LIMO ROS Based Examples

WeGo & LIMO

WeGo

목 차

WeGo

01 ROS 기반 예제

01 ROS 기반 예제

- https://github.com/WeGo-Robotics/limo_examples
- LIMO에서 해당 사이트에 접속하여, 아래에 있는 코드를 받아서 진행
- limo_examples/scripts/ros_based_examples 아래에 있는 코드를 사용
- Python 및 ROS Publisher / Subscriber에 대한 기반 지식 필요
- ROS 기반의 상세한 LIMO에 대한 사용 예시 포함

02 Basic Code

02 Basic Code

- Basic Code
  - 본 내용은 limo_examples/scripts/ros_based_examples/basic_example.py에 해당하는 내용입니다.
  - ROS 기반의 Publisher 및 Subscriber를 하나의 Node를 이용하여 처리하는 방식의 기본을 보여주는 코드입니다.

02 Basic Code

- 첫 번째 형태 (Class 사용 X)
  - 첫 번째 형태는 Class를 사용하지 않고, 단순히 1개의 Publisher와 1개의 Subscriber로 이루어진 형태입니다.
  - 여기서 진행하려는 내용은 Subscriber 내부에서 데이터를 확인하여 처리한 후, 이에 대한 결과로 새로운 Topic을 Publish하는 것입니다.
  - 이를 위해서는 Subscriber에 적용되는 Callback함수 내부에서 Publisher를 사용할 수 있어야합니다.

02 Basic Code

- 첫 번째 형태 (Class 사용 X)
  - 이를 사용하기 위해서, Global 키워드를 사용하여, Publisher를 전역변수처럼 활용할 수도 있지만, 이는 권장하지 않습니다.
  - 대신 Subscriber의 Callback함수에서 Publisher를 인자로 전달받아서 사용할 수 있습니다.

02 Basic Code

- 첫 번째 형태 (Class 사용 X)
  - rospy.Subscriber는 Topic이름, 메시지 타입, 사용하려는 Callback함수로 생성할 수 있지만, 이에 추가적으로 Callback함수에 전달하려는 인자를 함께 넣어서 생성할 수 있습니다.
  - rospy.Subscriber("cmd_vel", Twist, callback, arg1, arg2)
  - 위와 같은 형태로 arg1과 arg2를 callback함수의 인자로 전달할 수 있습니다.

02 Basic Code

- 첫 번째 형태 (Class 사용 X)
  - 여기에서는 추가적으로 전달해주어야하는 인자로 Publisher를 사용합니다.
  - rospy.Subscriber("cmd_vel", Twist, callback, cmd_vel_pub)
  - 위와 같이 작성하면, callback함수는 기본적으로 Subscribe하는 Topic의 데이터와 cmd_vel_pub이라는 인자를 각각 받게 됩니다.
  - def callback(data, publisher)
  - 따라서 위와 같은 형태로, callback함수를 정의해주셔야 합니다.

02 Basic Code

- 첫 번째 형태 (Class 사용 X)
  - 최종 결과물은 코드에서 확인할 수 있으며, 아래의 2번 형태 및 3번 형태 부분을 주석처리하시고, 1번 형태 부분만 주석을 해제하신 후, Turtlesim을 실행하고, 목표하는 x 좌표를 최상단의 TARGET_POSE_X의 값으로 입력한 후, 아래의 명령어를 입력하시면 실행하실 수 있습니다.
  - $ ROS_NAMESPACE=turtle1 rosrun limo_examples basic_example.py
  - 앞에 있는 ROS_NAMESPACE 부분은 코드 실행 시, NAMESPACE를 추가하는 부분으로, 실행되는 Node에 포함된 Topic의 이름 앞에 해당 값을 붙여줍니다.
  - cmd_vel → turtle1/cmd_vel 로 변경되는 효과를 얻을 수 있습니다.

02 Basic Code

- 두 번째 형태 (Class 사용 O, Timer 사용 X)
  - 두 번째 형태는 Class를 사용하여, 1개의 Publisher와 1개의 Subscriber를 연결하는 형태입니다.
  - 앞서 진행한 내용에서 확인하였듯 필요한 내용은 Subscriber에 연결된 Callback함수에서 Publisher에 접근하여 사용할 수 있게 만드는 것입니다.
  - 이에 대한 방법으로 같은 Class의 내부 변수로 Publisher와 Subscriber 및 Callback함수를 정의하여, 상호간 접근이 가능한 형태로 작성할 수 있습니다.

02 Basic Code

- 두 번째 형태 (Class 사용 O, Timer 사용 X)
  - 이와 같은 방법을 통해, Subscriber는 굳이 새로운 인자를 받을 필요 없으며, self 키워드를 이용하여, Publisher에 직접 접근할 수 있습니다.
  - Class 정의 시, 생성자 부분에 실행하는 내용을 모두 정리해주시면 됩니다.
  - 예를 들어 Publisher, Subscriber 정의 등에 대한 내용을 Class 생성자에 적어주시고, Class 메서드에 Callback함수를 정의해주시면 됩니다.

