

1. Yahboom_car spec 정리

1. Camera

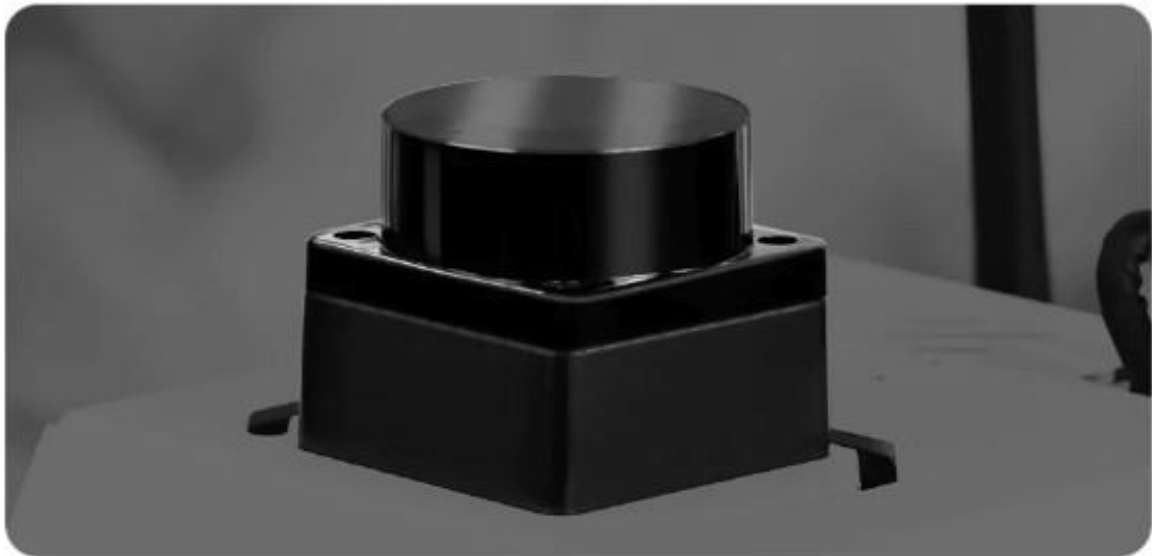


2DOF camera PTZ parameters			
PTZ freedom	2	Size	53*67*78mm
Weight	60g	Servo model	YB-SG90 9G digital servo
Support camera	High frame rate wide angle camera		
Camera Parameters			
Image pixels	2MP(1080)	Data Format	320*240/640*480/1280*720/1920*1080
Lens	F2.8mm	Field of view	80°~120° (Depends on video resolution)
Sleep current	<10mA	Working current	<200mA
Interface mode	USB2.0	Operating voltage	DC5V
Servo Parameters			
Operating voltage	4.8V-6.0VDC	No-load current	80mA(4.8V)
No-load speed	0.14sec/60°(4.8V)	Stall torque	1.3kgf.cm(4.8V)
Angle accuracy	180°±1°	Pulse width range	500~2500us corresponds to 0~180°
Gear material	Plastic teeth	Output shaft material	Plastic shaft
Working time	1,000,000Cycles(Min)	Weight	10±0.5g

2. LiDAR

TOF high-performance lidar (ORBEC MS200)

MS200 adopts TOF ranging method, withstands 30Klux of strong light, supports indoor and outdoor mapping navigation, measurement radius up to 12m, a measurement blind zone only 3cm, ranging error ± 2 mm within 2 meters, sampling frequency 4500 times/s, and scanning frequency 7HZ-15HZ, support 230400bps communication rate.



