|  |
| --- |
| HYULIM ROBOT_기본형_최소크기.jpg |
| **HerkuleX ROS Package Manual**  **ROS_cat.png** |
|  |
|  |

**History**

2019-12-10, 진욱빈, 초안 작성

2020-03-11, 진욱빈, 패키지 수정으로 인한 1차 업데이트.

**[목차]**

[1. HerkuleX ROS Package사용 전 준비사항 4](#_Toc17970146)

[2. HerkuleX Package의 구성 7](#_Toc17970148)

[3. HerkuleX Package의 실행 10](#_Toc17970150)

[4. ROS GUI개발도구(rqt)를 활용한 HerkuleX의 제어 13](#_Toc17970150)

[5. HerkuleX 패키지 활용 예시 21](#_Toc17970150)

1. **HerkuleX ROS Package사용 전 준비사항**

HerkuleX ROS Package의 사용을 위해서는 아래와 같은 사전 준비가 필요합니다.

* ROS Kinetic의 설치

사용자 PC의 OS는 Ubuntu16.04 LTS를 권장합니다. 해당 환경에서 ROS Kinetic을 설치해야 하며, 그 과정에 대해서는 ROS wiki에 정리된 웹 페이지를 참고하세요.

* 링크: <http://wiki.ros.org/kinetic/Installation/Ubuntu>
* HerkuleX와 PC간의 통신 연결을 위한 시리얼 인터페이스 장치와 전원공급 장치를 준비하여 아래 그림과 같이 연결합니다.

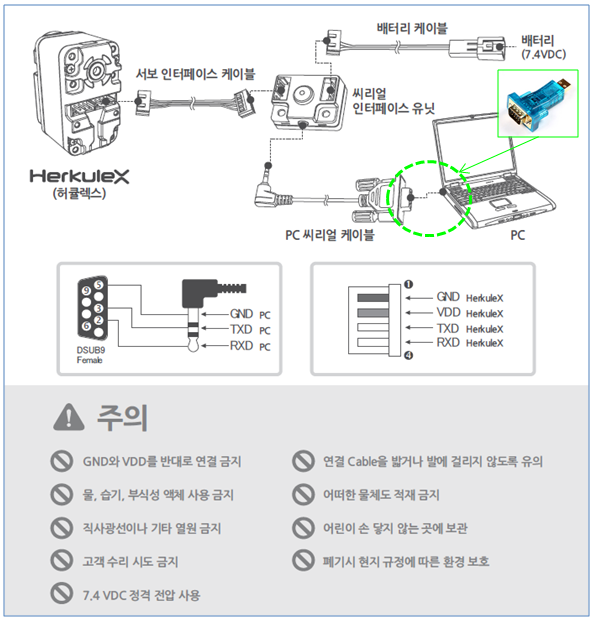


그림 1. HerkuleX와 PC간의 연결 설명서

* 요즘 PC에는 대부분 Serial Port가 없기 때문에 위의 그림과 같이 USB to Serial Device를 추가로 사용해야 합니다.
* 이 USB to Serial Device를 이용하여, PC와 HerkuleX간의 RS232통신을 하기 때문에 Ubuntu환경에서는 ttyUSB권한 설정 및 USB rule설정을 초기에 1회 해주어야 하며, 그 방법은 다음과 같습니다.

1. ttyUSB권한 설정.

HerkuleX의 USB Port에 대한 권한은 ‘sudo chmod 777 ttyUSB0’ 명령어를 통해서 줄 수 있지만, 매번 설정이 번거로우므로 dialout그룹에 추가하는 방법을 이용한다.

* sudo usermod -a -G dialout $USER

\* 위의 명령어를 실행 후 PC의 재 시작을 해야 적용된다.

1. ttyUSB rule의 설정을 위한 심볼릭 링크 ttyUSBx만들기

* 심볼릭 링크를 만드는데 필요한 정보는 아래 3가지 이다.
* Vender ID
* Product ID
* Serial Number
* 위의 3가지 정보는 아래 2개의 명령어로 알아낼 수가 있다.
* $ lsusb

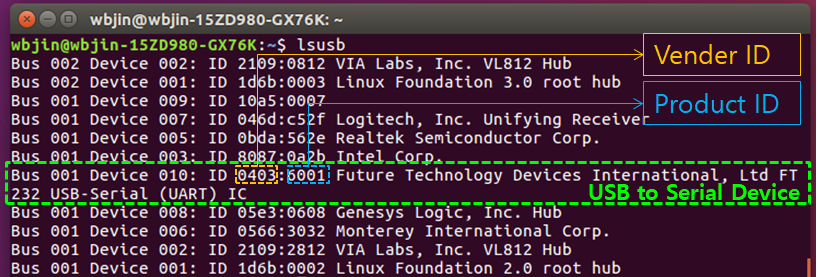


그림 2. USB list

* $ udevadm info -a /dev/ttyUSB0 | grep '{serial}'



그림 3. USB to Serial Device Serial Number

위에 출력된 정보가 USB to Serial Device의 Serial Number이며, 해당 번호는 제품별로 상이하다

* 알아낸 정보를 이용한 .rules file생성.

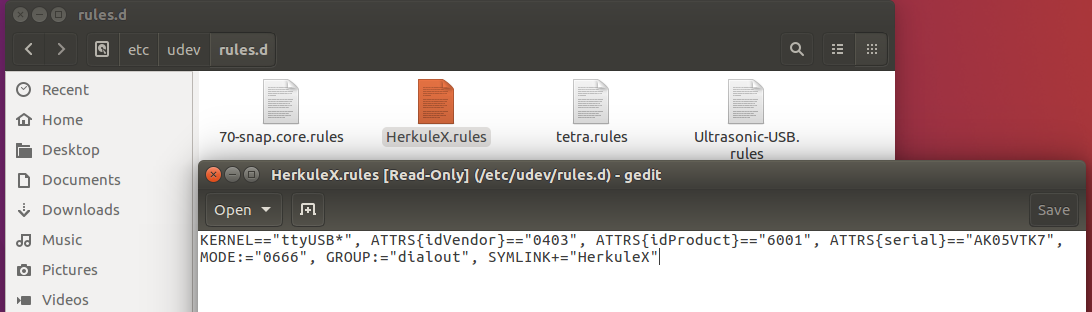


그림 4. USB rules file

‘HerkuleX.rules’ 파일을 ‘/etc/udev/rules.d’ 경로에 생성한 후에 해당 파일의 내용을 아래와 같이 작성한 후 저장한다.

KERNEL=="ttyUSB\*", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial}=="AK05VTK7", MODE:="0666", GROUP:="dialout", SYMLINK+="HerkuleX"

저장완료 후에는 아래와 같은 명령어를 이용하여 작성 내용의 확인이 가능하다.

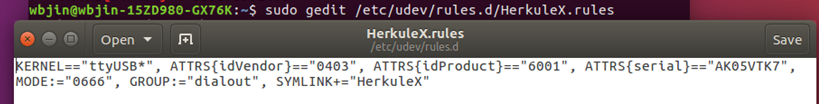


그림 5. HerkuleX.rules file확인

* udev 재시작 명령어를 호출한 후에 PC의 재 시작을 해주어야 적용이 된다.
* $ sudo service udev restart
* 심볼릭 등록의 확인. (아래와 같은 명령어를 이용하여, 심볼릭 링크로 설정한 ttyUSB장치가 tetra로 적용되었는지 여부를 확인한다.)
* $ ll /dev/