02 Basic Code

- 두 번째 형태 (Class 사용 O, Timer 사용 X)
  - Publisher 정의시에는 필수적으로 self 키워드를 붙여서, 내부 변수로서 사용할 수 있도록 해주셔야 합니다.
  - self.cmd_vel_pub = rospy.Publisher("cmd_vel", Twist, queue_size=5)
  - Subscriber의 Callback함수 지정시에도, self.callback 형태로 입력해주셔야 합니다.
  - rospy.Subscriber("pose", Pose, self.callback)

02 Basic Code

- 두 번째 형태 (Class 사용 O, Timer 사용 X)
  - 마찬가지로 코드 실행 시에는 2번 형태 부분을 제외한 나머지 부분을

    주석처리해주시고, 2번 부분만을 주석을 풀어주신 후 코드를 실행해주시면 됩니다.
  - Turtlesim은 기본적으로 실행된 상태에서 실행해주시면 됩니다.
  - 최상단의 TARGET_POSE_X를 변경하셔서 진행하시면 됩니다.
  - $ ROS_NAMESPACE=turtle1 rosrun limo_examples basic_example.py

02 Basic Code

- 세 번째 형태 (Class 사용 O, Timer 사용 O)
  - 마지막 형태는 Class를 사용하며, 1개의 Publisher와 1개의 Subscriber를 연결하고, 실제 동작을 위해 Timer를 활용하는 형태입니다.
  - Timer의 경우, 연결된 Callback함수를 일정 주기 마다 반복해서 호출해주는 역할로 사용할 수 있습니다.
  - Timer 생성을 위해서는 Callback함수 실행 주기 및 Callback함수를 전달해주셔야합니다.
  - rospy.Timer(rospy.Duration(1.0/30), callback)
  - 위와 같이 작성하면, 1초에 30번, callback함수를 호출하는 형태입니다.

02 Basic Code

- 세 번째 형태 (Class 사용 O, Timer 사용 O)
  - 기존에 사용하던 Subscriber의 Callback에서는 단순히 Class 내부 변수로 데이터를 저장만 진행하고, 실제 동작은 Timer Callback에서 처리하는 형태로 변경해주시면 됩니다.
  - 이 때, Subscriber가 데이터를 먼저 받은 후, Timer가 동작하는 것이 좋으므로, 초기화를 위한 변수를 사용해주는 것이 좋습니다.

02 Basic Code

- 세 번째 형태 (Class 사용 O, Timer 사용 O)
  - 코드 실행 시에는 3번 형태 부분을 제외한 나머지 부분을 주석처리해주시고, 2번 부분만을 주석을 풀어주신 후 코드를 실행해주시면 됩니다.
  - Turtlesim은 기본적으로 실행된 상태에서 실행해주시면 됩니다.
  - 최상단의 TARGET_POSE_X를 변경하셔서 진행하시면 됩니다.
  - $ ROS_NAMESPACE=turtle1 rosrun limo_examples basic_example.py
  -

WeGo

02 Basic Code

- 형태 별 비교
  - 1번 형태는 Class, Timer를 모두 사용하지 않고, Publisher와 Subscriber를 이용하여 동작하는 형태이며, 다수의 데이터를 처리해야하는 경우 부적합할 수 있습니다.
  - 2번 형태는 Class를 사용하지만, Timer는 사용하지 않으므로, 다수의 데이터도 처리할 수 있으나, Subscriber의 Callback에 묶여있으므로, Subscribe하는 Topic의 속도에 무조건 맞춰서 진행이 된다는 문제가 있습니다.
  - 3번 형태는 Class와 Timer를 모두 사용하며, 다수의 데이터 처리가 가능하며, Subscriber의 Callback와 Timer의 Callback이 별개로 동작하며, Timer Callback의 동작 속도를 원하는대로 정하여 사용할 수 있습니다.

03 LIMO Control Example

03 LIMO Control Examples

- LIMO 시스템
  - LIMO의 제어는 일반적인 ROS 기반 로봇과 동일하게, geometry_msgs/Twist 타입의 /cmd_vel Topic을 활용하여 제어할 수 있습니다.
  - 따라서, 단순히 Twist.linear.x와 Twist.angular.z의 두 가지의 값을 바꾸어주면 쉽게 LIMO를 제어할 수 있는 형태입니다.
  - 간단한 Publisher와 Subscriber 코드를 작성하여, LIMO를 제어하는 예제입니다.
  - Basic Code의 2번 형태를 사용합니다.

