

Gazebo_UR 조작하기

목차

- Moveit
- 새로운 Pose 정하기
- Joint 값 받기, Joint 움직이기
- 끝 부분 위치 받기, 움직이기

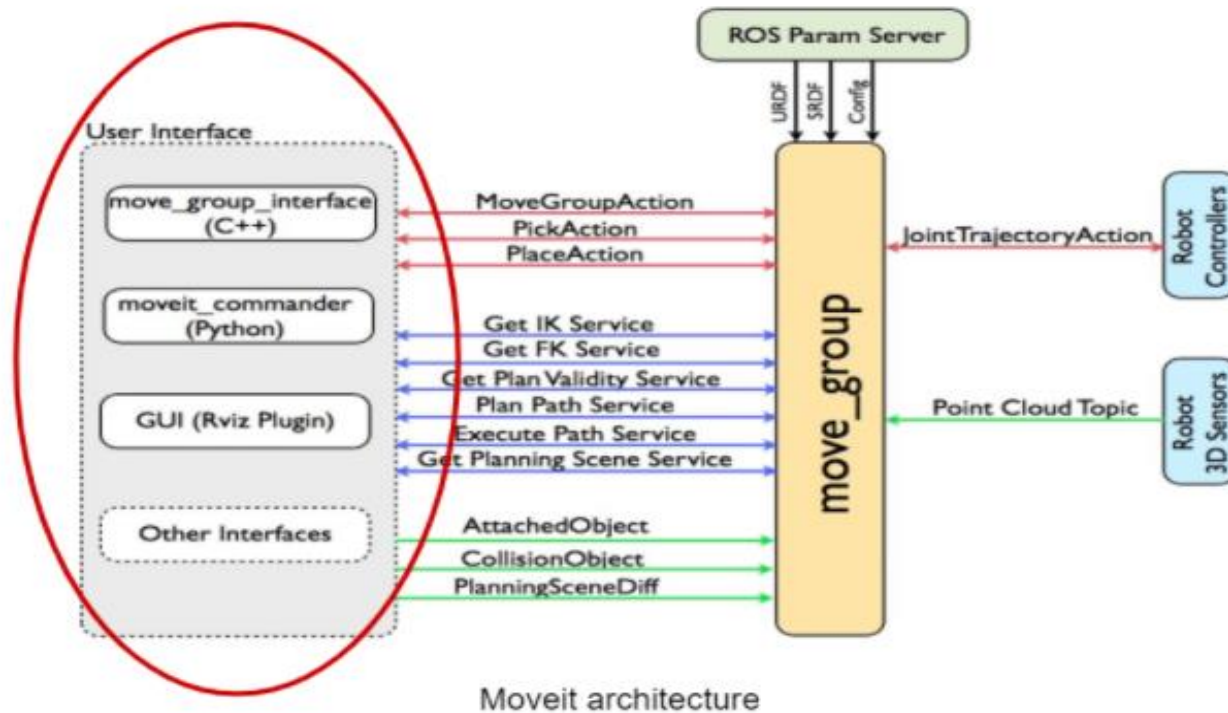
Moveit

- Moveit이란



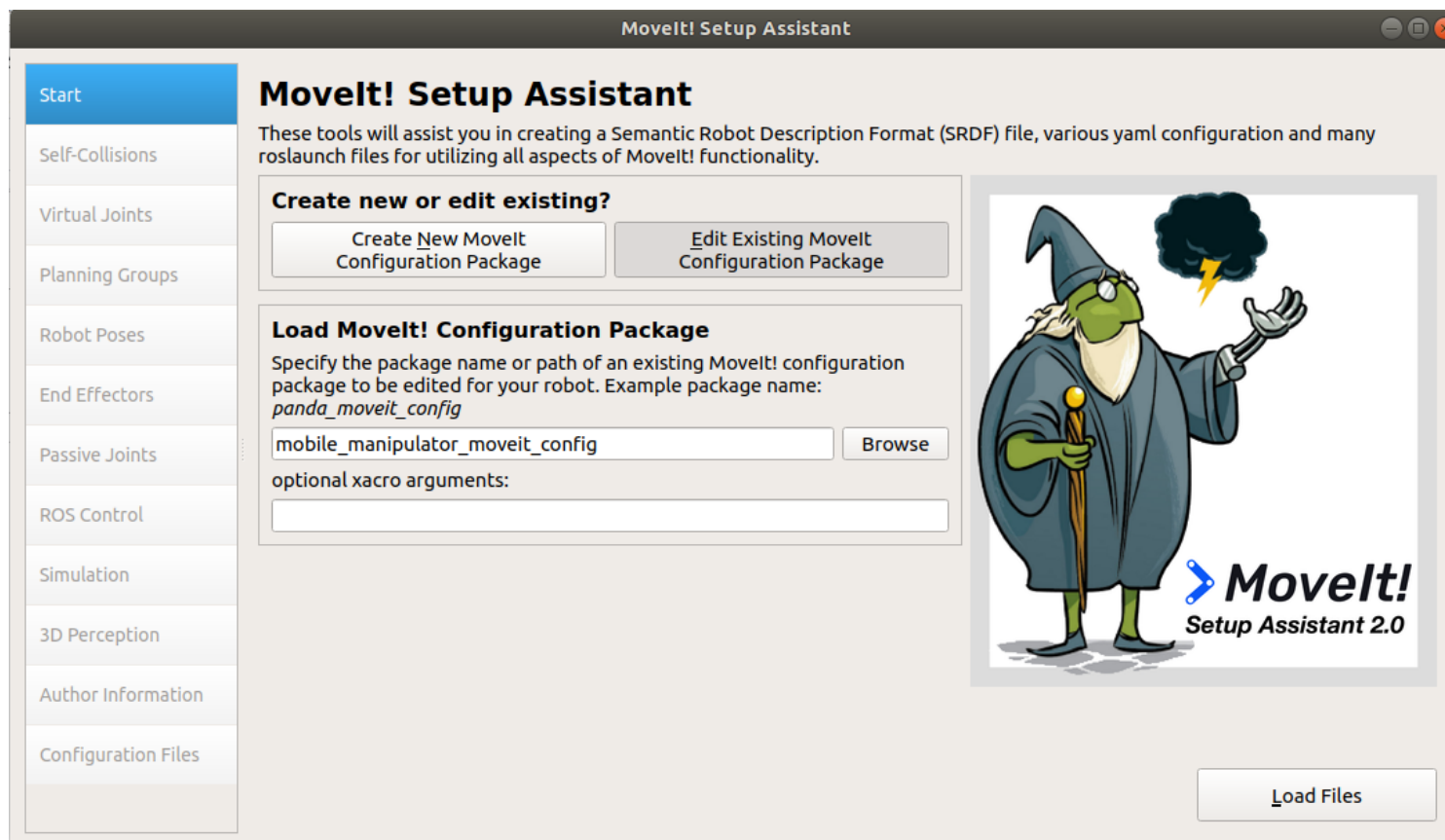
- 로봇 팔을 쉽게 제어하기 위해 ROS에서 만들어준 Package
- 126개의 로봇 팔들을 ROS에서 제어하기 위해서는 다음 Package를 사용함
- 다양한 Interface와 C++/Python 인터페이스를 제공하여 동작할 수 있음