Ranging principle	TOF ranging	Sampling frequency	4500 times/s
Scan angle	360°	Scanning frequency	7Hz~15Hz
Measurement angle accuracy	0.8°	Dimensions	37.7*37.5*33mm
Resistance to ambient light intensity	30KLux	Certified file	ROHS2.0,REACH,CE,FCC
Weight	40g	Ranging accuracy	≤4mm (0.2m~2m), ≤15mm (2m~12m)
Waterproof and dustproof	IP5X	Minimum measuring distance	0.03m
Measure radius	Black object:12m	Communication Interface	Standard asynchronous serial port(UART)
Communication rate	230400	Drive mode	Built-in brushless motor
Power supply	DC5.0 \pm 0.5V	ROS support	ROS1/ROS2
Windows support	Provide PC software on Windows		

링크 : <https://category.yahboom.net/collections/ros-roboticcar/products/microros-pi5>

2. IMU 실습

도훈님이 올려주신 자료 기반으로 공부했습니다.... 정말 감사합니다

1. IMU 토픽 읽어보기

ros2 topic echo /imu

```
- 0.0
- 0.0
- 0.0
- 0.0
angular_velocity:
  x: 0.0
  y: 0.0
  z: 0.0
angular_velocity_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
linear_acceleration:
  x: 0.0
  y: 0.0
  z: 9.802692413330078
linear_acceleration_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
```

가속도계로부터 가속도(m/s^2)를, 각속도계로부터 각속도(rad/s) 값을 지속적으로 받아 오는 것을 확인할 수 있다.

개념 설명 - 토픽

- 발행과 구독의 방식으로 이루어지는 **노드 간 비동기식 단방향 메세지 송수신 방식**이다
- 지속적으로 센서 데이터를 생성하고 이를 받아 처리할 때, 즉 **항시 정보를 주고 받아야 하는 부분**에 사용된다.

한편 아래 명령어로 /imu의 메세지 타입 (sensor_msgs/msg/Imu)가 어떤 형태로 구성되어 있는지도 확인할 수 있다.

```
1. ros2 interface show sensor_msgs/msg/Imu
```

```
std_msgs/Header header
  builtin_interfaces/Time stamp
    int32 sec
    uint32 nanosec
  string frame_id

geometry_msgs/Quaternion orientation
  float64 x 0
  float64 y 0
  float64 z 0
  float64 w 1
float64[9] orientation_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 angular_velocity
  float64 x
  float64 y
  float64 z
float64[9] angular_velocity_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 linear_acceleration
  float64 x
  float64 y
  float64 z
float64[9] linear_acceleration_covariance # Row major x, y z
```

아래의 명령어로 발행하고 있는 토픽 정보를 기록할 수도 있을 것이다.

```
1. ros2 bag record /imu
```

현재 위치하는 디렉토리에서 rosbag 파일이 생성된 것을 확인할 수 있다.

```
jungseong@jungseong-ubuntu:~/yahboomcar_ws/bag$ ros2 bag record /imu
[INFO] [1743774989.363105859] [rosbag2_recorder]: Press SPACE for pausing/resuming
[INFO] [1743774989.364375036] [rosbag2_storage]: Opened database 'rosbag2_2025_04_04-22_56_29/rosbag2_2025_04_04-22_56_29_0.db3' for READ_WRITE.
[INFO] [1743774989.365849405] [rosbag2_recorder]: Listening for topics...
[INFO] [1743774989.366081521] [rosbag2_recorder]: Event publisher thread: Starting
[INFO] [1743774989.369047193] [rosbag2_recorder]: Subscribed to topic '/imu'
[INFO] [1743774989.369232837] [rosbag2_recorder]: Recording...
[INFO] [1743774989.369490124] [rosbag2_recorder]: All requested topics are subscribed. Stopping discovery...
[INFO] [1743774997.291778208] [rosbag2_cpp]: Writing remaining messages from cache to the bag. It may take a while
[INFO] [1743774997.292762401] [rosbag2_recorder]: Event publisher thread: Exiting
[INFO] [1743774997.293309889] [rosbag2_recorder]: Recording stopped
jungseong@jungseong-ubuntu:~/yahboomcar_ws/bag$ ls -la
.  ..  rosbag2_2025_04_04-22_56_29
```

rosbag 파일을 실행하고 싶으면 다음 명령어를 사용하면 된다.

1. <code>ros2 bag play <rosbag 파일 이름></code>
--

3. ROS 표준 관련 공부

1. ROS2 표준 단위에 대한 규칙

ROS 커뮤니티에서는 단위 불일치로 인한 문제를 줄이기 위해 표준 단위에 대한 규칙을 세웠다.

