Simulation 2

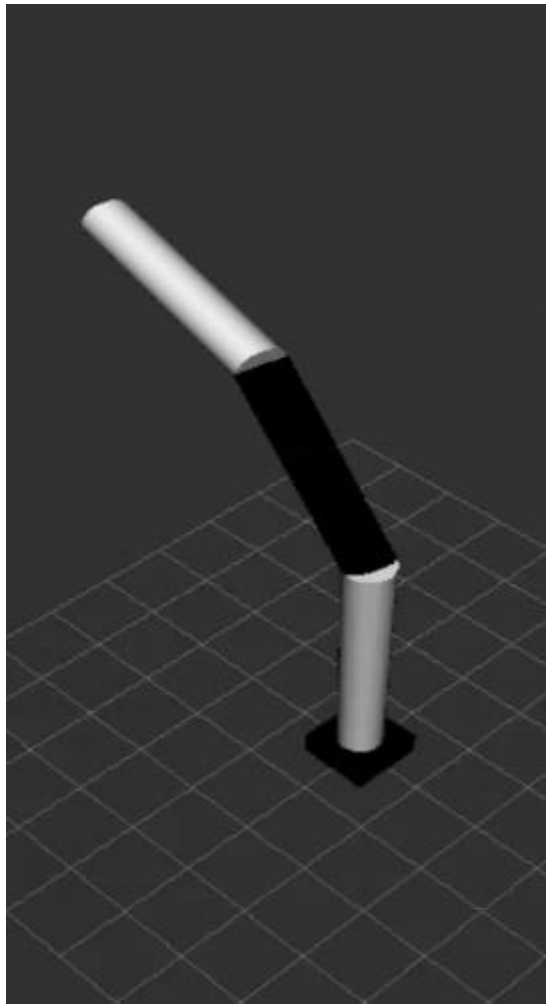
로봇팔 세미나 - 김혜윤 -

Contents

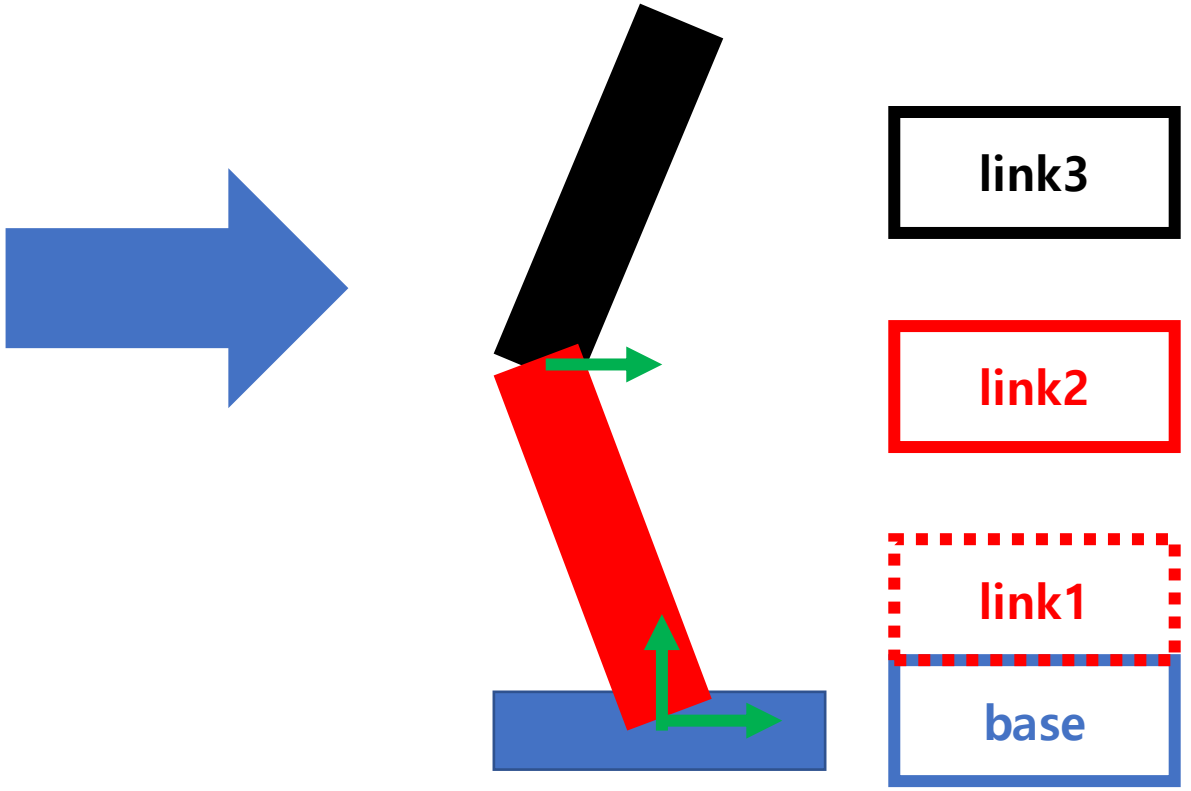
1. URDF 파일 수정
   - link 1 수정 / <collision>, <inertial> property
   - <transmission> / <gazebo>
2. Ros 용어
3. Gazebo
   - spawn.launch / control.launch
4. xyz 좌표로 joint angle 계산

URDF 파일 수정

link 1 수정



link 1의 길이와 반지름을 0으로 만들어 base와 link2를 연결한다.



link 1 수정

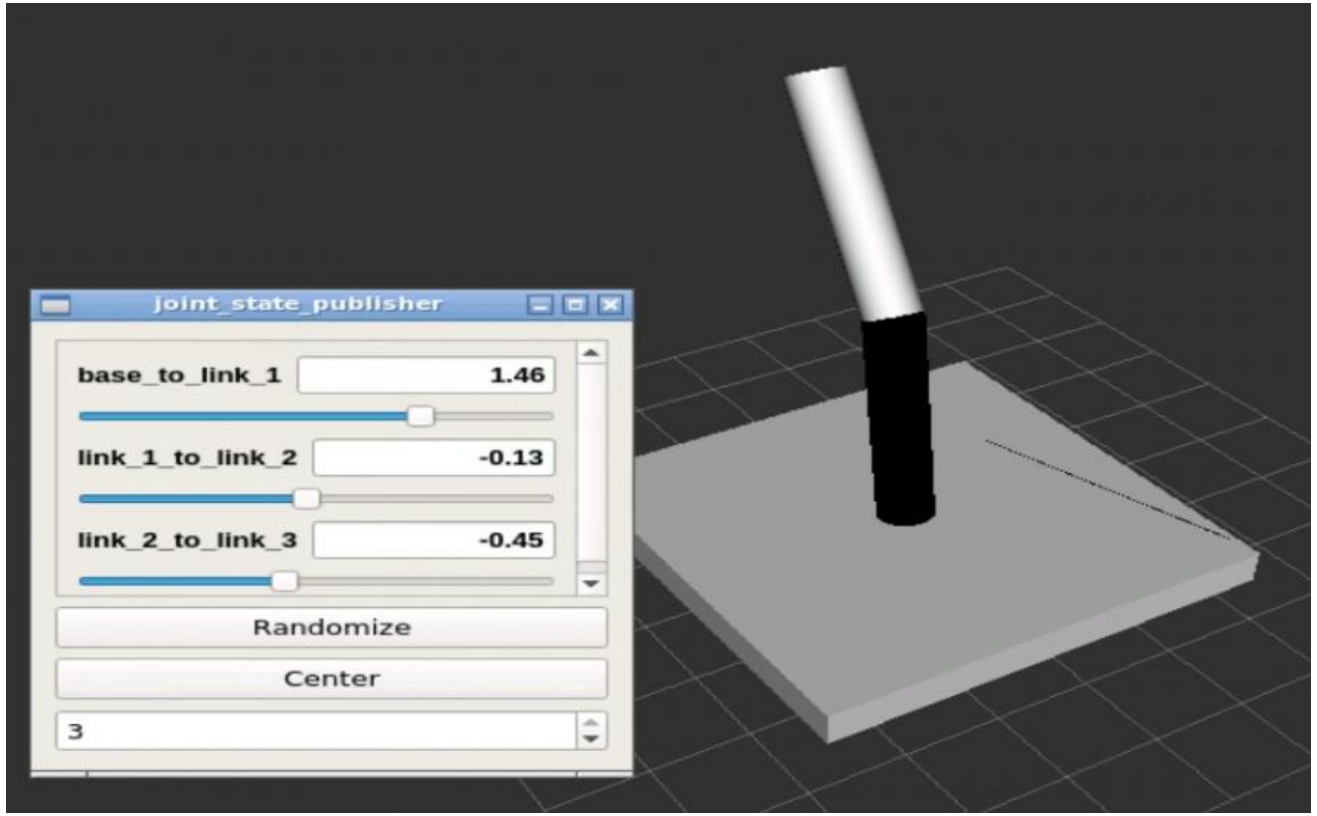
```
<link name='link_1'>
    <visual>
        <origin xyz='0 0 0' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.0' length='0.0'/>
        </geometry>
        <material name='White'/>
    </visual>
    <collision>
        <origin xyz='0 0 0.025' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.0' length='0.0'/>
        </geometry>
    </collision>
    <inertial>
        <mass value="1"/>
        <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
</link>
```