03 LIMO Control Examples

- 코드 소개
  - 해당 내용은

    limo_examples/scripts/ros_based_examples/move_example.py에 해당하는

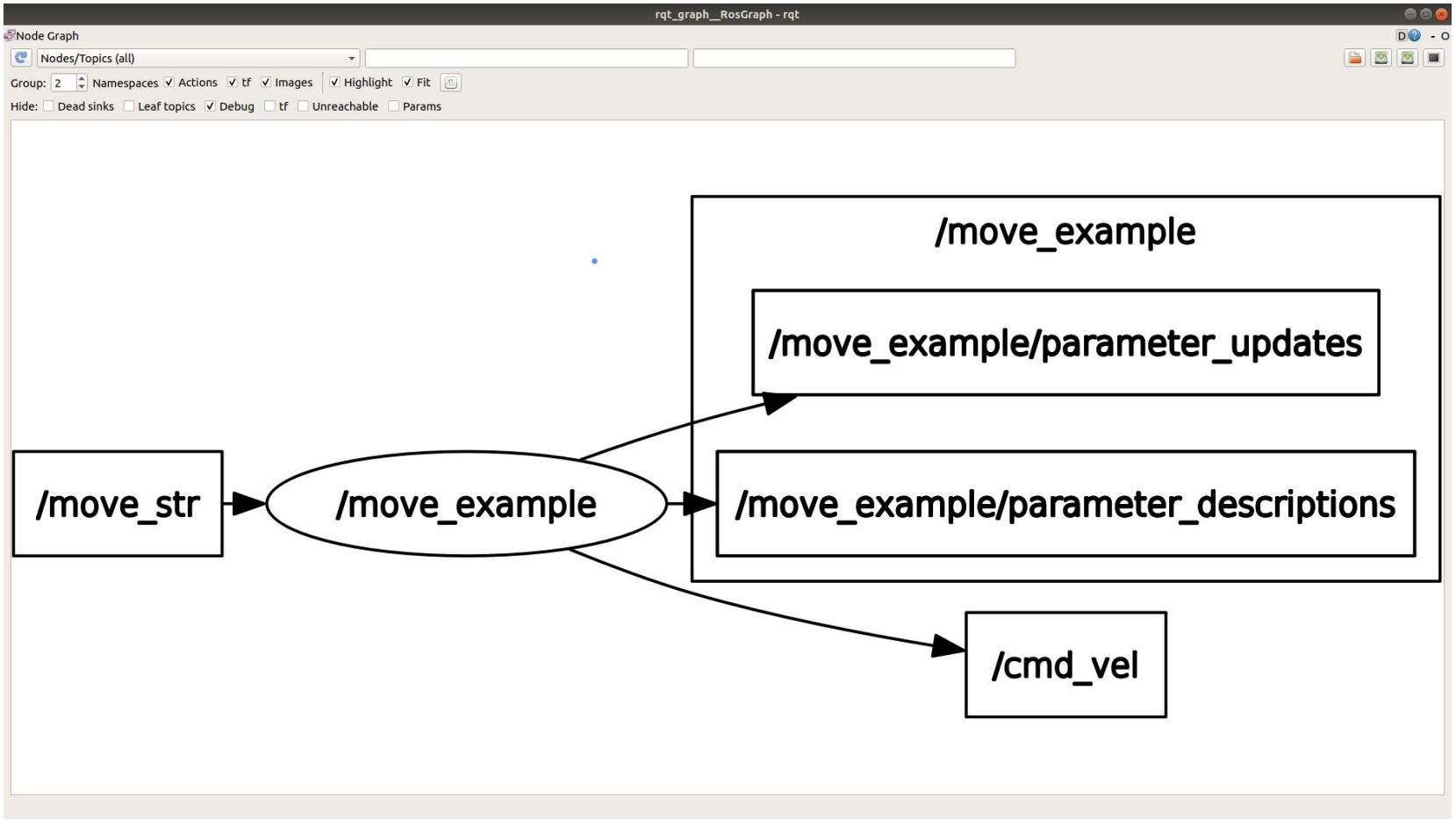
    내용입니다.
  - 해당 코드는 String Type의 “move_str” Topic를 Subscribe하고, Twist 타입의

    “cmd_vel” Topic을 Publish하는 코드입니다.
  - Dynamic_Reconfigure를 이용하여, 전진 속도 및 회전 속도를 실시간으로

    변경할수 있습니다.

03 LIMO Control Examples

- 코드 소개



WeGo

03 LIMO Control Examples

- 코드 소개
  - “move_str” Topic에는 “forward”, “rotate”, “stop” 세 가지의 명령을 내릴 수 있습니다. 이 외의 명령에는 반응하지 않습니다.
  - “forward”에 대해서는 정해진 직선 속도로 이동합니다.
  - “rotate”는 정해진 회전 속도로 회전합니다.
  - “stop”은 멈춥니다.