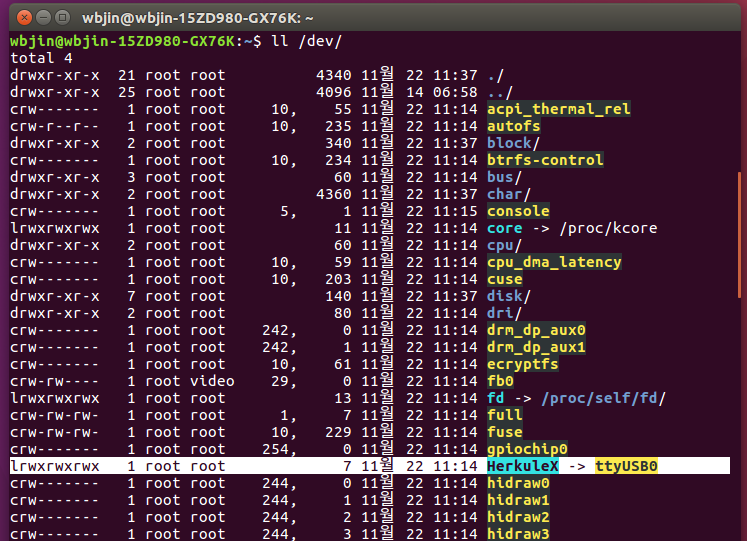


그림 6. Device List확인

1. **HerkuleX ROS Package의 구성**

HerkuleX ROS Package는 ‘rosserial’ Package를 이용하여, PC와 HerkuleX간의 Serial통신을 연결합니다. 때문에 종속성 패키지로 rosserial이 준비되어 있어야 합니다. 또한, HerkuleX Package는 Serail통신의 Board rate를 115,200bps로 고정하였기 때문에 PC에 연결된 HerkuleX의 Board rate또한 115,200bps로 설정되어 있어야 합니다. (HerkuleX Package의 Board rate의 변경을 위해서는 Package의 Code수정이 필요함)

HerkuleX ROS Package의 구성은 아래그림과 같습니다.

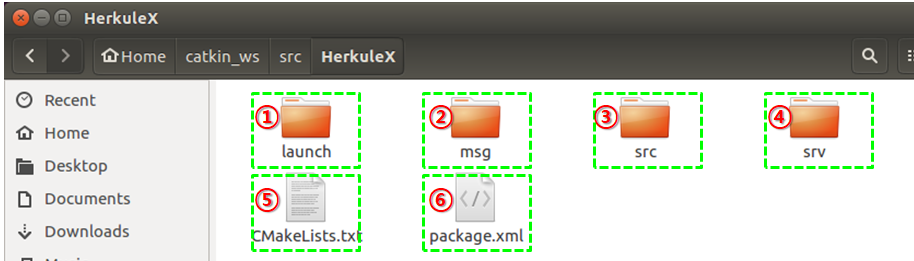


그림 7. HerkuleX ROS Package의 구성.

1. HerkuleX ROS Package의 실행을 위한 Launch file이 들어있는 폴더.
2. HerkuleX Package에서 Publish되는 Message가 정의되어 있는 폴더

* MsgHerkuleX\_EEP.msg: EEP Register의 메시지 정보가 정의되어 있음.
* MsgHerkuleX\_RAM.msg: RAM Register의 메시지 정보가 정의되어 있음.

1. HerkuleX Package의 C++ 소스코드 정보가 들어있는 폴더

* HerkuleX\_node.cpp: 함수들의 정의에 대해서 작성된 소스파일
* HerkuleX\_node.h 함수들의 선언과 사용되는 구조체들에 대해서 작성된 헤더파일

1. HerkuleX Package에서 사용되는 서비스 메시지 정보가 정의되어 있는 폴더

* HerkuleX\_PositionMove.srv: HerkuleX의 위치제어 서비스에 대한 정의.
* 해당 파일에는 위치제어 명령에 사용되는 Input Data의 정의와 Output Data에 대한 정의가 정리되어 있음.

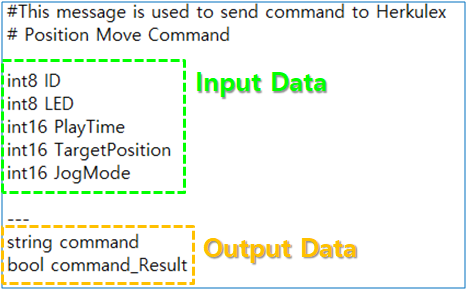


그림 8. Position Move Command Service 정의

* HerkuleX\_VelocityMove.srv: HerkuleX의 속도제어 서비스에 대한 정의.
* 해당 파일에는 속도제어 명령에 사용되는 Input Data의 정의와 Output Data에 대한 정의가 정리되어 있음.

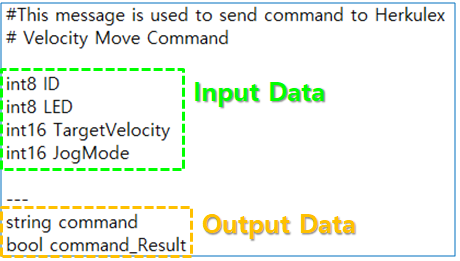


그림 9. Velocity Move Command Service 정의.

* HerkuleX\_RegisterCommand.srv: HerkuleX의 EEP & RAM Register에 해당 주소번지 별 데이터를 입력하여 Read/Write기능을 수행할 수 있는 서비스에 대한 정의.
* 해당 파일에는 Register Control(Read/Write)에 사용되는 Input Data의 정의와 Output Data에 대한 정의가 정리되어 있음.

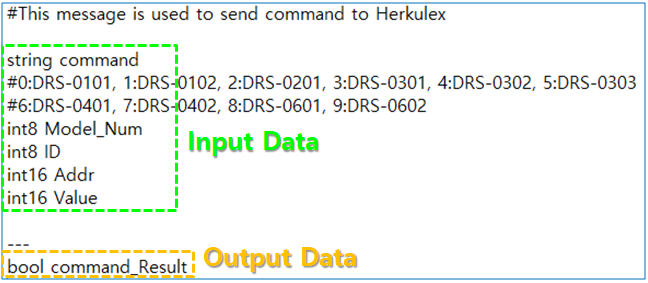


그림 10. Register Command Service 정의.

1. HerkuleX Package의 CMakeList파일.

* 해당 파일에서는 빌드에 필요한 관련 패키지의 탐색, ROS메시지와 서비스 생성기 정의, catkin관련 빌드 정보 및 실행 파일을 지정하는 순으로 구성되어 있으며, 해당 내용은 아래 그림과 같습니다.

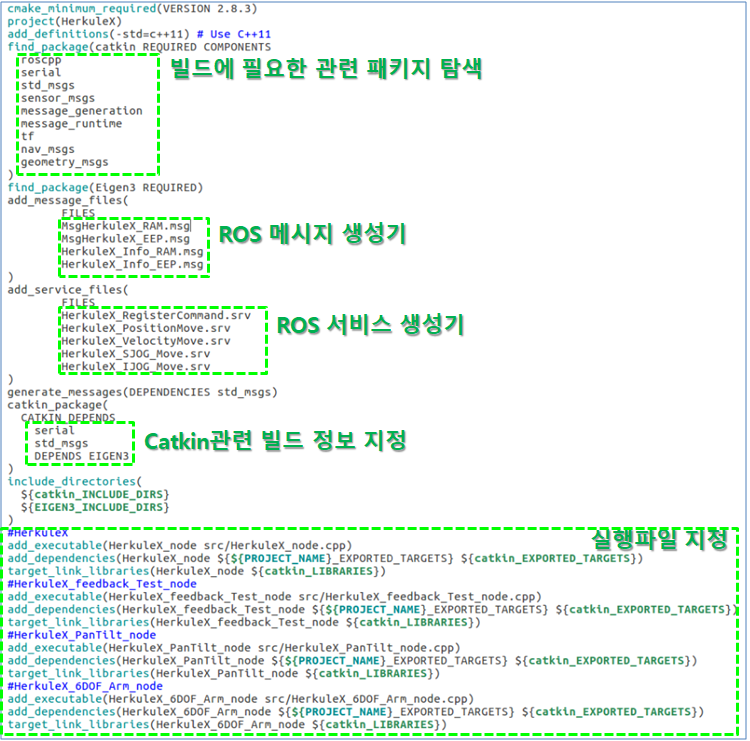


그림 11. CMakeLists.txt file의 구성

1. Package xml파일.

* 해당 파일에서는 빌드 시스템의 의존성과 패키지를 빌드 할 때 의존하는 패키지, 패키지를 실행할 때 의존하는 패키지 명이 기록되어 있으며, 그 내용은 아래 그림과 같다.