- Moveit의 아키텍처는 다음과 같으며 이번 강의에서는 C++ 코드를 작성하여 Gazebo 상에서 UR을 제어하는 방법에 대해서 알아보겠습니다.

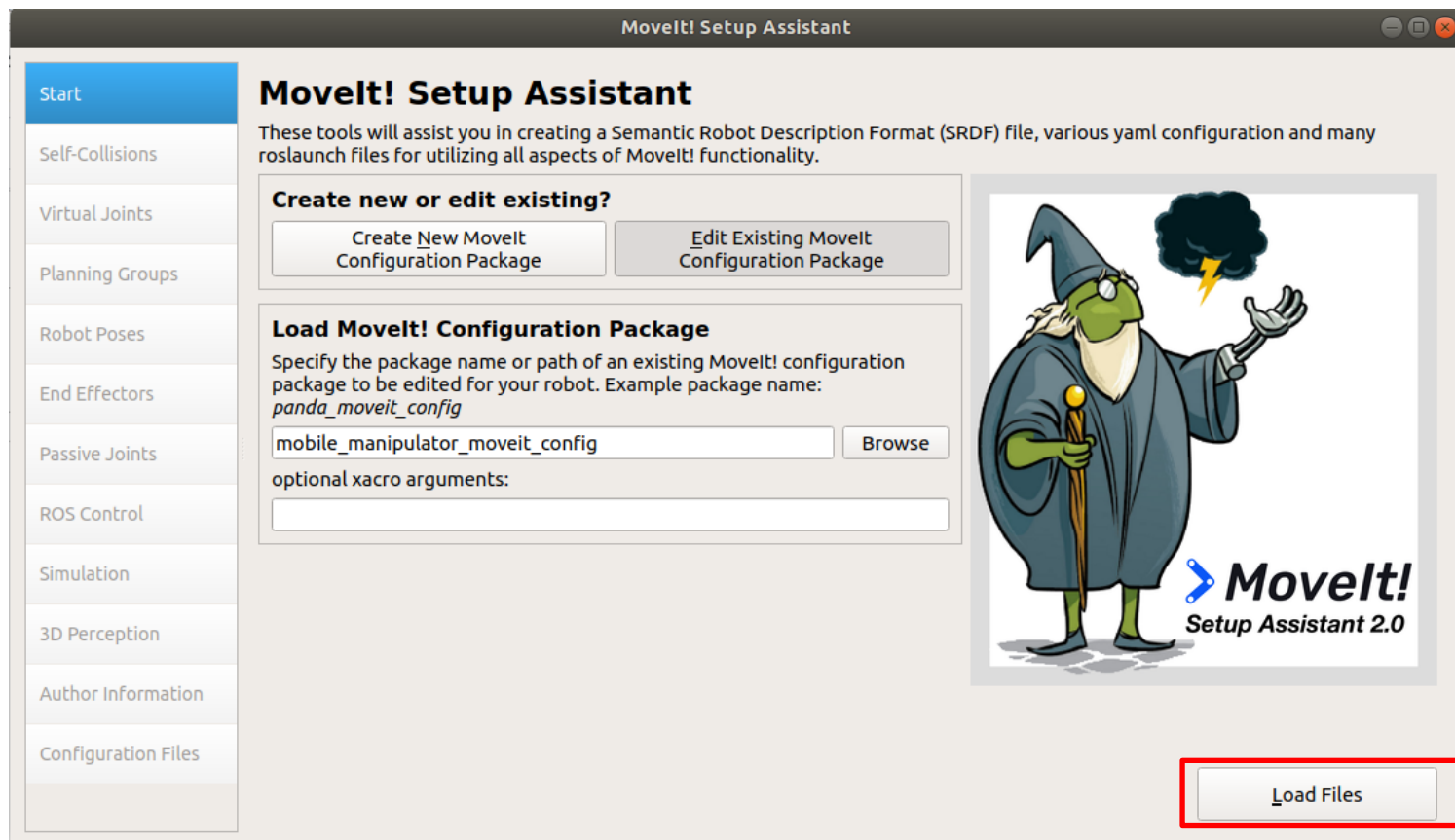


새로운 Pose 정하기

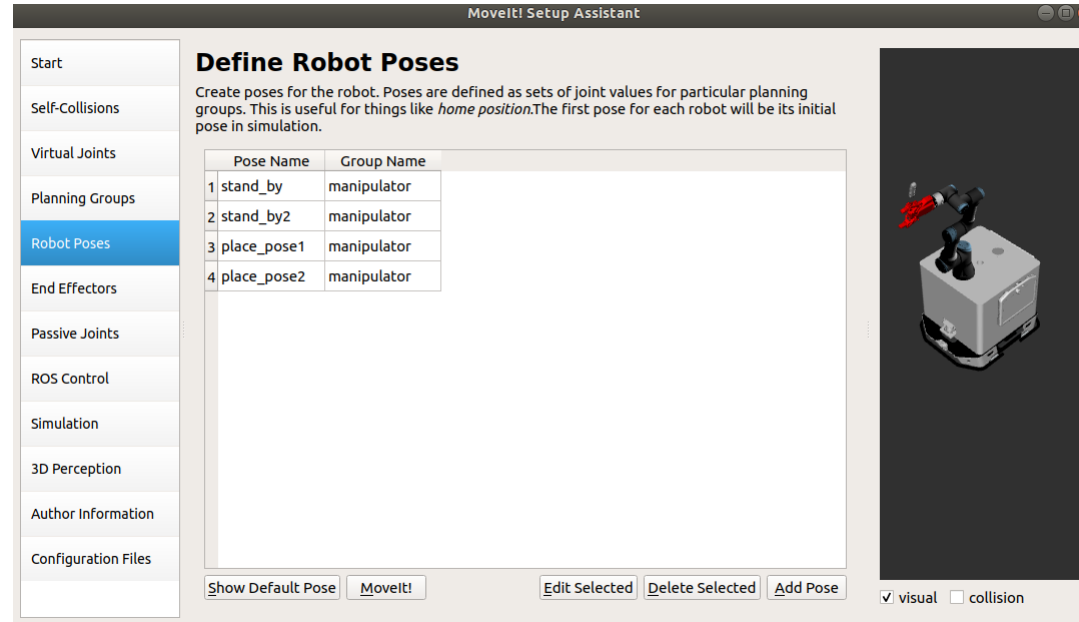
- 먼저 Moveit Setup Assistant에서 moveit 설정을 정합니다.
- `$ roslaunch mobile_manipulator_moveit_config setup_assistant.launch`



- 먼저 Moveit Setup Assistant에서 moveit 설정을 정합니다.
- Load Files를 눌러줍니다.