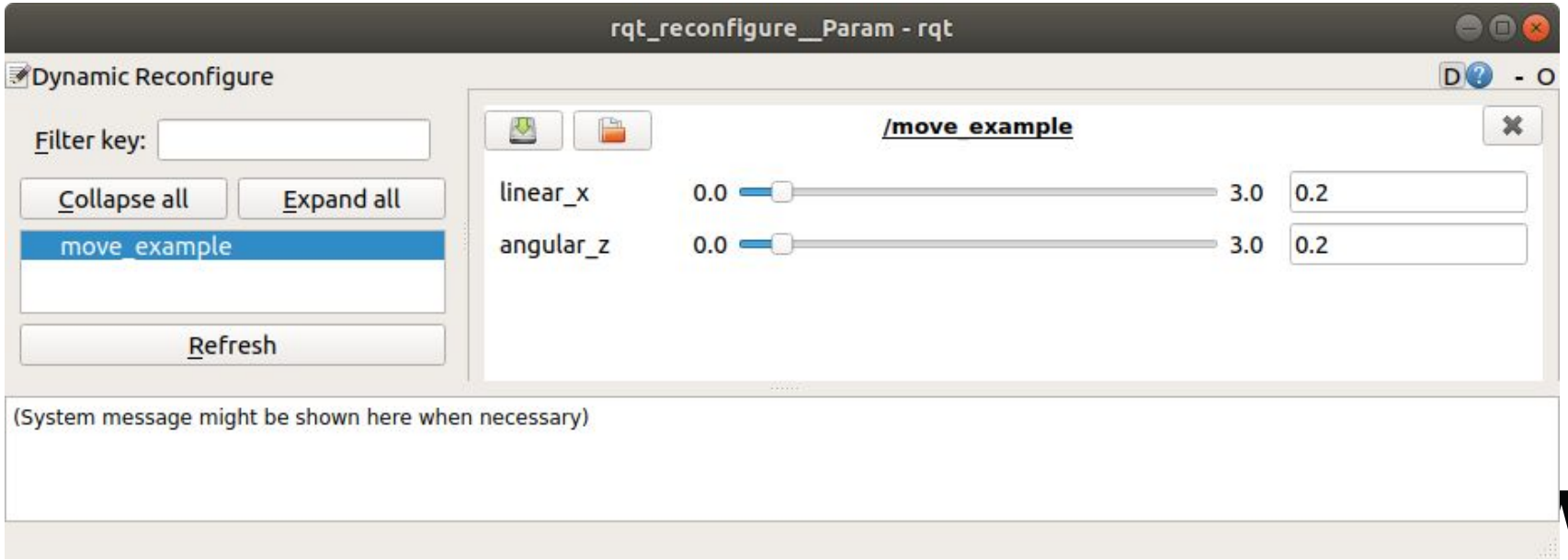
WeGo

03 LIMO Control Examples

- 코드 소개
  - LIMO에서 LIMO Driver 및 Camera Driver를 실행한 상태에서 아래 코드를 새로운 터미널에서 실행합니다.
  - $ rosrun limo_examples move_example.py
  - 새로운 터미널에서 아래 내용을 입력하여 제어할 수 있습니다.
  - $ rostopic pub /move_str std_msgs/String "data: 'forward'"
  - $ rostopic pub /move_str std_msgs/String “data: ‘rotate’”
  - $ rostopic pub /move_str std_msgs/String “data: ‘stop’”
  - 지속적으로 데이터를 보내기 위해서는 -r 5 등의 옵션을 추가합니다.

03 LIMO Control Examples

- 코드 소개
  - 이동하는 속도 및 회전하는 속도의 변경을 위해서는 새로운 터미널을 열어 아래의 내용을 입력합니다.
  - $ rosrun rqt_reconfigure rqt_reconfigure
  - 왼쪽 move_example을 클릭하여 오른쪽에 뜨는 창에서 값을 변경하면 됩니다.



04 LIMO LiDAR Example

WeGo

04 LIMO LiDAR Example

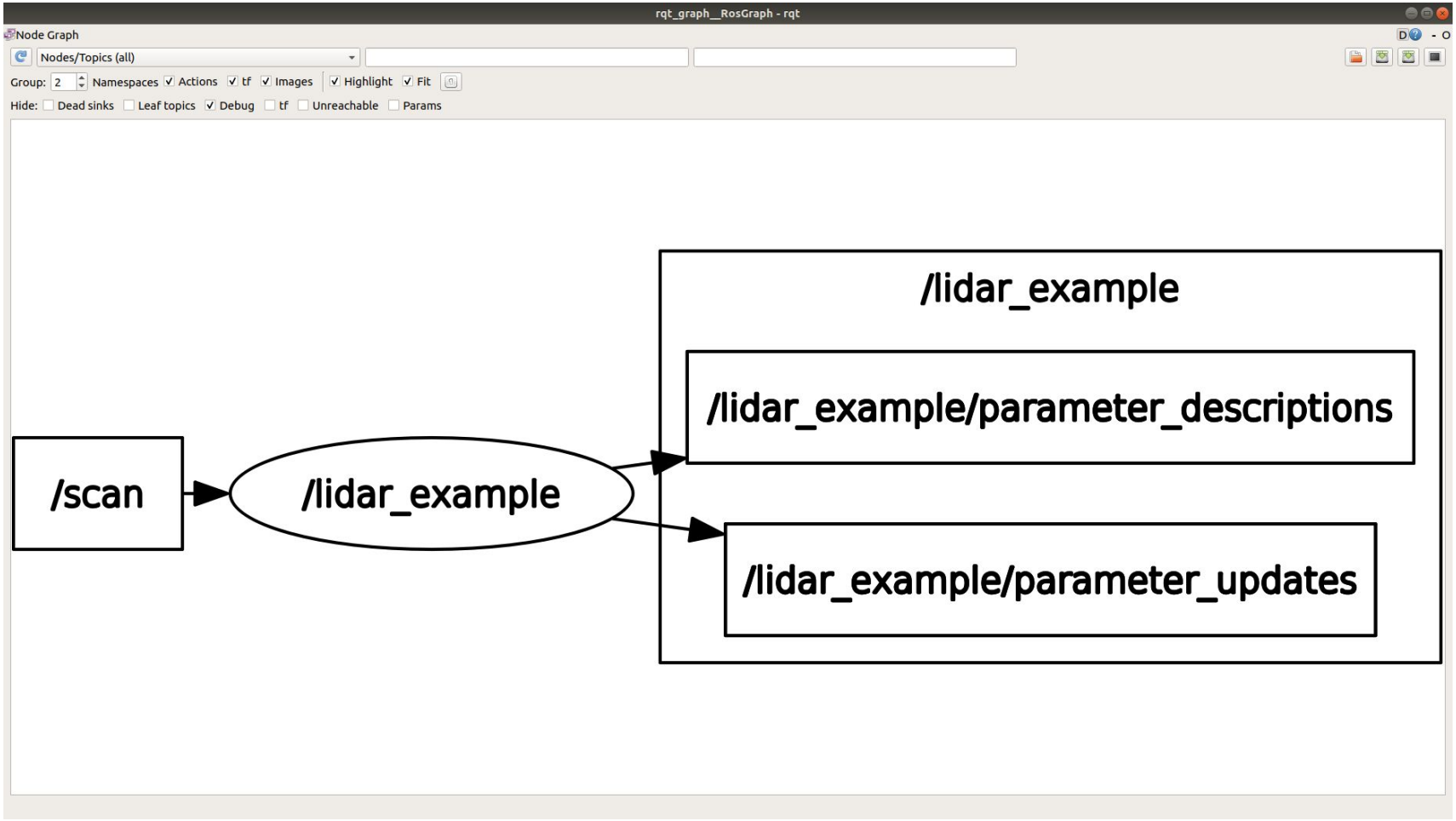
- LIMO LiDAR 소개
  - LIMO에 포함된 LiDAR
    - Frame ID : laser
    - Topic Name : /scan
    - 측정 범위 : -180 ~ 180 degree, 0.1 ~ 12.0 m
    - 약 8 ~ 9Hz로 측정

