특화 일지

2021년 8월 27일 금요일　　오후 1:29

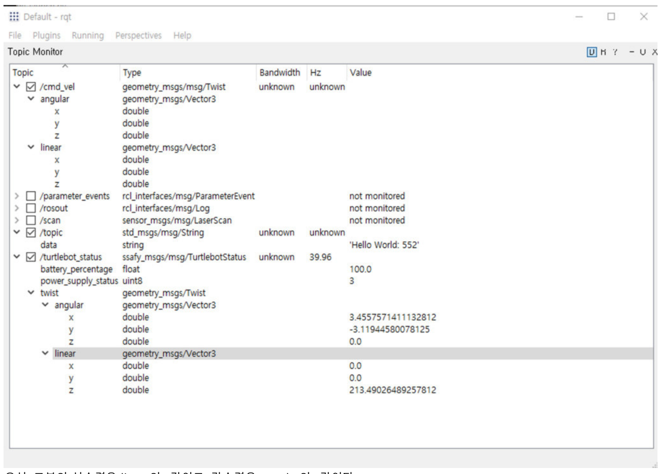
사전 학습

이거 2개는 핵심이다 핵심.... 빌드할때도 실행해주고 꼭 해주자
//ros2를 사용하기 위한 배치
//빌드한 노드를 실행하기 위한 배치

```
call C:₩dev₩ros2_eloquent₩setup.bat
call C:₩Users₩dlwlgns₩Desktop₩catkin_ws₩install₩local_setup.bat
```

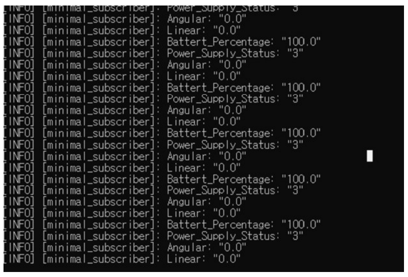
1강
launch 파일이 있는 경로로 이동 후
시뮬레이터와 연결하는 명령어

```
ros2 launch ssafybridge_launch.py
```

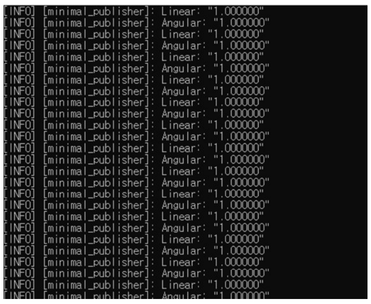


Topic	Type	Bandwidth	Hz	Value
/cmd_vel	geometry_msgs/msg/Twist	unknown	unknown	
angular	geometry_msgs/Vector3			
x	double			
y	double			
z	double			
linear	geometry_msgs/Vector3			
x	double			
y	double			
z	double			
/parameter_events	rcl_interfaces/msg/ParameterEvent			not monitored
/rosout	rcl_interfaces/msg/Log			not monitored
/scan	sensor_msgs/msg/LaserScan			not monitored
/topic	std_msgs/msg/String	unknown	unknown	
data	string			'Hello World: 552'
/turtlebot_status	ssafy_msgs/msg/TurtlebotStatus	unknown	39.96	
battery_percentage	float			100.0
power_supply_status	uint8			3
twist	geometry_msgs/Twist			
angular	geometry_msgs/Vector3			
x	double			3.4557571411132812
y	double			-3.11944580078125
z	double			0.0
linear	geometry_msgs/Vector3			
x	double			0.0
y	double			0.0
z	double			213.49026489257812

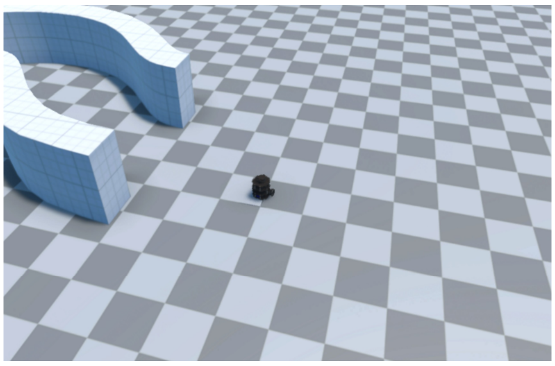
우선, 로봇의 선속력은 linear의 x값이고, 각속력은 angular의 z값이다.
코드는 my_package에서 만들었던 subscriber와 publisher를 참고하여 작성하였다.



```
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
[INFO] [minimal_subscriber]: Battert_Percentage: "100.0"
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
[INFO] [minimal_subscriber]: Battert_Percentage: "100.0"
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
[INFO] [minimal_subscriber]: Battert_Percentage: "100.0"
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
[INFO] [minimal_subscriber]: Battert_Percentage: "100.0"
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
[INFO] [minimal_subscriber]: Battert_Percentage: "100.0"
[INFO] [minimal_subscriber]: Power_Supply_Status: "3"
[INFO] [minimal_subscriber]: Angular: "0.0"
[INFO] [minimal_subscriber]: Linear: "0.0"
```



```
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
[INFO] [minimal_publisher]: Linear: "1.000000"
[INFO] [minimal_publisher]: Angular: "1.000000"
```



