Turtlebot3

이동

배치파일 확인

- 배치파일 확인

```
$ code ~/.bashrc
```

```
export ROS_DOMAIN_ID=30
export TURTLEBOT3_MODEL=waffle_pi
export LDS_MODEL=LDS-02
source /opt/ros/foxy/setup.bash
source ~/Workspaces/ros2_ws/install/setup.bash
```

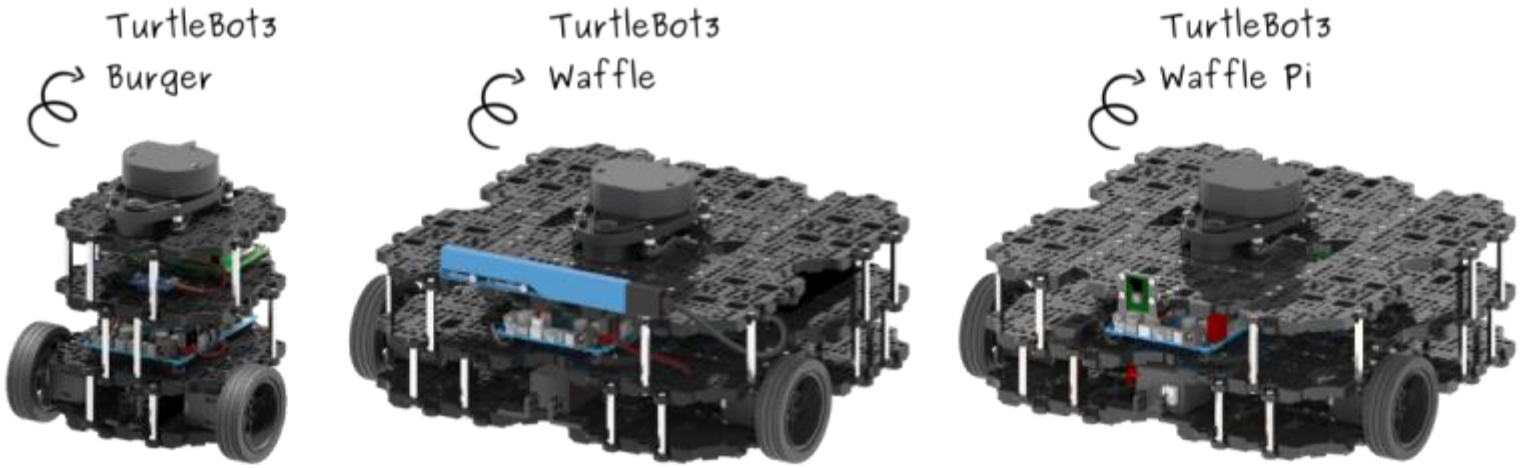
LDS-01 

LDS-02 

- 배치파일 적용

```
$ source ~/.bashrc
```

Turtlebot3 Model



Turtlebot3

이동

CLI 실습

[teleop_turtle]

Publisher
메시지 발행

[/turtle1/cmd_vel]

geometry_msgs/msg/Twist
메시지

[turtlesim]

Subscriber
메시지 구독

[geometry_msgs/msgs/Twist]

Vector3 linear
Vector3 angular

[geometry_msgs/msgs/Vector3]

float64 x
float64 y
float64 z

[geometry_msgs/msgs/Vector3]

float64 x
float64 y
float64 z

출처: https://velog.io/@dumok

토픽 /cmd_vel

로봇의 병진 및 회전 속도 제어
명령을 내릴 때 사용하는 토픽

- 토픽 리스트 확인

```
$ ros2 topic list
```

- 토픽 타입 확인

```
$ ros2 topic type /cmd_vel
```

- 인터페이스 확인

```
$ ros2 interface show geometry_msgs/msg/Twist
```

- 인터페이스의 기본 타입 표시

```
$ ros2 interface proto geometry_msgs/msg/Twist
```

geometry_msgs/msg/Twist

변수명	단위	설명
Vector3 **linear**	m/s	x, y, z축 방향으로의 병진속도(linear)
Vector3 **angular**	rad/s	x, y, z축에 대한 회전속도(angular)

geometry_msgs/msg/Twist

변수명	단위	설명
float64 **x**		x축
float64 **y**		y축
float64 **z**		z축

https://github.com/ros2/common_interfaces/blob/foxy/geometry_msgs/msg/Twist.msg

토픽 /cmd_vel 발행(publish)

