

TurtleBot3

ROBOTIS

Open Source Team

Yoonseok Pyo



온라인강좌

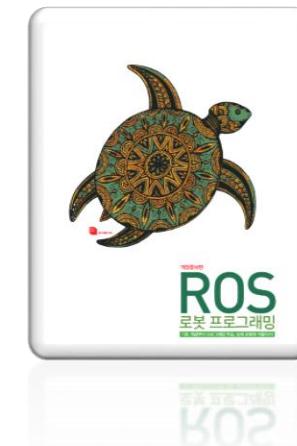


Subscribe

교재
Chapter
9, 10, 11, 12

Contents

1. ROS 소개
2. ROS 개발환경 구축
3. ROS 공식 로봇 플랫폼 TurtleBot
4. 모바일 로봇 개발을 위한 임베디드 프로그래밍 기초 실습: 입출력 및 IMU 센서
5. 모바일 로봇 개발을 위한 임베디드 프로그래밍 실전 실습: 데드 레커닝
6. 모바일 로봇 개발을 위한 원격제어 프로그래밍 실습
7. 시뮬레이터 Gazebo를 사용하기 위한 모바일 로봇 모델링
8. 시뮬레이터 Gazebo를 사용하기 위한 가상 로봇 프로그래밍
9. 시뮬레이터 Gazebo를 활용한 SLAM 프로그래밍
10. 시뮬레이터 Gazebo를 활용한 Navigation 프로그래밍
11. 실제 로봇을 활용한 SLAM 실습1
12. 실제 로봇을 활용한 SLAM 실습2
13. 실제 로봇을 활용한 Navigation 실습1
14. 실제 로봇을 활용한 Navigation 실습2
15. 위치기반 정보를 이용한 모바일 매니퓰레이터로서의 사용법
16. SLAM과 Navigation 응용 사례



[온라인강좌](#)



[Subscribe](#)



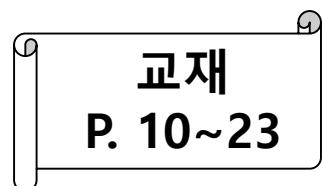
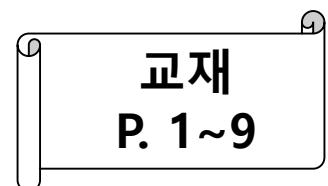
‘온 강의의 목적

“터틀넷3를 참고 삼아
자신만의 모나토일로봇 만들기”

ROS 소개

01_로봇_소프트웨어_플랫폼.pdf

02_로봇_운영체제_ROS.pdf



ROS 란?

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

<http://www.ros.org/wiki/>



ROS 란?

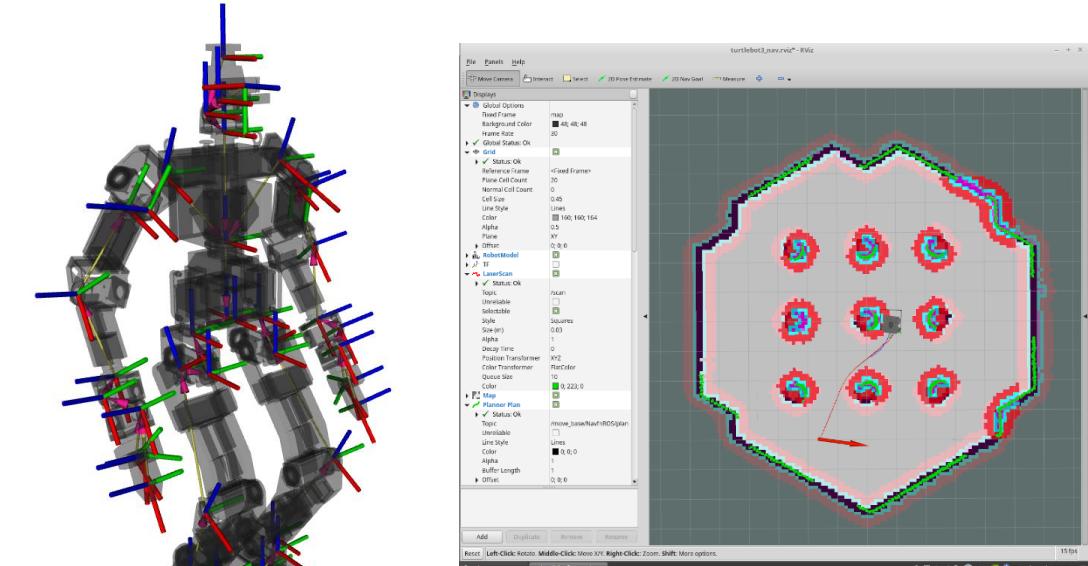
Client Layer	roscpp	rospy	roslisp	rosjava	roslibjs		
Robotics Application	MoveIt!	navigatioin	executive smach	descartes	rospeex		
	teleop pkgs	rocon	mapviz	people	ar track		
Robotics Application Framework	dynamic reconfigure	robot localization	robot pose ekf	Industrial core	robot web tools	ros realtime	mavros
	tf	robot state publisher	robot model	ros control	calibration	octomap mapping	
	vision opencv	image pipeline	laser pipeline	perception pcl	laser filters	ecto	
Communication Layer	common msgs	rosbag	actionlib	pluginlib	rostopic	rosservice	
	rosnode	roslaunch	rosparam	rosmaster	rosout	ros console	
Hardware Interface Layer	camera drivers	GPS/IMU drivers	joystick drivers	range finder drivers	3d sensor drivers	diagnostics	
	audio common	force/torque sensor drivers	power supply drivers	rosserial	ethercat drivers	ros canopen	
Software Development Tools	RViz	rqt	wstool	rospack	catkin	rosdep	
Simulation	gazebo ros pkgs	stage ros					

특징 1) 통신 인프라

- 노드 간 데이터 통신을 제공
- 통상적 미들웨어로 지칭되는 메시지 전달 인터페이스 지원
- 메시지 파싱 기능
 - 로봇 개발 시에 빈번히 사용되는 통신 시스템 제공
 - 캡슐화 및 코드 재사용을 촉진하는 노드들 간의 메시지 전달 인터페이스
- 메시지의 기록 및 재생
 - 노드 간 송/수신되는 데이터인 메시지를 저장하고 필요시에 재사용 가능
 - 저장된 메시지를 기반으로 반복적인 실험 가능, 알고리즘 개발에 용이함
- 메시지 사용으로 인한 다양한 프로그래밍 언어 사용 가능
 - 노드 간의 데이터 교환이 메시지를 사용하기 때문에 각 노드는 서로 다른 언어로 작성 가능
 - 클라이언트 라이브러리: roscpp, rospy, roslib, rosjava, roslua, rosccs, roseus, PhaROS, rosR
- 분산 매개 변수 시스템
 - 시스템에서 사용되는 변수를 글로벌 키값으로 작성하여 공유 및 수정하여 실시간으로 반영

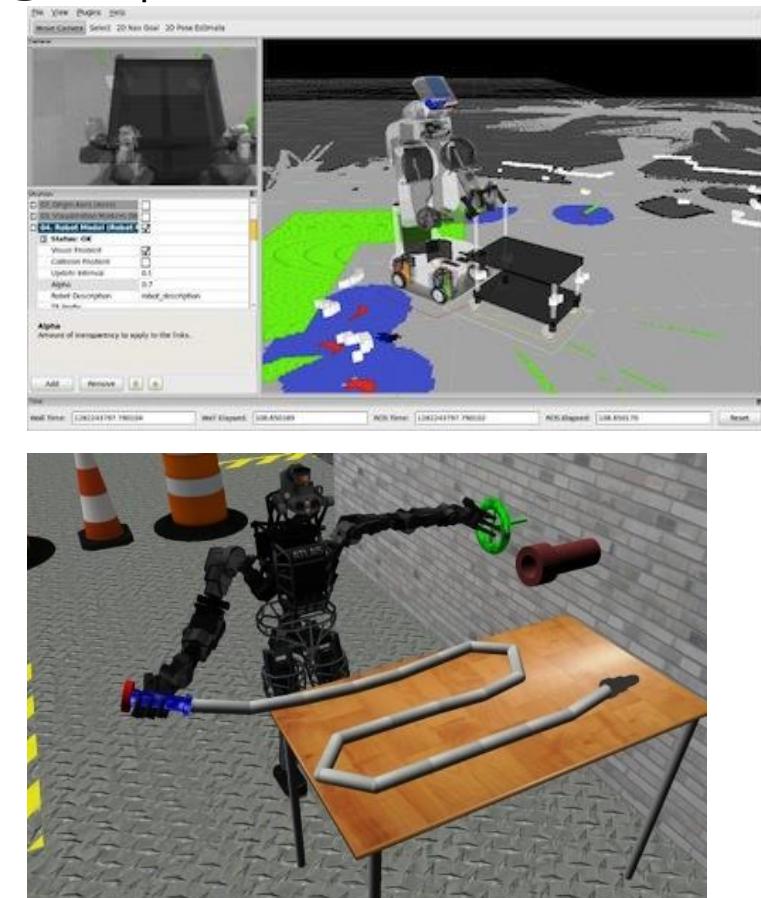
특징 2) 로봇 관련 다양한 기능

- **로봇에 대한 표준 메시지 정의**
 - 카메라, IMU, 레이저 등의 센서 / 오도메트리, 경로 및 지도 등의 내비게이션 데이터 등의 표준 메시지를 정의하여 모듈화, 협업 작업을 유도, 효율성 향상
- **로봇 기하학 라이브러리**
 - 로봇, 센서 등의 상대적 좌표를 트리화 시키는 TF 제공
- **로봇 기술 언어**
 - 로봇의 물리적 특성을 설명하는 XML 문서 기술
- **진단 시스템**
 - 로봇의 상태를 한눈에 파악할 수 있는 진단 시스템 제공
- **센싱/인식**
 - 센서 드라이버, 센싱/인식 레벨의 라이브러리 제공
- **내비게이션**
 - 로봇에서 많이 사용되는 로봇의 포즈(위치/자세) 추정, 지도내의 자기 위치 추정 제공
 - 지도 작성에 필요한 SLAM, 작성된 지도 내에서 목적지를 찾아가는 Navigation 라이브러리를 제공
- **매니퓰레이션**
 - 로봇 암에 사용되는 IK, FK 는 물론 응용단의 Pick and Place 를 지원하는 다양한 Manipulation 라이브러리 제공
 - GUI 형태의 매니퓰레이션 Tools 제공(MoveIt!)



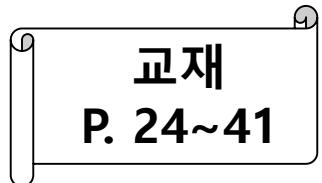
특징 3) 다양한 개발 도구

- 로봇 개발에 필요한 다양한 개발 도구를 제공
- 로봇 개발의 효율성 향상
- **Command-Line Tools**
 - GUI 없이 ROS에서 제공되는 명령어로만 로봇 억세스 및 거의 모든 ROS 기능 소화
- **RViz**
 - 강력한 3D 시각화툴 제공
 - 레이저, 카메라 등의 센서 데이터를 시각화
 - 로봇 외형과 계획된 동작을 표현
- **RQT**
 - 그래픽 인터페이스 개발을 위한 Qt 기반 프레임 워크 제공
 - 노드와 그들 사이의 연결 정보 표시(rqt_graph)
 - 인코더, 전압, 또는 시간이 지남에 따라 변화하는 숫자를 플로팅(rqt_plot)
 - 데이터를 메시지 형태로 기록하고 재생(rqt_bag)
- **Gazebo**
 - 물리 엔진을 탑재, 로봇, 센서, 환경 모델 등을 지원, 3차원 시뮬레이터
 - ROS와의 높은 호환성



ROS 개발환경 구축

03_ROS_개발환경_구축.Pdf



ROS에서 사용 가능한 통합개발환경(IDE)

- <http://wiki.ros.org/IDEs>
- 추천1순위: **Qtcreator** + [Qt Creator Plugin for ROS](#)
 - 설치: sudo apt-get install qtcreator
 - 장점: CmakeLists.txt 을 그대로 사용 가능, rqt 플러그인 및 GUI 개발하기 쉬움
- 추천2순위: **Visual Studio Code** + [ROS Extension](#)
 - 설치: <https://code.visualstudio.com/>
 - 장점: 간단한 텍스트 편집기 지향, 빠름
 - 비슷한 계열로 Atom, Sublime Text, Clion 등도 좋음
- 추천3순위: **Eclipse**
 - 설치: <http://www.eclipse.org/>
 - 장점: 많은 사람들이 사용하는 익숙한 통합개발환경 (단, 무거움)

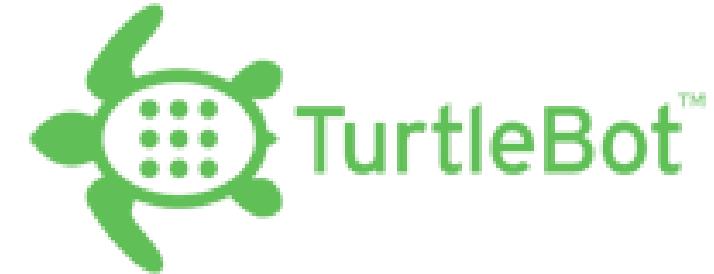
ROS 공식 키트 플랫폼 TurtleBot

10_로봇설계.pdf

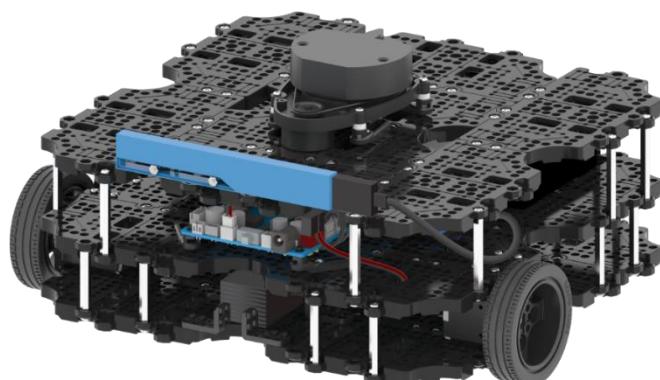
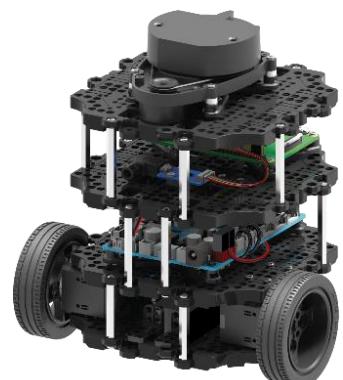


TurtleBot

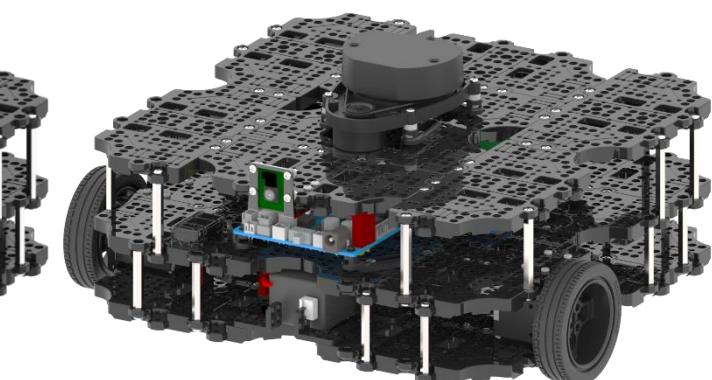
- ROS 공식 로봇 플랫폼
- 전 세계 수 많은 연구소, 학교, DIY 에서 사용 중
- SLAM, Navigation, Gazebo, RViz 서포트!
 - <http://www.turtlebot.com/>
 - <http://wiki.ros.org/Robots/TurtleBot>
 - <http://turtlebot3.robotis.com>