사진이지만 q 눌러서 오토모드로 혼자서 계속 돌고있다

2강
특정 패키지만 빌드하려면

```
colcon build --packages-select 패키지이름
```

ssafybridge 런치를 실행한 상태에서 odom_pub 를 실행
rviz에 odom 토픽 추가

Subscriber

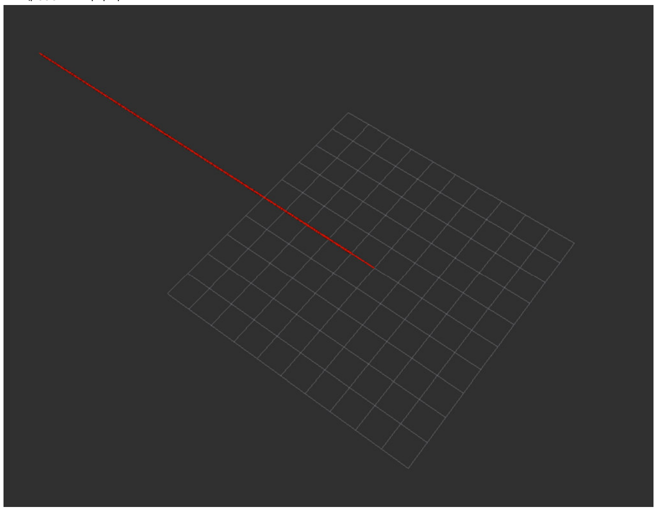
```
import rclpy
from rclpy.node import Node
from ssafy_msgs.msg import TurtlebotStatus #타
입 이름 가져오기


class MinimalSubscriber(Node):
    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscr
iption(
            TurtlebotStatus, #타입 이름
            'turtlebot_status', #토픽 이름
            self.listener_callback,
            10)
        self.subscription  # prevent unused va
riable warning
    def listener_callback(self, msg):
        self.get_logger().info('Battert_Percen
tage: "%s"' % msg.battery_percentage)
        self.get_logger().info('Power_Supply_S
tatus: "%s"' % msg.power_supply_status)
        self.get_logger().info('Angular: "%
s"' % msg.twist.angular.z)
        self.get_logger().info('Linear: "%
s"' % msg.twist.linear.x)


def main(args=None):
    rclpy.init(args=args)
    minimal_subscriber = MinimalSubscriber()
    rclpy.spin(minimal_subscriber)
    # Destroy the node explicitly
    # (optional - otherwise it will be done au
tomatically
    # when the garbage collector destroys the
node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()


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

publisher

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist

class MinimalPublisher(Node):
    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callba
ck)
    def timer_callback(self):
        msg = Twist()
        msg.angular.z = 1.0
        msg.linear.x = 1.0
        self.publisher_.publish(msg)
        self.get_logger().info('Angular: "%f"' % msg.angular.z)
        self.get_logger().info('Linear: "%f"' % msg.linear.x)

def main(args=None):
    rclpy.init(args=args)
    minimal_publisher = MinimalPublisher()
    rclpy.spin(minimal_publisher)
    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()

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

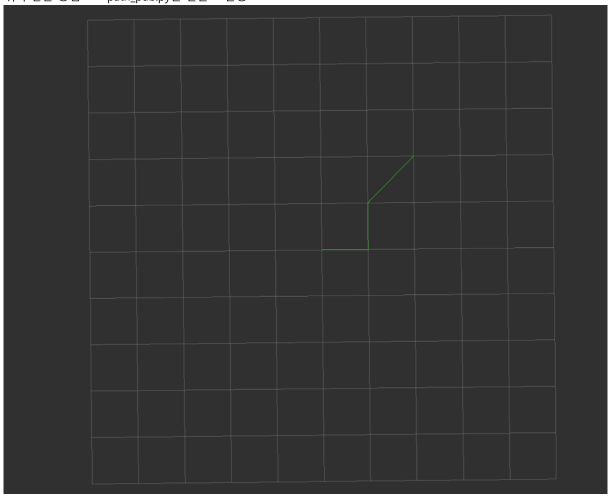
```
import rclpy
from rclpy.node import Node
from squaternion import Quaternion
from nav_msgs.msg import Odometry