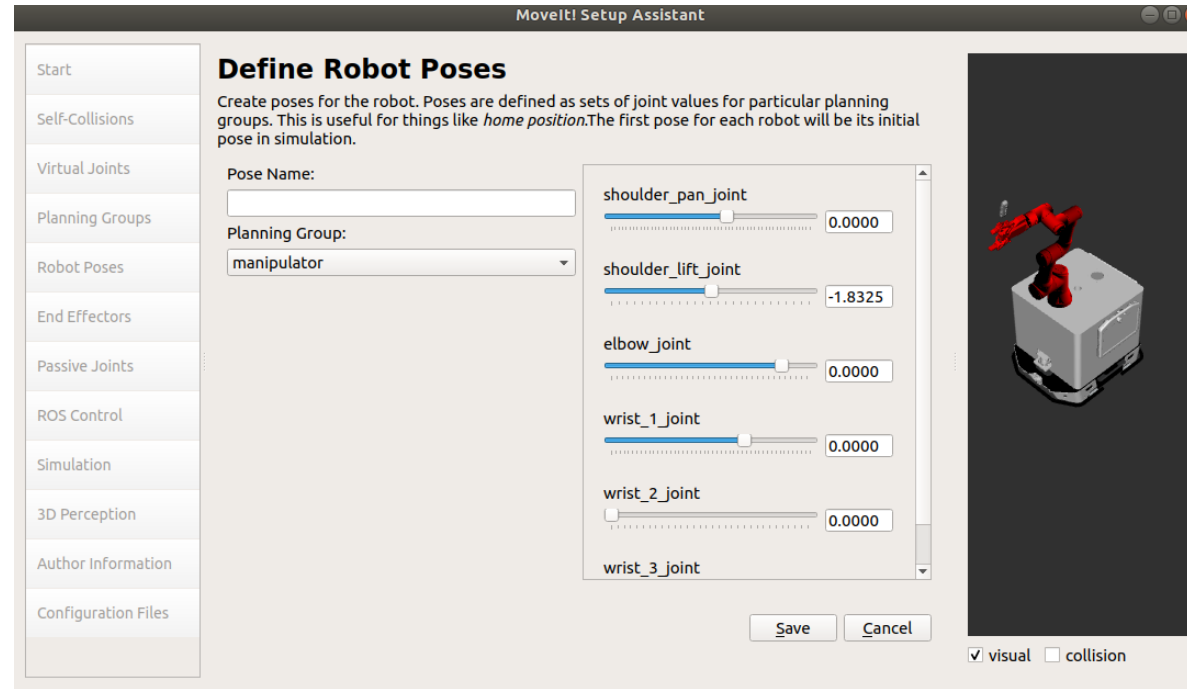


- Robot Pose 탭



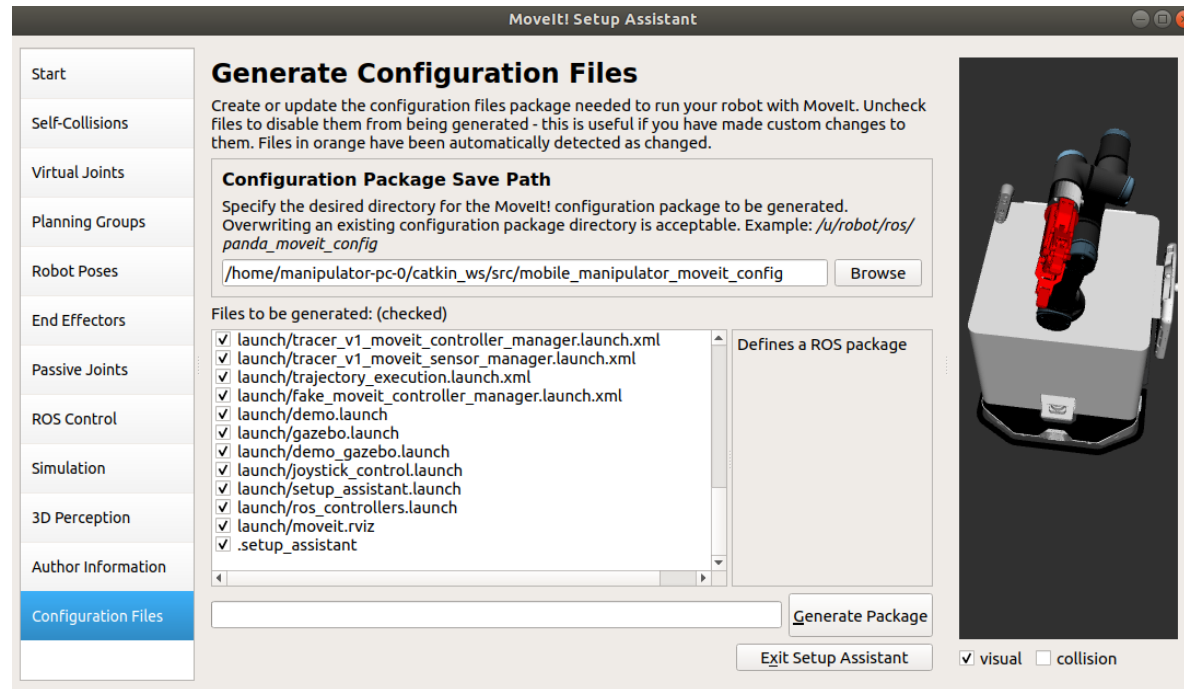
이 탭에서는 로봇의 특정한 position을 설정 할 수 있습니다. 사실 다른 탭들은 전부 저희가 설정을 해 놔기 때문에, 굳이 이를 설정할 필요가 없지만 이 탭은 새로운 pose를 설정 할 수 있기 때문에 유용하게 이용하실 수 있습니다. Add_Pose를 눌러 보 주시기 바랍니다.

- Robot Pose 탭



Add_pose를 누르면 다음과 같이 Group을 설정 한 후, 해당 Group에 대해서 새로운 Position을 만들 수 있습니다. Save를 누르면 새로 만들어 집니다.