04 LIMO LiDAR Example

- 코드 소개
  - 해당 코드는 단순히 Subscriber를 활용하여, LiDAR 데이터를 받는 코드입니다.
  - 특정 각도 범위의 데이터를 출력해줍니다.
  - Callback 함수 내부에서 전달 받은 LiDAR Data의 각 Point의 각도를 확인하여, 정해진 범위 내에 포함된 Point인 경우, 해당 각도와 거리를 출력해줍니다.
  - 마찬가지로 Dynamic Reconfigure를 이용하여, 측정하는 범위를 실시간으로 지정해줄 수 있습니다

04 LIMO LiDAR Example

- 코드 소개



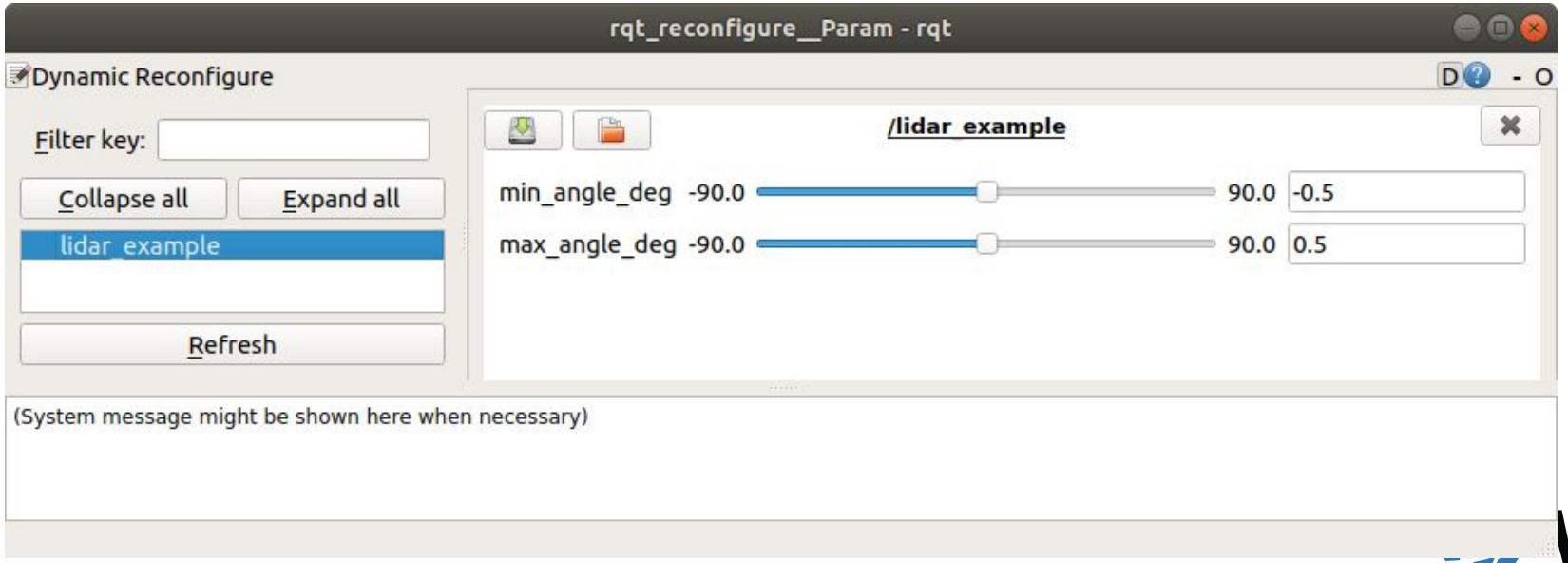
WeGo

04 LIMO LiDAR Example

- 코드 소개
  - LIMO에서 LIMO Driver 및 Camera Driver를 실행한 상태에서 아래 코드를 새로운 터미널에서 실행합니다.
  - $ rosrun limo_examples lidar_example.py

04 LIMO LiDAR Example

- 코드 소개
  - 측정하여 출력하는 범위를 지정하기 위해서는 새로운 터미널을 열어 아래의 명령어를 입력합니다.
  - $ rosrun rqt_reconfigure rqt_reconfigure
  - 왼쪽 lidar_example을 클릭하여 오른쪽에 뜨는 창에서 값을 변경하면 됩니다.



05 LIMO Camera Example

05 LIMO Camera Example

- LIMO Camera 소개
  - LIMO에 포함된 Camera
    - Frame ID : camera_link
    - Topic Name : /camera/ namespace에 있는 모든 Topic

