



ROS2 Python programming



RLmodel

ybbaek



Contents

- 1. ROS2 Jupyter notebook**
- 2. workspace, Package 만들기**
- 3. Topic handling**
- 4. launch file test**
- 5. Action**
- 6. Service**
- 7. cv_bridge**
- 8. Appendix**



프로그램 실습

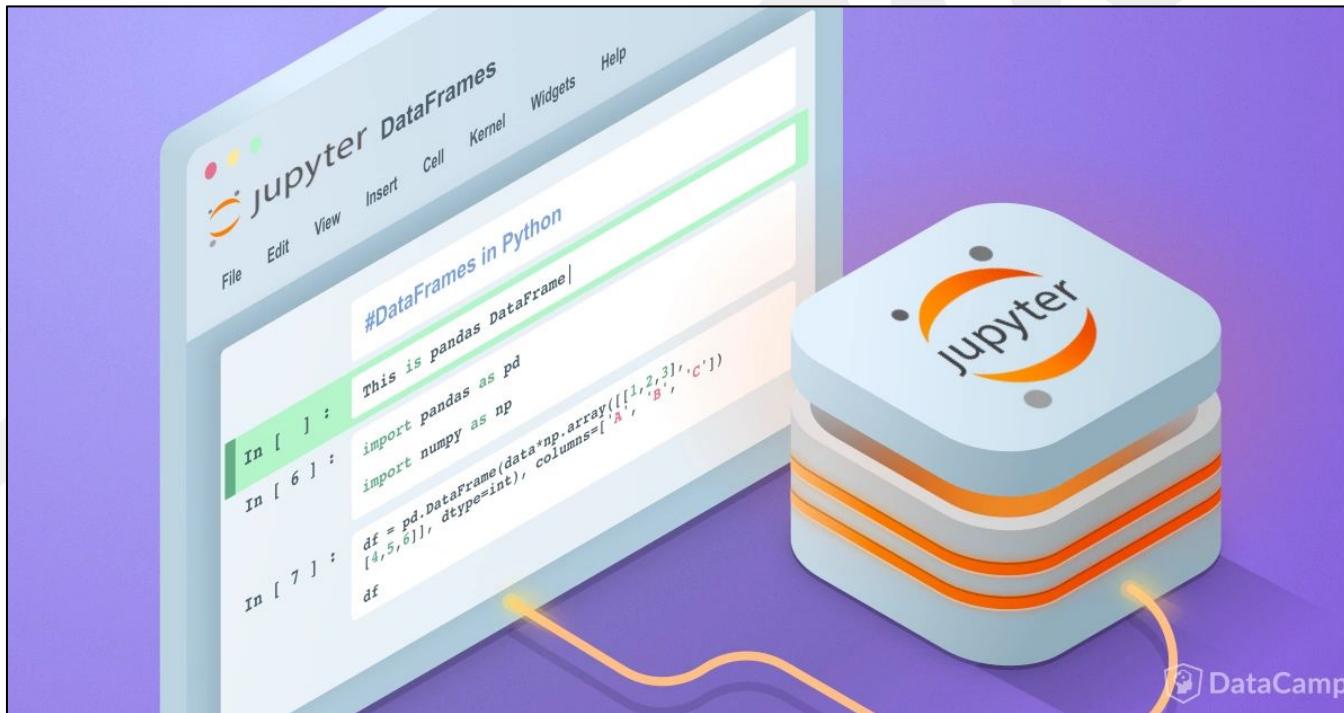
ROS2 Jupyter notebook



Jupyter notebook

설치 방법:

```
>sudo apt install python3-pip  
>pip3 install —upgrade pip  
>pip3 install jupyter ipywidgets pyyaml bqplot  
>sudo apt install jupyter-core
```





ROS2 Python 실습

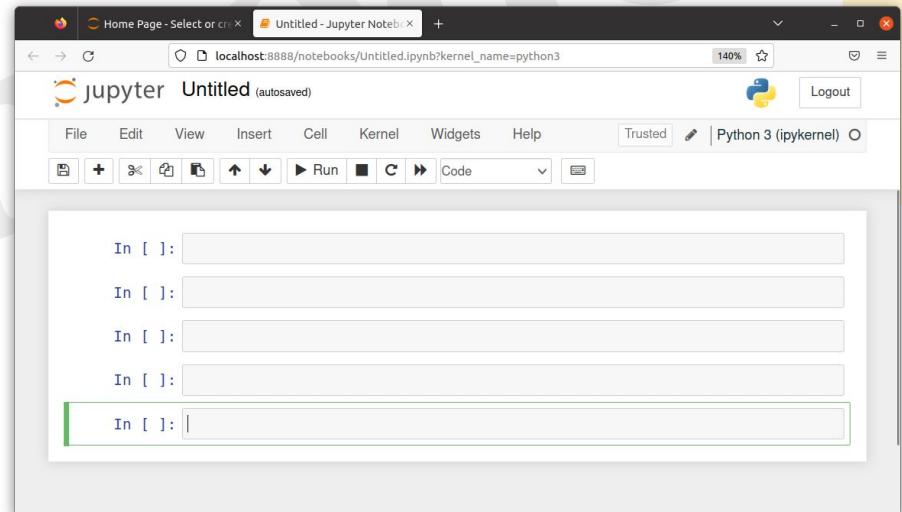
Jupyter notebook 실행

>jupyter notebook

>mark down 언어지원

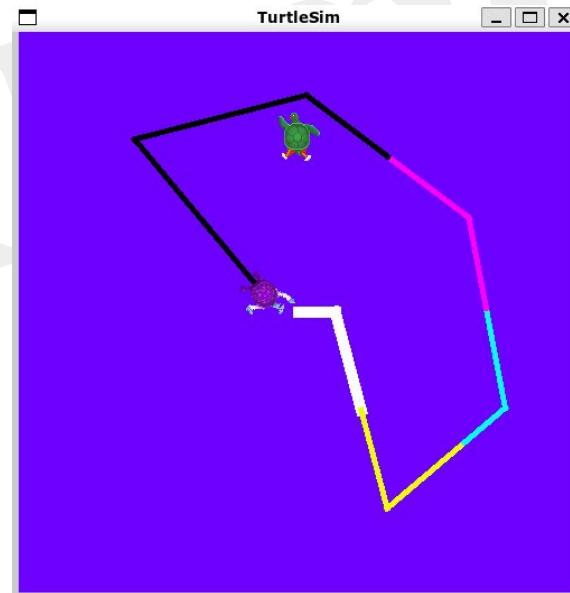
```
byb76@rlhp:~/ros2test/src/ROS2_edu/my_turtlesim_pkg/Jupyter
source /home/byb76/.ros_menu/plugins.bashrc/dds_bashrc 1
Source Eclipse Cyclone DDS successfully!
(ROS 2 foxy) byb76@rlhp:~/ros2test/src/ROS2_edu/my_turtlesim_pkg/Jupyter$ jupyter notebook
[I 11:24:09.403 NotebookApp] Serving notebooks from local directory: /home/byb76/ros2test/src/ROS2_edu/my_turtlesim_pkg/Jupyter
[I 11:24:09.404 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 11:24:09.404 NotebookApp] http://localhost:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:09.404 NotebookApp] or http://127.0.0.1:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:09.404 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip configuration).
[...]
[I 11:24:09.472 NotebookApp]

To access the notebook, open this file in a browser:
  file:///home/byb76/.local/share/jupyter/runtime/nbserver-19369-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
  or http://127.0.0.1:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:32.422 NotebookApp] Creating new notebook in
[...]
[I 11:24:34.649 NotebookApp] Kernel started: 5c962139-1b34-4037-a826-51e90c0253f9, name: python3
[...]
[I 11:26:34.570 NotebookApp] Saving file at /Untitled.ipynb
/home/byb76/.local/lib/python3.8/site-packages/nbformat/_init_.py:129: MissingIDFieldWarning: Code cell is missing an id field, this will become a hard error in future nbformat versions. You may want to use 'normalize()' on your notebooks before validations (available since nbformat 5.1.4). Previous versions of nbformat are fixing this issue transparently, and will stop doing so in the future.
  validate(nb)
/home/byb76/.local/lib/python3.8/site-packages/notebook/services/contents/manager.py:353: MissingIDFieldWarning: Code cell is missing an id field, this will become a hard error in future nbformat versions. You may want to use 'normalize()' on your notebooks before validations (available since nbformat 5.1.4). Previous version
```



Example1

turtlesim pose subscribe



ROS2 Python 실습

topic 수신

상세: turtlesim pose 토픽 수신

The screenshot shows a Jupyter Notebook interface with the title "topic_sub-test". The code in the notebook is as follows:

```
In [1]: import rclpy as rp
from turtlesim.msg import Pose

In [2]: rp.init()
test_node = rp.create_node('sub_test')

In [3]: def callback(data):
    print("turtle pose",data)
    print("X: ", data.x)
    print("Y: ", data.y)
    print("Heading: ",data.theta)

In [4]: test_node.create_subscription(Pose, '/turtle1/pose', callback, 10)
Out[4]: <rclpy.subscription.Subscription at 0x7f0664fd4640>

In [7]: rp.spin_once(test_node)
        ...

In [8]: test_node.destroy_node()

In [9]: rp.shutdown()

In [ ]:
```

ROS2 Python 실습

topic 수신

준비: \$Source /opt/ros/foxy/setup.bash

순서: python code 작성 > 노드 생성

확인: \$ros2 node list

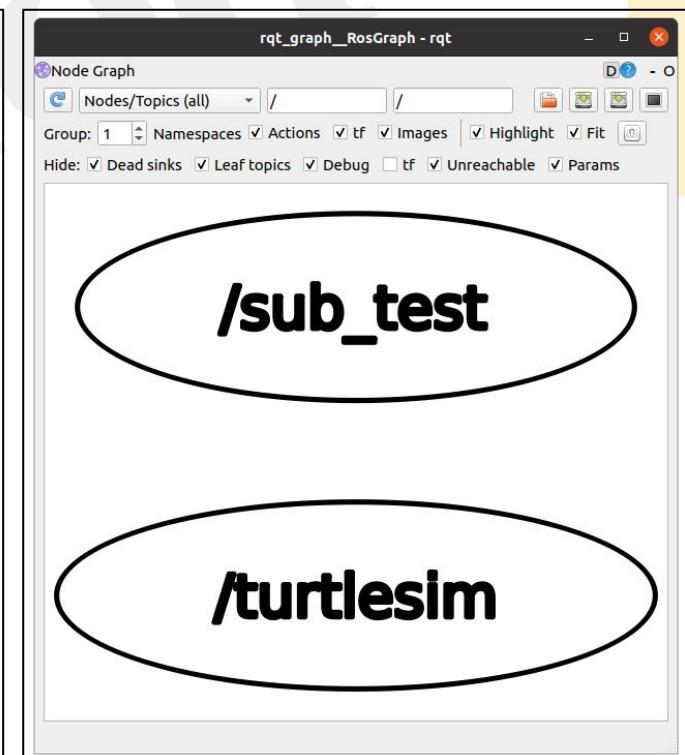
The screenshot shows a Jupyter Notebook interface with the title "topic_sub_test". The notebook has two cells:

```
In [4]: import rclpy as rp
from turtlesim.msg import Pose
```

```
In [5]: rp.init()
test_node = rp.create_node('sub_test')

1674529369.490302 [0]    python3: using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
```

The output of the fifth cell is highlighted in pink.



ROS2 Python 실습

topic 코드

> 토픽 수신 (subscription)

rp.spin_once(sub_node)

The screenshot shows a Jupyter Notebook interface with the title "topic_sub_test". The notebook contains the following code:

```
In [4]: import rclpy as rp
from turtlesim.msg import Pose

In [5]: rp.init()
test_node = rp.create_node('sub_test')
1674529369.490302 [0]    python3: using network interface wlp2s0 (udp/19
2.168.0.19) selected arbitrarily from: wlp2s0, docker0

In [6]: def callback(data):
    print("turtle pose")
    print("X: ", data.x)
    print("Y: ", data.y)
    print("Heading: ", data.theta)

In [7]: test_node.create_subscription(Pose, '/turtle1/pose', callback, 10)
Out[7]: <rclpy.subscription.Subscription at 0x7f5c6c4fdac0>

In [8]: rp.spin_once(test_node)
turtle pose turtlesim.msg.Pose(x=4.840552806854248, y=5.830162525177002,
theta=-0.7792094945907593, linear_velocity=0.0, angular_velocity=0.0)
X: 4.840552806854248
Y: 5.830162525177002
Heading: -0.7792094945907593
```

A callout box labeled "수신된 토픽 내용 확인" (Check received topic content) points to the output of In [7].

수신된 토픽 내용
확인

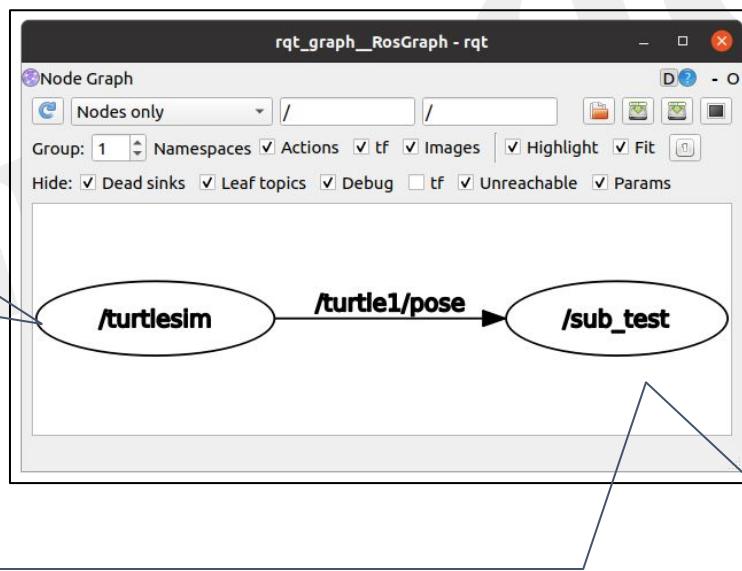
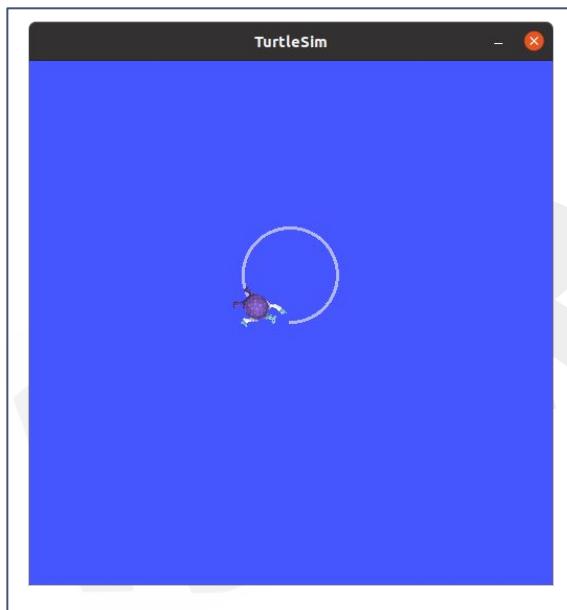


ROS2 Python 실습

topic 코드

> 토픽 구독

```
rp.spin_once(sub_node)
```



```
In [8]: rp.spin_once(test_node)
```

```
turtle pose turtlesim.msg.Pose(x=4.840552806854248, y=5.830162525177002,  
theta=-0.7792094945907593, linear_velocity=0.0, angular_velocity=0.0)  
X: 4.840552806854248  
Y: 5.830162525177002  
Heading: -0.7792094945907593
```



ROS2 Python 실습

topic 코드

> 링크: https://github.com/RLmodel/ROS2_edu/blob/main/my_turtlesim_pkg/Jupyter/py_sub-pose.ipynb

The screenshot shows a Jupyter Notebook interface with the following code:

```
In [7]: import rclpy as rp
from turtlesim.msg import Pose

In [8]: rp.init()
sub_node = rp.create_node('sub_test')

In [9]: def callback(data):
    print("turtle pose",data)
    print("X: ", data.x)
    print("Y: ", data.y)
    print("Heading: ",data.theta)

In [10]: sub_node.create_subscription(Pose, '/turtle1/pose', callback, 10)

Out[10]: <rclpy.subscription.Subscription at 0x7f0aec0360a0>

In [11]: rp.spin_once(test_node)

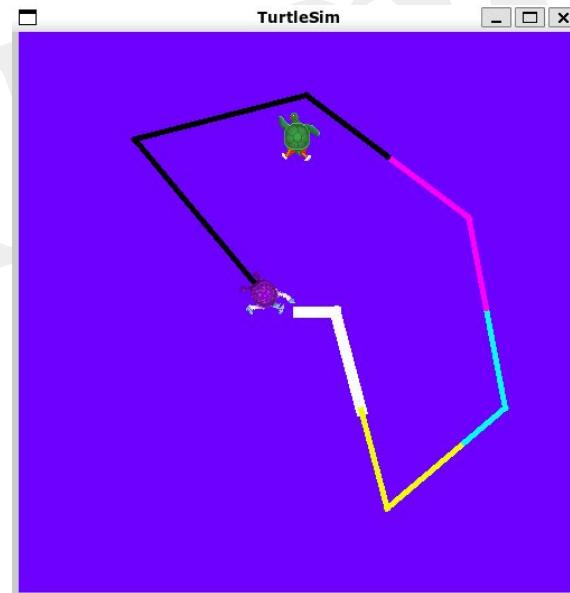
turtle pose turtlesim.msg.Pose(x=5.503449440002441, y=7.406180381774902, theta=-2.033808469772339, linear_velocity=0.0, angular_velocity=0.4000000059604645)
X: 5.503449440002441
Y: 7.406180381774902
Heading: -2.033808469772339

In [12]: sub_node.destroy_node()

In [13]: rp.shutdown()
```

Example2

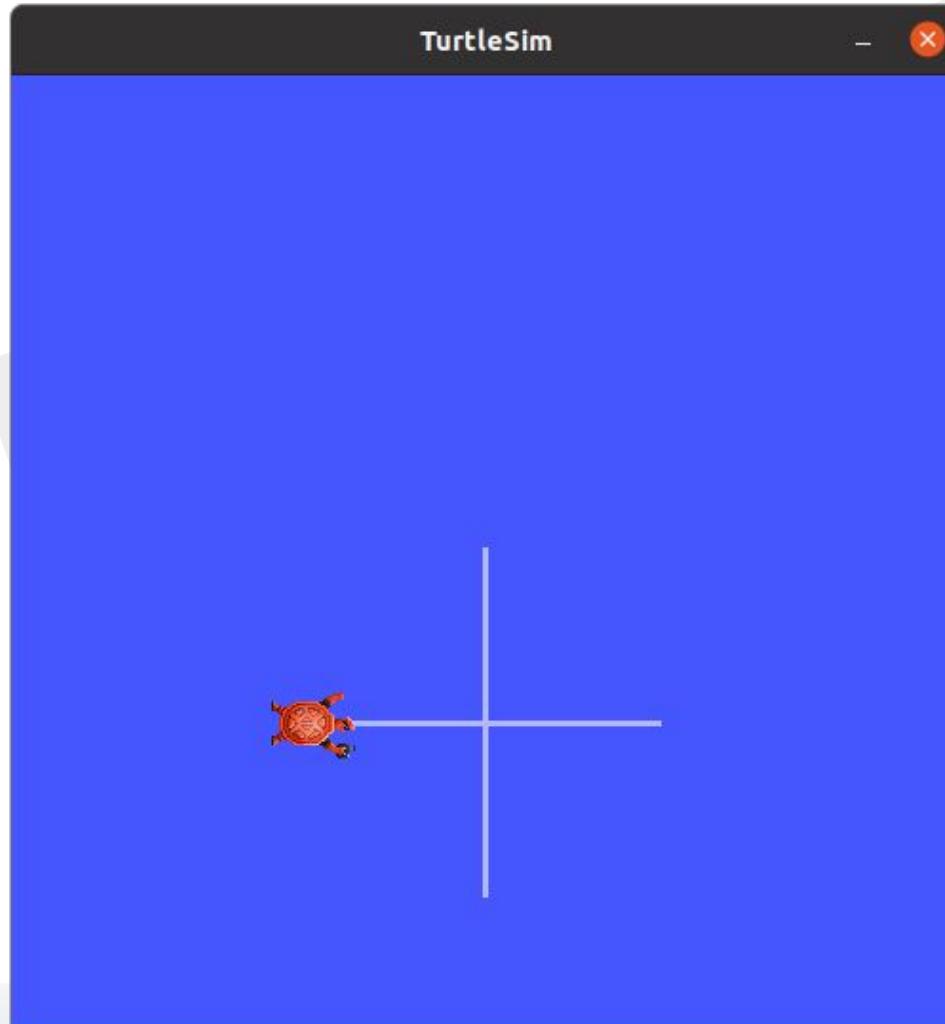
turtlesim Twist msg publish



ROS2 Python 실습

Topic 생성 코드 / turtlesim

기능: Twist msg (cmd_vel) publish



ROS2 Python 실습

Topic 생성 코드

기능: Twist msg (cmd_vel) publish

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the URL is `localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3`. The notebook has one cell selected, labeled In [5], which contains the Python code `print(msg)`. The rest of the cells show the setup of the node and publisher.

```
In [1]: import rclpy as rp
       from geometry_msgs.msg import Twist

In [2]: rp.init()

In [3]: pub_node = rp.create_node('pub_test')

In [4]: msg = Twist()

In [5]: print(msg)
      ...

In [10]: msg.linear.x = 0.0
        msg.linear.y = 2.0

In [7]: pub = pub_node.create_publisher(Twist, '/turtle1/cmd_vel', 10)

In [11]: pub.publish(msg)

In [12]: pub_node.destroy_node()

In [13]: rp.shutdown()
```

ROS2 Python 실습

Topic 생성 코드

기능: Twist msg (cmd_vel) publish

Link: https://github.com/RLmodel/ROS2_edu/blob/main/my_turtlesim_pkg/Jupyter/py_pub_twist.ipynb

The screenshot shows a GitHub repository page for 'ROS2_edu'. The main content area displays a Jupyter notebook titled 'py_pub_twist.ipynb'. The notebook contains the following Python code:

```
In [1]: import rclpy as rp
from geometry_msgs.msg import Twist

In [14]: rp.init()
pub_node = rp.create_node('pub_test')

In [15]: msg = Twist()

In [35]: print(msg)

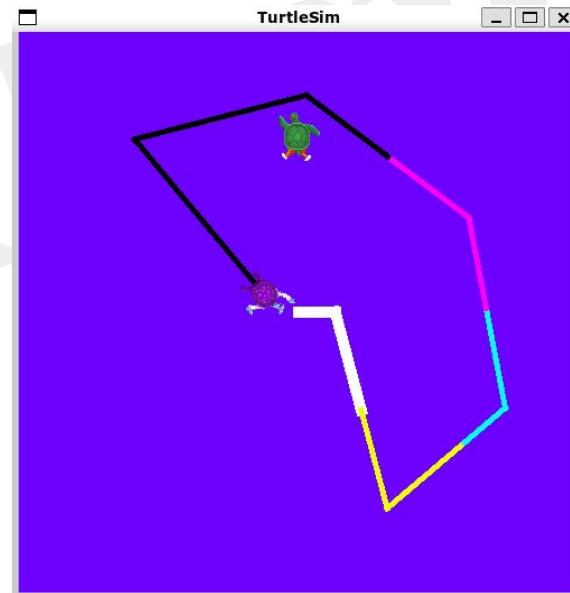
geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))

In [16]: msg.linear.x = 0.0
msg.linear.y = -2.0

In [17]: pub = pub_node.create_publisher(Twist, '/turtle1/cmd_vel', 10)
```

Example3

turtlesim service call



ROS2 Python 실습

service 코드

순서: node 생성 > client 생성 > client call >

The screenshot shows a Jupyter Notebook interface with the title "service_test" (autosaved). The notebook contains the following code:

```
In [1]: import rclpy as rp
from turtlesim.srv import TeleportAbsolute

In [2]: rp.init()
test_node = rp.create_node('client_test')
...
In [3]: service_name = '/turtle1/teleport_absolute'
cli = test_node.create_client(TeleportAbsolute, service_name)

In [4]: req = TeleportAbsolute.Request()

In [5]: req
...
In [12]: req.x = 5.
req.y = 2.
req.theta = 3.14
req
...
In [13]: cli.call_async(req)
...
In [11]: rp.spin_once(test_node)
```



ROS2 Python 실습

service 코드

link: https://github.com/RLmodel/ROS2_edu/blob/main/my_turtlesim_pkg/Jupyter/service_test.ipynb

The screenshot shows a Jupyter Notebook interface with the following code:

```
In [1]: import rclpy as rp
from turtlesim.srv import TeleportAbsolute

In [2]: rp.init()
srv_node = rp.create_node('client_test')

In [3]: service_name = '/turtle1/teleport_absolute'
cli = srv_node.create_client(TeleportAbsolute, service_name)

In [4]: req = TeleportAbsolute.Request()

In [5]: req

Out[5]: turtlesim.srv.TeleportAbsolute_Request(x=0.0, y=0.0, theta=0.0)

In [11]: req.x = 5.
req.y = 5.
req.theta = 1.14
req

Out[11]: turtlesim.srv.TeleportAbsolute_Request(x=5.0, y=5.0, theta=1.14)
```



프로그램 실습

ROS2 Python

ros2/rclpy

rclpy (ROS Client Library for Python)

⋮ 2

77 Contributors 72 Issues 156 Stars 161 Forks

ROS2 실습

Topic publish

<String msg>

link:

<https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html#write-the-publisher-node>

ROS 2 Documentation: Foxy

You're reading the documentation for a version of ROS 2 that has reached its EOL (end-of-life), and is no longer officially supported. If you want up-to-date information, please have a look at Iron.

Writing a simple publisher and subscriber (Python)

Goal: Create and run a publisher and subscriber node using Python.

Tutorial level: Beginner

Time: 20 minutes

Contents

- Background
- Prerequisites
- Tasks
 - 1 Create a package
 - 2 Write the publisher node
 - 3 Write the subscriber node
 - 4 Build and run
- Summary



ROS2 실습

Topic publish

<String msg>

link: <https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html#write-the-publisher-node>

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String

class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5 # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1

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()
```



ROS2 실습

workspace 생성

참고 링크: <https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html>

<순서>

- 1 Source ROS 2 environment
- 2 Create a new directory
- 3 Clone a sample repo
- 4 Resolve dependencies
- 5 Build the workspace with colcon
- 6 Source the overlay
- 7 Modify the overlay

underlay : ROS 2 설치의 기본 패키지와 라이브러리를 포함하는 디렉토리

overlay : 개발자가 추가로 설치한 패키지와 라이브러리를 포함하는 디렉토리

동작: ROS2는 먼저 **overlay** 디렉터리에서 찾아보고, 찾을 수 없으면 **underlay** 디렉토리에서 패키지를 찾습니다.



ROS2 실습

package 생성

참고링크: <https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html>

<순서>

```
$ mkdir -p ~/ros2_ws/src  
$ cd ~/ros2_ws/src  
$ git clone xxx  
$ cd..  
$ sudo apt install python3-rosdep  
$ sudo rosdep init  
$ rosdep update  
$ rosdep install -i --from-path src --rosdistro foxy -y  
$ colcon build  
$ . install/local_setup.bash
```

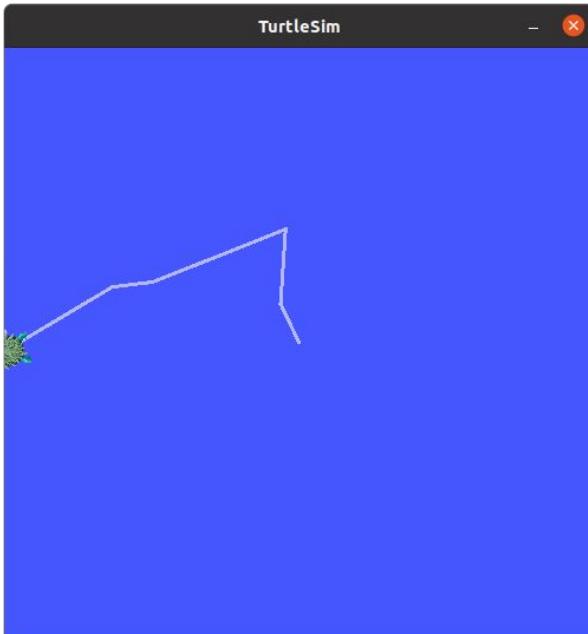
ROS2 실습

workspace 생성

생성: mkdir -p ~/ros2_ws/src

워크스페이스 : ros2_ws

link: https://github.com/yunbum/ROS2_edu.git



```
byb76@rlhp: ~/add_ros2_ws
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
```

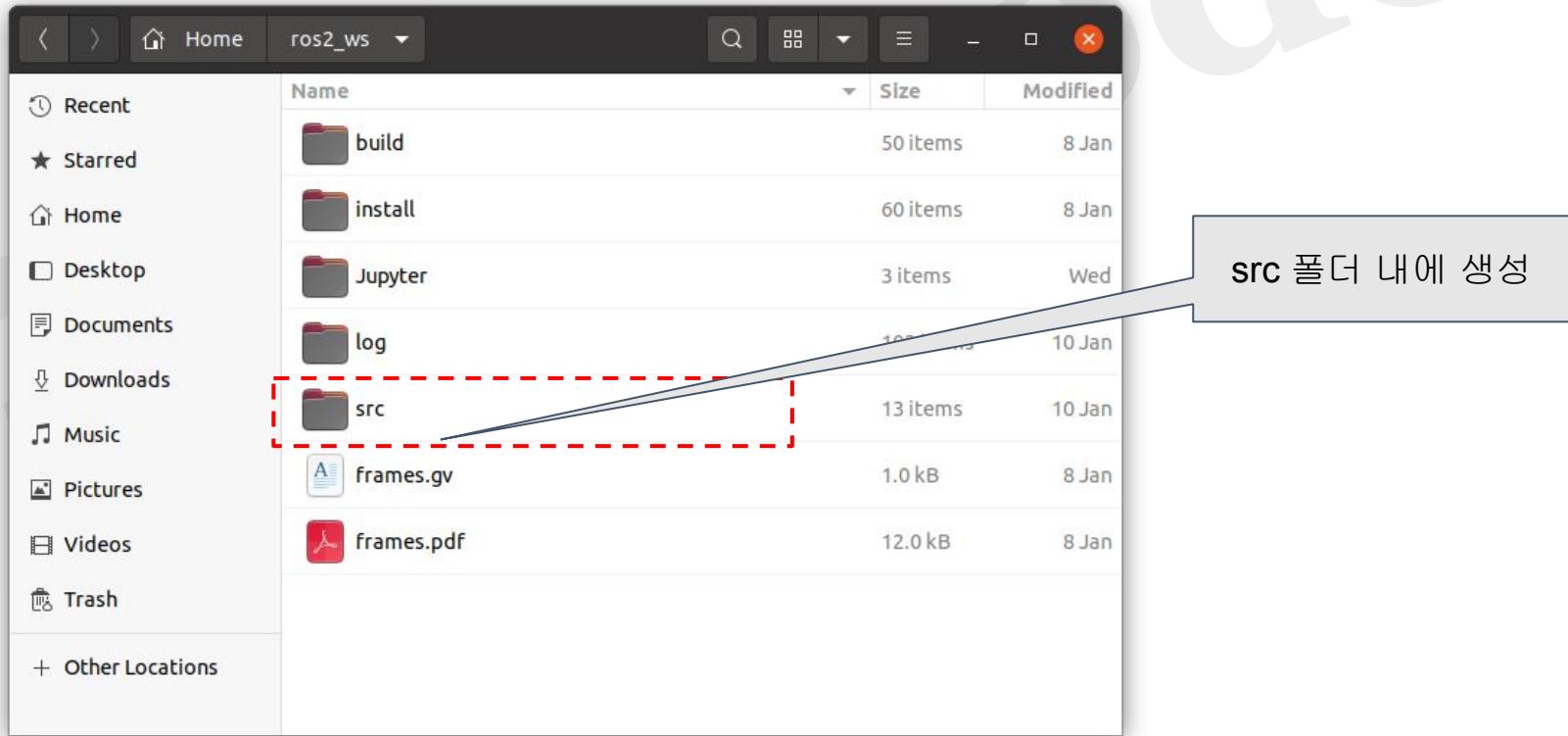
ROS2 실습

Package 생성

참고 링크: <https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Creating-Your-First-ROS2-Package.html>