<cylinder
radius='0.0'
length='0.0'/>

Origin 좌표도 수정

link 1 수정

```
user:~$ roslaunch manipulator_3dof rviz.launch
```



roslaunch manipulator_3dof rviz.launch

하단의 Graphical tools  **클릭해서 확인**

<collision>, <inertial> property

```
<link name='link_2'>
    <visual>
        <origin xyz='0 0 1.5' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.3' length='3'/>
        </geometry>
        <material name='Black'/>
    </visual>
    <collision>
        <origin xyz='0 0 1.5' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.3' length='3'/>
        </geometry>
    </collision>
    <inertial>
        <mass value="1"/>
        <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
</link>
```

Gazebo
시뮬레이션을 위한
collision, inertial
property 추가

<collision>, <inertial> property

```
<link name='link_2'>
    <visual>
        <origin xyz='0 0 1.5' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.3' length='3'/>
        </geometry>
        <material name='Black'/>
    </visual>
    <collision>
        <origin xyz='0 0 1.5' rpy='0 0 0'/>
        <geometry>
            <cylinder radius='0.3' length='3'/>
        </geometry>
    </collision>
    <inertial>
        <mass value="1"/>
        <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
</link>
```

- **<collision>**
  Gazebo에서 필수
  없으면 invisible로 취급

- **<inertial>**
  항상 mass > 0
  ixx, iyy, izz값이 0이면
  가속도가 무한대로 갈 수
  있음

<transmission>

```
<transmission name="trans_base_to_link_1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="base_to_link_1">
        <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor_base_to_link_1">
        <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>
```

- **<transmission>**

  actuator와 joint 연결

  Gazebo에서 hardwareInterface는 EffortJointInterface로

<gazebo>

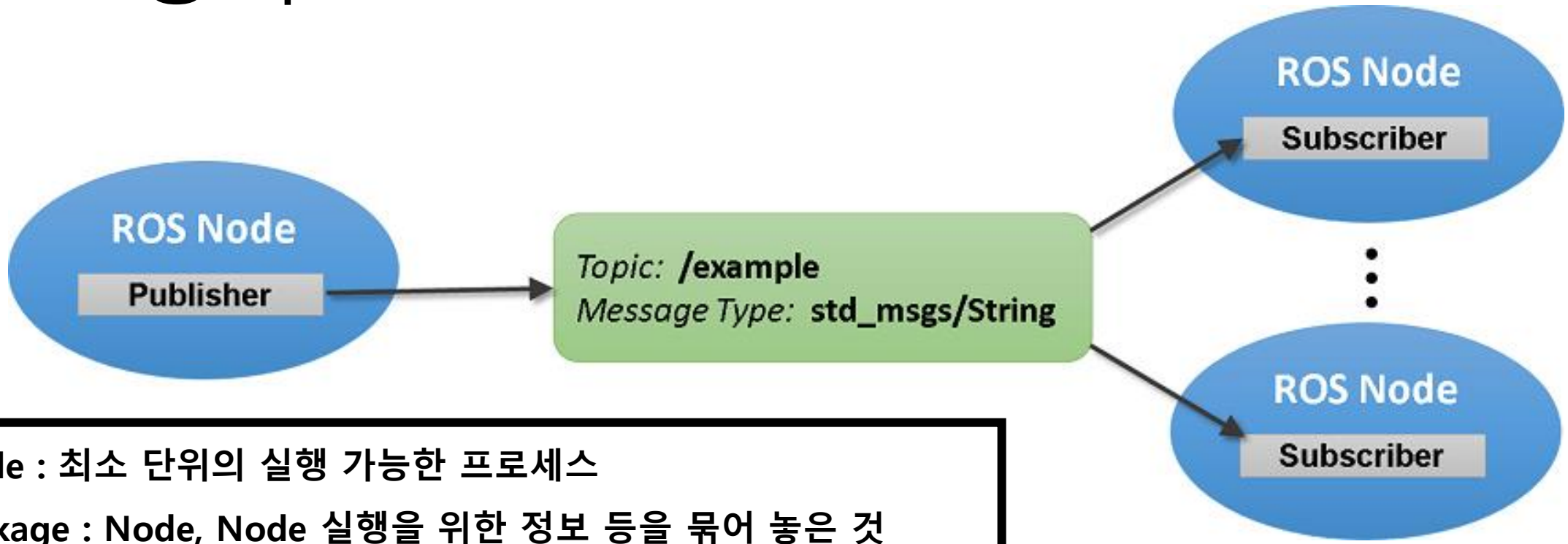
```
<gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so"/>
</gazebo>
```

<plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so"> </plugin>

Gazebo plugin 추가

Ros 용어

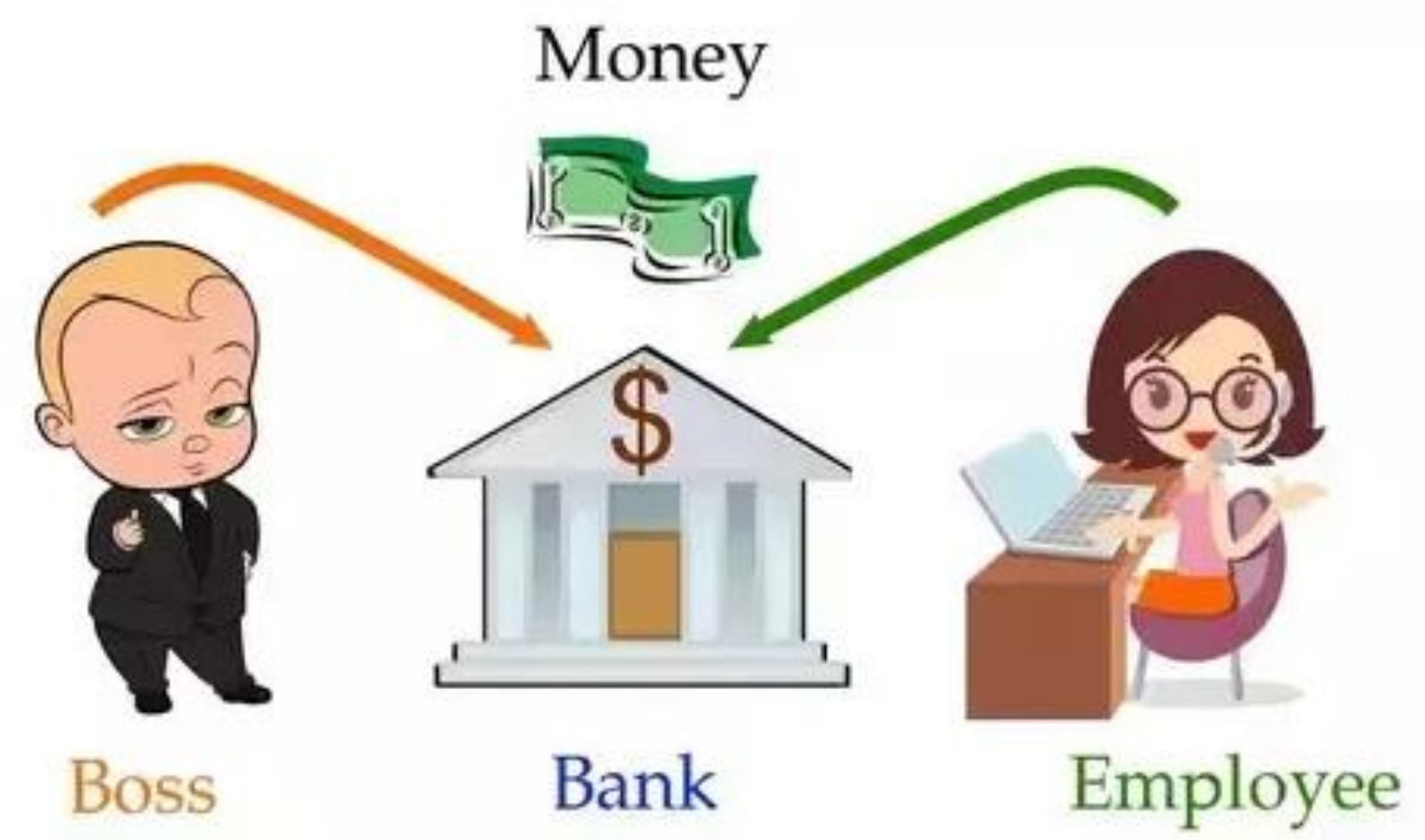
Ros 용어



- **Node : 최소 단위의 실행 가능한 프로세스**
- **Package : Node, Node 실행을 위한 정보 등을 묶어 놓은 것**
- **Message : Node간 주고 받는 변수 형태의 데이터**
- **Topic : 단방향, 연속성을 가진 통신 방법**
- **Parameter : 네트워크에 지정된 변수. 외부에서 값을 바꾸고 다른 Node에서 쓸 수 있음**

