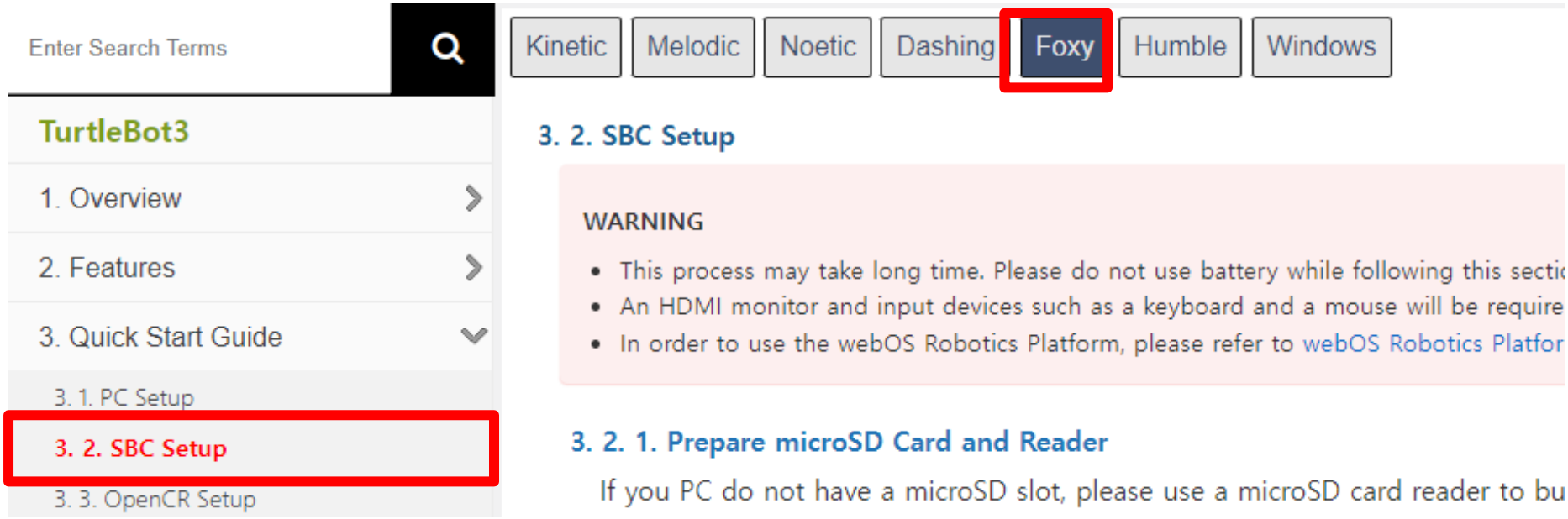
개발 환경 구성
(Turtlebot3 설치)

ROS 2 foxy

SBC 설치

설치 참고 사이트

- ROS 2 foxy 설치
  - https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#sbc-setup
  - 링크 접속 후 버전인 Foxy 를 반드시 선택



TurtleBot3 부팅디스크 생성

- microSD Card에 TurtleBot3 SBC Image를 저장함
  (guide 3.2.1~3.2.5 참조)
  - 준비: microSD, TurtleBot3 SBC Image, image굽는 프로그램
  - PC에 microSD를 연결
  - 이미지 굽는 프로그램을 실행함
  - microSD에 Image를 저장함
  - GParted GUI tool을 활용해서 Partition을 확장함

TurtleBot3 WiFi Network 설정

- TurtleBot3의 WiFi Network를 설정
  - PC에 microSD를 연결
  - 다음 코드를 실행함
    - microSD 이름이 '**writable**'

```
$ cd /media/$USER/writable/etc/netplan

$ sudo nano 50-cloud-init.yaml
```

  - nano 편집기 사용
    - '**WiFi_SSID**'와 '**WIFI_PASSWORD**' 수정
    - 저장(Ctrl+S)하고 종료(Ctrl+X)함
    - 들여쓰기는 공백문자 두칸씩