```
$ cd ~/ros2_ws/src
```

```
$ ros2 pkg create --build-type ament_python pkg_name
```



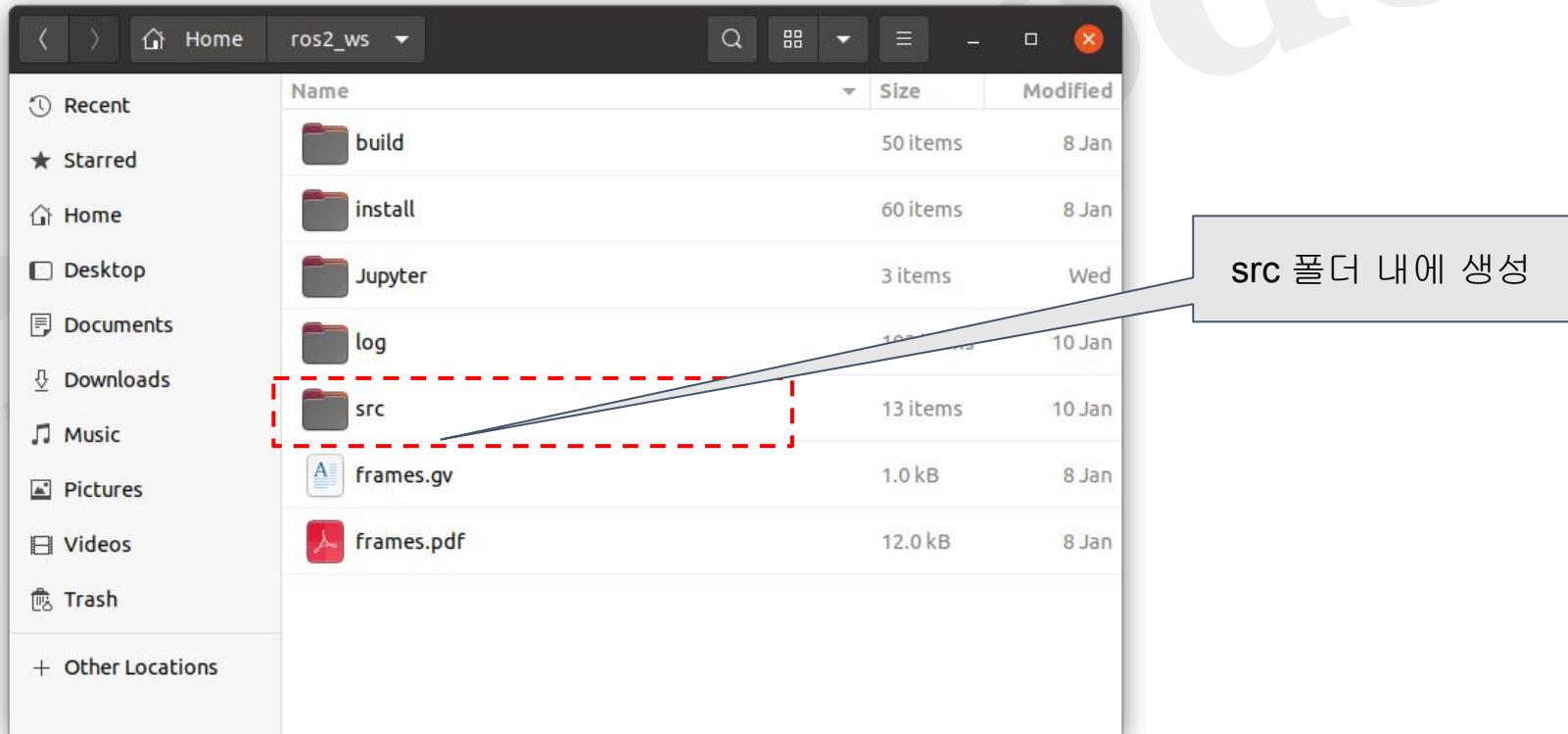
ROS2 실습

Topic publish

<twist msg cmd_vel>

\$ cd ~/ros2_ws/src

\$ ros2 pkg create --build-type ament_python py_twistpub



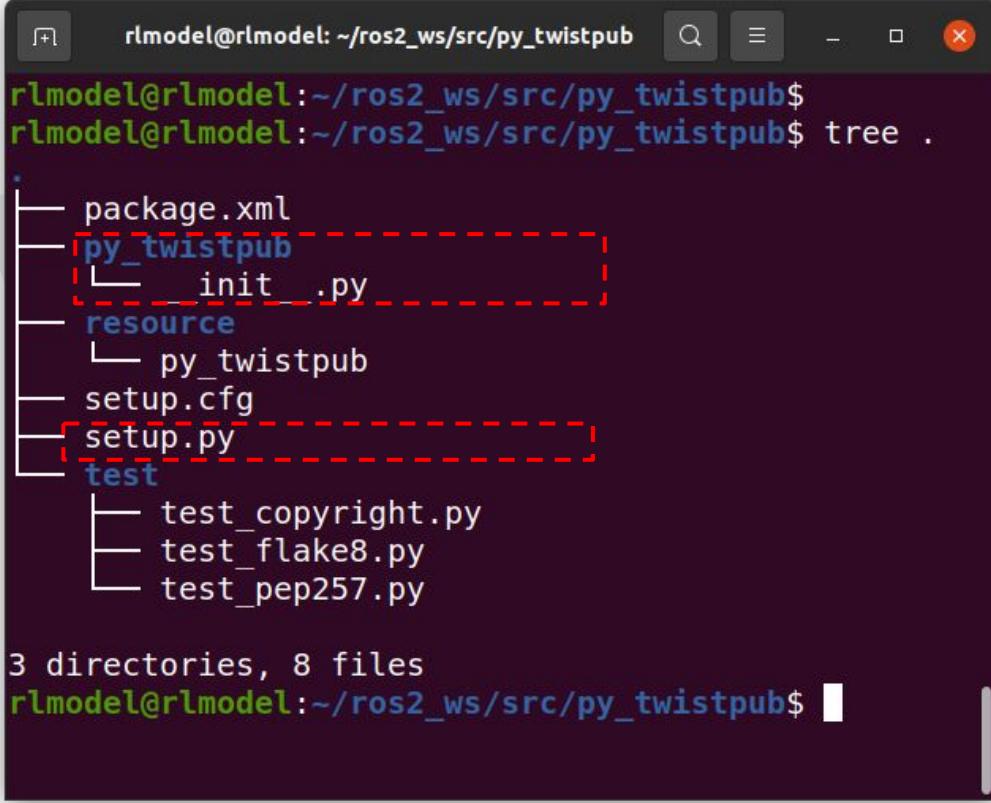
ROS2 실습

Topic publish

<twist msg cmd_vel>

\$ tree .

> sudo apt-get install tree



```
rlmodel@rlmodel:~/ros2_ws/src/py_twistpub$ tree .
.
├── package.xml
└── py_twistpub
    ├── __init__.py
    ├── resource
    │   └── py_twistpub
    ├── setup.cfg
    ├── setup.py
    └── test
        ├── test_copyright.py
        ├── test_flake8.py
        └── test_pep257.py

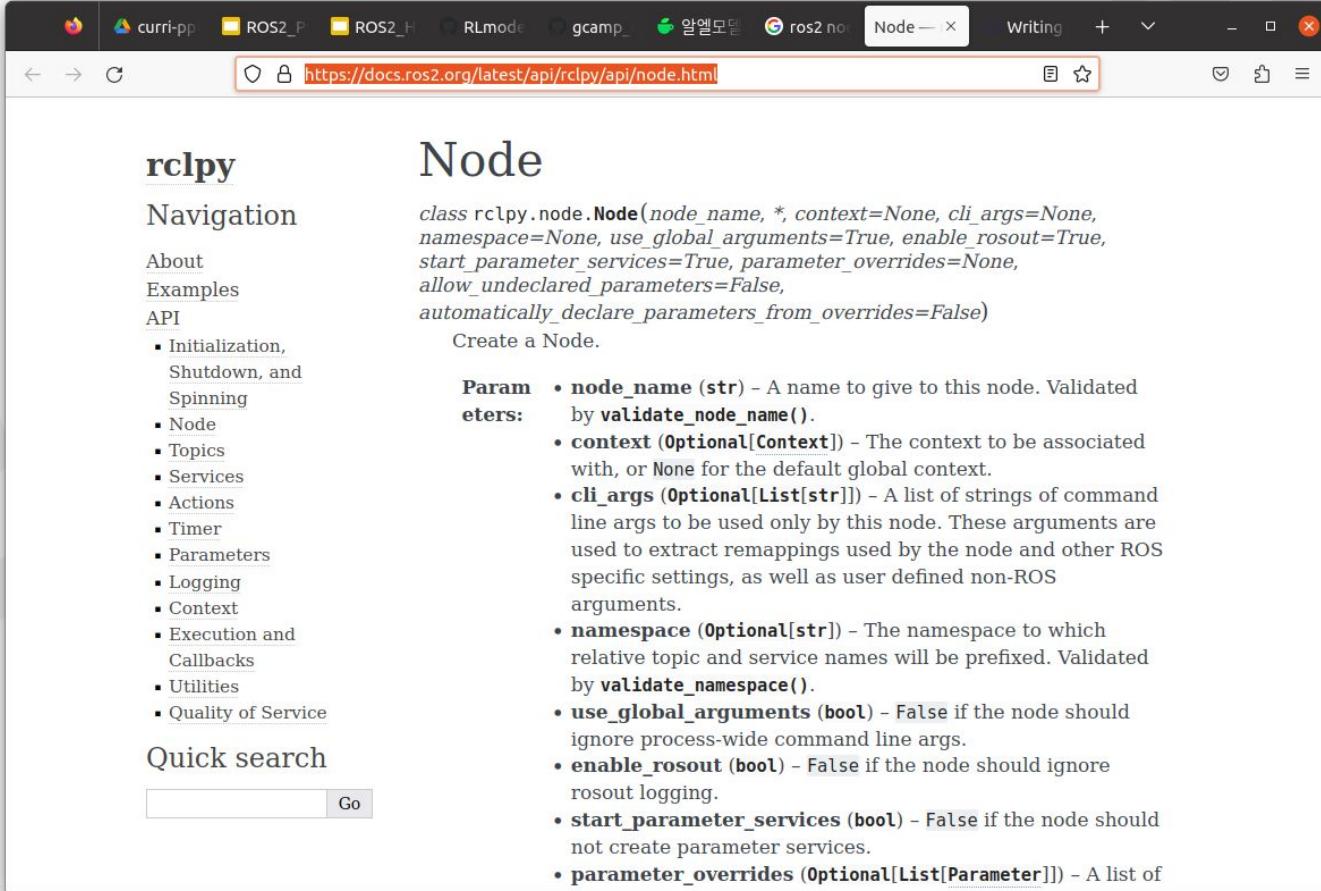
3 directories, 8 files
rlmodel@rlmodel:~/ros2_ws/src/py_twistpub$
```

ROS2 실습

Topic publish

<twist msg cmd_vel>

link: <https://docs.ros2.org/latest/api/rclpy/api/node.html>



The screenshot shows a web browser window displaying the official ROS2 documentation for the `rclpy.Node` class. The URL in the address bar is <https://docs.ros2.org/latest/api/rclpy/api/node.html>. The page title is "Node". On the left, there's a sidebar with navigation links for "rclpy", "Navigation", "About", "Examples", and "API", which is expanded to show sub-links like "Initialization", "Shutdown, and Spinning", "Node", "Topics", etc. Below the sidebar is a "Quick search" input field with a "Go" button. The main content area starts with the class definition:

```
class rclpy.node.Node(node_name, *, context=None, cli_args=None, namespace=None, use_global_arguments=True, enable_rosout=True, start_parameter_services=True, parameter_overrides=None, allow_undeclared_parameters=False, automatically_declare_parameters_from_overrides=False)
```

Below the code, there's a brief description: "Create a Node." Then, under the heading "Parameters:", a detailed list of parameters is provided:

- **node_name (str)** – A name to give to this node. Validated by `validate_node_name()`.
- **context (Optional[Context])** – The context to be associated with, or None for the default global context.
- **cli_args (Optional[List[str]])** – A list of strings of command line args to be used only by this node. These arguments are used to extract remappings used by the node and other ROS specific settings, as well as user defined non-ROS arguments.
- **namespace (Optional[str])** – The namespace to which relative topic and service names will be prefixed. Validated by `validate_namespace()`.
- **use_global_arguments (bool)** – False if the node should ignore process-wide command line args.
- **enable_rosout (bool)** – False if the node should ignore rosout logging.
- **start_parameter_services (bool)** – False if the node should not create parameter services.
- **parameter_overrides (Optional[List[Parameter]])** – A list of

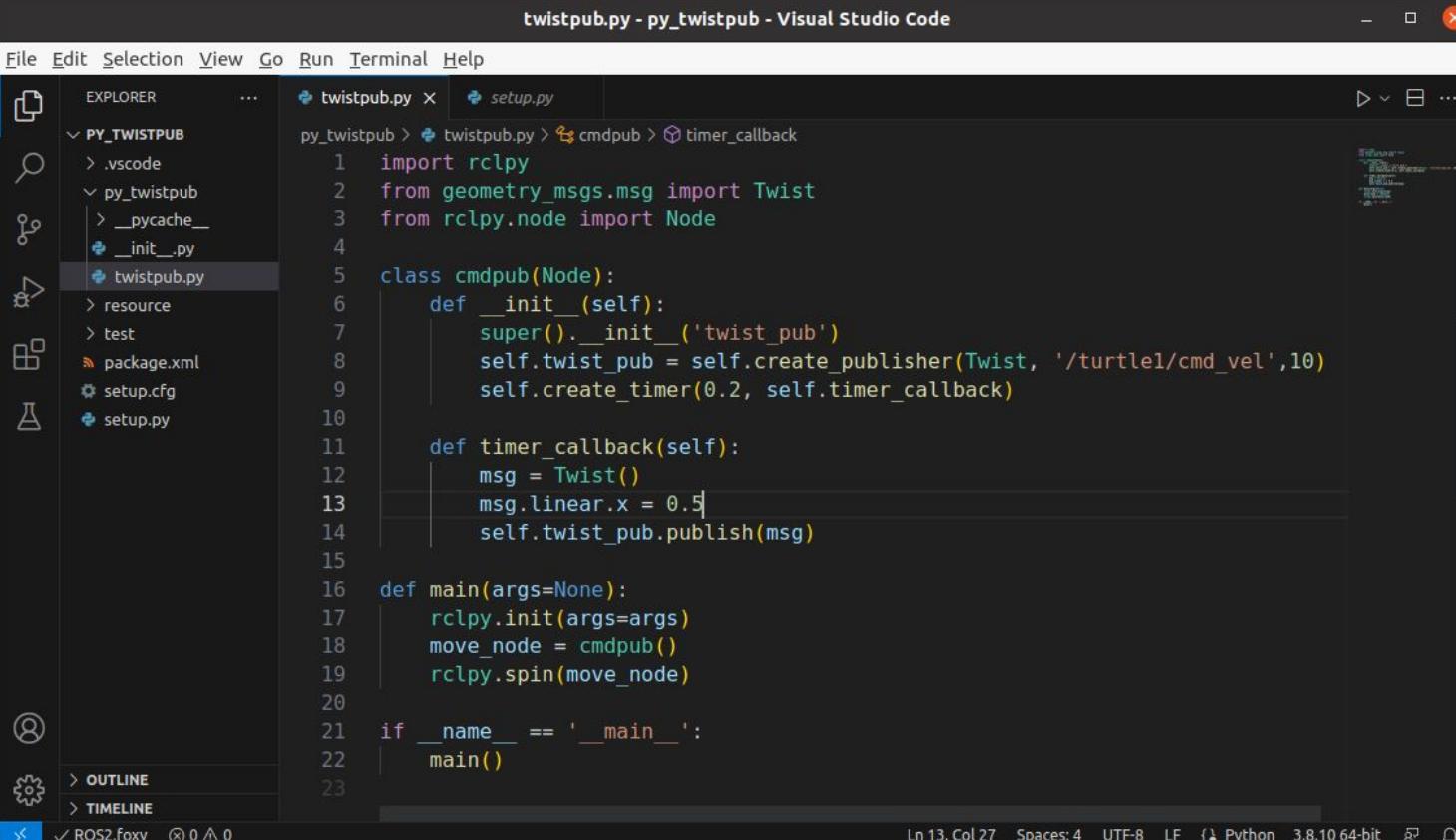
ROS2 실습

Topic publish

<twist msg cmd_vel>

\$ code .

>twistpub.py



```
twistpub.py - py_twistpub - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER      twistpub.py x  setup.py
PY_TWISTPUB
  .vscode
  py_twistpub
    __pycache__
    __init__.py
    twistpub.py
  resource
  test
  package.xml
  setup.cfg
  setup.py
py_twistpub > twistpub.py > cmdpub > timer_callback
1 import rclpy
2 from geometry_msgs.msg import Twist
3 from rclpy.node import Node
4
5 class cmdpub(Node):
6     def __init__(self):
7         super().__init__('twist_pub')
8         self.twist_pub = self.create_publisher(Twist, '/turtle1/cmd_vel', 10)
9         self.create_timer(0.2, self.timer_callback)
10
11     def timer_callback(self):
12         msg = Twist()
13         msg.linear.x = 0.5
14         self.twist_pub.publish(msg)
15
16     def main(args=None):
17         rclpy.init(args=args)
18         move_node = cmdpub()
19         rclpy.spin(move_node)
20
21     if __name__ == '__main__':
22         main()
23
```

Ln 13, Col 27 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit ⚙️ ⚙️ ⚙️

ROS2 실습



Topic publish

<twist msg cmd_vel>

```
$ code .
```

>setup.py

The screenshot shows the Visual Studio Code interface with the title bar "setup.py - py_twistpub - Visual Studio Code". The left sidebar contains icons for Explorer, Search, Open, Resource, Test, Package XML, Setup Configuration, and Outline/Timeline. The Explorer view shows a project structure with a folder "PY_TWI..." containing "vscode", "py_twistpub" (which includes "_init_.py" and "twistpub.py"), "resource", "test", "package.xml", "setup.cfg", and "setup.py" (which is currently selected). The main editor area displays the following Python code:

```
from setuptools import setup
package_name = 'py_twistpub'
setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='rlmodel',
    maintainer_email='ybbaek@rlmodel.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'twistpub = py_twistpub.twistpub:main'
        ],
    },
)
```

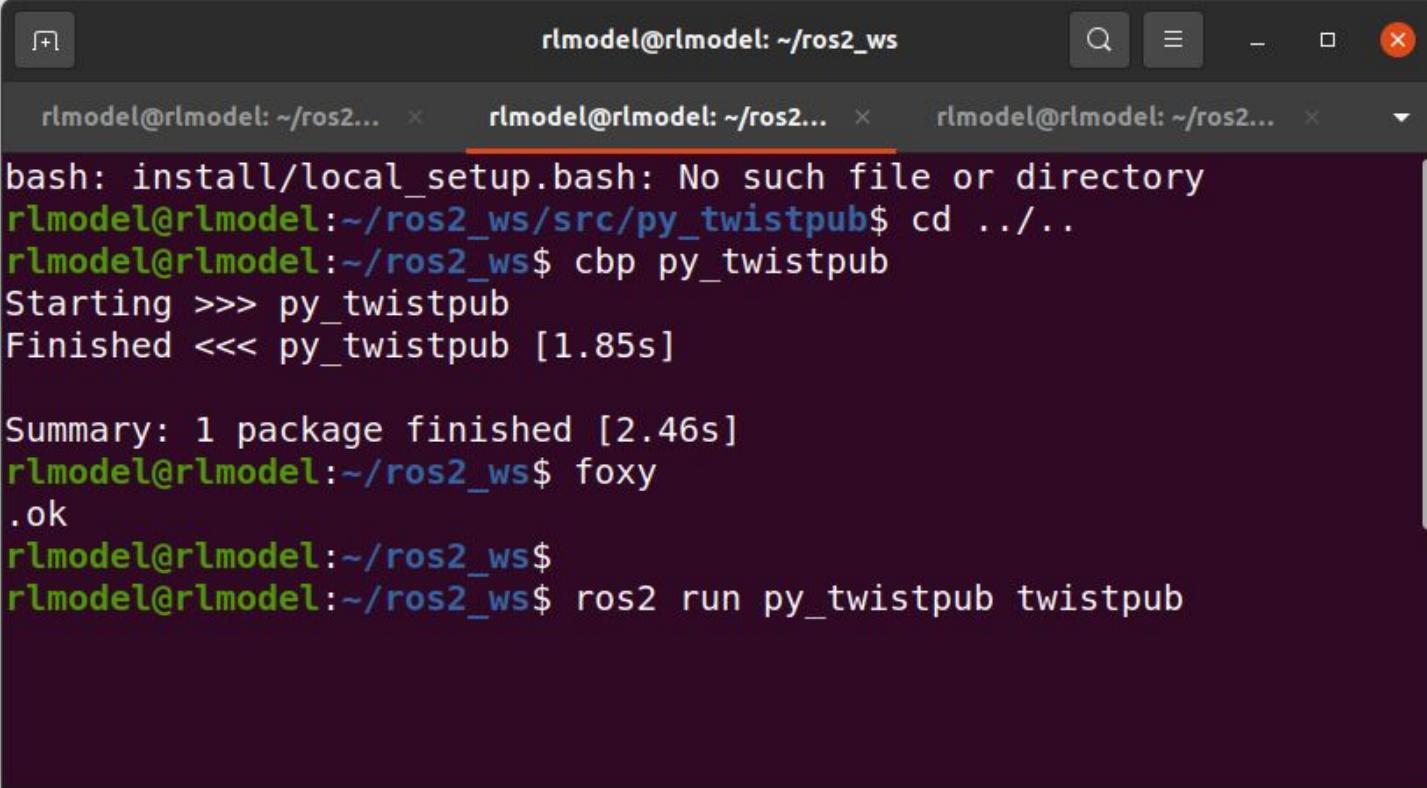


ROS2 실습

Topic publish

<twist msg cmd_vel>

```
$ colcon build --packages-select py_twistpub  
$ ros2 run py_twistpub twistpub
```



The screenshot shows a terminal window with three tabs open, all titled "rlmodel@rlmodel: ~/ros2_ws". The central tab is active and displays the following command-line session:

```
bash: install/local_setup.bash: No such file or directory  
rlmodel@rlmodel:~/ros2_ws/src/py_twistpub$ cd ../../  
rlmodel@rlmodel:~/ros2_ws$ cbp py_twistpub  
Starting >>> py_twistpub  
Finished <<< py_twistpub [1.85s]  
  
Summary: 1 package finished [2.46s]  
rlmodel@rlmodel:~/ros2_ws$ foxy  
.ok  
rlmodel@rlmodel:~/ros2_ws$  
rlmodel@rlmodel:~/ros2_ws$ ros2 run py_twistpub twistpub
```

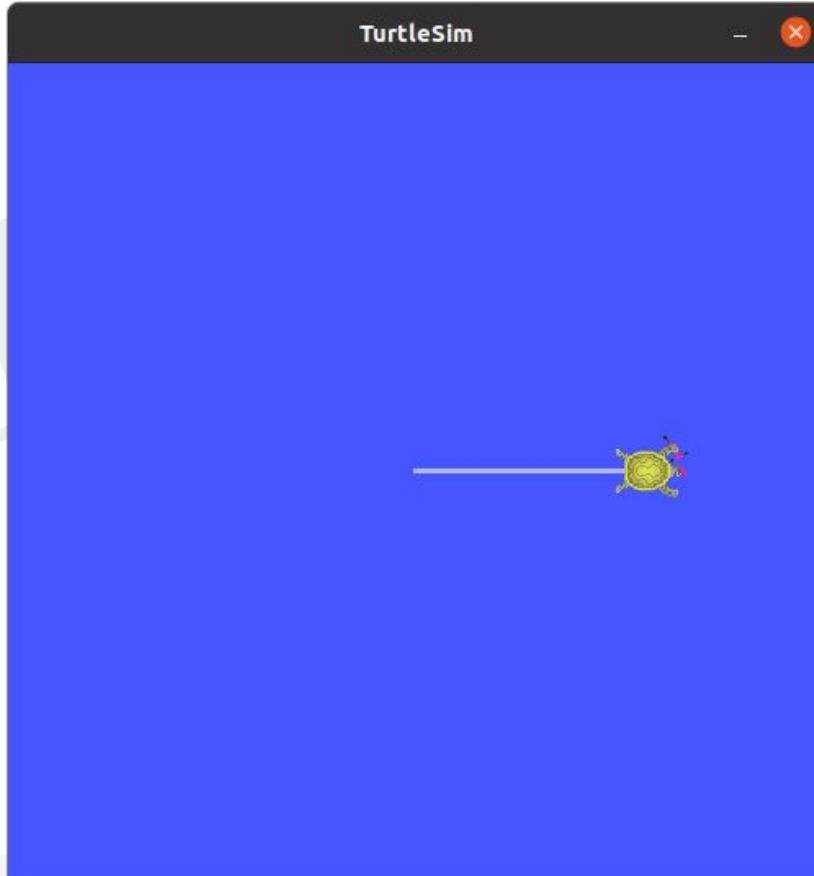
ROS2 실습

Topic publish

<twist msg cmd_vel>

```
$ colcon build --packages-select py_twistpub
```

```
$ ros2 run py_twistpub twistpub
```

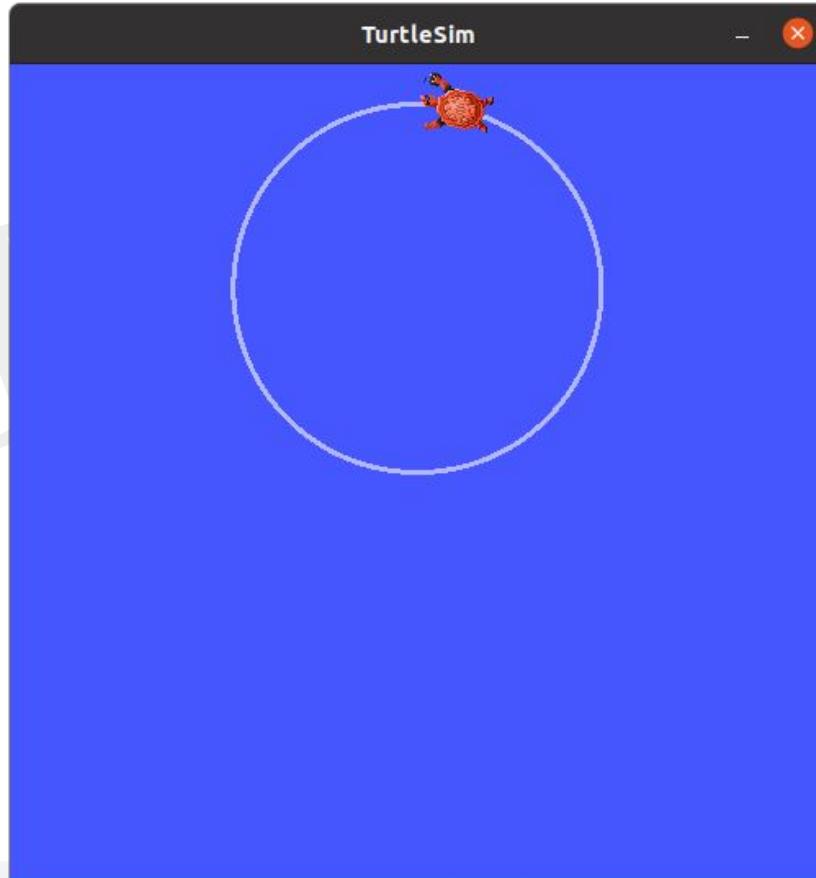


ROS2 실습

Topic publish

<twist msg cmd_vel>

```
$ colcon build --packages-select py_twistpub  
$ ros2 run py_twistpub twistpub
```



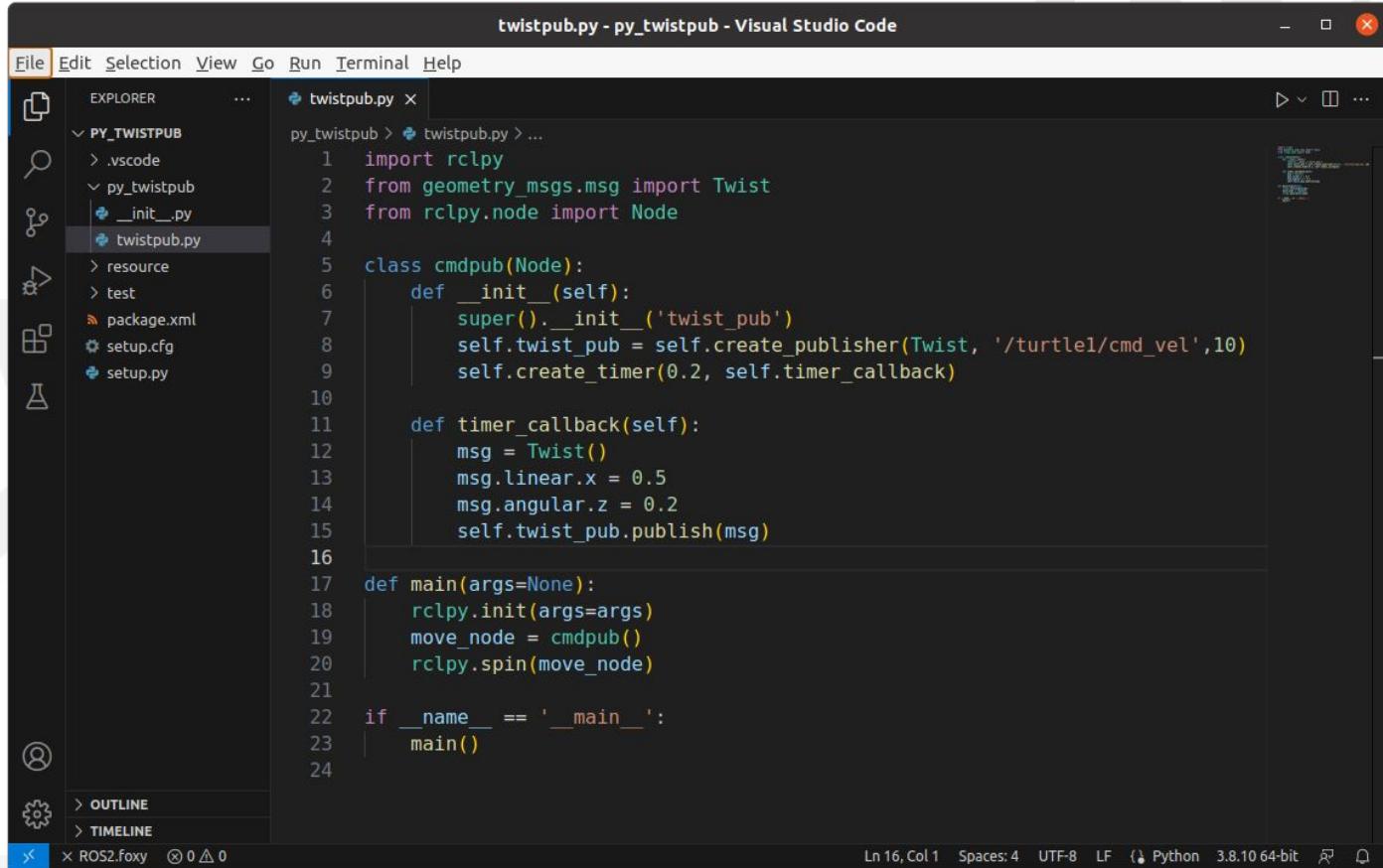
ROS2 실습

Topic publish

<twist msg cmd_vel>

\$ code .

>twistpub.py



The screenshot shows a Visual Studio Code window with the title "twistpub.py - py_twistpub - Visual Studio Code". The file "twistpub.py" is open in the editor. The code implements a ROS2 node that publishes a Twist message at 10 Hz. The node creates a timer that updates the message's linear and angular velocities.