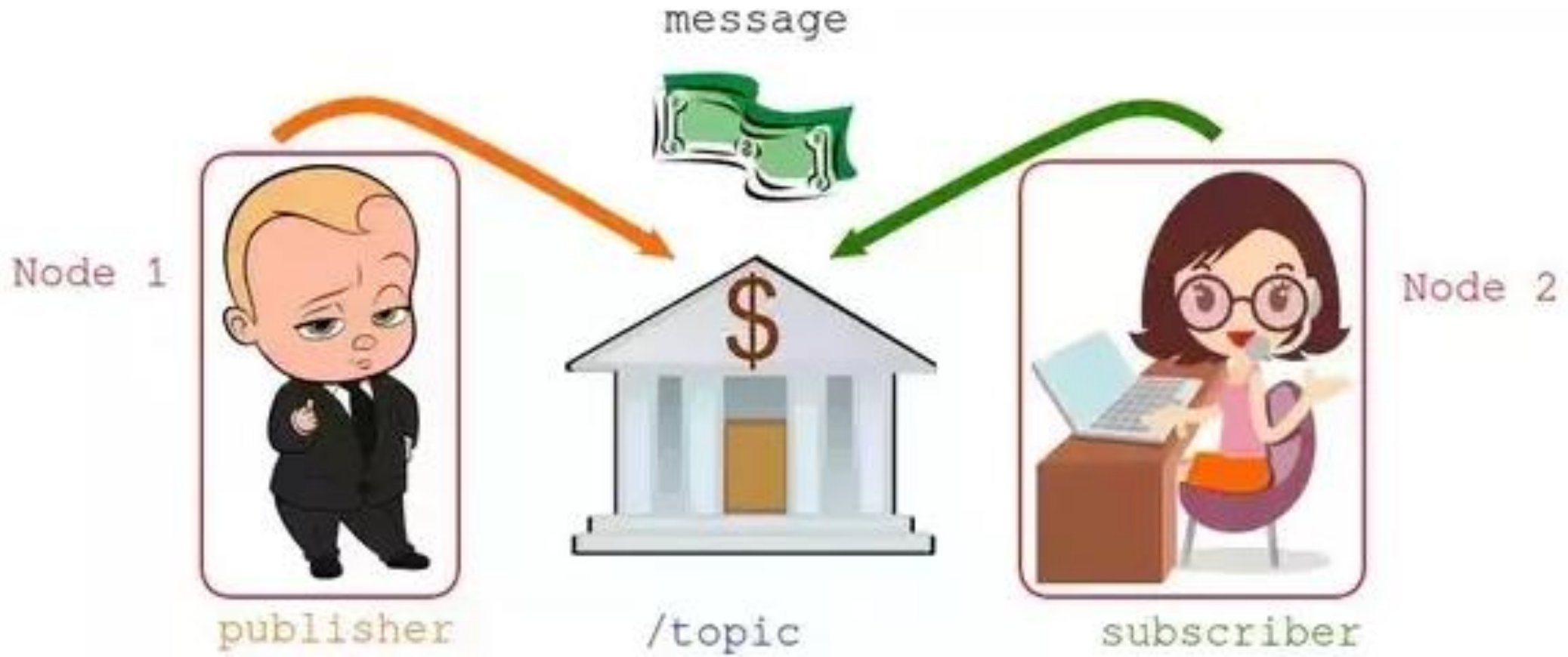
https://kr.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html

Ros 용어 예시



https://www.quora.com/What-is-ROS-Node-and-what-does-it-do

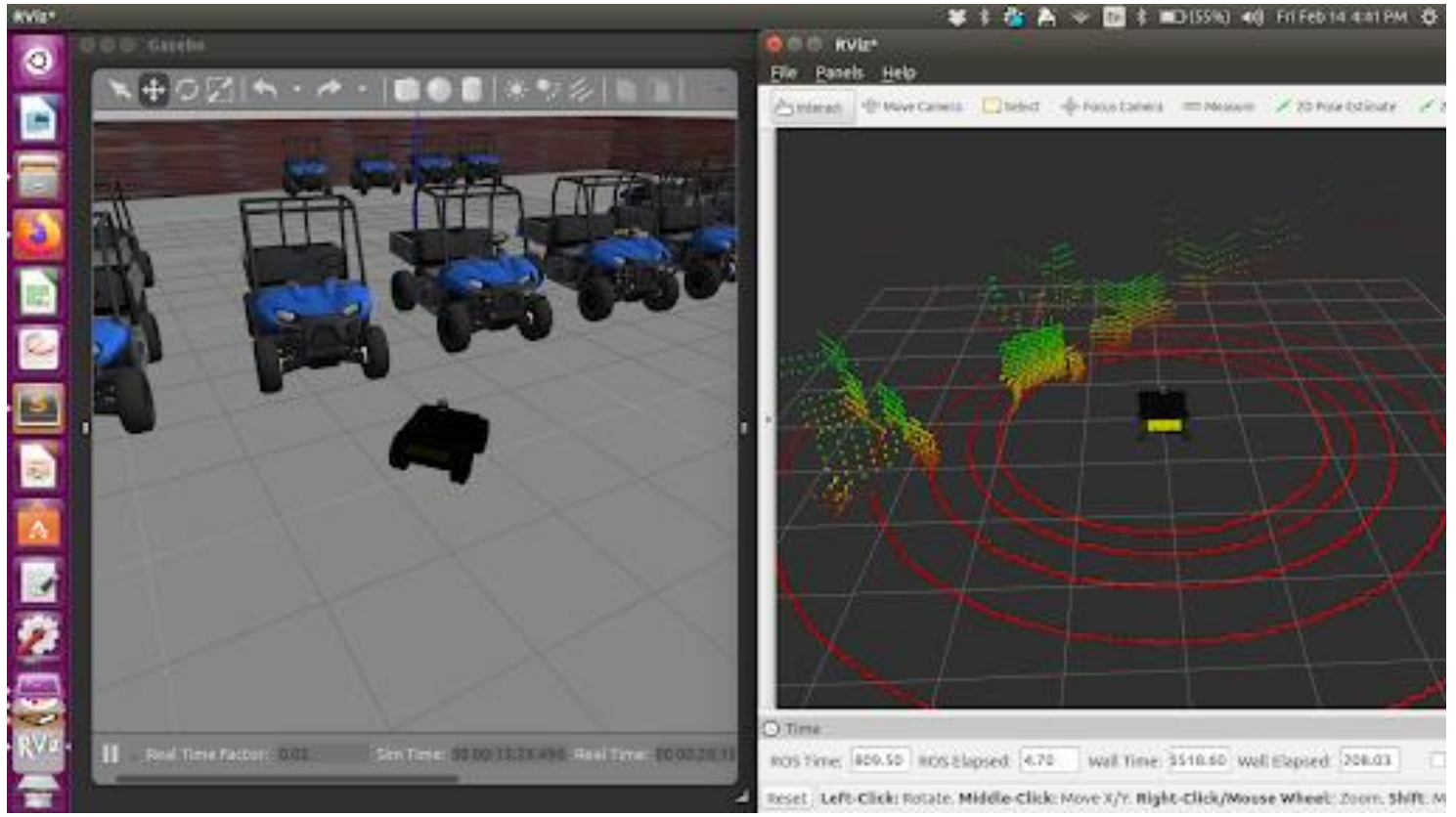
Ros 용어 예시



https://www.quora.com/What-is-ROS-Node-and-what-does-it-do

Gazebo

Gazebo



http://gazebosim.org/
https://www.google.com/url?sa=i&url=https%3A%2F%2Froboticsknowledgebase.com%2Fwiki%2Ftools%2Fgazebo-simulation%2F&psig=AOvVaw0Nd7rIqtzCbclIa2kZqxxH&ust=1610383332401000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCLCp7Mjnke4CFQAAAAAdAAAAABAD

