	127.0.0.1	ROBOT ID . single robot : dsr01 . multi robot: dsr01 부터 순차적으로 dsr02, dsr03, dsr04 …
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509

▪ 예제: 애뮬레이터 모드

1. 애뮬레이터 실행

$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF

$ sudo ./DRCF64 <port> ## 64bits OS , defult port = 12345

or

$ sudo ./DRCF32 <port> ## 32bits OS

Multi robot인 경우 로봇 개수에 맞게 각각 다른 port 로 DRCF를 실행합니다.

$ sudo ./DRCF64 12345

새 콘솔 창에서

$ sudo ./DRCF64 12346

2. launch

- launch 파일 수정

. $ cd ~/catkin_ws/src/Doosan-robot/dsr_launcher/launch

. multi_robot_*. launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

```
$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013
```

- multi robot in gazebo

```
$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule
```

- multi robot in rviz + gazebo

```
$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch
```

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

```
$ rosrun dsr_example_py multi _robot_simple.py
```

▪ 예제: 실제 로봇 모드

1. 로봇 제어기 연결

- IP : 192.168.127.100 , port = 12345

- multi robot 인 경우 각 로봇 제어기의 IP를 다르게 설정합니다.

2. launch

- launch 파일 수정

. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch

. multi_robot_*. launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

```
$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013
```

- multi robot in gazebo

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

$ rosrun dsr_example_py multi_robot_simple.py

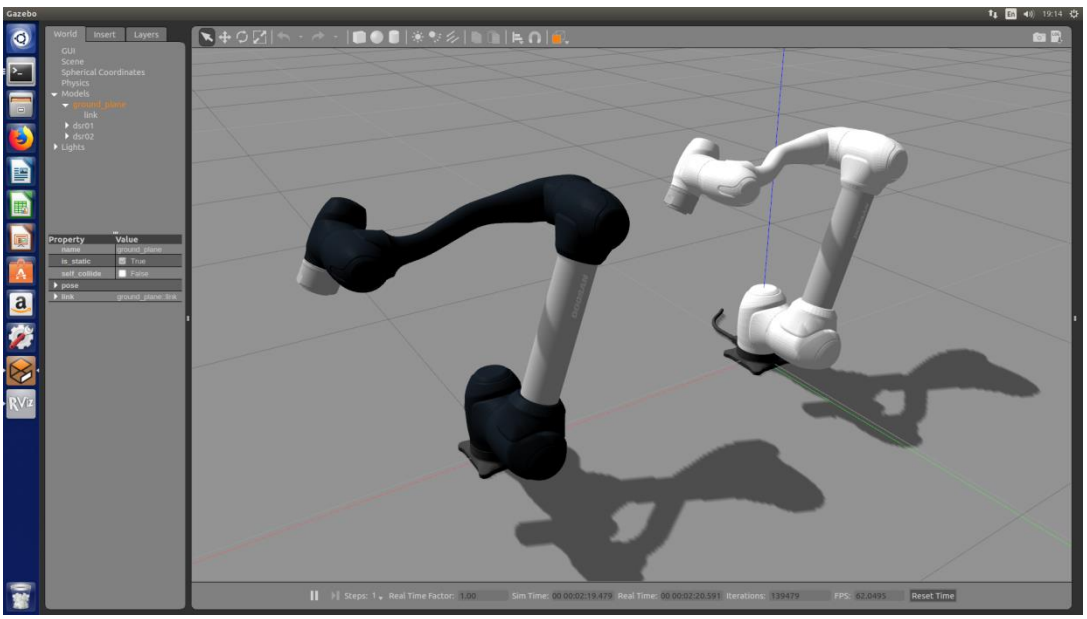


그림 6.3 multi robot

6.4 Gripper

▪ 기능

- Gripper 사용 예제를 제공합니다.

  (자세한 로봇 환경 구성은 5장. dsr_launcher 을 참조하시기 바랍니다.)

- 예제 파일은 python 으로 작성 되어 있습니다.

  - python 소스 위치: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

▪ 파라미터

인수명	자료형	기본값	설명
Robot ID	-	127.0.0.1	ROBOT ID . single robot : dsr01 . multi robot: dsr01 부터 순차적으로 dsr02, dsr03, dsr04 …
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509

▪ 예제: 애뮬레이터 모드

1. 애뮬레이터 실행

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
```

or

```
$ sudo ./DRCF32 <port>     ## 32bits OS
```

2. launch : single robot + gripper

- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 gripper:=robotiq_2f
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013
gripper:=robotiq_2f
```

- single robot in rviz + gazebo

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 gripper:=robotiq_2f

4. run application node

$ rosrun dsr_example_py pick_and_place.py

---

1. 애뮬레이터 실행

$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF

$ sudo ./DRCF64 <port> ## 64bits OS , defult port = 12345

or

$ sudo ./DRCF32 <port> ## 32bits OS

Multi robot인 경우 로봇 개수에 맞게 각각 다른 port 로 DRCF를 실행합니다.

$ sudo ./DRCF64 12345

새 콘솔 창에서

$ sudo ./DRCF64 12346

2. launch: multi robot + mobile

- launch 파일 수정

. $ cd ~/catkin_ws/src/Doosan-robot/dsr_launcher/launch

. multi_robot_*. launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013 gripper:=robotiq_2f

- multi robot in gazebo

$ roslaunch dsr_launcher multi _robot_gazebo.launch color:=bule gripper:=robotiq_2f

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi _robot_rviz_gazebo.launch gripper:=robotiq_2f

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

```
$ rosrun dsr_example_py pick_and_place.py
```

- **예제: 실제 로봇 모드**

1. 로봇 제어기 연결

- IP : 192.168.127.100 , port = 12345

2. launch : single robot + gripper

- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 gripper:=robotiq_2f
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013
gripper:=robotiq_2f
```

- single robot in rviz + gazebo

```
$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013
gripper:=robotiq_2f
```

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

. Robotiq-2Finger gripper가 ROS가 작동하고 있는 디바이스의 Serial Port와 연결되어 있어야 합니다.