05 LIMO Camera Example

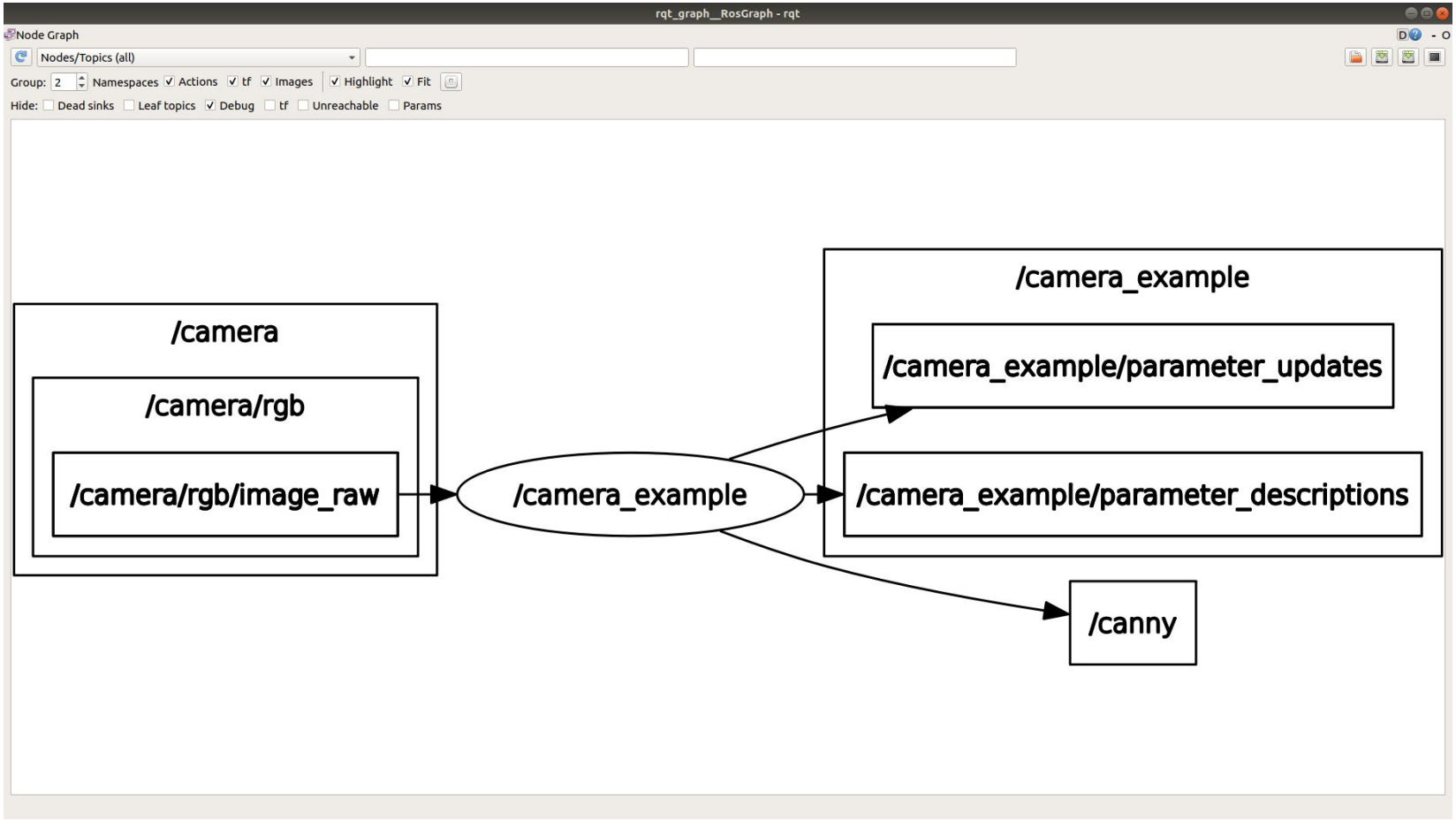
- 코드 소개
  - 해당 코드는 단순히 Subscriber를 활용하여, Camera 데이터를 받는 코드입니다.
  - 데이터를 받은 후, 간단한 Canny Edge Detector를 활용하여, Edge를 검출하여 새로운 Topic으로 Publish합니다.
  - Callback 함수 내부에서 전달받은 Image를 Canny Edge Detector를 이용하여 검출하고, 그 결과를 다시 Image로 만들어서 보내는 형식입니다.
  - 마찬가지로 Dynamic Reconfigure를 이용하여, Edge 검출에 대한 임계값을 실시간으로 변경해줄 수 있습니다

05 LIMO Camera Example

- 코드 소개
  - Image 처리의 경우, ROS에서 사용하는 Image 형식과 일반적으로 영상 처리에 사용하는 OpenCV에서 사용하는 Image 형식이 다르기 때문에, 이에 대한 변환이 필요합니다.
  - 해당 내용은 CV_Bridge에서 처리해주며, cvbridge의 cv2_to_imgmsg 또는 imgmsg_to_cv2를 통해서 변경할 수 있습니다.