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: yes
      dhcp6: yes
      optional: true
  wifis:
    wlan0:
      dhcp4: yes
      dhcp6: yes
      access-points:
        WIFI_SSID:
          password: WIFI_PASSWORD
```

TurtleBot3 부팅

- TurtleBot3 Raspberry Pi 부팅
  - 라즈베리파이 microHDMI 포트에 '**모니터**' 연결
  - 라즈베리파이 USB 포트에 '**키보드**' 연결
  - 라즈베리파이에 '**microSD 카드**'를 삽입
  - TurtleBot3 '**파워**' 연결
  - 부팅 종료 확인 - 다음 화면 나온 후 멈추면 부팅 종료임

```
Ubuntu 20.04.3 LTS ubuntu tty1
ubuntu login: [    50.039124]  cloud-init~~~
[    51.899742] cloud-init ~~
[    51.900414] cloud-init ~~
```

TurtleBot3 로그인

- 로그인하기
  - 부팅 종료 후 화면 멈추면 '**엔터키**' 클릭, 로그인 나옴

```
ubuntu login:
```

  - id는 '**ubuntu**', password는 '**turtlebot**' 입력

```
ubuntu login: ubuntu
Password: turtlebot
```

- IP 확인
  - wlan0의 inet 이후 적힌 숫자, **IP Address of Rasberry PI**

```
System information ~
IPv4 address for wlan0:  192.168.1.30
```

Remote PC에서 TurtleBot3 접속

- Remote PC의 터미널(Ctrl+Alt+T) 열기
- 동일 wifi로 연결
- 다음 명령어를 입력한 후 접속
  - ssh ubuntu@{**IP Address of Rasberry PI**}

```
$ ssh ubuntu@192.168.1.30
```

  - 질문이 나오면 '**yes**'

```
Are you sure want to continue connecting (yes/no/[])? yes
ubuntu@192.168.1.30's password: turtlebot
```

Remote PC에서 TurtleBot3 로그인

- 로그인
  - 질문이 나오면 '**yes**' (질문이 없을 수도 있음)

```
Are you sure want to continue connecting (yes/no/[])? yes
```

  - password는 '**turtlebot**' 입력 (아이디는 접속시 @ 앞에 입력함)

```
ubuntu@192.168.1.30's password: turtlebot
```

  - 로그인 성공시 아래 화면 보임

```
ubuntu@ubuntu:~$
```

배치파일 수정 및 적용

- 배치파일 확인

```
$ nano ~/.bashrc
```

- 배치파일 수정
  - 열린 배치파일에서 다음 내용을 추가 (작성시 '=' 전후 띄어쓰기X)
  - 저장(Ctrl+S)하고 종료(Ctrl+X)함

```
export ROS_DOMAIN_ID=30
export TURTLEBOT3_MODEL=waffle_pi
export LDS_MODEL=LDS-02
```

- 배치파일 적용

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

OpenCR 설치

- Remote PC에서 TurtleBot3 접속
- OpenCR 설치 (guide 3.3 참조) - 주의 OPENCR_MODEL은 waffle임

```
$ sudo dpkg --add-architecture armhf
$ sudo apt update
$ sudo apt install libc6:armhf

$ export OPENCR_PORT=/dev/ttyACM0
$ export OPENCR_MODEL=waffle
$ rm -rf ./opencr_update.tar.bz2

$ wget https://github.com/ROBOTIS-GIT/OpenCR-
Binaries/raw/master/turtlebot3/ROS2/latest/opencr_update.tar.bz2
$ tar -xvf ./opencr_update.tar.bz2

$ cd ~/opencr_update
$ ./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr
```

Bringup

- Remote PC의 터미널(Ctrl+Alt+T) 열기

```
$ ssh ubuntu@192.168.1.30
```

- Bring up 실행
  - TurtleBot3의 모든 장치들을 구동함
  - 터미널에 'Run!'이 출력되면 Bringup 성공