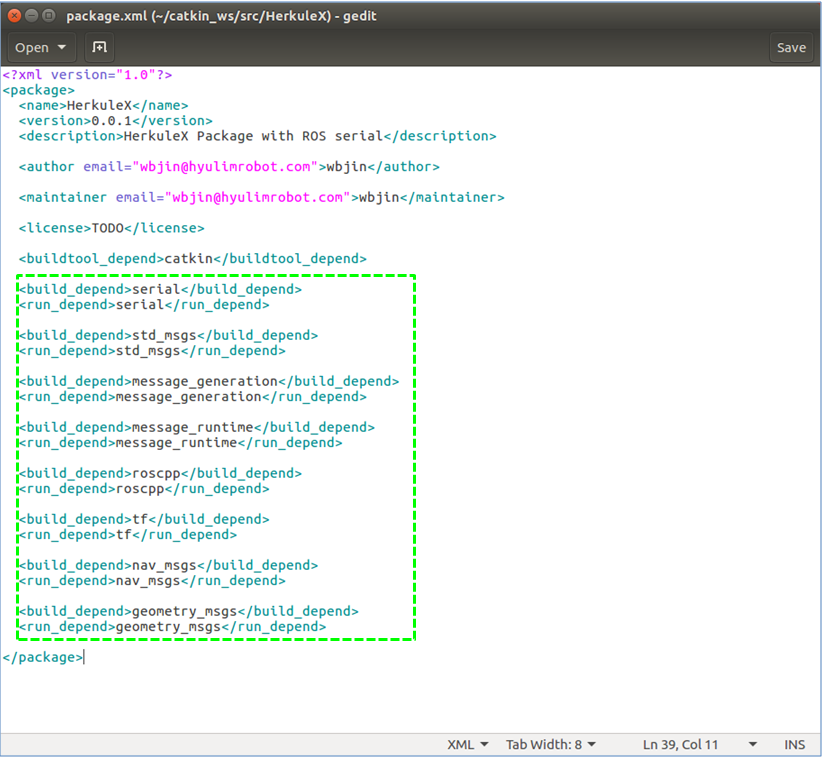


그림 12. Package.xml file의 내용.

1. **HerkuleX Package의 실행.**

ROS시스템에서는 실행하고자 하는 ROS Node를 구동하기 위해서는 초기에 ROS마스터를 실행해 주어야 각각의 Node사이에 연결과 메시지의 통신이 가능하다. HerkuleX Package에서는 이러한 번거로움을 해결하고자 Launch파일을 추가로 제작하였으며, 이 Launch만을 실행 함으로서 HerkuleX의 구동을 할 수 있도록 되어 있다. 해당 Launch file의 구성은 다음과 같다.

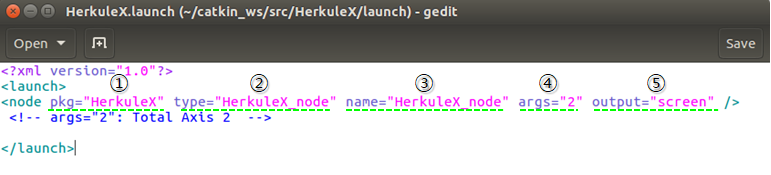


그림 13. HerkuleX Launch file의 구성

1. pkg: 패키지의 이름.
2. type: 실제 실행할 Node의 이름.
3. name: 위 type에 해당하는 Node가 실행될 때 붙여지는 이름.
4. 연결된 HerkuleX의 개수. (예시 그림에는 2개를 사용함으로 “2”를 기입)
5. output: 해당 터미널 창에 실행되는 Node들의 출력들 표시해주기 위한 옵션.

※ HerkuleX Node를 실행하기 위해서는 위의 그림과 같이 Launch file을 실행해도 되지만, roscore를 실행한 후에 HerkuleX\_Node를 실행하는 방법으로도 사용이 가능하다.

- HerkuleX를 실행하기 위한 Launch File의 실행은 터미널 창을 열어서 해당 Launch file을 호출하는 명령어를 입력하는 방식으로 진행하며, 그 방법은 다음과 같다.

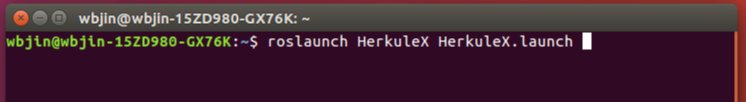


그림 14. HerkuleX Launch file의 실행을 위한 터미널 창 명령어 입력

* 명령어 입력: $ roslaunch HerkuleX HerkuleX.launch

- HerkuleX Launch file이 실행되면, 아래 그림과 같이 터미널 창에 HerkuleX의 동작을 위한 입력 창이 출력된다. 이때, HerkuleX와 PC간에 Serial통신이 정상적으로 수행되면 터미널창의 가장 하단에 ‘Serial Port initialized’라는 문구가 나오게 되는데 해당 문구가 출력되지 않으면, PC와 HerkuleX간에 연결이 정상적으로 되지 않는 것이다. (해당 경우에는 HerkuleX를 연결하는 USB to Serial의 연결 및 PC에서의 인식여부, HerkuleX에 공급되는 전원에 이상이 없는지 여부를 체크해야 한다.)

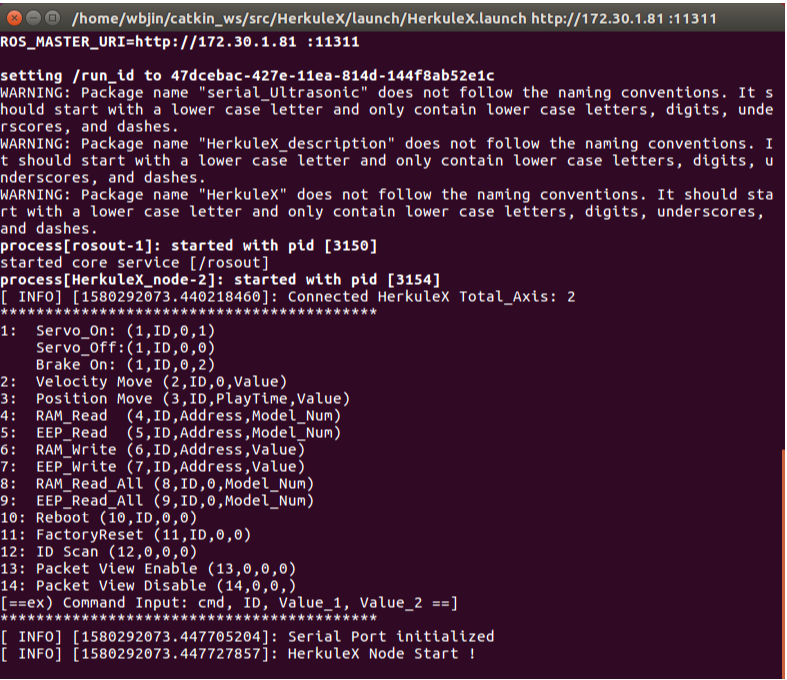


그림 15. HerkuleX Node실행 시 터미널 출력화면.

* HerkuleX Node에서는 총 14가지 타입의 입력으로 HerkuleX의 수동 조작이 가능하다.
* 입력인자는 총 4가지이며, 각각의 인자는 ‘,’(comma)로 구분한다. 첫 번째 인자로 명령어의 타입을 결정하며, 두 번째 인자로 HerkuleX의 ID값, 세 번째는 Register의 주소번지, 네 번째는 데이터 값이다. 각각의 명령 타입마다 이 4가지를 모두 사용하는 경우도 있지만, 그렇지 않은 경우도 있다. 명령어를 수행하는데 필요 없는 입력인자의 경우 ‘0’을 입력해 주면 된다. 이 14가지 타입의 입력에 대한 사용방법 및 설명은 다음과 같다. (모든 입력인자를 입력한 후에는 ‘Enter’키를 입력한다.)

1. : HerkuleX의 모터 토크를 On/Off하거나 브레이크 On시킬 수 있는 명령어이다.