Burger

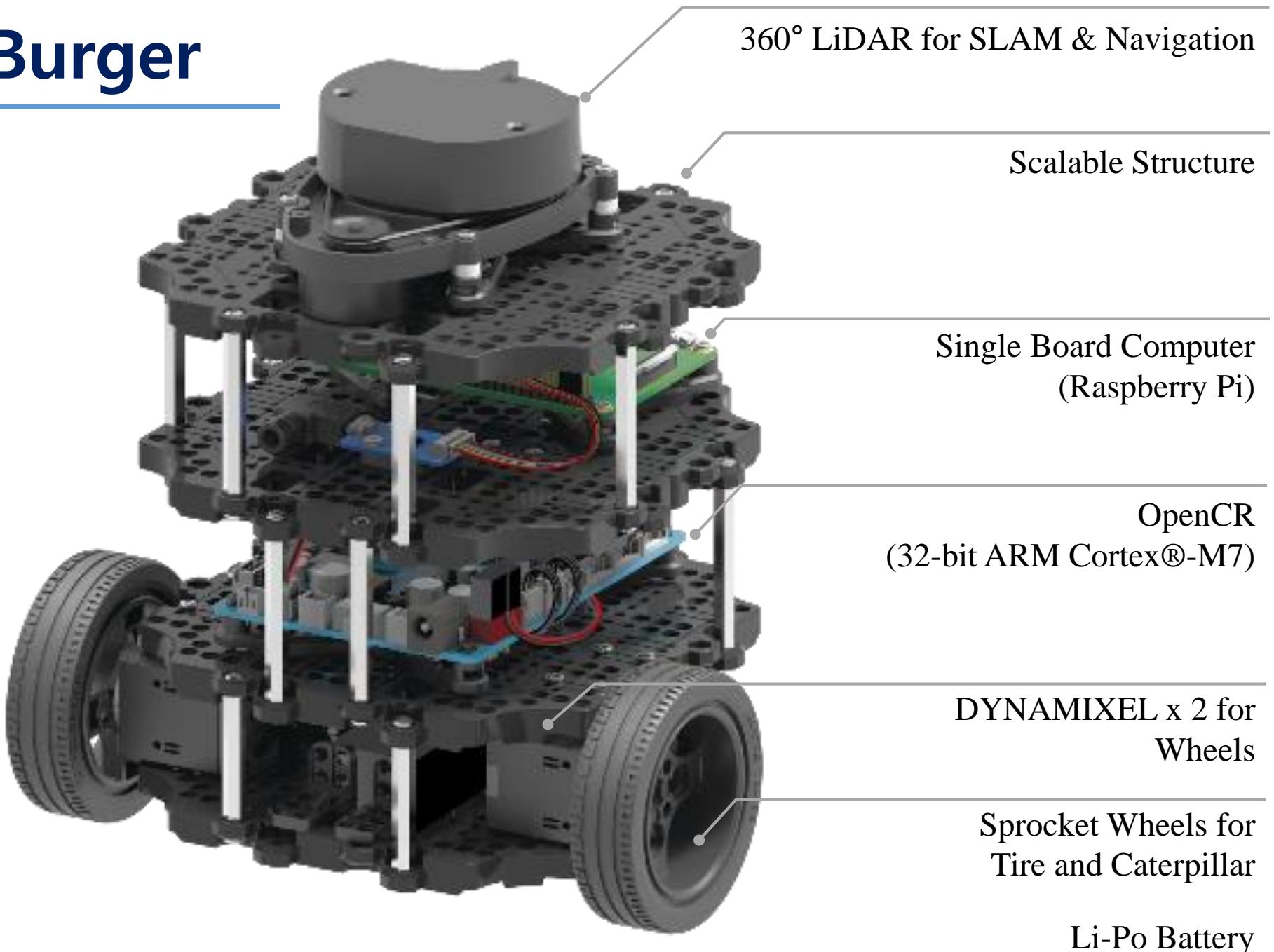


Waffle



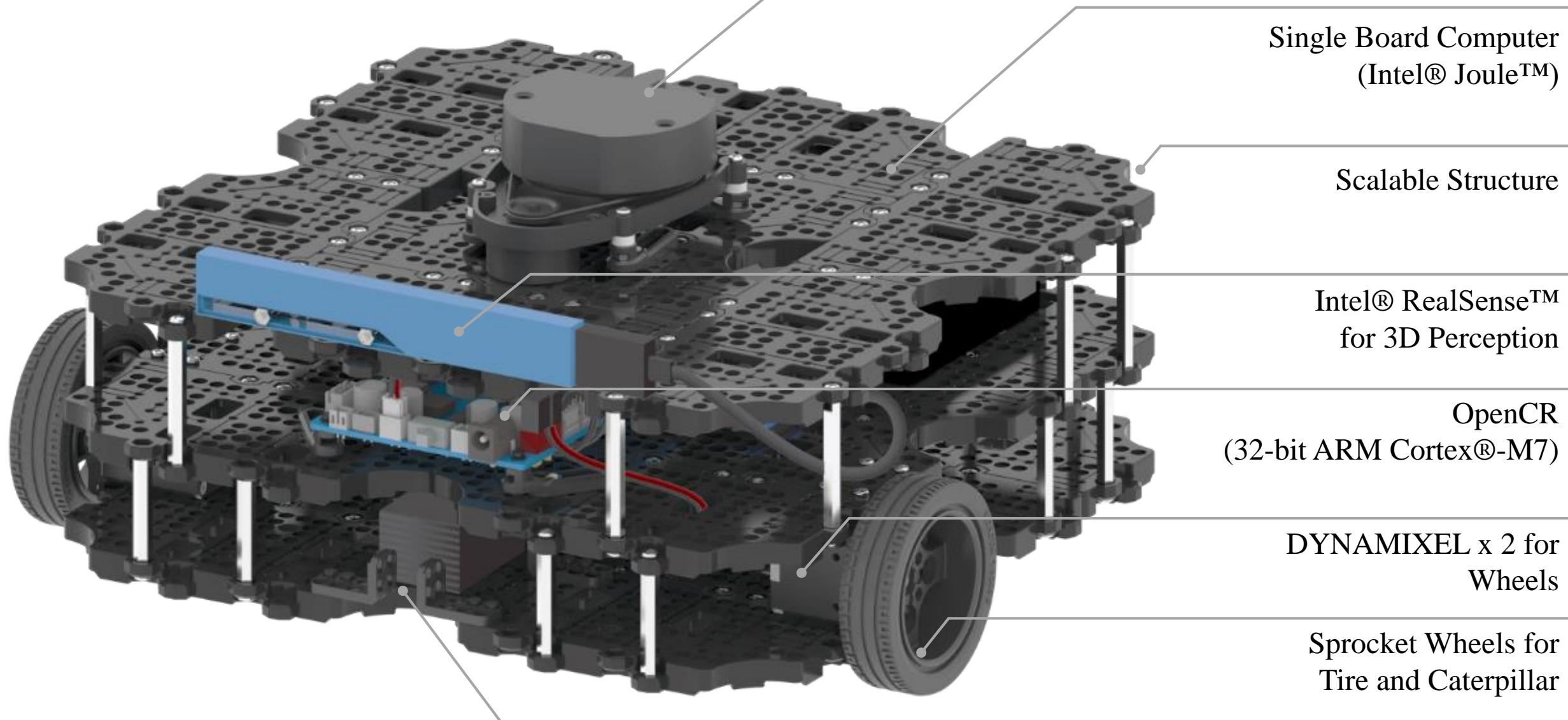
Waffle Pi

TurtleBot3 Burger

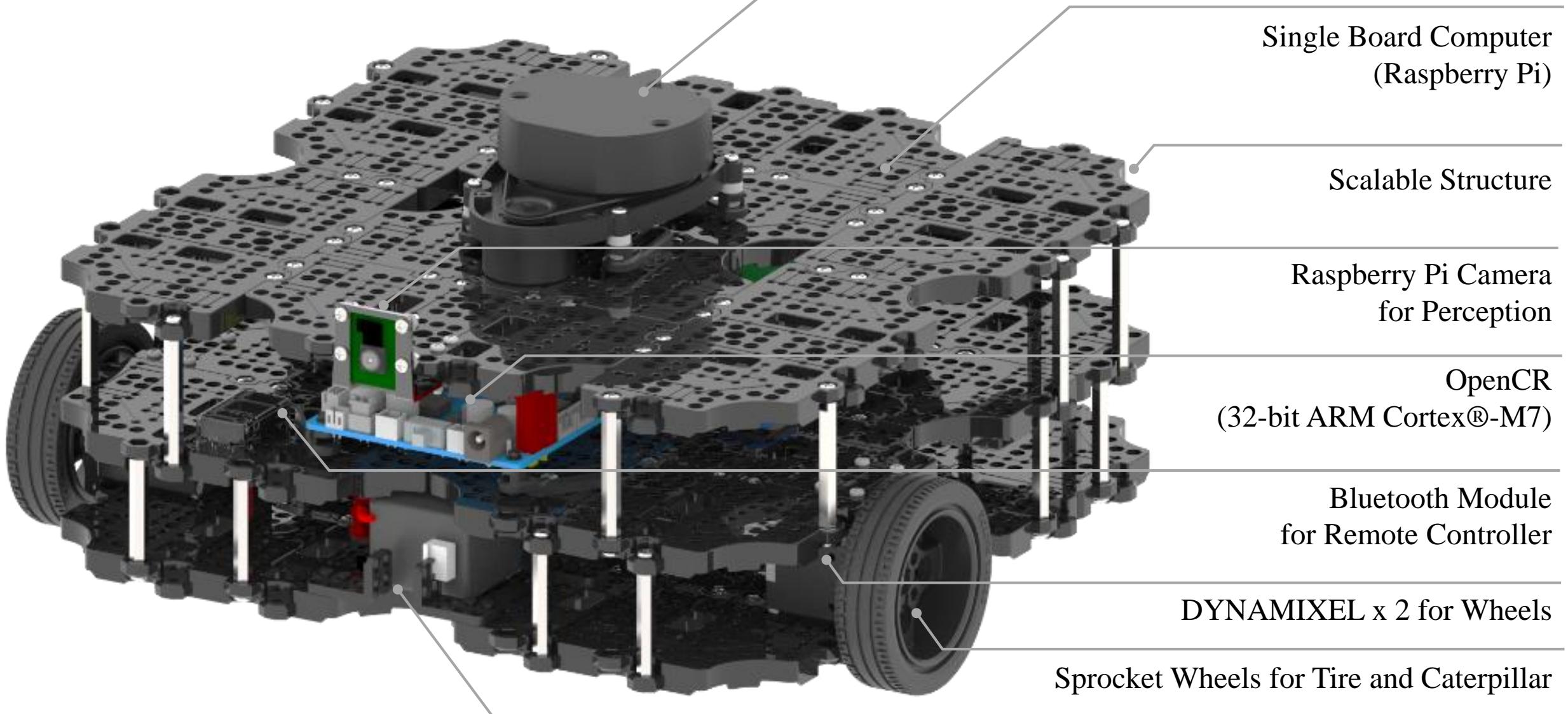


TurtleBot3 Waffle

360° LiDAR for SLAM & Navigation

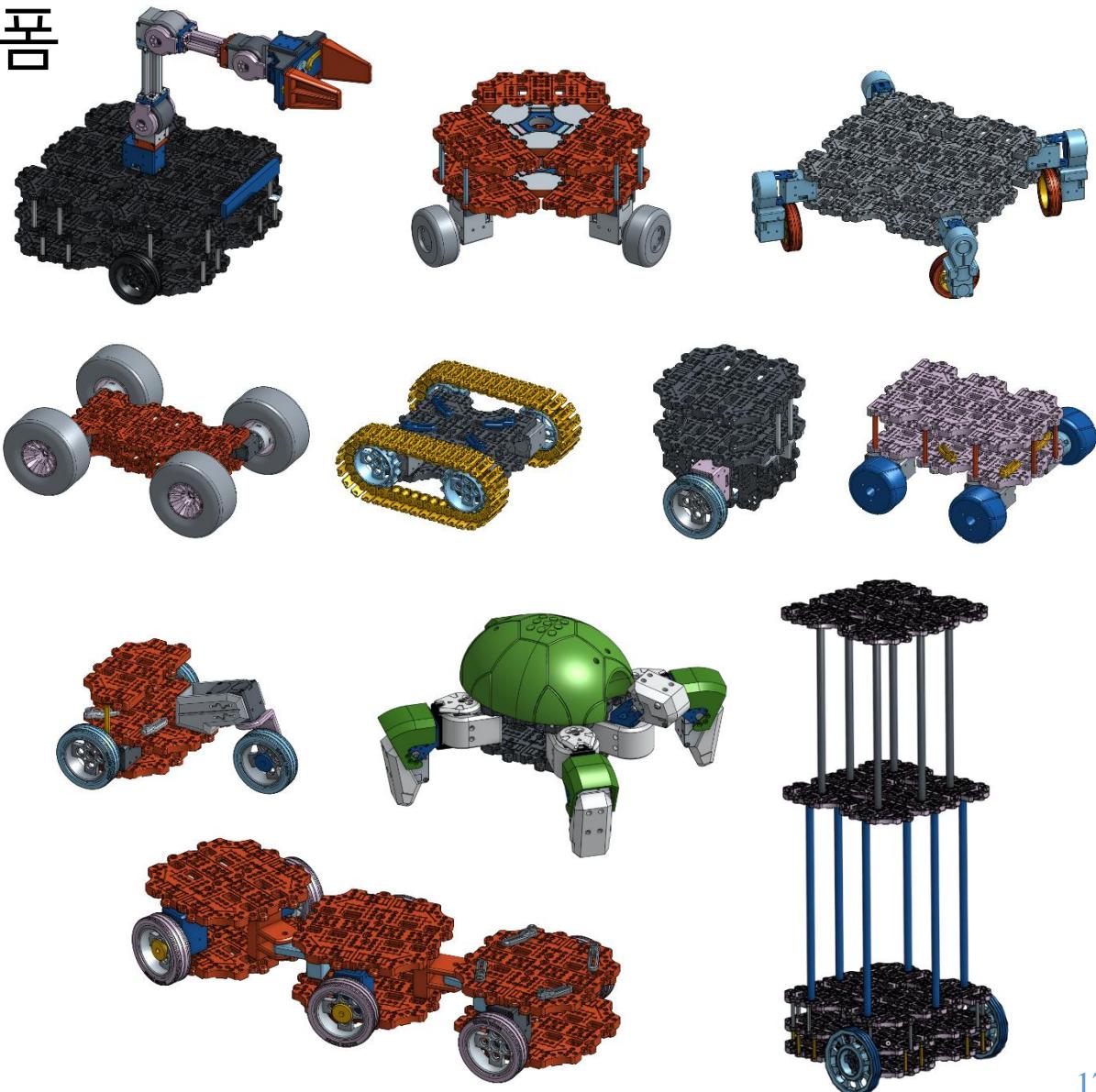
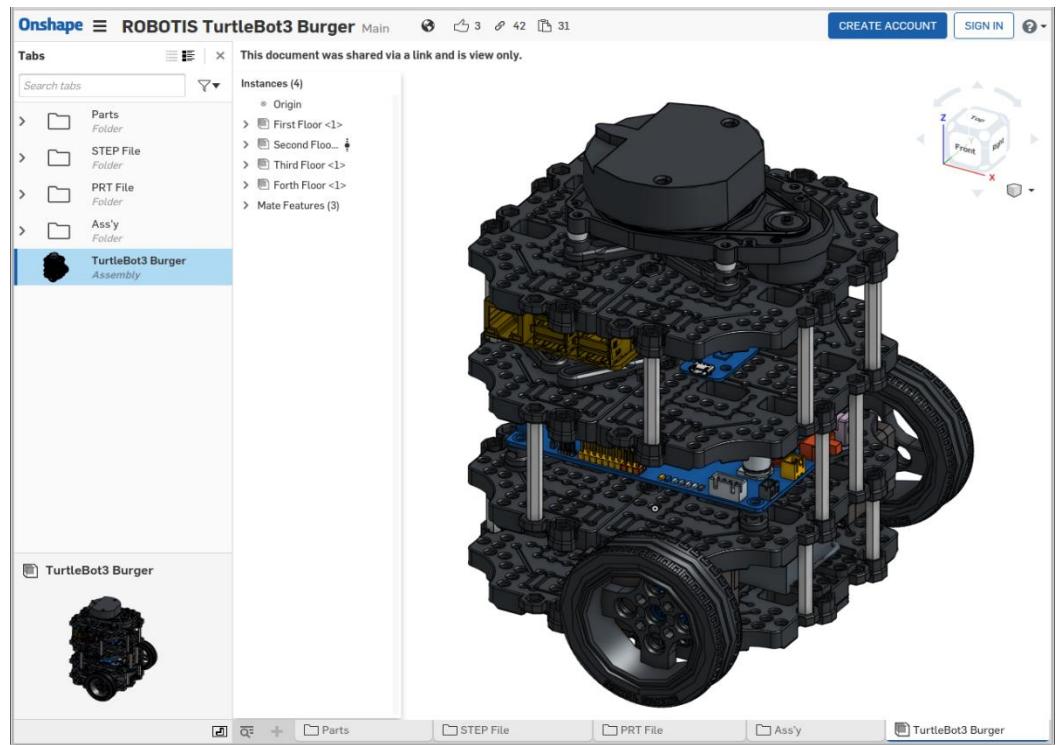


TurtleBot3 Waffle Pi



TurtleBot3 하드웨어 (오픈 하드웨어)

- 오픈 소스 하드웨어 기반 로봇 플랫폼
- 웹 브라우저에서 실행 ([Onshape](#))
- 3D 프린터로 출력 가능

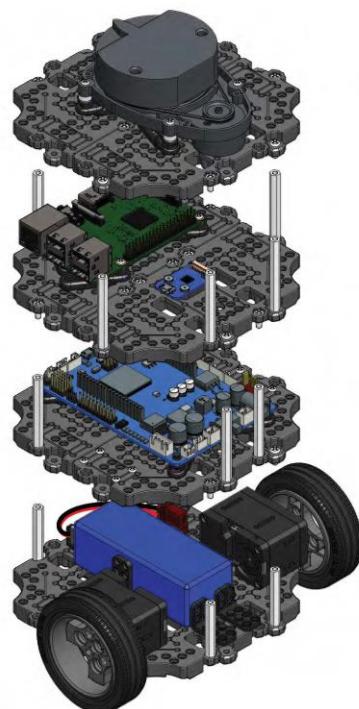


TurtleBot3 하드웨어 (오픈 하드웨어)

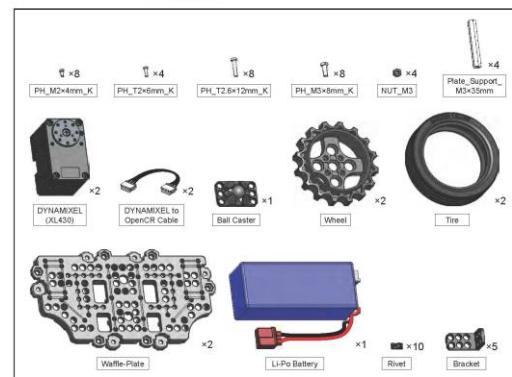
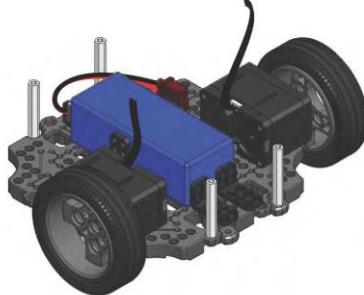
- TurtleBot3 Burger: <http://www.robotis.com/service/download.php?no=676>
- TurtleBot3 Waffle: <http://www.robotis.com/service/download.php?no=677>
- TurtleBot3 Waffle Pi: <http://www.robotis.com/service/download.php?no=678>
- TurtleBot3 Friends OpenManipulator Chain: <http://www.robotis.com/service/download.php?no=679>
- TurtleBot3 Friends Segway: <http://www.robotis.com/service/download.php?no=680>
- TurtleBot3 Friends Conveyor: <http://www.robotis.com/service/download.php?no=681>
- TurtleBot3 Friends Monster: <http://www.robotis.com/service/download.php?no=682>
- TurtleBot3 Friends Tank: <http://www.robotis.com/service/download.php?no=683>
- TurtleBot3 Friends Omni: <http://www.robotis.com/service/download.php?no=684>
- TurtleBot3 Friends Mecanum: <http://www.robotis.com/service/download.php?no=685>
- TurtleBot3 Friends Bike: <http://www.robotis.com/service/download.php?no=686>
- TurtleBot3 Friends Road Train: <http://www.robotis.com/service/download.php?no=687>
- TurtleBot3 Friends Real TurtleBot: <http://www.robotis.com/service/download.php?no=688>
- TurtleBot3 Friends Carrier: <http://www.robotis.com/service/download.php?no=689>

TurtleBot3 Assembly

- TurtleBot3 조립 방법
- http://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/#assembly-manual



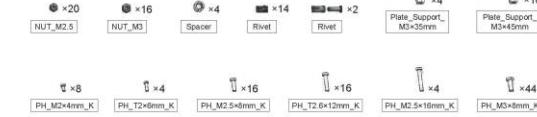
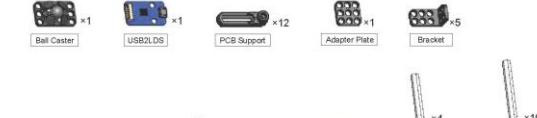
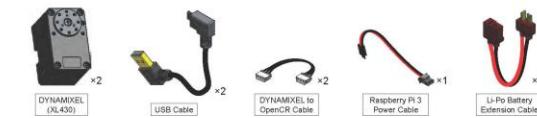
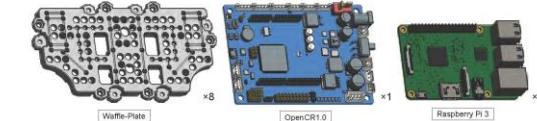
First Layer Assembly



11



Parts List



29

TurtleBot3 소프트웨어

- 오픈 소스 하드웨어 기반 로봇 플랫폼
 - Github에 모든 소프트웨어 공개
-
- https://github.com/ROBOTIS-GIT/robotis_tools → 3장
 - https://github.com/ROBOTIS-GIT/ros_tutorials → 4장, 7장, 13장
 - <https://github.com/ROBOTIS-GIT/DynamixelSDK> → 8장, 10장
 - <https://github.com/ROBOTIS-GIT/dynamixel-workbench> → 8장, 13장
 - <https://github.com/ROBOTIS-GIT/dynamixel-workbench-msgs> → 8장, 13장
 - https://github.com/ROBOTIS-GIT/hls_ifcd_ids_driver → 8장, 10장, 11장
 - <https://github.com/ROBOTIS-GIT/OpenCR> → 9장, 12장
 - <https://github.com/ROBOTIS-GIT/turtlebot3> → 10장, 11장
 - https://github.com/ROBOTIS-GIT/turtlebot3_msgs → 10장, 11장
 - https://github.com/ROBOTIS-GIT/turtlebot3_simulations → 10장, 11장
 - https://github.com/ROBOTIS-GIT/turtlebot3_applications → 10장, 11장
 - https://github.com/ROBOTIS-GIT/turtlebot3_delivery → 12장
 - https://github.com/ROBOTIS-GIT/open_manipulator → 13장

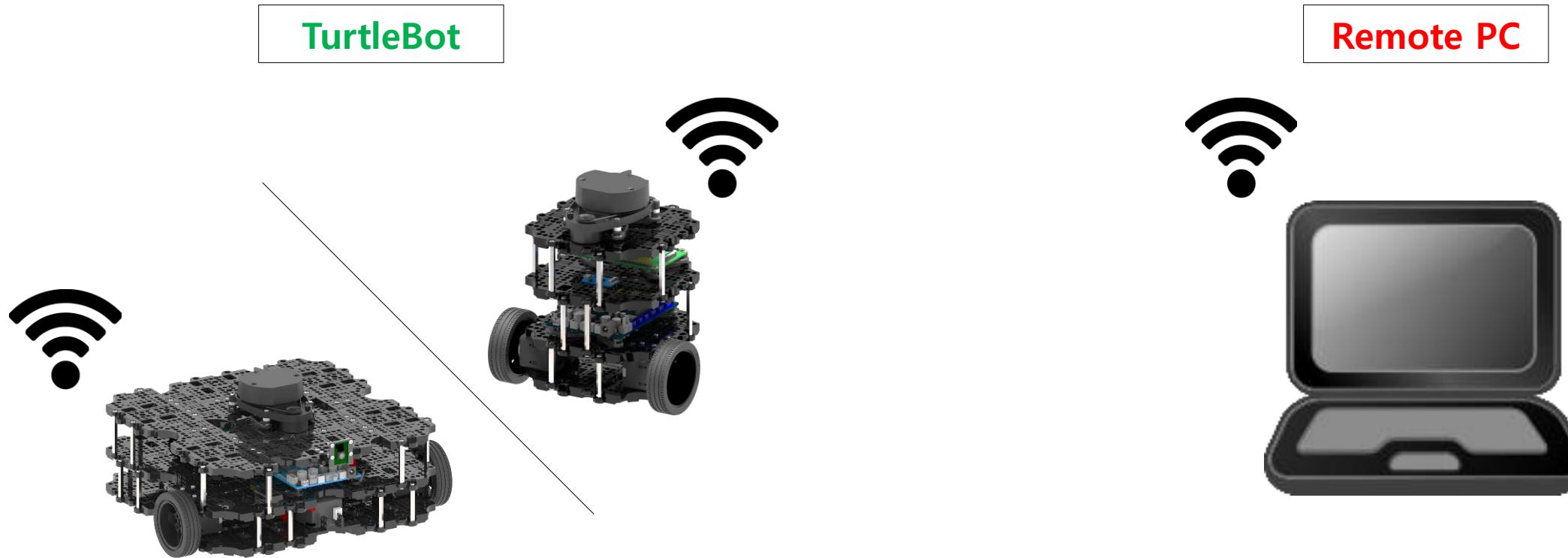
TurtleBot3 개발환경 (소프트웨어)

- 공식 터틀봇3 위키 참조
 - <http://turtlebot3.robotis.com>
- 기본 설치 패키지 (SLAM, Navigation 실습 때 사용 / Gazebo)

```
$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation
```

```
$ cd ~/catkin_ws/src/  
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git  
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git  
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git  
$ cd ~/catkin_ws && catkin_make
```

TurtleBot3 개발환경 (네트워크)



ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311

ROS_HOSTNAME = **IP_OF_TURTLEBOT**

ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311

ROS_HOSTNAME = **IP_OF_REMOTE_PC**

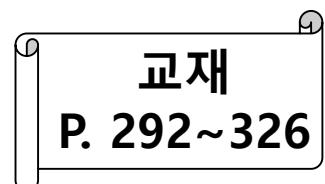
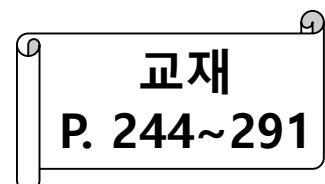
* ROS Master 를 Remote PC에서 구동했을 때의 예제

로봇을 위한 임베디드 프로그래밍 기초 실습

09_임베디드_시스템.pdf

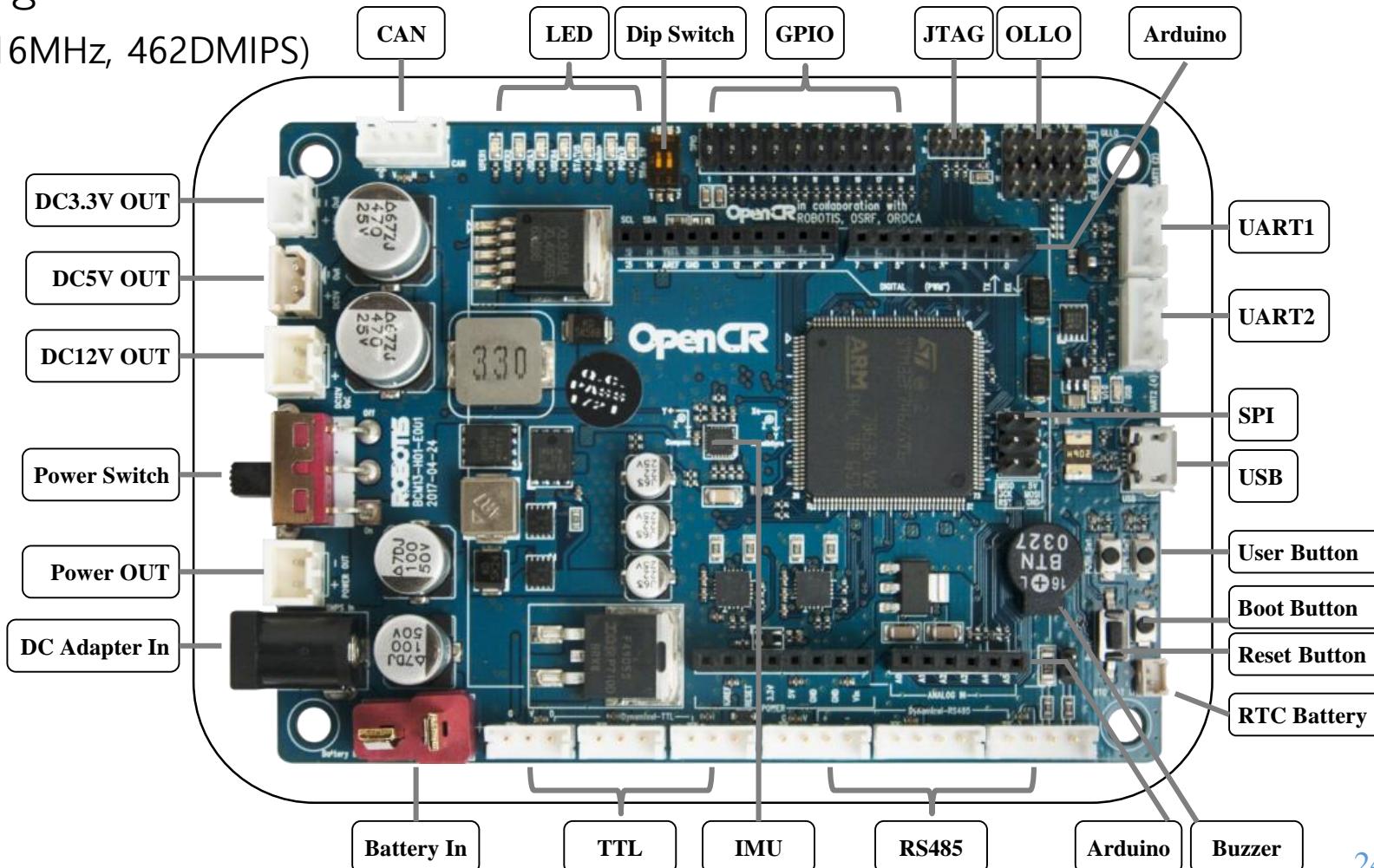
10_로봇_로봇.pdf

11_SLAM과_내비게이션.pdf



OpenCR (Open-source Control Module for ROS)

