HW3 보고서

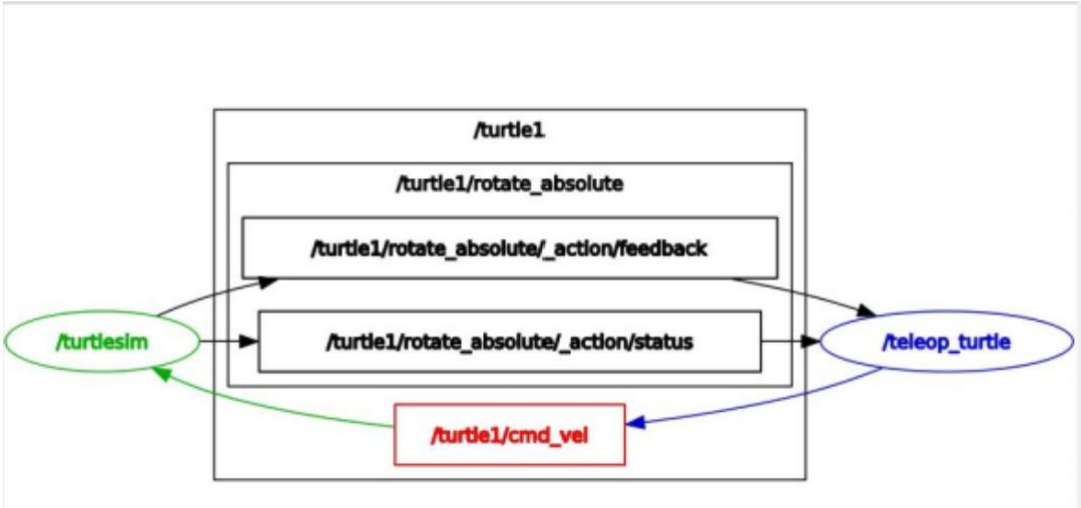
로빛 20기 인턴 모시온

2025407006 로봇학부

목차

1. 개요

본 보고서는 이전 과제 2에서 학습하였던 각각의 발행, 구독 노드를 활용해 토픽 통신을 바탕으로 지정 토픽을 발행하는 원리를 그대로 터틀 시뮬레이터에 접목하여 거북이를 사용자의 의도에 맞게 움직이도록 하는 실습입니다.



구체적으로 키보드의 조작에 따른 터틀심 조작 예제 실행시의 rqt_graph를 참고하였을 때, 터틀심 구독 노드로 전달되는 토픽(/turtlr1/cmd_vel)을 직접 코드로 토픽 명을 지정하여 원하는 동작을 이루도록 구현하는 것입니다.
(+@ 펜의 색깔 및 두께 조정을 위한 서비스 SetPen을 직접 전달하여 조절)

2. 코드

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from turtlesim.srv import SetPen
import time
```

터틀심의 조작을 위한 다음의 모듈 호출입니다.
인터넷의 예제를 참고하였을 때, 터틀봇의 움직임을 제어하는 코드는 다음과 같습니다.

ros2 topic pub --rate 0.1 turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.57}}"

이때, 발행되는 토픽(cmd_vel)과 메시지 종류(geometry_msgs/msg/Twist)를 동작 시킬 수 있도록 다음의 모듈을 호출하였으며, 일정 시간 동안 발행하였던 토픽을 유지하도록 추가적으로 time모듈을 호출하였습니다.

```
class KeyDraw(Node):
    def __init__(self):
        super().__init__('key_draw')
        self.publisher = self.create_publisher(Twist, '/turtle1/cmd_vel', 10)
        self.set_pen_client = self.create_client(SetPen, '/turtle1/set_pen')
```

Class KeyDraw입니다
토픽 및 서비스 발행을 위해 다음과 같이 생성하도록 하였습니다.
코드 구성은 각각 토픽 사용 예제 및 서비스 콜 예제를 참고하여 그대로 전달되도록 하였습니다.