예\_1) 1,1,0,1을 입력하는 경우 HerkuleX ID 1번에 모터토크를 On시킨다.

예\_2) 1,1,0,0을 입력하는 경우 HerkuleX ID 1번에 모터토크를 Off시킨다.

예\_3) 1,1,0,2을 입력하는 경우 HerkuleX ID 1번에 브레이크를 On시킨다.

1. : HerkuleX에 속도모드 지령을 전달하는 명령어이다.

예) 2,1,0,100을 입력하는 경우 HerkuleX ID1번을 목표속도 100으로 동작한다.

1. : HerkuleX에 위치모드 지령을 전달하는 명령어이다.

예) 3,1,100,512을 입력하는 경우 HerkuleX ID1번을 100ms시간 안에 목표위치 512로 동작한다.

1. : HerkuleX의 RAM Register 주소를 선택하여 해당 데이터를 확인하는 명령어이다.

예) 4,1,60,0을 입력하는 경우 HerkuleX ID1번의 절대위치(Absolute Position) 데이터를 확인한다.

. 해당 명령이 수행되면 선택된 주소의 RAM\_MAP topic의 값만 업데이트 된다.

1. : HerkuleX의 EEP Register 주소를 선택하여 해당 데이터를 확인하는 명령어이다.

예) 5,1,50,0을 입력하는 경우 HerkuleX ID1번의 위치에러 허용치(Inposition Margin) 데이터를 확인한다.

. 해당 명령이 수행되면 선택된 주소의 EEP\_MAP topic의 값만 업데이트 된다.

1. : HerkuleX의 RAM Register 주소를 선택하여 사용자가 원하는 데이터를 입력하는 명령어이다.

예) 6,1,53,1을 입력하는 경우 HerkuleX ID1번에LED Control값에 1을 입력하여 녹색 LED를 On.

1. : HerkuleX의 EEP Register 주소를 선택하여 사용자가 원하는 데이터를 입력하는 명령어이다.

예) 7,1,50,3을 입력하는 경우 HerkuleX ID1번에 위치에러 허용치(Inposition Margin) 데이터를 3으로 적용한다.

1. : HerkuleX의 모든 RAM Register 데이터를 읽어서 확인하는 명령어이다.

예) 8,1,0,0을 입력하는 경우 HerkuleX ID1번에 모든 RAM Register데이터를 읽어들인다.

. 해당 명령이 수행되면 RAM\_MAP topic의 모든 값이 업데이트 된다.

1. : HerkuleX의 모든 EEP Register 데이터를 읽어서 확인하는 명령어이다.

예) 9,1,0,0을 입력하는 경우 HerkuleX ID1번에 모든 EEP Register데이터를 읽어들인다.

. 해당 명령이 수행되면 EEP\_MAP topic의 모든 값이 업데이트 된다.

1. : HerkuleX의 재 부팅을 수행하는 명령어이다. (HW전원 Off/On와 동일한 기능)

예) 10,1,0,0을 입력하는 경우 HerkuleX ID1번에 전원 Off -> On이 수행된다.

[주의] RAM Register에 기록된 값은 모두 저장되지 않음. (휘발성 메모리 영역)

1. : HerkuleX의 RAM과 EEP Register의 값을 공장초기화 상태로 되돌리는 명령어

예) 11,1,0,0을 입력하는 경우 HerkuleX ID1번에 RAM&EEP Register데이터의 초기화 수행.

단, ID와 Board rate(bps)의 값은 변경되지 않고 유지됨.

1. : PC에 연결된 HerkuleX의 ID를 Scan하는 명령어로서 1초 간격으로 ID 1번에서 253번까지의 ID를 Scan한다.

예) 12,0,0,0을 입력하는 경우 HerkuleX의 ID는 모두 Scan한다.

1. : PC에서 HerkuleX로 송/수신하는 명령어들의 Packet정보를 터미널 창에 출력하는 기능을 활성화 한다.

예) 13,0,0,0을 입력하면 해당 기능이 활성화 된다.

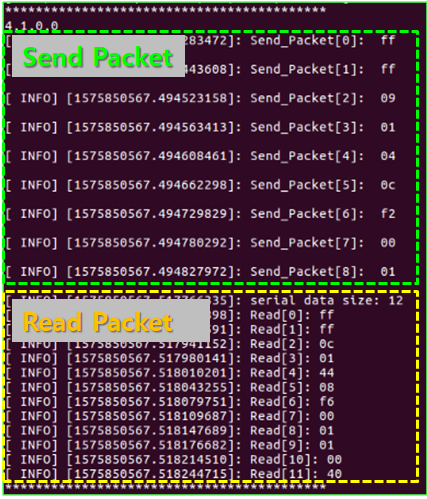


그림 16. Packet 송/수신 예시화면

. 해당 기능활성화 후에 4,1,0,0(HerkuleX ID1번에 RAM Register 0번 읽기)을 입력하는 경우, 위의 그림과 같이 송신된 Packet의 정보와 수신된 Packet의 정보가 터미널 창에 출력된다. 출력되는 Packet의 값은 16진수(Hex)의 값으로 출력된다.

1. : PC에서 HerkuleX로 전달하는 명령어들의 Packet정보를 터미널 창에 출력하는 기능을 비활성화 한다.

예) 14,0,0,0을 입력하면 해당 기능이 비활성화 된다.

**4. ROS GUI개발도구(rqt)를 활용한 HerkuleX의 제어.**

rqt는 ROS환경에서 사용 가능한 QT기반 프레임워크입니다. ROS사용자는 rqt\_service\_caller 이용하여, HerkuleX의 구동 및 RAM & EEP Register의 값을 손쉽게 확인할 수 있으며, rqt\_grapth를 이용하여, 실행되는 Node와 Node간의 토픽(Topic)정보를 한눈에 확인할 수 있습니다. 또한 rqt\_ploat을 활용하여, HerkuleX의 데이터를 그래프로 표시할 수도 있습니다. (단, 사용자PC의 ROS환경에는 rqt package가 설치되어 있어야 합니다.)

1. rqt\_service\_caller를 활용한 HerkuleX의 위치/속도 제어 및 Register 핸들링.

rqt\_service\_caller을 이용하기 위해서는 아래 그림과 같이 터미널 창에서 rqt\_service\_caller를 실행해 주어야 합니다.

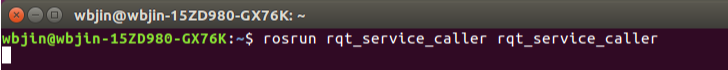


그림 17. rqt\_service\_caller의 실행.

* 명령어 입력: $ rosrun rqt\_service\_caller rqt\_service\_caller

rqt\_service\_caller가 실행되면 아래 그림과 같은 GUI환경이 출력되며, 해당 화면에서는 위치제어 명령, 속도제어 명령, 레지스터 제어 명령을 선택하여 각각의 서비스를 호출할 수가 있습니다.

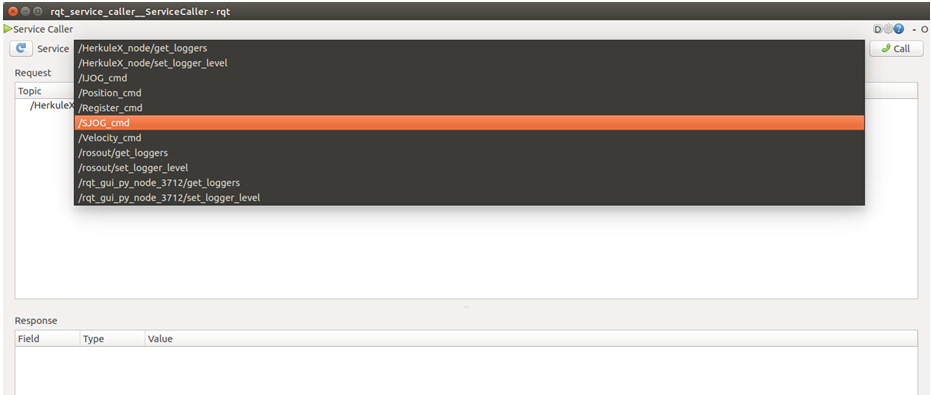


그림 18. Service Caller GUI화면.