spawn.launch

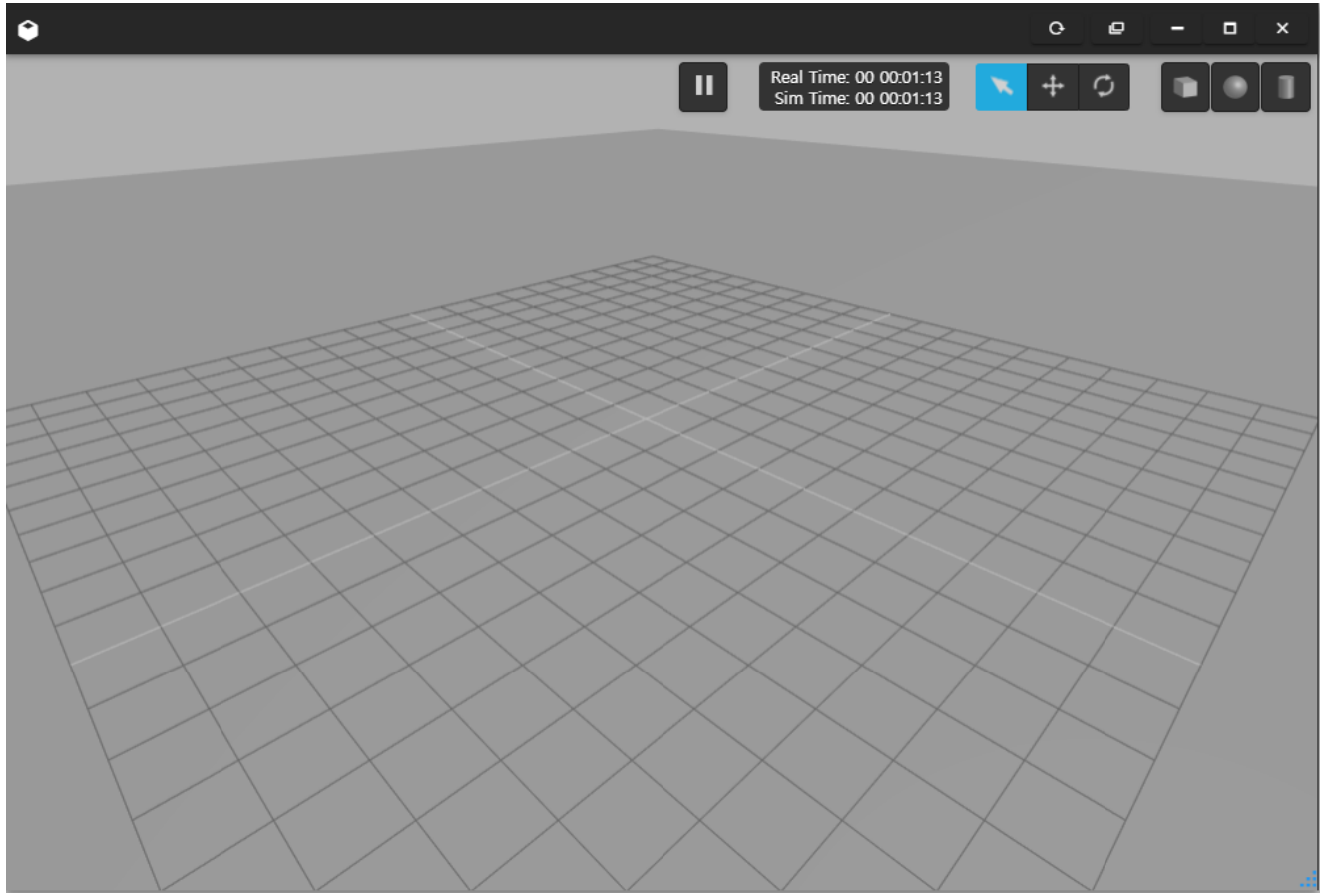
Launch폴더에 spawn.launch파일 생성

Gazebo 화면에 로봇을 생성하는 launch 파일

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>

    <param name="robot_description" textfile="$(find manipulator_3dof)/urdf/body_gazebo.urdf"/>

    <arg name="x" value="0.0" />
    <arg name="y" value="0.0" />
    <arg name="z" value="1.8" />
    <arg name="roll" value="0.0"/>
    <arg name="pitch" value="0.0"/>
    <arg name="yaw" value="0.0" />

    <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
          args="-urdf -param robot_description -model manipulator -x $(arg x) -y $(arg y) -z $(arg z)" />