```
$ rosrun serial_example_node serial_example_node ttyUSB0 115200
$ rosrun dsr_example_py real_pick_and_place.py
```

---

1. 로봇 제어기 연결

- IP : 192.168.127.100 , port = 12345

- multi robot 인 경우 각 로봇 제어기의 IP를 다르게 설정합니다.

2. launch : multi robot + gripper

- multi robot in rviz

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013 gripper:=robotiq_2f

- multi robot in gazebo

$ roslaunch dsr_launcher multi _robot_gazebo.launch color:=bule gripper:=robotiq_2f

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch gripper:=robotiq_2f

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

$ rosrun serial_example_node serial_example_node ttyUSB0 115200

$ rosrun dsr_example_py real_pick_and_place.py

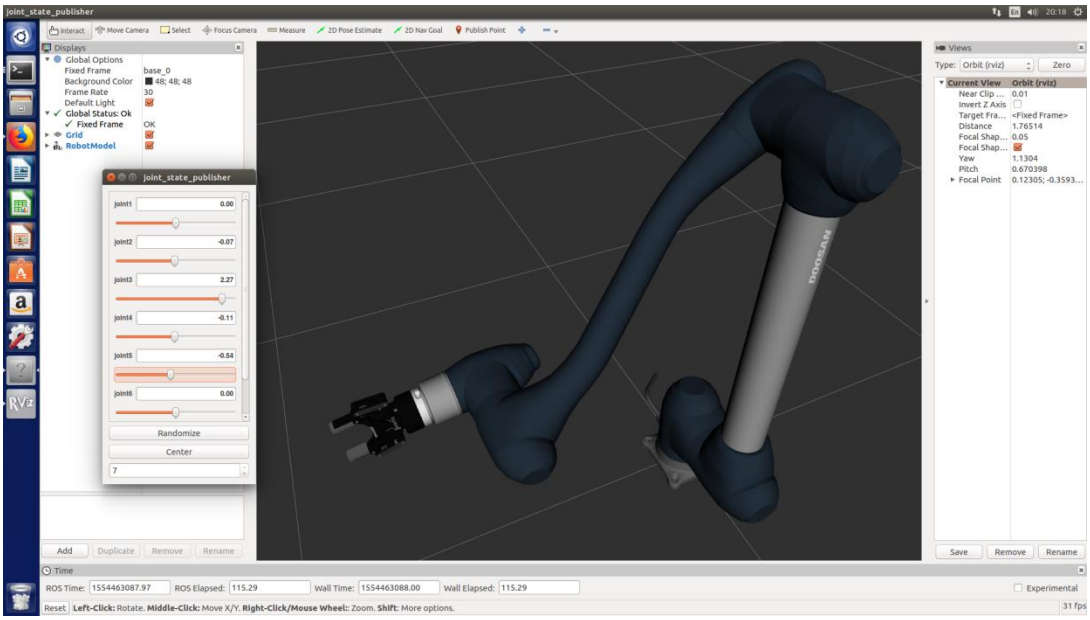


그림 6.4 robot + gripper

6.5 Mobile robot

▪ 기능

- 모바일 로봇 예제를 제공합니다.

  (자세한 로봇 환경 구성은 5장. dsr_launcher 을 참조하시기 바랍니다.)

- 예제 파일은 python 으로 작성 되어 있습니다.

  - python 소스 위치: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

▪ 파라미터

인수명	자료형	기본값	설명
Robot ID	-	127.0.0.1	ROBOT ID . single robot : dsr01 . multi robot: dsr01 부터 순차적으로 dsr02, dsr03, dsr04 …
model	-	m1013	로봇 모델 (4종) . m0609, m0617, m1013, m1509

▪ 예제: 애뮬레이터 모드

1. 애뮬레이터 실행

```
$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF
$ sudo ./DRCF64 <port>     ## 64bits OS , defult port = 12345
  or
$ sudo ./DRCF32 <port>     ## 32bits OS
```

2. launch : single robot + mobile

- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 mobile:=husky
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 mobile:=husky
```

- single robot in rviz + gazebo

```
$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013
```

mobile:=husky

3. run application node

$ rosrun dsr_example_py single_robot_moblie.py

---

1. 애뮬레이터 실행

$ cd ~/catkin_ws/doosan-robot/common/bin/DRCF

$ sudo ./DRCF64 <port> ## 64bits OS , defult port = 12345

or

$ sudo ./DRCF32 <port> ## 32bits OS

Multi robot인 경우 로봇 개수에 맞게 각각 다른 port 로 DRCF를 실행합니다.

$ sudo ./DRCF64 12345

새 콘솔 창에서

$ sudo ./DRCF64 12346

2. launch: multi robot + mobile

- launch 파일 수정

. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch

. multi_robot_*. launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013 mobile:=husky

- multi robot in gazebo

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule mobile:=husky

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch mobile:=husky

3. run application node

- 예제 파일 수정

. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

```
$ rosrun dsr_example_py multi_robot_mobile.py
```

- **예제: 실제 로봇 모드**

1. 로봇 제어기 연결
- IP : 192.168.127.100 , port = 12345

2. launch : single robot + mobile
- single robot in rviz

```
$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 mobile:=husky
```

- single robot in gazebo

```
$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 mobile:=husky
```

- single robot in rviz + gazebo

```
$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013
mobile:=husky
```

3. run application node
- 예제 파일 수정
. 구동하고자 하는 예제 파일을 열어 ROBOT_ID 와 모델을 맞게 수정합니다.
.. ex>
robot_id1 = "dsr01"; robot_model1 = "m1013"
robot_id2 = "dsr02"; robot_model2 = "m1013"

```
$ rosrun dsr_example_py single_robot_moblie.py
```

---

1. 로봇 제어기 연결
- IP : 192.168.127.100 , port = 12345
- multi robot 인 경우 각 로봇 제어기의 IP를 다르게 설정합니다.

2. launch : multi robot + mobile
- launch 파일 수정
. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch
. multi_robot_*. launch 파일을 각 상황에 맞게 수정합니다.

.. host 와 port 를 올바르게 수정해야 합니다.