class odom(Node):
    def __init__(self):
        super().__init__('odom')
        self.odom_publisher = self.create_publisher(Odometry, 'odom', 10)
        timer_period = 0.1  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.odom_msg=Odometry()
```

ssafybridge 런치를 실행한 상태에서 odom_pub 를 실행
rviz에 odom 토픽 추가



```python
class odom(Node):
    def __init__(self):
        super().__init__('odom')
        self.odom_publisher = self.create_publisher(Odometry, 'odom', 10)
        timer_period = 0.1  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.odom_msg=Odometry()
        self.odom_msg.header.frame_id='map'
        self.time=0.0
    def timer_callback(self):
        x=self.time
        q=Quaternion.from_euler(0,0,0)
        self.odom_msg.pose.pose.position.x=x
        self.odom_msg.pose.pose.position.y=0.0
        self.odom_msg.pose.pose.orientation.x=q.x
        self.odom_msg.pose.pose.orientation.y=q.y
        self.odom_msg.pose.pose.orientation.z=q.z
        self.odom_msg.pose.pose.orientation.w=q.w
        self.odom_publisher.publish(self.odom_msg)
        self.time+=0.1


def main(args=None):
    rclpy.init(args=args)
    odom_node = odom()
    rclpy.spin(odom_node)
    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    odom_node.destroy_node()
    rclpy.shutdown()


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

위와 같은 방법으로 path_pub.py를 만들고 실행

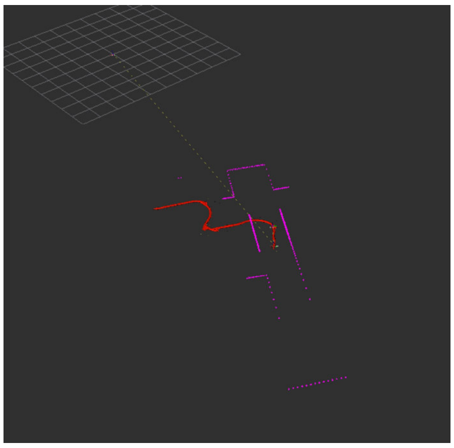


```python
import rclpy
from rclpy.node import Node
from squaternion import Quaternion
from nav_msgs.msg import Path
from geometry_msgs.msg import PoseStamped
class path(Node):
    def __init__(self):
        super().__init__('odom')
        self.global_path_pub = self.create_publisher(Path, 'global_path', 10)
        timer_period = 0.1  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.global_path_msg=Path()
        self.global_path_msg.header.frame_id='map'
        waypoints=[
            [0.0,0.0],
            [1.0,0.0],
            [1.0,1.0],
            [2.0,2.0]
        ]
        for waypoint in waypoints:
            point=PoseStamped()
            point.pose.position.x=waypoint[0]
            point.pose.position.y=waypoint[1]
            point.pose.orientation.w=1.0
            self.global_path_msg.poses.append(point)
    def timer_callback(self):
        self.global_path_pub.publish(self.global_path_msg)


def main(args=None):
    rclpy.init(args=args)
    test_path = path()
    rclpy.spin(test_path)
    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    test_path.destroy_node()
    rclpy.shutdown()


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

tf2 파이썬 사용법
https://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20static%20broadcaster%20%28Python%29

스켈레톤을 쓰면 ssafy_bridge에 있는 많은 노드들의 코드 내용이 바뀌기 때문에 문제풀이 영상대로 해도 제대로 되지 않는다.
부득이하게 예전 코드로 진행하도록 하자

Rviz에 TF, Odometry, LaserScan을 모두 Add 해준다
Odometry의 화살표 크기는 줄여주는게 좋다.
그리고 오른쪽에서 Target_Frame을 base_link로 해주도록 하자
당연히 시뮬레이터도 예전 시뮬레이터를 사용한다.