```
$ ros2 launch turtlebot3_bringup robot.launch.py
```

설치 확인

실습 과제

TurtleBot3 Bringup

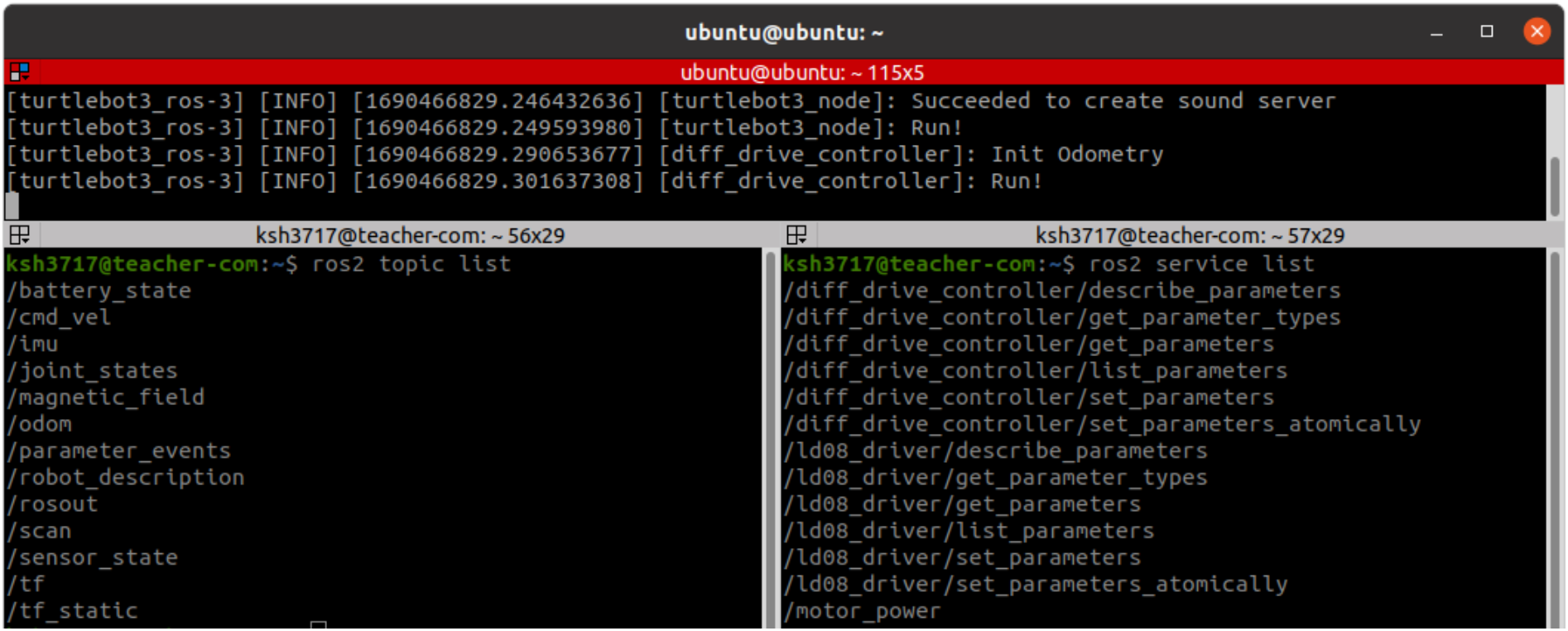
TurtleBot3를 Bringup한 후 다음 코드를 실행한 후 캡쳐함

과제1

```
$ ssh ubuntu@{IP_ADDRESS_OF_RASPBERRY_PI}
$ ros2 launch turtlebot3_bringup robot.launch.py
```

