

라이다 기반 장애물 회피 주행

프로그래머스 자율주행 코스 1기 조정민

라이다를 사용하여 장애물이 있으면 정지했다가 다시 주행하는 기능 구현

```
#!/usr/bin/env python
import rospy, time
from sensor_msgs.msg import LaserScan
from xycar_motor.msg import xycar_motor
motor_msg = xycar_motor()
distance = []
def callback(data):
    global distance, motor_msg
    distance = data.ranges
def drive_go():
    global motor_msg
    motor_msg.speed = 5
    motor_msg.angle = 0
    pub.publish(motor_msg)
def drive_stop():
    global motor_msg
    motor_msg.speed = 0
    motor_msg.angle = 0
    pub.publish(motor_msg)
```

라이다를 사용하여 장애물이 있으면 정지했다가 다시 주행하는 기능 구현

```
rospy.init_node('lidar_driver')
rospy.Subscriber('/scan', LaserScan, callback, queue_size = 1)
pub = rospy.Publisher('xycar_motor', xycar_motor, queue_size=1)

time.sleep(3) #ready to connect lidar

while not rospy.is_shutdown():
    ok = 0
    for degree in range(60,120):
        if distance[degree] <= 0.3:
            ok += 1
        if ok > 3:
            drive_stop()
            break
    if ok <= 3:
        drive_go()
```

장애물이 있으면 일단 후진한 뒤에 핸들을 꺾어 다른 방향으로 전진하는 방법 등으로 장애물을 피해서 계속 주행하는 기능 구현

```
#!/usr/bin/env python
```

```
 -*- coding:utf-8 -*-
```

```
import rospy, time
```

```
from sensor_msgs.msg import LaserScan
```

```
from xycar_motor.msg import xycar_motor
```

```
motor_msg = xycar_motor()
```

```
distance = []
```

```
def callback(data):
```

```
    global distance, motor_msg
```

```
    distance = data.ranges
```

```
def drive_go():
```

```
    global motor_msg
```

```
    motor_msg.speed = 5
```

```
    motor_msg.angle = 0
```

```
    pub.publish(motor_msg)
```

```
def drive_opposite(degree):
```

```
    global motor_msg
```

```
    # 후진
```

```
    motor_msg.speed = -5
```

```
    motor_msg.angle = 0
```

```
    for _ in range(20):
```

```
        pub.publish(motor_msg)
```

```
    # 반대방향 주행
```

```
    diff = 90 - degree
```

```
    motor_msg.speed = 5
```

```
    if(diff > 0):
```


```
        motor_msg.angle = 30
```

```
    else:
```

```
        motor_msg.angle = -30
```

```
    for _ in range(20):
```

```
        pub.publish(motor_msg)
```



장애물이 있으면 일단 후진한 뒤에 핸들을 꺾어 다른
방향으로 전진하는 방법 등으로 장애물을 피해서 계속
주행하는 기능 구현

```
rospy.init_node('lidar_driver')
rospy.Subscriber('/scan', LaserScan, callback, queue_size = 1)
pub = rospy.Publisher('xycar_motor', xycar_motor, queue_size=1)

time.sleep(3) #ready to connect lidar

while not rospy.is_shutdown():
    ok = 0
    for degree in range(60,120):
        if distance[degree] <= 0.3:
            ok += 1
        if ok > 3:
            drive_opposite(degree)
            break
    if ok <= 3:
        drive_go()
```