05 LIMO Camera Example

- 코드 소개

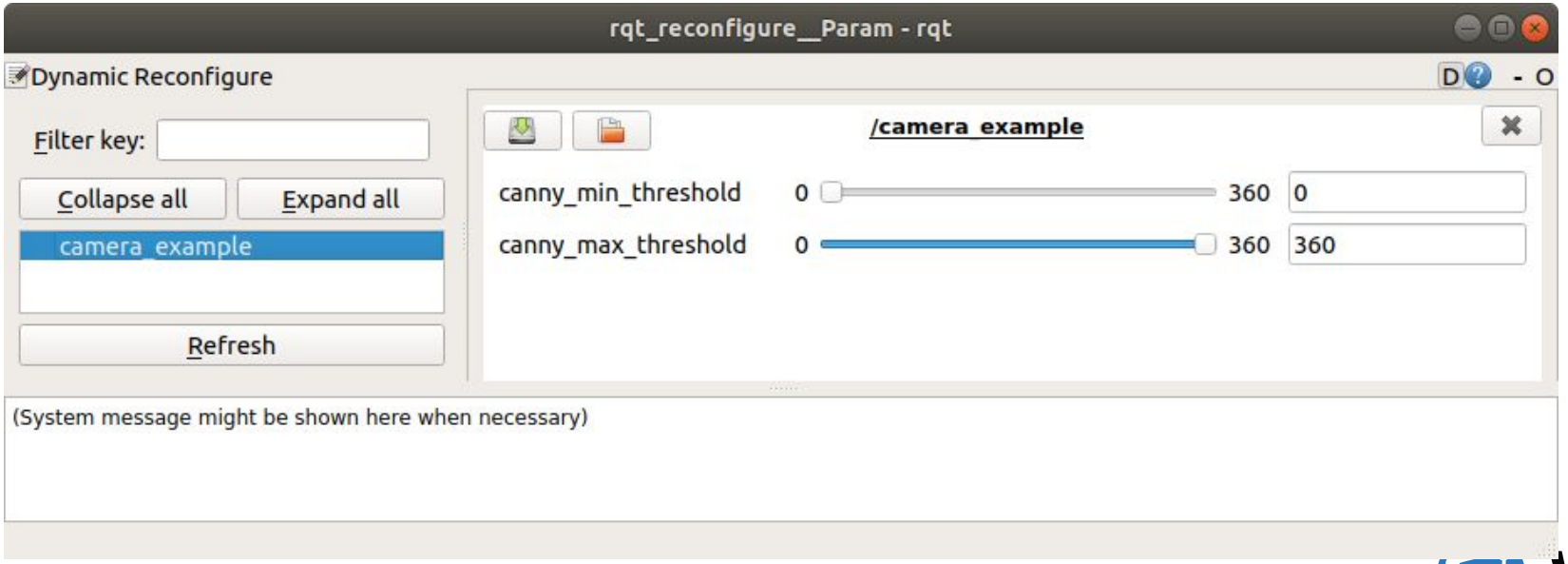


05 LIMO Camera Example

- 코드 소개
  - LIMO에서 LIMO Driver 및 Camera Driver를 실행한 상태에서 아래 코드를 새로운 터미널에서 실행합니다.
  - $ roslaunch limo_examples camera_example.py

05 LIMO Camera Example

- 코드 소개
  - Canny Edge Detector의 동작 파라미터를 지정하기 위해서는 새로운 터미널을 열어 아래의 명령어를 입력합니다.
  - $ rosrun rqt_reconfigure rqt_reconfigure
  - 왼쪽 camera_example을 클릭하여 오른쪽에 뜨는 창에서 값을 변경하면 됩니다.



WeGo

05 LIMO Camera Example

- 코드 소개
  - 결과 확인을 위해 새로운 터미널에서 다음 명령어를 입력합니다.
  - $ rosrun rqt_image_view rqt_image_view /canny
  - 새롭게 뜨는 창에서 /canny Topic으로 출력되는 결과를 확인할 수 있습니다.