```
$ ros2 topic list
```

```
$ ros2 service list
```



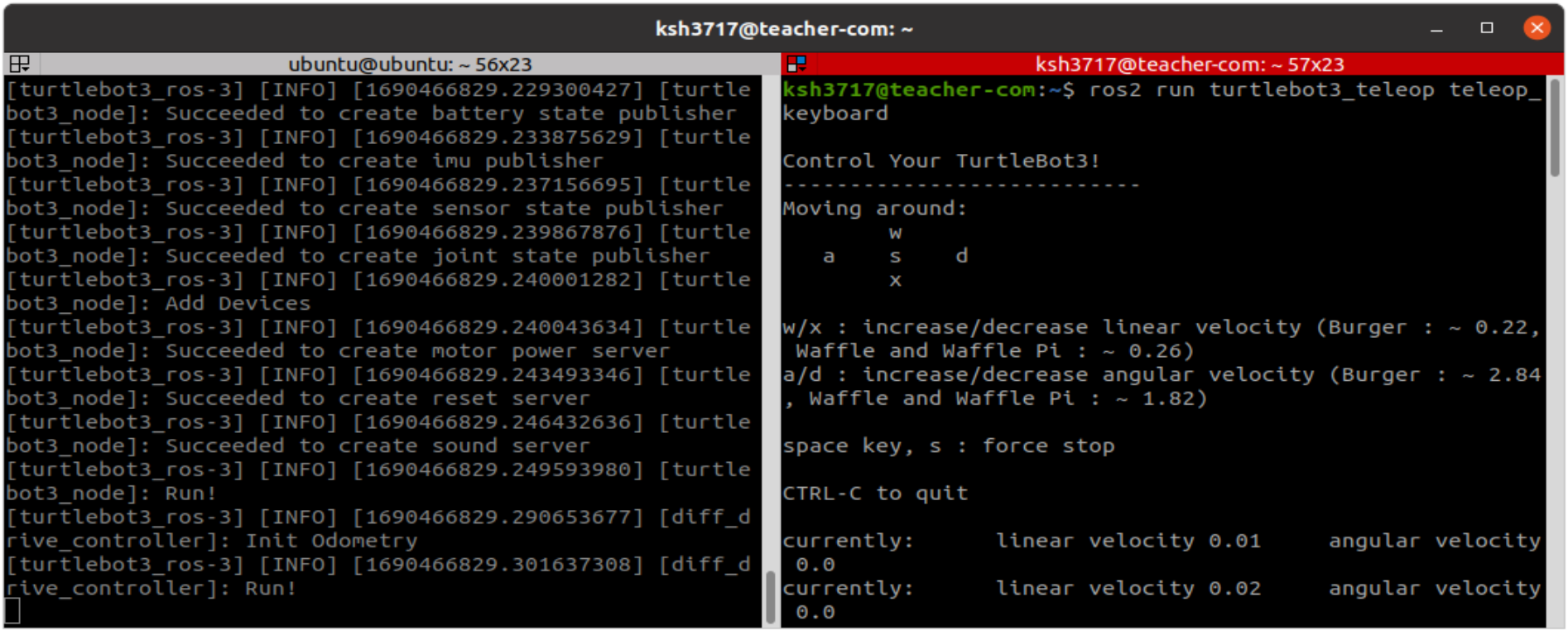
TurtleBot3 움직임

과제2

터미널을 2개 열고 아래 코드를 각각 실행시킨 다음
TurtleBot3를 Remote의 키보드로 움직이고, 결과를 캡쳐함

```
$ ssh ubuntu@{IP_ADDRESS_OF_RASPBERRY_PI}
$ ros2 launch turtlebot3_bringup robot.launch.py
```