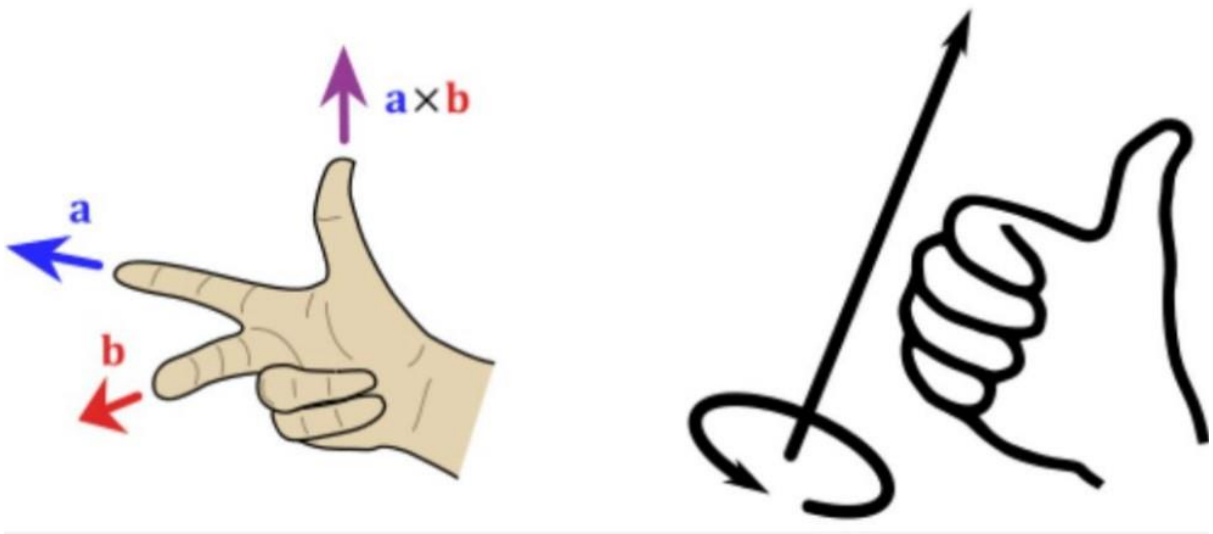
물리량	단위 (SI unit)	물리량	단위 (SI derived unit)
length (길이)	meter (m)	angle (평면각)	radian (rad)
mass (질량)	kilogram (kg)	frequency (주파수)	hertz (Hz)
time (시간)	second (s)	force (힘)	newton (N)
current (전류)	ampere (A)	power (일률)	watt (W)
		voltage (전압)	volt (V)
		temperature (온도)	celsius (°C)
		magnetism (자기장)	tesla (T)

2. ROS2 좌표 표현 통일에 관한 규칙

ROS 커뮤니티에서는 서로 다른 좌표 표현 방식을 사용할 경우 발생하는 좌표계 불일치를 사전에 막기 위해 좌표계 통일 규칙을 세웠다.

3. ROS2 좌표 표현의 기본 규칙

a. 회전 축의 경우 오른손 법칙으로 오른손으로 감는 방향이 회전 x 방향이다.



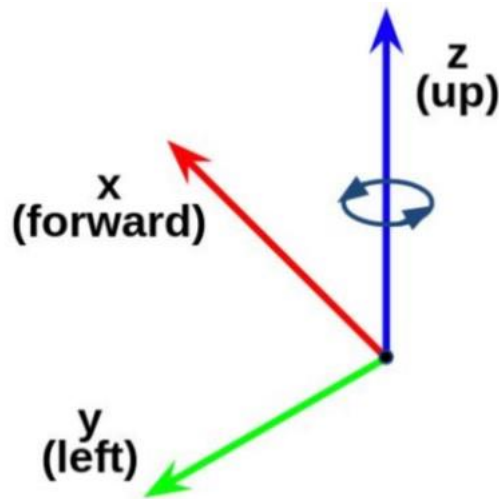
b. 회전 각의 경우 라디안을 사용한다.

2. ROS2 축 방향 기본 규칙

EX1) 카메라의 경우 컴퓨터 비전 분야에서는 z forward, x right, y down 을 기본 좌표계로써 사용하고 있었는데, 로봇은 기본적으로 x forward, y left, z up 을 기본 좌표계로써 사용한다.