- [창1] Turtlebot3 Gazebo 실행

```
$ ros2 launch turtlebot3_gazebo empty_world.launch.py
```

- [창2] 토픽 정보 확인

```
$ ros2 topic info /cmd_vel
```

- [창3] 토픽 내용 확인

```
$ ros2 topic echo /cmd_vel
```

- [창4] 토픽 발행

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

Turtlebot3 이동

- [창4] 토픽 발행 - 앞으로

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.2, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

- [창4] 토픽 발행 - 뒤으로

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear: {x: -0.2, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

- [창4] 토픽 발행 - 정지

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

Turtlebot3 회전

- [창4] 토픽 발행 - 왼쪽으로

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear:
{x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}' --once
```

- [창4] 토픽 발행 - 오른쪽으로

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear:
{x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: -0.2}}' --once
```

- [창4] 토픽 발행 - 정지

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear:
{x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

Turtlebot3

이동

Code 실습

move_pkg

- 패키지 생성

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create move_pkg --build-type ament_python
  --dependencies rclpy geometry_msgs
```

- 코드 생성

```
$ cd move_pkg/move_pkg
$ code go.py
$ code stop.py
```

- 설정파일 수정

```
$ cd ..
$ code setup.py
```

Turtlebot 앞으로 가기 src/move_pkg/move_pkg/go.py

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
#https://github.com/ros2/common_interfaces/blob/foxy/geometry_msgs/msg/Twist.msg

class Go(Node):

    def __init__(self):
        super().__init__('go_node')
        self.counter = 0
        self.pub = self.create_publisher(Twist, 'cmd_vel', 10)
        self.timer = self.create_timer(0.1, self.pub_cb)

    def pub_cb(self):
        msg = Twist()
        msg.linear.x = 0.2

        self.pub.publish(msg)
        self.get_logger().info('Published message: ' + str(msg.linear.x))
```

Turtlebot 앞으로 가기 src/move_pkg/move_pkg/go.py

```
def main(args=None):
    rclpy.init(args=args)
    node = Go()

    try:
        rclpy.spin_once(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()


if __name__ == '__main__':
    main()
```

Turtlebot 멈추기

src/move_pkg/move_pkg/stop.py

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
#https://github.com/ros2/common_interfaces/blob/foxy/geometry_msgs/msg/Twist.msg

class Stop(Node):

    def __init__(self):
        super().__init__('stop_node')
        self.counter = 0
        self.pub = self.create_publisher(Twist, 'cmd_vel', 10)
        self.timer = self.create_timer(0.1, self.pub_cb)

    def pub_cb(self):
        msg = Twist()
        msg.linear.x = 0.0

        self.pub.publish(msg)
        self.get_logger().info('Published message: ' + str(msg.linear.x))
```

Turtlebot 멈추기

src/move_pkg/move_pkg/stop.py

```
def main(args=None):
    rclpy.init(args=args)
    node = Stop()

    try:
        rclpy.spin_once(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()


if __name__ == '__main__':
    main()
```

설정파일 수정

src/move_pkg/setup.py

```
'console_scripts': [
    'go = move_pkg.go:main',
    'stop = move_pkg.stop:main'
],
```

빌드

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select move_pkg
$ source install/setup.bash
```

노드 실행(ros2 run)

```
$ ros2 pkg executables move_pkg
$ ros2 run move_pkg go
$ ros2 run move_pkg stop
```

빌드 & 실행

- 빌드

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select
move_pkg
```

- 설정 적용