- multi robot in rviz

$ roslaunch dsr_launcher multi_robot_rviz.launch model:=m1013 mobile:=husky

- multi robot in gazebo

$ roslaunch dsr_launcher multi_robot_gazebo.launch color:=bule mobile:=husky

- multi robot in rviz + gazebo

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch mobile:=husky

3. run application node

$ rosrun dsr_example_py multi_robot_mobile.py

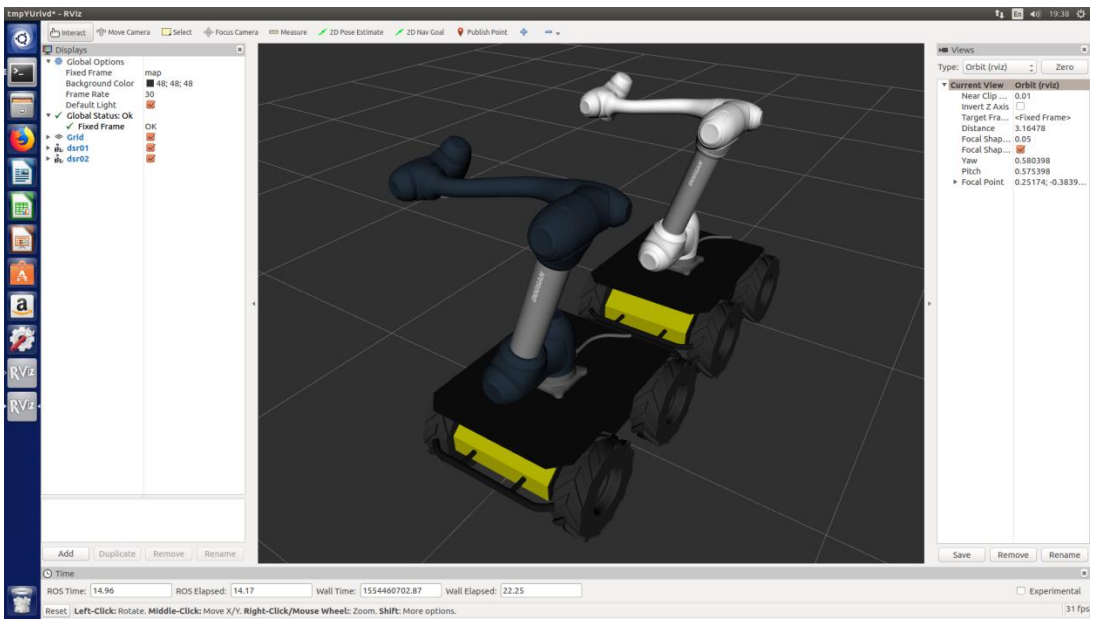


그림 6.5robot on mobile

7. dsr_msgs

7.1 Topic

7.1.1 RobotState.msg

- **기능**

로봇 상태 토픽 메시지

- **인수**

인수명	자료형	기본값	설명
robot_state	int32		로봇 상태 : enum.ROBOT_STATE 참조.
robot_state_str	string		로봇 상태를 string으로 표기
current_posj	float64[6]		로봇의 현재 joint 위치
current_posx	float64[6]		로봇의 현재 Task 위치
io_control_box	int32		로봇제어기의 디지털 입력 상태
access_control	int32		제어권 상태: enum.ACCESS.CONTROL 참조.
homming_completed	bool		로봇의 homming 여부
tp_initialized	bool		TP 초기화 상태
speed	int8		로봇의 현재 속도
mastering_need	int8		로봇 마스터링 필요 여부
drl_stopped	bool		DRL(Doosan Robot Languge) 정지 상태
disconnected	bool		통신 접속 상태

enum.ROBOT_STATE

순번	상수명	설명
0	STATE_INITIALIZING	T/P 어플리케이션에 의해서 자동으로 진입하는 상태로

순번	상수명	설명
		각종 파라미터 설정을 위한 초기화 상태임.
1	STATE_STANDBY	운용 가능한 기본 상태 지령 대기 상태임
2	STATE_MOVING	지령 대기 상태에서 지령 수신 후 동작시 자동으로 전환되는 지령 동작 상태임. 동작 완료시 자동 지령 대기 상태로 전환됨.
3	STATE_SAFE_OFF	기능 및 동작 오류로 인한 로봇 정지 모드로, 서보 오프 상태(제어 정지 후 모터 및 브레이크 전원을 차단한 상태)임
4	STATE_TEACHING	직접교시 상태
5	STATE_SAFE_STOP	기능 및 동작 오류로 인한 로봇 정지 모드로, 안전 정지 상태(제어 정지만 수행한 상태, 자동모드인 경우 프로그램 일시 정지 상태)
6	STATE_EMERGENCY_STOP:	비상 정지 상태
7	STATE_HOMMING	홈밍 모드 상태(로봇을 하드웨어적으로 정렬하는 상태).
8	STATE_RECOVERY	로봇 구동 범위 초과 등과 같은 오류로 인한 로봇 정지시, 구동 범위 이내로 이동시키기 위한 복구 모드 상태임.
9	eSTATE_SAFE_STOP2	eSTATE_SAFE_STOP 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
10	STATE_SAFE_OFF2	eSTATE_SAFE_OFF 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
11	STATE_RESERVED1	예약 사용
12	STATE_RESERVED2	예약 사용

enum.ROBOT_STATE

순번	상수명	설명
0	MANAGE_ACCESS_CONTROL_FORCE_REQUEEST	제어권 강제 회수 메시지 송신
1	MANAGE_ACCESS_CONTROL_REQUEST,	제어권 이양 요청 메시지 송신
2	MANAGE_ACCESS_CONTROL_RESPONSE_YES	제어권 이양 승락 메시지 송신
3	MANAGE_ACCESS_CONTROL_RESPONSE_NO	제어권 이양 거절 메시지 송신

7.1.2 RobotStop.msg

- **기능**

로봇 정지 토픽 메시지

- **인수**

인수명	자료형	기본값	설명
stop_mode	int32		robot stop mode : enum.STOP _MODE 참조.

enum.STOP_MODE

순번	상수명	설명
0	STOP_TYPE_QUICK_STO,	내부 예약 사용
1	STOP_TYPE_QUICK	빠른 정지(모션 궤적 유지)
2	STOP_TYPE_SLOW	느린 정지(모션 궤적 유지)
3	STOP_TYPE_HOLD	긴급 정지
	STOP_TYPE_EMERGENCY	긴급 정지

7.1.3 RobotError.msg

- **기능**

로봇 로그 및 알람 토픽 메시지

- **인수**

인수명	자료형	기본값	설명
Level	int32	-	로그 레벨 : enum.LOG _LEVEL 참조.
Group	int32	-	로그 그룹 : enum.LOG _GROUP 참조.
Code	int32	-	error code
msg1	string	-	error msg 1
msg2	string	-	error msg 2
msg3	string	-	error msg 3

enum.LOG_LEVEL

순번	상수명	설명
0	LOG_LEVEL_RESERVED	내부 예약 상태
1	LOG_LEVEL_SYSINFO	단순 기능 및 동작 오류에 대한 정보용 메시지
2	LOG_LEVEL_SYSWARN	단순 기능 및 동작 오류로 인한 로봇이 정지된 상태
3	LOG_LEVEL_SYSERROR	안전 이슈나 장치 오류로 인한 로봇이 정지된 상태

enum.LOG_GROUP

순번	상수명	설명
0	LOG_GROUP_RESERVED	
1	LOG_GROUP_SYSTEMFMK	하위 제어기(프레임워크)
2	eLOG_GROUP_MOTIONLIB,	하위 제어기(알고리즘)
3	LOG_GROUP_SMARTTP	상위 제어기 프로그램(GUI)

순번	상수명	설명
4	LOG_GROUP_INVERTER	로봇 인버터 보드
5	LOG_GROUP_SAFETYCONTROLLER	안전 보드(Safety Controller)

7.2 Service/motion

7.2.1 MoveJoint.srv

- **기능**

로봇제어기에서 로봇을 현재 관절위치에서 목표 관절위치까지 이동시키기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
pos	float64[6]	-	6개 축에 대한 목표 관절 위치
vel	float64	-	속도
acc	float64	-	가속도
time	float64	0.0	도달 시간 [sec]
radius	float64	0.0	blending시 radius
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	• BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