```
def publish_twist(self, linear_x=0.0, angular_z=0.0, repeat=0):
    twist = Twist()
    twist.linear.x = linear_x
    twist.angular.z = angular_z

    for i in range(repeat):
        self.publisher.publish(twist)
        time.sleep(1)

    twist.linear.x = 0.0
    twist.angular.z = 0.0
    self.publisher.publish(twist)
```

twist 동작에 따른 각각의 args를 입력할 수 있는 함수입니다.
args(arguments)의 내부 요소를 직접 정의하여 이를 발행함으로서 터틀봇이 행동으로 옮기도록 구성한 알고리즘입니다.

twist를 Twist()를 통해 메시지 종류를 정하였으며, 각각의 요소를 twist.linear.x와 .angular.z로 사용할 수 있도록 하였습니다.

이를 통해 직접적으로 활용하는 방법은 사용자가 원하는 터틀봇의 반복 동작을 매개변수 지정을 통해 전달합니다. 그때, 매개변수를 통해 전달 받은 값을 반복 횟수만큼 발행하여 해당 행동을 1초 간격으로 유지하도록 하였으며 이후 모든 요소 값을 0으로 초기화하여 정지하도록 구성하였습니다.

```
def pen(self, r, g, b, width, off=0):
    color = SetPen.Request()
    color.r = r
    color.g = g
    color.b = b
    color.width = width
    color.off = off

    future = self.set_pen_client.call_async(color)
    rclpy.spin_until_future_complete(self, future)
```

펜 색상 및 두께에 대한 서비스 콜을 담당하는 함수 입니다.
SetPen.Request()를 통해 서비스의 요청을 color 메시지로 저장하며,
.r, .g, .b를 통해 색상 정의와 .width의 두께 .off를 통해 펜의 작성 여부를 저장하였으며, 이후 정의되는 future를 통해 사용자의 지정 색깔을 요청을 보내며 응답이 이루어질 때까지 기다리도록 합니다.

이를 통해서 펜 설정이 문제없이 적용되도록 하였습니다.

```
def keyboard_loop(self):
    print("W: Triangle, S: Square, A: Circle1, D: Circle2")
    while True:
        key = keyboard_input()
        if key == 'w':
            self.draw_triangle()
        elif key == 's':
            self.draw_square()
        elif key == 'a':
            self.draw_circle1()
        elif key == 'd':
            self.draw_circle2()
```

키보드의 입력을 통한 도형 작성이므로, 각 키 입력에 따라 순서대로, 삼각형 작성함수, 사각형 작성 함수, 원1 작성 함수, 원2 작성 함수가 동작되도록 하였으며, 이어서

```
def draw_triangle(self):
    self.pen(255, 0, 0, 3)
    for i in range(3):
        self.publish_twist(linear_x=2.0, repeat=1)
        self.publish_twist(angular_z=2.0944, repeat=1)

def draw_square(self):
    self.pen(0, 255, 0, 5)
    for i in range(4):
        self.publish_twist(linear_x=2.0, repeat=1)
        self.publish_twist(angular_z=1.5708, repeat=1)

def draw_circle1(self):
    self.pen(0, 0, 255, 7)
    self.publish_twist(linear_x=2.0, angular_z=2.0, repeat=4)

def draw_circle2(self):
    self.pen(255, 255, 255, 9)
    self.publish_twist(linear_x=1.0, angular_z=3.0, repeat=4)
```

위와 같이 구성된 각 함수의 정의를 통해서 의도한 도형을 작성하도록 하였습니다.