```
twistpub.py - py_twistpub - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER    twistpub.py x
PY_TWISTPUB
  .vscode
  py_twistpub
    __init__.py
    twistpub.py
  resource
  test
  package.xml
  setup.cfg
  setup.py
twistpub.py
1 import rclpy
2 from geometry_msgs.msg import Twist
3 from rclpy.node import Node
4
5 class cmdpub(Node):
6     def __init__(self):
7         super().__init__('twist_pub')
8         self.twist_pub = self.create_publisher(Twist, '/turtle1/cmd_vel', 10)
9         self.create_timer(0.2, self.timer_callback)
10
11     def timer_callback(self):
12         msg = Twist()
13         msg.linear.x = 0.5
14         msg.angular.z = 0.2
15         self.twist_pub.publish(msg)
16
17     def main(args=None):
18         rclpy.init(args=args)
19         move_node = cmdpub()
20         rclpy.spin(move_node)
21
22     if __name__ == '__main__':
23         main()

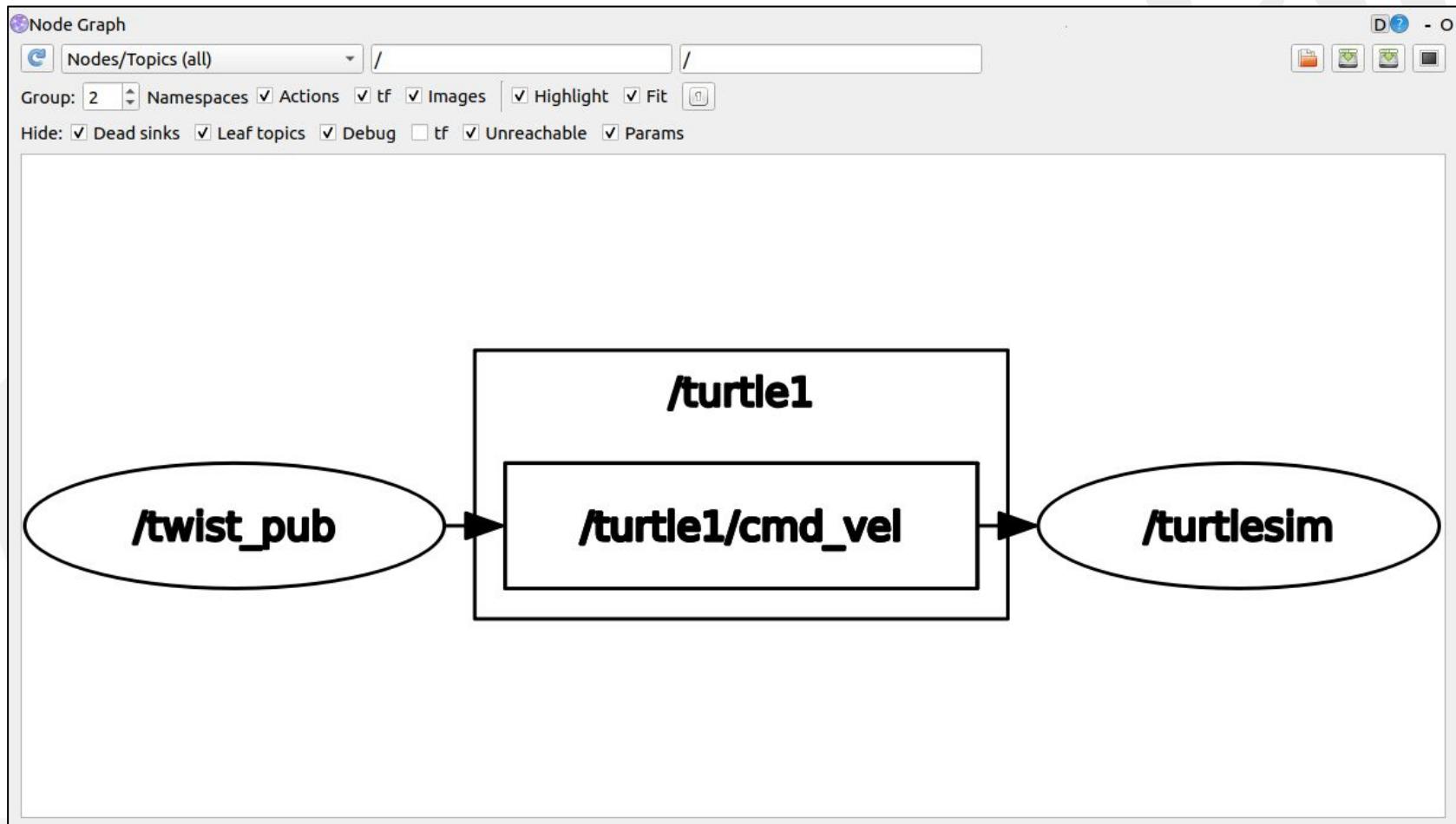
Ln 16, Col 1  Spaces: 4  UTF-8  LF  Python 3.8.10 64-bit  ⚙  ⚙
```

ROS2 실습

Topic publish

<twist msg cmd_vel>

\$ rqt_graph





ROS2 실습

Topic publish

<twist msg cmd_vel>

```
$ ros2 topic echo /turtle1/cmd_vel
```

The screenshot shows a terminal window titled "rlmodel@rlmodel: ~/ros2_ws". The window contains five tabs, all labeled "rlmodel@r...". The active tab is the rightmost one. The terminal output displays a JSON-like message structure:

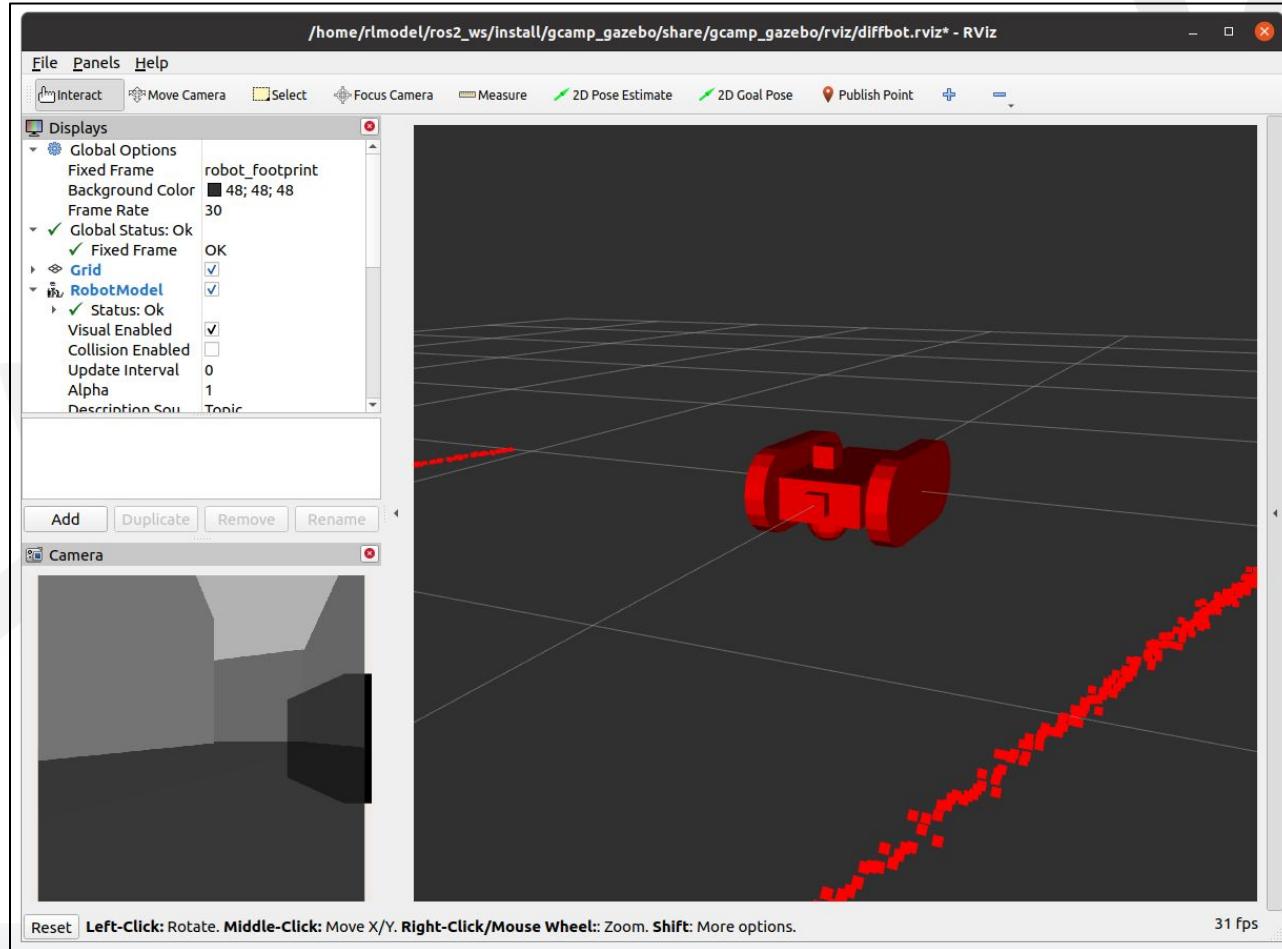
```
angular:
  x: 0.0
  y: 0.0
  z: 0.2
---
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.2
---
```

ROS2 실습

Topic publish

<twist msg cmd_vel>

example > diffbot turning





Break Time

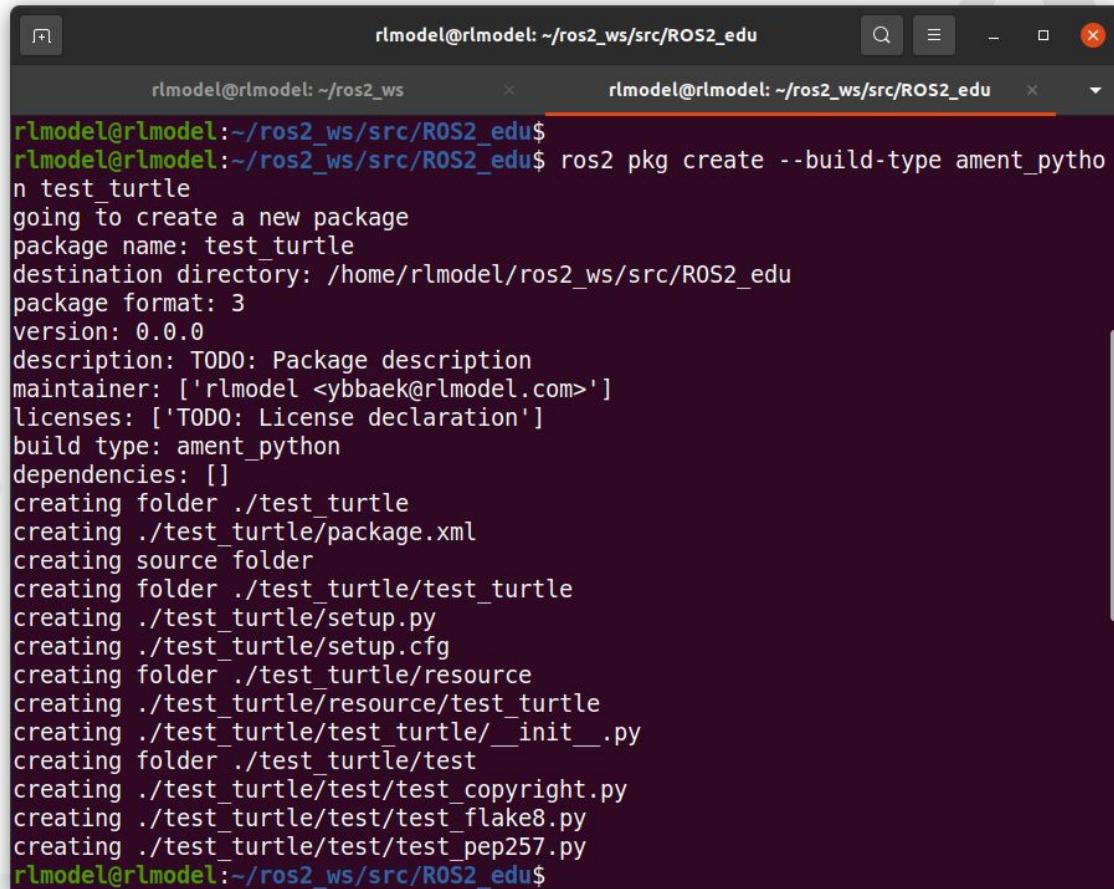


ROS2 실습

Package 생성

<turtlesim_subscriber>

>new pkg - test_turtle



```
rlmodel@rlmodel: ~/ros2_ws
```

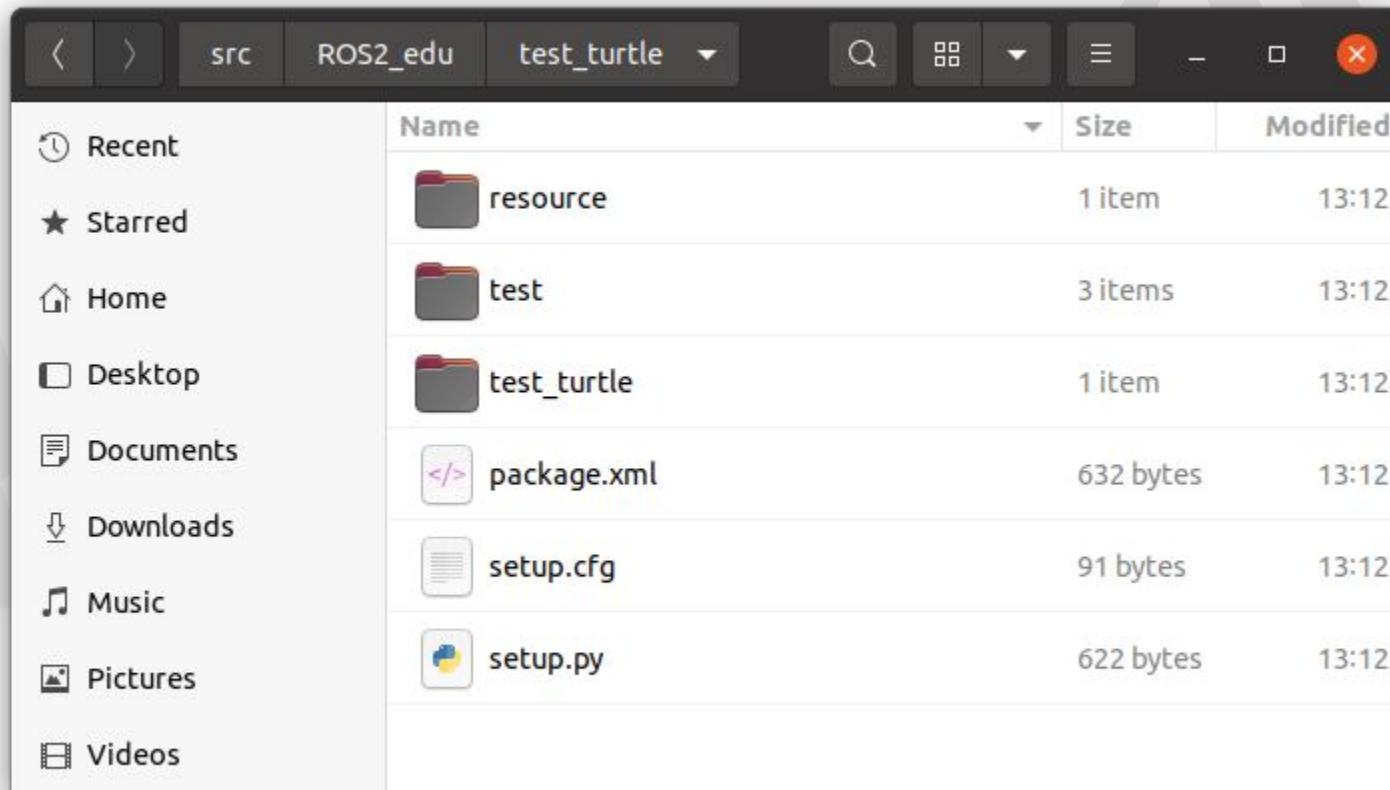
```
rlmodel@rlmodel: ~/ros2_ws/src/ROS2_edu$ ros2 pkg create --build-type ament_python test_turtle
going to create a new package
package name: test_turtle
destination directory: /home/rlmodel/ros2_ws/src/ROS2_edu
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['rlmodel <ybbaek@rlmodel.com>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: []
creating folder ./test_turtle
creating ./test_turtle/package.xml
creating source folder
creating folder ./test_turtle/test_turtle
creating ./test_turtle/setup.py
creating ./test_turtle/setup.cfg
creating folder ./test_turtle/resource
creating ./test_turtle/resource/test_turtle
creating ./test_turtle/test_turtle/_init__.py
creating folder ./test_turtle/test
creating ./test_turtle/test/test_copyright.py
creating ./test_turtle/test/test_flake8.py
creating ./test_turtle/test/test_pep257.py
rlmodel@rlmodel: ~/ros2_ws/src/ROS2_edu$
```

ROS2 실습

Package 생성

<turtlesim_subscriber>

>check folder & files



ROS2 실습

Package 생성

<turtlesim_subscriber>

>check folder & files

The screenshot shows a code editor window titled "package.xml" with the file path "~/ros2_ws/src/ROS2_edu/test_turtle". The window has standard OS X-style controls (Open, Save, Close) at the top. The code itself is an XML configuration file for a ROS2 package:

```
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>test_turtle</name>
5   <version>0.0.0</version>
6   <description>TODO: Package description</description>
7   <maintainer email="ybbaek@rlmodel.com">rlmodel</maintainer>
8   <license>TODO: License declaration</license>
9
10  <test_depend>ament_copyright</test_depend>
11  <test_depend>ament_flake8</test_depend>
12  <test_depend>ament_pep257</test_depend>
13  <test_depend>python3-pytest</test_depend>
14
15  <export>
16    <build_type>ament_python</build_type>
17  </export>
18 </package>
```

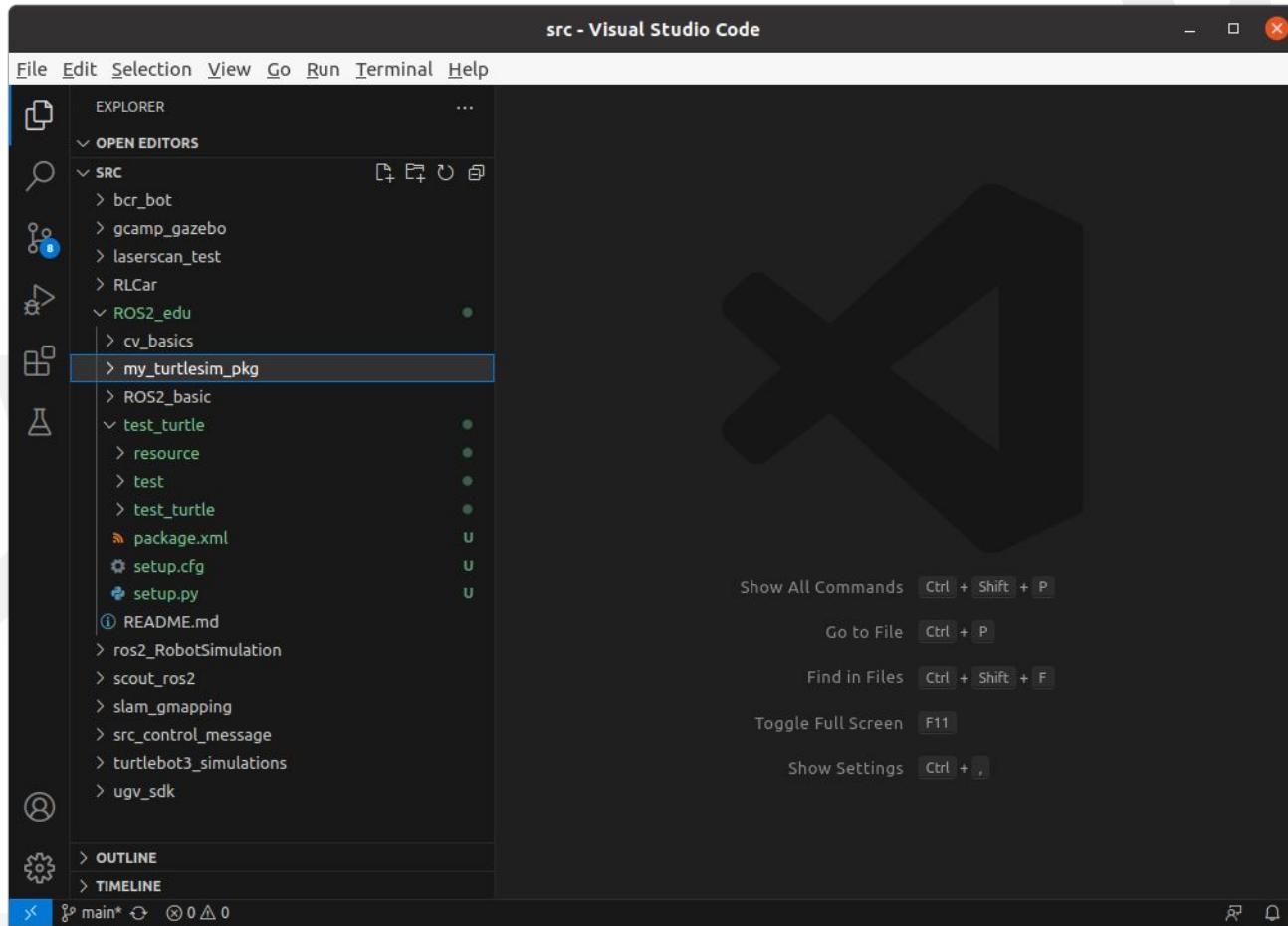
At the bottom of the editor, there are tabs for "XML" and "INS", along with status indicators for "Tab Width: 8", "Ln 1, Col 1", and a dropdown menu.

ROS2 실습

Package 생성

<turtlesim_subscriber>

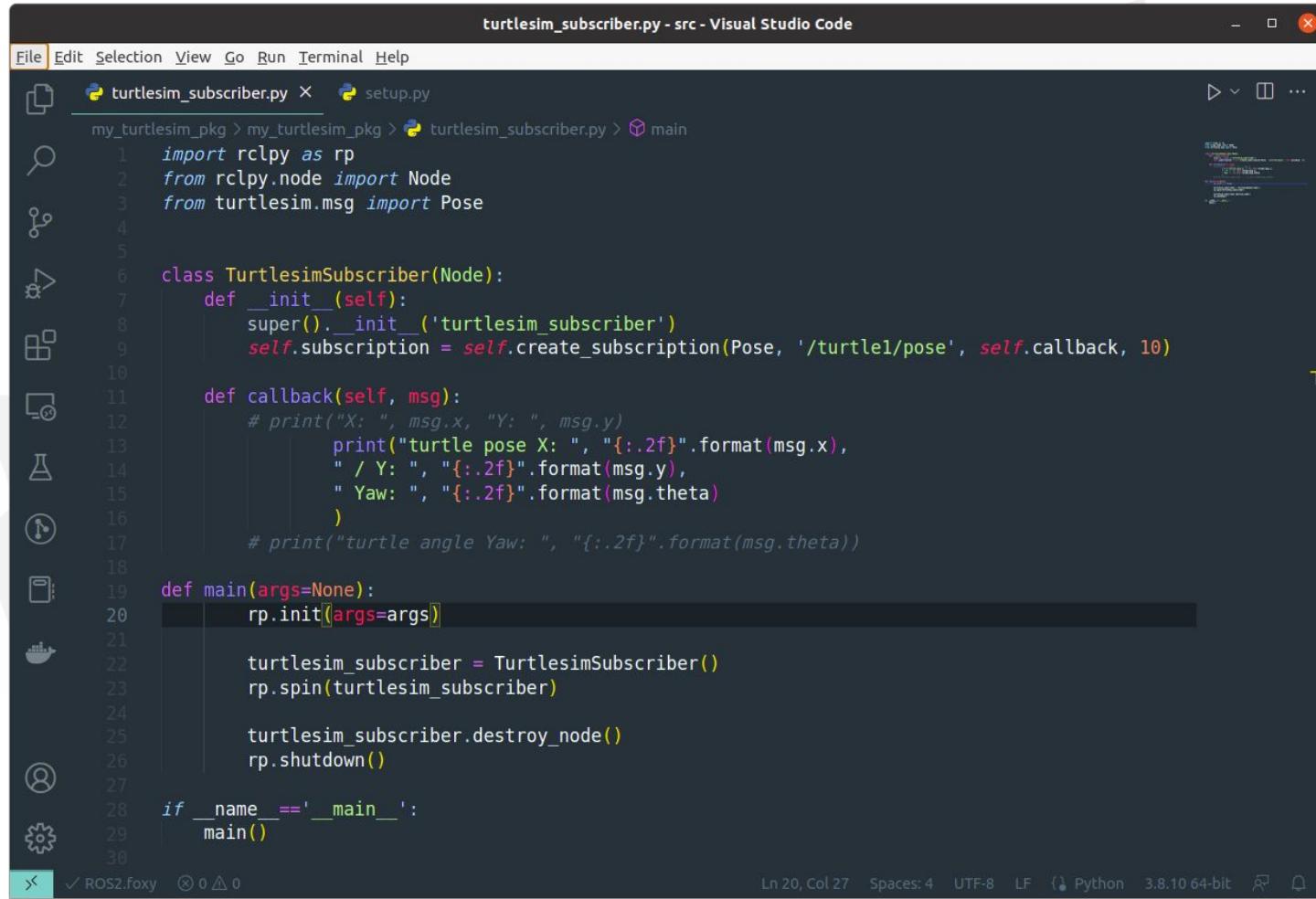
\$ code .



ROS2 실습

Python code 생성

- turtlesim_subscriber.py



The screenshot shows a Visual Studio Code window with the title "turtlesim_subscriber.py - src - Visual Studio Code". The file content is as follows:

```
File Edit Selection View Go Run Terminal Help
turtlesim_subscriber.py X setup.py
my_turtlesim_pkg > my_turtlesim_pkg > turtlesim_subscriber.py > main
1 import rclpy as rp
2 from rclpy.node import Node
3 from turtlesim.msg import Pose
4
5
6 class TurtlesimSubscriber(Node):
7     def __init__(self):
8         super().__init__('turtlesim_subscriber')
9         self.subscription = self.create_subscription(Pose, '/turtle1/pose', self.callback, 10)
10
11     def callback(self, msg):
12         # print("X: ", msg.x, "Y: ", msg.y)
13         #     print("turtle pose X: ", "{:.2f}".format(msg.x),
14         #          " / Y: ", "{:.2f}".format(msg.y),
15         #          " Yaw: ", "{:.2f}".format(msg.theta)
16         #          )
17         # print("turtle angle Yaw: ", "{:.2f}".format(msg.theta))
18
19     def main(args=None):
20         rp.init(args=args)
21
22         turtlesim_subscriber = TurtlesimSubscriber()
23         rp.spin(turtlesim_subscriber)
24
25         turtlesim_subscriber.destroy_node()
26         rp.shutdown()
27
28     if __name__ == '__main__':
29         main()
30
```

The status bar at the bottom indicates: ROS2.foxy ✓ 0 △ 0 Ln 20, Col 27 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit



ROS2 실습

Python code 생성

- turtlesim_subscriber.py

Node class 정보

link: <https://docs.ros2.org/latest/api/rclpy/api/node.html>

rclpy

- Navigation
- About
- Examples
- API
 - Initialization, Shutdown, and Spinning
 - Node
 - Topics
 - Services
 - Actions
 - Timer
 - Parameters
 - Logging
 - Context
 - Execution and Callbacks
 - Utilities
 - Quality of Service

Node

```
class rclpy.node.Node(node_name, *, context=None, cli_args=None, namespace=None, use_global_arguments=True, enable_rosout=True, start_parameter_services=True, parameter_overrides=None, allow_undeclared_parameters=False, automatically_declare_parameters_from_overrides=False)
```

Create a Node.

Parameters:

- **node_name (str)** - A name to give to this node. Validated by `validate_node_name()`.
- **context (Optional[Context])** - The context to be associated with, or None for the default global context.
- **cli_args (Optional[List[str]])** - A list of strings of command line args to be used only by this node. These arguments are used to extract remappings used by the node and other ROS specific settings, as well as user defined non-ROS arguments.
- **namespace (Optional[str])** - The namespace to which relative topic and service names will be prefixed. Validated by `validate_namespace()`.
- **use_global_arguments (bool)** - False if the node should ignore process-wide command line args.



ROS2 실습

dependency 확인

```
# cd if you're still in the ``src`` directory with the ``ros_tutorials`` clone
```

- cd ..
- rosdep install -i --from-path src --rosdistro foxy -y

colcon build 실행

- colcon build
-

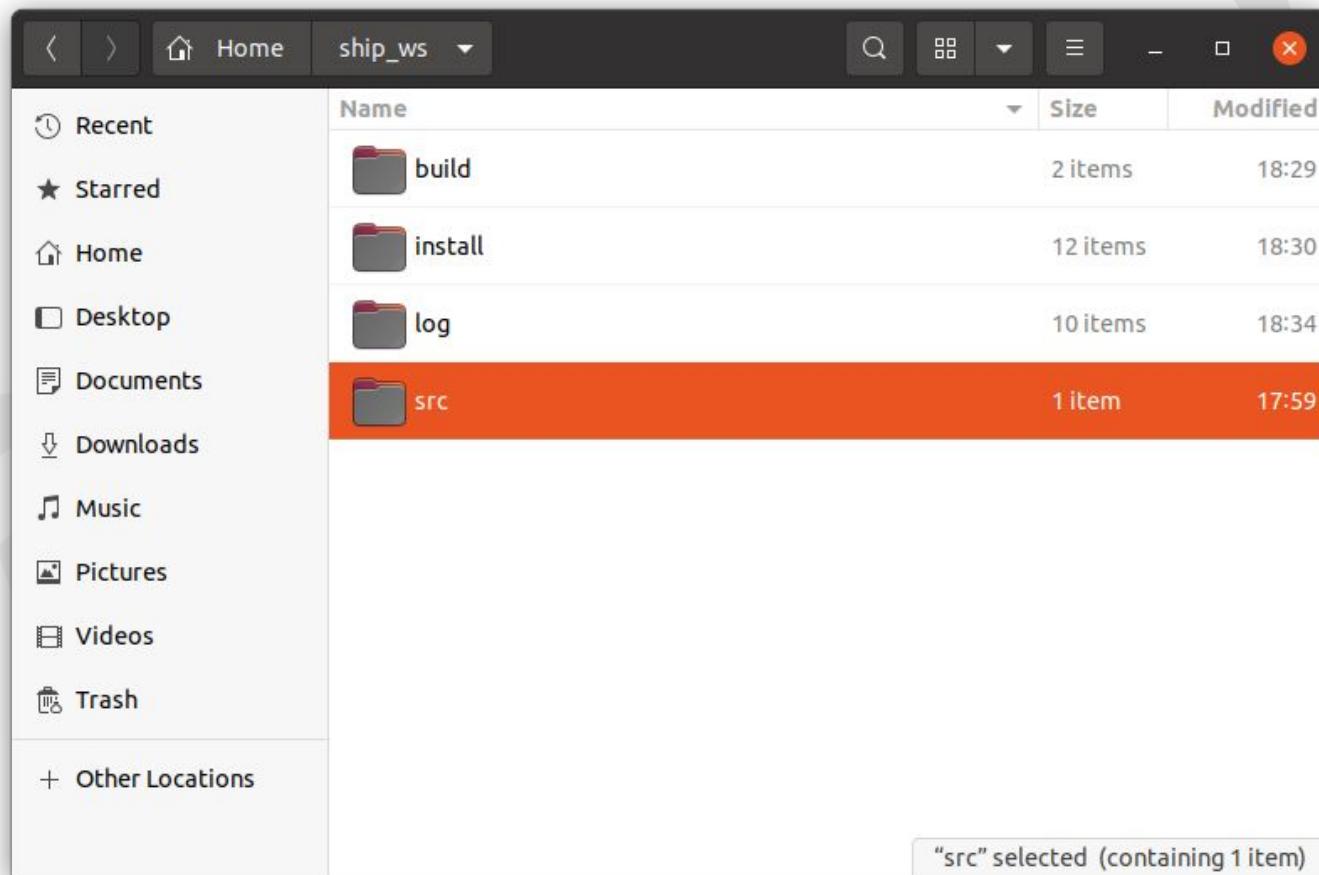
<결과>

- build, install, log 폴더 생성

ROS2 실습

build 결과 확인

- build, install, log 파일 생성





ROS2 실습

package 실행

- . install/local_setup.bash
- ros2 run my_turtlesim_pkg turtle_subscriber

The screenshot shows a terminal window with the following content:

```
(ROS 2 foxy) byb76@rlhp:~/ros2test$ ros2 run my_turtlesim_pkg turtlesim_node
Hi from my_turtlesim_pkg.
(ROS 2 foxy) byb76@rlhp:~/ros2test$ 
```

The terminal window has a dark background and light-colored text. The prompt indicates the user is running ROS 2 on a system named 'foxy' with a user account 'byb76'. The command 'ros2 run my_turtlesim_pkg turtlesim_node' was entered, followed by the output 'Hi from my_turtlesim_pkg.'. The window includes standard Linux terminal icons for tabs, search, and close.

ROS2 실습

turtlesim topic subscriber 노드 생성

파이썬 파일 생성: turtlesim_subscriber.py

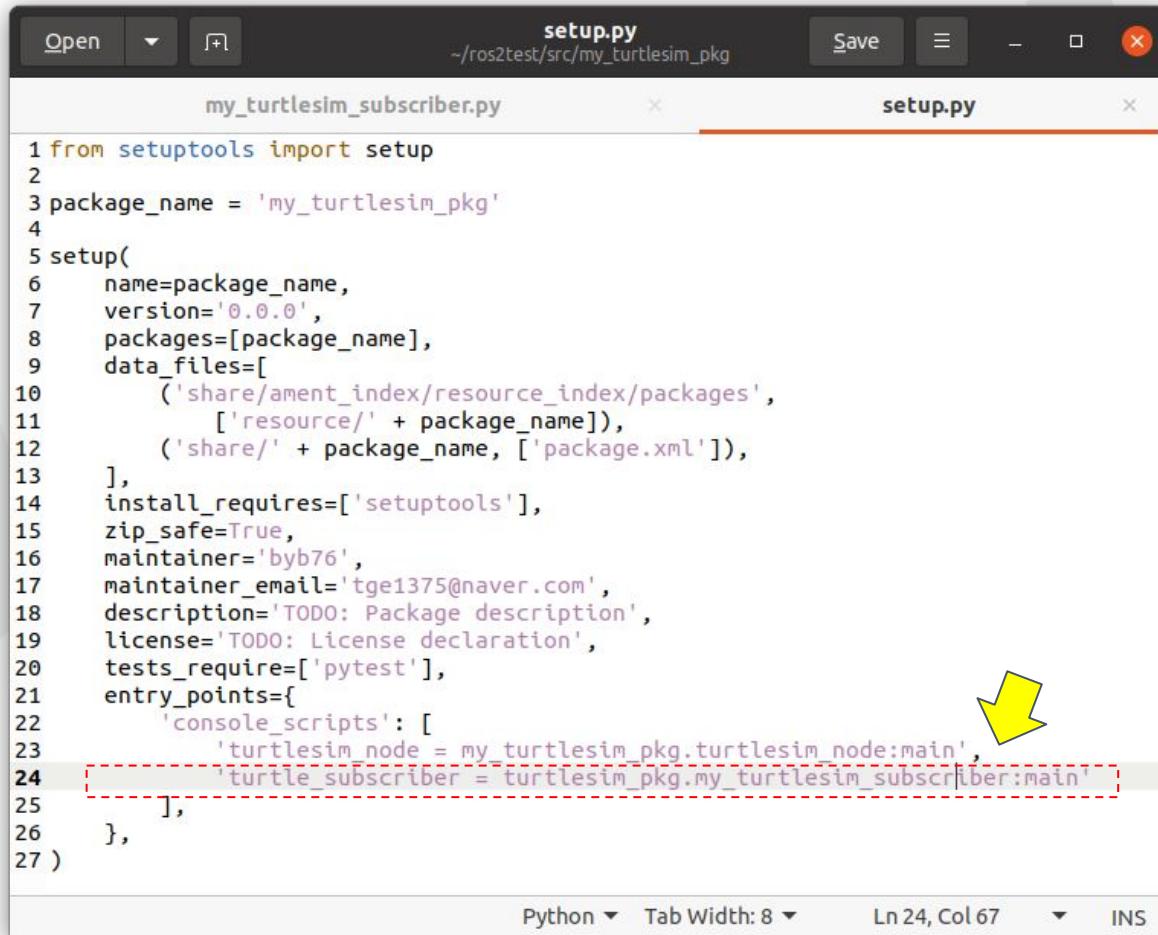
The screenshot shows a Visual Studio Code window with the title bar "turtlesim_subscriber.py - src - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar contains icons for file operations like Open, Save, Find, and Settings. The main editor area displays Python code for a ROS2 node named "turtlesim_subscriber.py". The code uses the rclpy library to create a subscriber for the "/turtle1/pose" topic, printing the received Pose message to the console. A status bar at the bottom indicates "Ln 31, Col 5" and "UTF-8 LF".