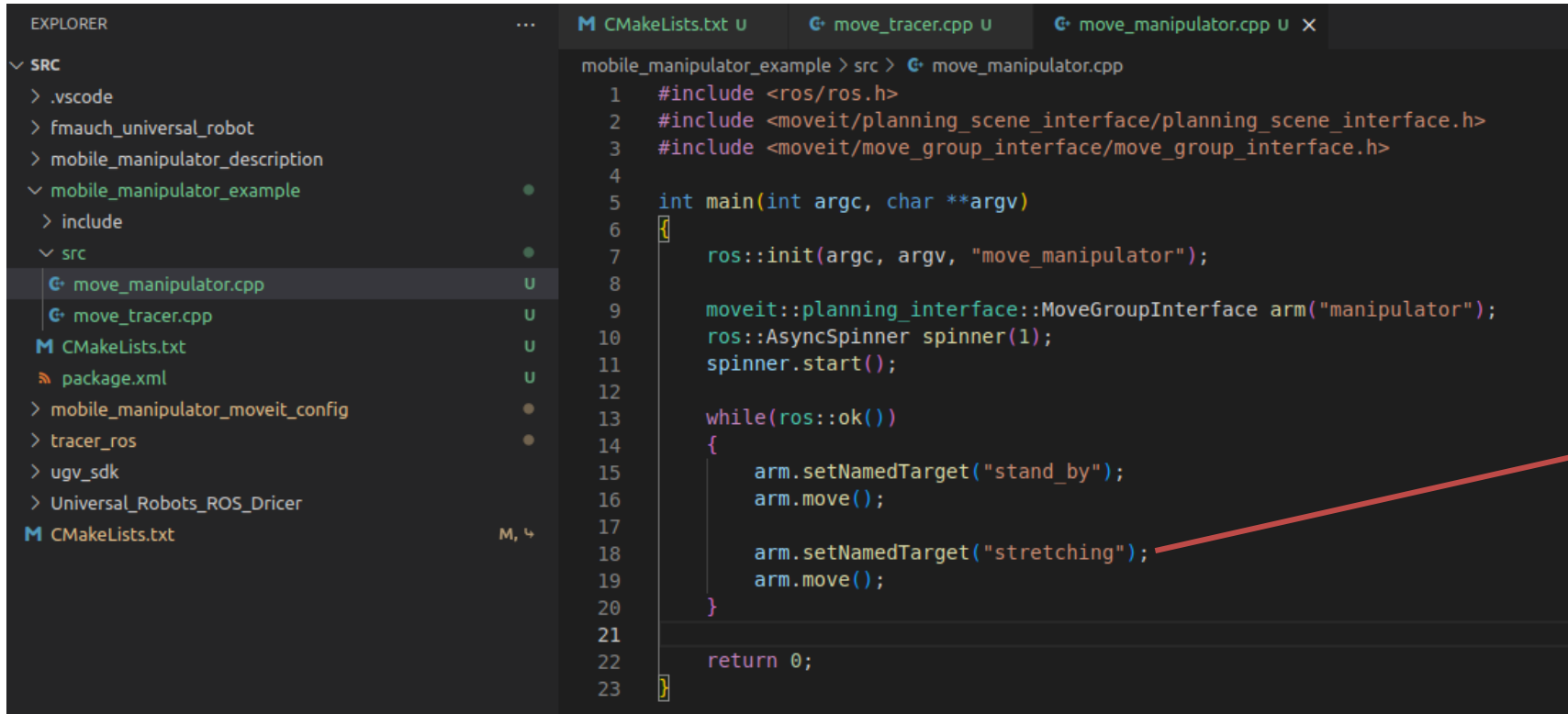
- Configuration Files탭



수정된 부분을 저장하는 탭입니다. 빨간색 부분과 검정색 부분만 체크하시고 Generate Package를 눌러주시기 바라겠습니다. 이렇게 하지 않으면 Moveit Package가 작동을 안 할 수 있습니다.

- 앞장에서 교시에서 실행해본 코드에서 새로 만든 Position으로 움직이도록 코드를 작성해 보겠습니다.

~/catkin_ws/src/mobile_manipulator_example/src/move_manipulator.cpp



```
mobile_manipulator_example > src > move_manipulator.cpp
1  #include <ros/ros.h>
2  #include <moveit/planning_scene_interface/planning_scene_interface.h>
3  #include <moveit/move_group_interface/move_group_interface.h>
4
5  int main(int argc, char **argv)
6  {
7      ros::init(argc, argv, "move_manipulator");
8
9      moveit::planning_interface::MoveGroupInterface arm("manipulator");