```python
import rclpy
from rclpy.node import Node
from squaternion import Quaternion
from nav_msgs.msg import Odometry
from ssafy_msgs.msg import TurtlebotStatus
import tf2_ros
import geometry_msgs.msg
from math import pi, cos, sin
class odom(Node):
    def __init__(self):
        super().__init__('odom')
        self.subscription = self.create_subscription(
            TurtlebotStatus, #타입 이름
            'turtlebot_status', #토픽 이름
            self.listener_callback, #메시지들어올때 호출될 함수
            10)
        self.odom_publisher = self.create_publisher(Odometry, 'odom', 10)

        self.broadcaster = tf2_ros.StaticTransformBroadcaster(self)
        self.odom_msg = Odometry()
        self.base_link_transform = geometry_msgs.msg.TransformStamped() #broadcast할 좌표계 메시지 생성
        self.laser_transform = geometry_msgs.msg.TransformStamped() #broadcast할 좌표계 메시지 생성
        self.is_status = False
        self.is_calc_theta = False
        self.x = 0
        self.y = 0
        self.theta = 0
        self.prev_time = 0
        self.odom_msg.header.frame_id = 'map' #odometry 메시지에 좌표계 이름을 정해줌
        self.odom_msg.child_frame_id = 'base_link'
        self.base_link_transform = geometry_msgs.msg.TransformStamped()
        self.base_link_transform.header.frame_id = 'map'
        self.base_link_transform.child_frame_id = 'base_link'
        self.laser_transform = geometry_msgs.msg.TransformStamped()
        self.laser_transform.header.frame_id = 'base_link'
        self.laser_transform.child_frame_id = 'laser'
        self.laser_transform.transform.translation.x = 0.0 #바뀌지 않는 값임
        self.laser_transform.transform.translation.y = 0.0
        self.laser_transform.transform.translation.z = 1.0
        self.laser_transform.transform.rotation.w = 1.0
        #base_link 로부터 laser 좌표계는 회전되어 있지 않다 그렇기때문에 init에 쓰는 것

    def listener_callback(self, msg):
        if self.is_status == False:
            self.is_status = True
            self.prev_time = rclpy.clock.Clock().now() #적분 구간을 정해주기 위해 시간을 저장한다.
        else:
            self.current_time = rclpy.clock.Clock().now()
            self.period = (self.current_time - self.prev_time).nanoseconds/1000000000
            linear_x = msg.twist.linear.x
            angular_z = -msg.twist.angular.z
            self.x += linear_x*cos(self.theta)*self.period #적분을 코드로 표현 => 누적으로 표현한다.
            self.y += linear_x*sin(self.theta)*self.period
            self.theta += angular_z*self.period #각속도 누적 => 향하고 있는 방향 표현
            q = Quaternion.from_euler(0,0,self.theta) #오일러에서 쿼터니언으로 변환
            self.base_link_transform.header.stamp = rclpy.clock.Clock().now().to_msg() #broadcast용 시간
            self.laser_transform.header.stamp = rclpy.clock.Clock().now().to_msg() #broadcast용 시간
            self.base_link_transform.transform.translation.x = self.x #계산한 x, y가 translation(이동)값이 됨
            self.base_link_transform.transform.translation.y = self.y
            self.laser_transform.transform.rotation.x = q.x #계산한 q가 roatation(회전)값이 됨
            self.laser_transform.transform.rotation.y = q.y
            self.laser_transform.transform.rotation.z = q.z
            self.laser_transform.transform.rotation.w = q.w
            self.odom_msg.pose.pose.position.x = self.x #odometry 메시지도 이동, 회전, 제어 값을 채움
            self.odom_msg.pose.pose.position.y = self.y
            self.odom_msg.pose.pose.orientation.x = q.x
            self.odom_msg.pose.pose.orientation.y = q.y
            self.odom_msg.pose.pose.orientation.z = q.z
            self.odom_msg.pose.pose.orientation.w = q.w
            self.odom_msg.twist.twist.linear.x = linear_x
            self.odom_msg.twist.twist.angular.x = angular_z
            self.broadcaster.sendTransform(self.base_link_transform) #좌표계 broadcast
            self.broadcaster.sendTransform(self.laser_transform)
            self.odom_publisher.publish(self.odom_msg) #odometry 메시지 publish
            self.prev_time=self.current_time




def main(args=None):
    rclpy.init(args=args)
    odom_node = odom()
    rclpy.spin(odom_node)
    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    odom_node.destroy_node()
    rclpy.shutdown()


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