EX2) LiDAR 및 IMU, Torque 센서도 제조사 별로 서로 다른 좌표계를 사용할 수 있다. 예를 들어 IMU 가 NED 타입 (x north, y east, z down ; relative to magnetic north)이나 ENU 타입 (x east, y north, z up ; relative to magnetic north)를 잘 살펴봐야한다.

1. ROS 커뮤니티에서는 축 방향으로 x forward, y left, z up 을 사용한다.
Gazebo & RViz 에서 Red 는 x 축, Green 은 y 축, Blue 는 z 축을 의미한다



2. ENU 좌표 지리적 위치(geographic locations)의 단거리 데카르트 표현의 경우 ENU(east north up) 규칙을 사용한다. 실내 로봇에서는 잘 다루지 않는 좌표이긴 하지만 비교적 큰 맵을 다루는 드론, 실외 자율주행 로봇에서 사용하는 좌표라고 생각하면 된다.
3. 접미사 프레임 사용 (Suffix Frames) 위에서 언급한 x forward, y left, z up 의 기본 3 축 및 ENU 좌표에서 벗어나는 경우 접미사 프레임을 사용하여 기본 좌표계와 구별하여 사용한다. 자주 사용되는 접미사로 프레임으로는 _optical 접미사와 _ned 접미사가 있으며 필요시 좌표 변환을 통해 사용하고 있다.
 - 3-1. _optical 접미사 컴퓨터 비전 분야의 경우, 카메라 좌표계로 많이 사용되는 z forward, x right, y down 를 사용하게 되는데 이럴 경우에는 카메라 센서의 메시지에 _optical 접미사를 붙여 구분한다. 이때에는 **z forward, x right, y down**의 카메라 좌표계와 **x forward, y left, z up**의 로봇 좌표계 간의 TF(transform)가 필요하다.
 - 3-2. _ned 접미사 실외에서 동작하는 시스템의 경우, 사용하는 센서 및 지도에 따라 **ENU 가 아닌 NED (north east down) 좌표계를 사용해야 할 때가 있다.** 이때에는 _ned 접미사를 붙여 구분한다.

3. ROS2 좌표 표현의 회전 표현 규칙

1. 쿼터니언 (quaternion)

- 간결한 표현방식으로 가장 널리 사용됨 (x, y, z, w)
- 특이점 없음 (No singularities)

2. 회전 매트릭스 (rotation matrix)

- 특이점 없음 (No singularities)

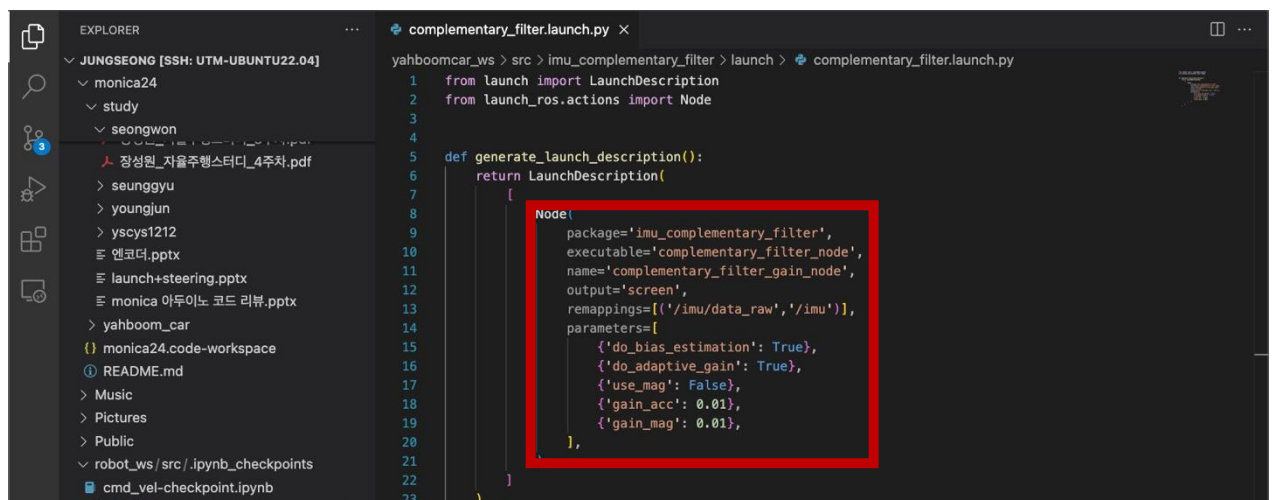
3. 고정축 roll, pitch, yaw (fixed axis roll, pitch, yaw about X, Y, Z axes respectively)