10     ros::AsyncSpinner spinner(1);
11     spinner.start();
12
13     while(ros::ok())
14     {
15         arm.setNamedTarget("stand_by");
16         arm.move();
17
18         arm.setNamedTarget("stretching");
19         arm.move();
20     }
21
22     return 0;
23 }
```

- 18줄 stretching 대신 새로 만든 Pose를 넣어 주시면 되겠습니다.

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

//변환 코드 빌드

```
$ roslaunch tracer_gazebo_sim tracer_empty_world.launch
```

//Gazebo simulation 실행

```
$ roslaunch mobile_manipulator_moveit_config move_group.launch
```

// moveit 실행

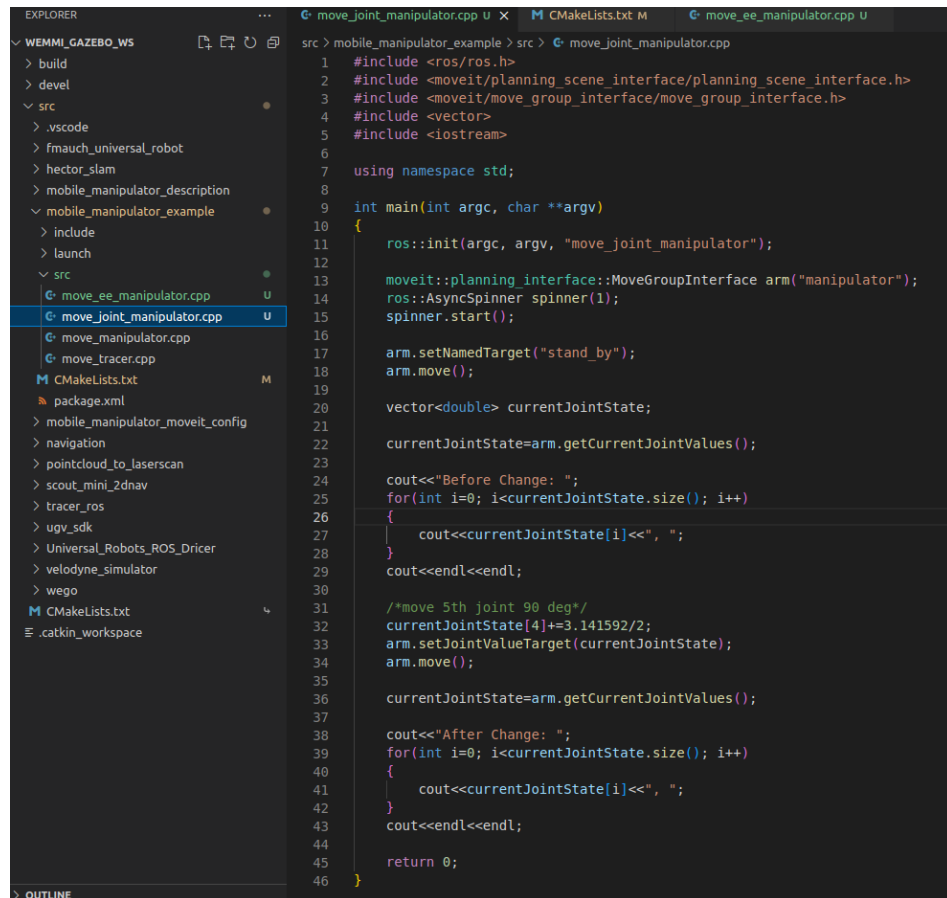
```
$ rosrn mobile_manipulator_example move_manipulator
```