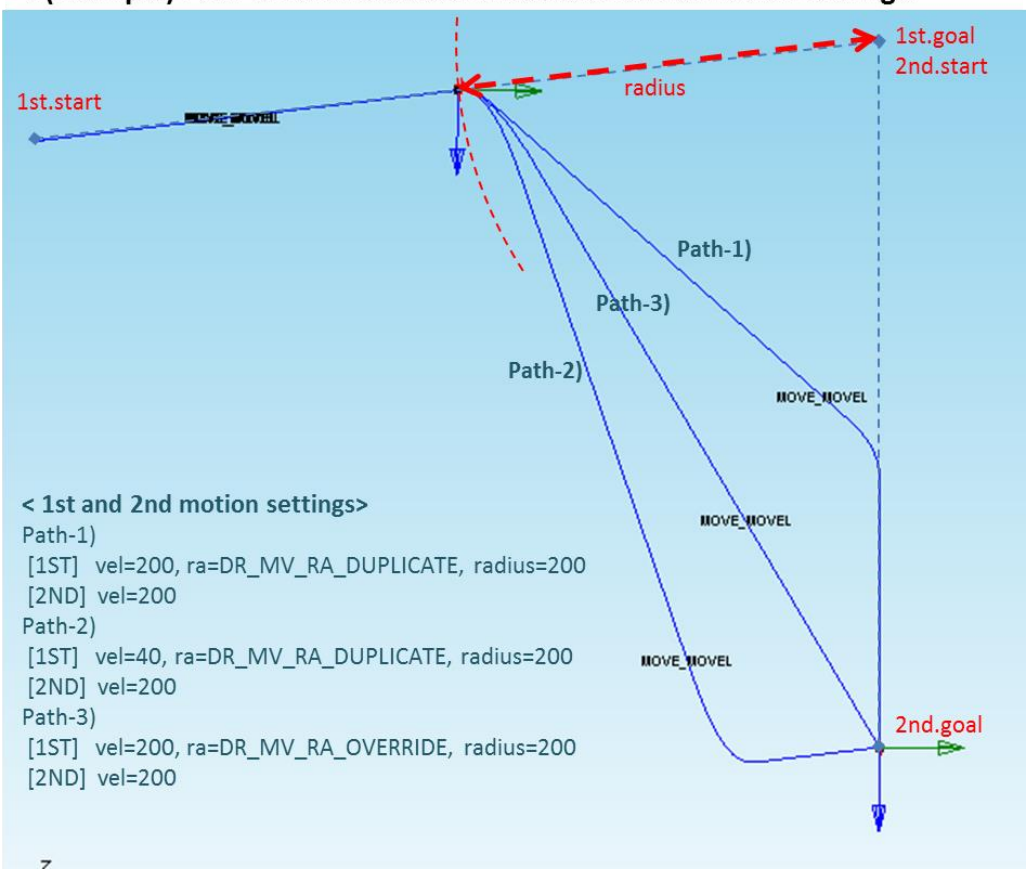
알아두기

· time 지정 시, vel, acc 를 무시하고 time 기준으로 처리됩니다.

주의

blendType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 radius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는 잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 참고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.2 MoveLine.srv

▪ 기능

로봇제어기에서 로봇을 작업 공간 안에서 목표 위치(pos)로 직선을 따라 이동시키기 위한 서비스입니다.

▪ 인수

인수명	자료형	기본값	설명
pos	float64[6]	-	6개 축에 대한 목적 TCP 위치
vel	float64[2]	-	선속도, 각속도
acc	float64[2]	-	선가속도, 각가속도
time	float64	0.0	도달 시간 [sec] * time 지정 시, vel, acc를 무시하고 time 기준으로 처리
radius	float64	0.0	blending시 radius
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	• BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

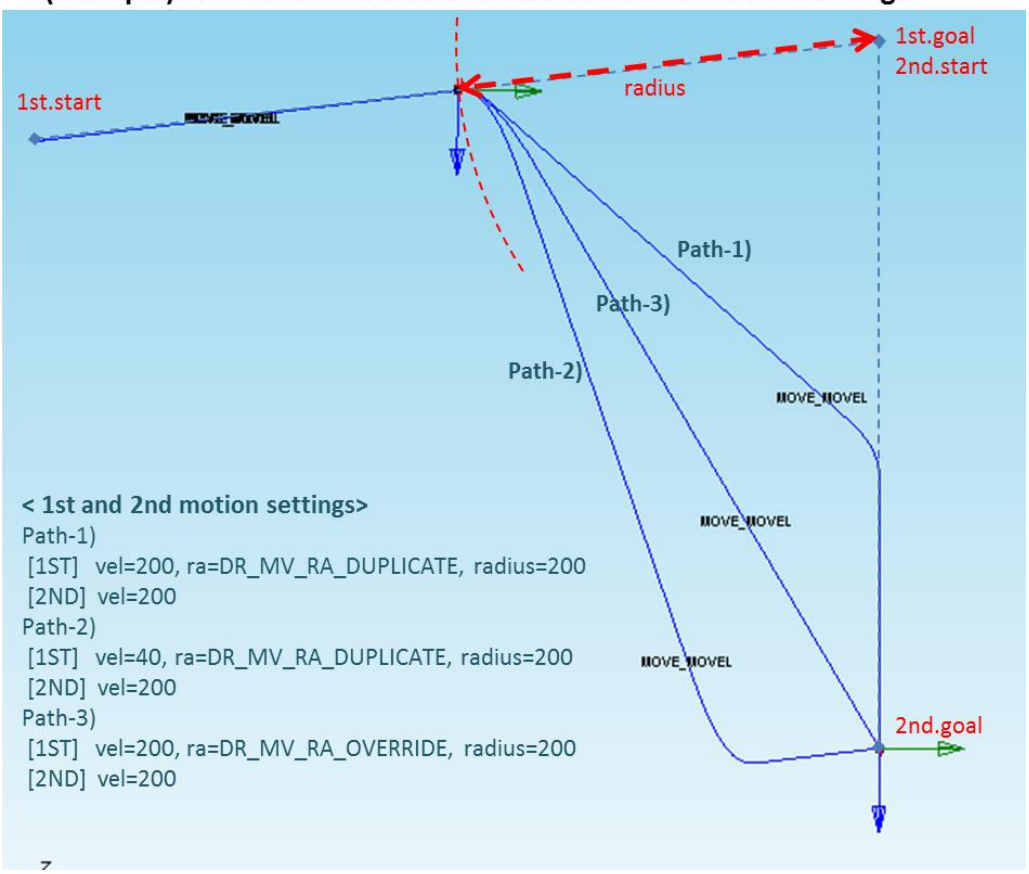
알아두기

- vel 에 하나의 인자를 입력한 경우(예를들어, vel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- acc 에 하나의 인자를 입력한 경우(예를들어, acc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- time 지정 시, vel, acc 를 무시하고 time 기준으로 처리됩니다.

주의

blendType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 radius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는 잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 참고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.3 MoveJointx.srv

▪ 기능

로봇제어기에서 로봇을 관절 공간 안에서 목표 위치로 이동시키기 위한 서비스 입니다. 목표 위치는 작업공간 상의 위치임으로 MoveL과 동일하게 이동하지만 로봇의 모션은 관절공간에서 이루어지기 때문에 목표 위치까지 직선경로가 보장되지 않습니다. 추가적으로 하나의 작업공간좌표에 대응하는 8가지의 관절조합형태(robot configuration)중 하나를 iSolutionSpace(solution space)에 지정해야 합니다.

▪ 인수

인수명	자료형	기본값	설명
pos	float64[6]	-	6개 축에 대한 목적 TCP 위치
vel	float64	-	속도
acc	float64	-	가속도
time	float64	0.0	도달 시간 [sec]
radius	float64	0.0	blending시 radius
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	• BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
sol	int8	0	관절조합형태(아래 설명 참조)
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- time 을 지정할 경우 vel, acc 를 무시하고 time 기준으로 처리된다.
- 상대모션으로 입력하는 경우(eMoveMode = MOVE_MODE_RELATIVE), 선행모션에 블렌딩을 사용하는 경우 에러가 발생하므로 MoveJoint() 또는 MoveLine()을 이용하여 블렌딩하는 것을 권장한다.
- 옵션 blendType 및 vel / acc 에 따른 블렌딩을 수행할 경우 MoveJoint.srv, MoveLine.srv 설명을 참조하십시오.

