Creating Your First ROS2 Application

Ameer Hamzah (Robotics Engineer)

Mar 4, 2024

1 Creating your first ROS Application

Up until now, we have seen the paradigm behind a ROS project; however, we have still no clue on how to start a project from scrap. The most attentious readers may have figured out (from what we said about the source command in the first lecture) that ROS works with different workspaces which contain everything you need to run your code, and they would be right. In fact, a ROS project is exactly another workspace to source below the ros2 dependencies, once you do that, running your ROS project is no different from running a new TurtleSim node.

We will now see how we can build workspace and packages to organize our code, and then, how to write a simple publisher function. At the end of this part, you should be able to instruct your turtle to move in a particular way just by running your written code.

Let's get started!

NOTE: Make sure that ROS2 Humble is installed before following the steps given below.

1.1 Installing and Using Colcon to Build ROS2 Packages

In this part, we will be using the tool colcon to build our first ROS package. Let's install it before. If you're using Ubuntu, just run:

```
sudo apt install python3-colcon-common-extensions
```

With that done, we are ready to use colcon.

1.2 Creating a Workspace for Your First Project

Let's start by creating a single folder *ros2* for your workspace. Then create a folder named *src* in it. You can use the following command to do that in one step:

```
mkdir -p ros2/src
```

Move to your *src* folder using `cd ros2/src` and create a new ros2 package by running:

```
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

Here, we are also creating a simple node as an example. If you only want to create a package, you can do it by running:

```
ros2 pkg create --build-type ament_python [package_name]
```

In terminal, move to your workspace using `cd ..` from your *src* folder and run:

```
colcon build
```

This will build your workspace. Make sure that you have sourced the ros2 environment before running build command using:

```
source /opt/ros/humble/setup.bash
```

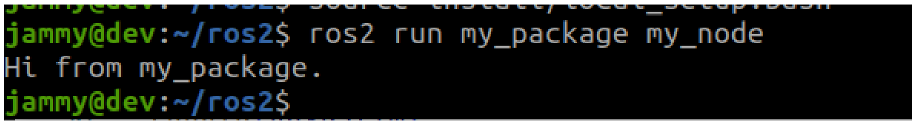
You have sucessfully built your package now. Let's source the newly built workspace.

```
source install/local_setup.bash
```

Now, let's run our node.

```
ros2 run my_package my_node
```

You should get this output in the terminal.



```
jammy@dev:~/ros2$ ros2 run my_package my_node
Hi from my_package.
jammy@dev:~/ros2$
```

Figure 1: Output of Node

Your *ros2* workspace folder is comprised of this content now:

```
ros2
├── build
├── install
├── log
└── src
    └── my_package
```

You will do most of your work in *src* folder. So, let's break down this folder to understand what's inside it.

```
src
└── my_package
    ├── my_package
    │   ├── __init.py__
    │   └── my_node.py
    ├── resource
    ├── test
    ├── package.xml
    ├── setup.cfg
    └── setup.py
```

Let's create a new node now. Go inside my_package folder in my_package and create a new file `test.py`. my_package folder has now this content after the creation of new file.

```
my_package
└── my_package
    ├── __init.py__
    ├── my_node.py
    └── test.py
```

Open `test.py` file and write the following code inside it.

```python
def main():
    print('I just created this new package.')


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

Listing 1: Script of Test Node

Save it and now we have to add entry points in the `setup.py` file so that we can get executable of this new python node.

```python
entry_points={
        'console_scripts': [
            'my_node = my_package.my_node:main',
            'test = my_package.test:main'
        ],
    },
```

Listing 2: Adding entry point for test node

Let's now build it after moving to workspace folder i-e *ros2* in this case

```
colcon build
```

This will build your workspace. Make sure that you have sourced the ros2 environment before running build command using:

```
source /opt/ros/humble/setup.bash
```

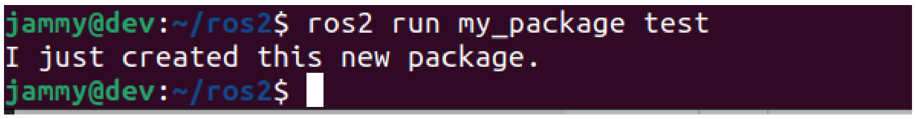
You have sucessfully built your package now. Let's source the newly built workspace.

```
source install/local_setup.bash
```

Now, let's run our node.

```
ros2 run my_package test.py
```

You should get this output in the terminal.



```
jammy@dev:~/ros2$ ros2 run my_package test
I just created this new package.
jammy@dev:~/ros2$
```

Figure 2: Output of Test Node

1.3 Creating a Launch File

To start multiple nodes at once, we use launch files. Here is an example of launch file written in python. In order to create a launch file, go inside the `my_package` folder and create a folder with name *launch*. Inside this folder, create a new python file with name `simple_launch.py`

```python
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='my_package',
            executable='my_node',
            name='node1'
        ),
        Node(
            package='my_package',
            executable='test',
            name='node2',
        ),

    ])
```

Listing 3: Launch File Script

To make this launch file work, you'll have to go to `setup.py` and modify the code as given below. Only add lines which are HASHED in the code below.

```
from setuptools import find_packages, setup
import os # add this line
from glob import glob # add this line

package_name = 'my_package'

setup(
    name=package_name,
    version='0.0.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch
    .[pxy][yma]*'))) # add this line
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='jammy',
    maintainer_email='Spellbreaker907@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'my_node = my_package.my_node:main',
            'test = my_package.test:main'
        ],
    },
)
```

Listing 4: Setup.py Modification for Launch File

Get back to your workspace and run `colcon build`. Source newly built workspace and run:

```
ros2 launch my_package simple_launch.py
```

Both nodes should run smoothly and you can check the data of both nodes by going to log folder as mentioned in the message that you will get once you run the launch file.

References

[1] Open Robotics. "*ROS2 Humble Official Documentation*"

[2] Ameer Hamzah "*Proceedings of Aerial Robotics Lab, SINES, NUST, Pakistan*". 2024.