</launch>
```

Gazebo - empty_world 생성

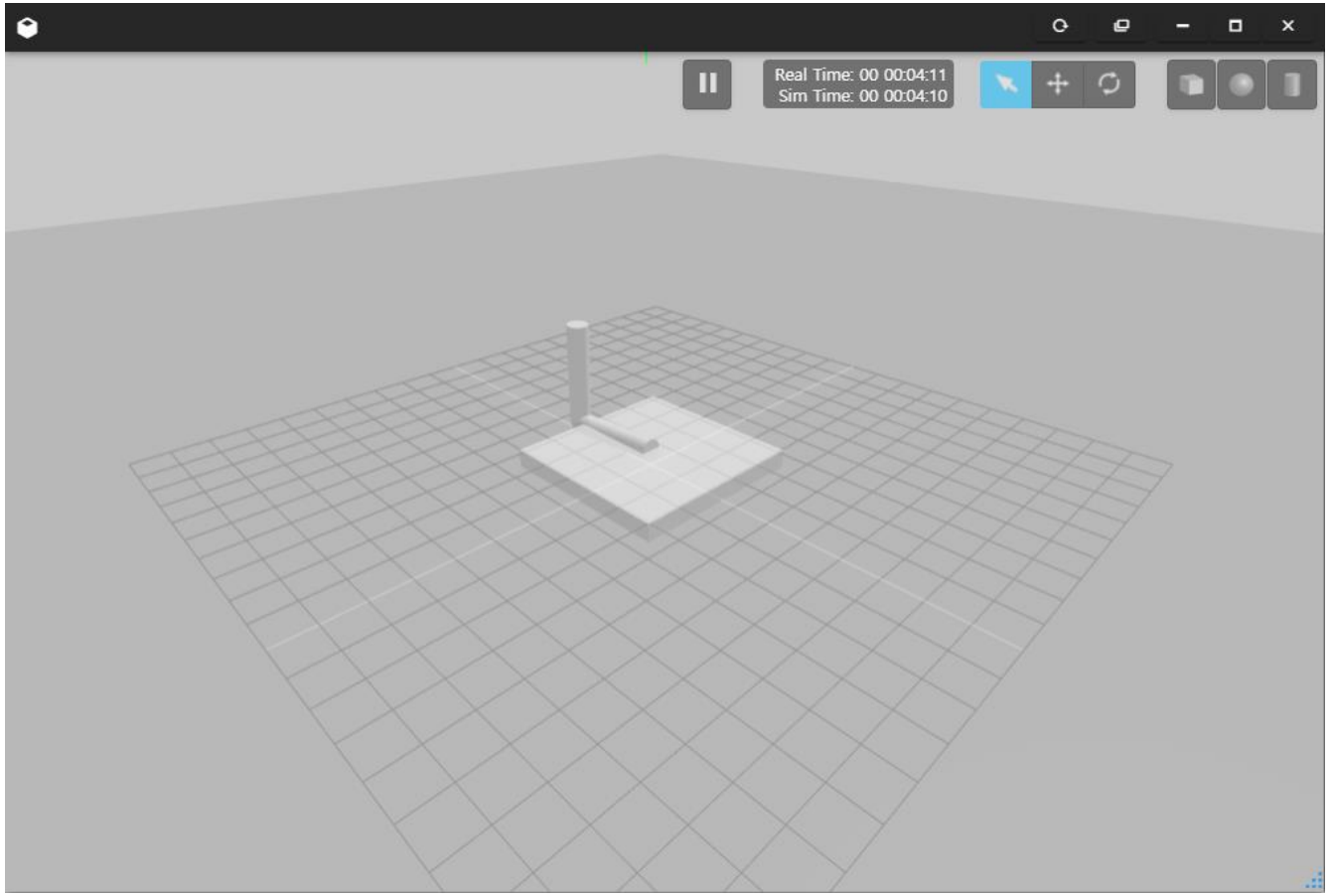
```
user:~$ roslaunch gazebo_ros empty_world.launch
```



roslaunch gazebo_ros empty_world.launch 실행 후 하단의 Gazebo 아이콘  클릭

Gazebo – spawn.lauch 실행

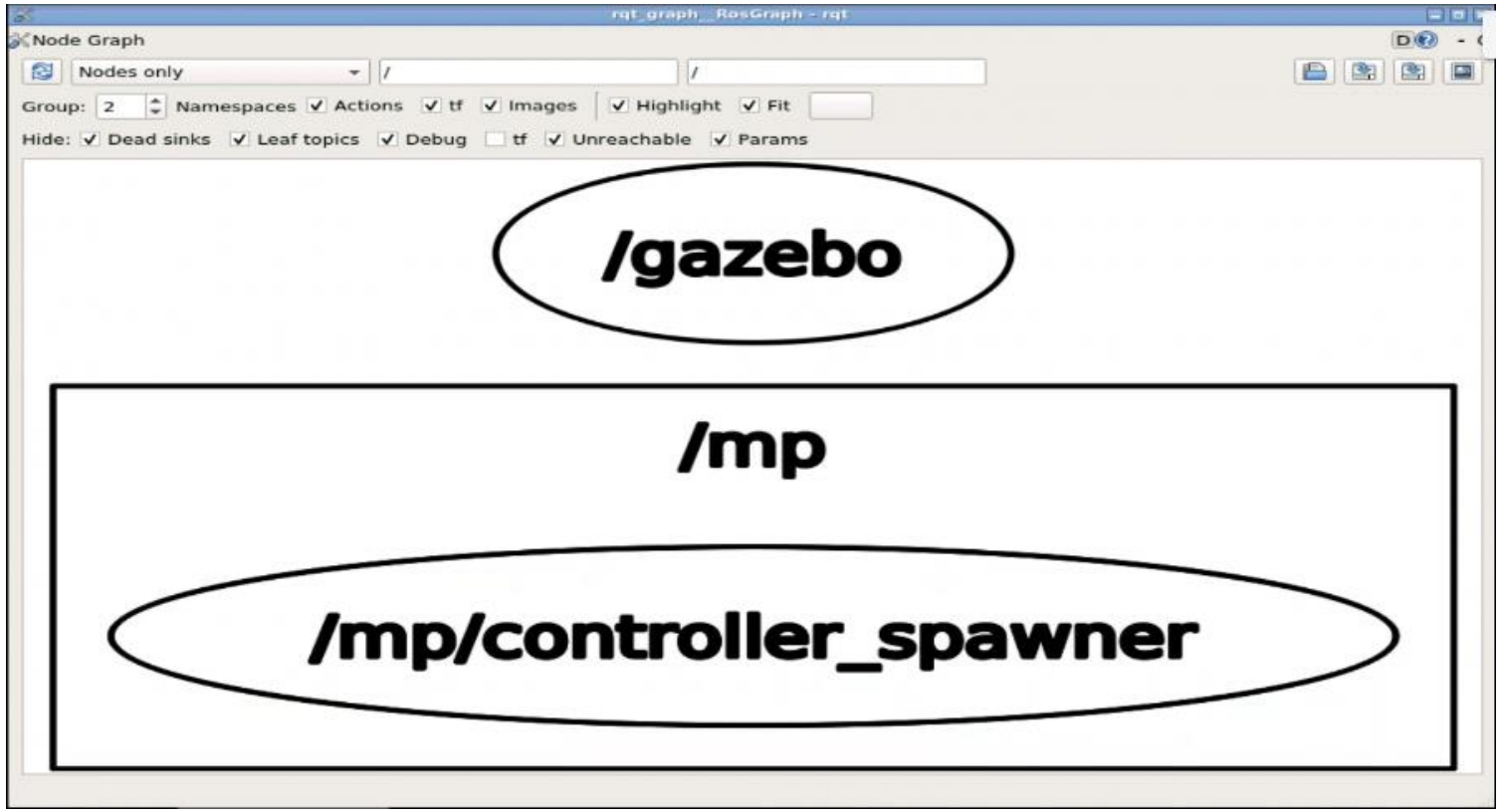
```
user:~$ roslaunch manipulator_3dof spawn.launch
```



roslaunch
manipulator_3dof
spawn.launch
실행 후
기다리면 urdf 파일의
robot이 생성됨

rqt_graph – node 그래프

```
user:~$ rqt_graph
```



rqt_graph 실행 후 하단의 Graphical tools  클릭 실행 중인 node 그래프로 확인

control.launch

launch파일에 control.launch 파일 생성

spawn.launch 파일에 controller_spawner만 추가해도 됨

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
    <group ns="/mp">
        <param name="robot_description" textfile="$(find manipulator_3dof)/urdf/body_gazebo.urdf"/>

        <arg name="x" value="0.0" />
        <arg name="y" value="0.0" />
        <arg name="z" value="1.8" />
        <arg name="roll" value="0.0"/>
        <arg name="pitch" value="0.0"/>
        <arg name="yaw" value="0.0" />
        <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
              args="-urdf -param robot_description -model manipulator -x $(arg x) -y $(arg y) -z $(arg z)" />

        <rosparam command="load" file="$(find manipulator_3dof)/config/control.yaml"/>

        <node name="controller_spawner" pkg="controller_manager" type="spawner"
            respawn="false" output="screen" ns="/mp"
            args="--namespace=/mp joint_state_controller base_to_link_1_position_controller link_1_to_link_2_position_controller link_2_to_link_3_position_controller"
        >
        </node>
    </group>
</launch>
```