- **Robot configuration (형태 vs. solution space)**

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip
7	111	Righty	Above	Flip

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.4 MoveCircle.srv

▪ 기능

로봇 제어기에서 작업공간을 기준으로 로봇이 현재 위치에서 경유 지점를 지나 목표 위치까지 원호 또는 지정한 각도로 원호를 따라 이동시키기 위한 서비스입니다.

▪ 인수

인수명	자료형	기본값	설명
pos	std_msgs/Float64MultiArray[]	-	target[2][6] • 경유 지점 • 목표 위치
vel	float64[2]	-	선속도, 각속도
acc	float64[2]	-	선가속도, 각가속도
time	float64	0.0	도달 시간 [sec]
radius	float64	0.0	blending시 radius
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
angle1	float64	0.0	angle1
angle2	float64	0.0	angle2
blendType	int8	0	• BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

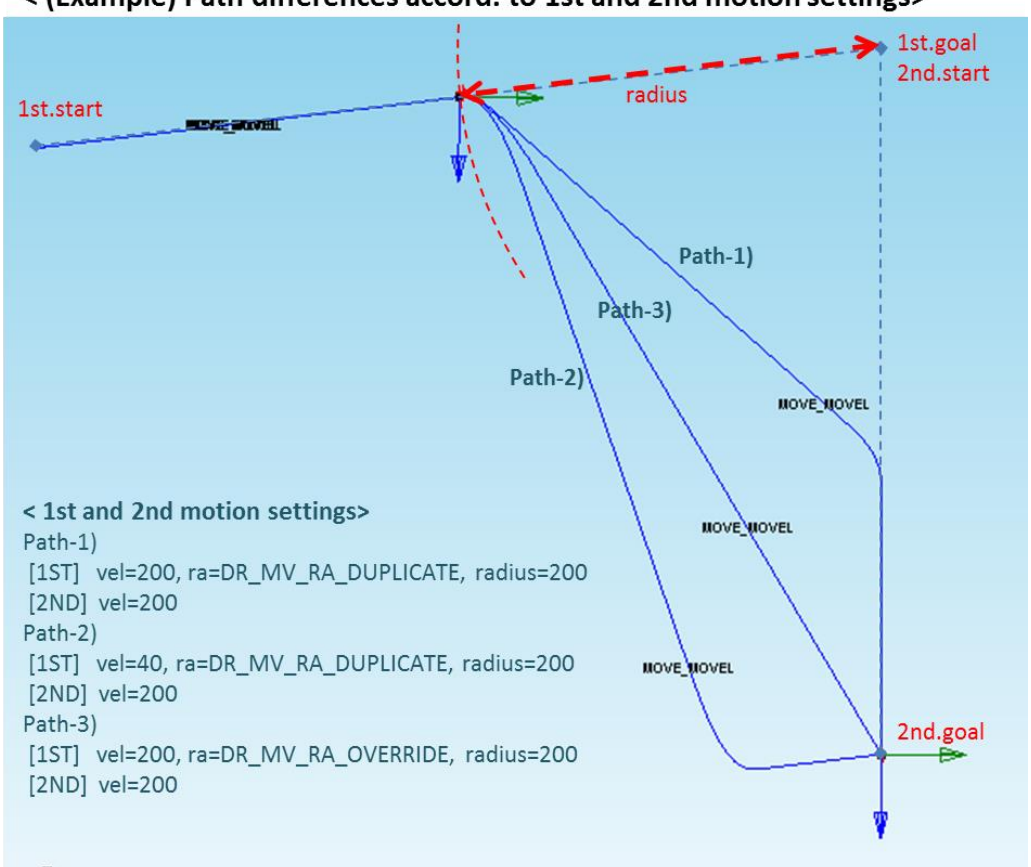
- vel 에 하나의 인자를 입력한 경우(예를들어, vel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- acc 에 하나의 인자를 입력한 경우(예를들어, acc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- time 지정 시, vel, acc 를 무시하고 time 기준으로 처리됩니다.
- mode 가 MOVE_MODE_RELATIVE 인 경우 pos[0] 과 pos[1] 는 각각 앞 선 위치값에 대한 상대좌표로 정의됩니다. (pos[0]은 시작점 대비 상대좌표, pos[1]는 pos[0]대비 상대좌표)

- angle1 이 0 보다 크고, angle2 이 0 인 경우 angle1 은 Circular path 상의 총 회전각이 적용됩니다.
- angle1 과 angle2 가 0 보다 큰 경우, angle1 은 circular path 상에서 정속으로 이동하는 총 회전각을, angle2 는 가속과 감속을 위한 회전 구간의 회전각을 의미합니다. 이때 총 이동각은 angle1+ 2 X angle2 만큼 circular path 상을 움직입니다.

⚠ 주의

blendType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 radius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는 잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 참고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.5 MoveSplineJoint.srv

▪ 기능

로봇 제어기에서 로봇을 현재 위치에서 관절공간의 경유점들을 거쳐 목표위치(마지막 경유점)까지 연결되는 스플라인 곡선경로를 따라 이동시키기 위한 서비스 입니다. 입력된 속도/가속도는 경로 중 최대 속도/가속도를 의미하며 입력되는 경유점의 위치에 따라 모션 중의 감속 또는 가속이 결정됩니다.

▪ 인수

인수명	자료형	기본값	설명
pos	std_msgs/Float64MultiArray[]	-	target pos [100][6] 최대 100개까지의 경유점 리스트
posCnt	int8	-	유효 경유점 개수
vel	float64	-	속도
acc	float64	-	가속도
time	float64	0.0	도달 시간 [sec]
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- time 을 지정할 경우 vel, acc 를 무시하고 time 기준으로 처리됩니다.
- mode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.6 MoveSplineTask.srv

▪ 기능

로봇 제어기에서 로봇을 현재 위치에서 작업공간의 경유점들을 거쳐 목표위치(마지막 경유점)까지 연결되는 스플라인 곡선경로를 따라 이동시키기 위한 서비스 입니다. 입력된 속도/가속도는 경로 중 최대 속도/가속도이며 정속모션 옵션을 선택할 경우 조건에 따라 입력한 속도로 정속도의 모션을 수행합니다.

▪ 인수

인수명	자료형	기본값	설명
pos	std_msgs/Float64MultiArray[]	-	target pos [100][6] 최대 100개까지의 경유점 리스트
posCnt	int8	-	유효 경유점 개수
vel	float64[2]	-	선속도, 각속도
acc	float64[2]	-	선가속도, 각가속도
time	float64	0.0	도달 시간 [sec]
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
opt	int8	0	• SPLINE_VELOCITY_OPTION_DEFAULT=0 • SPLINE_VELOCITY_OPTION_CONST=1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- vel 에 하나의 인자를 입력한 경우(예를들어, vel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- acc 에 하나의 인자를 입력한 경우(예를들어, acc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- time 지정 시, vel, acc 를 무시하고 time 기준으로 처리됩니다
- mode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)

· 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

⚠ 주의

opt 을 SPLINE_VELOCITY_OPTION_CONST 옵션(등속모션)을 선택할 경우 입력된 경유점 간 거리와 속도 조건에 따라 등속모션을 사용할 수 없을 수 있으며, 이 경우에 변속모션 (opt =SPLINE_VELOCITY_OPTION_DEFAULT)으로 자동 전환됩니다.