```
1 import rclpy as rp
2 from rclpy.node import Node
3 from turtlesim.msg import Pose
4
5
6 class TurtlesimSubscriber(Node):
7     def __init__(self):
8         super().__init__('turtlesim_subscriber')
9         self.subscription = self.create_subscription(Pose, '/turtle1/pose', self.callback, 10)
10
11    def callback(self, msg):
12        # print("X: ", msg.x, "Y: ", msg.y)
13        print("turtle pose X: ", "{:.2f}".format(msg.x),
14              " / Y: ", "{:.2f}".format(msg.y),
15              " Yaw: ", "{:.2f}".format(msg.theta)
16        )
17        # print("turtle angle Yaw: ", "{:.2f}".format(msg.theta))
18
19    def main(args=None):
20        rp.init(args=args)
21
22        turtlesim_subscriber = TurtlesimSubscriber()
23        rp.spin(turtlesim_subscriber)
24
25        turtlesim_subscriber.destroy_node()
26        rp.shutdown()
27
28    if __name__ == '__main__':
29        main()
```

ROS2 실습

turtlesim topic subscriber 노드 생성

setup.py 업데이트: entrypoint 추가 (, 주의)



```
setup.py
my_turtlesim_subscriber.py
setup.py

1 from setuptools import setup
2
3 package_name = 'my_turtlesim_pkg'
4
5 setup(
6     name=package_name,
7     version='0.0.0',
8     packages=[package_name],
9     data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='byb76',
17     maintainer_email='tge1375@naver.com',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'turtlesim_node = my_turtlesim_pkg.turtlesim_node:main',
24             'turtle_subscriber = turtlesim_pkg.my_turtlesim_subscriber:main'
25         ],
26     },
27 )
```

The screenshot shows a code editor with two tabs: "my_turtlesim_subscriber.py" and "setup.py". The "setup.py" tab is active, displaying Python code for setting up a ROS2 package. A yellow arrow points to the "entry_points" section, specifically to the line "entry_points={ 'console_scripts': ['turtlesim_node = my_turtlesim_pkg.turtlesim_node:main', 'turtle_subscriber = turtlesim_pkg.my_turtlesim_subscriber:main'] }". This line defines two command-line scripts: "turtlesim_node" and "turtle_subscriber". The "turtlesim_node" script is associated with the "turtlesim_node" function in the "my_turtlesim_pkg.turtlesim_node" module, and the "turtle_subscriber" script is associated with the "my_turtlesim_subscriber" function in the "turtlesim_pkg.my_turtlesim_subscriber" module.

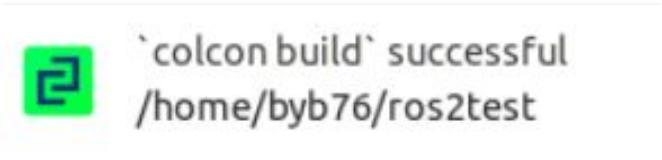


ROS2 실습

package build

```
$ colcon build
```

```
$. install/localsetup.bash
```



```
(ROS 2 foxy) byb76@rlhp:~/ros2test$ colcon build
Starting >>> my_turtlesim_pkg
Finished <<< my_turtlesim_pkg [2.13s]

Summary: 1 package finished [2.56s]
(ROS 2 foxy) byb76@rlhp:~/ros2test$
```



ROS2 실습

package test

```
$ colcon build
```

```
$ . install/localsetup.bash
```

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim draw_square
```

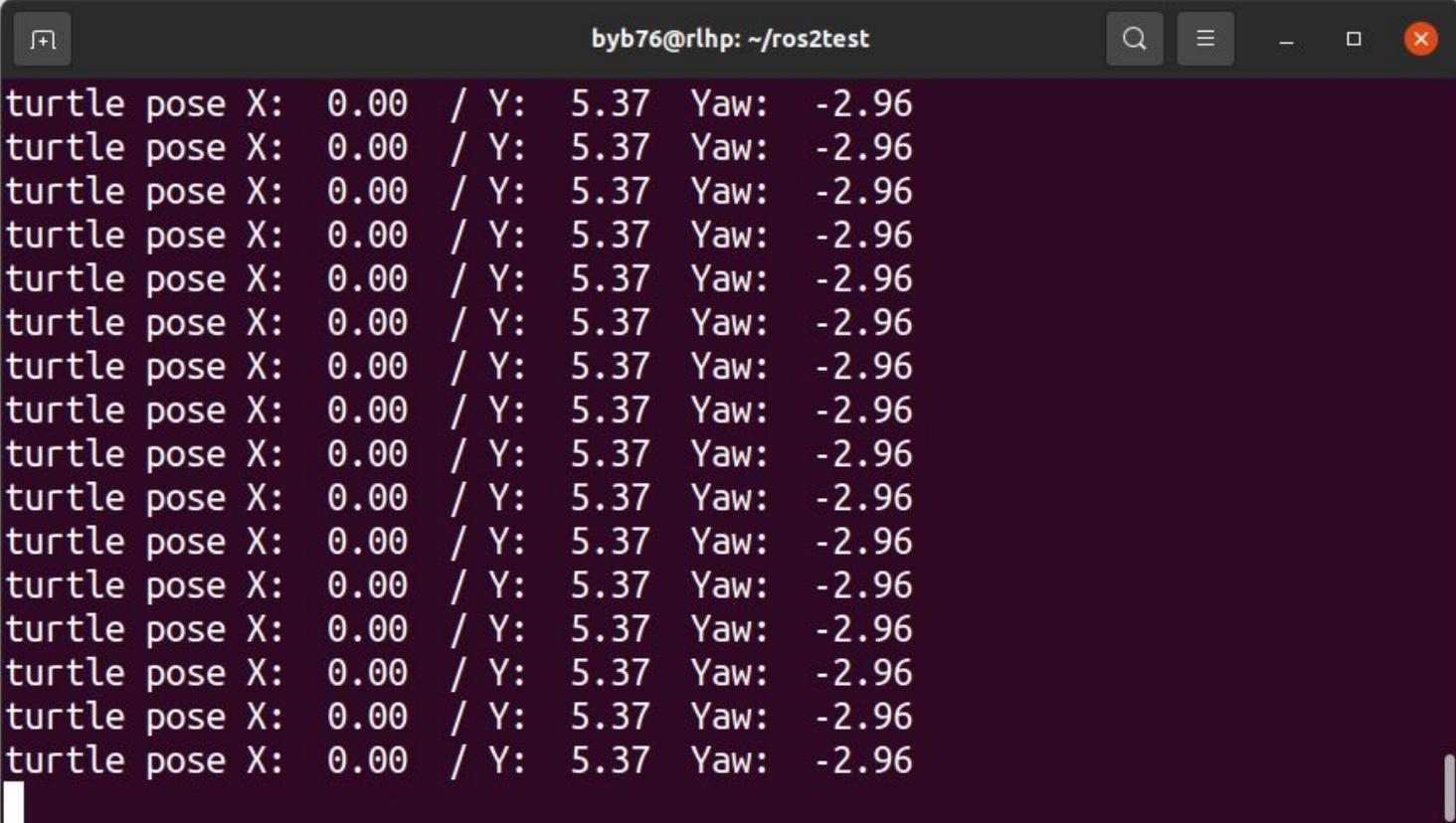


ROS2 실습

package 실행

```
$ros2 run my_turtlesim_pkg turtle_subscriber
```

>위 이름은 setup.py entrypoint 이름과 일치

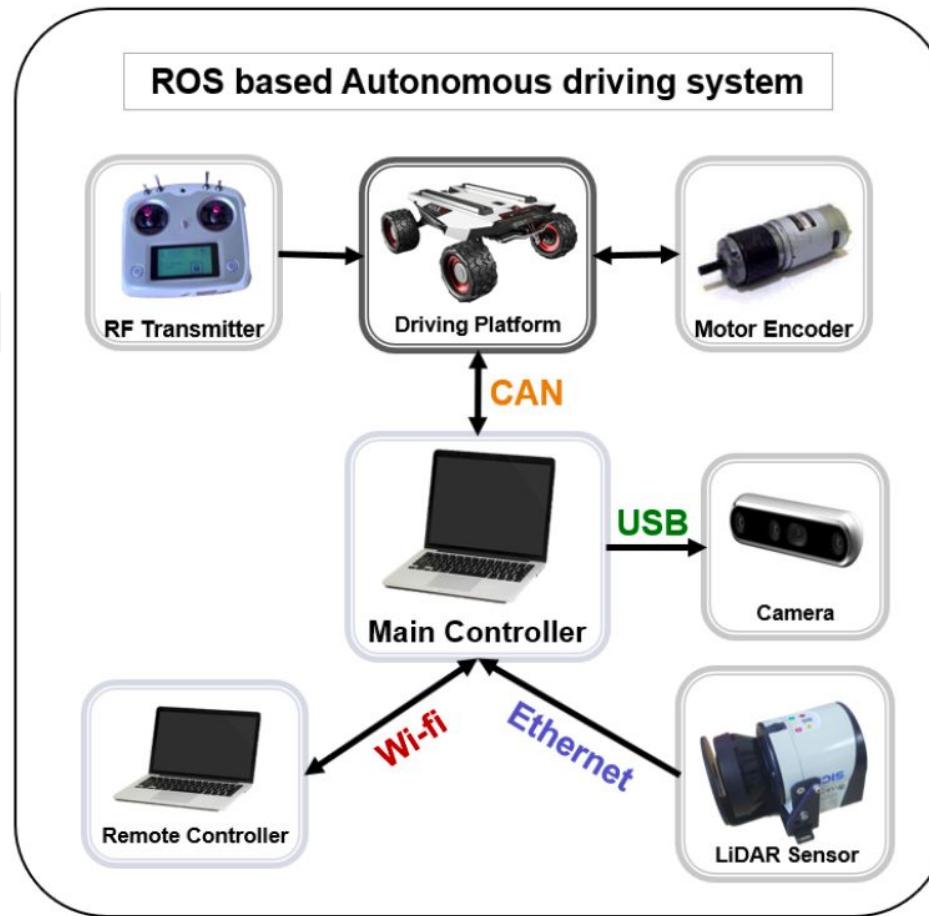


A screenshot of a terminal window titled "byb76@rlhp: ~/ros2test". The window contains a series of identical lines of text, each representing the pose of a turtle in a simulation. The text is as follows:

```
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
```

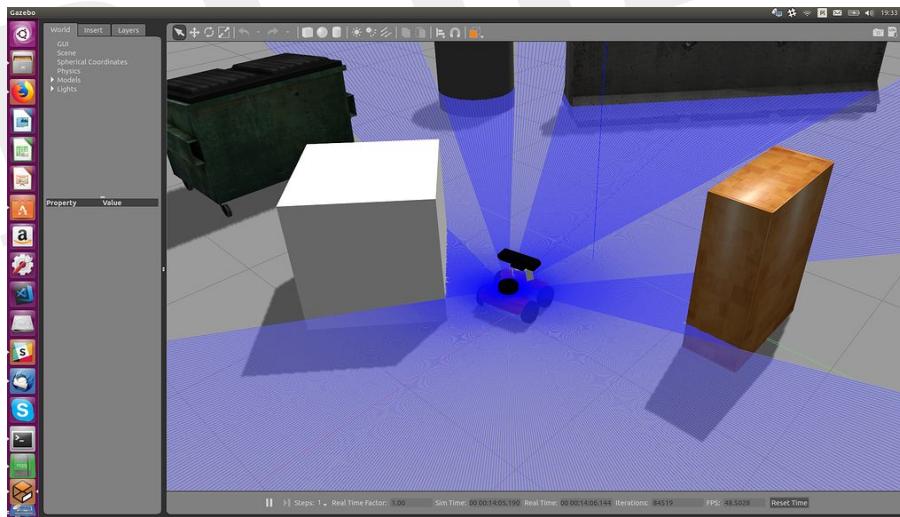
This pattern repeats 15 times. The terminal has a dark background with light-colored text. The title bar includes standard window controls (minimize, maximize, close) and a search icon.

ROS2 laser scan



Lidar sensor

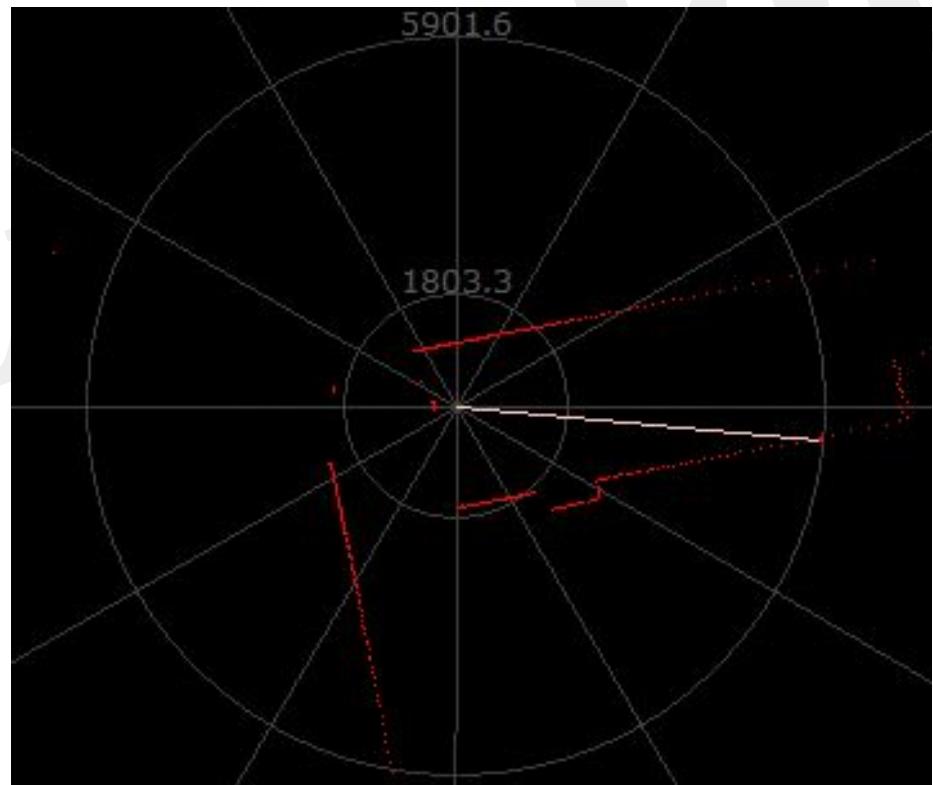
실습



ROS2 HW

Lidar

demo - rplidar a2





ROS2 구성

Service

demo - rplidar a2

```
$ ros2 service call /stop_motor std_srvs/srv/Empty {}
```

```
byb76@rlhp: ~/ros2_ws
byb76@rlhp: ~/ros2_ws 73x19
(ROS 2 foxy) byb76@rlhp:~/ros2_ws$ ros2 service call /stop_motor std_srvs/srv/Empty {}
1673182032.650391 [0]           ros2: using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

(ROS 2 foxy) byb76@rlhp:~/ros2_ws$
```

ROS2 HW



RPLidar

link: https://github.com/Slamtec/rplidar_ros

GitHub - Slamtec/rplidar_

Product Solutions Open Source Pricing

Search Sign in Sign up

Slamtec / rplidar_ros Public

Code Issues 67 Pull requests 20 Actions Projects Security Insights

master 6 branches 6 tags Go to file Code

WubinXia scan frequency configuration support 7b011f1 on Oct 17, 2022 61 commits

launch scan frequency configuration support 2 months ago

rviz First release of RPLIDAR ROS package 8 years ago

scripts update to RPLIDAR SDK 1.5.2 6 years ago

sdk support RPLIDAR S2E 7 months ago

src scan frequency configuration support 2 months ago

CHANGELOG.rst release v2.0.0 last year

CMakeLists.txt release v2.0.0 last year

LICENSE [release] rplidar_ros release 1.7.0 4 years ago

README.md Update README.md 3 months ago

package.xml upgrade sdk 1.10.0 4 years ago

rplidar_A1.png change rplidar_A1.png 6 years ago

rplidar_A2.png add new picture for install and fixed the default param of inverted a... 6 years ago

About No description, website, or topics provided.

Readme BSD-2-Clause license 334 stars 18 watching 399 forks

Releases 6 tags

Packages No packages published

Contributors 11

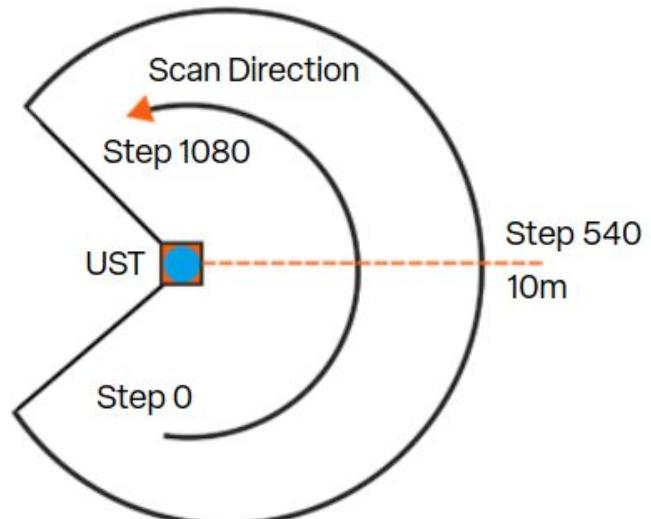
ROS2 HW

dummy_laser

Hukuyo 2D lidar



Measuring Steps 1081
Detection Angle 270°
Angular Resolution 0.25°





ROS2 HW

dummy_laser

Hukuyo 2D lidar



Product Name	Scanning Laser Range Finder
Model	UST-10LX
Supply voltage	12VDC/24VDC (Operation range 10 to 30V ripple within 10%)
Supply current	150mA or less (When using DC24V) (during start up 450mA is necessary.)
Light source	Laser semiconductor (905nm) Laser class 1(IEC60825-1:2007)
Detection range	0.06m to 10m (white Kent sheet) 0.06m to 4m (diffuse reflectance 10%) Max. detection distance : 30m
Accuracy	±40mm (*1)
Repeated accuracy	σ< 30mm (*1)
Scan angle	270°
Scan speed	25ms (Motor speed 2400rpm)
Angular resolution	0.25°
Start up time	Within 10 sec (start up time differs if malfunction is detected during start up)
Input	IP reset input, photo-coupler input (current 4mA at ON)
Output	Synchronous Output, photo coupler open collector output 30VDC 50mA MAX.
Interface	Ethernet 100BASE-TX
LED display	Power supply LED display (Blue): Blinks during start up and malfunction state.
Surrounding intensity	Less than 80,000lx Note : Avoid direct sunlight or other illumination sources as it may cause sensor malfunction
Ambient temperature and humidity	-10°C to +50°C, below 85%RH (without dew, frost)
Storage temperature and humidity	-30°C to +70°C, below 85%RH (without dew, frost)
Vibration resistance	10 to 55Hz double amplitude of 1.5mm for 2hrs in each X, Y, and Z direction 55 to 200Hz 98m / s ² sweep of 2min for 1hr in each X,Y and Z direction
Vibration resistance (Operating)	55 to 150Hz 19.6m / s ² sweep of 2min for 30min in each X,Y and Z direction
Shock resistance	196m/s ² (20G) X,Y and Z direction 10 times.



ROS2 HW

dummy_laser

```
$ ros2 run dummy_sensors dummy_laser
```

A screenshot of a terminal window titled "rlmodel@rlmodel: ~/ros2_ws". The window contains three tabs, all showing the same command line. The command being run is "ros2 run dummy_sensors dummy_laser". The terminal output shows three INFO messages from the "dummy_laser" node. The first message indicates an angle increment of 0.004363. The second message indicates a scan size of 1081. The third message indicates a scan time increment of 0.000028.

```
rlmodel@rlmodel:~/ros2_ws$ ros2 run dummy_sensors dummy_laser
[INFO] [1688216739.713275612] [dummy_laser]: angle inc: 0.004363
[INFO] [1688216739.713710749] [dummy_laser]: scan size: 1081
[INFO] [1688216739.713806255] [dummy_laser]: scan time increment: 0.000028
```

ROS2 HW



dummy_laser

>new python file - publisher

ROS2_basic/py_topic_tutorial/py_topic_tutorial/tp_laserscan_sub.py

File Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITORS

- tp_laserscan_sub.py py_topic_tutorial/py_t...
- setup.py py_topic_tutorial

ROBOTS

py_topic_tutorial

- > add_interface
- > py_action_tutorial
- > py_node_tutorial
- > py_param_tutorial
- > py_service_tutorial
- > py_streampubsub
- > .vscode
- > py_topic_tutorial
- > __pycache__
- tp_laserscan_sub.py
- topic_example_1_publisher.py
- topic_example_2_subscriber.py
- topic_example_3_pub_and_sub.py
- tp_twistcmd_pub.py
- > resource
- > test
- package.xml
- setup.cfg
- setup.py

OUTLINE

TIMELINE

tp_laserscan_sub.py x setup.py

py_topic_tutorial > tp_laserscan_sub.py > ScanSubNode > __init__

```
51 def __init__(self):
52     """Node Initialization.
53
54     You must type name of the node in inherited initializer.
55     """
56
57     super().__init__('scan_sub_node')
58
59     queue_size = 10 # Queue Size
60     # You can create subscriber with create_subscription function
61     # this function get those params
62     #
63     # msg type, topic name, callback function, queue_size
64     #
65     # topic name must exists and coincident with exact topic name
66     self.pose_subscriber = self.create_subscription(
67         LaserScan, 'scan', self.sub_callback, my_profile
68     )
69
70     def sub_callback(self, msg):
71         """Timer will run this function periodically."""
72         self.get_logger().info(f' \
73             \nmsg.ranges[0] : {round(msg.ranges[0], 2)} \
74             \nmsg.ranges[30] : {round(msg.ranges[30], 2)} \
75             \nmsg.ranges[60] : {round(msg.ranges[60], 2)} \
76             \nmsg.ranges[90] : {round(msg.ranges[90], 2)} \
77             \nmsg.ranges[119] : {round(msg.ranges[119], 2)} \
78         ')
79         # self.get_logger().info(f'''x : {msg.x:.3f} / y : {msg.y:.3f} / \
80         # linear_velocity : {msg.linear_velocity} / angular_velocity : {msg.angular_velocity}'''
```

Ln 53, Col 32 Spaces: 4 UTF-8 CRLF Python 3.8.10 64-bit



ROS2 HW

LaserScan msg

>link:http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/LaserScan.html

sensor_msgs/LaserScan Message

File: `sensor_msgs/LaserScan.msg`

Raw Message Definition

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header          # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        #
                        # in frame frame_id, angles are measured around
                        # the positive Z_axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min      # start angle of the scan [rad]
float32 angle_max      # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment  # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points
float32 scan_time       # time between scans [seconds]

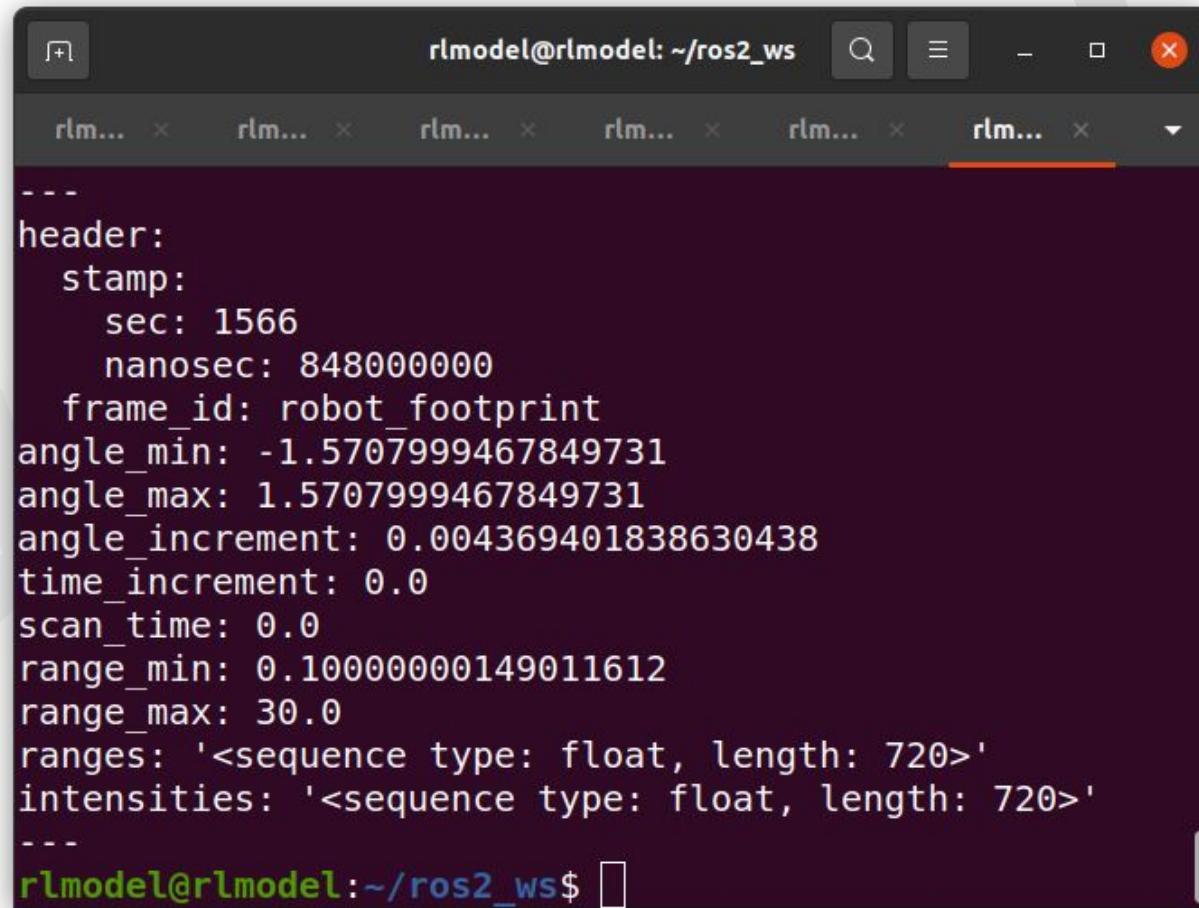
float32 range_min       # minimum range value [m]
float32 range_max       # maximum range value [m]

float32[] ranges        # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities   # intensity data [device-specific units]. If your
                        # device does not provide intensities, please leave
                        # the array empty.
```

ROS2 HW

/diffbot/scan

```
$ ros2 topic echo /scan --no-arr
```

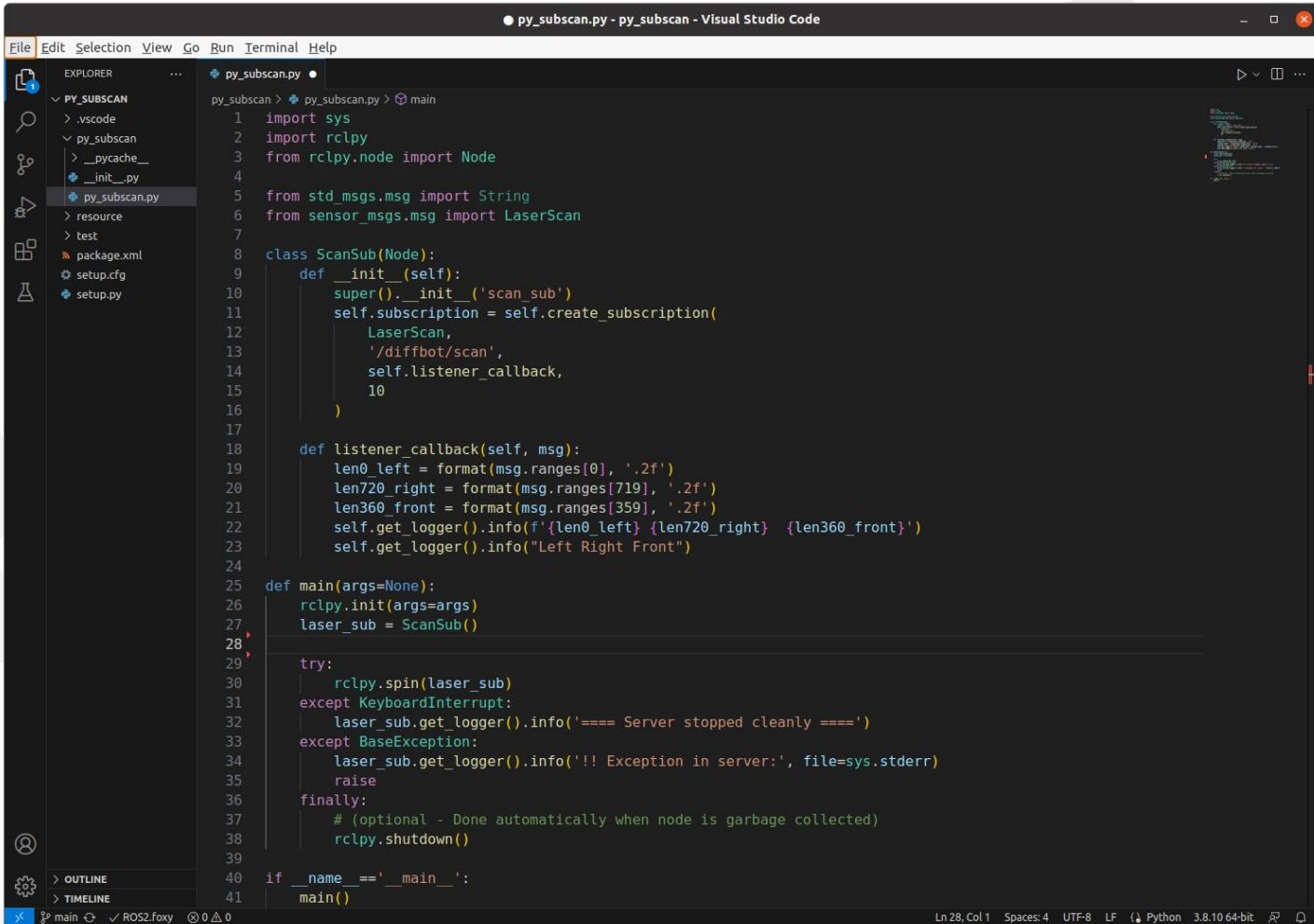
A terminal window titled "rlmodel@rlmodel: ~/ros2_ws" showing the output of the command "ros2 topic echo /scan --no-arr". The output is a ROS message definition for a "Scan" message, including fields for header, stamp, frame_id, angle_min, angle_max, angle_increment, time_increment, scan_time, range_min, range_max, ranges, and intensities.