```
$ source install/setup.bash
```

- 패키지 확인

```
$ ros2 pkg executables move_pkg
```

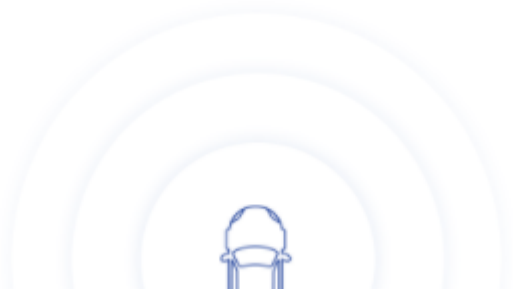
- 코드 실행

```
$ ros2 run move_pkg go
$ ros2 run move_pkg stop
```

Turtlebot3
360° LiDAR
CLI 실습

LiDAR

- LiDAR(Light Detection and Ranging)
  - LiDAR는 '빛 감지 및 거리 측정'의 약어
  - 레이저 빔을 사용하여 환경 내 정확한 거리와 움직임을 실시간으로 측정하는 원격 감지 기술

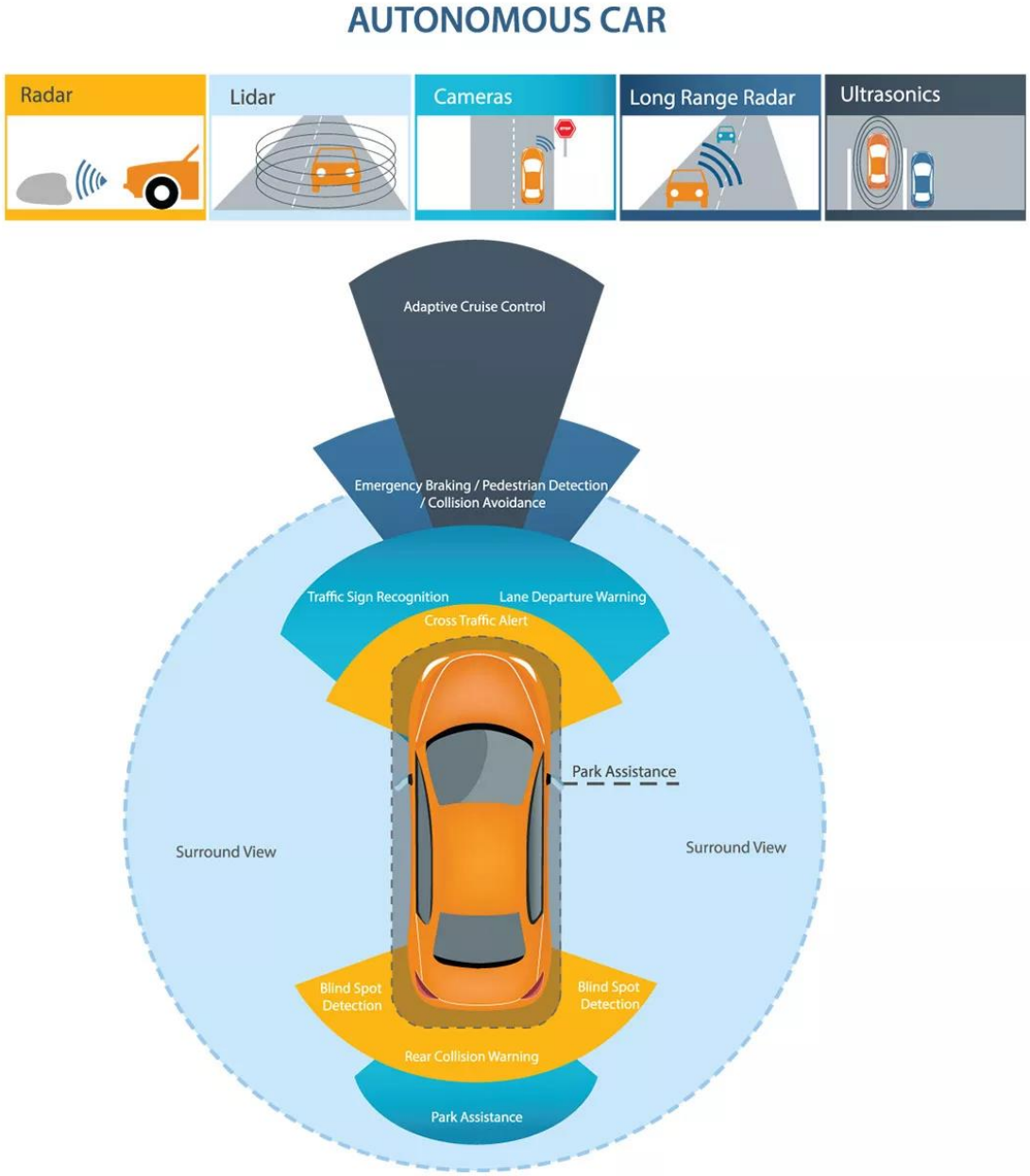


1. 라이다 센서가
주변 환경으로
빛을 방출합니다.
(레이저 스캐너)

2. 이 빛은
주변의 물체를 맞고
센서로 돌아옵니다.
(LiDAR 센서)

3. 각각의 빛이
센서로 돌아온
시간을 사용하여
물체와 센서 간의
거리를 계산합니다.
(LiDAR 포인트 클라우드)

CAR



출처: https://www.synopsys.com/ko-kr/glossary/what-is-lidar.html#D

LiDAR

출처: https://youtu.be/NZKvf1cXe8s?feature=shared

LDS-01 Turtlebot3 LiDAR

360도 수평 범위에서 2D 레이저 스캔을 수행함

Items	Specifications
Sampling Rate	1.8kHz
Distance Range	120 ~ 3,500mm
Distance Accuracy(120mm ~ 499mm)	±15mm
Distance Accuracy(500mm ~ 3,500mm)	±5.0%
Distance Precision(120mm ~ 499mm)	±10mm
Distance Precision(500mm ~ 3,500mm)	±3.5%
Scan Rate	300±10 rpm