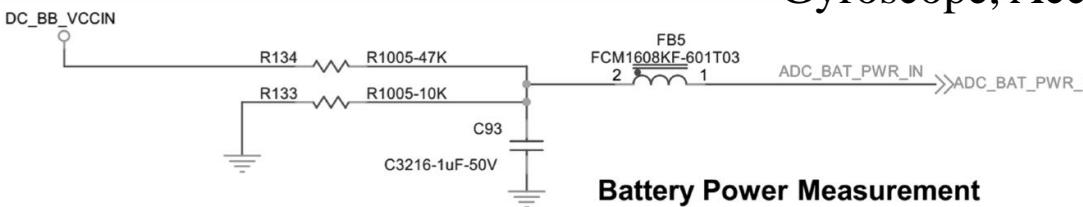
- ROS를 지원하는 임베디드 보드이며 터틀봇3에서 메인 제어기로 사용된다.
- 오픈소스 H/W, S/W : 회로, BOM, 거버 데이터 등의 H/W 정보 및 OpenCR의 모든 S/W를 오픈소스로 공개
- **rosserial의 제약 사항을 극복하기 위한 구성**
 - 32-bit ARM Cortex-M7 with FPU (216MHz, 462DMIPS)
 - 1MB 플래시 메모리
 - 320KB SRAM
 - Float64 지원
 - UART가 아닌 USB 패킷 전송 사용
- SBC 계열의 컴퓨터와 다양한 센서와 함께 사용하기 위한 **전원 설계**
 - 12V@1A, 5V@4A, 3.3V@800mA
- **확장 포트**
 - 32 pins(L 14, R 18)
*Arduino connectivity
 - OLLO Sensor module x 4 pins
 - Extension connector x 18 pins



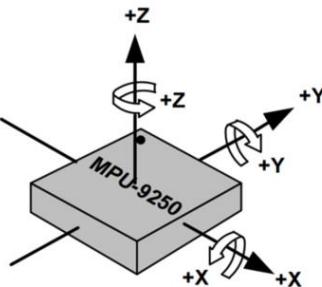
기본 장착 센서 및 통신 지원

- 기본 장착 센서

- Gyroscope 3Axis
- Accelerometer 3Axis
- Magnetometer 3Axis
- 전압 측정 회로



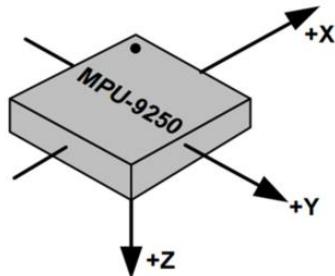
Gyroscope, Accelerometer



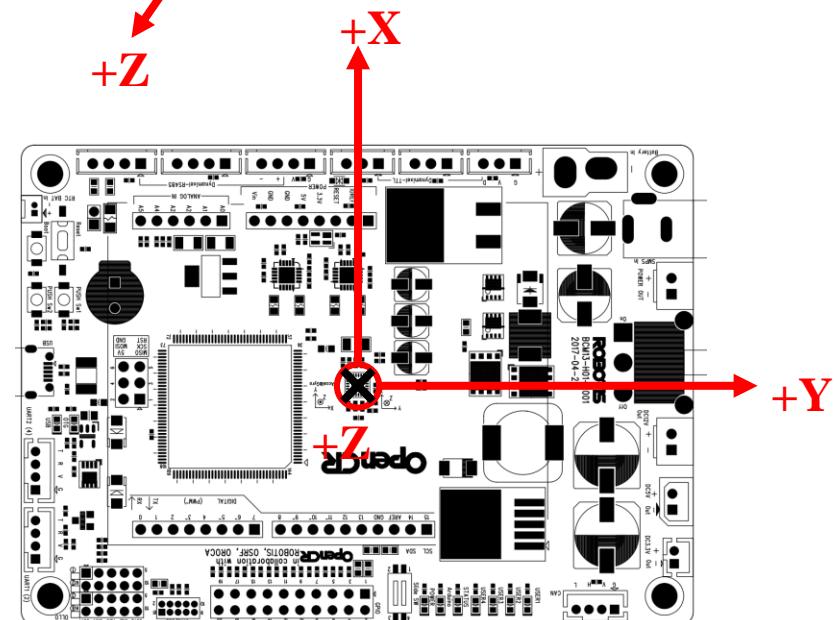
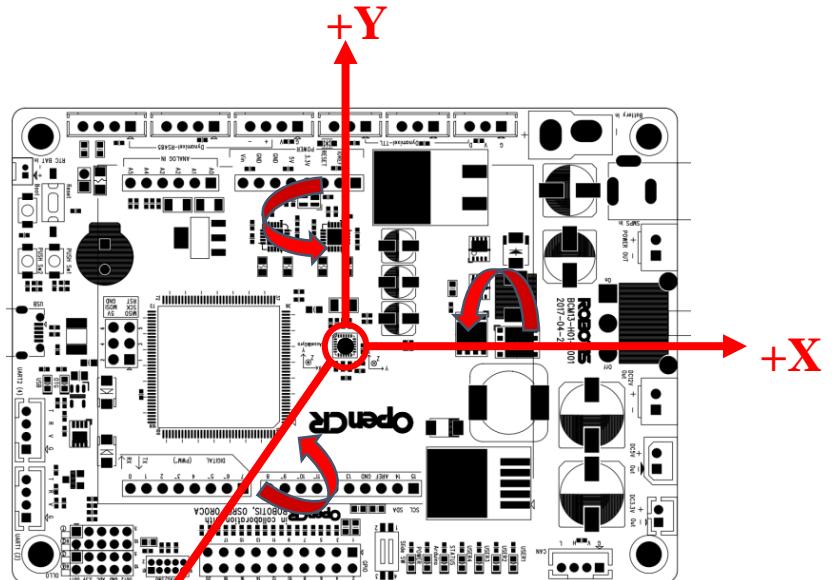
Battery Power Measurement

- 통신 지원

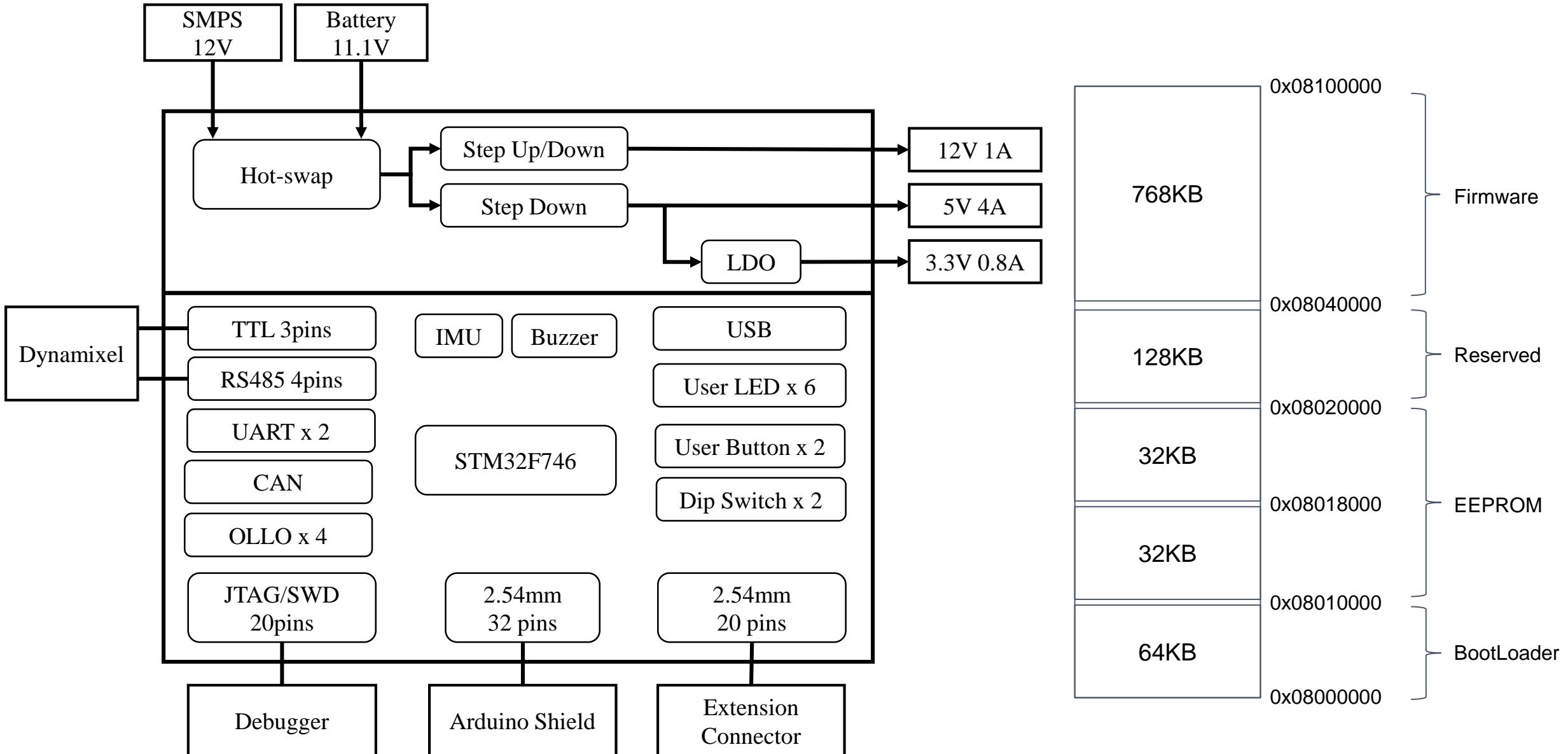
- USB, SPI, I2C
- TTL, RS485, CAN



Magnetometer



블록 다이어그램 및 플래시 메모리 맵



모바일 로봇 개발을 위한 임베디드 프로그래밍 기초 실습

- **입출력**

- rostopic pub을 이용하여 led_out에 값을 입력하여 LED를 제어하기
- rqt에서 topic 을 퍼블리쉬하여 LED를 제어하기
- sound 라는 토픽을 받아 멜로디 출력해보기

- **IMU 센서**

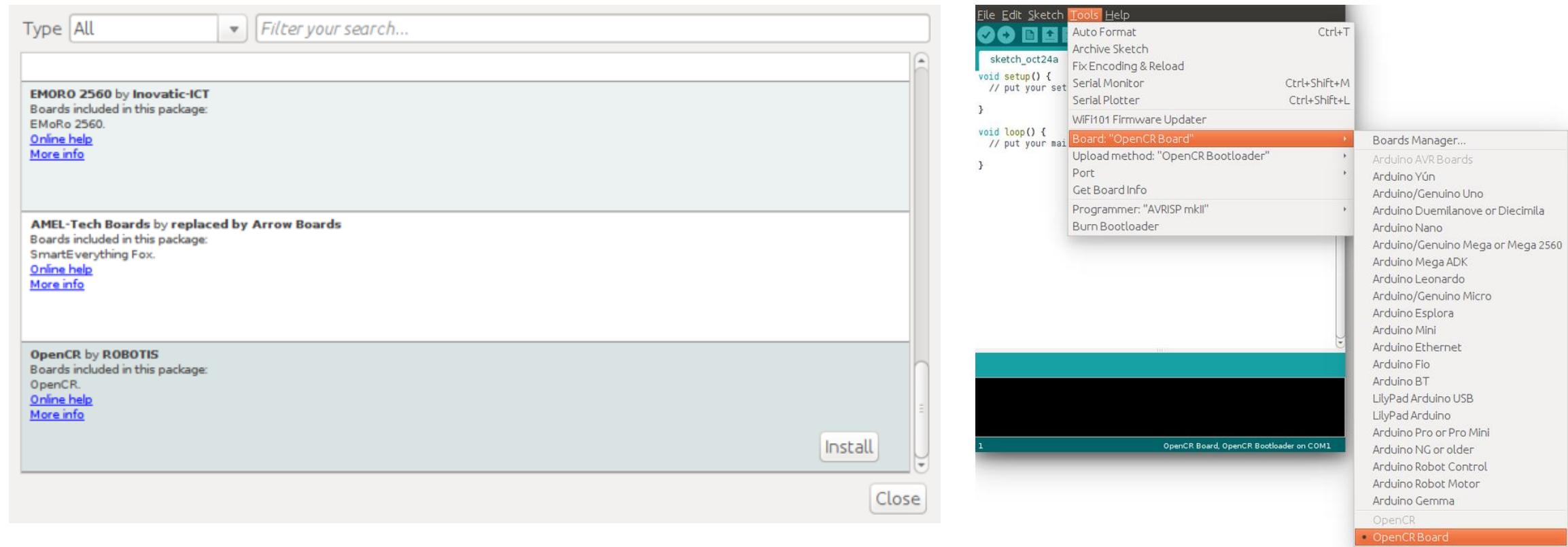
- imu 토픽을 RViz에서 확인하기
- imu 토픽을 rqt에서 확인하기
- imu 토픽을 스마트폰에서 확인하기

- **데드 레커닝**

- 터틀봇3의 데드 레커닝 소스 코드 분석
- 자신만의 로봇에 적용하기

개발환경 구축

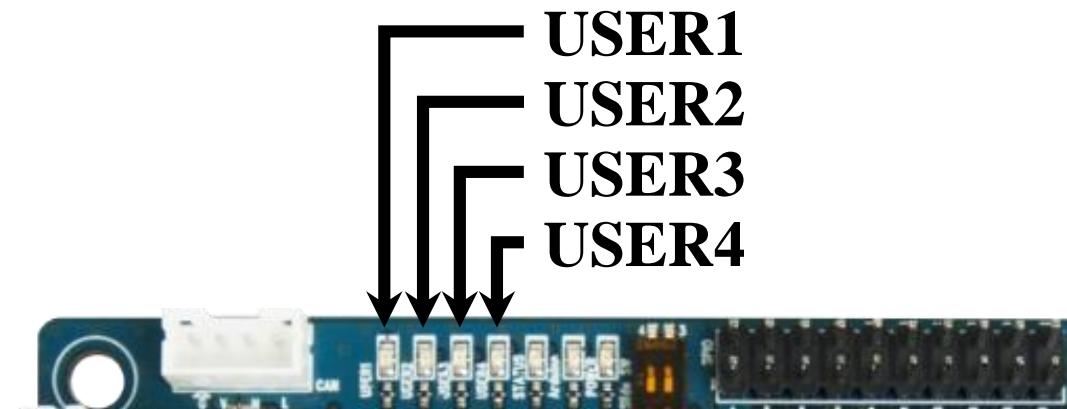
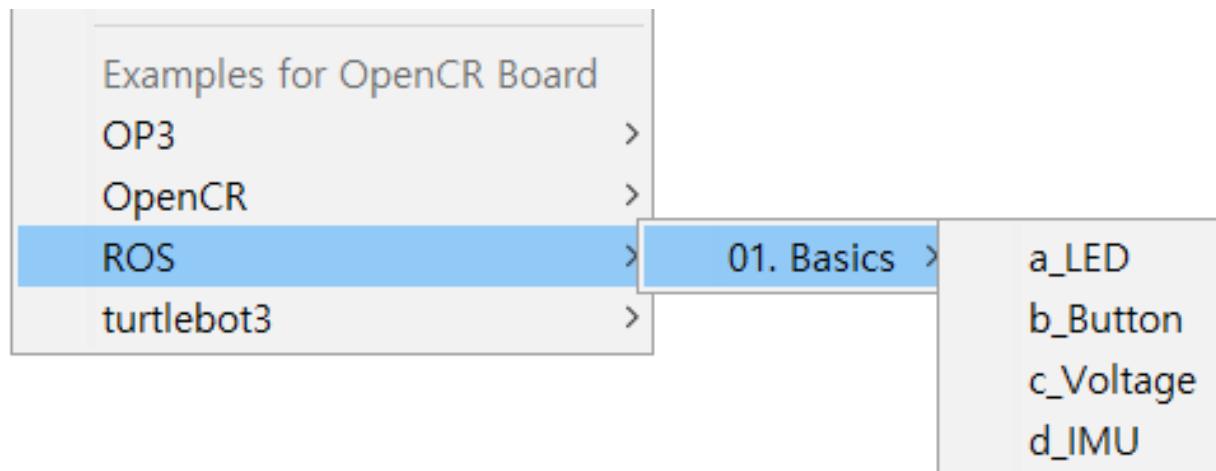
- OpenCR은 Arduino IDE를 지원함
- OpenCR 개발환경 구축 방법
 - http://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_opencr1_0/
 - <http://emanual.robotis.com/docs/en/parts/controller/opencr10/>



rosserial 예제 (LED 제어)

\$ arduino

- Arduino를 실행 후, [File] > [Examples] > [ROS] > [01. Basics] > [a_LED]라는 기본 예제를 불러와 빌드하여 업로드하자.
- 이 예제는 LED 4개를 ROS 표준 데이터 타입인 std_msg/Byte를 이용하여 led_out 서비스스크라이버를 정의 한다.
- 서비스스크라이버의 콜백 함수가 호출되면 전달된 메시지 값의 0~3번 비트 값에 해당하는 LED 를 비트가 1이면 LED를 켜고 0이면 끈다.



rosserial 예제 (LED 제어)

```
#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Byte.h>

int led_pin_user[4] = { BDPIN_LED_USER_1, BDPIN_LED_USER_2,
BDPIN_LED_USER_3, BDPIN_LED_USER_4 };

ros::NodeHandle nh;

void messageCb( const std_msgs::Byte& led_msg) {
    int i;

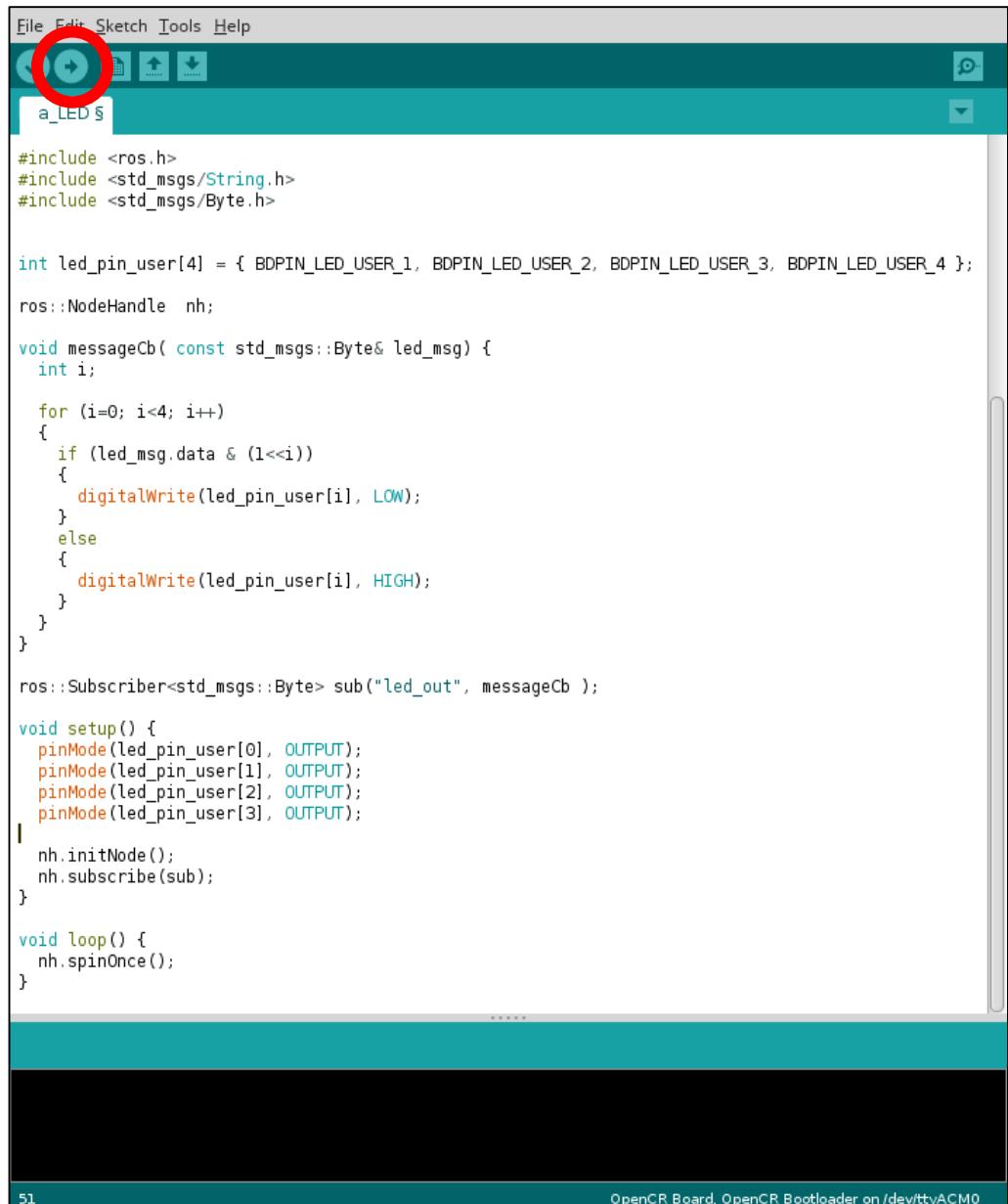
    for (i=0; i<4; i++)
    {
        if (led_msg.data & (1<<i))
        {
            digitalWrite(led_pin_user[i], LOW);
        }
        else
        {
            digitalWrite(led_pin_user[i], HIGH);
        }
    }

    ros::Subscriber<std_msgs::Byte> sub("led_out", messageCb );
}

void setup() {
    pinMode(led_pin_user[0], OUTPUT);
    pinMode(led_pin_user[1], OUTPUT);
    pinMode(led_pin_user[2], OUTPUT);
    pinMode(led_pin_user[3], OUTPUT);

    nh.initNode();
    nh.subscribe(sub);
}

void loop(){
    nh.spinOnce();
}
```



```
File Edit Sketch Tools Help
a_LED_S
#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Byte.h>

int led_pin_user[4] = { BDPIN_LED_USER_1, BDPIN_LED_USER_2, BDPIN_LED_USER_3, BDPIN_LED_USER_4 };

ros::NodeHandle nh;

void messageCb( const std_msgs::Byte& led_msg) {
    int i;

    for (i=0; i<4; i++)
    {
        if (led_msg.data & (1<<i))
        {
            digitalWrite(led_pin_user[i], LOW);
        }
        else
        {
            digitalWrite(led_pin_user[i], HIGH);
        }
    }

    ros::Subscriber<std_msgs::Byte> sub("led_out", messageCb );
}

void setup() {
    pinMode(led_pin_user[0], OUTPUT);
    pinMode(led_pin_user[1], OUTPUT);
    pinMode(led_pin_user[2], OUTPUT);
    pinMode(led_pin_user[3], OUTPUT);

    nh.initNode();
    nh.subscribe(sub);
}

void loop(){
    nh.spinOnce();
}
```