namespace는 마음대로 지정, 위에서는 mp

Controller 이름을 argument로 넣어줌

control.yaml

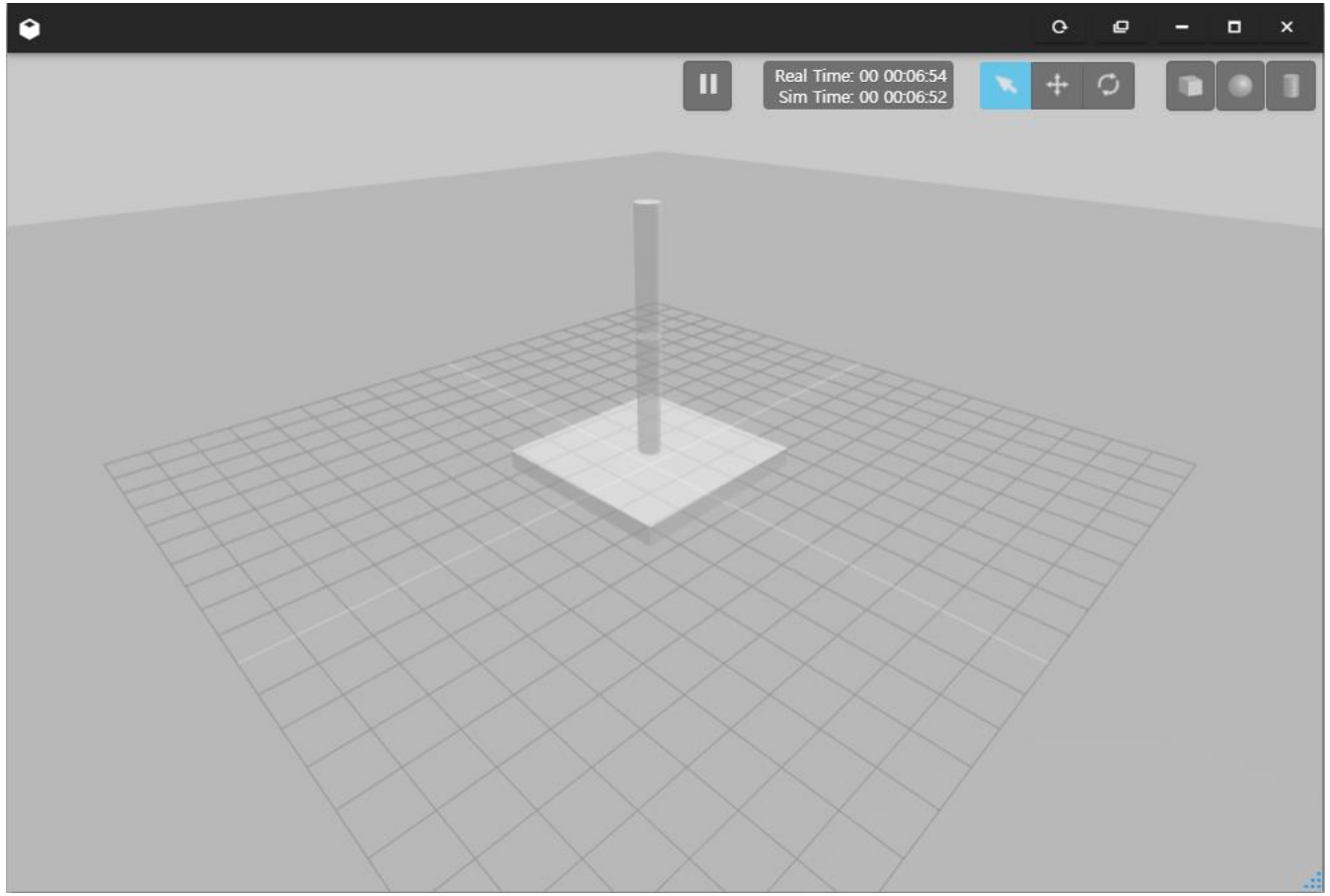
config파일에 control.yaml파일 생성

Controller type, joint 이름(urdf), pid 계수 입력

```yaml
# Publish all joint states -----------------------
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50

# Position Controllers ---------------------------
base_to_link_1_position_controller:
  type: effort_controllers/JointPositionController
  joint: base_to_link_1
  pid: {p: 100.0, i: 0.01, d: 10.0}
link_1_to_link_2_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_1_to_link_2
  pid: {p: 100.0, i: 0.01, d: 10.0}
link_2_to_link_3_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_2_to_link_3
  pid: {p: 100.0, i: 0.01, d: 10.0}
```

Gazebo – control.lauch 실행

```
user:~$ roslaunch manipulator_3dof control.launch
```



roslaunch

manipulator_3dof

control.launch

실행 후

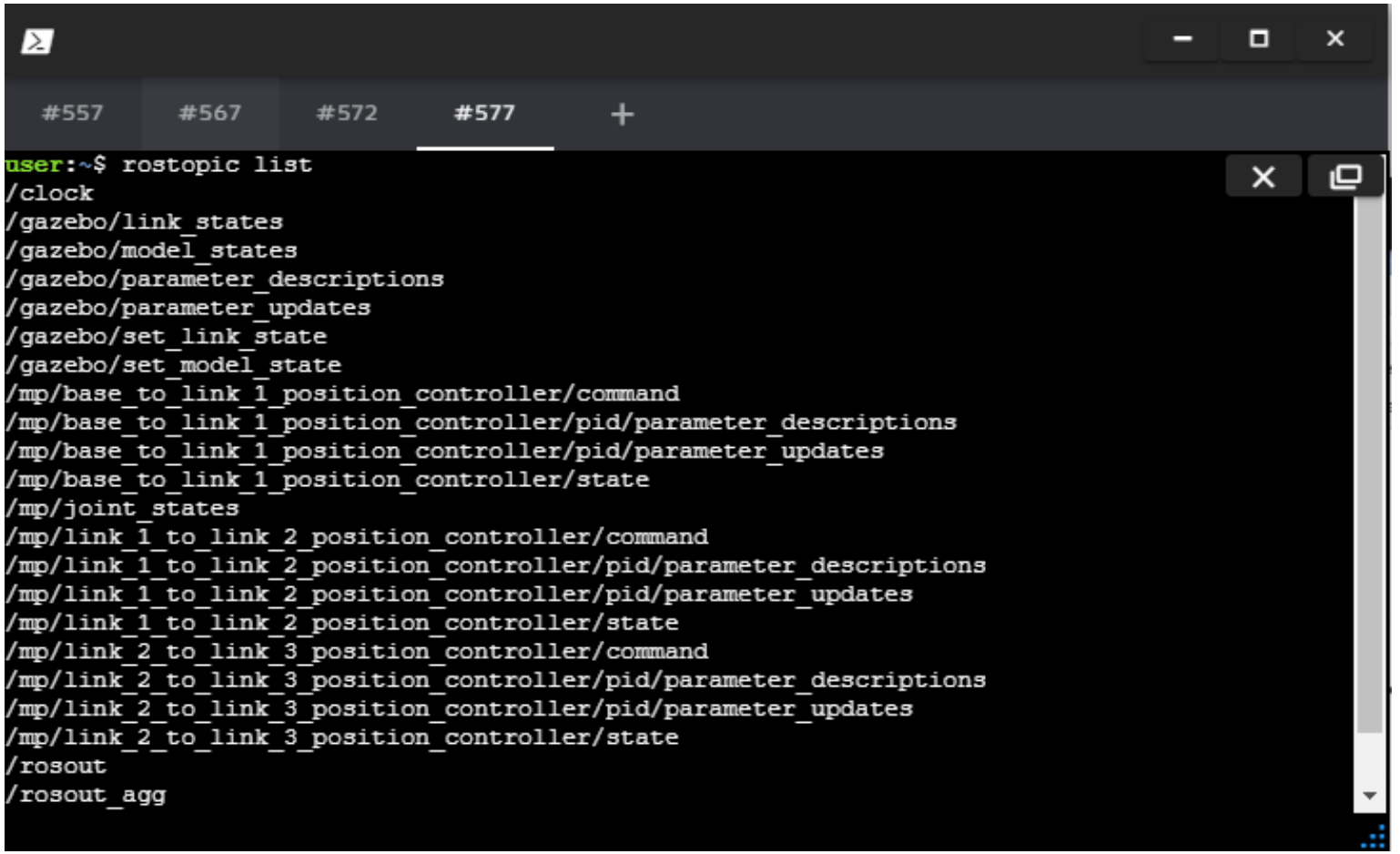
기다리면 urdf 파일의

robot이 생성됨

rostopic list

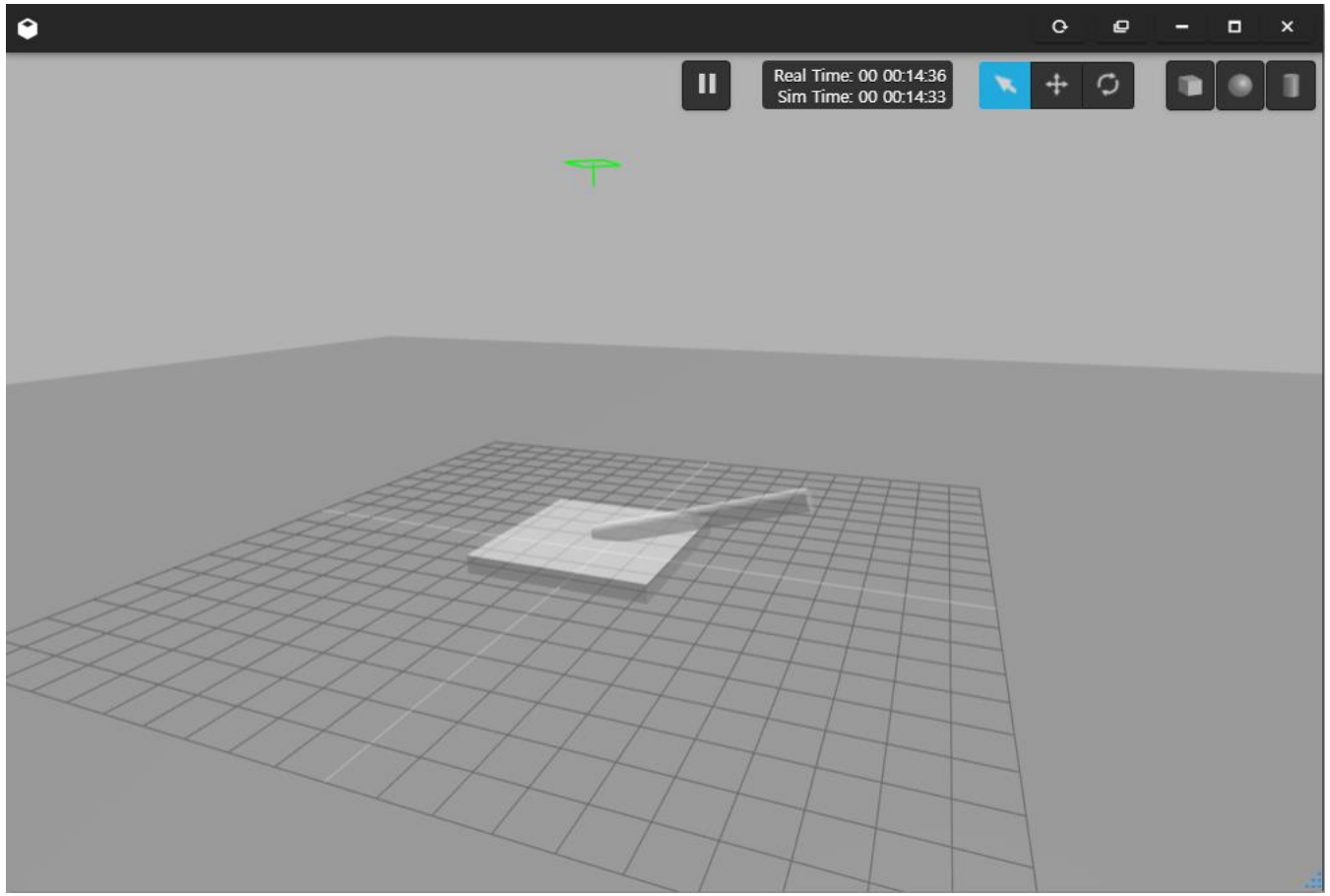
rostopic list : 현재 topic 확인

```
user:~$ rostopic list
```