* Position\_cmd 서비스 설명

해당 기능에서는 HerkuleX에서 제공하는 JOG(S\_JOG와 I\_JOG)명령어를 이용하여 HerkuleX를 사용자가 원하는 위치, 원하는 동작시간으로 동작시킬 수가 있고, LED의 색상을 선택하여 On/Off가 가능합니다. 해당 서비스의 Input인자로는 HerkuleX의 ID와 LED의 색상, 동작시간, 목표위치, JOG명령의 선택이 가능합니다.

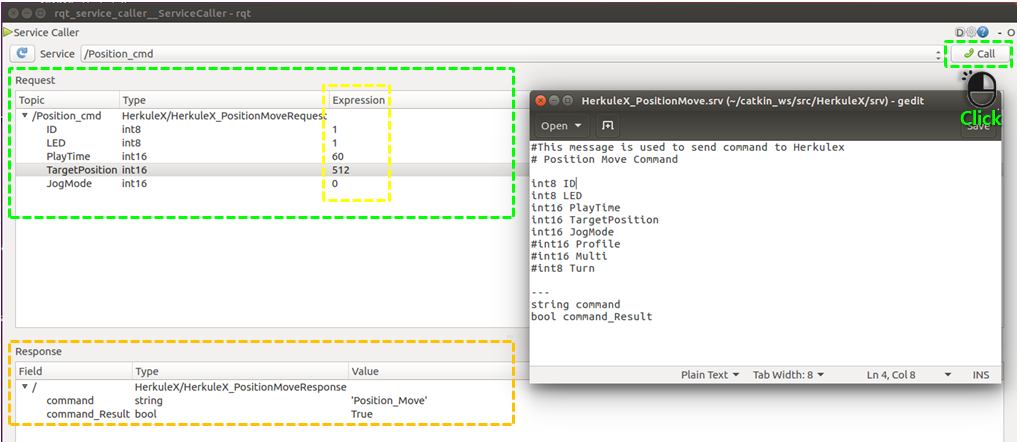


그림 19. Position\_cmd서비스 사용방법과 srv 파일 내용.

* 위의 그림과 같이 Input인자의 값을 넣고 ‘Call’버튼을 클릭하면 서비스가 호출되며, 정상적으로 서비스가 전달되면, 하단에 있는 응답 창으로 전달한 Command의 정보와 호출에 대한 리턴 값 정보가 업데이트 됩니다.
* 해당 예시 설명: ID 1번 HerkuleX에 녹색 LED를 On시키고, 672(11.2 \* 60)ms의 속도로 0도 위치(512)로 이동.
* Velocity\_cmd 서비스 설명.

해당 기능에서는 HerkuleX에서 제공하는 JOG(S\_JOG와 I\_JOG)명령어를 이용하여 HerkuleX를 사용자가 원하는 속도로 동작시킬 수가 있고, LED의 색상을 선택하여 On/Off가 가능합니다.

해당 서비스의 Input인자로는 HerkuleX의 ID와 LED의 색상, 목표속도, JOG명령의 선택이 가능합니다.

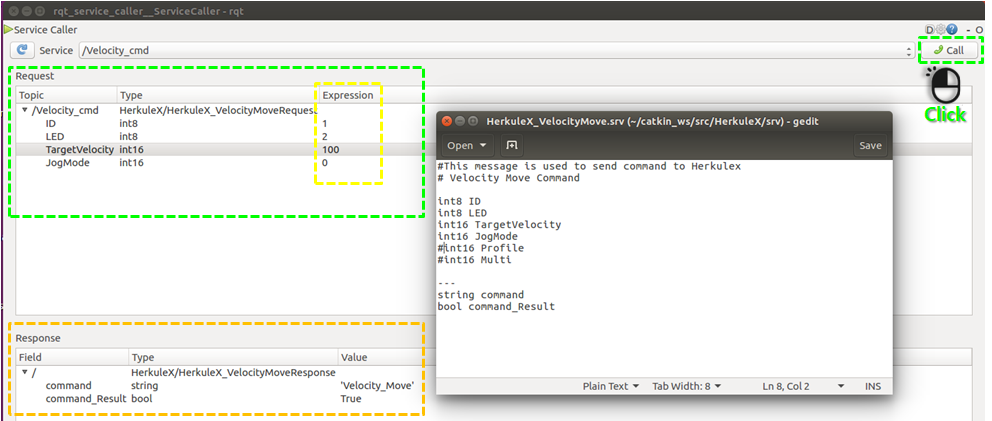


그림 20. Velocity\_cmd 서비스 사용방법과 srv 파일 내용

* 위의 그림과 같이 Input인자의 값을 넣고 ‘Call’버튼을 클릭하면 서비스가 호출되며, 정상적으로 서비스가 전달되면, 하단에 있는 응답 창으로 전달한 Command의 정보와 호출에 대한 리턴 값 정보가 업데이트 됩니다.
* 해당 예시 설명: ID 1번 HerkuleX에 청색 LED를 On시키고, 100(범위: 0~1024)의 속도로 무한회전.
* Register\_cmd 서비스 설명.

해당 기능에서는 HerkuleX의 RAM과 EEP Register의 값을 사용자가 직접 Read/Write가 가능합니다. 해당 서비스의 Input인자로는 서비스 명령어 문자열(String), HerkuleX ID, Register주소번지, 데이터의 입력으로 호출이 가능합니다.

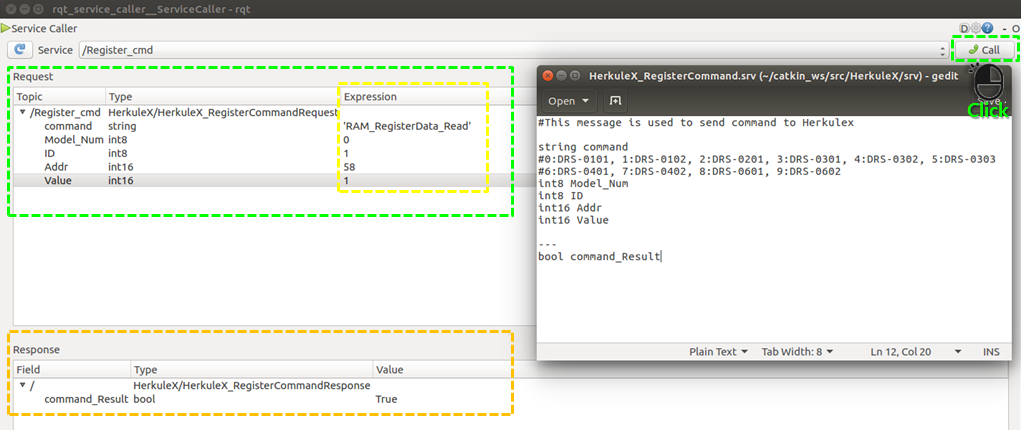


그림 21. Register\_cmd 서비스 사용방법과 srv 파일 내용.

* 위의 그림과 같이 Command의 문자열에 ‘RAM\_RegisterData\_Read’를 입력하고 HerkuleX ID1번에 Register주소 0번(ID), 데이터 0을 입력하면, HerkuleX ID1번의 ID값이 RAM\_MAP topic에 업데이트되며, 하단에 있는 응답 창으로 전달한 Command의 호출에 대한 리턴 값 정보가 업데이트 됩니다.
* 터미널 창에서 rostopic 의 정보를 확인하면, ID에 대한 메시지가 ‘1’로 업데이트 되어있음을 확인할 수 있습니다.
* 해당 서비스의 입력 가능한 명령어 문자열은 총 10가지가 정의되어 있으며, 그 내용은 다음과 같습니다.