```
$ ros2 run turtlebot3_teleop teleop_keyboard
```

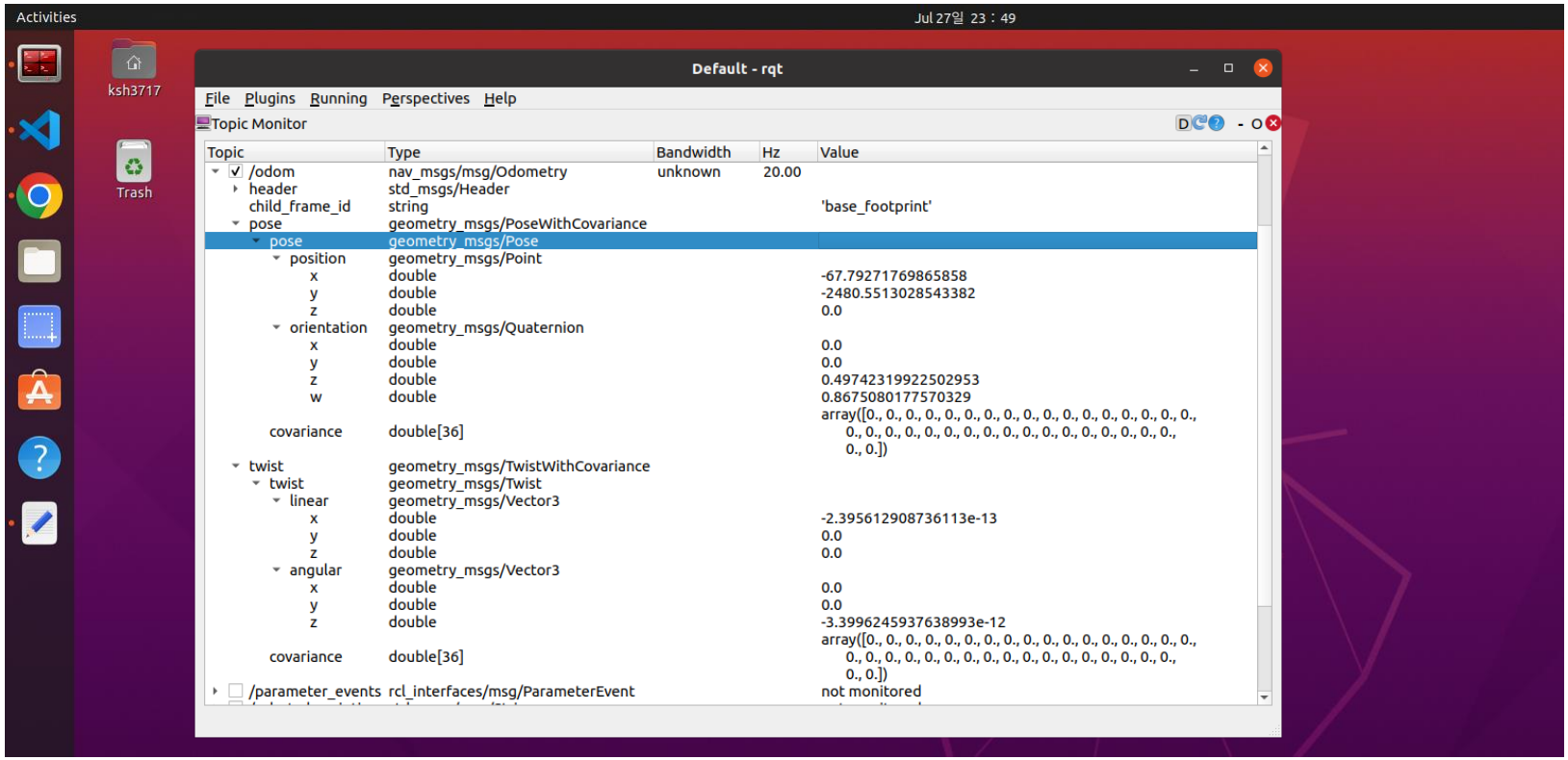


Topic Monitor

TurtleBot3를 움직이며 다음 ROS 도구를 실행한 후 변화를 캡쳐함 **과제3**