```
rlmodel@rlmodel: ~/ros2_ws$ ros2 topic echo /scan --no-arr
---
header:
  stamp:
    sec: 1566
    nanosec: 848000000
    frame_id: robot_footprint
angle_min: -1.5707999467849731
angle_max: 1.5707999467849731
angle_increment: 0.004369401838630438
time_increment: 0.0
scan_time: 0.0
range_min: 0.10000000149011612
range_max: 30.0
ranges: '<sequence type: float, length: 720>'
intensities: '<sequence type: float, length: 720>'
---
rlmodel@rlmodel:~/ros2_ws$
```

ROS2 HW

/diffbot/scan

create pkg



The screenshot shows a Visual Studio Code window with the title "py_subscan.py - py_subscan - Visual Studio Code". The code editor displays Python code for a ROS2 node named "py_subscan". The code defines a class `ScanSub` that inherits from `Node`. It subscribes to a topic named `/diffbot/scan` and defines a callback function `listener_callback` that prints the first three ranges of the received LaserScan message. The main function initializes the node and starts the spin loop. The code uses ROS2 standard message types like `String` and `LaserScan` from the `std_msgs` and `sensor_msgs` packages respectively.

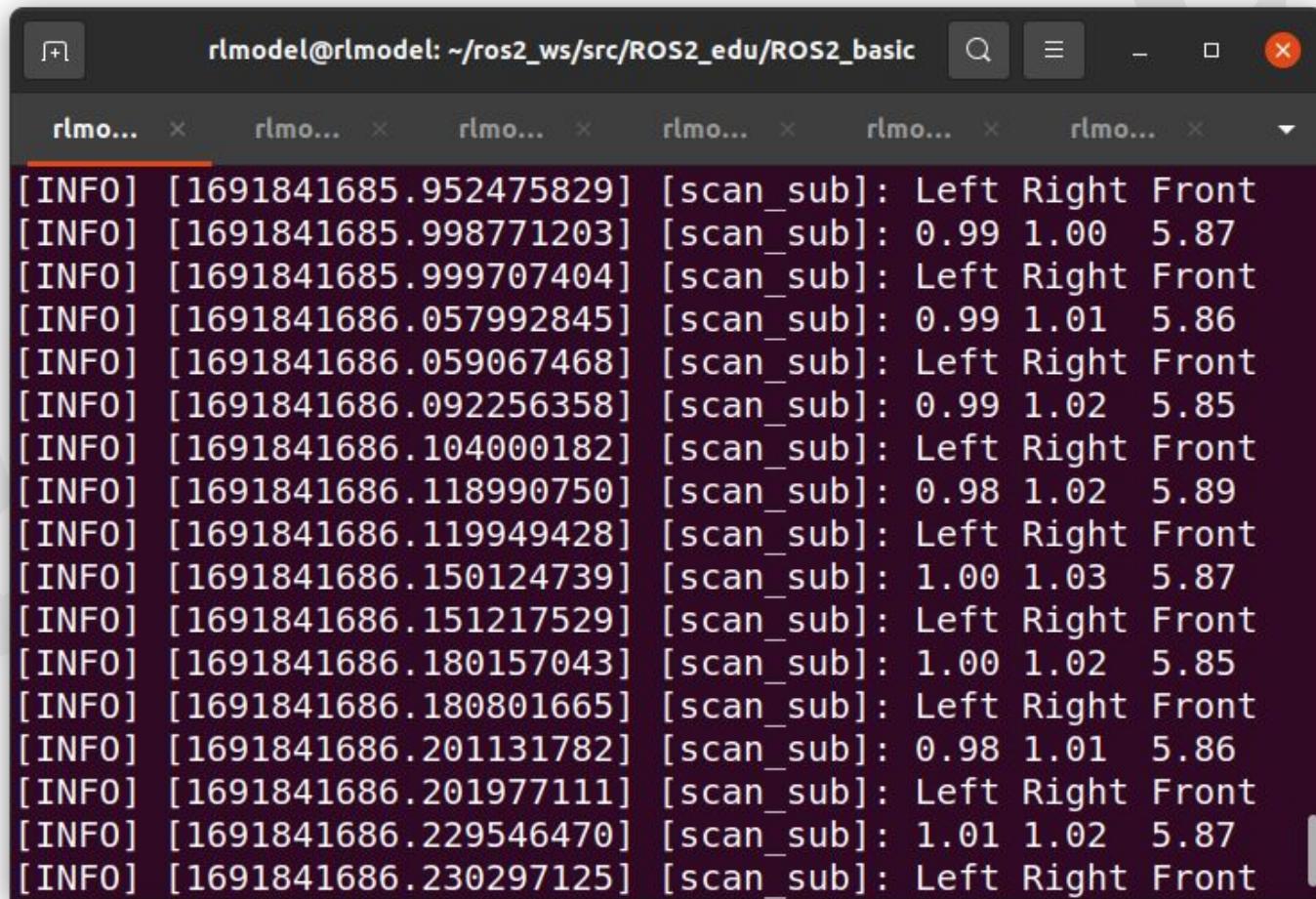
```
File Edit Selection View Go Run Terminal Help
EXPLORER PY_SUBSCAN .vscode py_subscan __init__.py py_subscan.py resource test package.xml setup.cfg setup.py
● py_subscan.py •
py_subscan > py_subscan.py > main
1 import sys
2 import rclpy
3 from rclpy.node import Node
4
5 from std_msgs.msg import String
6 from sensor_msgs.msg import LaserScan
7
8 class ScanSub(Node):
9     def __init__(self):
10         super().__init__('scan_sub')
11         self.subscription = self.create_subscription(
12             LaserScan,
13             '/diffbot/scan',
14             self.listener_callback,
15             10
16         )
17
18     def listener_callback(self, msg):
19         len0_left = format(msg.ranges[0], '.2f')
20         len720_right = format(msg.ranges[719], '.2f')
21         len360_front = format(msg.ranges[359], '.2f')
22         self.get_logger().info(f'{len0_left} {len720_right} {len360_front}')
23         self.get_logger().info("Left Right Front")
24
25 def main(args=None):
26     rclpy.init(args=args)
27     laser_sub = ScanSub()
28
29     try:
30         rclpy.spin(laser_sub)
31     except KeyboardInterrupt:
32         laser_sub.get_logger().info('==== Server stopped cleanly ====')
33     except BaseException:
34         laser_sub.get_logger().info('!! Exception in server:', file=sys.stderr)
35         raise
36     finally:
37         # (optional - Done automatically when node is garbage collected)
38         rclpy.shutdown()
39
40 if __name__ == '__main__':
41     main()
```

Ln 28, Col 1 Spaces: 4 UTF-8 LF ⓘ Python 3.8.10 64-bit ⓘ ⓘ

ROS2 HW

/diffbot/scan

create pkg / compile build / run



The screenshot shows a terminal window with multiple tabs, all of which are currently active and displaying the same log output. The title bar of the terminal window reads "rlmodel@rlmodel: ~/ros2_ws/src/ROS2_edu/ROS2_basic". The log output consists of 20 [INFO] messages, each timestamped with a date and time (e.g., [1691841685.952475829]) and containing the message "[scan_sub]: Left Right Front" followed by three numerical values (e.g., 0.99 1.00 5.87). The terminal window has a dark background and light-colored text.

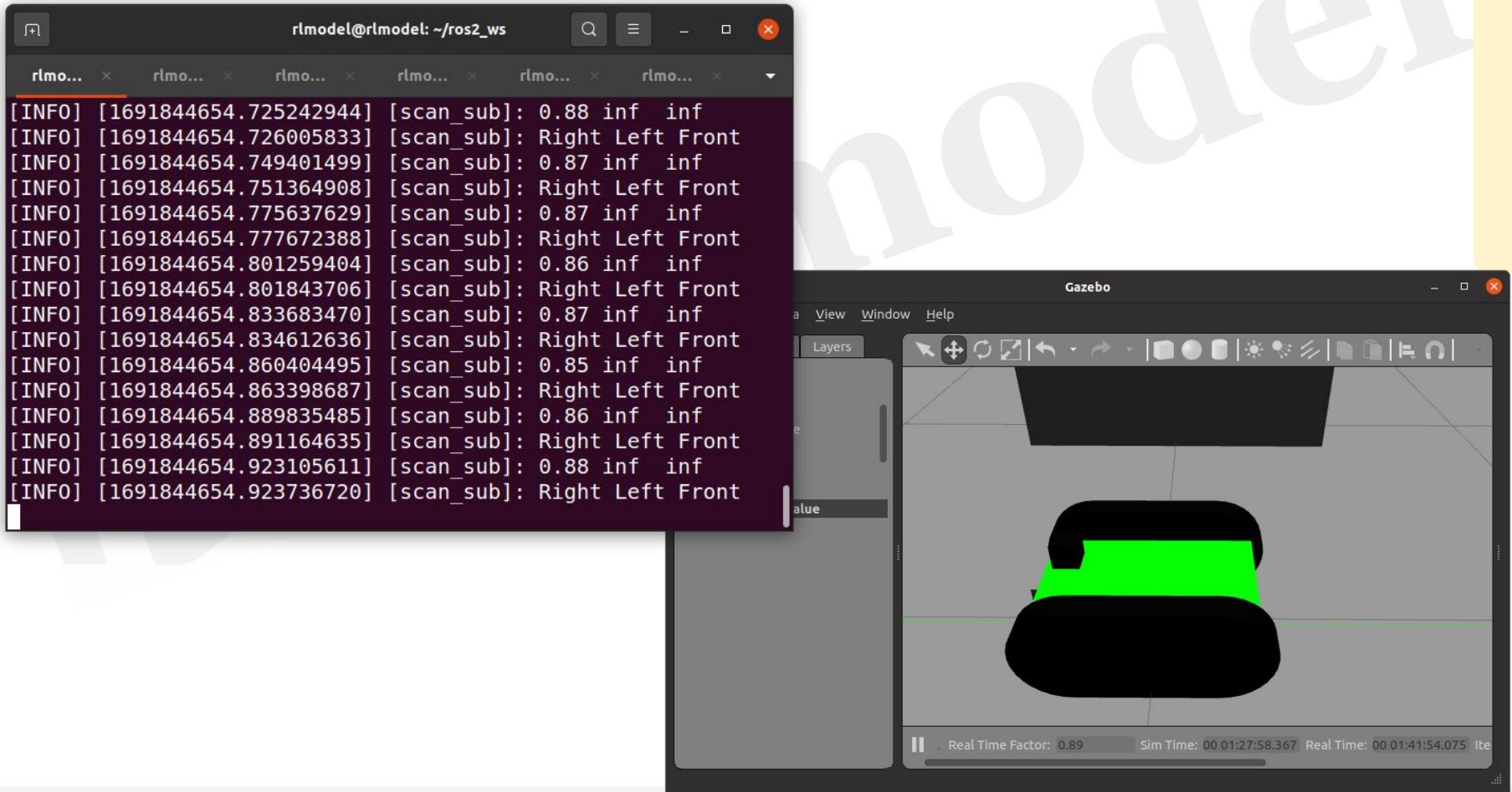
```
[INFO] [1691841685.952475829] [scan_sub]: Left Right Front
[INFO] [1691841685.998771203] [scan_sub]: 0.99 1.00 5.87
[INFO] [1691841685.999707404] [scan_sub]: Left Right Front
[INFO] [1691841686.057992845] [scan_sub]: 0.99 1.01 5.86
[INFO] [1691841686.059067468] [scan_sub]: Left Right Front
[INFO] [1691841686.092256358] [scan_sub]: 0.99 1.02 5.85
[INFO] [1691841686.104000182] [scan_sub]: Left Right Front
[INFO] [1691841686.118990750] [scan_sub]: 0.98 1.02 5.89
[INFO] [1691841686.119949428] [scan_sub]: Left Right Front
[INFO] [1691841686.150124739] [scan_sub]: 1.00 1.03 5.87
[INFO] [1691841686.151217529] [scan_sub]: Left Right Front
[INFO] [1691841686.180157043] [scan_sub]: 1.00 1.02 5.85
[INFO] [1691841686.180801665] [scan_sub]: Left Right Front
[INFO] [1691841686.201131782] [scan_sub]: 0.98 1.01 5.86
[INFO] [1691841686.201977111] [scan_sub]: Left Right Front
[INFO] [1691841686.229546470] [scan_sub]: 1.01 1.02 5.87
[INFO] [1691841686.230297125] [scan_sub]: Left Right Front
```



ROS2 HW

/diffbot/scan

create pkg -> compile build -> run -> check scan data

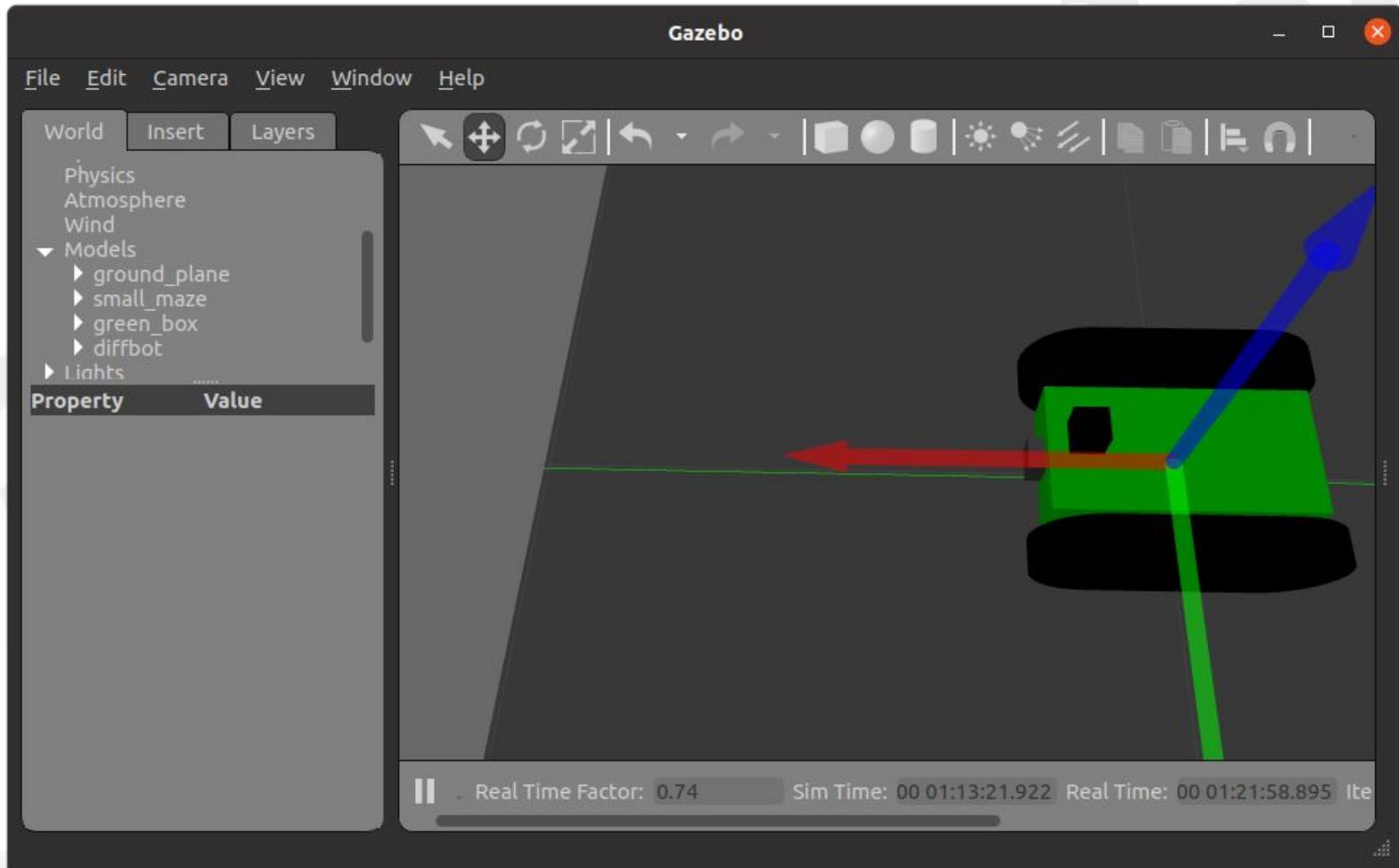




ROS2 HW

/diffbot/scan

challenge: parking.





Camera & Vision

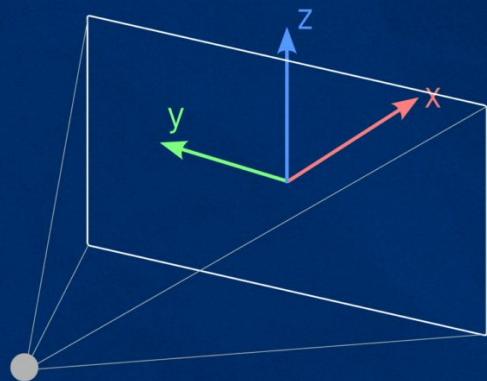


ROS2 HW

CAMERA

demo - webcam mage

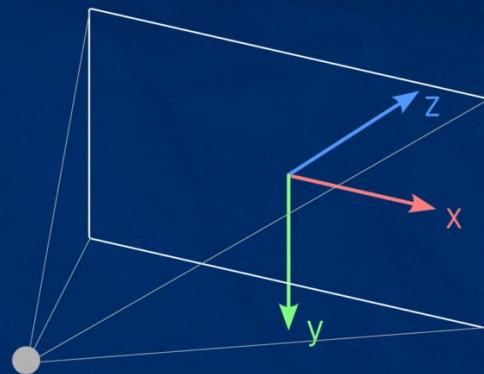
ROS Body Standard



`camera_link`

See ROS Rep 103 for details...

Image Standard



`camera_link_optical`

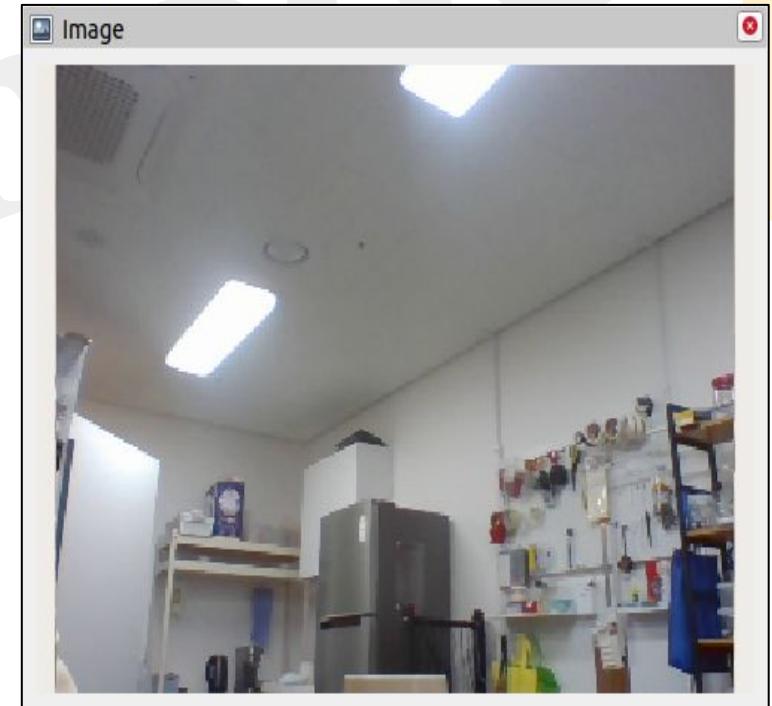
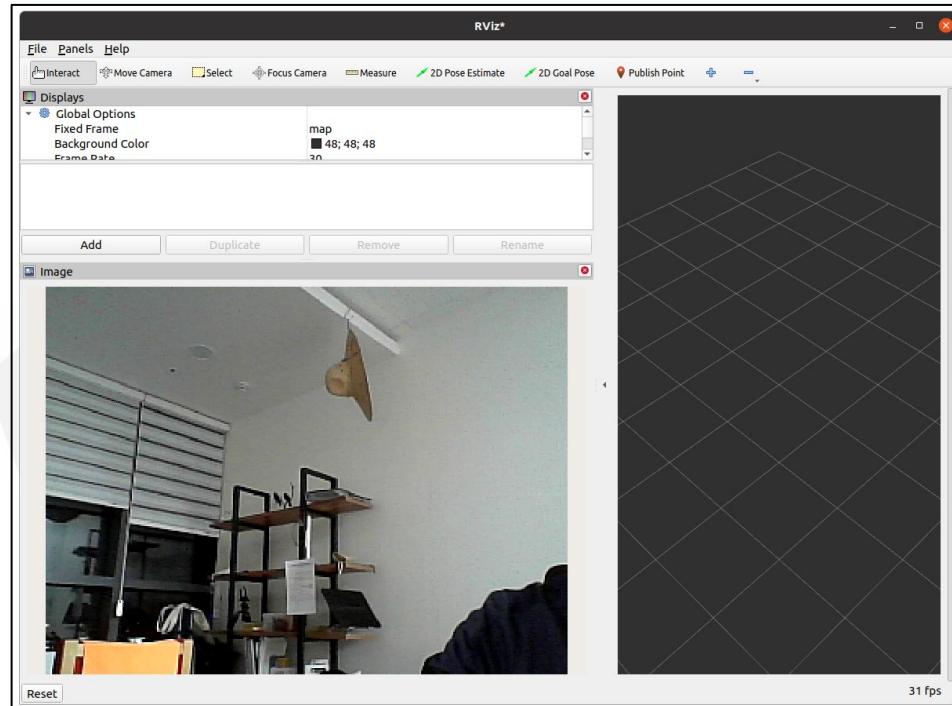
Use as TF frame for all Images etc.

ROS2 HW

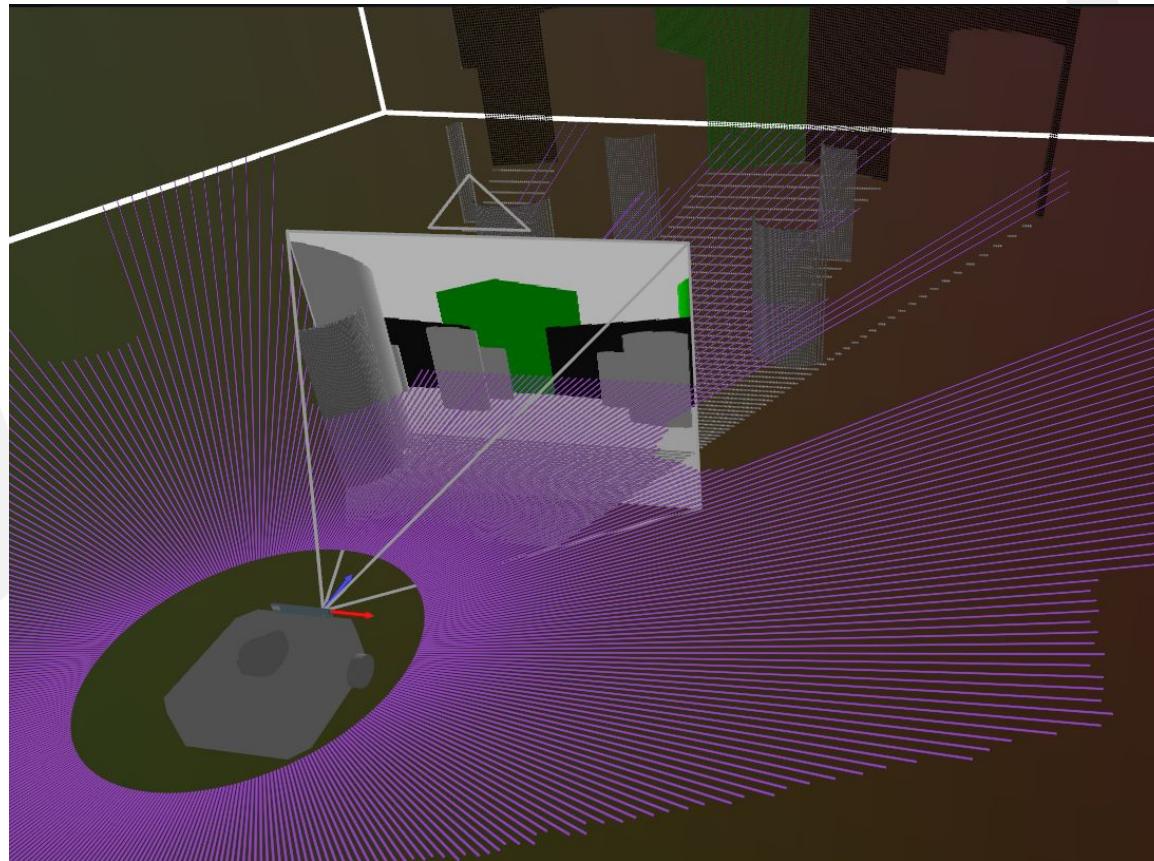
CAMERA

명령어: \$ros2 run image_tools cam2image

확인: rviz2 -> add -> By topic -> /image/Image



cv_bridge

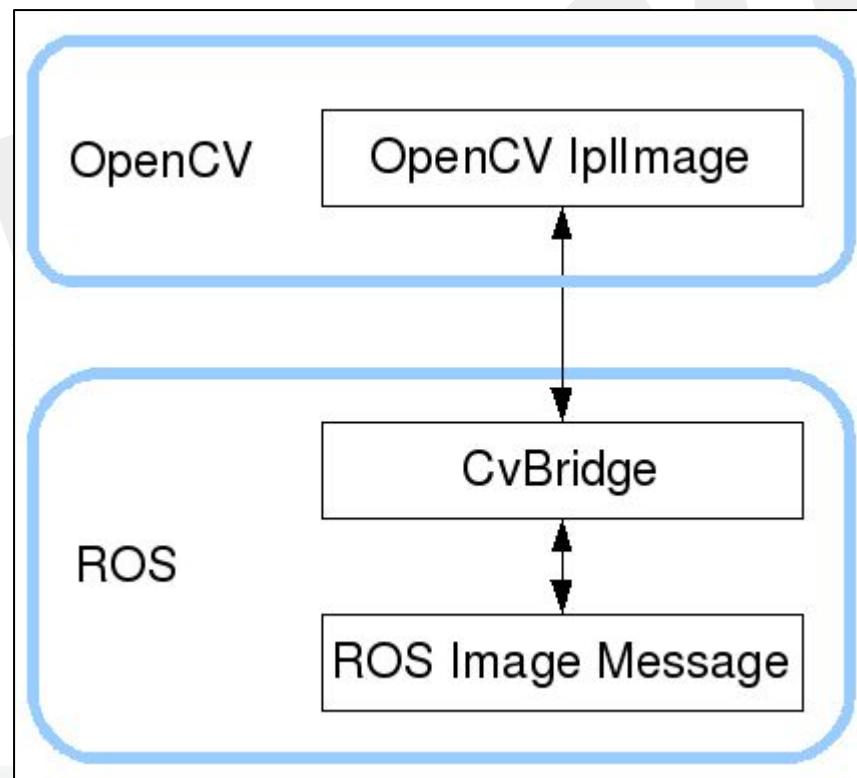


cv_bridge

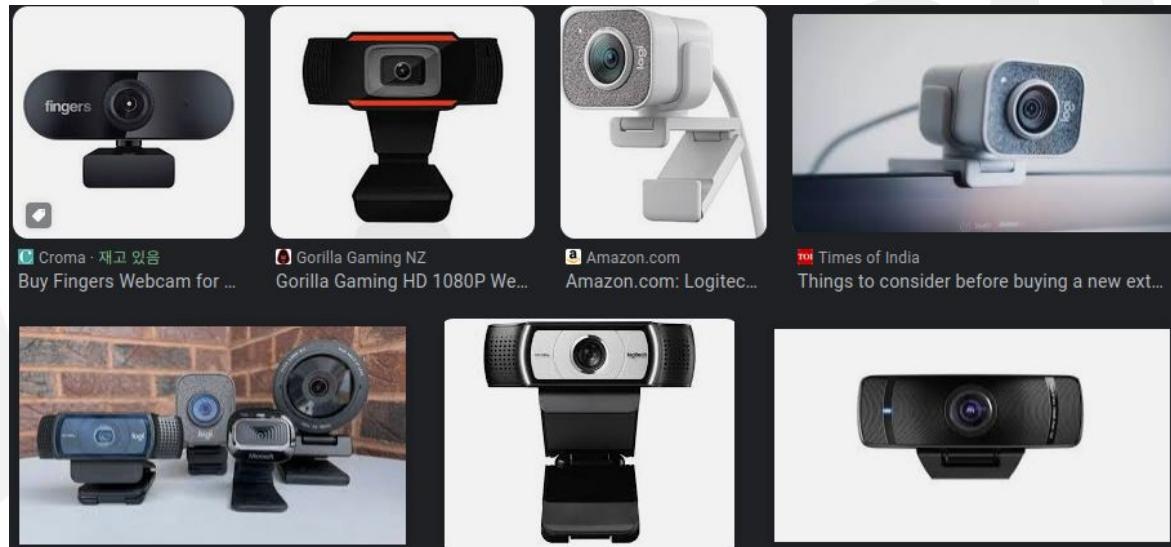
기본설명

cv_bridge converts between ROS 2 image messages and OpenCV image representation for perception applications. As follows:

https://index.ros.org/p/cv_bridge/



Camera test





image_tools

image_tools package

HW확인: \$ lsusb

ref: genicam (Appendix)



```
ybbaek@ubuntu:~/ros2_ws$ lsusb
Bus 001 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 013: ID 0bda:58fe Realtek Semiconductor Corp. Integrated_Webcam
HD
Bus 001 Device 011: ID 2109:8817 VIA Labs, Inc. USB Billboard Device
Bus 001 Device 010: ID 2109:8824 VIA Labs, Inc. USB Billboard Device
Bus 001 Device 009: ID 0bda:9210 Realtek Semiconductor Corp. RTL9210B-CG
Bus 001 Device 015: ID 1902:8301 Generic HD camera
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 008: ID 08bb:2707 Texas Instruments PCM2707 stereo audio DAC
Bus 002 Device 007: ID 0e0f:0008 VMware, Inc. VMware Virtual USB Mouse
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
ybbaek@ubuntu:~/ros2_ws$
```



image_tools

image_tools package

영상장치 확인: \$ ls /dev

>video0, video1 항목 확인

```
ybbaek@ubuntu: ~/ros2_ws$ ls /dev
full          null          tty19         tty53         ttyS29       vga_arbiter
fuse          nvram         tty2          tty54         ttyS3        vhci
hidraw0       port          tty20         tty55         ttyS30       vhost-net
hidraw1       ppp           tty21         tty56         ttyS31       vhost-vsock
hpet          psaux         tty22         tty57         ttyS4        video0
hugepages     ptmx          tty23         tty58         ttyS5        video1
hwrng         pts           tty24         tty59         ttyS6        video2
initctl       random        tty25         tty6          ttyS7        video3
input          rfkill        tty26         tty60         ttyS8        video4
kmsg          rtc           tty27         tty61         ttyS9        video5
log           rtc0          tty28         tty62         udmabuf      vmci
loop0         sda           tty29         tty63         uhid         vsock
loop1         sda1          tty3          tty7          uinput       zero
loop10        sda2          tty30         tty8          urandom      zfs
loop11        sda5          tty31         tty9          userio
loop12        sdb           tty32         ttyprintk    v4l
loop13        sg0           tty33         ttyS0         vcs
```



image_tools

image_tools package

```
$ ros2 run image_tools cam2image  
$ ros2 run image_tools cam2image video0:=video2  
$ ros2 run image_tools cam2image --ros-args --remap video0:=video1
```

The image shows a screenshot of an Ubuntu desktop environment. On the left, there's a dock with icons for a browser, a file manager, and other applications. In the center, a terminal window is open with the command `ros2 run image_tools cam2image` entered. The terminal output shows the process of publishing images from a camera interface. The background of the desktop is a red gradient.