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.7 MoveBlending.srv

▪ **기능**

로봇 제어기에서 하나 이상의 라인 또는 원호 구성된 경로 정보을 받아 로봇을 경로 정보에 설정된 blending radius로 블렌딩하여 등속으로 이동시키기 위한 서비스 입니다.

▪ **인수**

인수명	자료형	기본값	설명
pos	std_msgs/Float64MultiArray[]	-	(pos1[6]:pos2[6]:type[1]:radius[1]) x 50(max) 최대 50개까지의 경로 정보
posCnt	int8		유효 경로 정보 개수
vel	float64[2]	-	선속도, 각속도
acc	float64[2]	-	선가속도, 각가속도
time	float64	0.0	도달 시간 [sec]
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
mode	int8	0	• MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- vel 에 하나의 인자를 입력한 경우(예를들어, vel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- acc 에 하나의 인자를 입력한 경우(예를들어, acc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- time 지정 시, vel, acc 를 무시하고 time 기준으로 처리됩니다
- mode 가 MOVE_MODE_RELATIVE 인 경우 posb list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다.

주의

- posb 에서 blending radius 가 0 인 경우, 사용자 입력 오류가 나타납니다.
- 연속된 Line-Line segment 가 같은 방향을 가질 경우 Line 의 중복입력으로 사용자 입력 오류가 나타납니다.

- 블렌딩 구간에서 조건에 따라 급격하게 방향전환이 발생하게 되는 경우 급가속을 방지하기 위해 사용자 입력오류가 나타납니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.8 MoveSpiral.srv

▪ 기능

로봇제어기에서 방사형 방향으로 반경이 증가하며 회전하는 Spiral motion과 축 방향으로 병행하면서 이동시키기 위한 서비스 입니다. 현재 위치에서 eMoveReference 로 지정한 좌표계 상의 axis 방향에 수직인 평면에서의 나선궤적과 axis 방향으로의 직선궤적을 동시에 따라 이동합니다.

▪ 인수

인수명	자료형	기본값	설명
revolution	float64	-	총 회전수 [revolution]
maxRadius	float64		spiral 최종 반경 [mm]
maxLength	float64		axis 방향으로 이동하는 거리 [mm]
vel	float64[2]	-	선속도, 각속도
acc	float64[2]	-	선가속도, 각가속도
time	float64	0.0	총 수행시간 [sec]
taskAxis	int8	0	• TASK_AXIS_X = 0 • TASK_AXIS_Y = 1 • TASK_AXIS_Z = 2
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- revolution 는 spiral 모션의 총 회전수를 의미합니다.
- maxRadius 는 spiral 모션의 최대 반경을 의미합니다.
- maxLength 는 모션 동안 axis 방향으로 병진하는 거리를 의미합니다. 단, 음수인 경우 –axis 방향 병진합니다.
- vel 은 spiral 모션의 이동 속도를 의미합니다.
- acc 는 spiral 모션의 이동 가속도를 의미합니다.
- time 을 지정할 경우 vel, acc 를 무시하고 time 기준으로 처리됩니다.
- taskAxis 는 Spiral 모션이 정의하는 평면에 수직인 축을 정의합니다.

- ref 는 spiral 모션이 정의하는 기준 좌표계를 의미합니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

⚠ **주의**

- 경로 생성 시 Spiral 경로에 의한 회전각 가속도를 연산하여 값이 큰 경우 안정적인 모션을 위하여 에러가 발생나타날 수 있습니다. 이 경우 vel, acc 또는 time 값을 작게 조정하는 것을 권장합니다.

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.9 MovePeriodic.srv

▪ 기능

로봇제어기에서 현재 위치에서 시작하는 상대 모션으로 입력된 기준 좌표계(eMoveReference)의 각 축(병진 및 회전)에 대한 Sine 함수 기반으로 주기 모션을 수행합니다. 각 axis 별 모션의 특성은 fAmplitude와 fPeriodic 에 의해 결정되고, 가감속 시간과 총 모션 시간은 주기, 반복, 횟수에 의해 설정됩니다.

▪ 인수

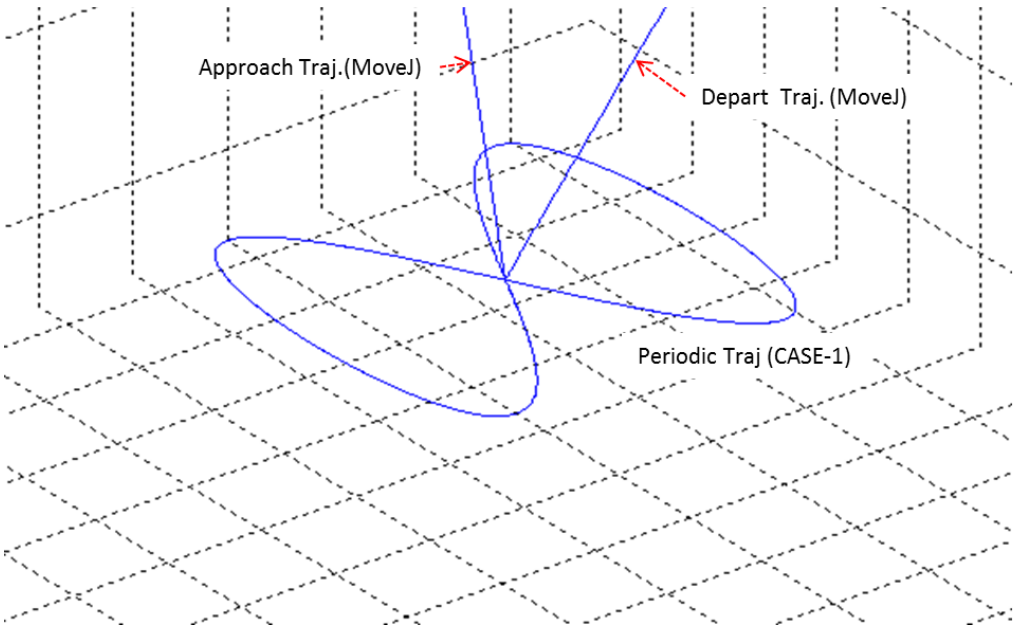
인수명	자료형	기본값	설명
amp	float64[6]	-	Amplitude(-amp에서 +amp사이 모션) [mm] or [deg]
periodic	float64[6]	-	period(1주기 소요 시간)[sec]
acc	float64	-	Acceleration
time	float64	-	Acc-, dec- time [sec]
repeat	int8	-	반복 횟수
ref	int8	0	• MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