```
$ rqt
```

[Plugins]-[Topics]-[Topic Monitor] 메뉴 클릭
'/odom' 왼쪽 체크 후 header, pose, twist의 모든 값을 연 후 값의 변화를 확인함

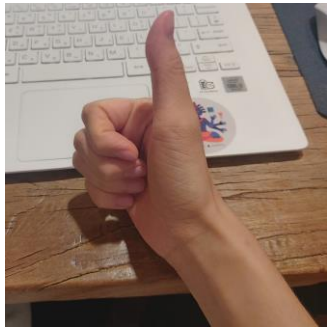


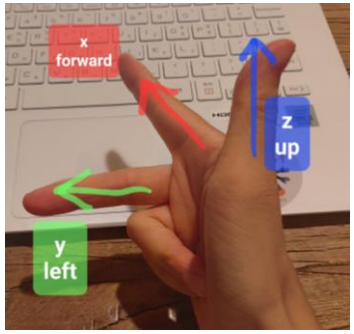
메시지 구조 확인하기

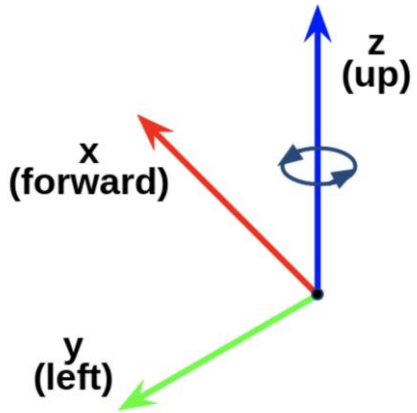
- 실행 중인 노드 확인
  - $ ros2 node list
- 노드가 퍼블리시하거나 구독하는 토픽 확인
  - $ ros2 node info /turtlebot3_node
- 현재 사용 중인 토픽 목록 확인
  - $ ros2 topic list
- 특정 토픽의 메시지 타입 확인
  - $ ros2 topic info /cmd_vel
- 메시지의 구조 확인
  - $ ros2 interface show geometry_msgs/msg/Twist

ROS 2 오른손 좌표계

- ROS 2에서 TurtleBot3와 같은 로봇 시스템은 기본적으로 오른손 좌표계를 따릅니다.
- 오른손 좌표계 정의
  - **X축**: 로봇의 전방 방향 (로봇이 직진하는 방향)
  - **Y축**: 로봇의 좌측 방향 (로봇에서 왼쪽으로 나가는 방향)
  - **Z축**: 로봇의 상방 (로봇이 서 있는 방향에서 위쪽)
- TurtleBot3에서 Twist 메시지를 사용해 속도를 명령
  - X축의 선속도(linear.x)는 앞으로/뒤로 움직임을 제어
  - Z축의 각속도(angular.z)는 좌우 회전을 제어



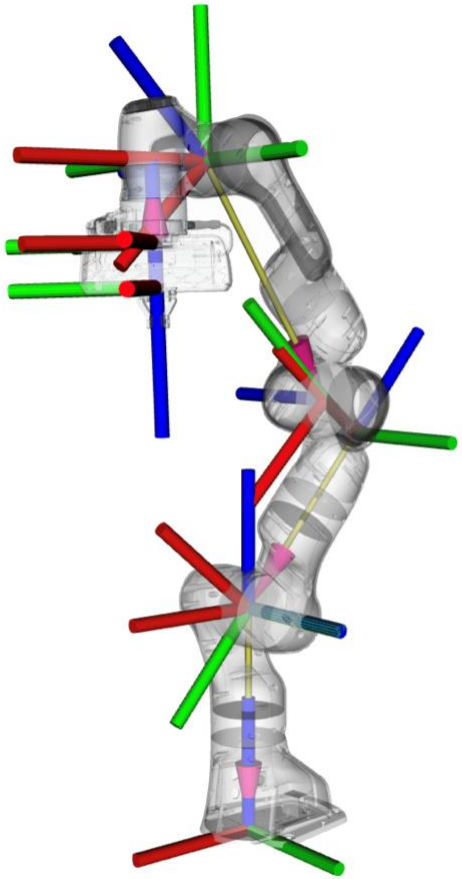


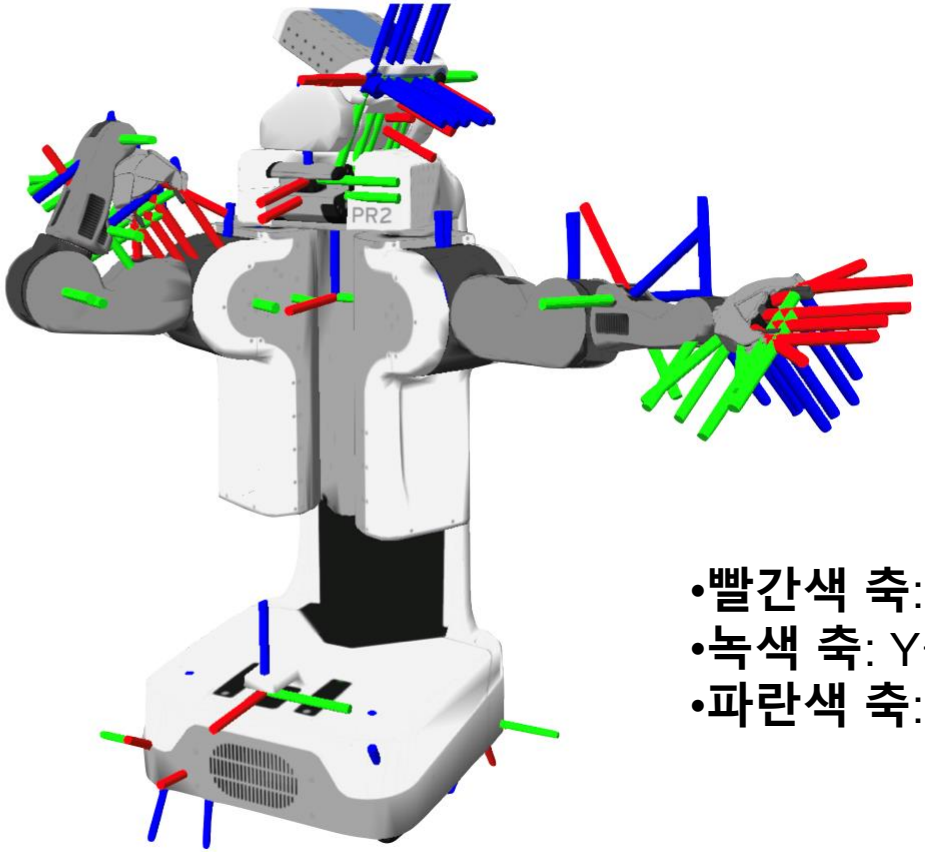


https://velog.io/@hwang-chaewon/ROS-2018

TF(Transform Frames)

TF는 로봇의 각 부분마다 좌표계 표시, TF는 이 좌표계들 간의 상대적인 위치와 방향을 추적하여, 로봇이 움질일 때 각 파트가 어떻게 변화하는지 계산함





- **빨간색 축**: X축
- **녹색 축**: Y축
- **파란색 축**: Z축

https://ros-learnings.hashnode.dev/understanding-tf-viewframes-in-ros-an-essential-component-for-rviz

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

class TurtleBot3Teleop(Node):

    def __init__(self):
        super().__init__('turtlebot3_teleop')
        self.pub = self.create_publisher(Twist, '/cmd_vel', 10)
        self.cmd = Twist()

    def move_forward(self):
        self.cmd.linear.x = 0.5
        self.cmd.angular.z = 0.0
        self.pub.publish(self.cmd)

    def rotate(self):
        self.cmd.linear.x = 0.0
        self.cmd.angular.z = 0.5
        self.pub.publish(self.cmd)

def main(args=None):
    rclpy.init(args=args)
    teleop = TurtleBot3Teleop()
    teleop.move_forward()  # 로봇이 앞으로 이동
    teleop.rotate()  # 로봇이 회전
    teleop.destroy_node()
    rclpy.shutdown()

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