- 각속도에 사용

4. 오일러 각도 yaw, pitch, roll (euler angles yaw, pitch, and roll about Z, Y, X axes respectively)

- 전역 좌표계에서 회전이 발생하기 때문에 한 축의 회전이 다른 축의 회전과 겹치는 문제(일명 짐벌락) 로 인해 사용을 권장하지 않는다.

4. 런치 프로그래밍 공부



```
1 from launch import LaunchDescription
2 from launch_ros.actions import Node
3
4
5 def generate_launch_description():
6     return LaunchDescription(
7         [
8             Node(
9                 package='imu_complementary_filter',
10                 executable='complementary_filter_node',
11                 name='complementary_filter_gain_node',
12                 output='screen',
13                 remappings=[('/imu/data_raw', '/imu')],
14                 parameters=[
15                     {'do_bias_estimation': True},
16                     {'do_adaptive_gain': True},
17                     {'use_mag': False},
18                     {'gain_acc': 0.01},
19                     {'gain_mag': 0.01},
20                 ],
21             )
22         ]
23     )
```

imu_complememntary_filter/launch/complementary_filter.launch.py

[EX]

'Node' 클래스에서...

package : 실행할 패키지 이름

executable : 실행 가능한 노드의 이름

name : 지정한 노드를 실행할 때 실제로 사용할 이름

remapping : 특정 이름을 변경할 수 있는 것

EX) 위의 경우 /imu/data_raw 토픽이라는 이름을 /imu 로 변경

parameters : 특정 파라미터 값을 넣어준다

imu_complementary_filter/src/complementary_filter_node.cpp

```
1. #include "imu_complementary_filter/complementary_filter_ros.h"
2.
3. int main(int argc, char **argv)
4. {
5.     rclcpp::init(argc, argv);
6.     auto filter = std::make_shared<imu_tools::ComplementaryFilterROS>();
7.     rclcpp::spin(filter);
8.     rclcpp::shutdown();
9.     return 0;
10. }
```

imu_tools:ComplementaryFilterROS() 함수를 호출함.

ROS2 CPP 프로그래밍 공부

1. 패키지 생성

ros2 pkg create CLI 명령어로 패키지를 생성한다

```
1. ros2 pkg create [패키지이름] --build-type [빌드 타입] --dependencies [의존하는 패키지 1]
[의존하는 패키지 n]
```

명령어를 실행하면 아래와 같이 폴더와 파일이 생성된다

```
.
├── include
│   └── my_first_ros_rclcpp_pkg
├── src
├── CMakeLists.txt
└── package.xml

3 directories, 2 files
```

2. 패키지 설정

2-1. 패키지 설정 파일 (package.xml) : RCL 라이브러리가

C++이면 build_type 로 ament_cmake

Python 이면 build_type 로 ament_python 으로 설정

```
1. <?xml version="1.0"?>
2. <?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
3. <package format="3">
4.   <name>my_first_ros_rclcpp_pkg</name>
5.   <version>0.0.1</version>
6.   <description>ROS 2 rclcpp basic package for the ROS 2 seminar</description>
7.   <maintainer email="pyo@robotis.com">Pyo</maintainer>
8.   <license>Apache License 2.0</license>
9.   <author>Mikael Arguedas</author>
10.  <author>Morgan Quigley</author>
11.  <author email="jacob@openrobotics.org">Jacob Perron</author>
12.  <author email="pyo@robotis.com">Pyo</author>
13.
14.  <buildtool_depend>ament_cmake</buildtool_depend>
15.
16.  <depend>rclcpp</depend>
17.  <depend>std_msgs</depend>
18.
19.  <test_depend>ament_lint_auto</test_depend>
20.  <test_depend>ament_lint_common</test_depend>
21.
22.  <export>
23.    <build_type>ament_cmake</build_type>
24.  </export>
25. </package>
26.
```