Angular Range	360°
Angular Resolution	1°

Sampling Rate(샘플링 속도): 센서가 환경에서 데이터를 수집하는 빈도, 샘플의 수
Distance Range(거리 범위): 센서가 측정할 수 있는 최소~최대 거리
Scan Rate(스캔 속도): 라이다 빔이 주변 환경을 촬영하거나 샘플링하는 빈도
Angular Range(각도 범위): 라이다 센서가 얼마나 넓은 각도를 스캔할 수 있는지 나타냄
Angular Resolution(각도 해상도): 라이다 센서가 얼마나 작은 간격으로 데이터를 수집하는지 나타냄

출처: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/

LDS-02 Turtlebot3 LiDAR

360도 수평 범위에서 2D 레이저 스캔을 수행함

Items	Specifications
Sampling Rate	2.3kHz (Fixed)
Distance Range	160 ~ 8,000mm
Distance Accuracy(160 ~ 300 mm)	±10mm
Distance Accuracy(300 ~ 6,000 mm)	±3.0%
Distance Precision(6,000 ~ 8,000 mm)	±5.0%
1 Scan Frequency	5Hz or above
Angular Range	360 °
2 Angular Resolution	1 °

Distance Accuracy(거리 정확도): 측정된 거리가 실제 거리와 얼마나 가까운지
값이 작을수록 센서가 더 정확하게 거리를 측정한다는 의미
Distance Precision(거리 정밀도): 여러 번의 측정 중에서 동일한 조건에서
얼마나 일관되게 거리를 측정할 수 있는지
표준 편차와 같은 통계적 지표를 사용하며, 값이 작을수록 거리 측정이 더 정밀함을 의미

출처: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_02/

토픽 /scan

- 토픽 리스트 확인