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
import sys
import termios
import tty

class TurtleBot3Teleop(Node):

    def __init__(self):
        super().__init__('turtlebot3_teleop')
        self.pub = self.create_publisher(Twist, '/cmd_vel', 10)
        self.cmd = Twist()

    def get_key(self):
        # 키 입력을 받아오는 함수
        tty.setraw(sys.stdin.fileno())
        key = sys.stdin.read(1)
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, termios.tcgetattr(sys.stdin))
        return key

    def control_loop(self):
        print("Use 'w', 'a', 's', 'd' to move. Press 'q' to quit.")
        # 작성하시오

def main(args=None):
    rclpy.init(args=args)
    teleop = TurtleBot3Teleop()
    teleop.control_loop()
    teleop.destroy_node()
    rclpy.shutdown()


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

```
def control_loop(self):
        print("Use 'w', 'a', 's', 'd' to move. Press 'q' to quit.")
        try:
            while True:
                key = self.get_key()
                if key == 'w':
                    self.cmd.linear.x = 0.5
                    self.cmd.angular.z = 0.0
                elif key == 's':
                    self.cmd.linear.x = -0.5
                    self.cmd.angular.z = 0.0
                elif key == 'a':
                    self.cmd.linear.x = 0.0
                    self.cmd.angular.z = 0.5
                elif key == 'd':
                    self.cmd.linear.x = 0.0
                    self.cmd.angular.z = -0.5
                elif key == 'q':
                    break
                else:
                    self.cmd.linear.x = 0.0
                    self.cmd.angular.z = 0.0

                self.pub.publish(self.cmd)

        except Exception as e:
            print(f"Error: {e}")

        finally:
            # 종료 시 로봇 정지
            self.cmd.linear.x = 0.0
            self.cmd.angular.z = 0.0
            self.pub.publish(self.cmd)
            print("Shutting down.")
```