06 LIMO LiDAR + Control Example

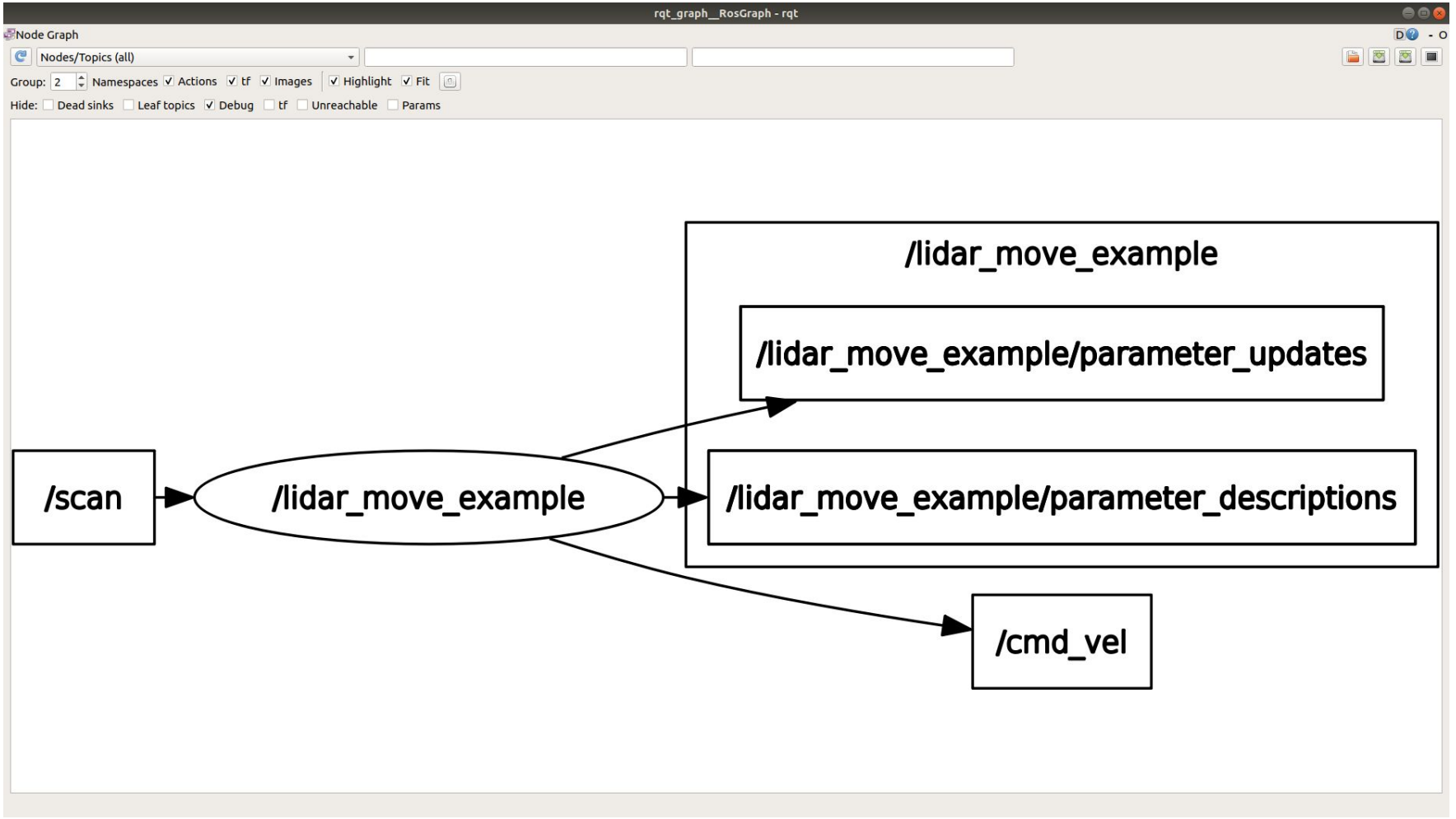
WeGo

06 LIMO LiDAR + Control Example

- 코드 소개
  - 해당 코드는 2번 코드의 형태를 사용하여, LiDAR Data를 Subscribe 하고, LIMO 제어를 위한 Data를 Publish하는 코드입니다.
  - LiDAR Data를 받아서 일정 범위 내부에 물체가 존재할 경우, 이동하지 않고 정지하며, 존재하지 않을 경우 직진하는 코드입니다.
  - 마찬가지로 Dynamic Reconfigure를 이용하여, 물체 존재 여부를 확인할 범위 (LiDAR 각도 범위 및 거리)를 지정할 수 있으며, 이동 속도에 해당하는 부분도 실시간으로 변경해줄 수 있습니다.

06 LIMO LiDAR + Control Example

- 코드 소개

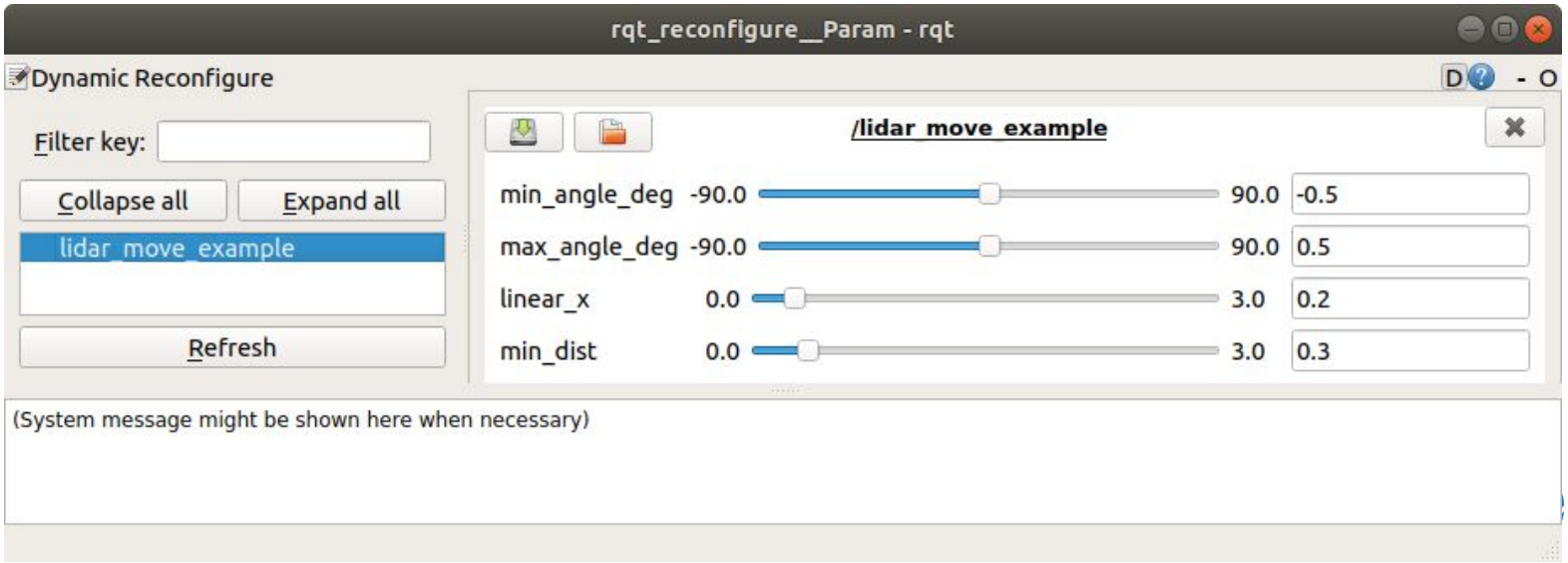


06 LIMO LiDAR + Control Example

- 코드 소개
  - LIMO에서 LIMO Driver 및 Camera Driver를 실행한 상태에서 아래 코드를 새로운 터미널에서 실행합니다.
  - $ rosrun limo_examples lidar_move_example.py

06 LIMO LiDAR + Control Example

- 코드 소개
  - 정지에 사용할 LiDAR의 범위 및 거리 값을 변경하기 위해서는 새로운 터미널에서 아래 명령어를 입력하여 변경할 수 있습니다.
  - $ rosrun rqt_reconfigure rqt_reconfigure
  - 왼쪽 lidar_move_example을 클릭하여 오른쪽에 뜨는 창에서 값을 변경하면 됩니다.



WeGo

WeGo Robotics

Tel. 031 – 229 – 3553

Fax. 031 – 229 – 3554

제품 문의: go.sales@wego-robotics.com

기술 문의: go.support@wego-robotics.com