알아두기

- amp 는 진폭(amplitude)을 의미하며, 각 축(x, y, z, rx, ry, rz) 별로 amp 를 값으로 하는 6 개 원소의 list 형태로 입력해야 합니다. 단, 주기 모션을 진행하지 않는 축 방향은 amp 를 0 으로 입력해야 합니다.
- periodic 는 해당 방향 모션의 1 회 반복 시간을 의미하며, 각 축(x, y, z, rx, ry, rz) 별 period 를 값으로 하는 총 6 개 원소의 list 형태로 입력하거나 대표값을 입력해야합니다.
- acc 은 주기모션의 시작과 끝의 가속 및 감속 시간을 의미합니다. 입력된 가감속시간과 최대주기*1/4 중 큰 값이 적용됩니다. 입력된 가감속 시간이 전체모션시간의 1/2 을 초과하는 경우 에러가 발생합니다.
- repeat 은 가장 큰 period 값을 가지는 축(기준 축)의 반복 횟수를 정의하며, 이에 따라 총 모션 시간이 결정됩니다. 나머지 축의 반복 횟수는 모션 시간에 따라 자동결정됩니다.모션이 정상 종료되는 경우 종료 위치가 시작 위치와 일치하게 하도록 나머지 축 모션은 기준 축 모션이 종료되기 전에 먼저 종료될 수 있습니다. 모든 축의 모션이 동시 종료되지 않는 경우 감속구간에서의 경로는 이전 경로에서 벗어나게 됩니다. 관련한 사항은 아래 이미지를 참조하십시오

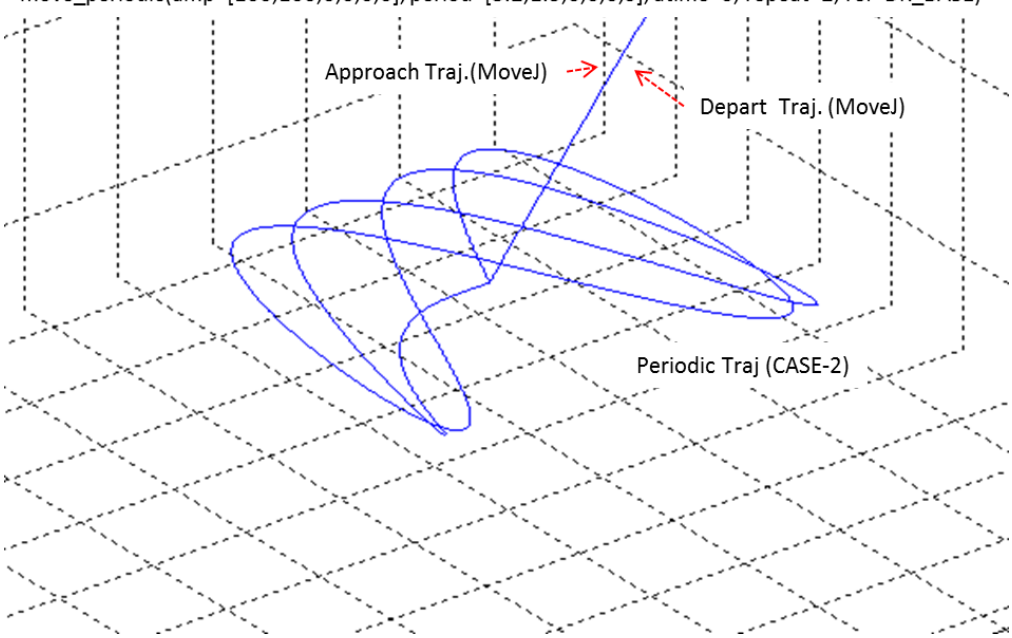
CASE-1) All-axis motions end at the same time

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.6,0,0,0,0], atime=3.1, repeat=2, ref=DR_BASE)



CASE-2) Diff-axis motions end individually

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.5,0,0,0,0], atime=0, repeat=2, ref=DR_BASE)



- ref 는 반복 모션의 기준 좌표계를 의미합니다.
- 모션명령 수행 시 최대속도 에러가 발생하는 경우 다음의 식을 참조하여 진폭 및 주기를 조정할 것을 제안합니다.
  최대속도=진폭(amp)*2*pi(3.14)/주기(period)
  (예, 진폭=10mm, 주기=1 초인 경우 최대속도=62.83mm/sec)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.2.10 MoveWait.srv

- **기능**

로봇제어기에서 선행된 모션명령어의 동작이 종료되기를 기다리기 위한 서비스 입니다. 비동기 모션 명령어와 본 함수를 결합하여 사용하면 동기 모션 명령어와 동일한 동작을 수행할 수 있습니다.

- **인수**

인수명	자료형	기본값	설명
없음	-	-	-

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.3 Service/tcp

7.3.1 ConfigCreateTcp.srv

▪ 기능

로봇 TCP 정보를 안전상 사전에 등록하여 사용하기 위한 서비스 입니다, 본 서비스를 이용하여 등록된 TCP 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능합니다.

▪ 인수

인수명	자료형	기본값	설명
name	string	-	TCP 이름
pos	float64[6]	-	TCP 정보

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.3.2 ConfigDeleteTcp.srv

- **기능**

로봇 제어기에 사전에 등록된 TCP 정보를 삭제하기 위한 서비스 입니다

- **인수**

인수명	자료형	기본값	설명
name	string	-	TCP 이름

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.3.3 GetCurrentTcp.srv

- **기능**

로봇 제어기에서 현재 설정된 TCP 정보를 가져오는 서비스 입니다. 설정된 Tool 정보가 없을 경우, 빈 문자열이 반환됩니다.

- **인수**

인수명	자료형	기본값	설명
없음	-	-	-

- **리턴**

인수명	자료형	기본값	설명
info	string	-	TCP 이름

7.3.4 SetCurrentTcp.srv

- **기능**

로봇 제어기에 사전에 등록되어 있는 TCP 정보 중 현재 장착된 TCP에 대한 정보를 설정하는 서비스 입니다. 현재 장착된 TCP가 없을 경우, 빈 문자열을 전달하면 현재 설정되어 있는 정보가 초기화됩니다.

인수

인수명	자료형	기본값	설명
name	string	-	Tool 이름

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.4 Service/tool

7.4.1 ConfigCreateTool.srv

▪ 기능

로봇 끝단에 장착될 Tool 정보를 안전상 사전에 등록하여 사용하기 위한 서비스 입니다, 본 서비스를 이용하여 등록된 Tool 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능합니다.

▪ 인수

인수명	자료형	기본값	설명
name	string	-	Tool 이름
weight	float		Tool 무게
cog	float64[3]		무게 중심
inertia	float64[6]		관성 정보

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.4.2 ConfigDeleteTool.srv

- **기능**

로봇 제어기에 사전에 등록된 Tool 정보를 삭제하기 위한 서비스 입니다

- **인수**

인수명	자료형	기본값	설명
name	string	-	Tool 이름

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.4.3 GetCurrentTool.srv

- **기능**

로봇 제어기에서 현재 설정된 Tool 정보를 가져오는 서비스 입니다. 설정된 Tool 정보가 없을 경우, 빈 문자열이 반환됩니다.

- **인수**

인수명	자료형	기본값	설명
없음	-	-	-

- **리턴**

인수명	자료형	기본값	설명
info	string	-	Tool 이름

7.4.4 SetCurrentTool.srv

- **기능**

로봇 제어기에 사전에 등록되어 있는 Tool 정보 중 현재 장착된 Tool에 대한 정보를 설정하는 서비스 입니다. 현재 장착된 Tool 이 없을 경우, 빈 문자열을 전달하면 현재 설정되어 있는 정보가 초기화 됩니다.

- **인수**

인수명	자료형	기본값	설명
name	string	-	Tool 이름

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5 Service/io

7.5.1 SetCtlBoxDigitalOutput.srv

▪ 기능

로봇 제어기에서 컨트롤 박스에 장착된 디지털 접점에 신호를 출력하기 위한 서비스 입니다.