2-2. 빌드 설정 파일 (CMakeLists.txt) :

```
1. # Set minimum required version of cmake, project name and compile options
2. cmake_minimum_required(VERSION 3.5)
3. project(my_first_ros_rclcpp_pkg)
4.
5. if(NOT CMAKE_CXX_STANDARD)
6.     set(CMAKE_CXX_STANDARD 14)
7. endif()
8.
9. if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
10.     add_compile_options(-Wall -Wextra -Wpedantic)
11. endif()
12.
13. # Find dependencies
14. find_package(ament_cmake REQUIRED)
15. find_package(rclcpp REQUIRED)
16. find_package(std_msgs REQUIRED)
17.
18. # Build
19. add_executable(helloworld_publisher src/helloworld_publisher.cpp)
20. ament_target_dependencies(helloworld_publisher rclcpp std_msgs)
21.
22. add_executable(helloworld_subscriber src/helloworld_subscriber.cpp)
23. ament_target_dependencies(helloworld_subscriber rclcpp std_msgs)
24.
25. # Install
26. install(TARGETS
27.     helloworld_publisher
28.     helloworld_subscriber
29.     DESTINATION lib/${PROJECT_NAME})
30.
31. # Test
32. if(BUILD_TESTING)
33.     find_package(ament_lint_auto REQUIRED)
34.     ament_lint_auto_find_test_dependencies()
35. endif()
36.
37. # Macro for ament package
38. ament_package()
39.
```

3. 퍼블리셔 노드의 작성

퍼블리셔 노드 예시 :

```
1. #include <chrono>
2. #include <functional>
3. #include <memory>
4. #include <string>
5.
6. #include "rclcpp/rclcpp.hpp"
7. #include "std_msgs/msg/string.hpp"
```

```

8.
9. using namespace std::chrono_literals;
10.
11.
12. class HelloworldPublisher : public rclcpp::Node
13. {
14. public:
15.     HelloworldPublisher()
16.     : Node("helloworld_publisher"), count_(0)
17.     {
18.         auto qos_profile = rclcpp::QoS(rclcpp::KeepLast(10));
19.         helloworld_publisher_ = this->create_publisher<std_msgs::msg::String>(
20.             "helloworld", qos_profile);
21.         timer_ = this->create_wall_timer(
22.             1s, std::bind(&HelloworldPublisher::publish_helloworld_msg, this));
23.     }
24.
25. private:
26.     void publish_helloworld_msg()
27.     {
28.         auto msg = std_msgs::msg::String();
29.         msg.data = "Hello World: " + std::to_string(count_++);
30.         RCLCPP_INFO(this->get_logger(), "Published message: '%s'", msg.data.c_str());
31.         helloworld_publisher_->publish(msg);
32.     }
33.     rclcpp::TimerBase::SharedPtr timer_;
34.     rclcpp::Publisher<std_msgs::msg::String>::SharedPtr helloworld_publisher_;
35.     size_t count_;
36. };
37.
38.
39. int main(int argc, char * argv[])
40. {
41.     rclcpp::init(argc, argv);
42.     auto node = std::make_shared<HelloworldPublisher>();
43.     rclcpp::spin(node);
44.     rclcpp::shutdown();
45.     return 0;
46. }
47.

```

std::bind

- std::function 과 같이 boost 라이브러리에 포함되어 있으며 (#include <functional>로 불러옴), C++11 에서 사용됨
- 함수를 매번 정의하지 않아도 되도록, 필요한 경우에만 함수를 만들어주는 기능을 제공함.

std::make_shared

- #include <memory>의 함수
- c++11 부터 지원. c++에서는 메모리 누수로부터 프로그램의 안정성을 보장하기 위해 스마트 포인터를 제공한다.
- 스마트 포인터는 생성하면 힙 메모리에 올라가며, **사용이 끝난 메모리를 자동으로 해제해준다.**
- 여러 개의 스마트 포인터에서 동일한 메모리를 가리킬 때 사용한다

