6장. 로봇과 시뮬레이터

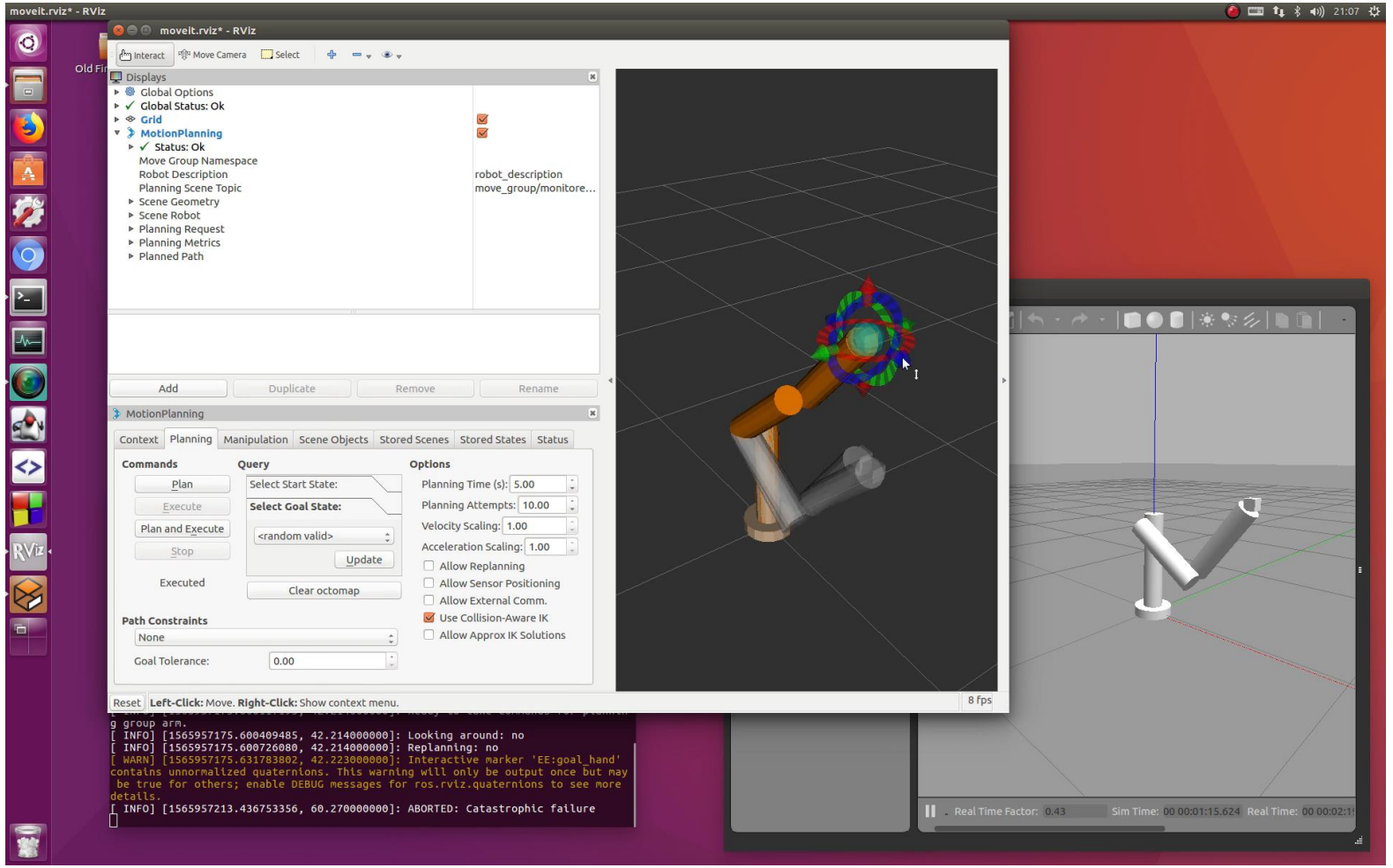
동의대학교 컴퓨터소프트웨어공학과
이종민 교수

목차

- 서브시스템: actuation, sensing, computing
- 완전한 로봇
- 시뮬레이터

서브시스템 (Sub-systems)

- 로봇은 여러 개의 서브시스템으로 구성: 구동(actuation), 감지(sensing), 계산(computing)