Gazebo – topic 생성

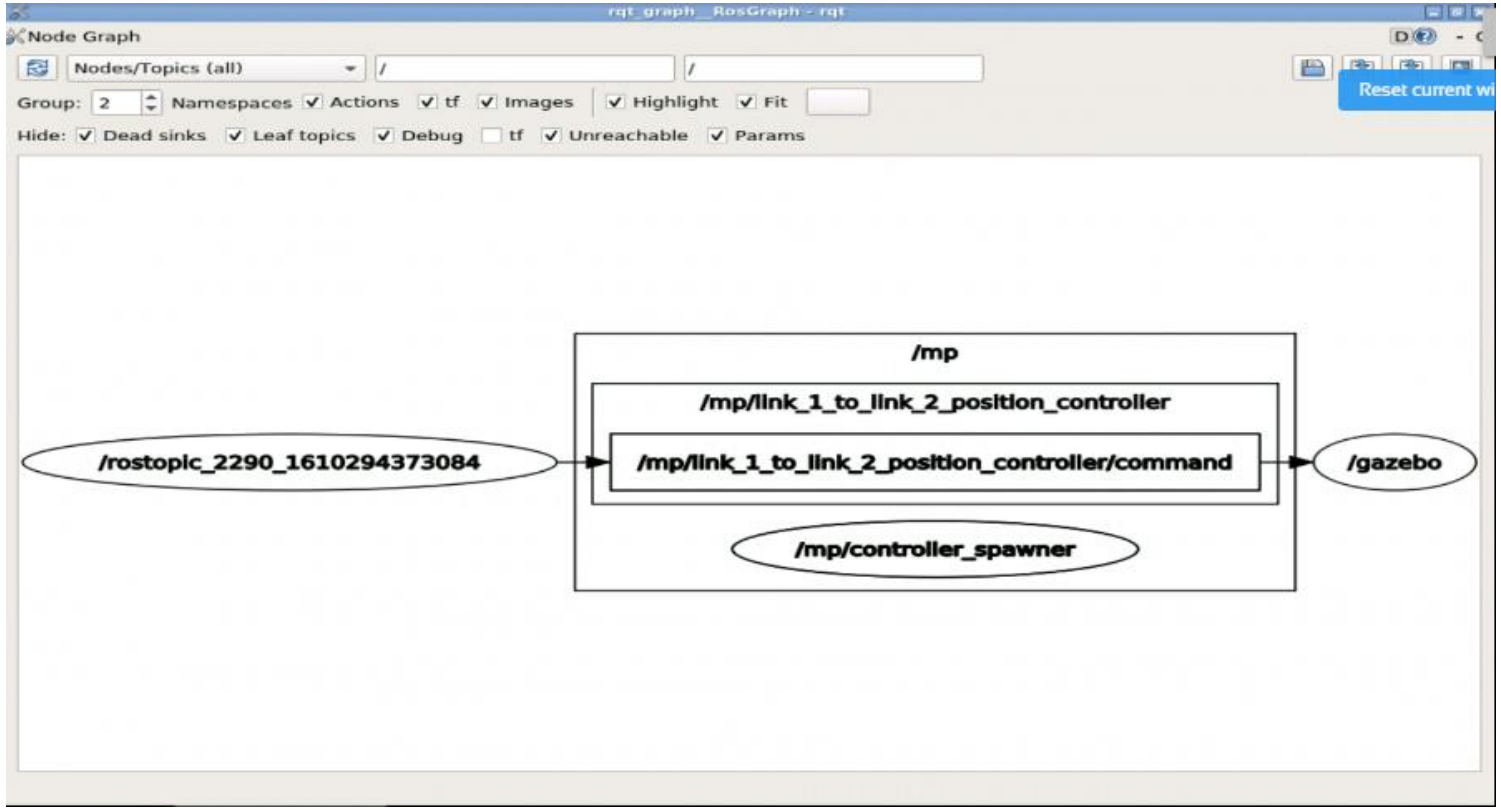
```
user:~$ rostopic pub -1 /mp/link_1_to_link_2_position_controller/command std_msgs/Float64 "
data: 1.0"
```



rostopic 명령어로
data를 1.0(rad)로 바꾸어
topic을 생성(pub)해서
Joint를 움직인다
-1 옵션 : topic 1번 생성

Gazebo – topic 생성

```
user:~$ rostopic pub /mp/link_1_to_link_2_position_controller/command std_msgs/Float64 "dat
a: 1.0"
```



rqt_graph를 실행하여 Node에서 topic을 publish/subscribe하는 것을 볼 수 있다

-1 옵션이 없어서 topic을 계속 생성함

xyz 좌표로 joint angle 계산

manipulator_pub.py

```python
#!/usr/bin/python
import rospy
import math
import time
from std_msgs.msg import Float64

class ManipulatorPub(object):

    def __init__(self):

        self.base_to_link_1_position = rospy.Publisher(
            '/mp/base_to_link_1_position_controller/command', Float64, queue_size=1)
        self.link_1_to_link_2_position = rospy.Publisher(
            '/mp/link_1_to_link_2_position_controller/command', Float64, queue_size=1)
        self.link_2_to_link_3_position = rospy.Publisher(
            '/mp/link_2_to_link_3_position_controller/command', Float64, queue_size=1)

        rospy.loginfo("Starting ManipulatorPub...")

    def xyz_move(self, x, y, z):···

def xyz_input_test():
    manipulatorpub_obj = ManipulatorPub()

    while not rospy.is_shutdown():
        print(">>>>>>  Input  >>>>>>\n")
        x = float(raw_input("Input x=>"))
        y = float(raw_input("Input y=>"))
        z = float(raw_input("Input z=>"))
        manipulatorpub_obj.xyz_move(x=x, y=y, z=z)
        print(">>>>>>@@@@@@@@@>>>>>>>>>>\n")

if __name__ == "__main__":
    rospy.init_node('pan_and_tilt_client')
    xyz_input_test()
```