rosserial server 실행 및 LED 제어를 위한 토픽 퍼블리시

- roscore를 실행한 후, rosserial sever를 실행시킨다.

```
$ roscore
```

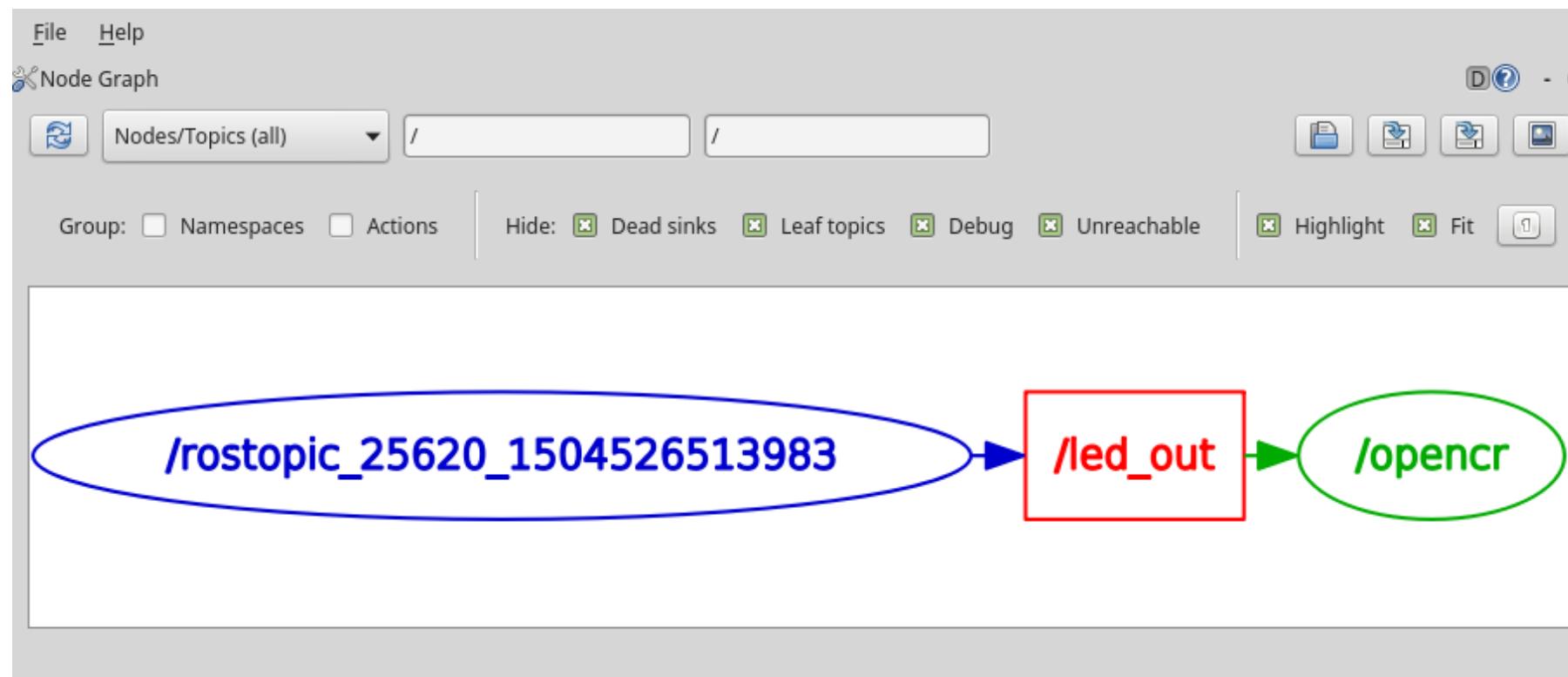
```
$ rosrun rosserial_python serial_node.py __name:=opencr _port:=/dev/ttyACM0 _baud:=115200
[INFO] [1495609829.326019]: ROS Serial Python Node
[INFO] [1495609829.336151]: Connecting to /dev/ttyACM0 at 115200 baud
[INFO] [1495609831.454144]: Note: subscribe buffer size is 1024 bytes
[INFO] [1495609831.454994]: Setup subscriber on led_out [std_msgs/Byte]
```

- rostopic pub을 이용하여 led_out에 값을 입력하여 LED를 제어해보자.

\$ rostopic pub -1 led_out std_msgs/Byte 1	→ USER1 LED On
\$ rostopic pub -1 led_out std_msgs/Byte 2	→ USER2 LED On
\$ rostopic pub -1 led_out std_msgs/Byte 4	→ USER3 LED On
\$ rostopic pub -1 led_out std_msgs/Byte 8	→ USER4 LED On
\$ rostopic pub -1 led_out std_msgs/Byte 0	→ LED Off

LED 제어를 위한 퍼블리셔 노드와 서브스크라이버 노드

- rqt_graph를 실행해 보자.
- rostopic 명령어가 퍼블리셔 노드로 opencr(rosserial server)가 서브스크라이버로 동작하고 있으며, 두 노드간에 '/led_out'이라는 토픽명으로 정보가 송수신되고 있는 것을 확인할 수 있다.



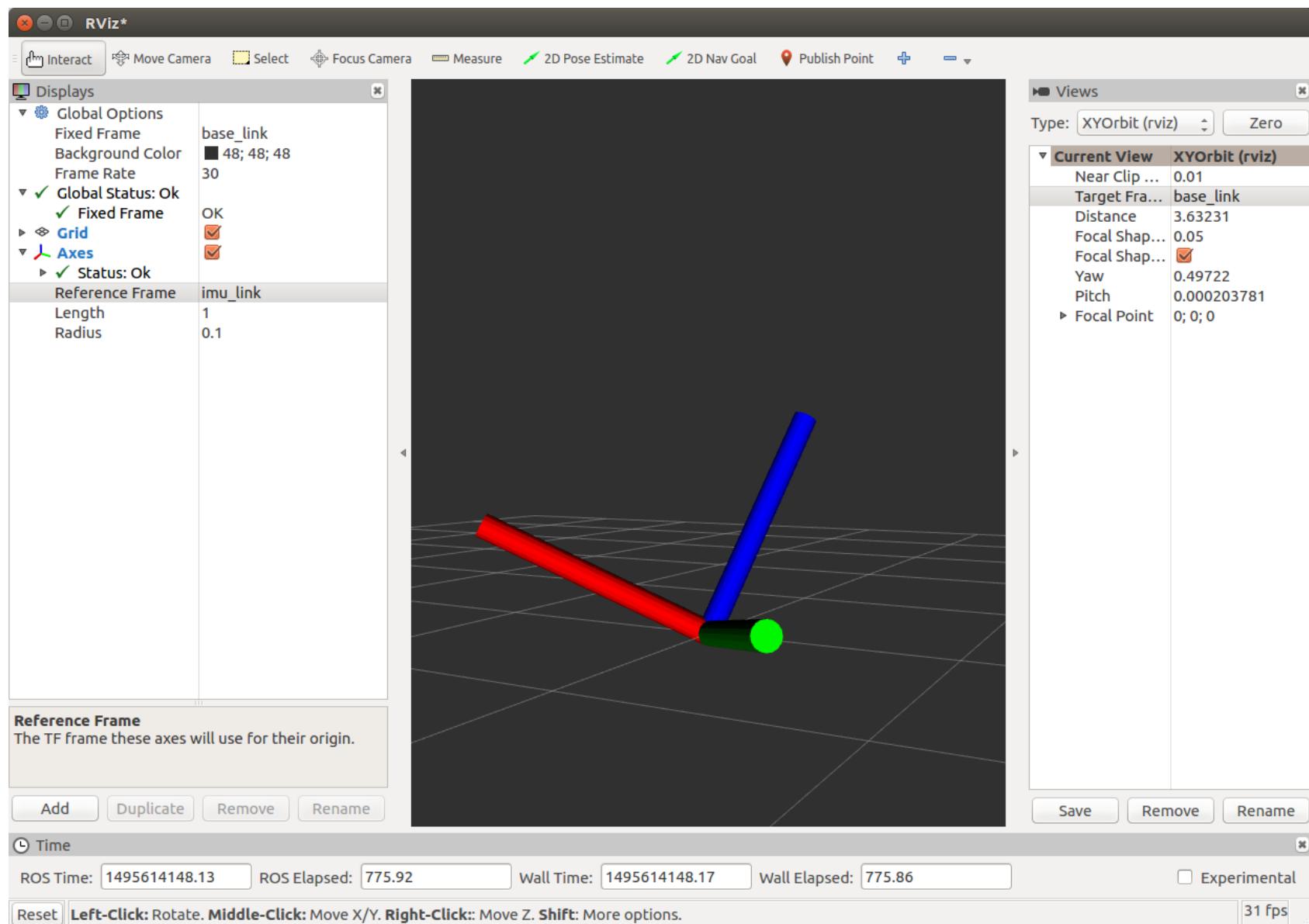
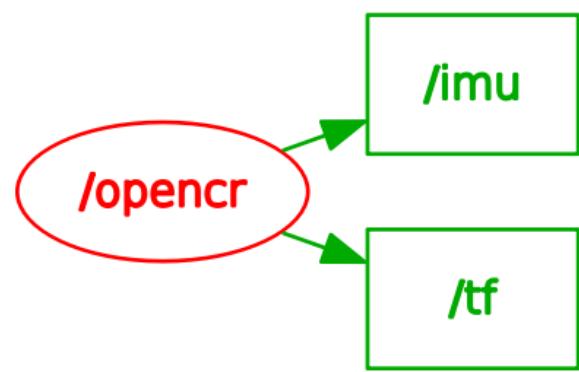
실습 시간

rostopic pub을 이용하여 **led_out**에 값을 입력하여 LED를 제어하기

rqt에서 **topic** 을 퍼블리쉬하여 LED를 제어하기

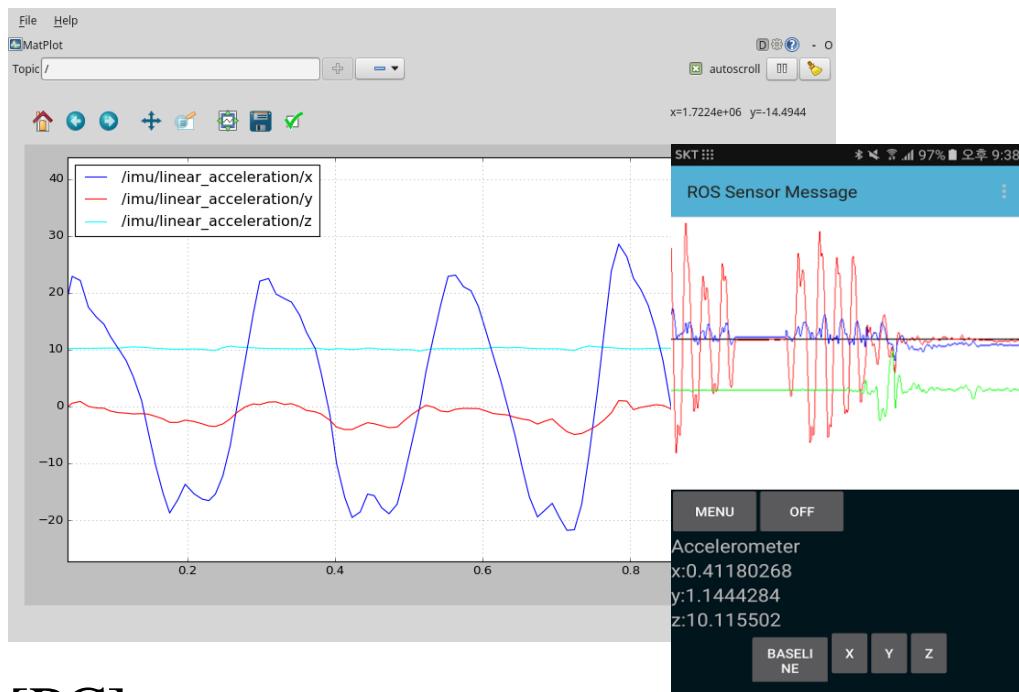
sound 라는 토픽을 받아 멜로디 출력해보기

rosserial 예제 (IMU제어)



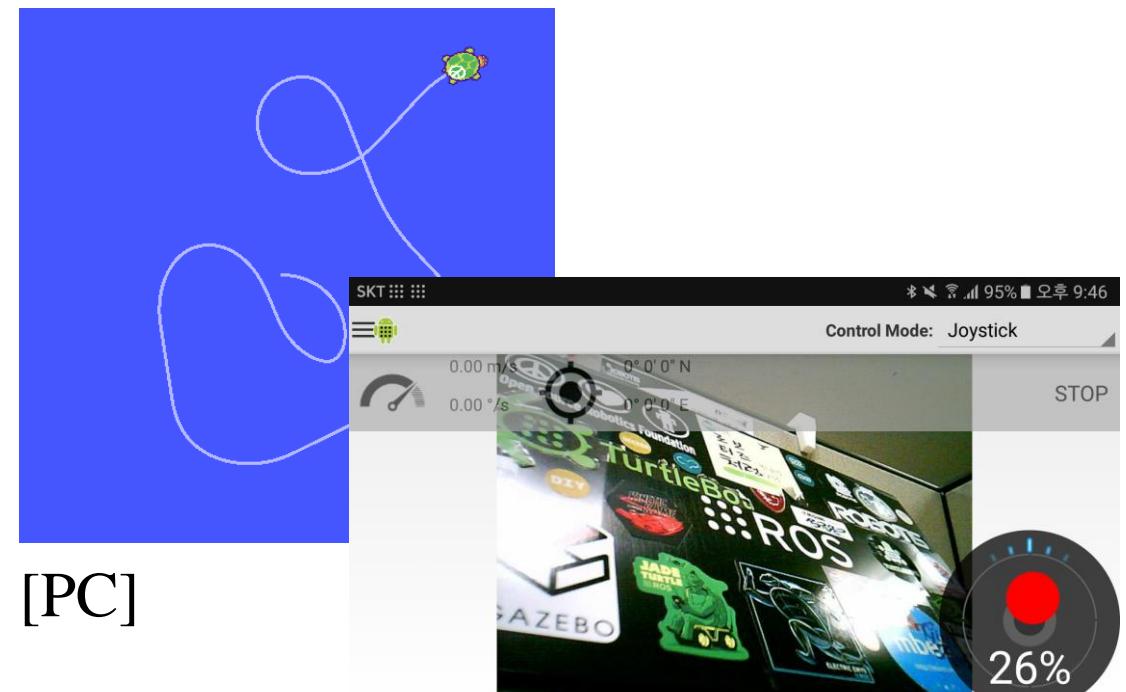
이기종 디바이스 간의 통신 예제

- 예제1: 원격으로 이미지 전송 (8장 카메라 참고)
- 예제2: 안드로이드 스마트폰의 가속도 값을 PC에서 확인하기 ([APP](#))
- 예제3: 안드로이드 스마트폰으로 TurtleBot 제어하기 ([APP](#))



[PC]

[Smartphone]



[Smartphone]

실습 시간

imu 토픽을 RViz에서 확인하기

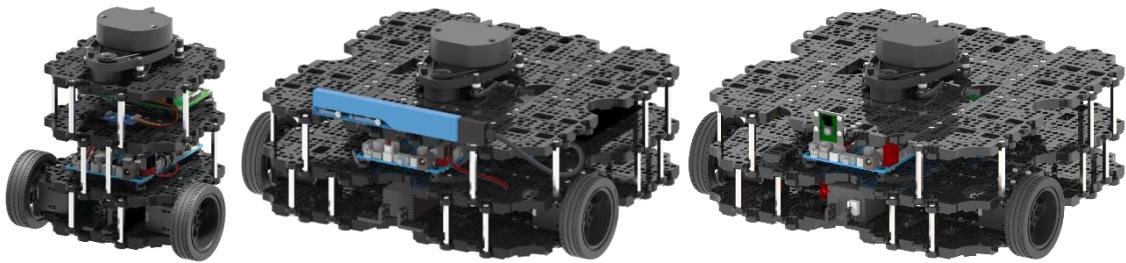
imu 토픽을 rqt에서 확인하기

imu 토픽을 스마트폰에서 확인하기

로봇의 위치 계측/추정하는 기능

■ 추측 항법(dead reckoning, 데드 레커닝)

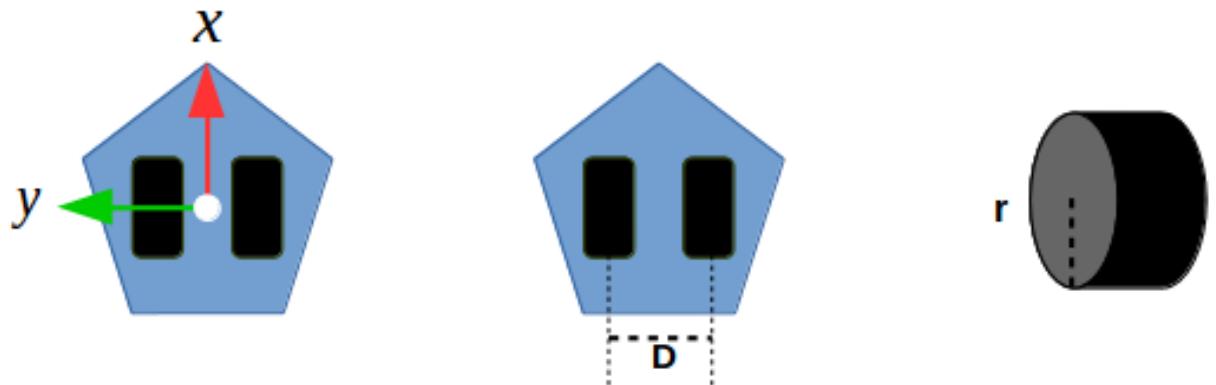
- 양 바퀴 축의 회전 값을 이용
- 이동 거리와 회전 값을 계산, 위치 측정
- 바닥 슬립, 기계적, 누적 오차 발생
- IMU 등의 관성 센서, 필터로 위치 보상
- 칼만필터 시리즈...



TurtleBot 3

■ 필요한 정보

- 양 바퀴 축의 엔코더 값 E
(모터 축인 경우 기어비로 재계산)
- 바퀴 간 거리 D
- 바퀴 반지름 r



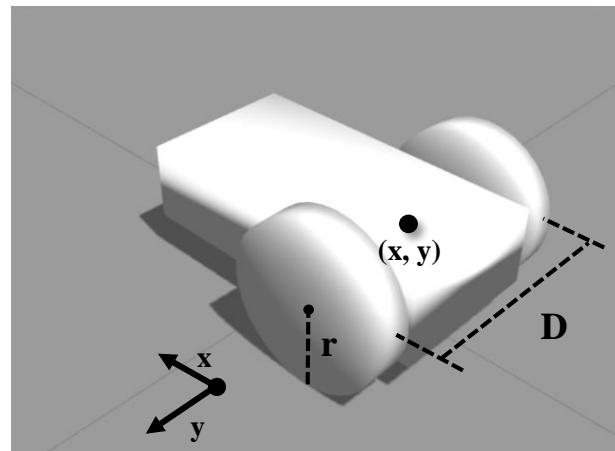
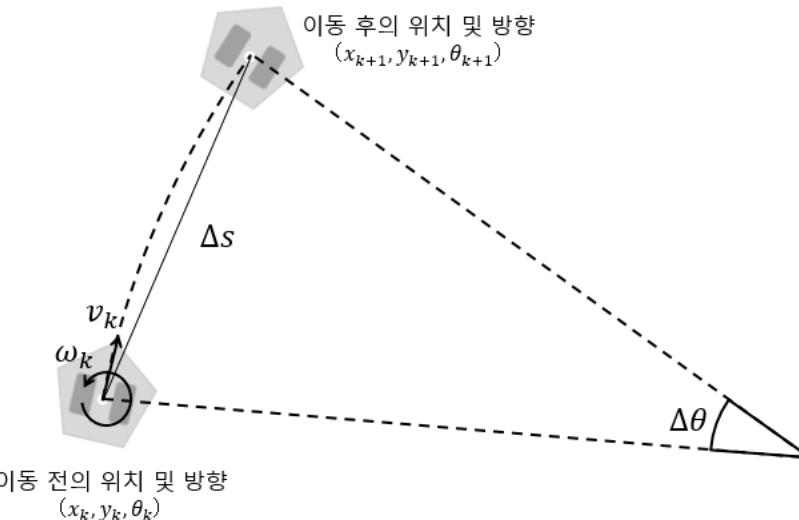
로봇의 위치 계측/추정하는 기능

■ 데드레커닝 계산

- 선속도(linear velocity: v)
- 각속도(angular velocity: w)