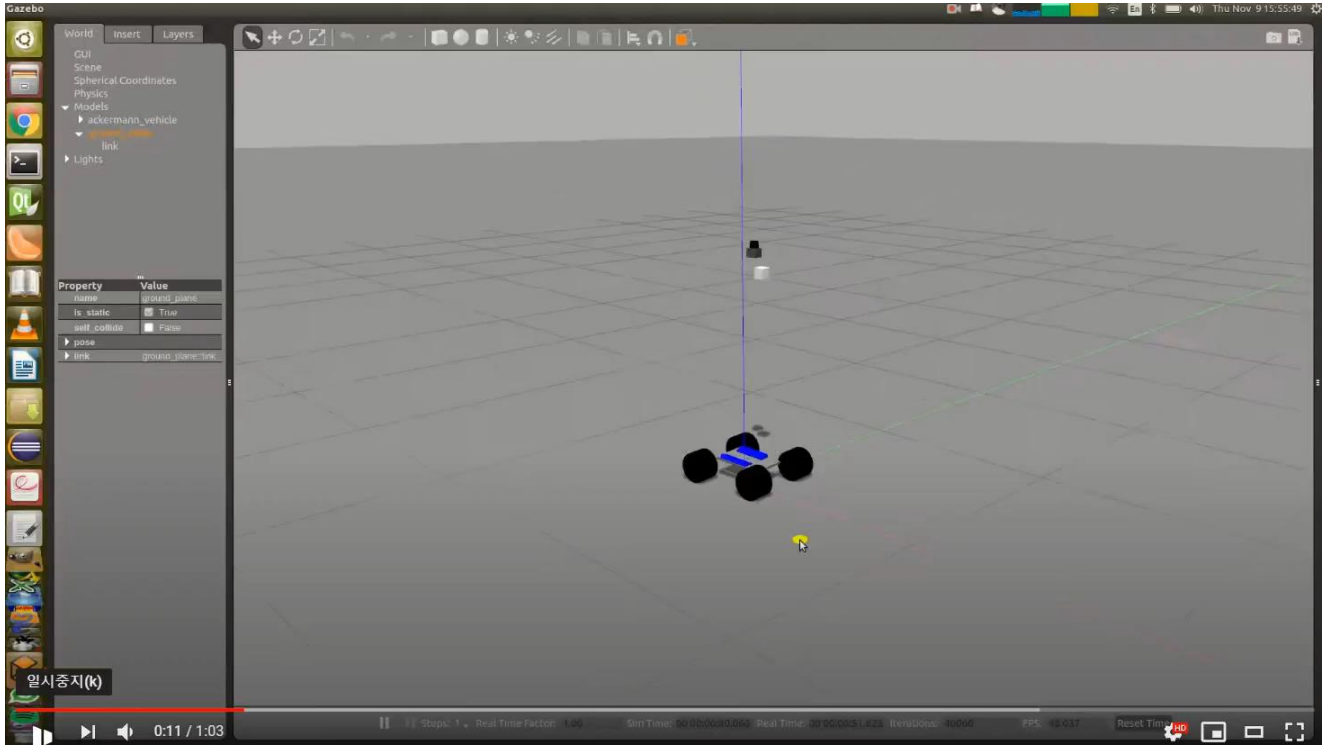


구동: 이동 플랫폼

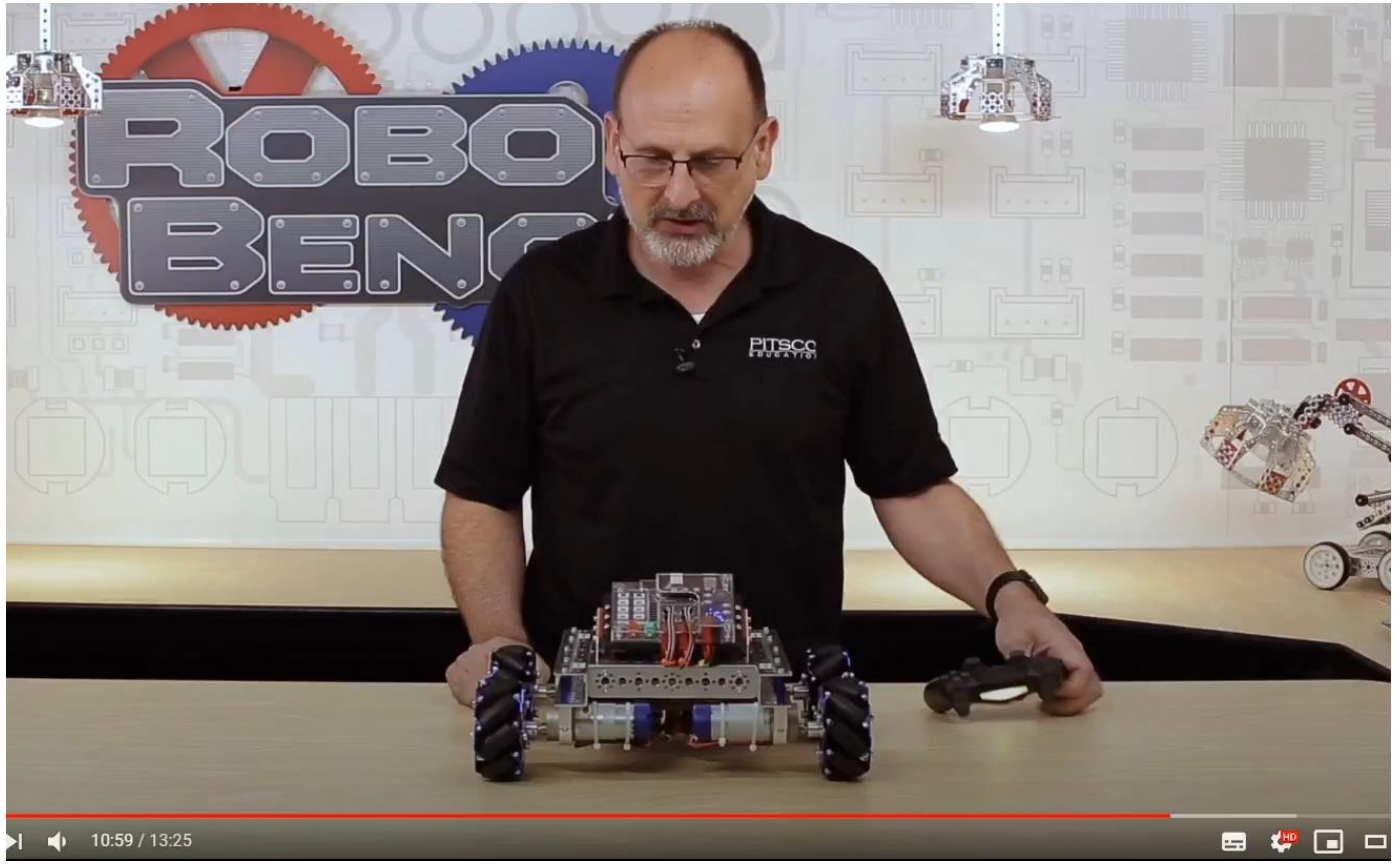
- 구동 방식: 차동 구동, 미끄럼 조향, 애커만 플랫폼
- 차동 구동(differential drive)
  - 독립적으로 구동되는 두 개의 바퀴를 앞뒤로 회전하여 전방향 이동
  - 예) 터틀봇

- 미끄럼 조향(skid steering)
  - 두 개의 바퀴를 사용하는 차동 구동을 4개 이상의 바퀴를 사용하도록 확장
  - 참고. https://robohub.org/drive-kinematics-skid-steer-and-mecanum-ros-twist-included/

- 애커만(Ackermann) 플랫폼
  - 승용차 구동 방식: 뒷바퀴는 전방을 향하고 있으며, 앞바퀴를 회전하여 이동 시 방향 조정
  - 참고
    - http://ros-developer.com/2017/11/09/ackermann-steering-car-robot-model-with-simulation-in-gazebo/
    - https://github.com/trainman419/ackermann_vehicle-1
    - https://youtu.be/o5wnRCzdUMo



- 메카넘 휠(mechnum wheel) 방식 - 홀로노믹(holonomic) 플랫폼
  - 앞의 논홀로노믹(non-holonomic) 방식에 비해 전방향 이동이 자유로움.
  - 참고. https://youtu.be/iTsWy9z32G0



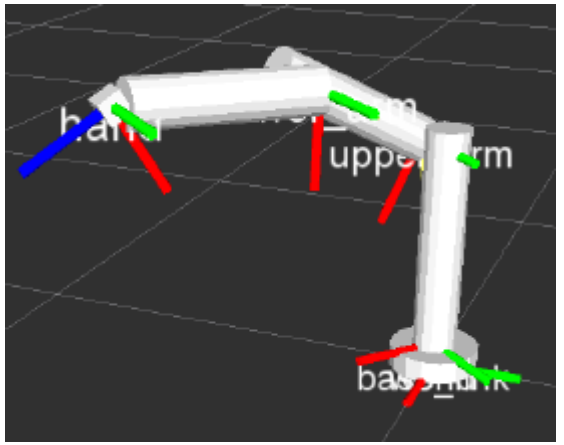
Twist 메시지

- 선속도와 각속도를 정의
  - 선속도: x축 방향으로의 이동 속도 (m/sec)
  - 각속도: x, y, z축 방향의 회전 속도 (radian/sec) - 참고: π rad = 180°

- geometry_msgs/Twist.msg
  - geometry_msgs/Vector3 linear
    geometry_msgs/Vector3 angular

- geometry_msgs/Vector3.msg
  - float64 x
    float64 y
    float64 z

구동: 로봇 팔



- 로봇 팔은 로봇과 마찬가지로 **링크(link)**와 **관절(joint)**로 구성

- 관절의 종류
  - 회전 관절(revolute joint)
  - 선형 관절(linear joint) or 직동(prismatic) 관절

- 말단 장치(end effector) - https://en.wikipedia.org/wiki/Robot_end_effector
  - 로봇 팔의 끝 부분으로 집게와 같이 외부와 상호 작용할 수 있는 장치를 의미

- 참고:
  - https://pinkwink.kr/906
  - https://pinkwink.kr/1007
  - https://pinkwink.kr/1013
  - https://github.com/jhu-dvrk/dvrk-ros

로봇 팔의 자유도

- 자유도 (DOF, Degree Of Freedom)
  - 관절의 수 - 보통 액추에이터의 수와 같음.
  - 자유로운 이동을 위해서는 6축 자유도 이상 필요