// 작성된 코드 실행

**Joint 값 받기,
Joint 움직이기**

- 이번에는 현재 Joint 값을 받고, 하나의 Joint 값을 변화시켜 로봇을 움직이는 코드를 작성해 보겠습니다.

~/catkin_ws/src/mobile_manipulator_example/src/move_joint_manipulator.cpp



- 17~18줄: Stand_by pose로 움직이기
- 20~29줄: 현재 조인트들의 각도 값 표시하기
- 31~34줄: 5번째 Joint 90도 돌리기
currentJointState[0] → 1번째 Joint
currentJointState[1] → 2번째 Joint
currentJointState[2] → 3번째 Joint
currentJointState[3] → 4번째 Joint
currentJointState[4] → 5번째 Joint
currentJointState[5] → 6번째 Joint
(이 부분을 조작해서 여러가지 Joint들을 바꿔 봅니다.)
- 36~43줄: 현재 조인트들의 각도 값 표시하기

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

//변환 코드 빌드

```
$ roslaunch tracer_gazebo_sim tracer_empty_world.launch
```

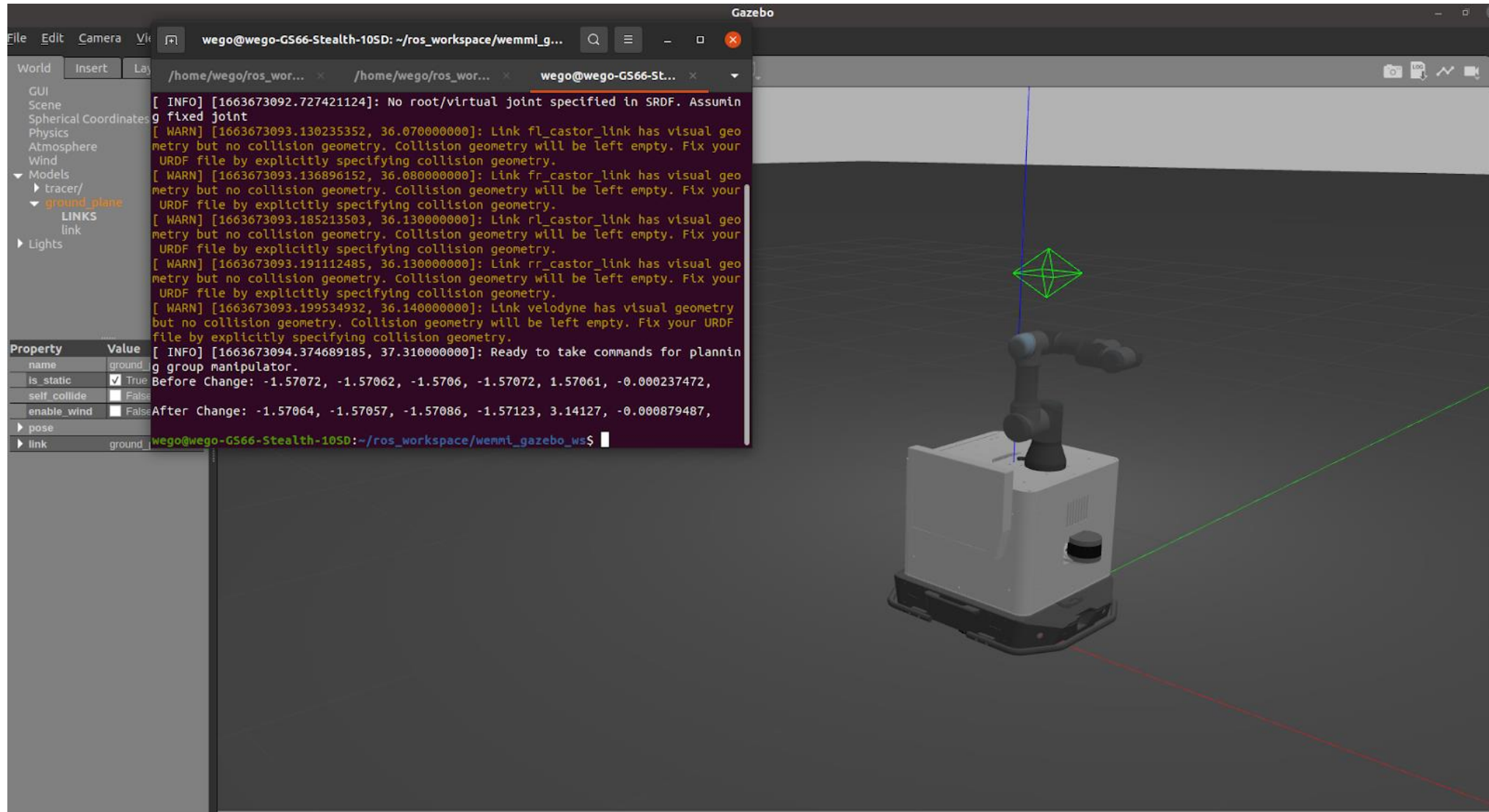
//Gazebo simulation 실행

```
$ roslaunch mobile_manipulator_moveit_config move_group.launch
```