1. **RAM\_RegisterData\_Read\_All**: RAM Register의 모든 값을 읽어 들이는 명령이며, Input인자로는 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력.
2. **EEP\_RegisterData\_Read\_All**: EEP Register의 모든 값을 읽어 들이는 명령이며, Input인자로는 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력
3. **RAM\_RegisterData\_Read**: RAM Register의 특정 주소의 값을 읽어 들이는 명령이며, Input인자로 HerkuleX의 ID와 RAM주소번지를 선택하고, Data는 0을 입력.
4. **EEP\_RegisterData\_Read**: EEP Register의 특정 주소의 값을 읽어 들이는 명령이며, Input인자로 HerkuleX의 ID와 EEP주소번지를 선택하고, Data는 0을 입력.
5. **RAM\_RegisterData\_Write**: RAM Register의 특정 주소에 데이터를 쓰는 명령이며, Input인자로 HerkuleX의 ID와 RAM주소번지, 원하는 데이터를 입력. (단, 주소번지 별 데이터의 크기와 범위는 HerkuleX 데이터시트를 참고.)
6. **EEP\_RegisterData\_Write**: EEP Register의 특정 주소에 데이터를 쓰는 명령이며, Input인자로 HerkuleX의 ID와 EEP주소번지, 원하는 데이터를 입력. (단, 주소번지 별 데이터의 크기와 범위는 HerkuleX 데이터시트를 참고.)
7. **SERVO\_ON**: HerkuleX의 토크를 On시키는 명령이며, Input인자로 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력.
8. **SERVO\_OFF**: HerkuleX의 토크를 Off시키는 명령이며, Input인자로 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력.
9. **BRAKE\_ON**: HerkuleX의 브레이크를 On시키는 명령이며, Input인자로 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력.
10. **ERROR\_CLEAR**: HerkuleX에서 발생된 모든 Error를 초기화 시키는 명령이며, Input인자로 HerkuleX의 ID값만 선택하고 나머지 인자(Addr, Value)는 모두 0을 입력.

* IJOG\_cmd 서비스 설명.

해당 기능에서는 HerkuleX의 IJOG명령어를 이용하여, 여러 개의 HerkuleX에 속도 지정이 개별로 가능하며, 동시에 위치제어를 할 수 있습니다. (최대 43개의 제어가능) 해당 서비스의 Input인자는 연결된 HerkuleX의 ID가 정의된 배열, LED에 색상에 대한 배열, 목표위치에 대한 배열, 동작시간에 대한 배열, 연결된 HerkuleX의 총 개수에 대한 데이터가 있다.

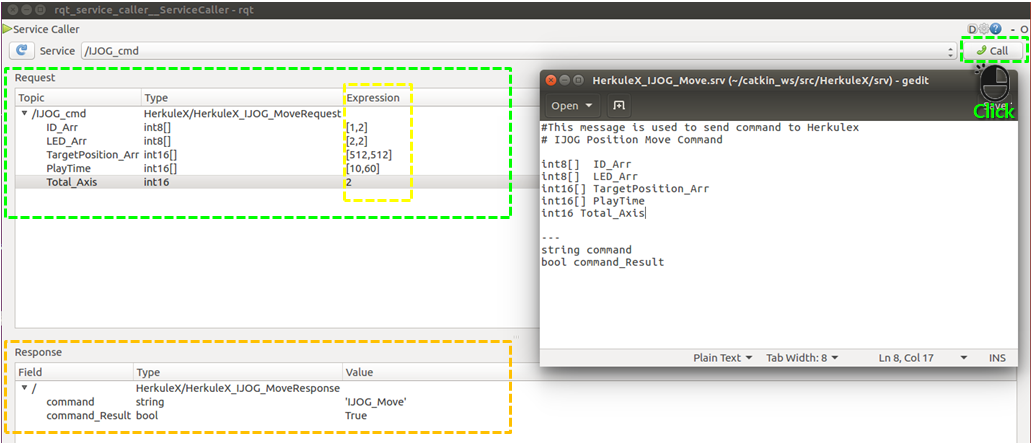


그림 22. IJOG\_cmd 서비스 사용방법과 srv파일 내용.

* 위의 그림과 같이 연결된 각각의 HerkuleX ID와 LED색상, 목표위치, 동작 시간을 배열로 입력하고, 연결된 HerkuleX의 총 개수를 입력한 후 서비스를 호출하면, 호출된 Command의 문자열과 호출의 결과가 리턴 된다.
* SJOG\_cmd 서비스 설명.

해당 기능에서는 HerkuleX의 SJOG명령어를 이용하여, 여러 개의 HerkuleX를 동시에 위치제어를 할 수 있습니다. (최대 53개의 제어가능)해당 서비스의 Input인자는 연결된 HerkuleX의 ID가 정의된 배열, LED에 색상에 대한 배열, 목표위치에 대한 배열, 동작시간, 연결된 HerkuleX의 총 개수에 대한 데이터가 있다.

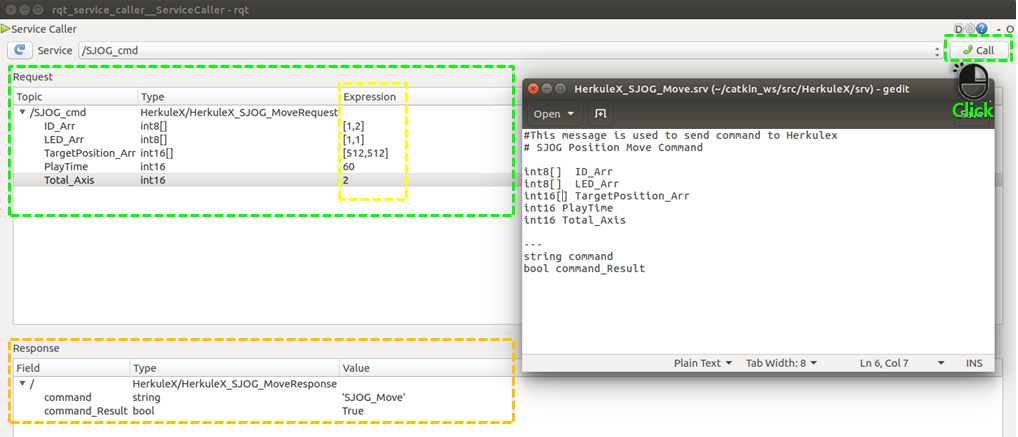


그림 23. SJOG\_cmd 서비스 사용방법과 srv파일 내용.

* 위의 그림과 같이 연결된 각각의 HerkuleX ID와 LED색상, 목표위치를 배열로 입력하고, 연결된 HerkuleX의 동작시간과 연결된 HerkuleX의 총 개수를 입력한 후 서비스를 호출하면, 호출된 Command의 문자열과 호출의 결과가 리턴 된다.

1. RAM\_MAP topic과 EEP\_MAP topic 내용을 한번에 확인하는 방법.

HerkuleX의 RAM과 EEP Register Map의 내용은 터미널 창에서 rostopic echo명령을 이용하여 확인할 수 있습니다.

1. RAM MAP의 정보를 확인하는 방법.
   * 명령어 입력: $ rostopic echo /Info\_RAM\_ID\_1
2. EEP MAP의 정보를 확인하는 방법.
   * 명령어 입력: $ rostopic echo /Info\_EEP\_ID\_1

위의 예시 명령어는 HerkuleX ID 1번에 대해서 정보를 확인하는 방법이며, ID가 다를 경우 명령어 끝에 있는 숫자를 해당 ID의 숫자로 수정하여 호출하시면 됩니다.

해당 명령어가 수행되면, 터미널 창에 출력되는 내용은 아래 그림과 같습니다.