```
(ROS 2 foxy) byb76@rlhp:~/rlmodel_ws$ ros2 run image_tools cam2image  
1687091429.495349 [0] cam2image: using network interface wlp2s0 (udp/192  
.168.0.19) selected arbitrarily from: wlp2s0, docker0  
[ WARN:0] global ..../modules/videoio/src/cap_gstremer.cpp (935) open Open  
CV | GStreamer warning: Cannot query video position: status=0, value=-1,  
duration=-1  
[INFO] [1687091430.567417637] [cam2image]: Publishing image #1  
[INFO] [1687091430.599342195] [cam2image]: Publishing image #2  
[INFO] [1687091430.632319996] [cam2image]: Publishing image #3  
[INFO] [1687091430.665406763] [cam2image]: Publishing image #4  
[INFO] [1687091430.698305136] [cam2image]: Publishing image #5  
[INFO] [1687091430.731358797] [cam2image]: Publishing image #6  
[INFO] [1687091430.764139087] [cam2image]: Publishing image #7  
[INFO] [1687091430.797168119] [cam2image]: Publishing image #8  
[INFO] [1687091430.830250826] [cam2image]: Publishing image #9  
[INFO] [1687091430.863534147] [cam2image]: Publishing image #10  
[INFO] [1687091430.896348395] [cam2image]: Publishing image #11  
[INFO] [1687091430.929422974] [cam2image]: Publishing image #12  
[INFO] [1687091430.962383051] [cam2image]: Publishing image #13  
[INFO] [1687091430.995449920] [cam2image]: Publishing image #14
```

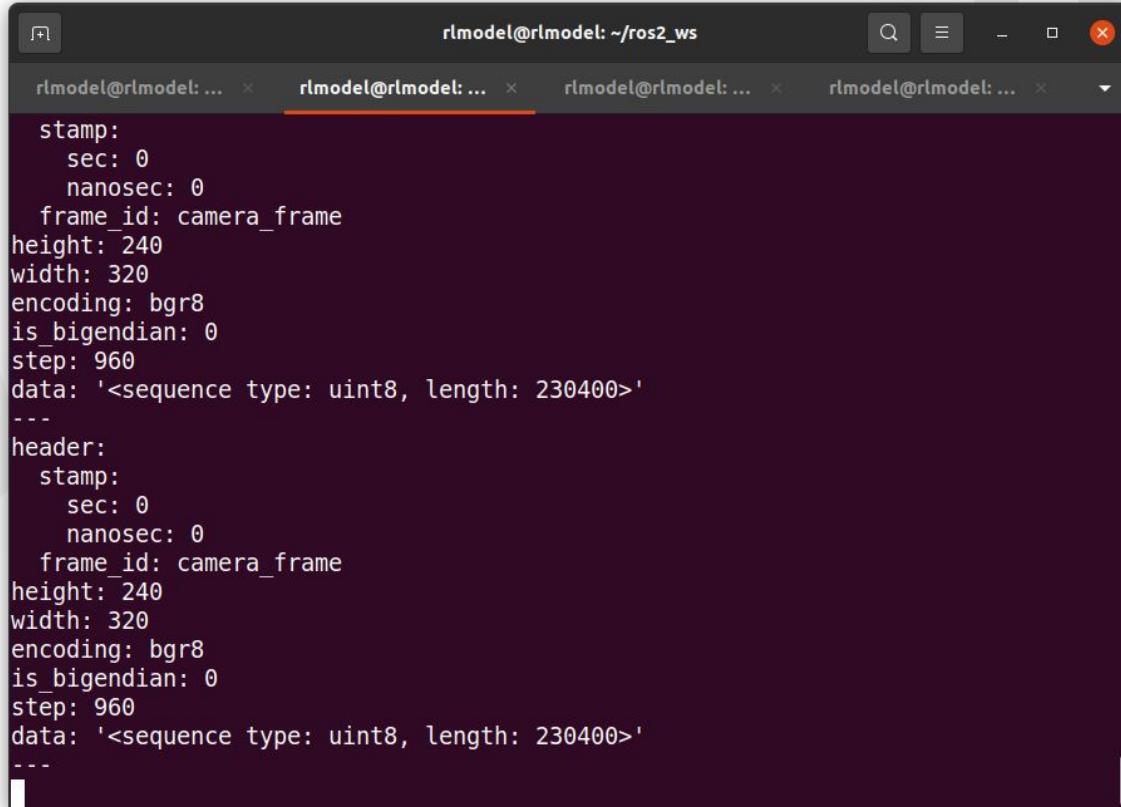
image_tools

image_tools package

image topic check:

```
$ ros2 run image_tools cam2image
```

```
$ ros2 topic echo --no-arr /image_raw
```



The screenshot shows a terminal window with four tabs, with the fourth tab active. The terminal output displays two messages received on the `/image_raw` topic. Each message is a ROS2 message structure for an image, containing fields such as `stamp`, `height`, `width`, `encoding`, and `data`.

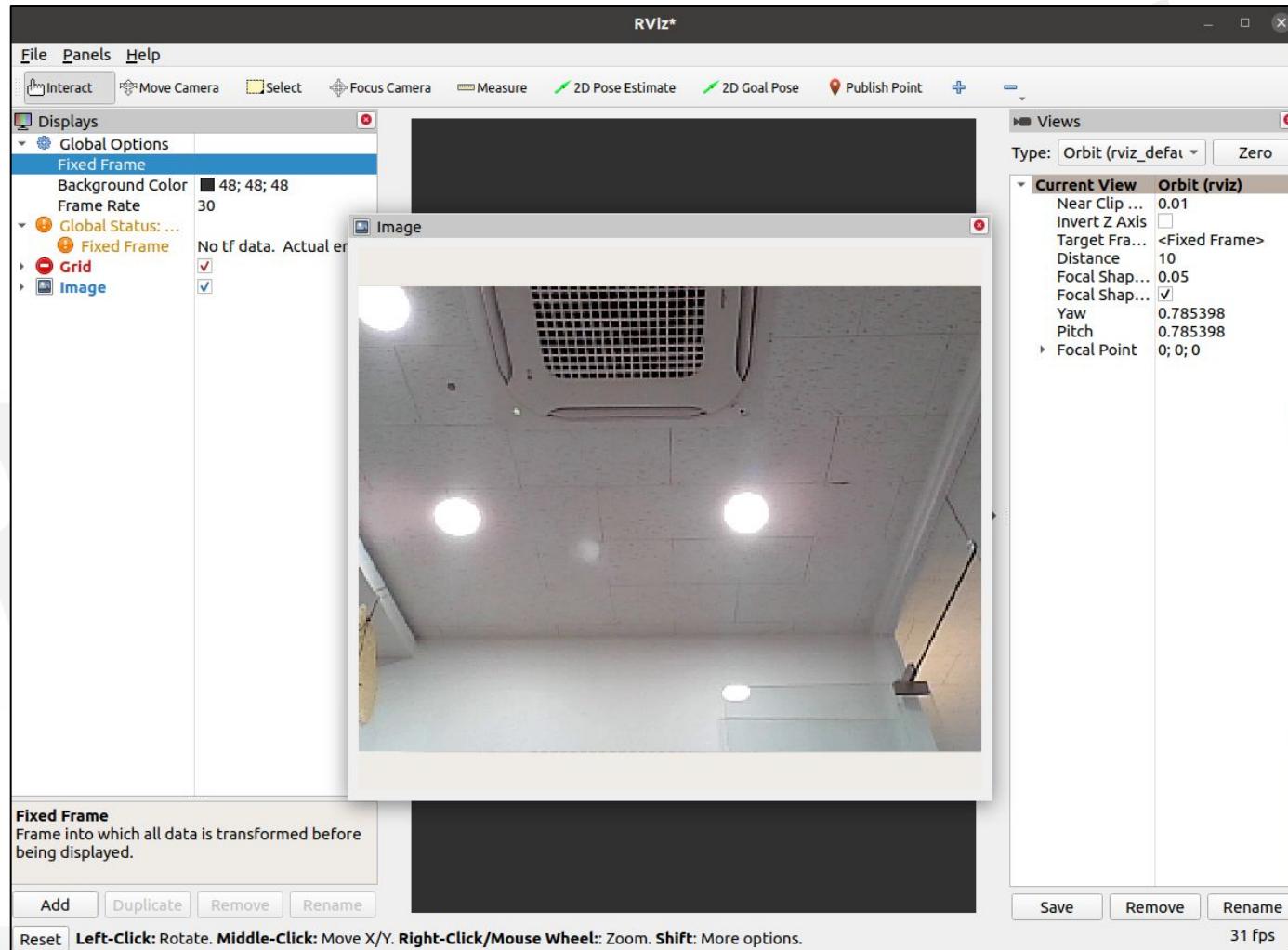
```
rlmodel@rlmodel: ~/ros2_ws
```

```
stamp:
  sec: 0
  nanosec: 0
frame_id: camera_frame
height: 240
width: 320
encoding: bgr8
is_bigendian: 0
step: 960
data: '<sequence type: uint8, length: 230400>'
---
header:
  stamp:
    sec: 0
    nanosec: 0
  frame_id: camera_frame
height: 240
width: 320
encoding: bgr8
is_bigendian: 0
step: 960
data: '<sequence type: uint8, length: 230400>'
```

image_tools

image_tools package

확인: rviz2





image_tools

image_tools package

에러 케이스

```
ybbaek@ubuntu:~/ros2_ws$ ros2 run image_tools cam2image
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (1758) handleMessage
OpenCV | GStreamer warning: Embedded video playback halted; module v4l2src0 r
eported: Could not read from resource.
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (888) open OpenCV |
GStreamer warning: unable to start pipeline
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (480) isPipelinePlay
ing OpenCV | GStreamer warning: GStreamer: pipeline have not been created
[ WARN:0] global ../modules/videoio/src/cap_v4l.cpp (887) open VIDEOIO(V4L2:/d
ev/video0): can't open camera by index
[ERROR] [1687090730.884620438] [cam2image]: Could not open video stream
terminate called after throwing an instance of 'std::runtime_error'
  what(): Could not open video stream
ybbaek@ubuntu:~/ros2_ws$
```



image_tools

image_tools package

```
$ ls /dev | grep video  
$ ls /dev/video*
```

The screenshot shows a terminal window with four tabs, all labeled "rlmodel@rlmodel...". The active tab is highlighted with a red underline. The terminal displays the following command-line session:

```
rlmodel@rlmodel:~/rlmodel$ ls /dev | grep video
video0
video1
rlmodel@rlmodel:~/rlmodel$ ls /dev/video*
/dev/video0  /dev/video1
rlmodel@rlmodel:~/rlmodel$
```



usb_cam package

usb_cam (ros2 driver)

Link: https://github.com/ros-drivers/usb_cam

The screenshot shows the GitHub repository page for the `usb_cam` package. The repository is public and has 5 branches and 27 tags. The main table lists recent commits from `twdragon`, including updates to `AUTHORS.md`, `.github/workflows`, `config`, `include/usb_cam`, `launch`, `src`, `.gitignore`, `.travis.yml`, `AUTHORS.md`, `CHANGELOG.rst`, `CMakeLists.txt`, `LICENSE`, and `README.md`. The commits range from 9 years ago to May 26, 2024. The right sidebar provides an **About** section with details like the ROS Driver for V4L USB Cameras, a link to the wiki, and repository statistics (398 stars, 30 watching, 539 forks). It also includes sections for **Releases** and **Report repository**.

File / Commit	Description	Time Ago
<code>twdragon Update AUTHORS.md</code>	7b480fd on May 26	303 commits
<code>.github/workflows</code>	change workflows name	4 months ago
<code>config</code>	Dynamic generation of ROS parameters	3 months ago
<code>include/usb_cam</code>	Added reference to <code>ros/forwards.h</code>	3 months ago
<code>launch</code>	refactor <code>usb_cam</code> launches	3 months ago
<code>src</code>	fix memory leak in <code>camera_driver.cpp</code>	3 months ago
<code>.gitignore</code>	cleanup of <code>readme</code> and such	9 years ago
<code>.travis.yml</code>	add noetic <code>.travis.yml</code>	3 years ago
<code>AUTHORS.md</code>	Update <code>AUTHORS.md</code>	last month
<code>CHANGELOG.rst</code>	0.3.7	3 months ago
<code>CMakeLists.txt</code>	boost::bind workaround update (by @knorth55)	3 months ago
<code>LICENSE</code>	cleanup of <code>readme</code> and such	9 years ago
<code>README.md</code>	README update - bleeding-edge version	2 months ago



usb_cam package

usb_cam (ros2 driver)

Link: branch ros2

The screenshot shows the GitHub repository page for `ros-drivers/usb_cam`. The repository has 5 branches and 27 tags. The `develop` branch is currently selected. The main content area displays a list of 303 commits, with the most recent being a merge commit from `7b480fd` on May 26. Other commits include updates to workflows, ROS parameters, and camera driver logic. The right sidebar contains an 'About' section with a description of the repository as a ROS Driver for V4L USB Cameras, links to the wiki, and various statistics: 398 stars, 539 forks, and 30 people watching. There are also sections for 'Releases' (with one latest release) and 'Report repository'.



usb_cam package

usb_cam (ros2 driver)

```
$ git clone -b ros2 https://github.com/ros-drivers/usb\_cam.git
```

The screenshot shows a terminal window with four tabs open, all titled "rlmodel@rlmodel: ~/rlmodel/src". The active tab displays the command-line session for cloning the "usb_cam" package:

```
rlmodel@rlmodel:~/rlmodel$ l
build/ install/ log/ src/
rlmodel@rlmodel:~/rlmodel$ cd src
rlmodel@rlmodel:~/rlmodel/src$ l
articubot_one/ cv_basics/ gcamp_gazebo/ RLCar/ ROS2_edu/
rlmodel@rlmodel:~/rlmodel/src$ git clone -b ros2 https://github.com
/ros-drivers/usb_cam.git
Cloning into 'usb_cam'...
remote: Enumerating objects: 2100, done.
remote: Counting objects: 100% (349/349), done.
remote: Compressing objects: 100% (164/164), done.
remote: Total 2100 (delta 184), reused 260 (delta 158), pack-reused
1751
Receiving objects: 100% (2100/2100), 844.44 KiB | 12.99 MiB/s, done
.
Resolving deltas: 100% (937/937), done.
rlmodel@rlmodel:~/rlmodel/src$
```

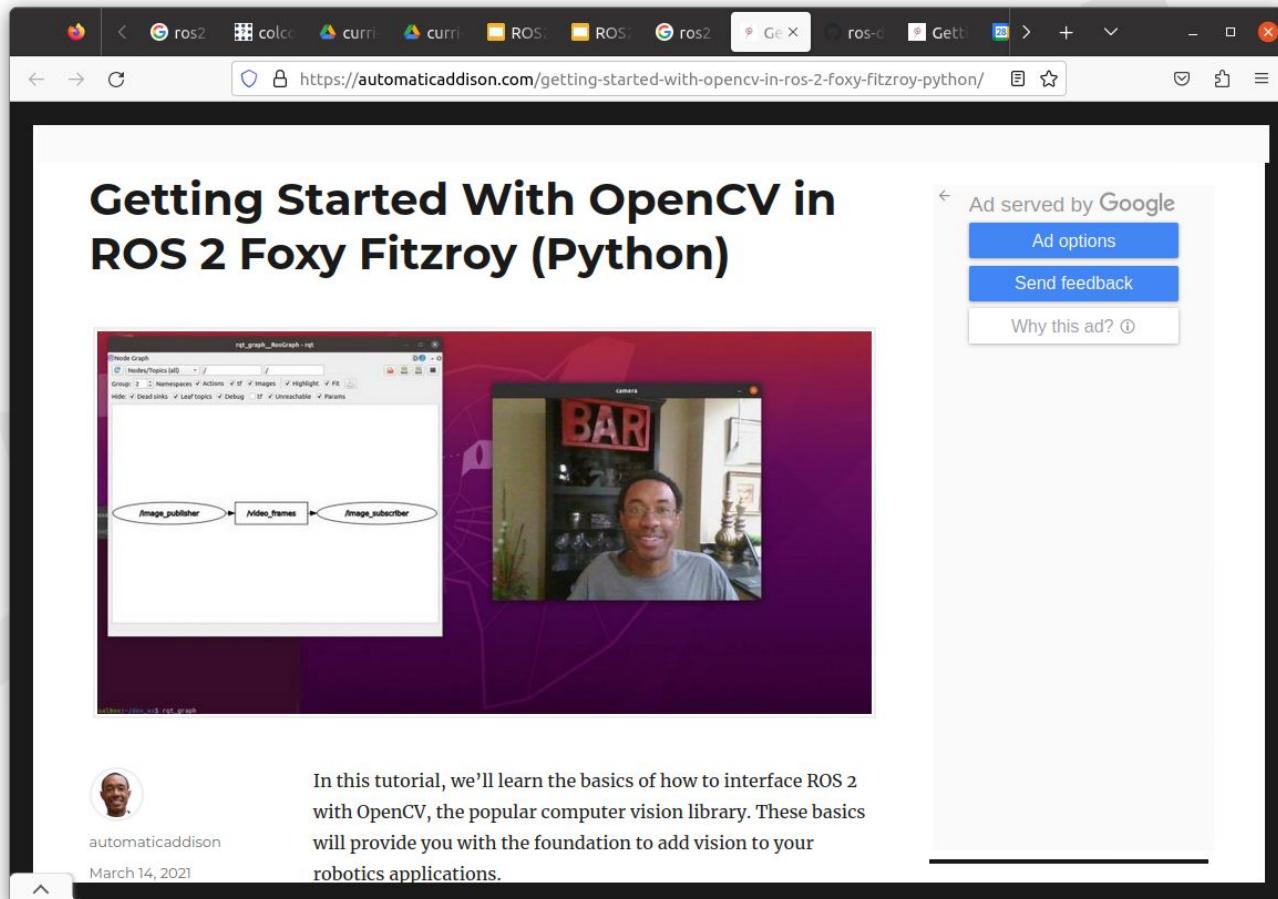


cv_bridge Package example

cv package

ref

<https://automaticaddison.com/getting-started-with-opencv-in-ros-2-foxy-fitzroy-python/>





cv package

Package creation

```
ros2 pkg create --build-type ament_python cv_basics --dependencies rclpy image_transport  
cv_bridge sensor_msgs std_msgs opencv2
```

A terminal window titled "rlmodel@rlmodel: ~/ros2_ws/src" is shown. The command "ros2 pkg create --build-type ament_python cv_basics --dependencies rclpy image_transport cv_bridge sensor_msgs std_msgs opencv2" is entered. The output shows the creation of a new package named "cv_basics" in the directory "/home/rlmodel/ros2_ws/src". The package is version 0.0.0 with a TODO description and maintainer "rlmodel <ybbae@rlmodel.com>". It uses ament_python build type and has dependencies on rclpy, image_transport, cv_bridge, sensor_msgs, std_msgs, and opencv2. The process creates a package.xml file, a source folder, a cv_basics folder, setup.py, setup.cfg, resource folder, cv_basics/_init_.py, test folder, test_copyright.py, test_flake8.py, and test_pep257.py files.

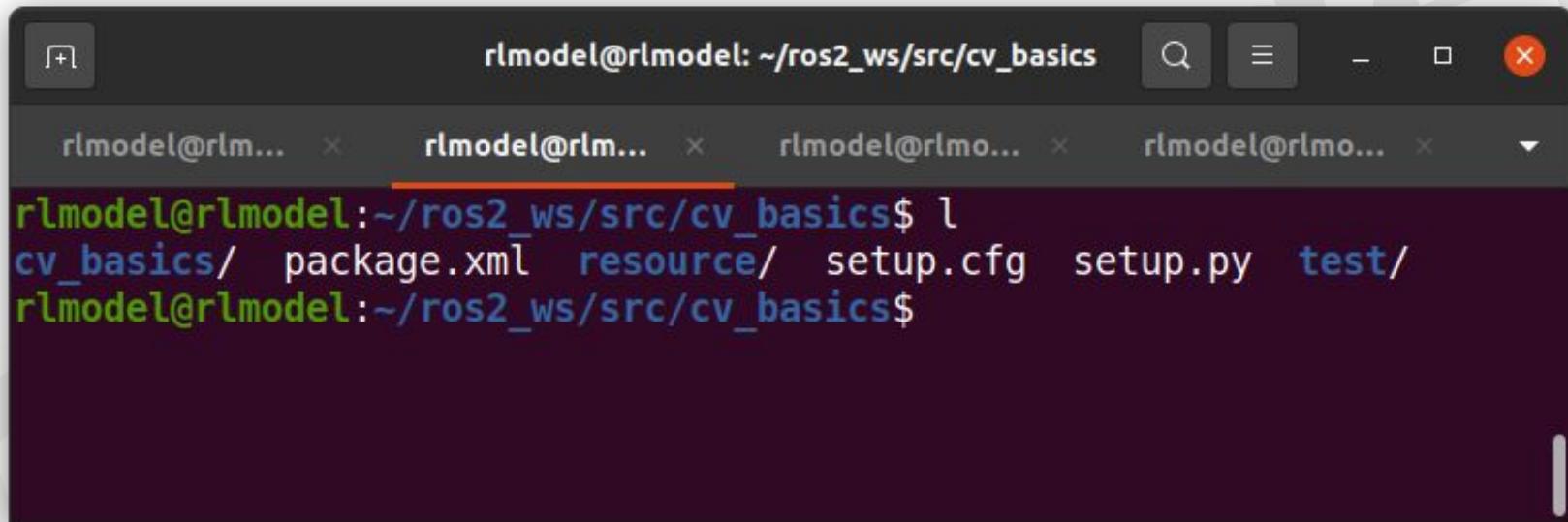
```
rlmodel@rlmodel:~/ros2_ws/src$ ros2 pkg create --build-type ament_python cv_basics --dependencies rclpy  
image_transport cv_bridge sensor_msgs std_msgs opencv2  
going to create a new package  
package name: cv_basics  
destination directory: /home/rlmodel/ros2_ws/src  
package format: 3  
version: 0.0.0  
description: TODO: Package description  
maintainer: ['rlmodel <ybbae@rlmodel.com>']  
licenses: ['TODO: License declaration']  
build type: ament_python  
dependencies: ['rclpy', 'image_transport', 'cv_bridge', 'sensor_msgs', 'std_msgs', 'opencv2']  
creating folder ./cv_basics  
creating ./cv_basics/package.xml  
creating source folder  
creating folder ./cv_basics/cv_basics  
creating ./cv_basics/setup.py  
creating ./cv_basics/setup.cfg  
creating folder ./cv_basics/resource  
creating ./cv_basics/resource/cv_basics  
creating ./cv_basics/cv_basics/_init_.py  
creating folder ./cv_basics/test  
creating ./cv_basics/test/test_copyright.py  
creating ./cv_basics/test/test_flake8.py  
creating ./cv_basics/test/test_pep257.py  
rlmodel@rlmodel:~/ros2_ws/src$
```



cv package

Package check

ros2_ws/src/cv_basics



```
rlmodel@rlmodel: ~/ros2_ws/src/cv_basics$ ls
cv_basics/  package.xml  resource/  setup.cfg  setup.py  test/
rlmodel@rlmodel:~/ros2_ws/src/cv_basics$
```

cv package

Package check

location: ros2_ws/src/cv_basics

check: package.xml

The screenshot shows a code editor window titled "package.xml" with the file path "~/ros2_ws/src/cv_basics". The window has standard OS X-style controls (Open, Save, Close) at the top. The code itself is a well-formed XML document with line numbers on the left:

```
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>cv_basics</name>
5   <version>0.0.0</version>
6   <description>TODO: Package description</description>
7   <maintainer email="ybbaek@rlmodel.com">rlmodel</maintainer>
8   <license>TODO: License declaration</license>
9
10  <depend>rclpy</depend>
11  <depend>image_transport</depend>
12  <depend>cv_bridge</depend>
13  <depend>sensor_msgs</depend>
14  <depend>std_msgs</depend>
15  <depend>opencv2</depend>
16
17  <test_depend>ament_copyright</test_depend>
18  <test_depend>ament_flake8</test_depend>
19  <test_depend>ament_pep257</test_depend>
20  <test_depend>python3-pytest</test_depend>
21
22  <export>
23    <build_type>ament_python</build_type>
24  </export>
25 </package>
```

At the bottom of the editor, there are status indicators: "XML" (with a dropdown arrow), "Tab Width: 8", "Ln 1, Col 1", and "INS".



cv package

New python file

new file: gedit webcam_pub.py

```
rlmodel@rlmodel:~/ros2_ws/src/cv_basics/cv_basics$ gedit webcam_pub.py
```

gedit window title: webcam_pub2.py
File path: ~/ros2_ws/src/cv_basics/cv_basics

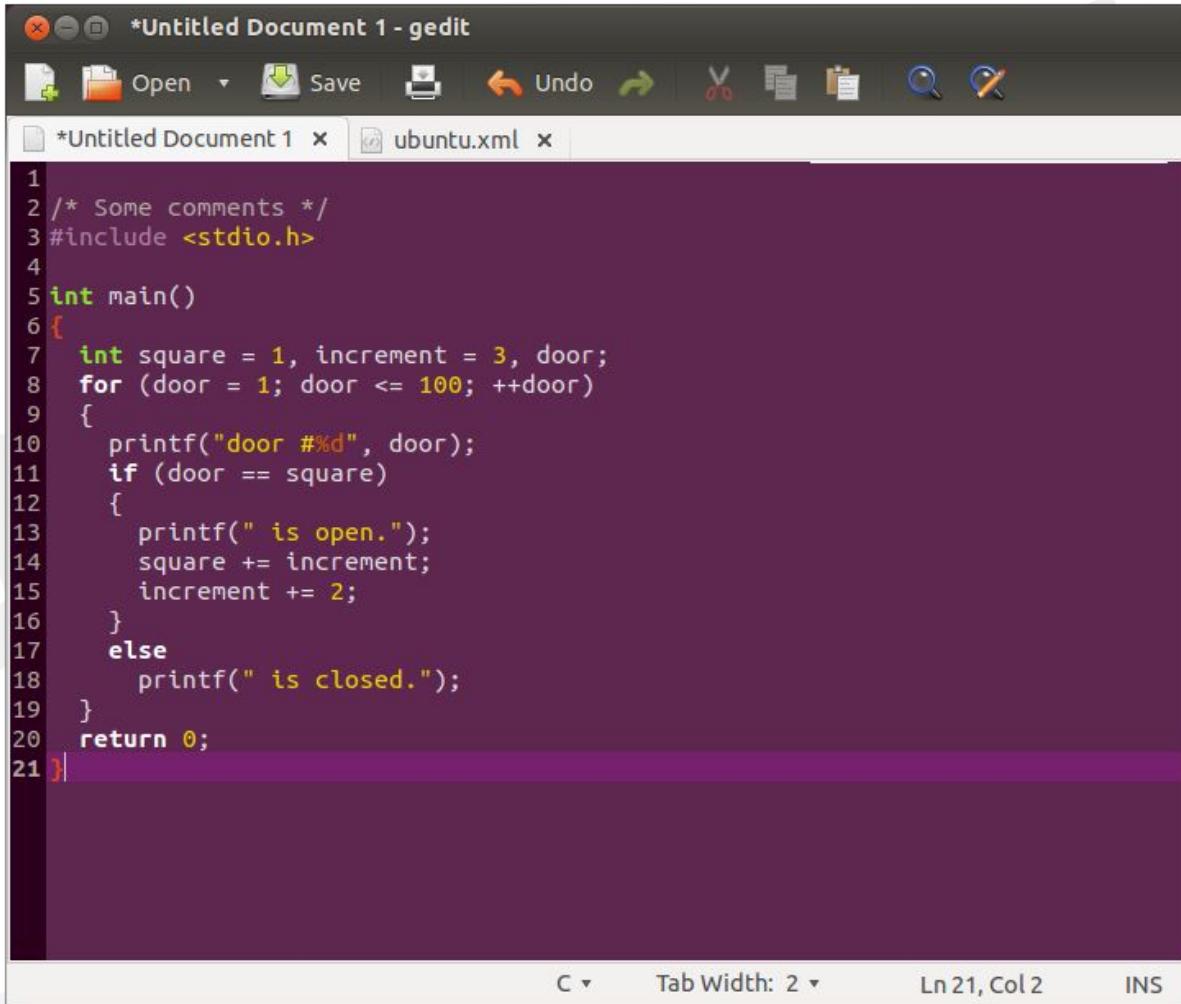
```
1
```

Editor status bar: Python ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

cv package

geditor

install: sudo apt-get install gedit (if error)



The screenshot shows the Gedit text editor interface. The title bar reads "*Untitled Document 1 - gedit". The menu bar includes "File", "Edit", "View", "Insert", "Search", "Format", "Tools", and "Help". The toolbar contains icons for Open, Save, Undo, Redo, Cut, Copy, Paste, Find, and Replace. The main window displays a C program with syntax highlighting. The code prints a message for each door from 1 to 100, checking if it's a square number. If it is, the increment is increased by 2; otherwise, it remains at 3. The status bar at the bottom shows "C", "Tab Width: 2", "Ln 21, Col 2", and "INS".

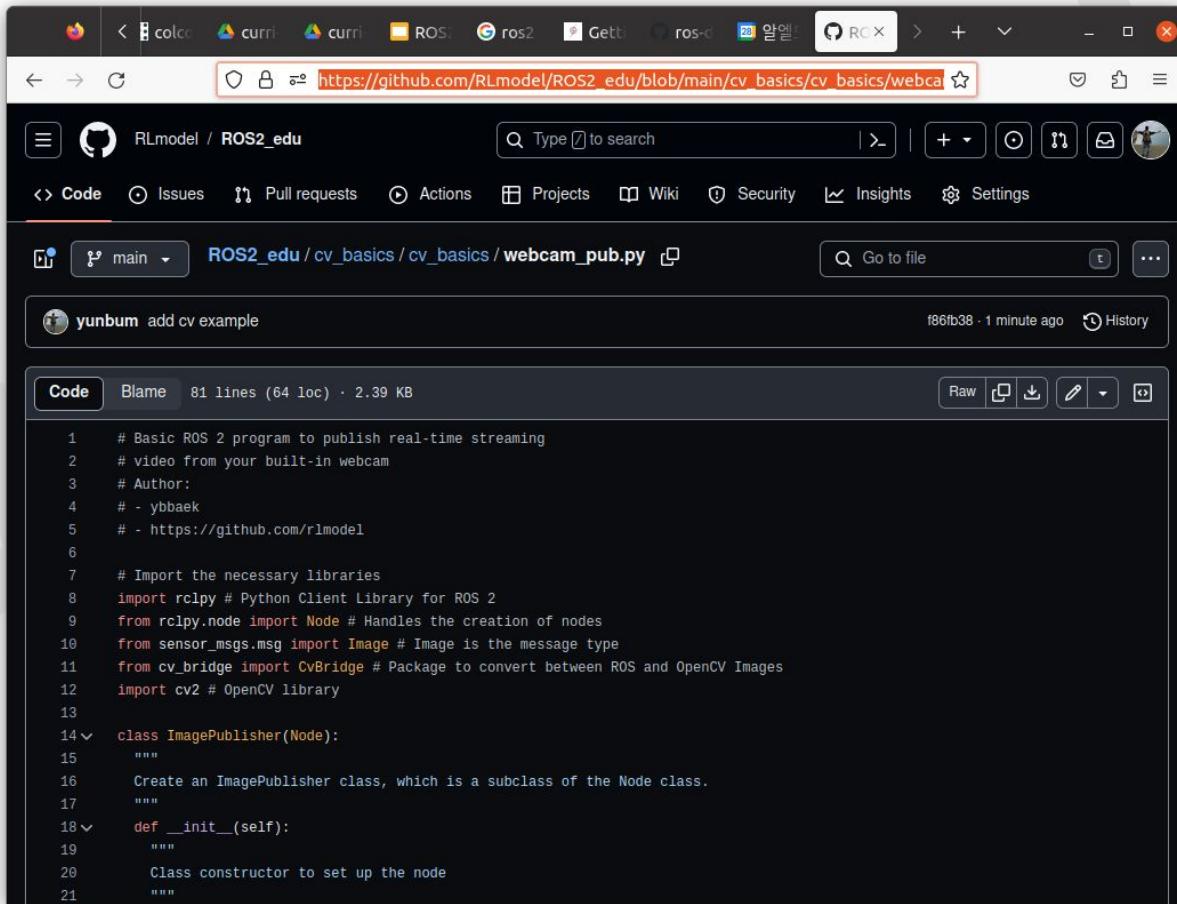
```
1 /* Some comments */
2 #include <stdio.h>
3
4 int main()
5 {
6     int square = 1, increment = 3, door;
7     for (door = 1; door <= 100; ++door)
8     {
9         printf("door #%d", door);
10        if (door == square)
11        {
12            printf(" is open.");
13            square += increment;
14            increment += 2;
15        }
16        else
17            printf(" is closed.");
18    }
19    return 0;
20 }
```

cv package

geditor

source:

https://github.com/RLmodel/ROS2_edu/blob/main/cv_basics/cv_basics/webcam_pub.py



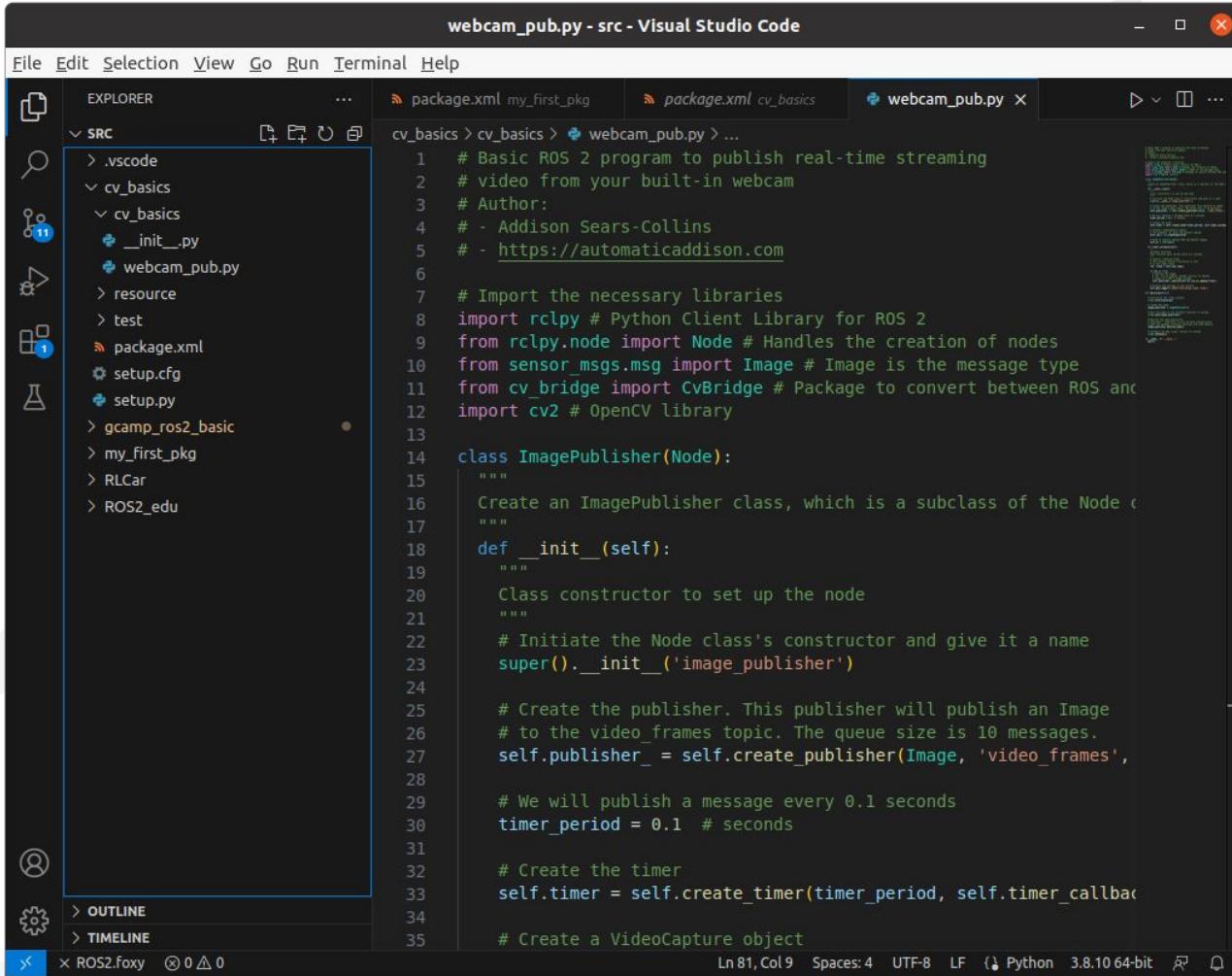
The screenshot shows a web browser window displaying a GitHub repository page. The URL in the address bar is https://github.com/RLmodel/ROS2_edu/blob/main/cv_basics/cv_basics/webcam_pub.py. The page title is "ROS2_edu / cv_basics / cv_basics / webcam_pub.py". The commit message is "yunbum add cv example" and the commit hash is "f86fb38 · 1 minute ago". The code editor shows the following Python script:

```
1 # Basic ROS 2 program to publish real-time streaming
2 # video from your built-in webcam
3 # Author:
4 # - ybbaek
5 # - https://github.com/rlmodel
6
7 # Import the necessary libraries
8 import rclpy # Python Client Library for ROS 2
9 from rclpy.node import Node # Handles the creation of nodes
10 from sensor_msgs.msg import Image # Image is the message type
11 from cv_bridge import CvBridge # Package to convert between ROS and OpenCV Images
12 import cv2 # OpenCV library
13
14 class ImagePublisher(Node):
15     """
16     Create an ImagePublisher class, which is a subclass of the Node class.
17     """
18     def __init__(self):
19         """
20         Class constructor to set up the node
21         """
```

cv package

cv_bridge coding

code review: class, node, image handling



The screenshot shows a Visual Studio Code window with the title "webcam_pub.py - src - Visual Studio Code". The Explorer sidebar on the left shows a project structure under "SRC" with files like .vscode, cv_bridge, cv_bridge/_init_.py, cv_bridge/webcam_pub.py, resource, test, package.xml, setup.cfg, setup.py, gcamp_ros2_basic, my_first_pkg, RLCar, and ROS2_edu. The main editor area displays Python code for a ROS 2 node named "ImagePublisher". The code imports rclpy, Node, Image, CvBridge, and cv2. It defines a class ImagePublisher(Node) with an __init__ method that initializes the node, creates a publisher for the 'video_frames' topic, and sets a timer to publish messages every 0.1 seconds. A terminal tab at the bottom shows ROS 2 Foxy output.