■ Runge-Kutta 공식 이용

- 이동한 위치의 근사 값 x, y
- 회전 각도 θ



$$v_l = \frac{(E_l c - E_l p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$v_r = \frac{(E_r c - E_r p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$V_l = v_l \cdot r \text{ (meter/sec)}$$

$$V_r = v_r \cdot r \text{ (meter/sec)}$$

$$v_k = \frac{(V_r + V_l)}{2} \text{ (meter/sec)}$$

$$\omega_k = \frac{(V_r - V_l)}{D} \text{ (radian/sec)}$$

$$\Delta s = v_k T_e \quad \Delta\theta = \omega_k T_e$$

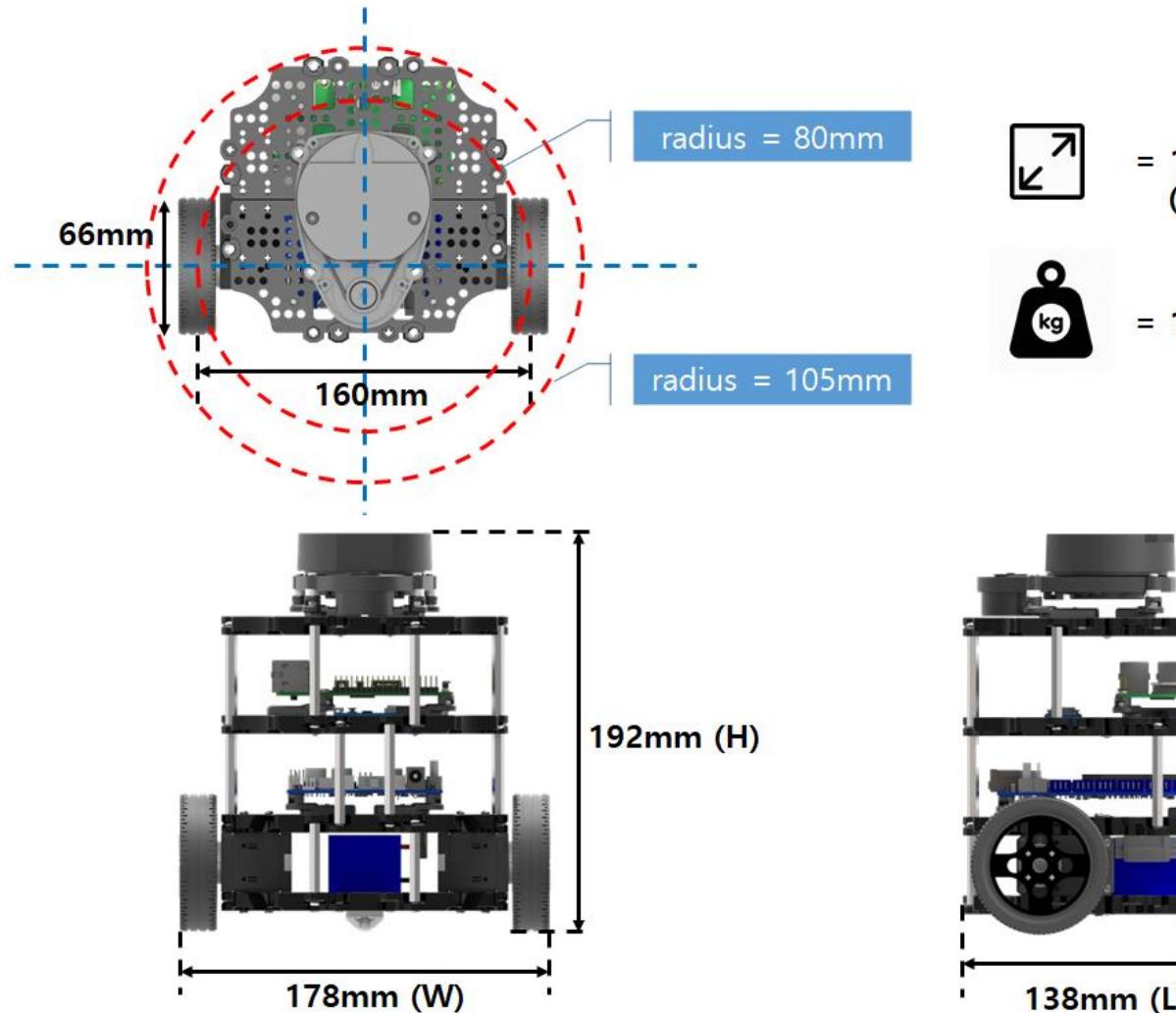
$$x_{(k+1)} = x_k + \Delta s \cos\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$y_{(k+1)} = y_k + \Delta s \sin\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$\theta_{(k+1)} = \theta_k + \Delta\theta$$

TurtleBot3 Burger의 바퀴 간 거리 D 와 바퀴 반지름 r

TurtleBot3 Burger



= $138 \times 178 \times 192$
(L x W x H, mm)



= 1 Kg

turtlebot_core 예제 (calcOdometry 함수)

```
bool calcOdometry(double diff_time)
{
    float* orientation;
    double wheel_l, wheel_r;
    double delta_s, theta, delta_theta;
    static double last_theta = 0.0;
    double v, w;
    double step_time;

    wheel_l = wheel_r = 0.0;
    delta_s = delta_theta = theta = 0.0;
    v = w = 0.0;
    step_time = 0.0;

    step_time = diff_time;

    if (step_time == 0)
        return false;

    wheel_l = TICK2RAD * (double)last_diff_tick[LEFT];
    wheel_r = TICK2RAD * (double)last_diff_tick[RIGHT];

    if (isnan(wheel_l))
        wheel_l = 0.0;

    if (isnan(wheel_r))
        wheel_r = 0.0;
```

```
delta_s    = WHEEL_RADIUS * (wheel_r + wheel_l) / 2.0;
orientation = sensors.getOrientation();
theta      = atan2f(orientation[1]*orientation[2] +
                   orientation[0]*orientation[3],
                   0.5f - orientation[2]*orientation[2] -
                   orientation[3]*orientation[3]);

delta_theta = theta - last_theta;

v = delta_s / step_time;
w = delta_theta / step_time;

last_velocity[LEFT] = wheel_l / step_time;
last_velocity[RIGHT] = wheel_r / step_time;

// compute odometric pose
odom_pose[0] += delta_s * cos(odom_pose[2] + (delta_theta / 2.0));
odom_pose[1] += delta_s * sin(odom_pose[2] + (delta_theta / 2.0));
odom_pose[2] += delta_theta;

// compute odometric instantaneouse velocity
odom_vel[0] = v;
odom_vel[1] = 0.0;
odom_vel[2] = w;

last_theta = theta;

return true;
}
```

실습 시간

터틀봇3의 calcOdometry 소스 코드 분석

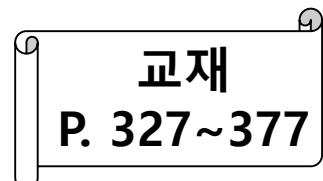
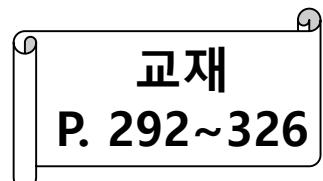
터틀봇3의 turtlebot_core 소스 코드 분석

자신만의 로봇에 적용하기

로봇 파일로봇 개발을 위한 원격제어 프로그래밍 실습

10_로봇 파일로봇.pdf

11_SLAM과_내비게이션.pdf



TurtleBot3의 원격 제어 방식

- Keyboard
- RC100
- PS Joystick
- XBOX 360 Joystick
- Wii Remote
- Nunchuk
- Android App
- LEAP Motion
- Myo
- <http://emanual.robotis.com/docs/en/platform/turtlebot3/teleoperation/#teleoperation>



TurtleBot3 원격 제어

- roscore 구동 [Remote PC]

```
$ roscore
```

- turtlebot3_robot.launch 런치 파일 실행 [TurtleBot]

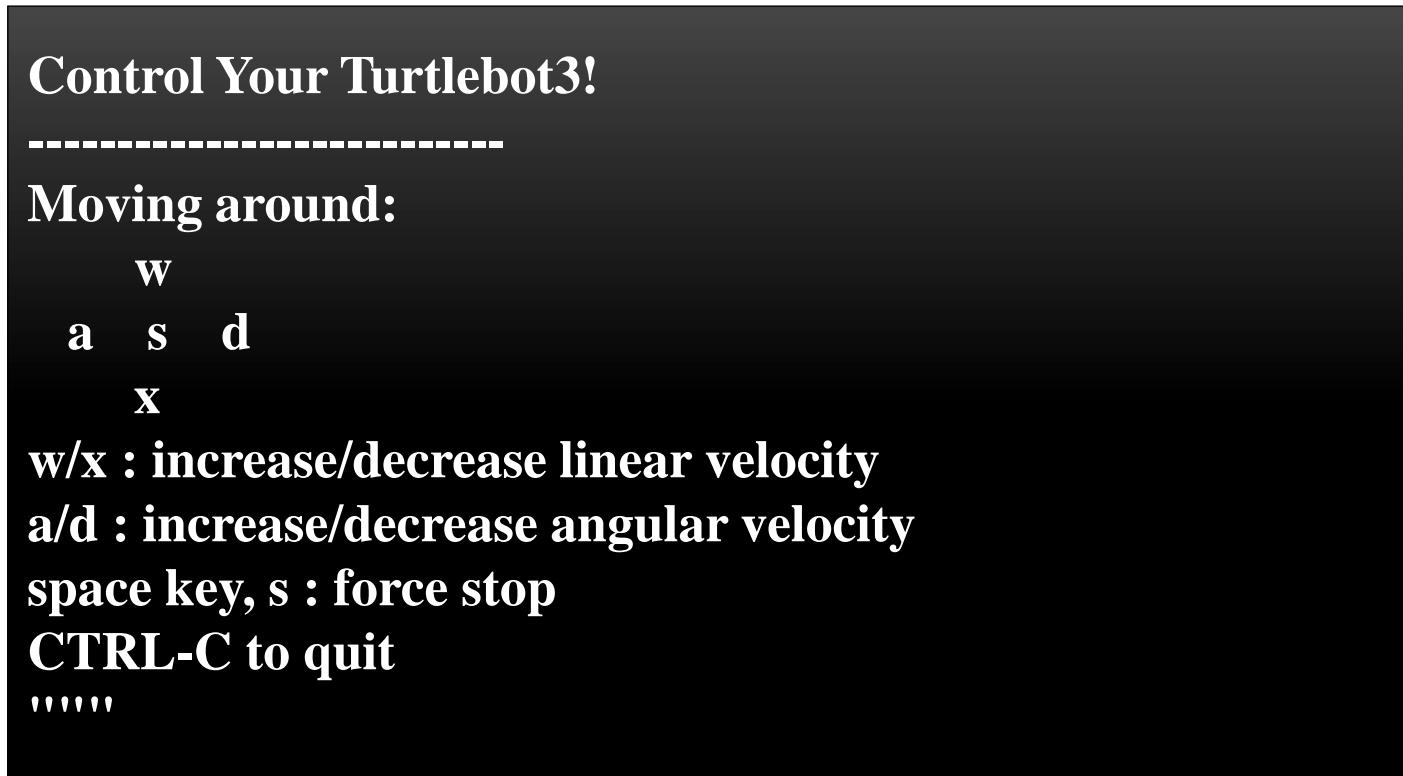
```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch --screen
```

- turtlebot3_teleop_key.launch 런치 파일 실행 [Remote PC]

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch --screen
```

TurtleBot3의 원격 제어 방식

- Keyboard 를 이용하는 방법을 파헤쳐보자.
 - turtlebot3_teleop 패키지
 - https://github.com/ROBOTIS-GIT/turtlebot3/blob/master/turtlebot3_teleop/scripts/turtlebot3_teleop_key



TurtleBot3의 원격 제어 방식

- 외부 조정기를 이용하는 방법을 파헤쳐보자.
 - OpenCR의 RC100B

```
void Turtlebot3Controller::getRCdata(float *get_cmd_vel)
{
    uint16_t received_data = 0;

    static float lin_x = 0.0, ang_z = 0.0;

    if (rc100_.available())
    {
        received_data = rc100_.readData();

        if (received_data & RC100_BTN_U)
        {
            lin_x += VELOCITY_LINEAR_X * scale_lin_vel_;
        }
        else if (received_data & RC100_BTN_D)
        {
            lin_x -= VELOCITY_LINEAR_X * scale_lin_vel_;
        }
        else if (received_data & RC100_BTN_L)
        {
            ang_z += VELOCITY_ANGULAR_Z * scale_ang_vel_;
        }
        else if (received_data & RC100_BTN_R)
        {
            ang_z -= VELOCITY_ANGULAR_Z * scale_ang_vel_;
        }
        else if (received_data & RC100_BTN_6)
        {
            lin_x = const_cmd_vel_;
            ang_z = 0.0;
        }
    }
}
```

```
else if (received_data & RC100_BTN_6)
{
    lin_x = const_cmd_vel_;
    ang_z = 0.0;
}
else if (received_data & RC100_BTN_5)
{
    lin_x = 0.0;
    ang_z = 0.0;
}
else
{
    lin_x = lin_x;
    ang_z = ang_z;
}

if (lin_x > max_lin_vel_)
{
    lin_x = max_lin_vel_;
}

if (ang_z > max_ang_vel_)
{
    ang_z = max_ang_vel_;
}

get_cmd_vel[0] = lin_x;
get_cmd_vel[1] = ang_z;
}
```

실습 시간

외부 디바이스로 로봇 원격 제어하기

Keyboard / RC100 / PS Joystick / XBOX 360 Joystick

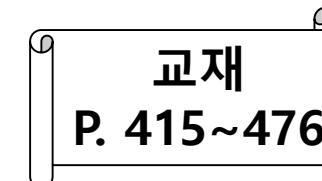
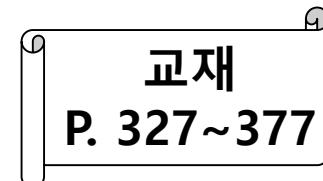
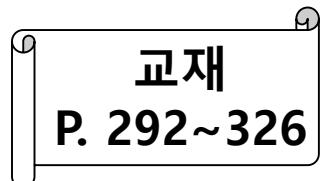
Wii Remote / Nunchuk / Android App / LEAP Motion / Myo

시뮬레이터 Gazebo를 사용하기 위한 모바일 로봇 모델링

10_모바일_로봇.pdf

11_SLAM과_내비게이션.pdf

13_매니퓰레이터.pdf

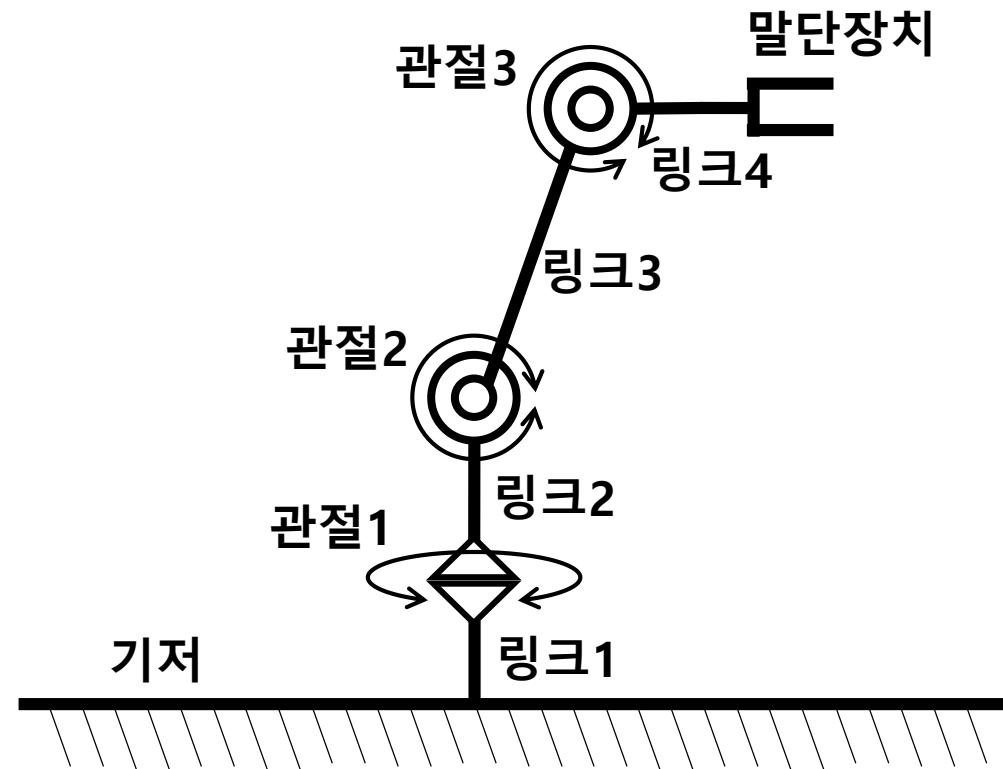


연습! 3관절 로봇 암 모델링

- 3관절 로봇 암의 구성
 - 관절 3개, 링크 4개

- URDF 작성

```
$ cd ~/catkin_ws/src  
$ catkin_create_pkg testbot_description urdf  
$ cd testbot_description  
$ mkdir urdf  
$ cd urdf  
$ gedit testbot.urdf
```



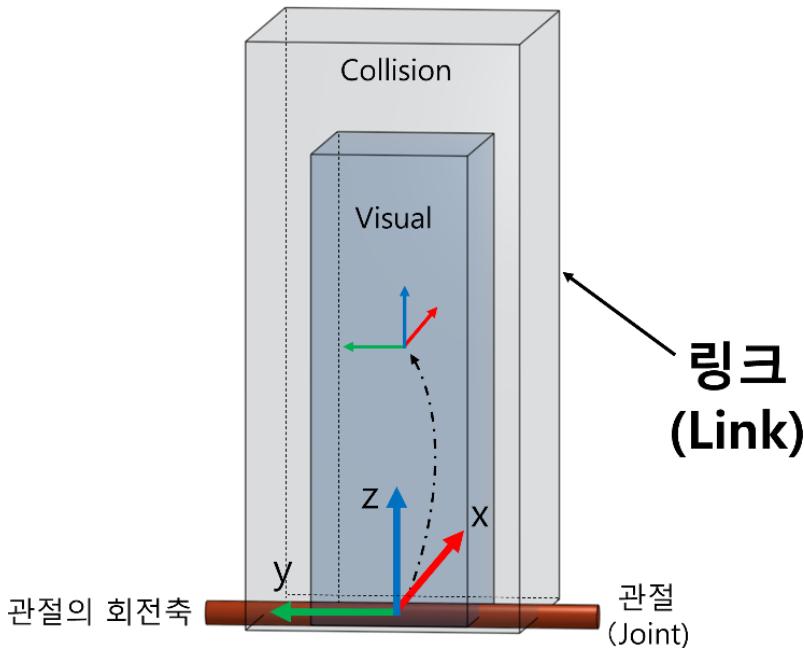
(전문)

https://github.com/ROBOTIS-GIT/ros_tutorials/blob/master/testbot_description/urdf/testbot.urdf

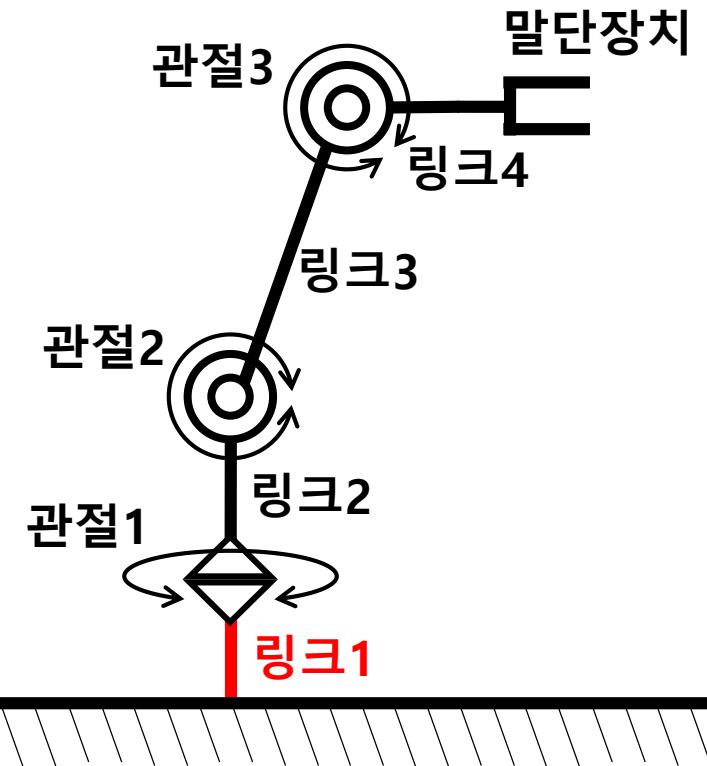
연습! 3관절 로봇 암 모델링

- 소스 일부분 : 링크 1

```
<link name="link1">  
  
  <collision>  
    <origin xyz="0 0 0.25" rpy="0 0 0"/>  
    <geometry>  
      <box size="0.1 0.1 0.5"/>  
    </geometry>  
  </collision>  
  
  <visual>  
    <origin xyz="0 0 0.25" rpy="0 0 0"/>  
    <geometry>  
      <box size="0.1 0.1 0.5"/>  
    </geometry>  
    <material name="black"/>  
  </visual>  
  
  <inertial>  
    <origin xyz="0 0 0.25" rpy="0 0 0"/>  
    <mass value="1"/>  
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"  
           iyy="1.0" iyz="0.0"  
           izz="1.0"/>  
  </inertial>  
  
</link>
```



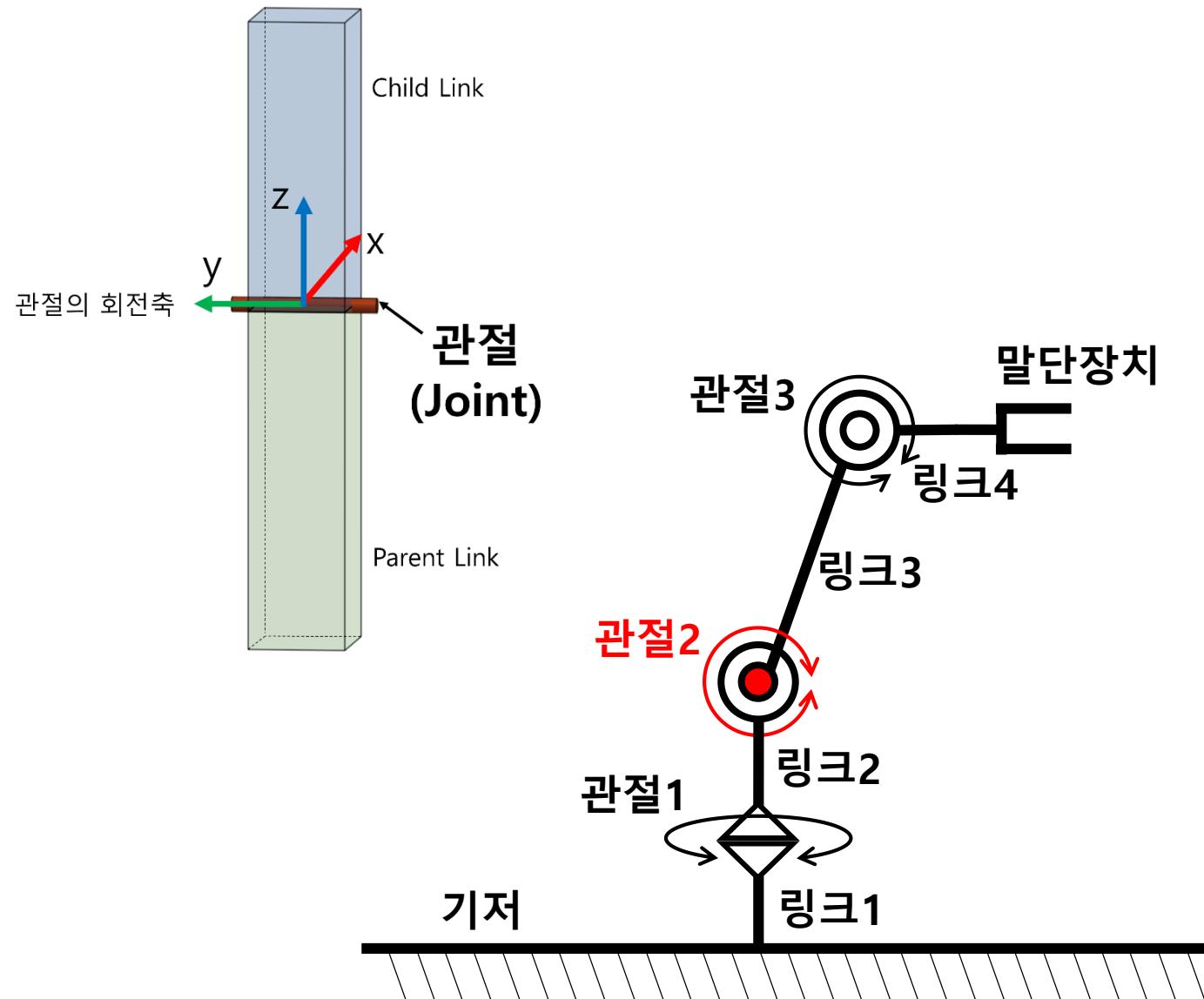
링크
(Link)



연습! 3관절 로봇 암 모델링

- 소스 일부분 : 관절 2