센서

- 광학 카메라
  - 자료형: **sensor_msgs/Image**

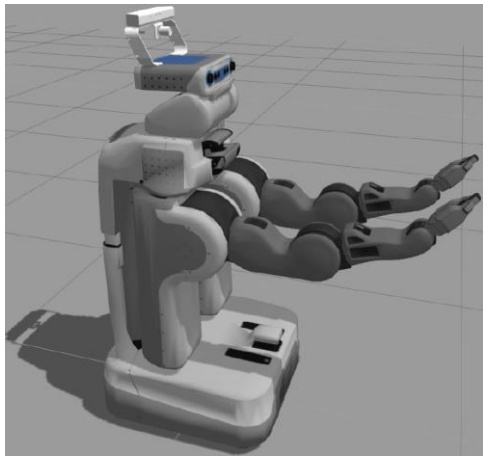
- 깊이 카메라 (depth camera)
  - 자료형: **sensor_msgs/PointCloud2**
  - 구조광(structured light) 방식 - Microsoft Kinect
  - 비행 시간(time-of-flight) 방식

- 레이저 스캐너 or LiDAR (Light Detection And Ranging)
  - https://en.wikipedia.org/wiki/Lidar
  - 자료형: **sensor_msgs/LaserScan**
  - 로봇 주위 물체와의 거리 정보 계산에 활용

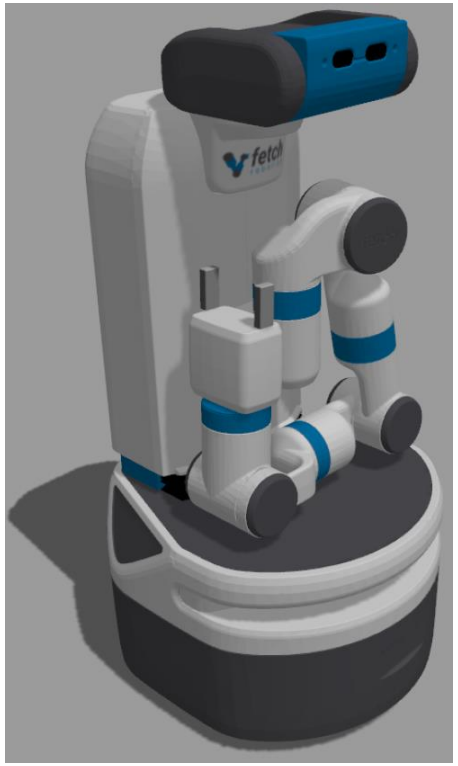
- 축 인코더 - 바퀴의 회전 수 계산하여 정확한 이동 거리 계산에 활용

완전한 로봇

PR2



Fetch

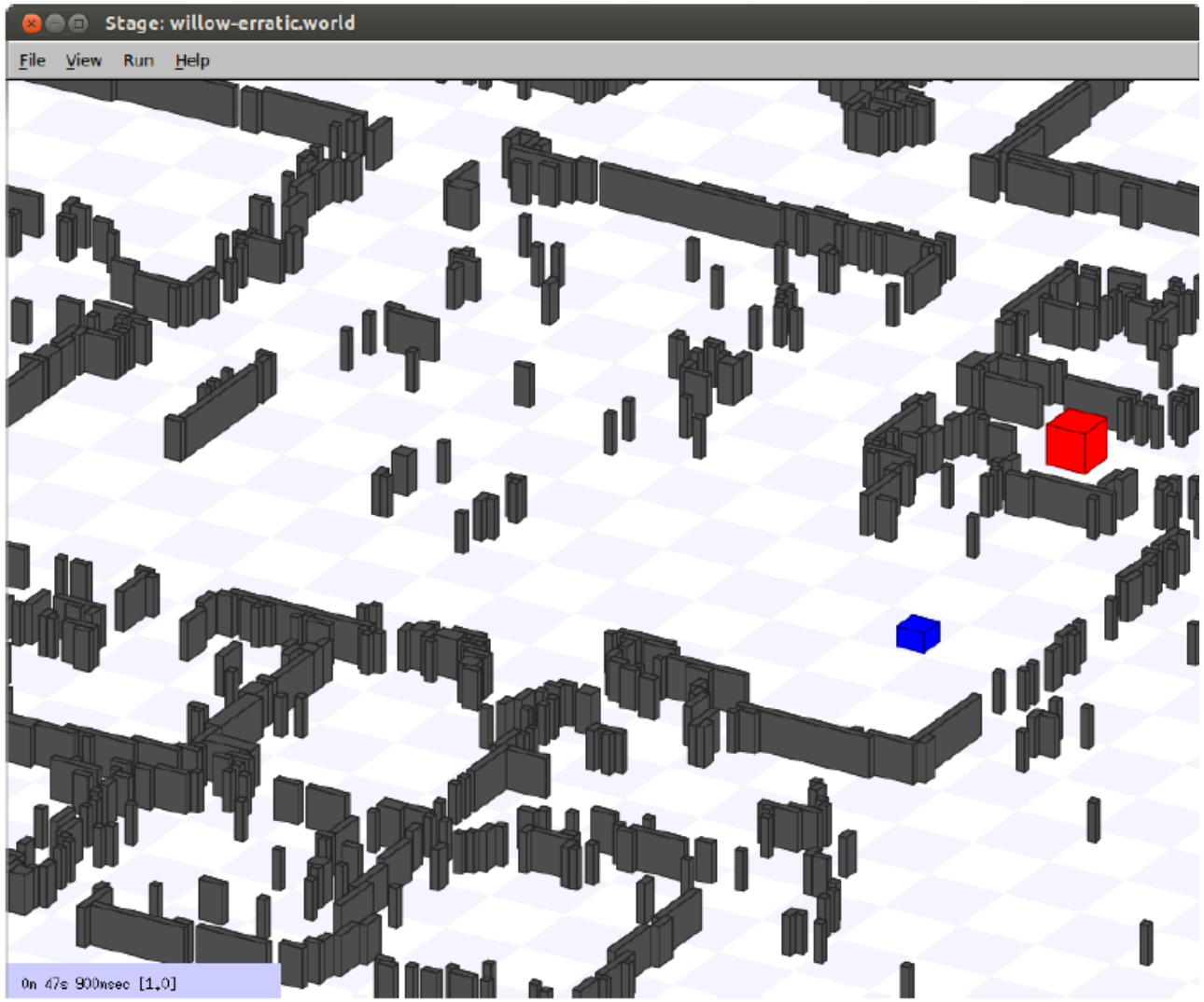


NASA Robotnat 2 (R2)