// moveit 실행

```
$ rosrun mobile_manipulator_example move_joint_manipulator
```

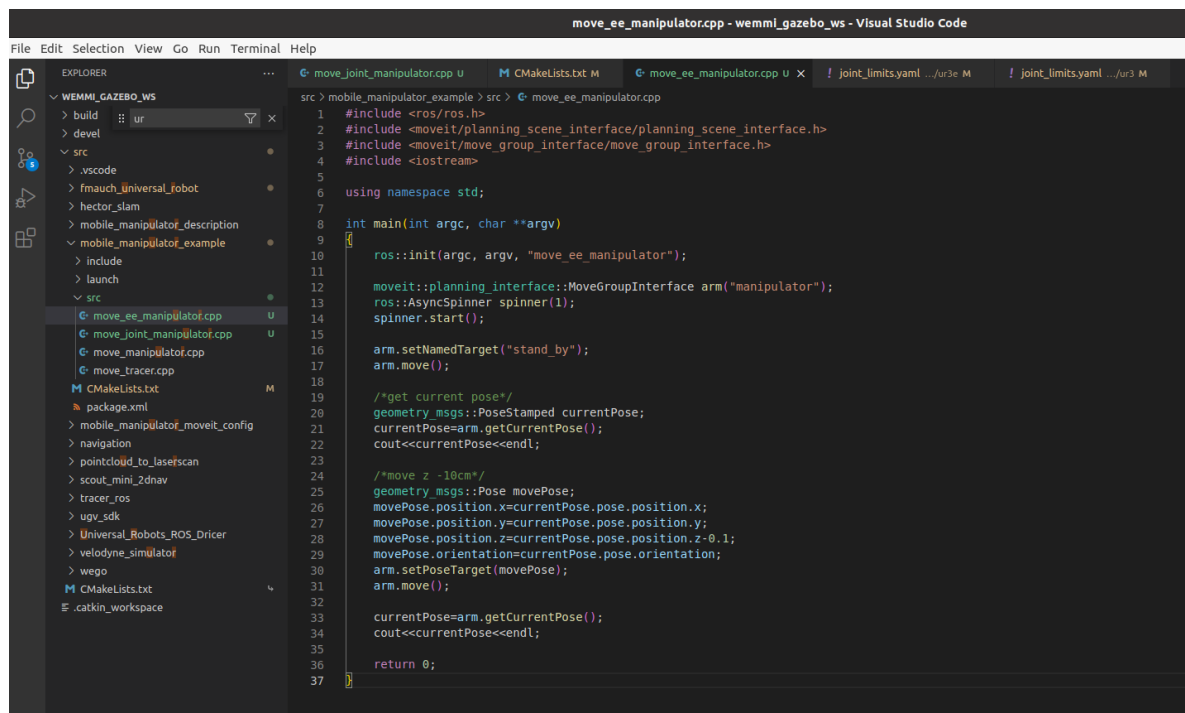
// 작성된 코드 실행



**끝 부분 값 받기,
움직이기**

- 이번에는 현재 Joint 값을 받고, 하나의 Joint 값을 변화시켜 로봇을 움직이는 코드를 작성해 보겠습니다.

~/catkin_ws/src/mobile_manipulator_example/src/move_ee_manipulator.cpp



```

1 #include <ros/ros.h>
2 #include <moveit/planning_scene_interface/planning_scene_interface.h>
3 #include <moveit/move_group_interface/move_group_interface.h>
4 #include <iostream>
5
6 using namespace std;
7
8 int main(int argc, char **argv)
9 {
10     ros::init(argc, argv, "move_ee_manipulator");
11
12     moveit::planning_interface::MoveGroupInterface arm("manipulator");
13     ros::AsyncSpinner spinner(1);
14     spinner.start();
15
16     arm.setNamedTarget("stand_by");
17     arm.move();
18
19     /*get current pose*/
20     geometry_msgs::PoseStamped currentPose;
21     currentPose=arm.getCurrentPose();
22     cout<<currentPose<<endl;
23
24     /*move z -10cm*/
25     geometry_msgs::Pose movePose;
26     movePose.position.x=currentPose.pose.position.x;
27     movePose.position.y=currentPose.pose.position.y;
28     movePose.position.z=currentPose.pose.position.z-0.1;
29     movePose.orientation=currentPose.pose.orientation;
30     arm.setPoseTarget(movePose);
31     arm.move();
32
33     currentPose=arm.getCurrentPose();
34     cout<<currentPose<<endl;
35
36     return 0;
37 }
    
```