```
import sys
import termios
import tty
```

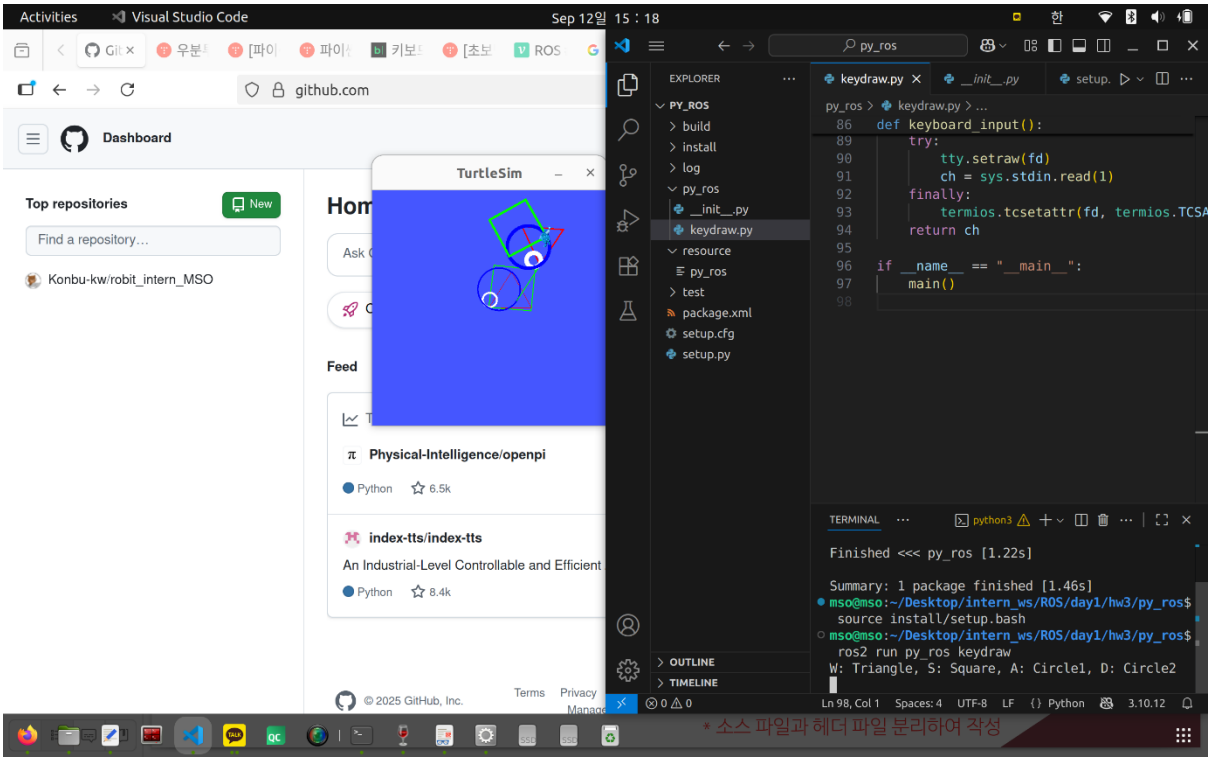
```
def keyboard_input():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(fd)
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch
```

keyboard 모듈을 호출하거나 get_key모듈을 호출함으로서 발생하는 권한(root)문제와 이를 일시적으로 해결하기 위해서 sudo를 활용하여 시도한 결과 Path에 지정되어 있지않아 실행이 안되는 등, 이후 평가에 대해 해당 코드가 로컬 컴퓨터 뿐만이 아닌 다른 컴퓨터에서도 동작될 수 있도록 하기 위해 다음의 모듈 및 함수를 활용하였습니다.

https://blog.naver.com/chandong83/222567117554
위의 링크의 예제를 참고하여 새로 제작하였으며 코드 설명은 다음과 같습니다.

이를 위해 터미널 입력 모드를 일시적으로 /teleop_key 터틀봇 동작 터미널 코드와 같이(해당 터미널 코드도 참고하였습니다.)
raw 모드로 전환(tty.setraw)하여 사용자의 별도의 엔터 입력 없이도 문자 단위 입력을 즉시 읽어올 수 있도록 처리하였으며, 입력된 값은 sys를 호출함으로서 사용 가능한 sys.stdin.read(1)을 통해 한 글자씩 가져오도록 하였습니다.
또한 프로그램 실행 전의 터미널 설정값을 termios.tcgetattr로 저장한 후, 실행이 끝난 뒤에는 반드시 termios.tcsetattr로 초기화함으로써 터미널 환경에 쓰레기 값이 남지 않도록 하며 안정적으로 동작할 수 있게 하였습니다.

3. 결과



4. 참조

https://stackoverflow.com/questions/78328203/how-to-get-python-out-of-the-raw-terminal-mode
https://blog.naver.com/chandong83/222567117554