```
<joint name="joint2" type="revolute">  
  <parent link="link2"/>  
  <child link="link3"/>  
  <origin xyz="0 0 0.5" rpy="0 0 0"/>  
  <axis xyz="0 1 0"/>  
  <limit effort="30" lower="-2.617"  
    upper="2.617" velocity="1.571"/>  
</joint>
```



```

<?xml version="1.0" ?>
<robot name="test_robot">
  <material name="black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="orange">
    <color rgba="1.0 0.4 0.0 1.0"/>
  </material>
  <link name="base"/>
  <joint name="fixed" type="fixed">
    <parent link="base"/>
    <child link="link1"/>
  </joint>

```

이거 이해 했어!

(URDF 전문 링크)

```

<link name="link1">
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
  <geometry>
    <box size="0.1 0.1 0.5"/>
  </geometry>
  </collision>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
    <material name="orange"/>
  </visual>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
             iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

<joint name="joint1" type="revolute">
  <parent link="link1"/>
  <child link="link2"/>
  <origin rpy="0 0 0" xyz="0 0 0.5"/>
  <axis xyz="0 0 1"/>
  <limit effort="30" lower="-2.617"
        upper="2.617" velocity="1.571"/>
</joint>

```

```

<link name="link2">
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
  <geometry>
    <box size="0.1 0.1 0.5"/>
  </geometry>
  </collision>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
    <material name="orange"/>
  </visual>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
             iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

<joint name="joint2" type="revolute">
  <parent link="link2"/>
  <child link="link3"/>
  <origin rpy="0 0 0" xyz="0 0 0.5"/>
  <axis xyz="0 1 0"/>
  <limit effort="30" lower="-2.617"
        upper="2.617" velocity="1.571"/>
</joint>

<link name="link3">
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.5"/>
  <geometry>
    <box size="0.1 0.1 1"/>
  </geometry>
  </collision>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.5"/>
    <geometry>
      <box size="0.1 0.1 1"/>
    </geometry>
    <material name="black"/>
  </visual>

```

```

<inertial>
  <origin rpy="0 0 0" xyz="0 0 0.5"/>
  <mass value="1"/>
  <inertia ixx="1.0" ixy="0.0" ixz="0.0"
            iyy="1.0" iyz="0.0" izz="1.0"/>
</inertial>
</link>

<joint name="joint3" type="revolute">
  <parent link="link3"/>
  <child link="link4"/>
  <origin rpy="0 0 0" xyz="0 0 1.0"/>
  <axis xyz="0 1 0"/>
  <limit effort="30" lower="-2.617"
        upper="2.617" velocity="1.571"/>
</joint>

<link name="link4">
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
  <geometry>
    <box size="0.1 0.1 0.5"/>
  </geometry>
  </collision>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
    <material name="orange"/>
  </visual>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
             iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

</robot>

```

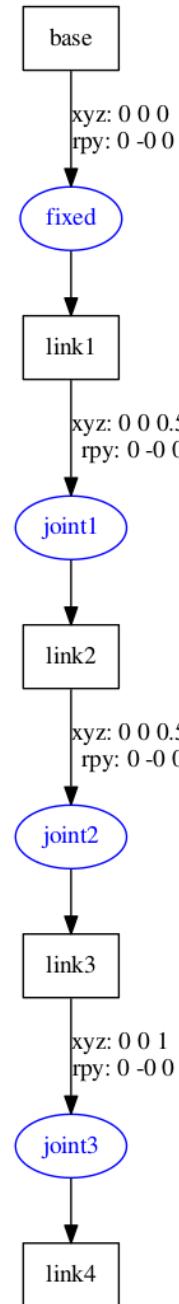
URDF 문법 검사 및 그래프 표시

- check_urdf (문법 검사)

```
$ check_urdf testbot.urdf
robot name is: test_robot
----- Successfully Parsed XML -----
root Link: base has 1 child(ren)
    child(1): link1
        child(1): link2
            child(1): link3
                child(1): link4
```

- urdf_to_graphiz (그래프 표시)

```
$ urdf_to_graphiz testbot.urdf
Created file test_robot.gv
Created file test_robot.pdf
```



URDF 모델 확인

- RViz를 이용한 URDF 모델 확인 및 동작

- robot_description

: 로봇 모델을 XML형식으로 기술, 모델을 RViz에서 3차원으로 가식화 가능

- joint_state_publisher

: 관절의 각도 값 퍼블리시

- robot_state_publisher

: 관절 각도 값을 받아 관절과 링크간의 3차원 위치, 자세 정보를 tf로 퍼블리시

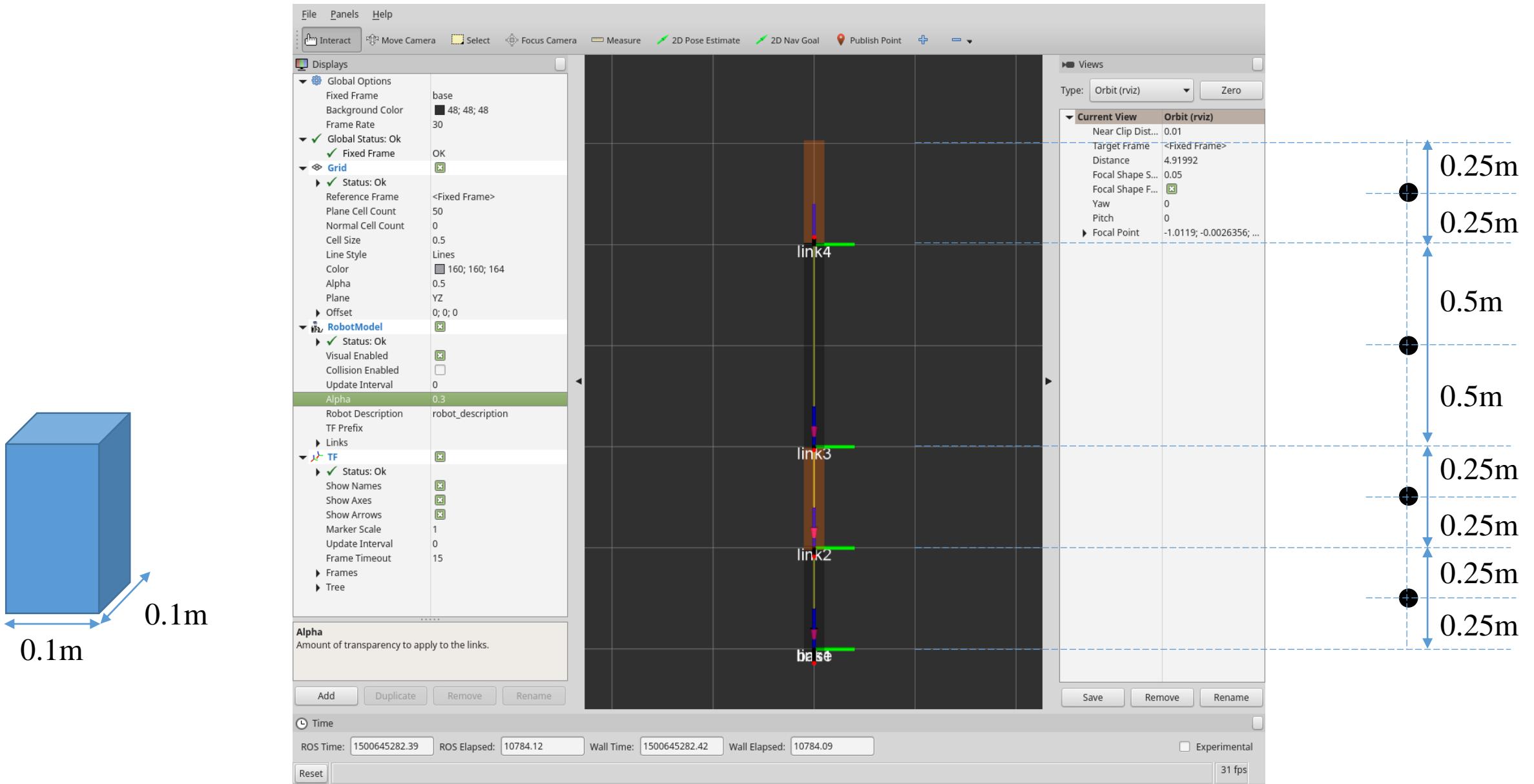


```
$ cd ~/catkin_ws/src/testbot_description  
$ mkdir launch  
$ cd launch  
$ gedit testbot.launch
```

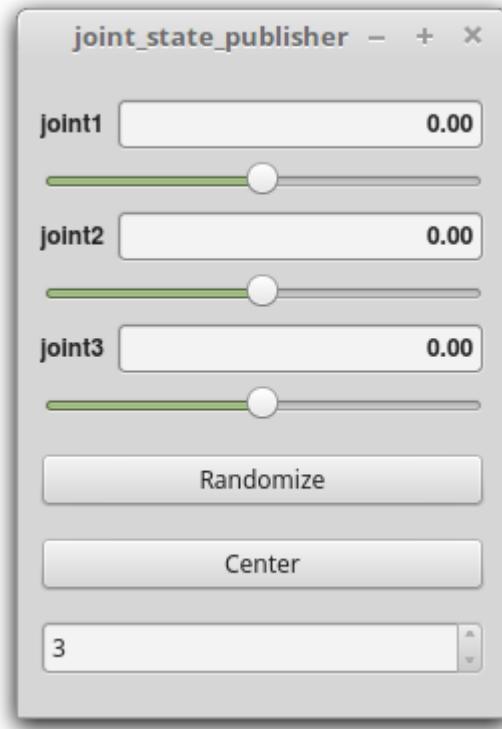
```
<launch>  
  <arg name="model" default="$(find testbot_description)/urdf/testbot.urdf" />  
  <arg name="gui" default="True" />  
  <param name="robot_description" textfile="$(arg model)" />  
  <param name="use_gui" value="$(arg gui)"/>  
  <node pkg="joint_state_publisher" type="joint_state_publisher" name="joint_state_publisher" />  
  <node pkg="robot_state_publisher" type="state_publisher" name="robot_state_publisher" />  
</launch>
```

```
$ roslaunch testbot_description testbot.launch  
$ rviz
```

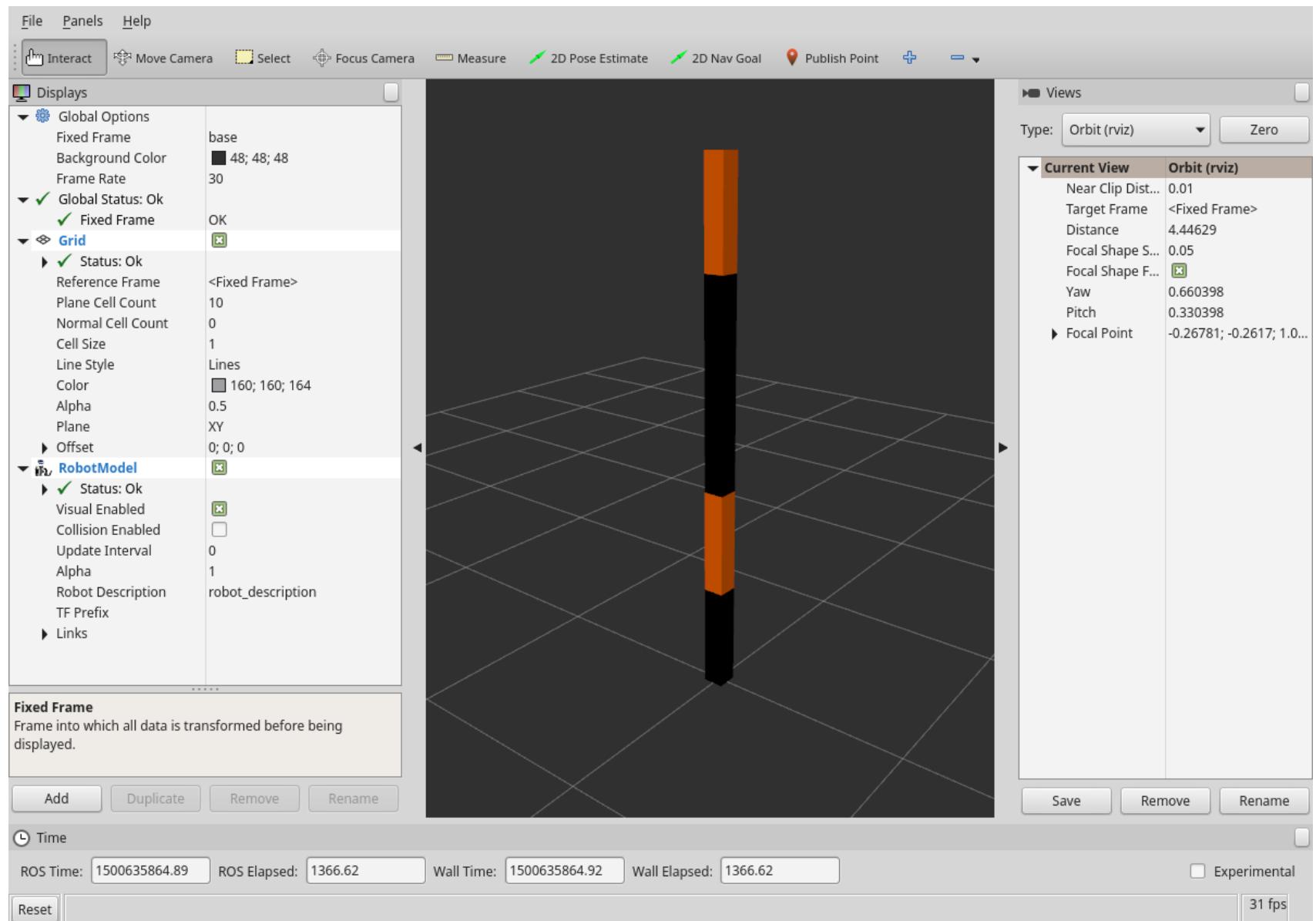
RViz에서 확인



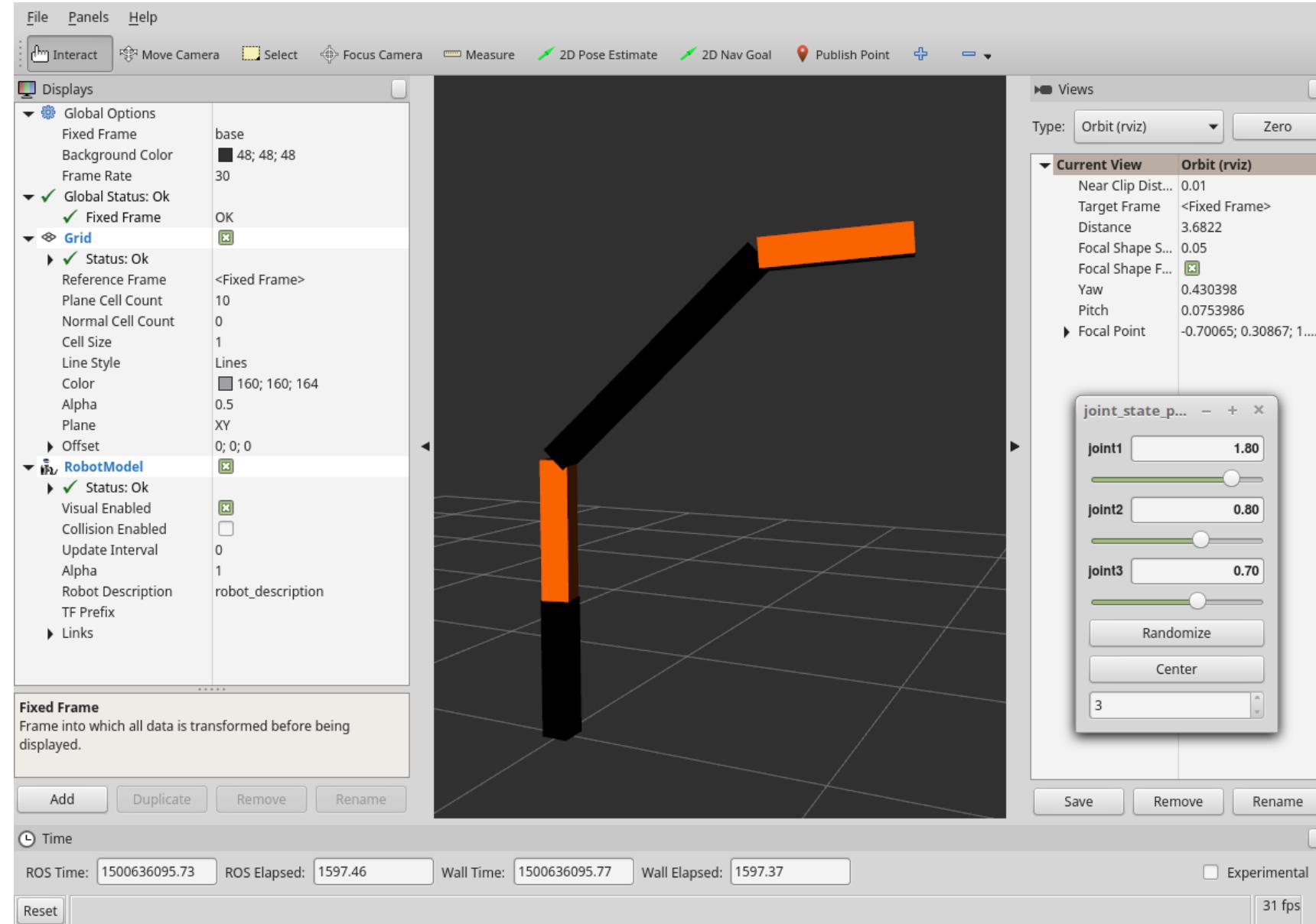
RViz에서 확인



조인트 7회로
이전경 해보세요.



RViz에서 확인

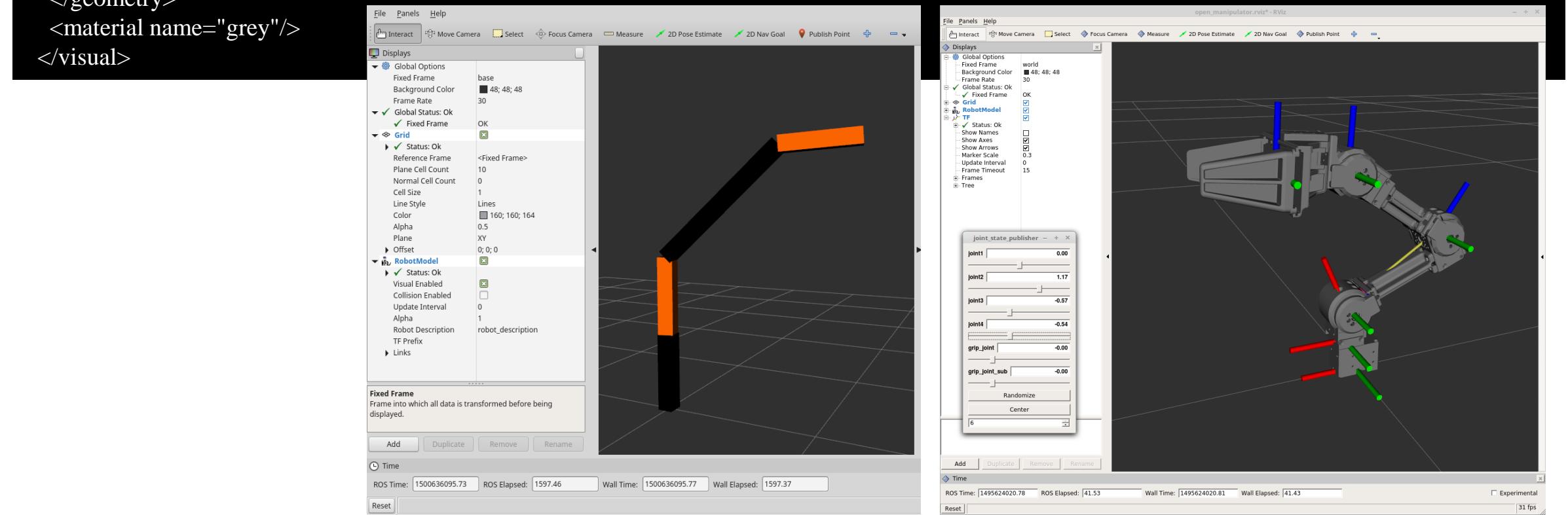


모델링에서 외관의 차이

- URDF의 link>visual>geometry 속성에서 STL 또는 DAE 파일 이용

예제) open_manipulator/open_manipulator_description/urdf/open_manipulator_chain.xacro

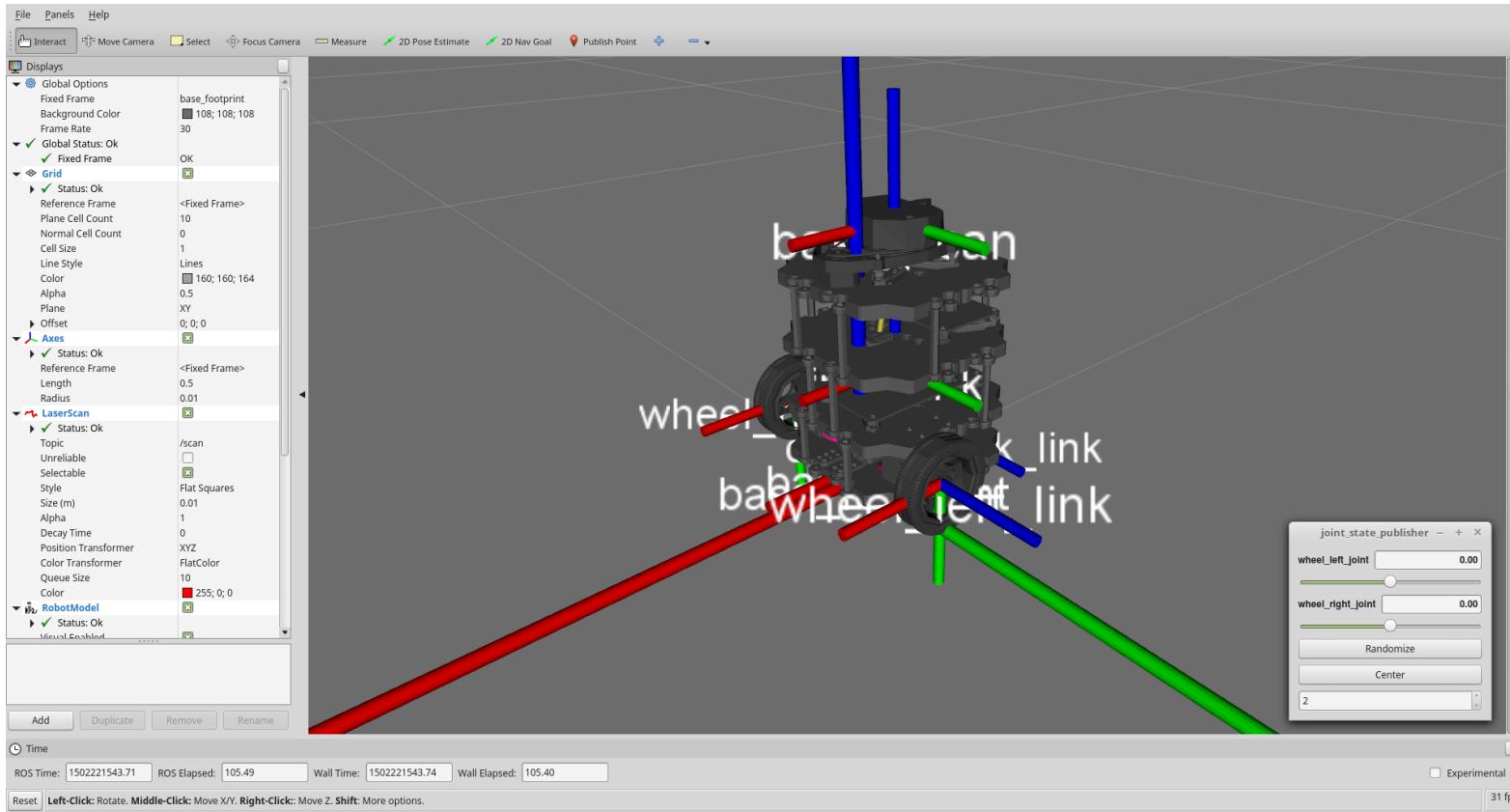
```
<visual>
  <origin xyz="0.0 0.0 0.0" rpy="0 0 0"/>
  <geometry>
    <mesh filename="package://open_manipulator_description/meshes/chain_link1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
  <material name="grey"/>
</visual>
```



TurtleBot3 모델링?

- RViz 실행 [Remote PC]]

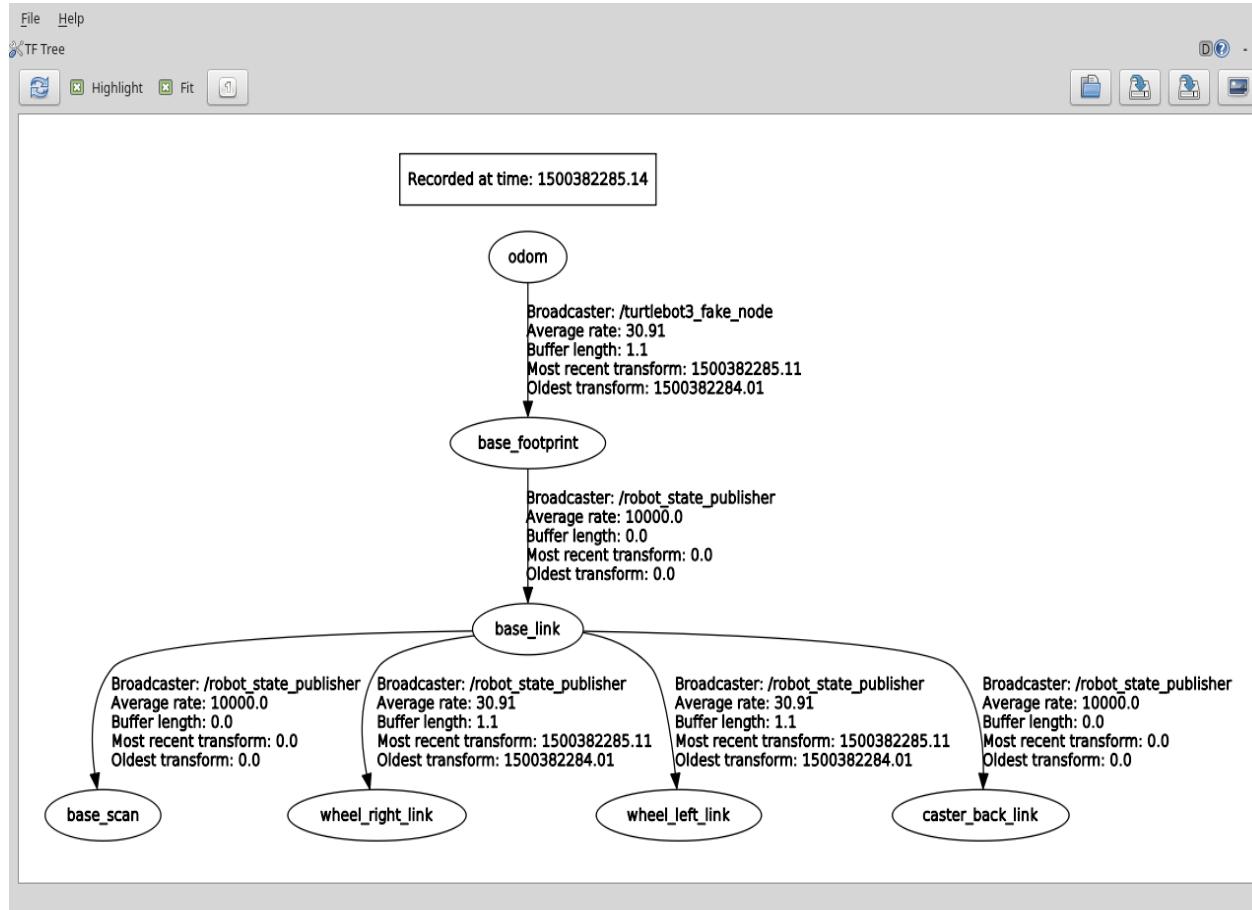
```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_bringup turtlebot3_model.launch
```



TurtleBot3 모델링과 TF 관계