4. 빌드

workspace 로 이동 이후 colcon build 명령어로 전체를 빌드

- 특정 패키지만 선택해서 빌드하려면 --packages-select 옵션 붙이기
- symlink 를 이용하려면 --symlink-install 옵션 붙이기

1. (워크스페이스내의 모든 패키지 빌드하는 방법)
2. `$ cd ~/robot_ws && colcon build --symlink-install`
- 3.
4. (특정 패키지만 빌드하는 방법)
5. `$ cd ~/robot_ws && colcon build --symlink-install --packages-select [패키지 이름 1]`
`[패키지 이름 2] [패키지 이름 N]`
- 6.
7. (특정 패키지 및 의존성 패키지를 함께 빌드하는 방법)
8. `$ cd ~/robot_ws && colcon build --symlink-install --packages-up-to [패키지 이름]`
- 9.

첫 빌드 때에는 환경 설정 파일을 불러와 실행 가능한 패키지의 노드 설정을 해주어야 한다

1. `. ~/robot_ws/install/local_setup.bash`

5. imu 센서 rviz 시각화

5-1. complementary_filter.launch.py 런치 파일 실행

[결과] : /imu 와 /imu/data 토픽이 생긴 것을 확인 완료

```
jungseong@jungseong-ubuntu:~/yahboomcar_ws$ ros2 topic list
/battery
/beep
/cmd_vel
/imu
/imu/data
/odom_raw
/parameter_events
/rosout
/scan
/servo_s1
/servo_s2
/tf
jungseong@jungseong-ubuntu:~/yahboomcar_ws$
```

+ /imu 와 /imu/data 토픽의 구분 확인 완료



orientation:	orientation:
x: 0.0	x: 0.0016877452787003894
y: 0.0	y: -0.005755550131823930
z: 0.0	z: 0.07980303745710005
w: 1.0	w: 0.9967878010577791
orientation_covariance:	orientation_covariance:
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
angular_velocity:	angular_velocity:
x: 0.0	x: -1.8521532647451447e-00
y: 0.0	y: -0.25243750305345e-07
z: 0.0	z: 0.0
angular_velocity_covariance:	angular_velocity_covariance:
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
- 0.0	- 0.0
linear_acceleration:	linear_acceleration:
x: 0.00579791009420117	x: 0.00579791009420117
y: 0.0	y: 0.0
z: 9.777508817138072	z: 9.777508817138072

5-2. Euler -> Quaternion 로 변환을 위한 complementary_filter.launch.py 런치 파일과 rviz 시각화를 위한 imu_rviz.launch.py 런치 파일 실행

```
jungseong@jungseong-ubuntu:~/yahboomcar_ws$ ros2 launch imu_complementary_filter complementary_filter.launch.py
[INFO] [launch]: All log files can be found below /home/jungseong/.ros/log/2025-04-19-10-37-07-475339-jungseong-ubuntu-4444
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [complementary_filter_node-1]: process started with pid [4445]
[complementary_filter_node-1] [INFO] [1745026627.608198999] [complementary_filter_gain_node]: Starting ComplementaryFilterROS
```



```
jungseong@jungseong-ubuntu:~/yahboomcar_ws$ ros2 launch imu_complementary_filter imu_
rviz.launch.py
[INFO] [launch]: All log files can be found below /home/jungseong/.ros/log/2025-04-19
-10-37-32-105026-jungseong-ubuntu-4516
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [static_transform_publisher-1]: process started with pid [4517]
[INFO] [static_transform_publisher-2]: process started with pid [4519]
[INFO] [rviz2-3]: process started with pid [4521]
[static_transform_publisher-1] [WARN] [1745026652.224065534] []: Old-style arguments
are deprecated; see --help for new-style arguments
[static_transform_publisher-2] [WARN] [1745026652.232070107] []: Old-style arguments
are deprecated; see --help for new-style arguments
[static_transform_publisher-2] [INFO] [1745026652.313311024] [imu_link_to_imu_frame]:
Spinning until stopped - publishing transform
```

5-3. 결과

rviz 시각화가 잘 이루어지는 것을 확인할 수 있었다