```
$ ros2 topic list
```

- 토픽 타입 확인

```
$ ros2 topic type /scan
```

- 인터페이스 확인

```
$ ros2 interface show sensor_msgs/msg/LaserScan
```

- 인터페이스의 기본 타입 표시

```
$ ros2 interface proto sensor_msgs/msg/LaserScan
```

sensor_msgs/msg/LaserScan

변수명	단위	설명
std_msgs/Header **header**		unint32 seq, time stamp, string frame_id time은 스캔하여 데이터를 획득한 시간
float32 **angle_min**	rad	스캔의 시작 각도
float32 **angle_max**	rad	스캔의 끝 각도
float32 **angle_increment**	rad	측정되는 각의 단위 예) 0.01은 0.01라디안 마다 측정됨
float32 **time_increment**	sec	측정되는 시간의 단위
float32 **scan_time**	m	측정되는 시간 간격
float32 **range_min**	m	최소 range(거리) 값
float32 **range_max**	m	최대 range(거리) 값
float32[] **ranges**	m	최소 각도~최대 각도까지 단위 각도씩 분할한 해당 각도에서 측정한 거리 예) 0.1°씩 측정하고 최소각도가 0이면 5번째 요소는 0+0.1*5=0.5°이며, range[5]의 값은 0.5°에서 측정된 물체까지의 거리 값
float32[] **intensities**	m	레이저가 부딪혀 돌아오는 강도

https://github.com/ros2/common_interfaces/blob/foxy/sensor_msgs/msg/LaserScan.msg

토픽 /scan 확인

- [창1] Turtlebot3 Gazebo 실행

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

- [창2] 토픽 정보 확인

```
$ ros2 topic info /scan
```

- [창3] 토픽 내용 확인

```
$ ros2 topic echo /scan
```

- [창4] 토픽 전송 주기 확인

```
$ ros2 topic hz /scan
```

Turtlebot3
360° LiDAR
Code 실습

info_pkg

- 패키지 생성

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create info_pkg --build-type ament_python
  --dependencies rclpy sensor_msgs
```

- 코드 생성

```
$ cd info_pkg/info_pkg
$ code scan_info.py
```

- 설정파일 수정

```
$ cd ..
$ code setup.py
```

설정파일 수정

src/move_pkg/setup.py

```
'console_scripts': [
    'scan_info = info_pkg.scan_info:main'
],
```

LiDAR 센싱 정보

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import LaserScan

class ScanInfo(Node):

    def __init__(self):
        super().__init__('scan_info_node')
        self.sub = self.create_subscription(LaserScan, 'scan', self.sub_cb, 10)
        self.get_logger().info('ScanInfo Node Running...')

    def sub_cb(self, msg):
        self.get_logger().info('Received message: ')

        print('time stamp = ' + str(msg.header.stamp))
        print('angle_min = ' + str(msg.angle_min) + ' rad')
        print('angle_max = ' + str(msg.angle_max) + ' rad')
        print('angle_increment = ' + str(msg.angle_increment) + ' rad')

        print('time_increment = ' + str(msg.time_increment) + ' seconds')
        print('scan_time = ' + str(msg.scan_time) + ' seconds')

        # Distance Range(120 ~ 3,500mm)
        print('range_min = ' + str(msg.range_min) + ' m')
        print('range_max = ' + str(msg.range_max) + ' m')
```

```
 # Angular Range(360 degrees), Angular Resolution(1
degree)
        print('ranges len = ' + str(len(msg.ranges)))
        print('intensities len = ' + str(len(msg.intensities)))

        print('ranges - 0 degree = ' + str(msg.ranges[0]))
        print('ranges - 90 degree = ' + str(msg.ranges[90]))
        print('ranges - 180 degree = ' + str(msg.ranges[180]))
        print('ranges - 270 degree = ' + str(msg.ranges[270]))

def main(args=None):
    rclpy.init(args=args)
    node = ScanInfo()

    try:
        rclpy.spin_once(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()


if __name__ == '__main__':
    main()
```

빌드 & 실행

- 빌드

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select info_pkg
```

- 설정 적용

```
$ source install/setup.bash
```

- [창1] 가즈보 실행

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

- [창2] 코드 실행

```
$ ros2 run info_pkg scan_info
```

실습 과제

월드 초기화

- [창1] Turtlebot3 Gazebo 실행

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

- [창2] Turtlebot3 멈추기

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}' --once
```

- [창3] world 초기화

```
$ ros2 service call /reset_world std_srvs/srv/Empty
```

- [창4] 토픽 확인

```
$ rqt
```

TurtleBot3 앞으로 가다가 멈추기

과제1

TurtleBot3가 앞으로 가다가 벽을 만나면 멈추는 프로그램을 작성하시오.

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```