```
$ rqt_graph
```

```
$ rqt [Plugins > Visualization > TF Tree]
```



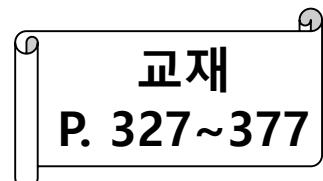
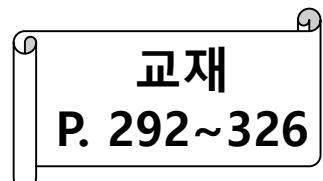
실습 시간

터틀봇3 모델을 참고하여 나만의 모바일 로봇 모델링하기
(turtlebot3_description 패키지 참고)

시뮬레이터 Gazebo를 사용하기 위한 가상 로봇 프로그래밍!

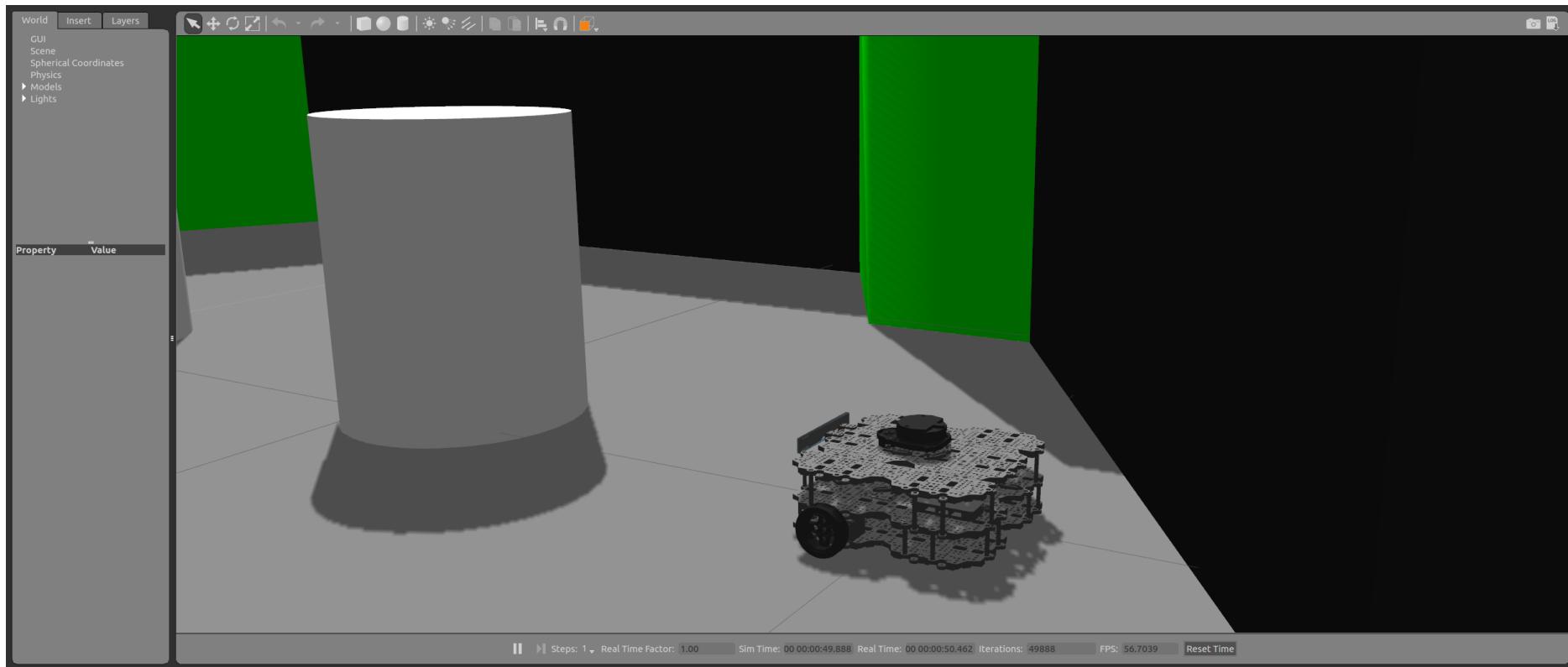
10_모바일로봇.pdf

11_SLAM과_내비게이션.pdf



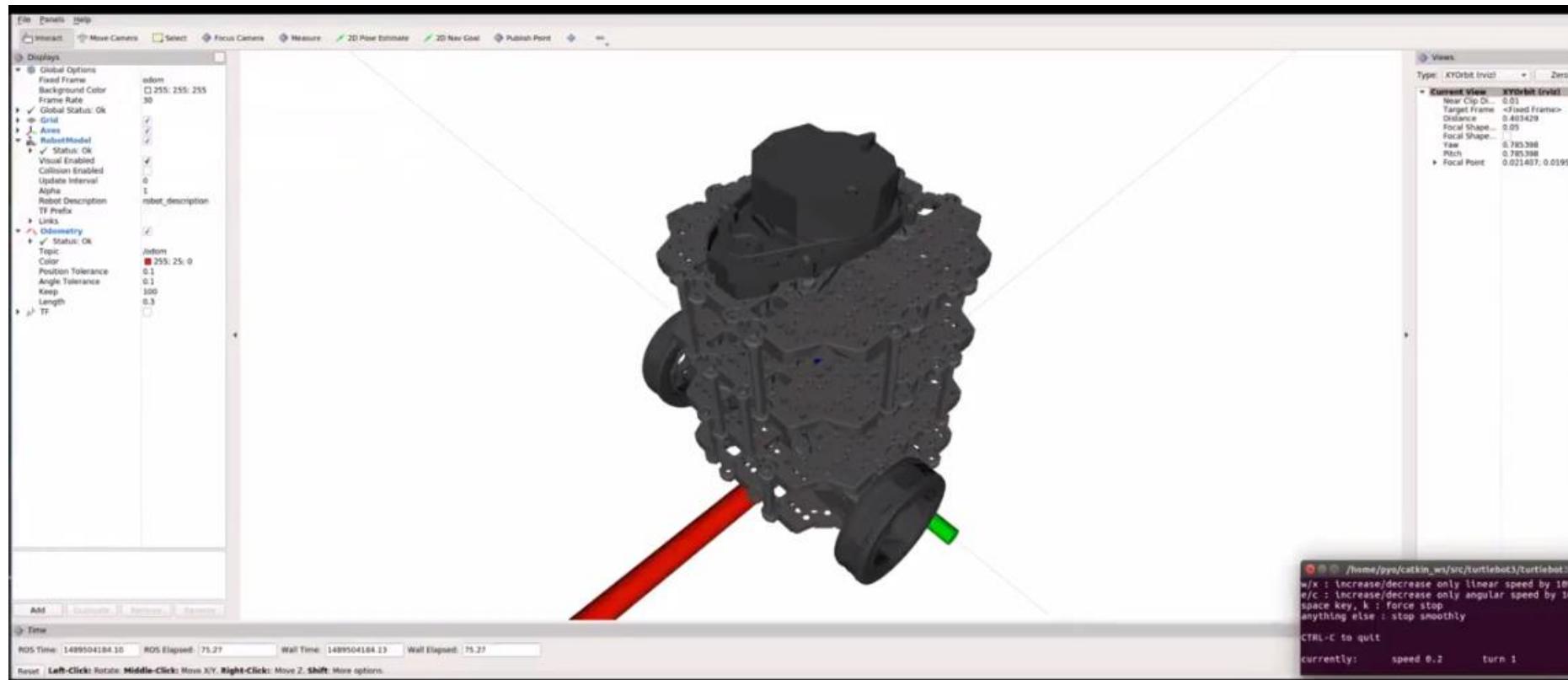
TurtleBot3 + 시뮬레이션

- 시뮬레이션을 위한 두 가지 방법
 - ROS의 3차원 시각화 도구인 RViz를 이용
 - 3차원 로봇 시뮬레이터 Gazebo를 이용
 - <http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>



시뮬레이션 / RViz를 뷰어로 사용할 경우

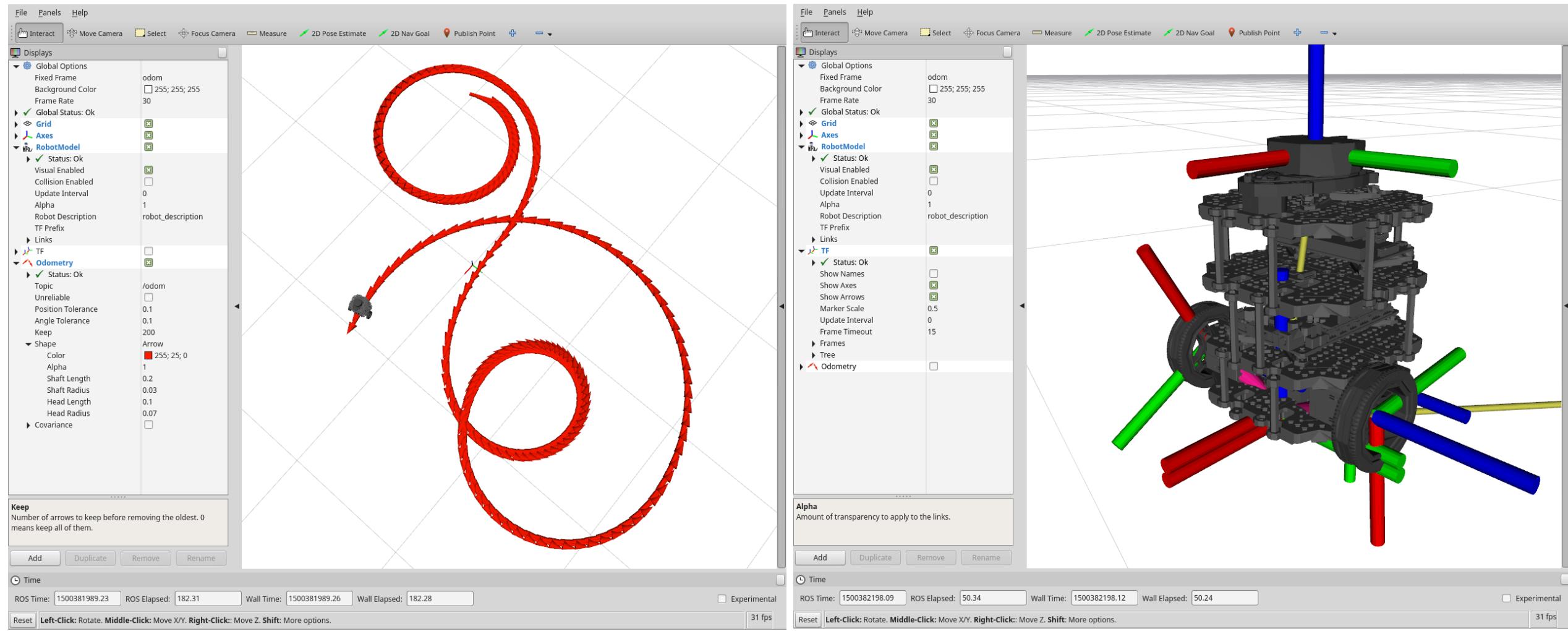
```
$ export TURTLEBOT3_MODEL=burger  
$ rosrun turtlebot3_fake turtlebot3_fake.launch  
  
$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch
```



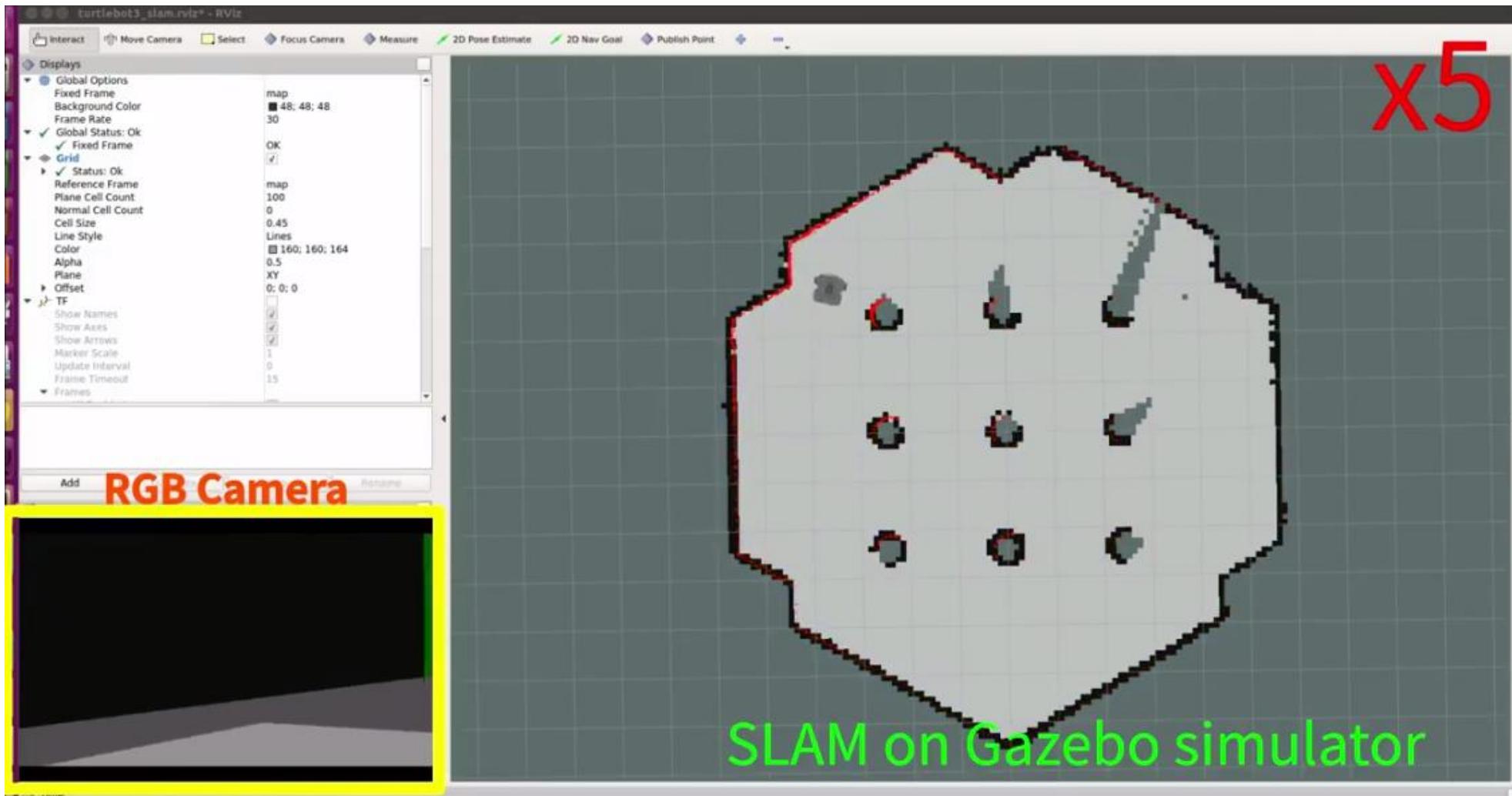
<https://youtu.be/iHXZSLBJHMg>

시뮬레이션 / RViz를 뷰어로 사용할 경우

- 로봇을 이동시켜 보면서 Odometry와 tf를 확인해 보자!



시뮬레이션 / Gazebo를 이용한 경우 / TurtleBot3 in Gazebo



https://youtu.be/xXM5r_SVkWM

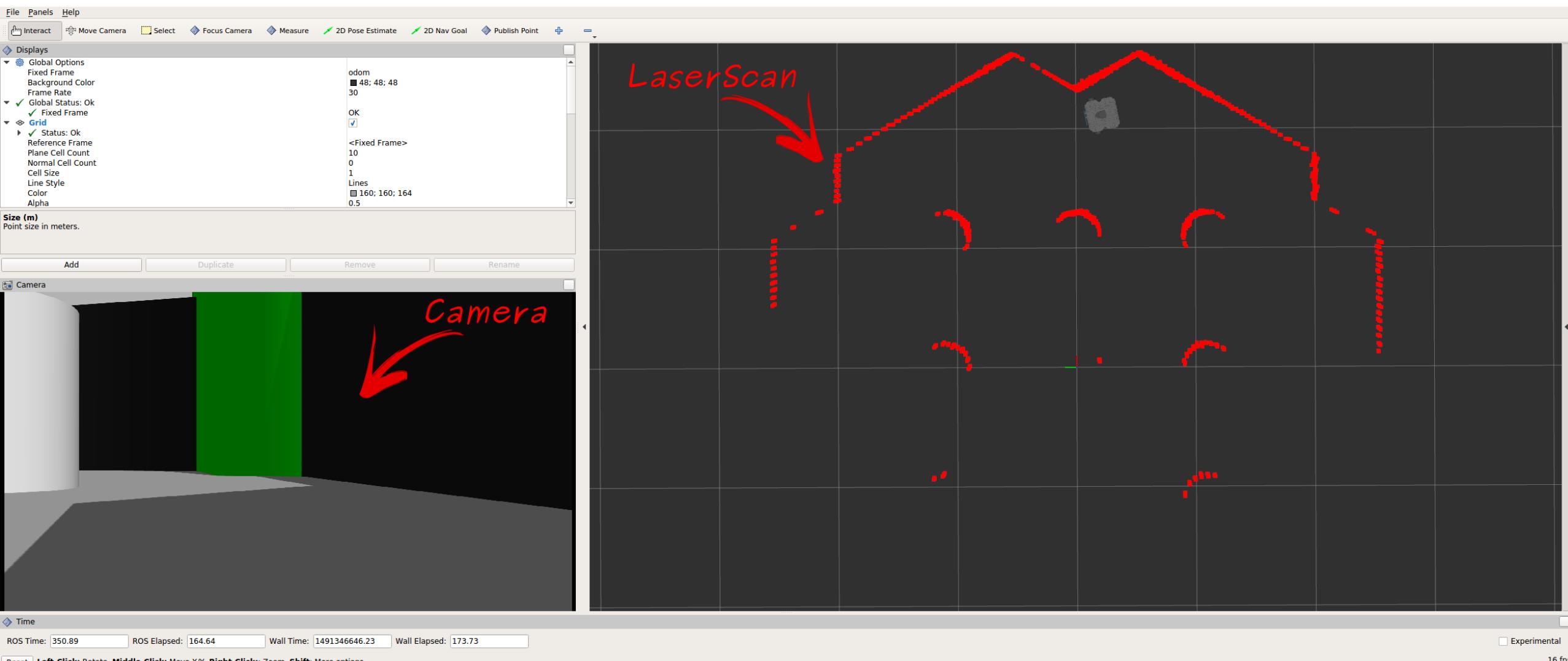
가상 로봇 실행 with Gazebo