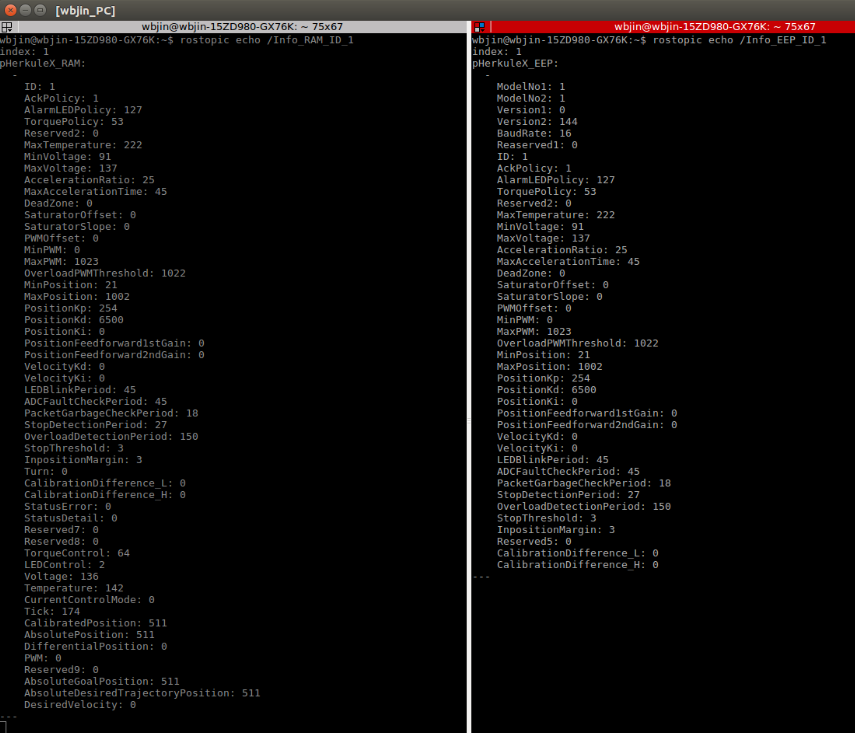


그림 24. rostopic echo명령을 이용한 RAM과 EEP의 정보확인 화면

**5. HerkuleX ROS Package 활용 예시**

HerkuleX ROS패키지에는 2개의 HerkuleX를 이용한 Pan/Tilt의 예제와 6개의 HerkuleX를 이용한 6DOF\_Arm의 예제가 포함되어 있습니다. HerkuleX\_description 패키지에는 HerkuleX의 외형과 장착되는 부품들의 STL파일, 해당 부품들을 이용한 Pan/Tilt와 6DOF Arm의 URDF파일이 포함되어 있습니다. 해당 URDF를 이용하여 RVIZ화면에 HerkuleX를 출력해주기 위한 Launch파일도 같이 포함되어 있습니다.

* **해당 패키지들을 사용하기 위해서는 패키지들에서 요청하는 의존성 패키지들을 모두 설치해야 합니다.**

1. HerkuleX 2개와 Webcam을 이용한 Pan/Tilt 예제

* HerkuleX PanTilt\_node.cpp에 2개의 HerkuleX를 이용한 Pan/Tilt의 기본적인 동작을 위한 코드가 구현되어 있습니다.
* ‘HerkuleX\_Pantilt\_facetracking.launch’는 공개 패키지인 ‘usb\_cam’과 ‘simple\_face\_detection’를 활용하였으며, 해당 내용은 launch파일에 정의되어 있습니다.

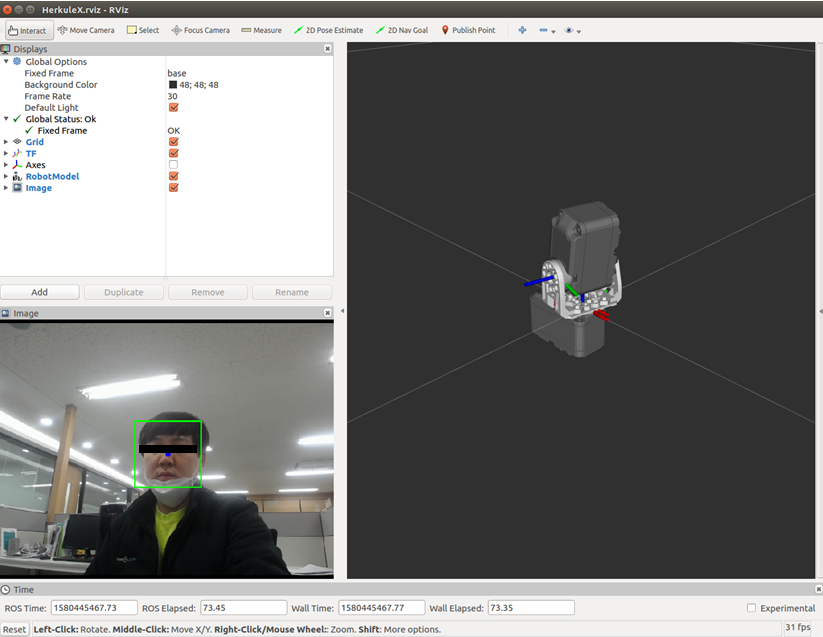


그림25. HerkuleX\_Pantilt\_facetracking 실행화면.

* ‘HerkuleX\_pantilt\_Manual\_control.launch’는 joint\_state\_publisher GUI를 이용하여, Pan과 Tilt를 수동으로 조작할 수 있는 예제입니다.

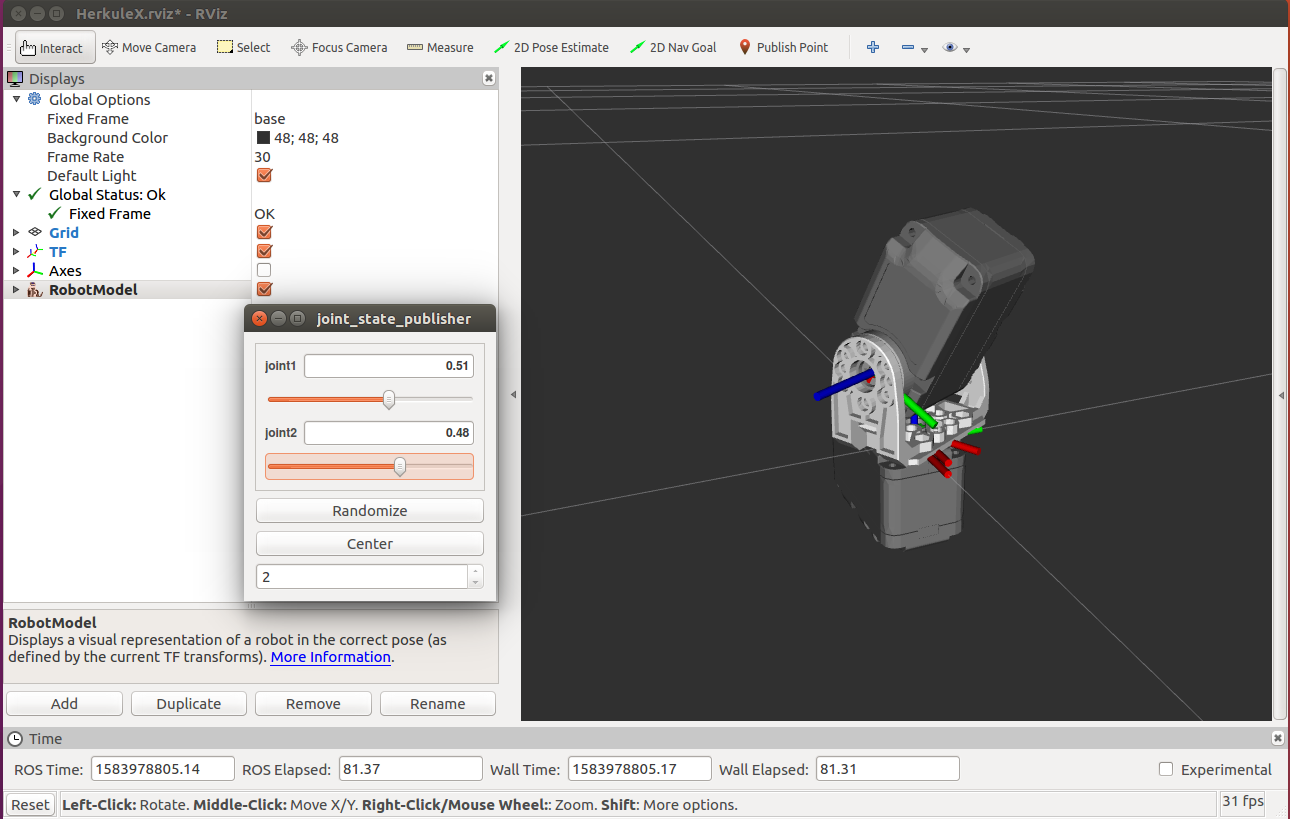


그림 26. HerkuleX\_pantilt\_Manual\_control.launch 실행화면.