- 16~17줄: Stand_by pose로 움직이기
- 20~22줄: 현재 끝 부분의 위치 및 자세 표시하기
- 25~31줄: 끝 부분의 위치 변화시키기
x,y,z 값과 변화량을 조절해서 코드를 변화시킬 수 있습니다.
(z 위치를 +10cm 움직입니다. 단위 m)
- 33~34줄: 현재 끝 부분의 위치 및 자세를 표시하기

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

//변환 코드 빌드

```
$ roslaunch tracer_gazebo_sim tracer_empty_world.launch
```

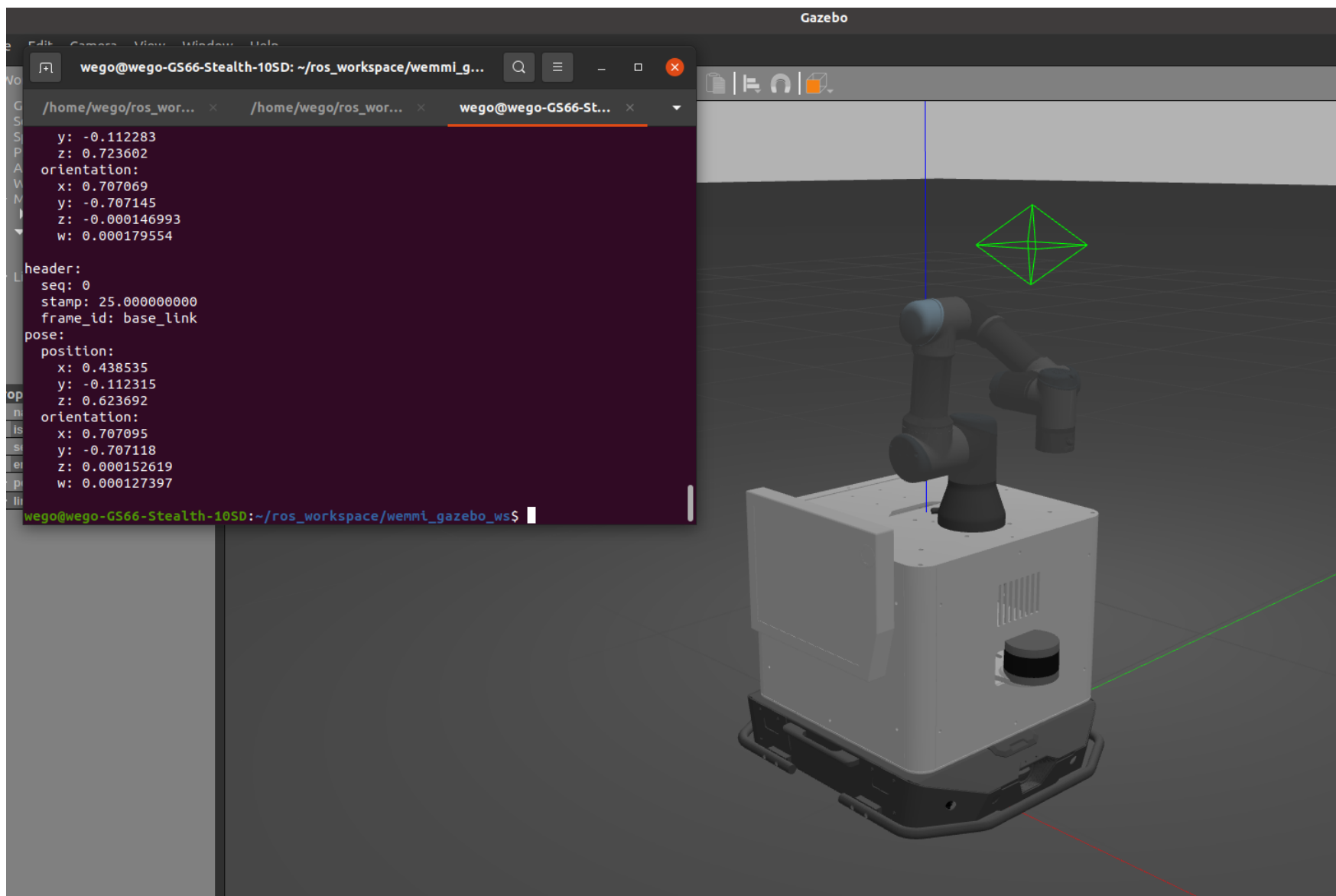
//Gazebo simulation 실행

```
$ roslaunch mobile_manipulator_moveit_config move_group.launch
```

// moveit 실행

```
$ rosrn mobile_manipulator_example move_ee_manipulator
```

// 작성된 코드 실행





본사(기술연구소 및 사무실) : 16914 경기도 용인시 기흥구 구성로 357(청덕동) 용인테크노밸리 B동 513호

기술연구소(서울) : 04799 서울특별시 성동구 성수동2가 280-13, 삼환디지털벤처타워 401호

대표전화 : 031 – 229 – 3553

팩스 : 031 – 229 – 3554

제품문의: go.sales@wego-robotics.com

기술문의: go.support@wego-robotics.com