```
$ rosrun turtlebot3_gazebo turtlebot3_world.launch
```

```
$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch
```

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ rosrun turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

가상 로봇 실행 with Gazebo



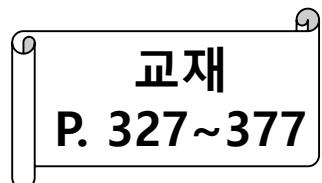
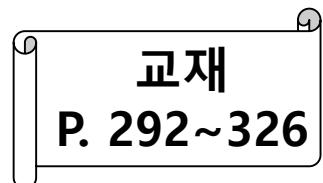
실습 시간

나만의 모바일 로봇을 Gazebo에서 사용해보기
(turtlebot3_gazebo 패키지 참고)

시뮬레이터 Gazebo를 활용한 SLAM 프로그래밍!

10_모바일로봇.pdf

11_SLAM과_내비게이션.pdf



가상 SLAM with Gazebo

- Gazebo 실행

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

- SLAM 실행

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
```

- RViz 실행

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz
```

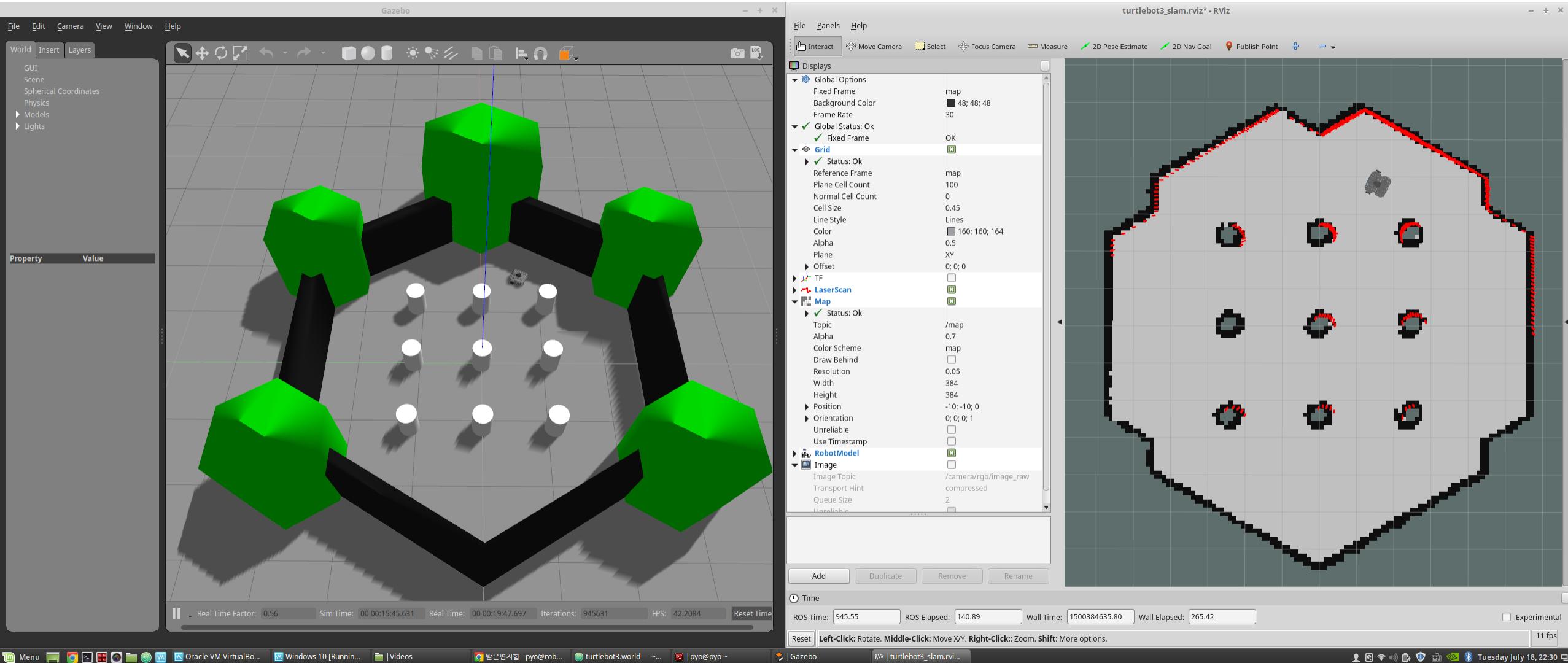
- 터틀봇 원격 조종

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

- 지도 출력

```
$ rosrun map_server map_saver -f ~/map
```

가상 SLAM with Gazebo



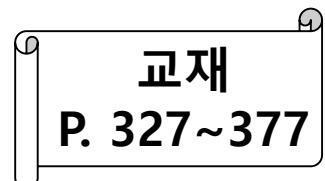
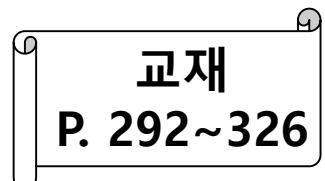
실습 시간

나만의 모바일 로봇을 이용하여
Gazebo상에서 SLAM 해보기
(turtlebot3_gazebo 패키지 참고)

시뮬레이터 Gazebo를 활용한 Navigation 프로그래밍!

10_로봇_로봇.pdf

11_SLAM과_내비게이션.pdf



가상 내비게이션 with Gazebo

- **Gazebo 실행**

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

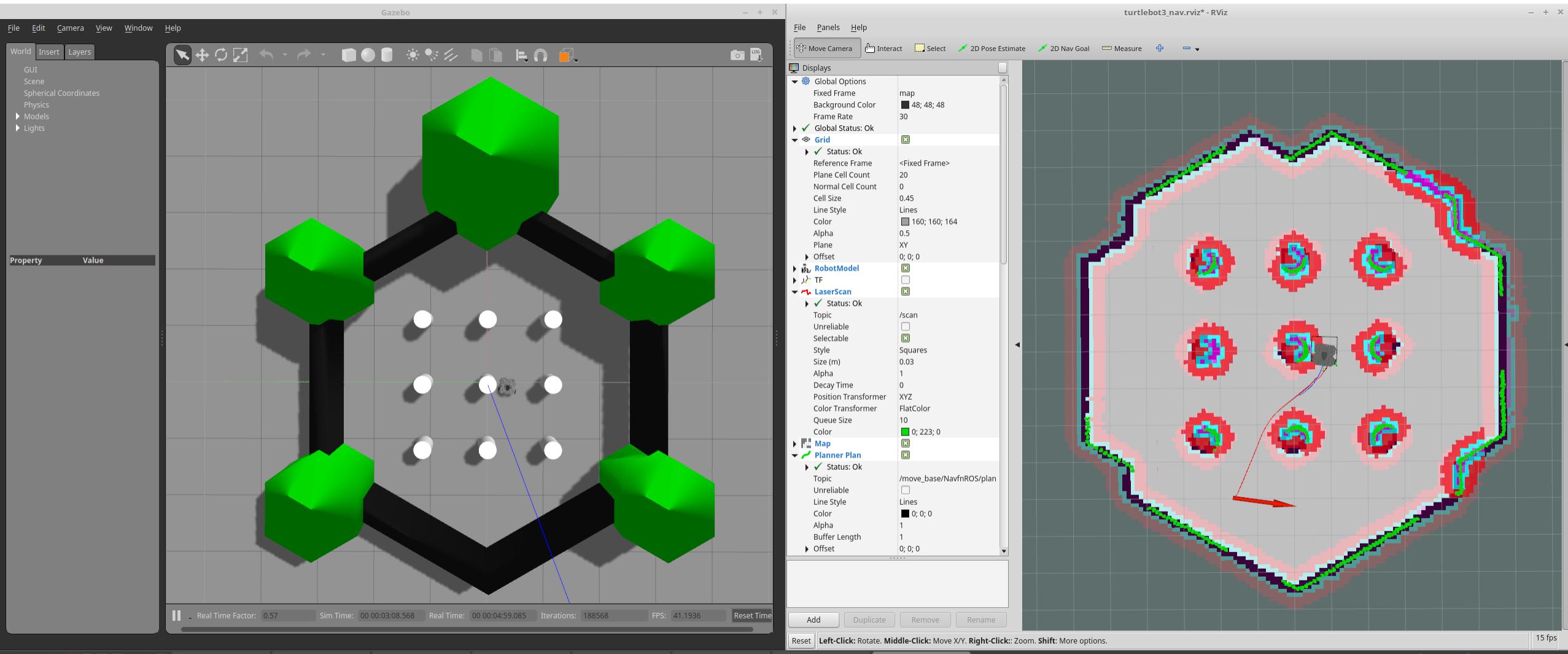
- **Navigation 실행**

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

- **RViz 실행 및 목적지 설정**

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ rosrun rviz rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_nav.rviz
```

가상 내비게이션 with Gazebo



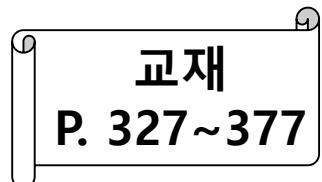
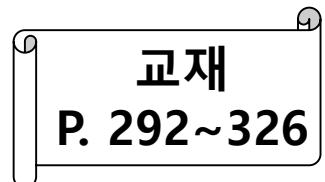
실습 시간

나만의 모바일 로봇을 이용하여
Gazebo상에서 Navigation 해보기
(turtlebot3_gazebo 패키지 참고)

실제 로봇을 활용한 SLAM 실습

10_로봇을 활용한 SLAM.pdf

11_SLAM과 내비게이션.pdf

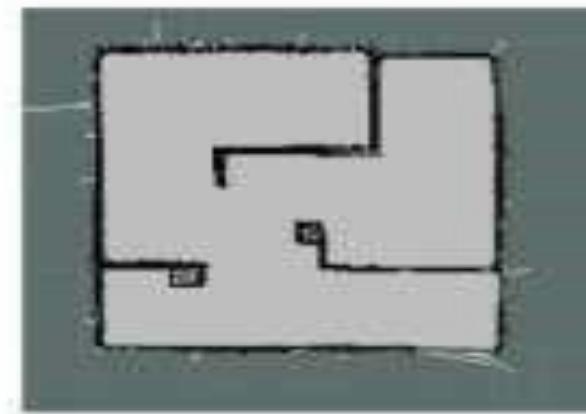


SLAM

TurtleBot³
BURGER



TurtleBot³
WAFFLE



SLAM Demo

SLAM

- [TurtleBot] TurtleBot 실행

```
$ rosrun turtlebot3_bringup turtlebot3_robot.launch
```

- [Remote PC] SLAM 실행

```
$ export TURTLEBOT3_MODEL=burger  
$ rosrun turtlebot3_slam turtlebot3_slam.launch
```

- [Remote PC] RViz 실행

```
$ export TURTLEBOT3_MODEL=burger  
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz
```

- [Remote PC] 터틀봇 원격 조종

```
$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch
```

- [Remote PC] 지도 출력

```
$ rosrun map_server map_saver -f ~/map
```

설정 시간

지도 만들기

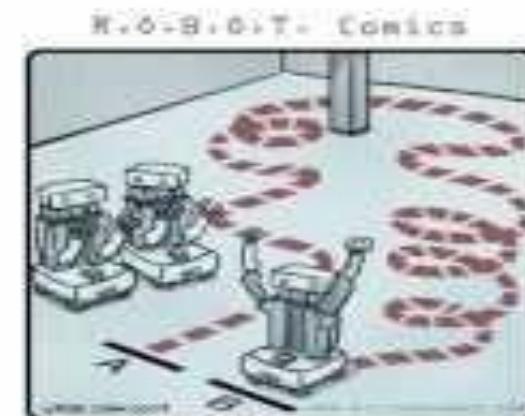
실제 로봇을 활용한 Navigation 실습

10_로봇을_활용한.pdf

11_SLAM과_내비게이션.pdf



Navigation



Navigation Demo

Navigation

- [TurtleBot] TurtleBot 실행

```
$ rosrun turtlebot3_bringup turtlebot3_robot.launch
```

- [Remote PC] 내비게이션 실행

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ rosrun turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

- [Remote PC] RViz 실행 및 목적지 설정

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ rosrun rviz rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_nav.rviz
```

실습 시간

만들어진 지도에서 목표 위치를 선정하여 이동하기

미니 프로젝트

미니 프로젝트를 위한 힌트! #1



Basic
Operation

미니 프로젝트를 위한 힌트! #2



Automatic
parking Demo

실습 시간

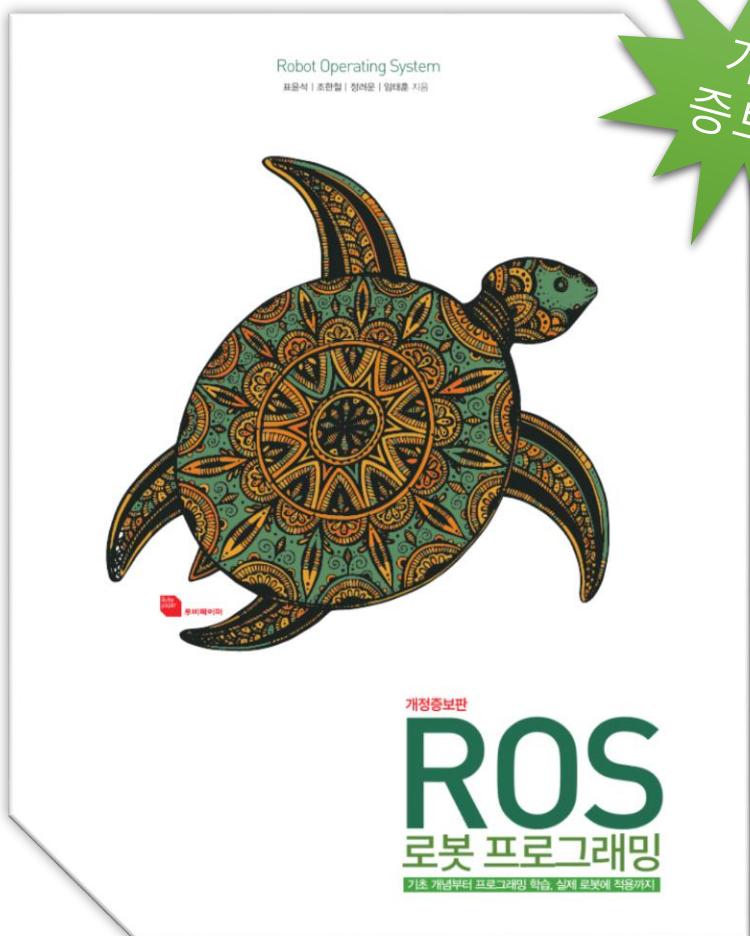
SLAM을 통하여 제작된 지도에서 Navigation 응용하기

TurtleBot3 Example를 참고하여 각자 미니 프로젝트 수행
rqt common plugin / Interactive Markers / Obstacle Detection
Point Operation / Patrol / Automatic Parking

질문 대환영!

* 온라인 상의 질문이라면
오로카 및 로열모로 이용해주세요!

여기서! 광고 하나 나가요~



✓ 한국어판 구매 링크

✓ 4개 언어로 출판!



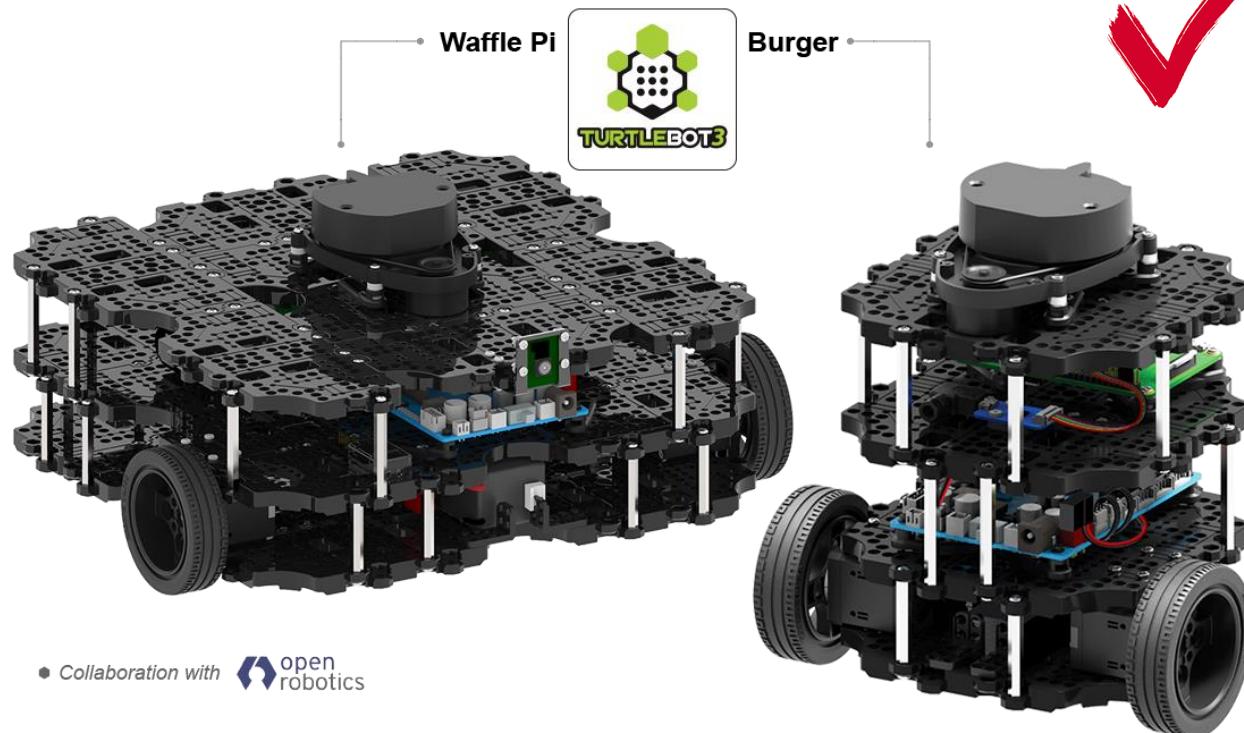
국내 유일! 최초! ROS 참고서!
ROS 공식 플랫폼 **TurtleBot3** 개발팀이
직접 저술한 바이블급 ROS 책

여기서! 광고 둘 나가요~

TURTLEBOT3

인공지능(AI) 연구의 시작,
ROS 교육용 공식 로봇 플랫폼

터틀봇3는 ROS기반의 저가형 모바일 로봇으로
교육, 연구, 제품개발, 취미 등 다양한 분야에서
활용할 수 있습니다.



✓ Direct Link

• Collaboration with 

여기서! 광고 셋 나가요~

오로카



- 오로카
- www.oroca.org
- 오픈 로보틱스 지향
- 풀뿌리 로봇공학의 저변 활성화
- 공개 강좌, 세미나, 프로젝트 진행

- 로봇공학을 위한 열린 모임 (KOS-ROBOT)
- www.facebook.com/groups/KoreanRobotics
- 로봇공학 통합 커뮤니티 지향
- 일반인과 전문가가 어울려지는 한마당
- 로봇공학 정보 공유
- 연구자 간의 협력

- RobotSource
- www.robotsource.org
- 글로벌 로보틱스 커뮤니티 지향
- 로봇공학 정보 공유
- 자신의 로봇 프로젝트 공유
- DIY 로봇 프로젝트 진행

혼자 하기에는 답답하시다고요?

커뮤니티에서 함께 해요~

구
E



표윤석

Yoonseok Pyo
pyo@robotis.com
www.robotpilot.net

www.facebook.com/yoonseok.pyo