▪ 인수

인수명	자료형	기본값	설명
index	int8	-	제어기 내 디지털 출력 접점 번호(1~16)
value	int8		출력 값 : OFF =0, ON =1

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5.2 GetCtlBoxDigitalInput.srv

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 디지털 접점의 신호를 확인하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
index	int8	-	제어기 내 디지털 입력 접점 번호(1~16)

- **리턴**

인수명	자료형	기본값	설명
value	bool	-	OFF =0, ON =1

7.5.3 SetToolDigitalOutput.srv

- **기능**

로봇 제어기에서 로봇 끝단에 장착된 디지털 접점에 신호를 출력하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
index	int8	-	로봇 플랜지 단 디지털 출력 접점 번호(1~6)
value	int8		출력 값 : OFF =0, ON =1

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5.4 GetToolDigitalIntput.srv

- **기능**

로봇 제어기에서 로봇 끝단에 장착된 디지털 접점의 신호를 확인하기 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
index	int8	-	로봇 플랜지 단 디지털 입력 접점 번호(1~6)

- **리턴**

인수명	자료형	기본값	설명
value	bool	-	OFF =0, ON =1

7.5.5 SetCtlBoxAnalogOutputType.srv

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 출력 접점에 대한 채널 모드를 설정하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
channel	int8	-	제어기 내 아날로그 출력 채널 : 1 or 2
mode	int8	-	동작 모드 • current =0 • voltage =1

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5.6 SetCtlBoxAnalogInputType.srv

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 입력 접점에 대한 채널 모드를 설정하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
channel	int8	-	제어기 내 아날로그 입력 채널 : 1 or 2
mode	int8	-	동작 모드 • current =0 • voltage =1

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5.7 SetCtlBoxAnalogOutput.srv

▪ 기능

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 접점에 신호를 출력하기 위한 서비스 입니다.

▪ 인수

인수명	자료형	기본값	설명
channel	int8	-	제어기 내 아날로그 출력 채널 : 1 or 2
value	float64	-	아날로그 신호 출력 • 전류 모드인 경우: 4.0~20.0 [mA] • 전압 모드인 경우: 0~10.0 [V]

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.5.8 GetCtlBoxAnalogInput.srv

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 접점의 신호를 확인하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
channel	int8	-	제어기 내 아날로그 입력 채널 : 1 or 2

- **리턴**

인수명	자료형	기본값	설명
value	float	-	해당 채널의 입력 값 • 전류 모드인 경우: 4.0~20.0 [mA] • 전압 모드인 경우: 0~10.0 [V]

7.6 Service/modbus

7.6.1 ConfigCreateModbus.srv

▪ 기능

로봇 제어기에서 Modbus의 I/O 신호 사전에 등록하여 사용하기 위한 서비스이다, 본 함수를 이용하여 등록된 Modbus I/O 신호 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능합니다.

▪ 인수

인수명	자료형	기본값	설명
name	string	-	modbus signal 이름
ip	string	-	modbus 모듈 ip 주소
port	int8	-	modbus 모듈 port
reg_type	int8	-	modbus 레지스터 타입 • MODBUS_REGISTER_TYPE_DISCRETE_INPUTS • MODBUS_REGISTER_TYPE_COILS • MODBUS_REGISTER_TYPE_INPUT_REGISTER • MODBUS_REGISTER_TYPE_HOLDING_REGISTER
index	int8	-	Modbus signal의 index
value	int8	-	type이 MODBUS_REGISTER_TYPE_COILS 또는 MODBUS_REGISTER_TYPE_HOLDING_REGISTER 일 때 출력값 (그 외 경우에는 무시됩니다.)

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.6.2 ConfigDeleteModbus.srv

- **기능**

로봇 제어기에 사전에 등록된 Modbus I/O 신호 정보를 삭제하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
name	string	-	등록된 modbus 신호의 이름

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.6.3 SetModbusOutput.srv

- **기능**

로봇 제어기에서 Modbus I/O 신호 접점에 신호를 출력하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
name	string	-	modbus 이름
value	int32	-	• Modbus digital I/O 인 경우: 0 or 1 • Modbus analog I/O 인 경우: 데이터

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.6.4 GetModbuInput.srv

- **기능**

로봇 제어기에서 Modbus I/O 신호 접점의 신호를 확인하기 위한 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
name	string	-	modbus 이름

- **리턴**

인수명	자료형	기본값	설명
value	int32	-	• Modbus Digital I/O 인 경우: 0 or 1 • Modbus Analog 모듈인 경우: 데이터

7.7 Service/drl

7.7.1 DrlStart.srv

▪ 기능

로봇 제어기에서 DRL 언어로 구성된 프로그램(태스크)을 실행하기 위한 서비스 입니다.

▪ 인수

인수명	자료형	기본값	설명
robotSystem	int8	-	
Code	string	-	실행 시킬 DRL 프로그램 문자열

▪ 리턴

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

알아두기

- 로봇 운용 상태가 지령 대기상태(STATE_STANDBY)이어야 하며, 로봇 모드가 자동모드일 때 사용해야 정상 동작한다.
- DRL 프로그램 작성은 별로 Programming Manual 문서을 참조해서 작성해야 합니다.

7.7.2 DrlStop.srv

- **기능**

로봇 제어기에서 현재 실행중인 DRL 프로그램(태스크)을 정지하기 위한 서비스입니다. 인자로 받는 eStopType에 따라 다르게 정지하며, 현재 수행하고 있는 구간의 모션을 정지합니다.

- **인수**

인수명	자료형	기본값	설명
stop_mode	int8	-	drl stop mode • STOP_TYPE_QUICK_STO = 0 • STOP_TYPE_QUICK = 1 • STOP_TYPE_SLOW = 2 • STOP_TYPE_HOLD = STOP_TYPE_EMERGENCY = 3

- **리턴**

인수명	자료형	기본값	설명
Success	bool	-	성공 여부 : True or False

7.7.3 DrlPause.srv

- **기능**

로봇제어기에서 현재 실행 중인 DRL 프로그램(태스크)을 일시 정지하기 위한 서비스입니다.

- **인수**

인수명	자료형	기본값	설명
없음	-	-	-

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.7.4 DrlResume.srv

- **기능**

로봇 제어기에서 현재 일시 정지된 DRL 프로그램(태스크)을 재개하기 위한 서비스입니다.

- **인수**

인수명	자료형	기본값	설명
없음	-	-	-

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.8 Service/gripper

7.8.1 SerialSendData.srv

- **기능**

실질적인 serial 통신을 통하여 gripper를 제어합니다.
serial_node_example

- **인수**

인수명	자료형	기본값	설명
data	string	-	전송하고자 하는 스트링

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

7.8.2 RobotiqMove.srv

- **기능**

  시뮬레이터 환경에서 robotiq 사의 gripper를 제어하는 서비스 입니다.

- **인수**

인수명	자료형	기본값	설명
width	float	-	2핑거 그립퍼 폭 : 0.0(open)~0.8(close)

- **리턴**

인수명	자료형	기본값	설명
success	bool	-	성공 여부 : True or False

Doosan Robotics

www.doosanrobotics.com