- **script 폴더에 manipulator_pub.py 파일 생성**
- **xyz 좌표를 입력받아 joint angle을 publish하는 Publisher 필요**
- **ManipulatorPub Class 내부에 joint angle topic 3개를 publish 하는 Publisher 3개 생성**

xyz_move() - xyz 좌표로 joint angle 계산

```python
def xyz_move(self, x, y, z):
    l2 = 3
    l3 = 3

    base_to_link_1_msg = Float64()
    base_to_link_1_angle = math.atan2(y, x)
    base_to_link_1_msg.data = base_to_link_1_angle
    cos1 = math.cos(base_to_link_1_angle)
    sin1 = math.sin(base_to_link_1_angle)

    cos3 = (x**2 + y**2 + z**2 - l2*l2 - l3*l3)/(2*l2*l3)
    sin3 = math.sqrt(1 - cos3**2)
    link_2_to_link_3_msg = Float64()
    link_2_to_link_3_msg.data = math.atan2(sin3, cos3)

    alpha = math.atan2(z, math.sqrt(x**2 + y**2))
    beta = math.atan2(l3*sin3, l2+l3*cos3)
    link_1_to_link_2_msg = Float64()
    link_1_to_link_2_msg.data = alpha - beta

    rospy.loginfo("calculated x: " + str(cos1*(l2*math.cos(link_1_to_link_2_msg.data)+l3*math.cos(link_2_to_link_3_msg.data+link_1_to_link_2_msg.data))))
    rospy.loginfo("calculated y: " + str(sin1*(l2*math.cos(link_1_to_link_2_msg.data)+l3*math.cos(link_2_to_link_3_msg.data+link_1_to_link_2_msg.data))))
    rospy.loginfo("calculated z: " + str(l2*math.sin(link_1_to_link_2_msg.data)+l3*math.sin(link_2_to_link_3_msg.data+link_1_to_link_2_msg.data)))

    link_1_to_link_2_msg.data = link_1_to_link_2_msg.data - math.pi/2

    self.base_to_link_1_position.publish(base_to_link_1_msg)
    self.link_1_to_link_2_position.publish(link_1_to_link_2_msg)
    self.link_2_to_link_3_position.publish(link_2_to_link_3_msg)

    rospy.loginfo("Sending base_to_link_1 msg: " + str(base_to_link_1_msg))
    rospy.loginfo("Sending link_1_to_link_2 msg: " + str(link_1_to_link_2_msg))
    rospy.loginfo("Sending link_2_to_link_3 msg: " + str(link_2_to_link_3_msg))
```

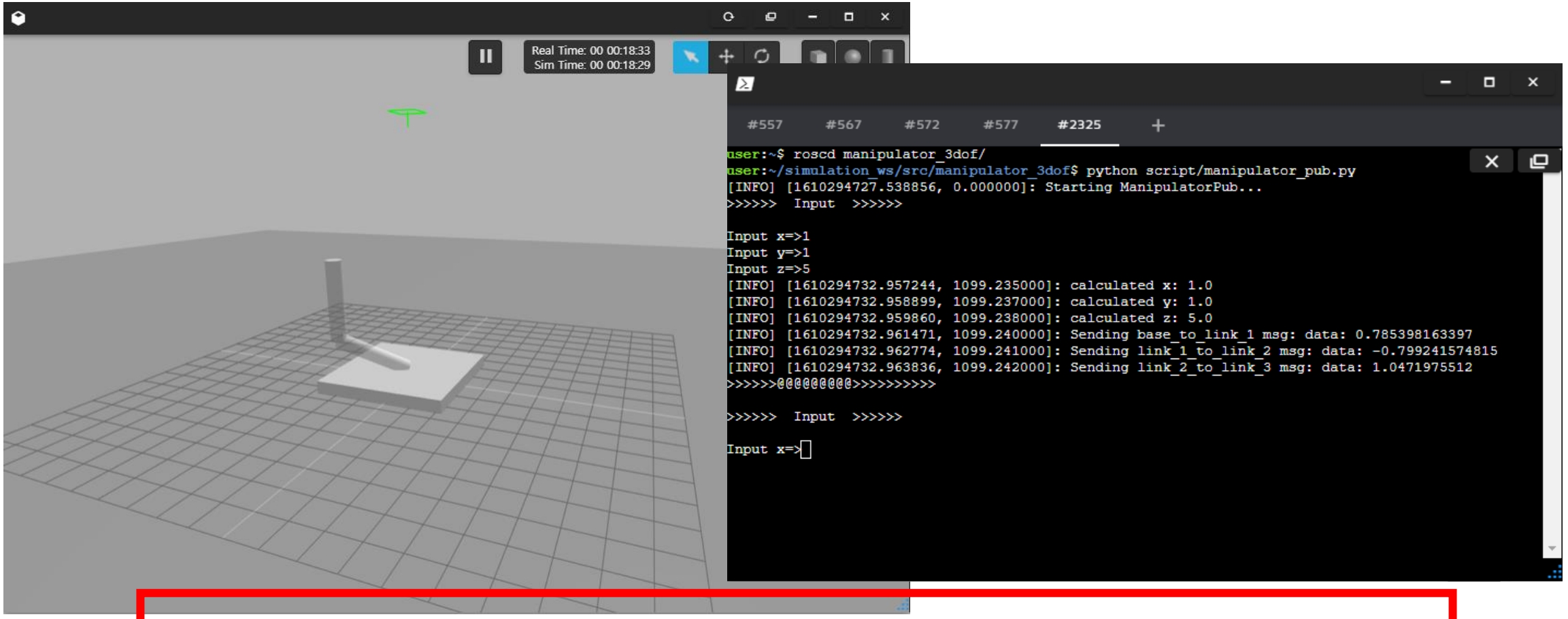
2, 3강 내용 위주로 joint angle 3개 계산

계산한 joint angle로 xyz 좌표 역계산으로 계산 결과 확인

각도 기준 맞추기 위해 90° 뺌

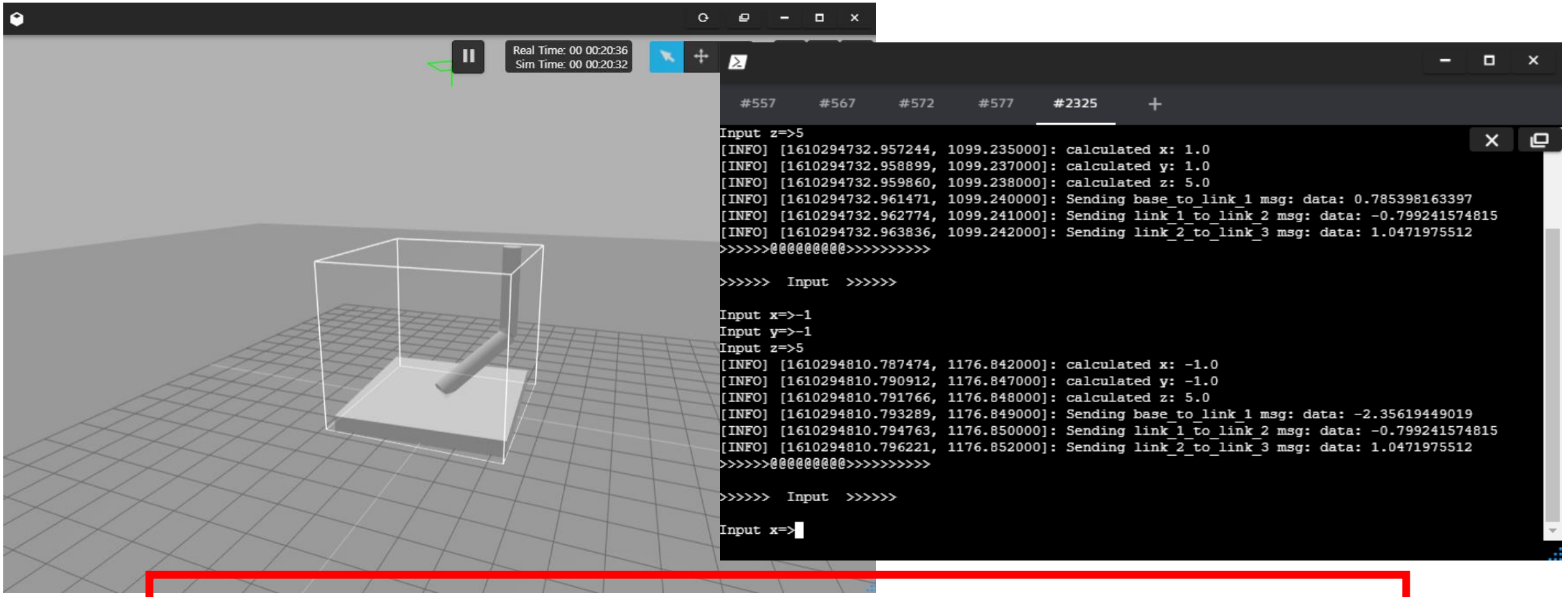
publish method로 topic publish

manipulator_pub.py 실행



manipulator_pub.py 실행 후
x: 1, y: 1, z: 5 입력하면 manipulator가 해당 좌표를 가리키는 모양으로 움직인다

manipulator_pub.py 실행



x: -1, y: -1, z: 5를 입력하면 manipulator가 높이를 유지한 채 돌아가는 것을 확인할 수 있다

Thank you