```
# Basic ROS 2 program to publish real-time streaming
# video from your built-in webcam
# Author:
# - Addison Sears-Collins
# - https://automaticaddison.com

# Import the necessary libraries
import rclpy # Python Client Library for ROS 2
from rclpy.node import Node # Handles the creation of nodes
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and
import cv2 # OpenCV library

class ImagePublisher(Node):
    """
    Create an ImagePublisher class, which is a subclass of the Node class
    """
    def __init__(self):
        """
        Class constructor to set up the node
        """
        # Initiate the Node class's constructor and give it a name
        super().__init__('image_publisher')

        # Create the publisher. This publisher will publish an Image
        # to the video_frames topic. The queue size is 10 messages.
        self.publisher_ = self.create_publisher(Image, 'video_frames',
                                                10)

        # We will publish a message every 0.1 seconds
        timer_period = 0.1 # seconds

        # Create the timer
        self.timer = self.create_timer(timer_period, self.timer_callback)

        # Create a VideoCapture object
        self.cap = cv2.VideoCapture(0)
```

Ln 81, Col 9 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit



cv package

cv_bridge coding

code review: class, node, image handling

```
Open  ▾  ⌂  webcam_pub.py  ~/ros2_ws/src/cv_basics/cv_basics  Save  ⏺  -  ⌂  X
1 # Basic ROS 2 program to publish real-time streaming
2 # video from your built-in webcam
3 # Author:
4 # - Addison Sears-Collins
5 # - https://automaticaddison.com
6
7 # Import the necessary libraries
8 import rclpy # Python Client Library for ROS 2
9 from rclpy.node import Node # Handles the creation of nodes
10 from sensor_msgs.msg import Image # Image is the message type
11 from cv_bridge import CvBridge # Package to convert between ROS and OpenCV Images
12 import cv2 # OpenCV library
13
14 class ImagePublisher(Node):
15     """
16     Create an ImagePublisher class, which is a subclass of the Node class.
17     """
18     def __init__(self):
19         """
20         Class constructor to set up the node
21         """
22         # Initiate the Node class's constructor and give it a name
23         super().__init__('image_publisher')
24
25         # Create the publisher. This publisher will publish an Image
26         # to the video_frames topic. The queue size is 10 messages.
27         self.publisher_ = self.create_publisher(Image, 'video_frames', 10)
28
29         # We will publish a message every 0.1 seconds
30         self.timer_ = self.create_timer(0.1, self.publish_video_frame_callback)
```

Python ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

cv package



cv_bridge coding

code review: class, node, image handling

The screenshot shows a Visual Studio Code interface with the title bar "webcam_pub.py - cv_basics - Visual Studio Code". The left sidebar contains icons for Explorer, Search, Open, Resource, Test, Package XML, Setup Configuration, and Setup Script. The "EXPLORER" tab is selected, showing a file tree with a folder "CV_BASICS" containing ".vscode", "cv_basics", "__init__.py", "webcam_pub.py", "resource", "test", "package.xml", "setup.cfg", and "setup.py". The "webcam_pub.py" file is currently open in the main editor area. The code is a Python script using the ROS 2 Client Library (rclpy) to publish real-time streaming video from a built-in webcam. It imports necessary libraries, defines a class `ImagePublisher` that inherits from `Node`, and creates a publisher named `image_publisher` that publishes `Image` messages to the `video_frames` topic with a queue size of 10. A timer is set to publish a message every 0.1 seconds.

```
# Basic ROS 2 program to publish real-time streaming
# video from your built-in webcam
# Author:
# - RLmodel ybbaek
# - https://github.com/rlmodel

# Import the necessary libraries
import rclpy # Python Client Library for ROS 2
from rclpy.node import Node # Handles the creation of nodes
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and
# OpenCV library

class ImagePublisher(Node):
    """
    Create an ImagePublisher class, which is a subclass of the Node class
    """
    def __init__(self):
        """
        Class constructor to set up the node
        """
        # Initiate the Node class's constructor and give it a name
        super().__init__('image_publisher')

        # Create the publisher. This publisher will publish an Image
        # to the video_frames topic. The queue size is 10 messages.
        self.publisher_ = self.create_publisher(Image, 'video_frames',
                                              10)

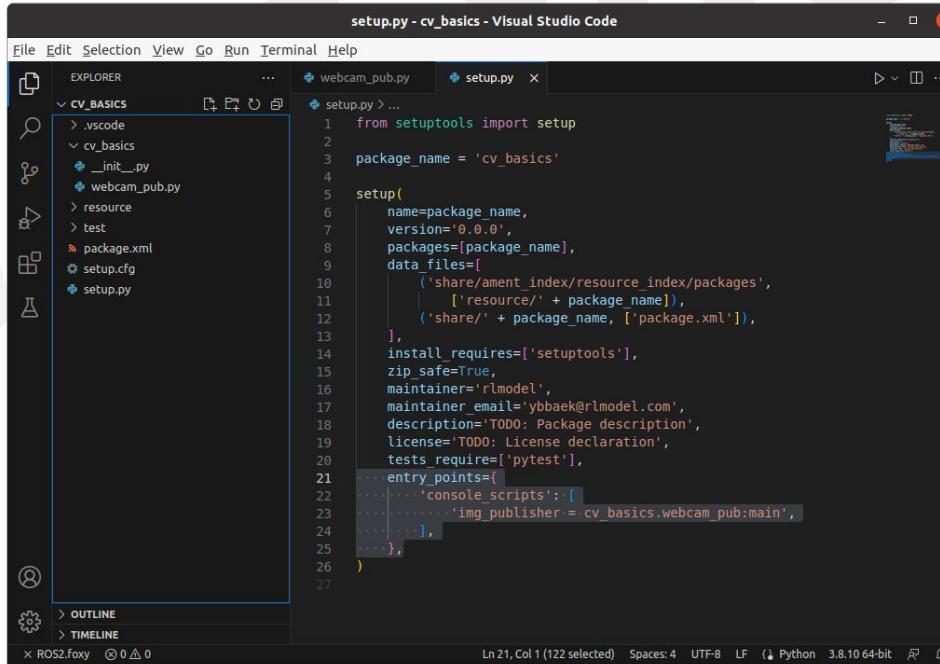
        # We will publish a message every 0.1 seconds
        timer_period = 0.1 # seconds
```

cv package

cv_bridge coding

entry_point fks: node python connection

```
entry_points={  
    'console_scripts': [  
        'img_publisher = cv_basics.webcam_pub:main',  
    ],  
},
```



```
setup.py - cv_basics - Visual Studio Code  
File Edit Selection View Go Run Terminal Help  
EXPLORER ... Webcam_pub.py setup.py ...  
< .vscode > ...  
< cv_basics >  
< _init_.py >  
< webcam_pub.py >  
< resource >  
< test >  
< package.xml >  
< setup.cfg >  
< setup.py >  
  
setup(  
    name=package_name,  
    version='0.0.0',  
    packages=[package_name],  
    data_files=[  
        ('share/ament_index/resource_index/packages',  
         ['resource/' + package_name]),  
        ('share/' + package_name, ['package.xml']),  
    ],  
    install_requires=['setuptools'],  
    zip_safe=True,  
    maintainer='rlmodel',  
    maintainer_email='ybbaek@rlmodel.com',  
    description='TODO: Package description',  
    license='TODO: License declaration',  
    tests_require=['pytest'],  
    entry_points={  
        'console_scripts': [  
            'img_publisher = cv_basics.webcam_pub:main',  
        ],  
    },  
)  
  
Ln 21, Col 1 (122 selected) Spaces: 4 UTF-8 LF (Python 3.8.10 64-bit) ⌂ ⌂
```



cv package

cv_bridge coding

compile: colcon build --packages-select cv_basic

```
rlmodel@rlmodel:~/ros2_ws$ colcon build --packages-select cv_basics
Starting >>> cv_basics
Finished <<< cv_basics [1.33s]

Summary: 1 package finished [1.69s]
rlmodel@rlmodel:~/ros2_ws$ █
```

cv package

cv_bridge coding

sourcing: foxy (.bashrc alias)

```
>. /opt/ros/foxy/setup.bash
```

```
>. install/setup.bash
```

```
rlmodel@rlmodel:~/ros2_ws$ ros2 pkg list
ackermann_msgs
action_msgs
action_tutorials_cpp
action_tutorials_interfaces
action_tutorials_py
actionlib_msgs
ament_cmake
-----
```

```
rlmodel@rlmodel:~/ros2_ws$ ros2 pkg list
cpp_service_pkg
cpp_service_tutorial
cpp_topic_pkg
custom_interfaces
cv_basics
cv_bridge
demo_nodes_cpp
demo_nodes_cpp_native
demo_nodes_py
-----
```



cv package

cv_bridge coding

execute: ros2 run cv_basic img_publisher

```
rlmodel@rlmodel:~/ros2_ws$ ros2 run cv_basics img_publisher
[ WARN:0] global ..../modules/videoio/src/cap_gstreamer.cpp (935) open OpenCV | GStreamer warning: Cannot query video position: status=0, value=-1, duration=-1
[INFO] [1687995231.875946280] [image_publisher]: Publishing video frame
[INFO] [1687995232.044108275] [image_publisher]: Publishing video frame
[INFO] [1687995232.249454115] [image_publisher]: Publishing video frame
[INFO] [1687995232.450184318] [image_publisher]: Publishing video frame
[INFO] [1687995232.649637064] [image_publisher]: Publishing video frame
[INFO] [1687995232.846305031] [image_publisher]: Publishing video frame
[INFO] [1687995233.044035859] [image_publisher]: Publishing video frame
[INFO] [1687995233.247078651] [image_publisher]: Publishing video frame
[INFO] [1687995233.446980411] [image_publisher]: Publishing video frame
[INFO] [1687995233.647596244] [image_publisher]: Publishing video frame
[INFO] [1687995233.847047275] [image_publisher]: Publishing video frame
[INFO] [1687995234.047025997] [image_publisher]: Publishing video frame
```



cv package

cv_bridge coding

check topic: ros2 topic list

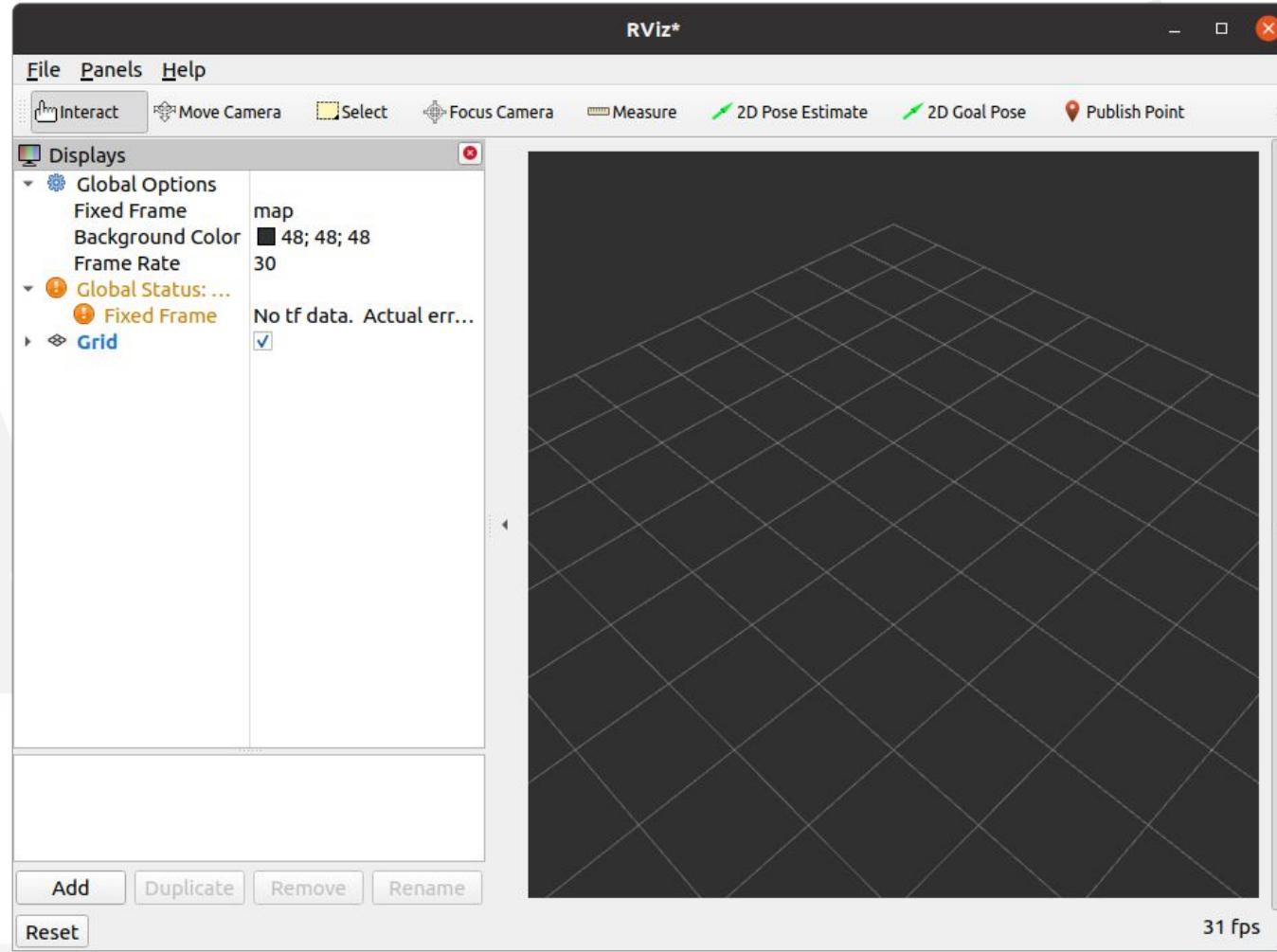
```
rlmodel@rlmodel: ~/ros2_ws$ ros2 topic list
/parameter_events
/rosout
/video_frames
rlmodel@rlmodel:~/ros2_ws$
```



cv package

cv_bridge coding

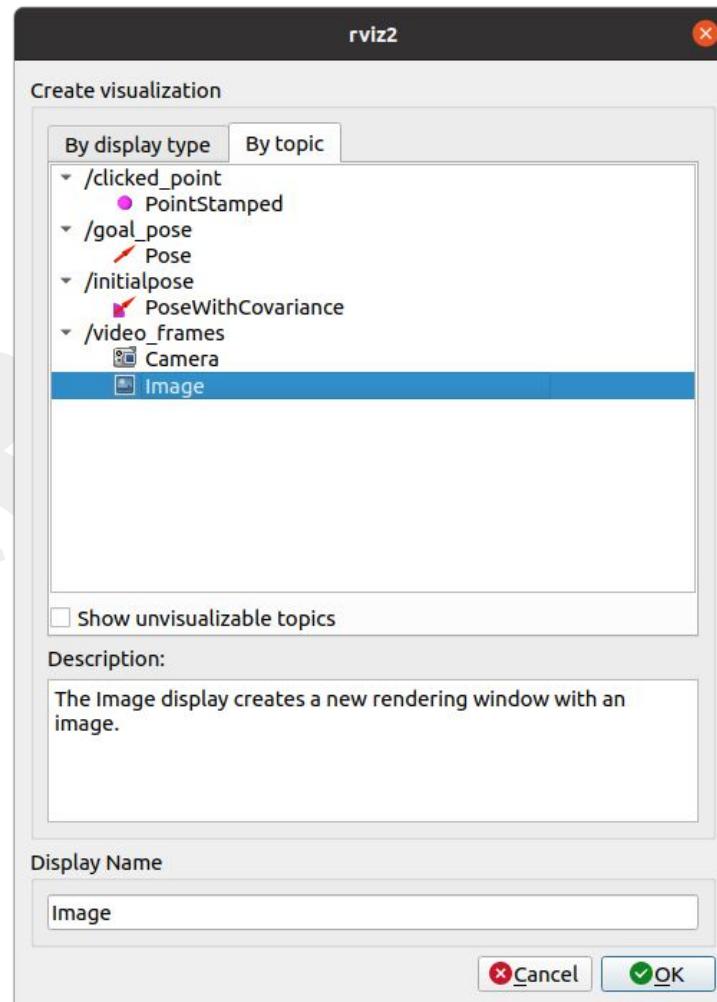
check topic: rviz



cv package

cv_bridge coding

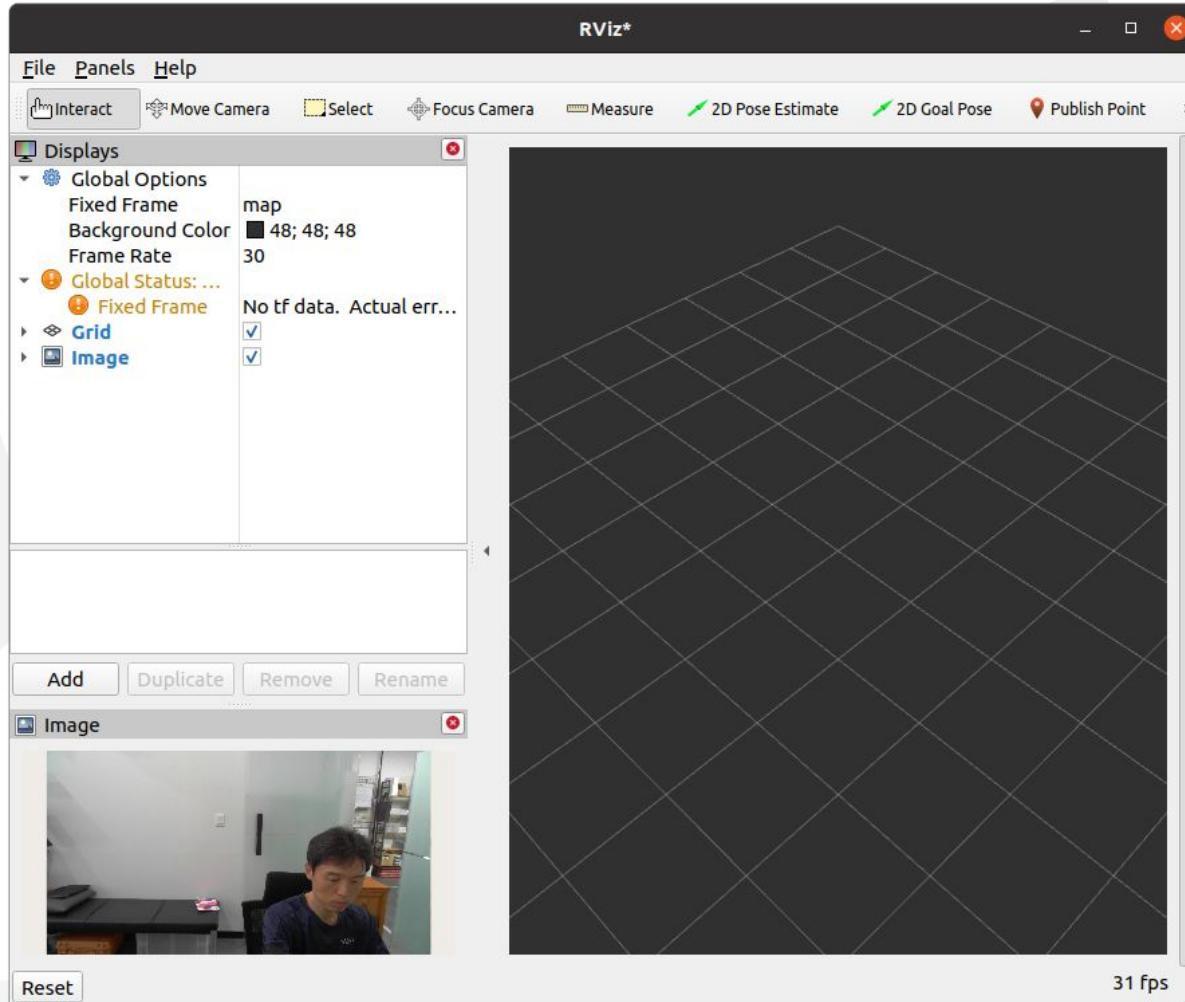
check topic: rviz



cv package

cv_bridge coding

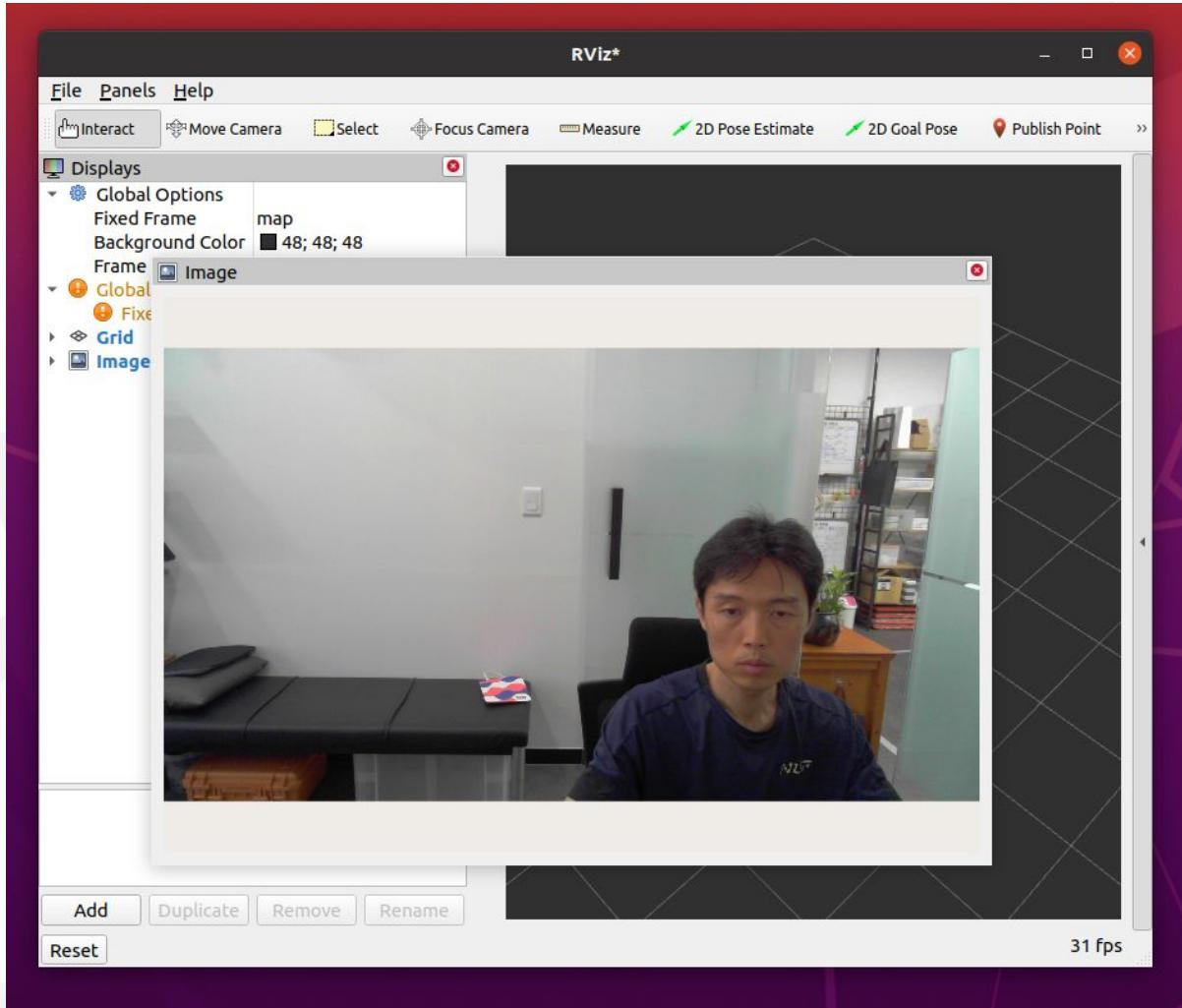
check topic: rviz



cv package

cv_bridge coding

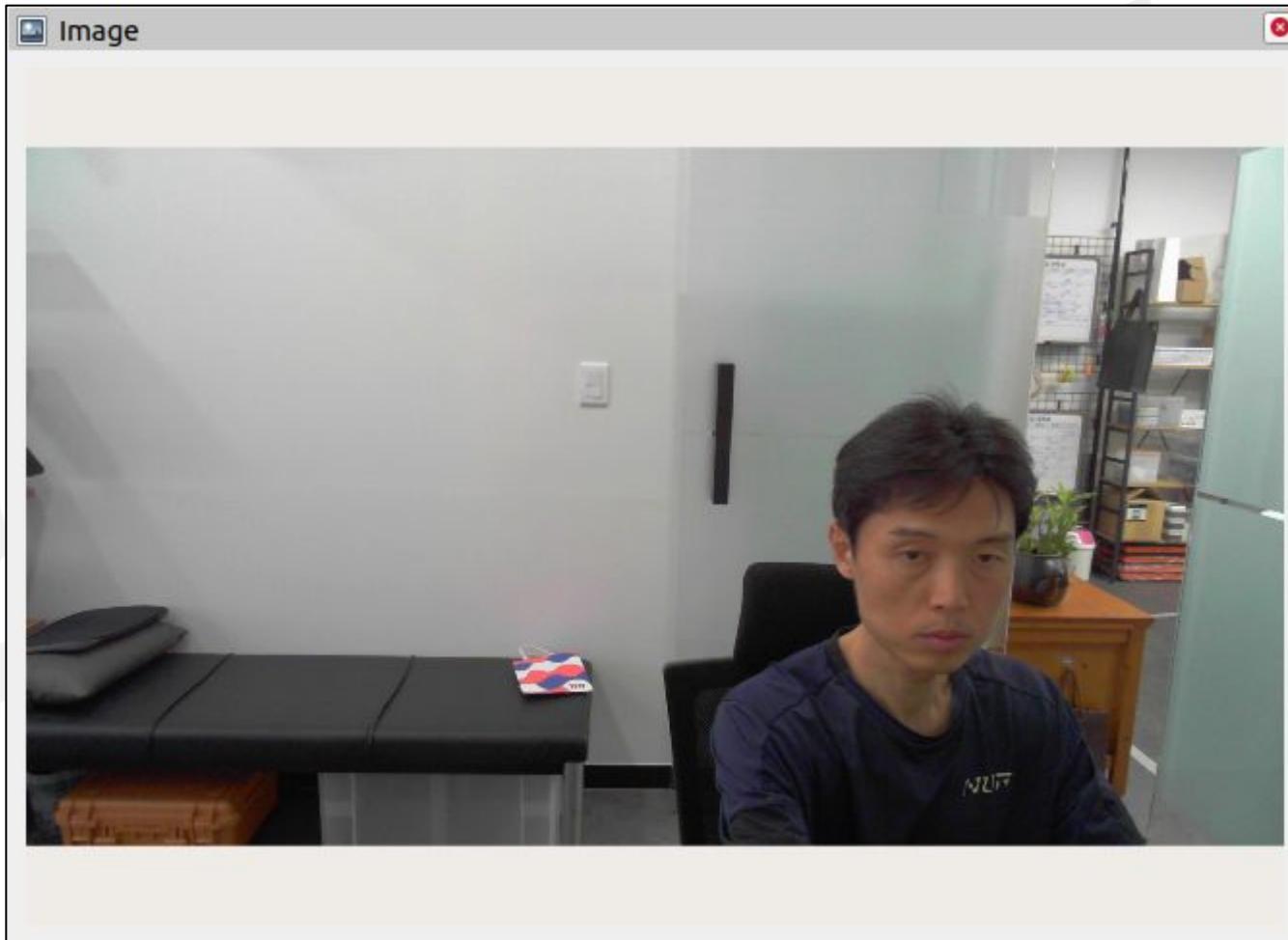
check topic: rviz



cv package

cv_bridge coding

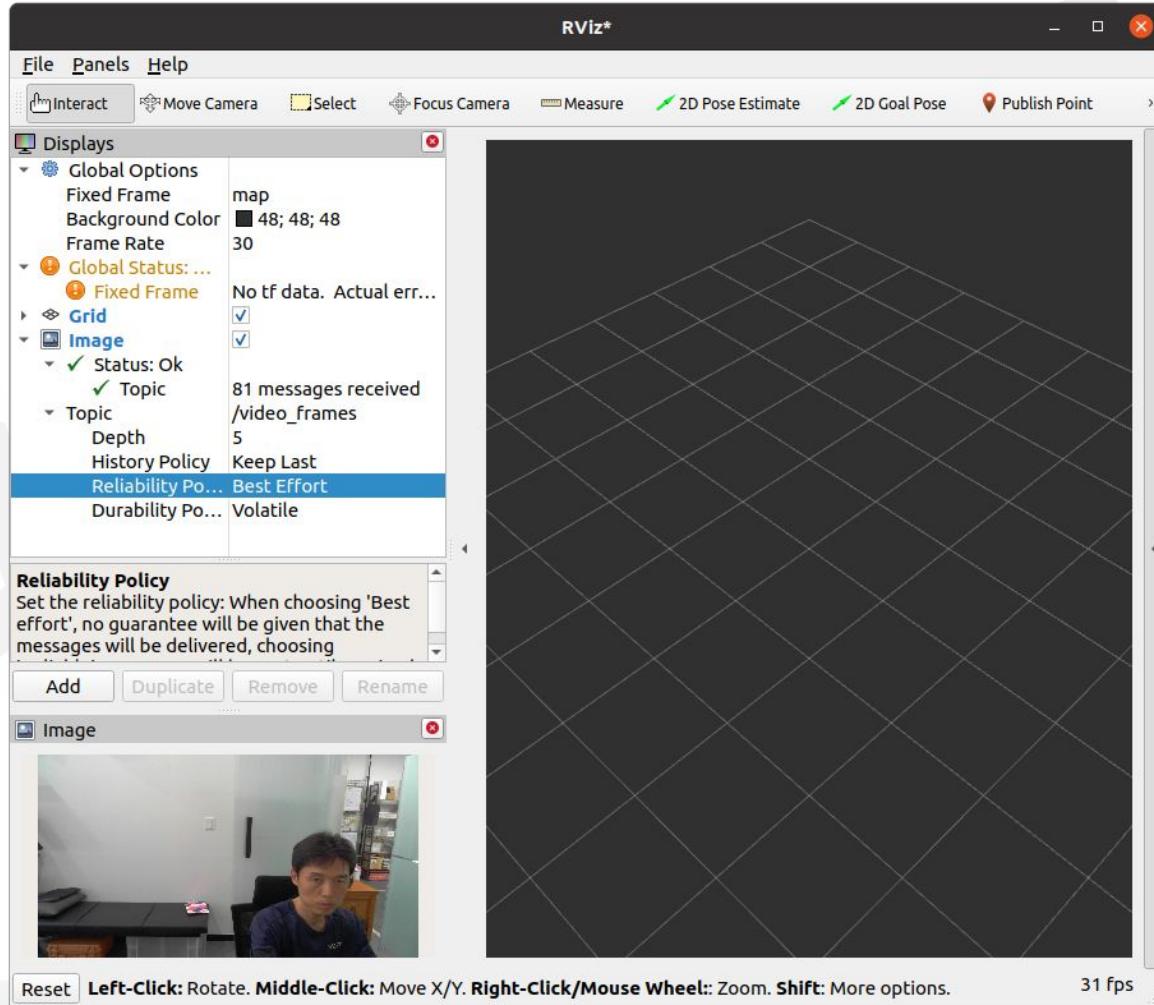
check topic: rviz



cv package

cv_bridge coding

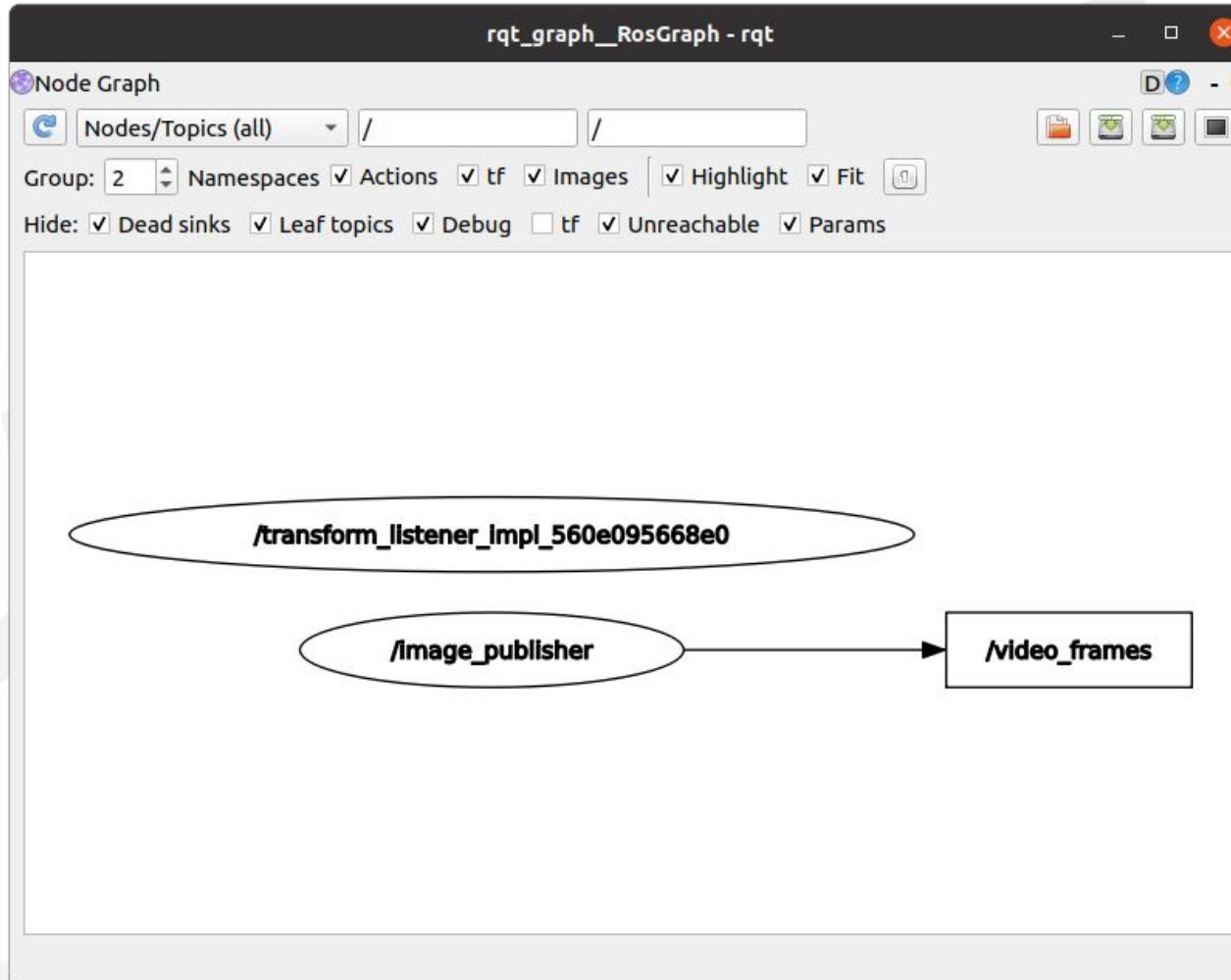
check DDS: Best effort, Reliable



cv package

cv_bridge coding

check rqt_graph: ros2 run rqt_graph rqt_graph





cv package

cv_bridge coding

image topic subscribe: new webcam_sub.py python file

The screenshot shows the Visual Studio Code interface with the title bar "webcam_sub.py - cv_basics - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left shows the project structure under "CV_BASICS": .vscode, cv_basics (containing __init__.py, webcam_pub.py, and webcam_sub.py), resource, test, package.xml, setup.cfg, and setup.py. The "webcam_sub.py" file is selected in the Explorer. The main editor area displays the Python code for the "webcam_sub.py" file:

```
1 # Basic ROS 2 program to subscribe to real-time streaming
2 # video from your built-in webcam
3 # Author:
4 # - RLmodel ybbaek
5 # - https://www.rlmodel.com
6
7 # Import the necessary libraries
8 import rclpy # Python library for ROS 2
9 from rclpy.node import Node # Handles the creation of nodes
10 from sensor_msgs.msg import Image # Image is the message type
11 from cv_bridge import CvBridge # Package to convert between ROS and OpenCV images
12 import cv2 # OpenCV library
13
14 class ImageSubscriber(Node):
15     """
16     Create an ImageSubscriber class, which is a subclass of the Node
17     """
18     def __init__(self):
19         """
20         Class constructor to set up the node
21         """
22         # Initiate the Node class's constructor and give it a name
23         super().__init__('image_subscriber')
24
25         # Create the subscriber. This subscriber will receive an Image
26         # from the video_frames topic. The queue size is 10 messages.
27         self.subscription = self.create_subscription(
28             Image,
29             'video_frames',
30             self.listener_callback,
```

At the bottom of the status bar, it says "ROS2.foxy" and "Ln 5, Col 28 Spaces: 2 UTF-8 LF Python 3.8.10 64-bit". The page number "107" is located at the bottom right.

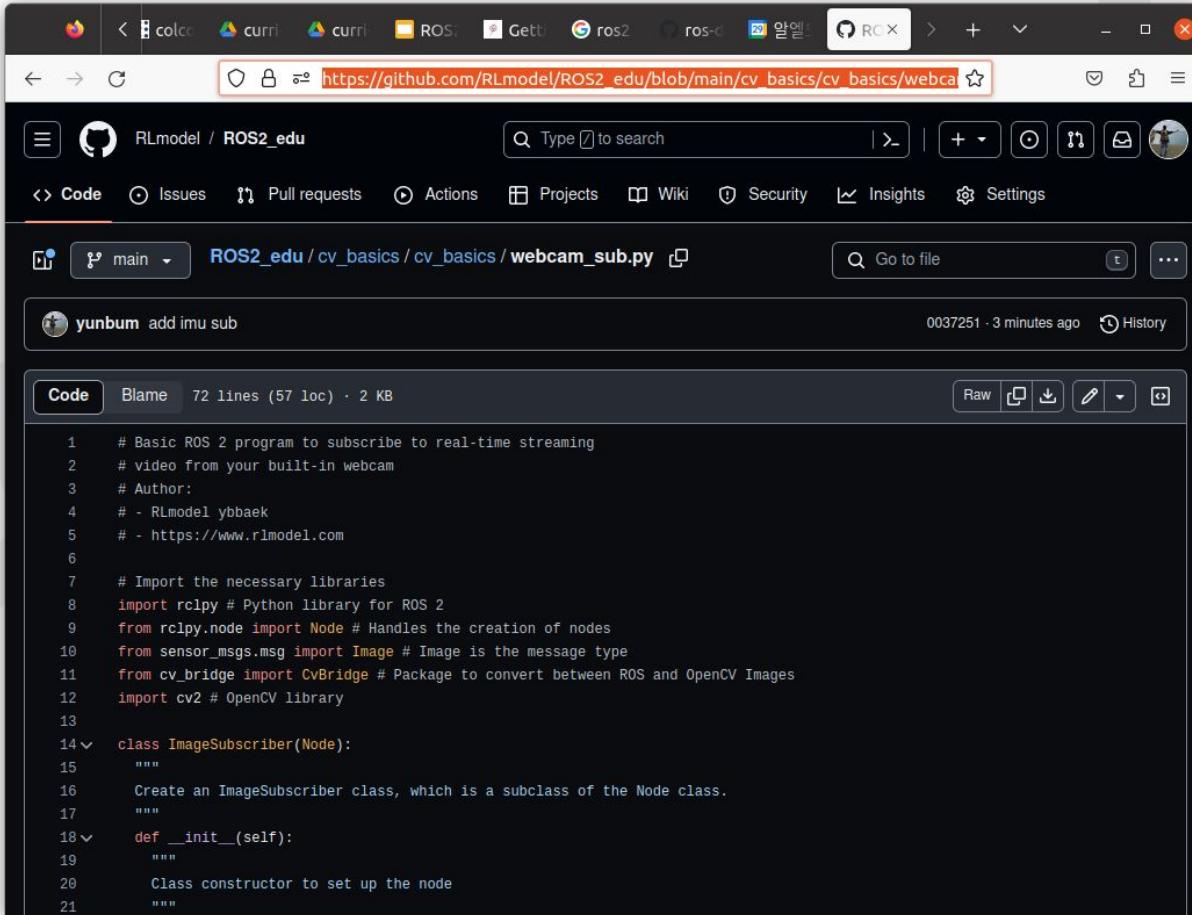


cv package

cv_bridge coding

source link:

https://github.com/RLmodel/ROS2_edu/blob/main/cv_basics/cv_basics/webcam_sub.py



The screenshot shows a GitHub code editor interface for the file `webcam_sub.py`. The repository is `ROS2_edu / cv_basics / cv_basics`. The commit was made by `yunbum` with the message `add imu sub` on 0037251 at 3 minutes ago.

The code editor displays the following Python script:

```
1 # Basic ROS 2 program to subscribe to real-time streaming
2 # video from your built-in webcam
3 # Author:
4 # - RLmodel ybbaek
5 # - https://www.rlmodel.com
6
7 # Import the necessary libraries
8 import rclpy # Python library for ROS 2
9 from rclpy.node import Node # Handles the creation of nodes
10 from sensor_msgs.msg import Image # Image is the message type
11 from cv_bridge import CvBridge # Package to convert between ROS and OpenCV Images
12 import cv2 # OpenCV library
13
14 class ImageSubscriber(Node):
15     """
16     Create an ImageSubscriber class, which is a subclass of the Node class.
17     """
18     def __init__(self):
19         """
20         Class constructor to set up the node
21         """
```

cv package



cv_bridge coding

```
update entry_points: 'img_subscriber = cv_basics.webcam_sub:main',
```

The screenshot shows the Visual Studio Code interface with the title bar "setup.py - cv_basics - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left shows a project structure with a folder "CV_BASICS" containing files like .vscode, cv_basics, __init__.py, webcam_pub.py, webcam_sub.py, resource, test, package.xml, setup.cfg, and setup.py. The setup.py file is currently selected and open in the main editor area. The code is a Python script using setuptools to define a package named 'cv_basics'. It includes setup parameters like name, version, packages, data_files, install_requires, zip_safe, maintainer, and entry_points. The entry_points section defines two console scripts: 'img_publisher = cv_basics.webcam_pub:main' and 'img_subscriber = cv_basics.webcam_sub:main'. The status bar at the bottom shows "Ln 24, Col 13 (45 selected)" and other details about the Python environment.

```
from setuptools import setup
package_name = 'cv_basics'
setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='rlmodel',
    maintainer_email='ybbaek@rlmodel.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'img_publisher = cv_basics.webcam_pub:main',
            'img_subscriber = cv_basics.webcam_sub:main',
        ],
    },
)
```



cv package

cv_bridge coding

update entry_points: setup.py

```
entry_points={  
    'console_scripts': [  
        'img_publisher = cv_basics.webcam_pub:main',  
        'img_subscriber = cv_basics.webcam_sub:main',  
    ],  
},  
)
```



cv package

cv_bridge coding

build: colcon build --packages-select cv_basics

```
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws × rlmodel@rlmodel: ~/ros2_ws × rlmodel@rlmodel: ~/ros2_ws ×
rlmodel@rlmodel:~/ros2_ws$ colcon build --packages-select cv_basics
Starting >>> cv_basics
Finished <<< cv_basics [1.44s]

Summary: 1 package finished [1.80s]
rlmodel@rlmodel:~/ros2_ws$ █
```



cv package

cv_bridge coding

sourcing: foxy

>check auto completion

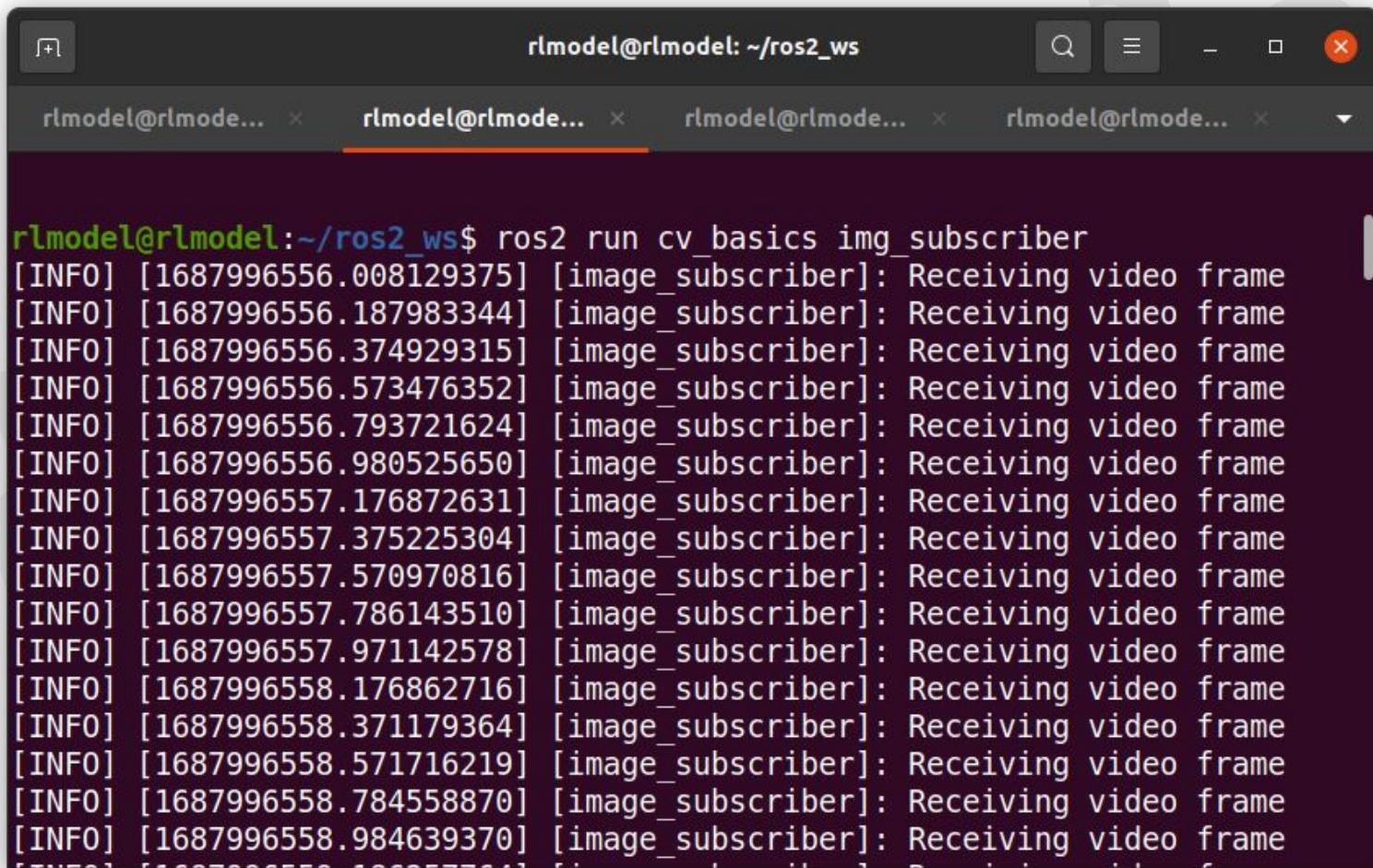
A screenshot of a terminal window titled "rlmodel@rlmodel: ~/ros2_ws". The window has three tabs, all showing the same directory: "rlmodel@rlmodel: ~/ros2_ws". The third tab is active. The command "ros2 run cv_basics img_subscriber" is typed into the terminal. The command is highlighted with a red underline, indicating it is being processed or completed.



cv package

cv_bridge coding

execute: img_subscriber

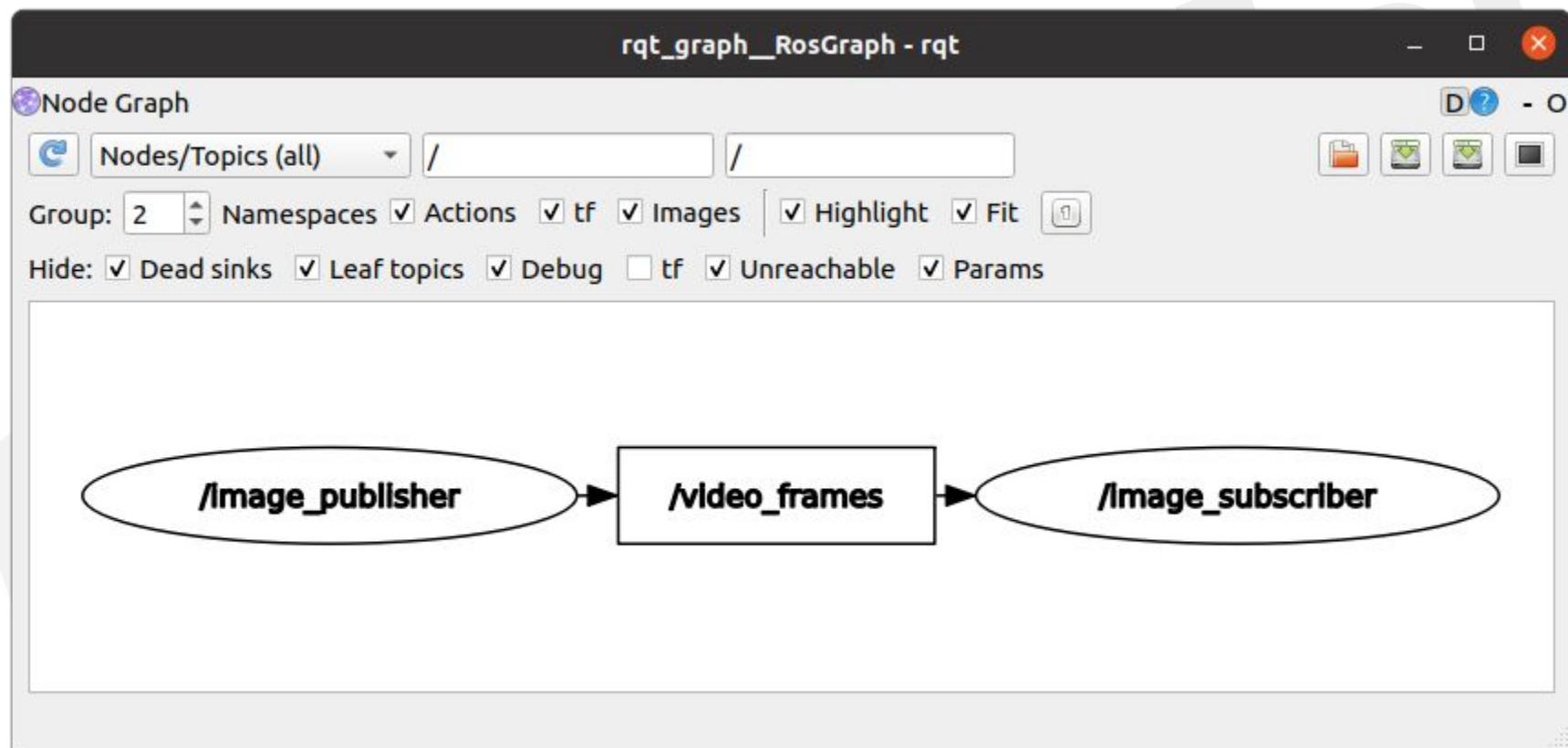


```
rlmodel@rlmodel:~/ros2_ws$ ros2 run cv_basics img_subscriber
[INFO] [1687996556.008129375] [image_subscriber]: Receiving video frame
[INFO] [1687996556.187983344] [image_subscriber]: Receiving video frame
[INFO] [1687996556.374929315] [image_subscriber]: Receiving video frame
[INFO] [1687996556.573476352] [image_subscriber]: Receiving video frame
[INFO] [1687996556.793721624] [image_subscriber]: Receiving video frame
[INFO] [1687996556.980525650] [image_subscriber]: Receiving video frame
[INFO] [1687996557.176872631] [image_subscriber]: Receiving video frame
[INFO] [1687996557.375225304] [image_subscriber]: Receiving video frame
[INFO] [1687996557.570970816] [image_subscriber]: Receiving video frame
[INFO] [1687996557.786143510] [image_subscriber]: Receiving video frame
[INFO] [1687996557.971142578] [image_subscriber]: Receiving video frame
[INFO] [1687996558.176862716] [image_subscriber]: Receiving video frame
[INFO] [1687996558.371179364] [image_subscriber]: Receiving video frame
[INFO] [1687996558.571716219] [image_subscriber]: Receiving video frame
[INFO] [1687996558.784558870] [image_subscriber]: Receiving video frame
[INFO] [1687996558.984639370] [image_subscriber]: Receiving video frame
```

cv package

cv_bridge coding

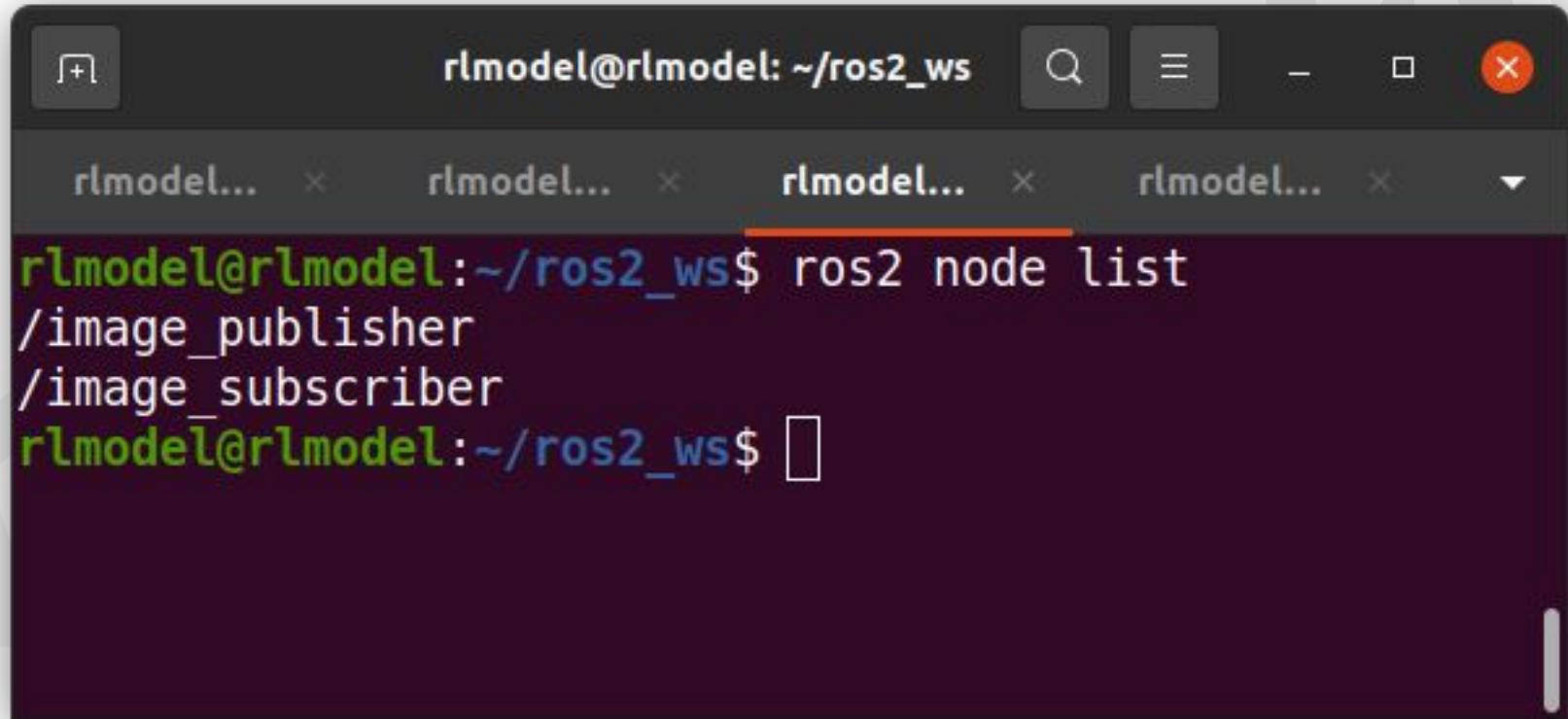
check node: rqt_graph



cv package

cv_bridge coding

check node: ros2 node list



```
rlmodel@rlmodel: ~/ros2_ws$ ros2 node list
/image_publisher
/image_subscriber
rlmodel@rlmodel: ~/ros2_ws$ 
```

A screenshot of a terminal window titled "rlmodel@rlmodel: ~/ros2_ws". The window has four tabs, all labeled "rlmodel...". The current tab is active and shows the command "ros2 node list" followed by two nodes: "/image_publisher" and "/image_subscriber". The prompt "rlmodel@rlmodel: ~/ros2_ws\$ " is at the bottom.

cv package

cv_bridge coding

check topic usage: ros2 topic info /video_frames

The screenshot shows a terminal window with four tabs open, all labeled 'rlmodel@rlmodel'. The fourth tab is active and contains the following text:

```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic info /video_frames
Type: sensor_msgs/msg/Image
Publisher count: 1
Subscription count: 1
rlmodel@rlmodel:~/ros2_ws$
```



cv package

cv_bridge coding

sourcing: foxy

>check auto completion



Appendix



opencv

Open Source Computer Vision

link: <https://docs.opencv.org/3.4/index.html>

ENHANCED BY Google

OpenCV 3.4.20

Open Source Computer Vision

Main Page Related Pages Modules Namespaces Classes Files Examples Java documentation Search

OpenCV modules

- [Introduction](#)
- [OpenCV Tutorials](#)
- [OpenCV-Python Tutorials](#)
- [OpenCV.js Tutorials](#)
- [Tutorials for contrib modules](#)
- [Frequently Asked Questions](#)
- [Bibliography](#)
- Main modules:
 - core. [Core functionality](#)
 - imgproc. [Image Processing](#)
 - imgcodecs. [Image file reading and writing](#)
 - videoio. [Video I/O](#)
 - highgui. [High-level GUI](#)
 - video. [Video Analysis](#)
 - calib3d. [Camera Calibration and 3D Reconstruction](#)
 - features2d. [2D Features Framework](#)
 - objdetect. [Object Detection](#)
 - dnn. [Deep Neural Network module](#)
 - ml. [Machine Learning](#)



Genicam

intro

GenICam은 머신 비전 카메라용 일반 프로그래밍 인터페이스입니다.

이 표준의 목표는 사용자 애플리케이션 프로그래밍 인터페이스에서 산업용 카메라 인터페이스 기술을 분리하는 것입니다.

> GenICam은 유럽 머신 비전 협회에서 관리

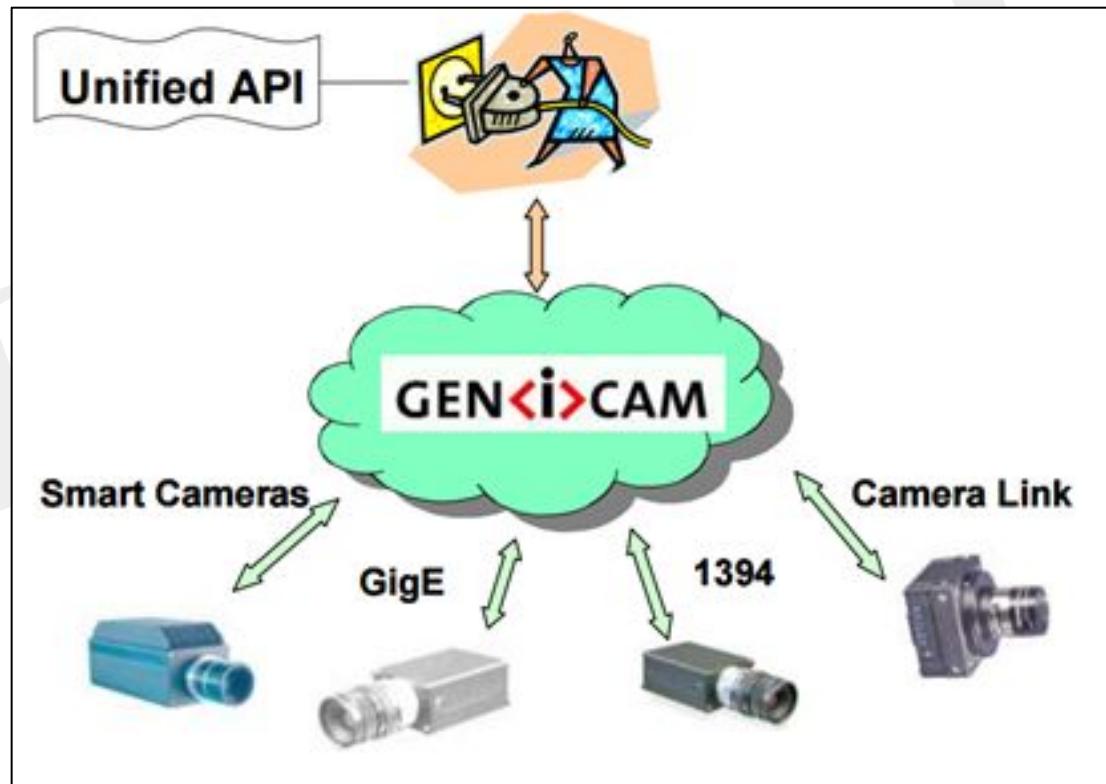
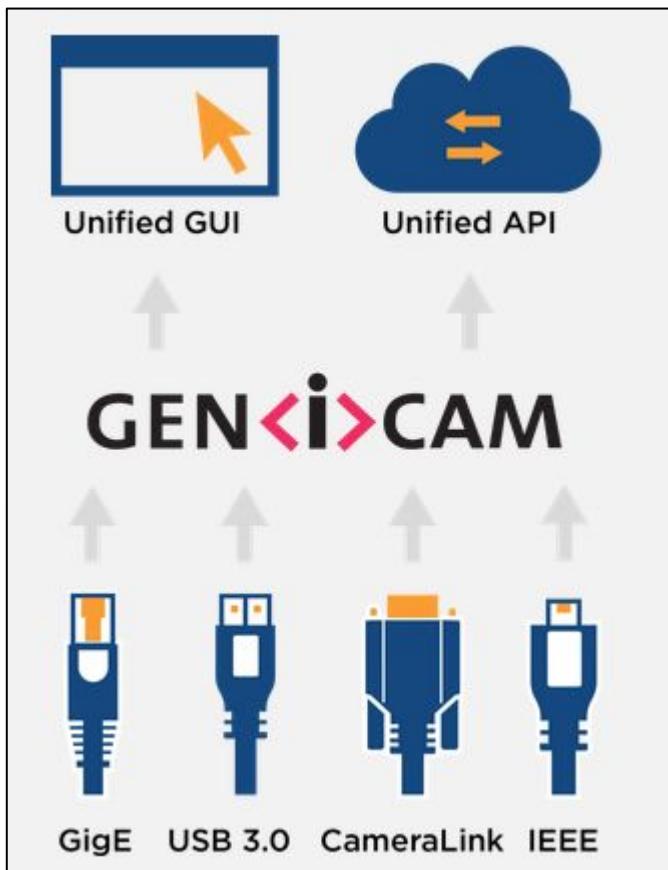
GenICam을 사용해야 하는 이유는?

GenICam은 범용 소프트웨어 인터페이스를 제공합니다. 이것은 GigE Vision, USB3 Vision, Camera Link® 및 IEEE 1394를 포함한 광범위한 표준 인터페이스를 위한 종단간 구성 인터페이스를 제공해 모든 카메라 종류 및 이미지 형식을 커버합니다. 이 방식은 카메라별 구성 필요 없이 GenICam 표준과 호환되는 카메라를 쉽게 연결할 수 있게 만들어줍니다.



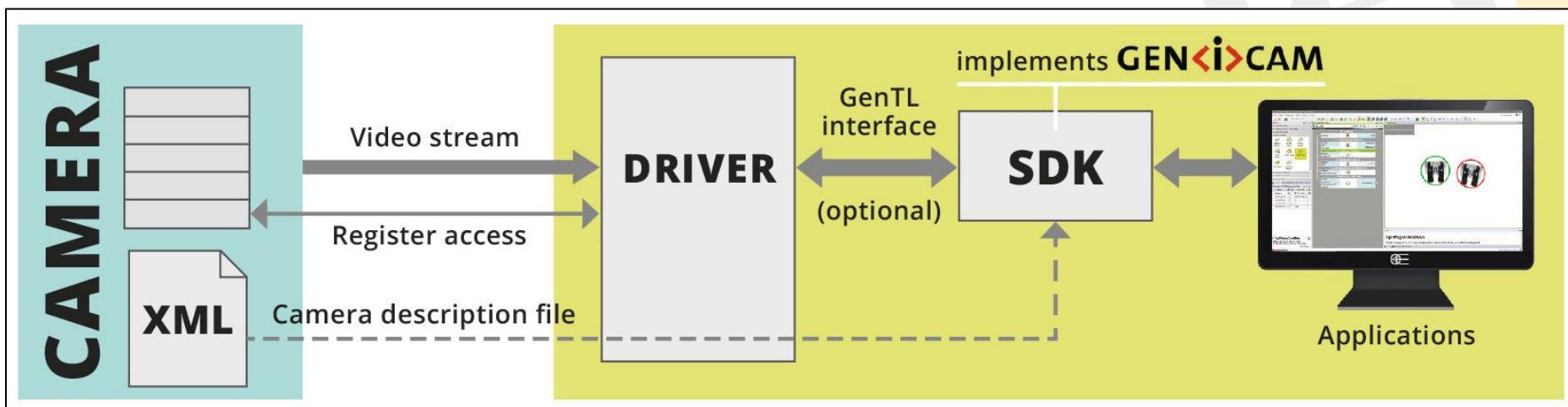
Genicam

Configuration



Genicam

Block diagram



video viewer

guvcview

명령어: \$ sudo apt-get install guvcview