1. HerkuleX 6개(그리퍼 제외)와 HerkuleX전용 브라켓을 이용한 6DOF Arm 예제.

* 해당 예제는 ROS에서 제공하는 Moveit을 활용한 예제로서 HerkuleX\_6DOF\_Arm\_node.cpp에 6개의 HerkuleX를 동작하기 위한 코드가 구현되어 있으며, Moveit Setup Assistant를 이용하여 생성한 moveit\_HerkuleX\_Arm패키지를 이용하여 동작을 할 수 있습니다
* **Moveit Setup Assistant에 대해서는 아래 링크를 참고해 주세요.**

**🡪** <http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html>

* 6DOF Arm의 동작을 위해서는 아래그림과 같이 2개의 launch파일을 실행해 주어야 합니다.

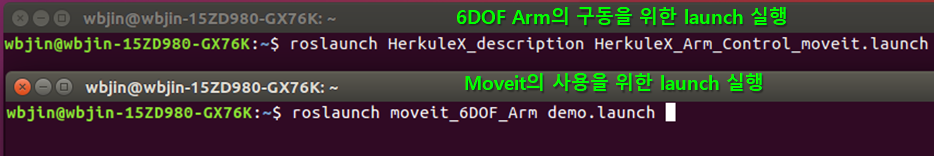


그림 27. 6DOF Arm의 동작을 위한 launch실행.

* 위의 그림과 같이 2개의 launch를 실행하면, 아래그림과 같이 RVIZ화면에 6DOF Arm의 화면이 출력됩니다.

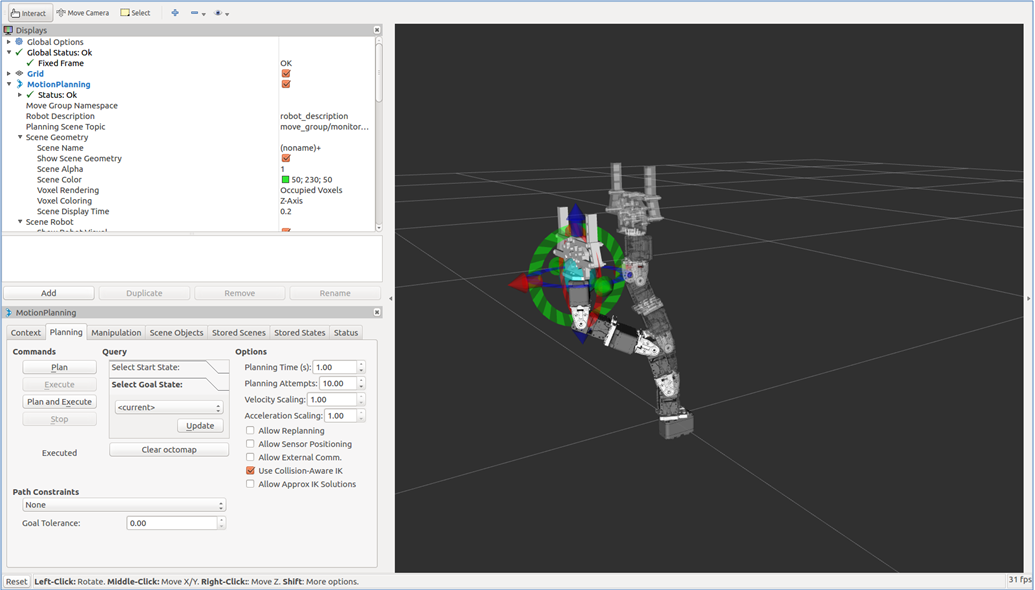


그림 28. Moveit을 이용한 6DOF Arm제어용 RVIZ화면.

* Arm의 끝 단에 위치한 작용점 부분을 원하는 위치로 이동 시킨 후 Planning 탭에 있는 Plan and Execute버튼을 클릭하면, Arm이 움직이는 경로에 대한 시뮬레이션과 함께, 실제 Arm이 동작합니다.
* Topic의 구성은 아래그림과 같습니다.

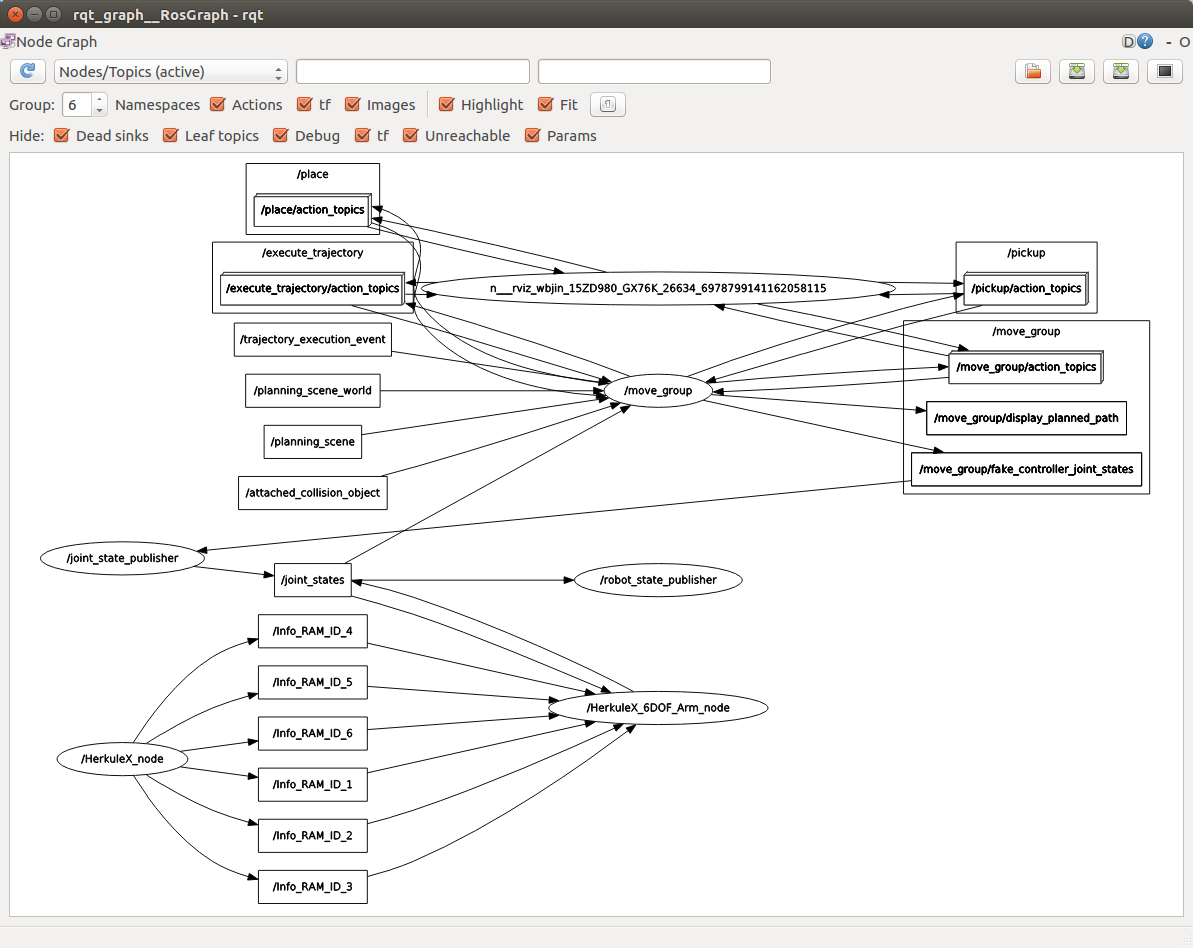


그림 29. rqt\_grah로 확인한rostopic의 구성.