Turtlebot2

시뮬레이터

- Stage Robot Simulator
  - http://rtv.github.io/Stage/

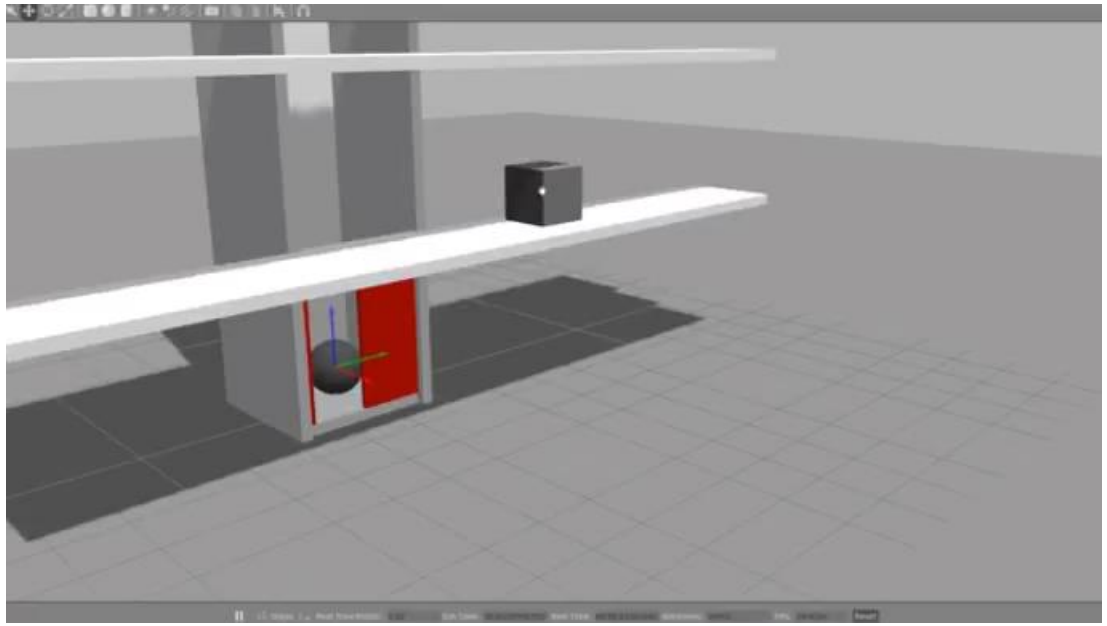


- Gazebo
  - http://gazebosim.org/
  - ODE, Bullet, Simbody 등의 동역학 시뮬레이션 지원
  - SDF (Simulation Description Format) 지원
  - Gazebo 안 가상 세계에 접근할 수 있는 플러그인 지원 및 개발 가능

```
<?xml version="1.0" ?>
<sdf version="1.5">
  <world name="default">

    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>
```



7장. WANDER-BOT

동의대학교 컴퓨터소프트웨어공학과
이종민 교수

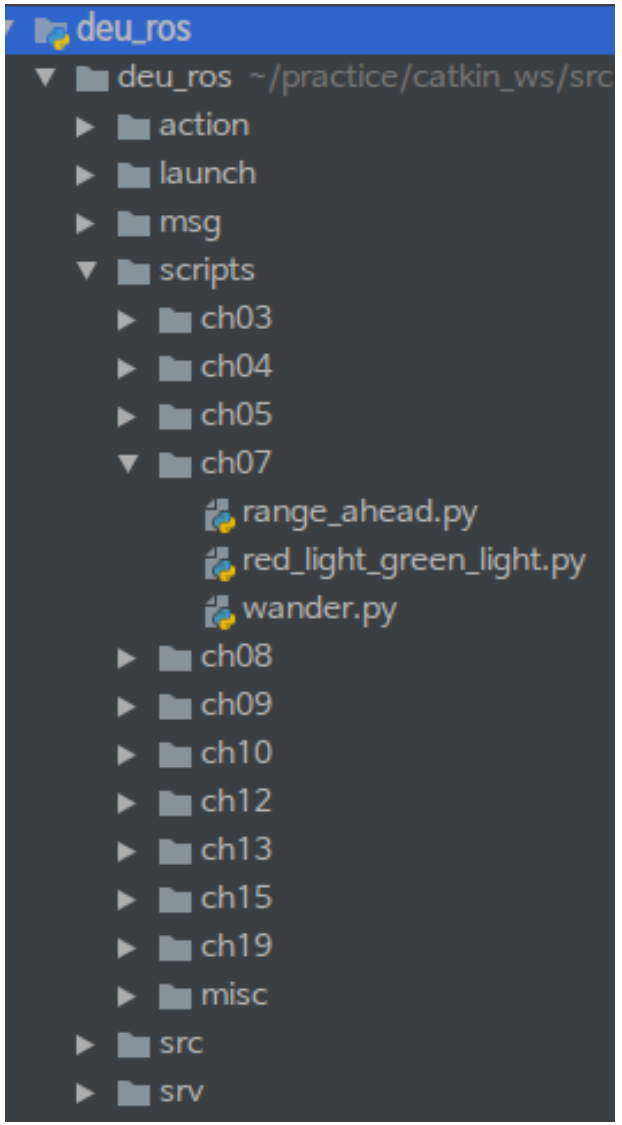
목차

- 주행과 정지
- LiDAR 센서 데이터 읽기
- 원더봇 구현

프로그램 구조

- 교재
  - wanderbot 패키지를 생성하여 구현
    $ cd ~/practice/catkin_ws/src
    $ catkin_create_pkg wanderbot rospy **geometry_msgs sensor_msgs**

- 강의
  - 기존 deu_ros 패키지에 구현
  - scripts/ch07 디렉토리에 파이썬 스크립트 구현



package.xml 확인

```
51  <buildtool_depend>catkin</buildtool_depend>
52  <build_depend>rospy</build_depend>
53  <build_depend>std_msgs</build_depend>
54  <build_depend>actionlib_msgs</build_depend>
55  <build_depend>message_generation</build_depend>
56  <build_depend>geometry_msgs</build_depend>
57  <build_depend>sensor_msgs</build_depend>
58  <build_export_depend>rospy</build_export_depend>
59  <build_export_depend>std_msgs</build_export_depend>
60  <exec_depend>rospy</exec_depend>
61  <exec_depend>std_msgs</exec_depend>
62  <exec_depend>actionlib_msgs</exec_depend>
63  <exec_depend>message_runtime</exec_depend>
64  <exec_depend>geometry_msgs</exec_depend>
65  <exec_depend>sensor_msgs</exec_depend>
```

CMakeLists.txt

```
10  find_package(catkin REQUIRED COMPONENTS
11    rospy
12    std_msgs
13    message_generation
14    actionlib_msgs
15    geometry_msgs
16    sensor_msgs
17  )
```

```
76  generate_messages(
77    DEPENDENCIES
78    std_msgs
79    geometry_msgs   # ch15/FakeSensor.srv
80    actionlib_msgs
81  )
```

red_light_green_light.py

```
1  #!/usr/bin/env python
2
3  import rospy
4  from geometry_msgs.msg import Twist
5
6  cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)  # <1>
7  rospy.init_node('red_light_green_light')
8
9  red_light_twist = Twist()  # <2>
10 green_light_twist = Twist()
11 green_light_twist.linear.x = 0.5  # <3>
12
13 driving_forward = False
14 light_change_time = rospy.Time.now()
15 rate = rospy.Rate(10)
```

cmd_vel 토픽 발행자 정의

Twist 객체 생성: 선속도/각속도 모두 0.0

선속도 0.5 m/sec 로 직진

직진/정지 발행 변환 시각을 현재 시각으로 초기화

10Hz 주기로 동작

```
17 while not rospy.is_shutdown():
18     if driving_forward:
19         cmd_vel_pub.publish(green_light_twist)  # <4>
20     else:
21         cmd_vel_pub.publish(red_light_twist)
22     if rospy.Time.now() > light_change_time:  # <5>
23         driving_forward = not driving_forward
24         light_change_time = rospy.Time.now() + rospy.Duration(3)
25     rate.sleep()  # <6>
```

직진 cmd_vel 발행

정지 cmd_vel 발행

직진/정지 토글

직진/정지 발행 변환 시각을 3초 뒤로 설정

Twist 메시지

- 메시지 정의

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

- geometry_msgs/Vector3

```
float64 x
float64 y
float64 z
```

- 사용법
  - linear.x : 선속도 (m/sec)
  - angular.z : 각속도 (rad/sec)

실행 방법

Terminal 1:

$ roscore

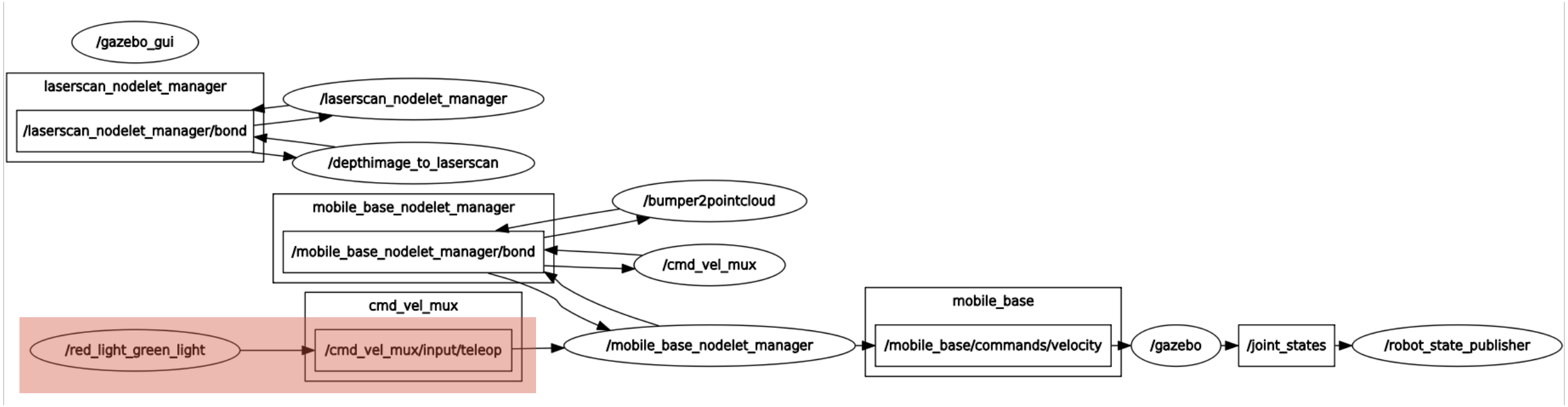
Terminal 2:

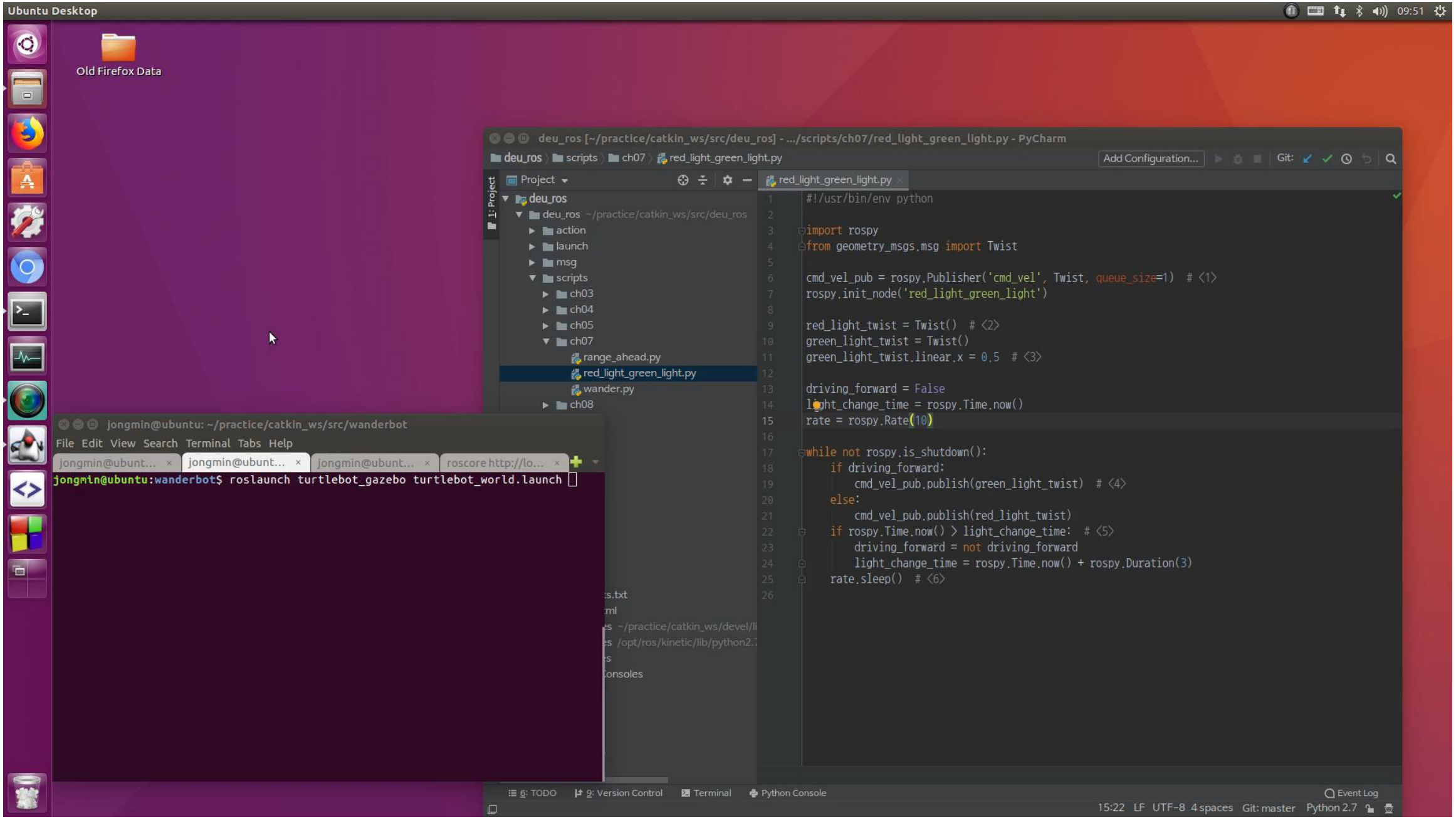
$ roslaunch turtlebot_gazebo turtlebot_world.launch

Terminal 3:

$ rosrun deu_ros red_light_green_light.py **cmd_vel:=cmd_vel_mux/input/teleop**

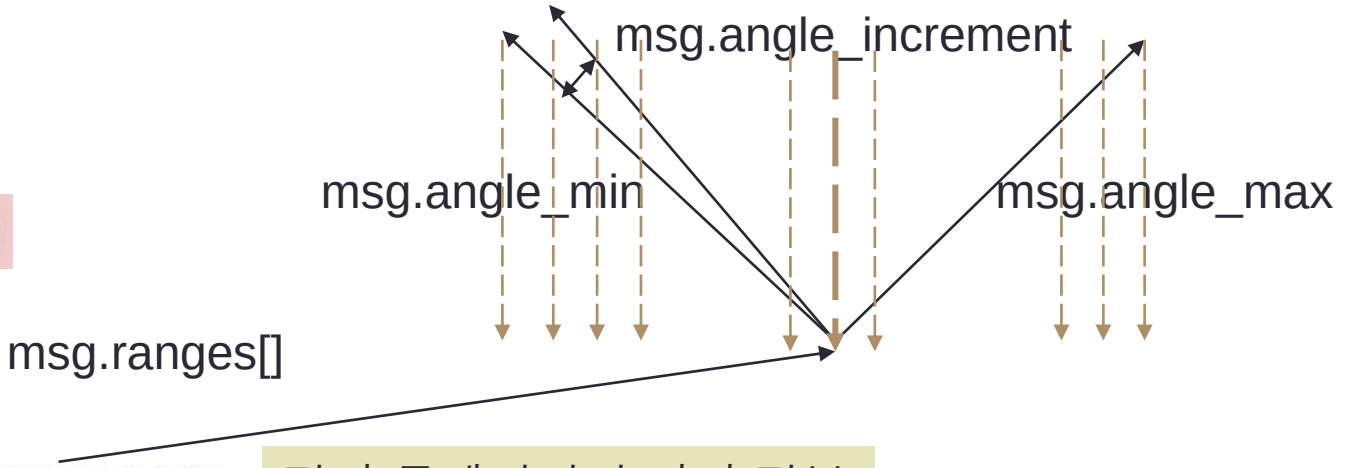
ROS 그래프: red_light_green_light




Ubuntu Desktop
09:51
Old Firefox Data
deu_ros [~/practice/catkin_ws/src/deu_ros] - .../scripts/ch07/red_light_green_light.py - PyCharm
deu_ros scripts ch07 red_light_green_light.py
Add Configuration...
Git:
Project
deu_ros
deu_ros ~/practice/catkin_ws/src/deu_ros
action
launch
msg
scripts
ch03
ch04
ch05
ch07
range_ahead.py
red_light_green_light.py
wander.py
ch08
red_light_green_light.py
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist
cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1) # <1>
rospy.init_node('red_light_green_light')
red_light_twist = Twist() # <2>
green_light_twist = Twist()
green_light_twist.linear.x = 0.5 # <3>
driving_forward = False
light_change_time = rospy.Time.now()
rate = rospy.Rate(10)
while not rospy.is_shutdown():
if driving_forward:
cmd_vel_pub.publish(green_light_twist) # <4>
else:
cmd_vel_pub.publish(red_light_twist)
if rospy.Time.now() > light_change_time: # <5>
driving_forward = not driving_forward
light_change_time = rospy.Time.now() + rospy.Duration(3)
rate.sleep() # <6>
jongmin@ubuntu: ~/practice/catkin_ws/src/wanderbot
File Edit View Search Terminal Tabs Help
jongmin@ubunt...
jongmin@ubunt...
jongmin@ubunt...
roscore http://lo...
jongmin@ubuntu:wanderbot$ roslaunch turtlebot_gazebo turtlebot_world.launch
6: TODO
9: Version Control
Terminal
Python Console
Event Log
15:22 LF UTF-8 4 spaces Git: master Python 2.7

range_ahead.py

```
1  #!/usr/bin/env python
2
3  import math
4
5  import rospy
6  from sensor_msgs.msg import LaserScan
7
8
9  def scan_callback(msg):
10     range_ahead = msg.ranges[len(msg.ranges) / 2]
11     # LJM - added : begin
12     angle_min_in_degree = msg.angle_min * 180.0 / math.pi
13     angle_max_in_degree = msg.angle_max * 180.0 / math.pi
14     angle_inc_in_degree = msg.angle_increment * 180.0 / math.pi
```



정면 물체까지의 거리 정보

최소 각도 (rad → degree)
최대 각도
각도 증가분

```
15      rospy.loginfo('angle_min = %f, angle_max = %f, angle_inc = %f',
16                    angle_min_in_degree, angle_max_in_degree,
17                    angle_inc_in_degree)
18      # LJM - added : end
19      print "range ahead: %0.1f" % range_ahead
20
21
22  rospy.init_node('range_ahead')
23  scan_sub = rospy.Subscriber('scan', LaserScan, scan_callback)
24  rospy.spin()
```

정면 물체까지의 거리 출력

LaserScan 자료형의 /scan 토픽을 사용하는 구독자 객체 생성

LaserScan 메시지

- URL: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html

```
Header header               # timestamp in the header is the acquisition time of
                            # the first ray in the scan.
                            #
                            # in frame frame_id, angles are measured around
                            # the positive Z axis (counterclockwise, if Z is up)
                            # with zero angle being forward along the x axis

float32 angle_min           # start angle of the scan [rad]
float32 angle_max           # end angle of the scan [rad]
float32 angle_increment     # angular distance between measurements [rad]

float32 time_increment      # time between measurements [seconds] - if your scanner
                            # is moving, this will be used in interpolating position
                            # of 3d points
float32 scan_time           # time between scans [seconds]

float32 range_min           # minimum range value [m]
float32 range_max           # maximum range value [m]

float32[] ranges            # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities       # intensity data [device-specific units].  If your
                            # device does not provide intensities, please leave
                            # the array empty.
```

실행 방법

Terminal 1:

$ roscore

Terminal 2:

$ roslaunch turtlebot_gazebo turtlebot_world.launch

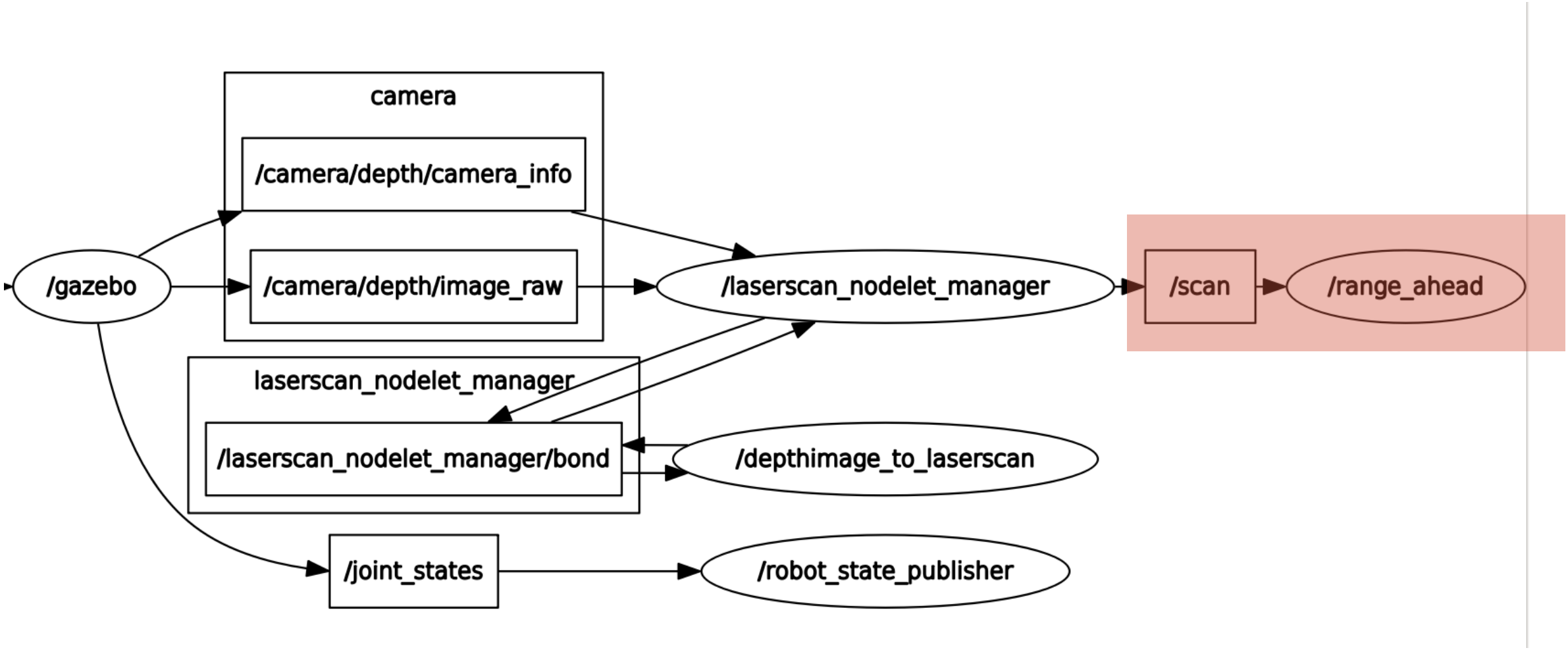
Terminal 3:

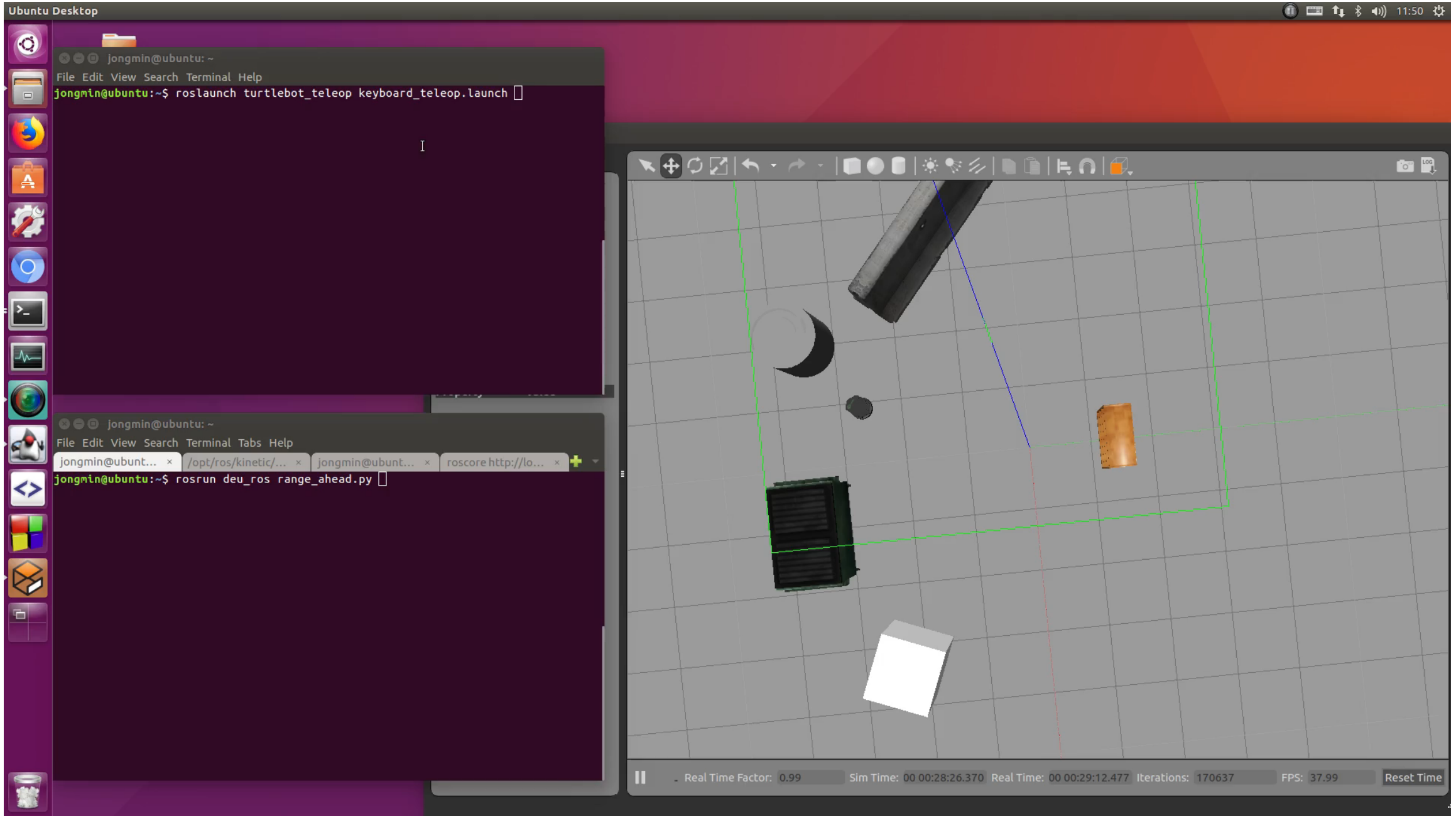
$ roslaunch turtlebot_teleop keyboard_teleop.launch

Terminal 4:

$ rosrun deu_ros range_ahead.py

ROS 그래프: range_ahead




Ubuntu Desktop
11:50
jongmin@ubuntu: ~
File Edit View Search Terminal Help
jongmin@ubuntu:~$ roslaunch turtlebot_teleop keyboard_teleop.launch
jongmin@ubuntu: ~
File Edit View Search Terminal Tabs Help
jongmin@ubunt...
/opt/ros/kinetic/...
jongmin@ubunt...
roscore http://lo...
jongmin@ubuntu:~$ rosrun deu_ros range_ahead.py
Real Time Factor: 0.99
Sim Time: 00 00:28:26.370
Real Time: 00 00:29:12.477
Iterations: 170637
FPS: 37.99
Reset Time

wander.py

```
1  #!/usr/bin/env python
2
3  import rospy
4  from geometry_msgs.msg import Twist
5  from sensor_msgs.msg import LaserScan
6
7
8  def scan_callback(msg):
9      global g_range_ahead
10     g_range_ahead = min(msg.ranges)
11     print 'g_range_ahead = %.1f' % g_range_ahead
12
13
14 g_range_ahead = 1  # anything to start
15 scan_sub = rospy.Subscriber('scan', LaserScan, scan_callback)
16 cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)
17 rospy.init_node('wander')
18 state_change_time = rospy.Time.now()
19 driving_forward = True
20 rate = rospy.Rate(10)
```

로봇과 가장 가까운 물체까지의 거리

LaserScan 자료형의 scan 토픽 구독자 객체 생성
Twist 자료형의 cmd_vel 토픽 발행자 객체 생성

```
22 while not rospy.is_shutdown():
23     if driving_forward:
24         if g_range_ahead < 0.8 or rospy.Time.now() > state_change_time:
25             driving_forward = False
26             state_change_time = rospy.Time.now() + rospy.Duration(5)  # Turn for 5 seconds
27     else:  # we're not driving_forward
28         if rospy.Time.now() > state_change_time:
29             driving_forward = True  # we're done spinning, time to go forwards!
30             state_change_time = rospy.Time.now() + rospy.Duration(7)  # LJM: 3 instead of 30
31     twist = Twist()
32     if driving_forward:
33         twist.linear.x = 0.3  # 1 ==> 0.5
34     else:
35         twist.angular.z = 0.5  # 1 ==> 0.5
36     cmd_vel_pub.publish(twist)
37
38     rate.sleep()
```

직진 중일 경우

가까운 물체까지의 거리가 0.8m이내이거나 상태 변화 시간이 지났으면

선속도(linear.x)를 0.3으로 설정

각속도(angular.z)를 0.5로 설정

twist 객체를 사용하여 cmd_vel 토픽 발행

실행 방법

Terminal 1:

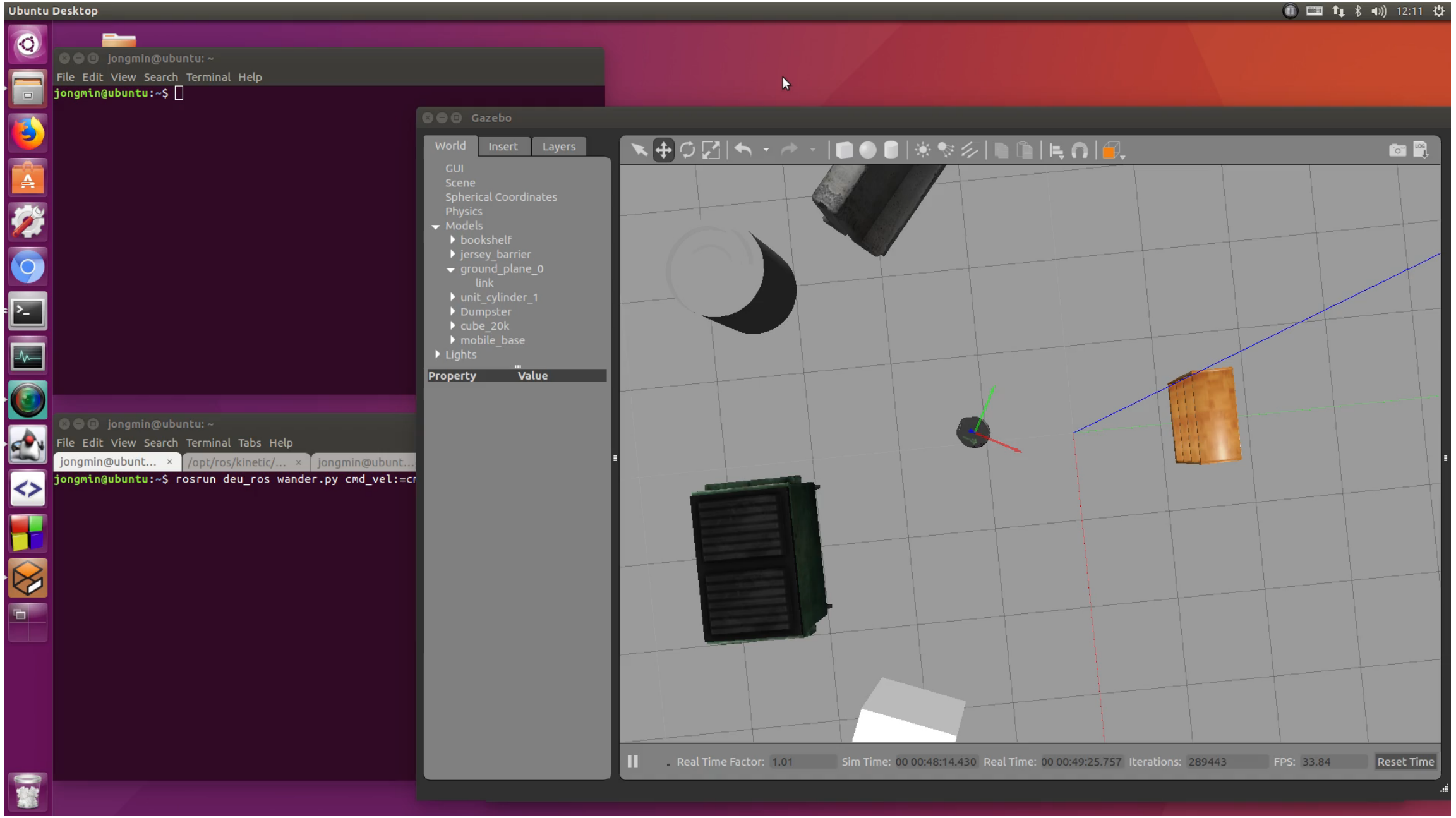
$ roscore

Terminal 2:

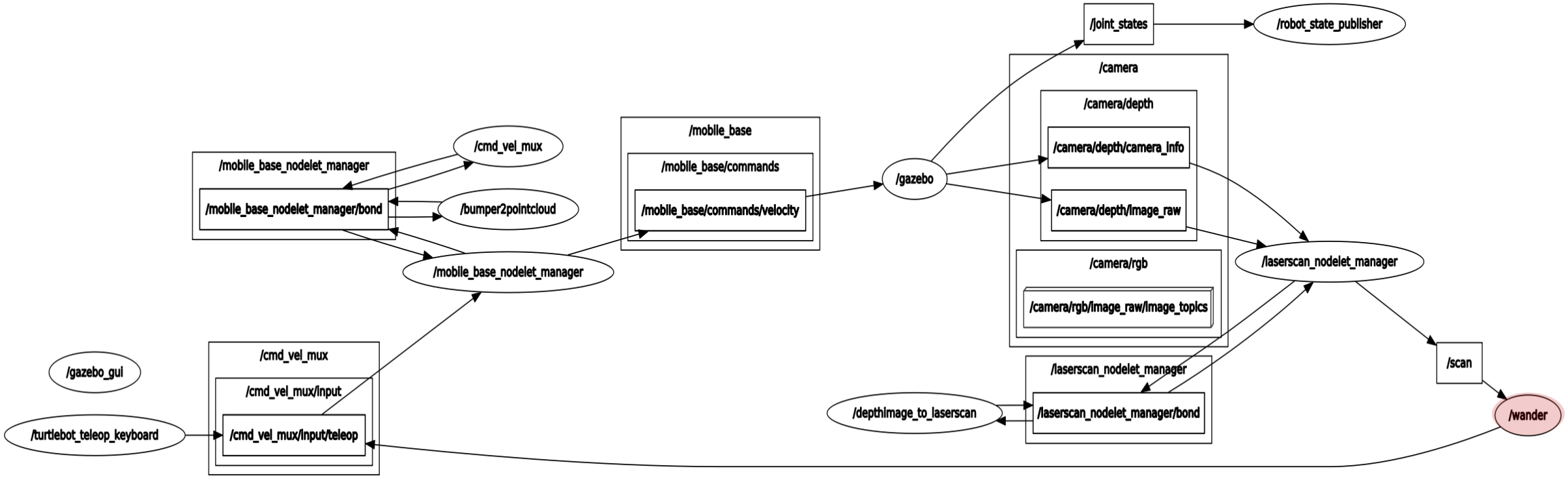
$ roslaunch turtlebot_gazebo turtlebot_world.launch

Terminal 3:

$ rosrun deu_ros wander.py **cmd_vel:=cmd_vel_mux/input/teleop**


Ubuntu Desktop
12:11
jongmin@ubuntu: ~
File Edit View Search Terminal Help
jongmin@ubuntu:~$
jongmin@ubuntu: ~
File Edit View Search Terminal Tabs Help
jongmin@ubunt...
/opt/ros/kinetic/...
jongmin@ubunt...
jongmin@ubuntu:~$ rosrun deu_ros wander.py cmd_vel:=cr
Gazebo
World
Insert
Layers
GUI
Scene
Spherical Coordinates
Physics
Models
bookshelf
jersey_barrier
ground_plane_0
link
unit_cylinder_1
Dumpster
cube_20k
mobile_base
Lights
Property
Value
Real Time Factor: 1.01
Sim Time: 00 00:48:14.430
Real Time: 00 00:49:25.757
Iterations: 289443
FPS: 33.84
Reset Time

rqt_graph 실행 결과



요약

- 로봇 이동 방법
  - geometry_msgs/Twist 메시지
  - /cmd_vel 토픽
- 거리 정보 획득
  - sensor_msgs/LaserScan 메시지
  - /scan 토픽
- 원더-봇
  - /scan 토픽을 청취하여
  - /cmd_vel 토픽을 발행