04. ROS 2 Launch, Param, Bag

Lecture

ROS 2 Launch

- Launch multiple nodes
- Set arguments
- Monitor running nodes
- React on changes in the state of nodes
- Python, XML and YAML file formats

Usage

```
ros2 launch package_name file.launch
ros2 launch irob_robot dvrk_server.launch arm_typ:=PSM1
```

ROS 2 Parameters

- Configure nodes at startup or during runtime without changing the code
- Associated with individual nodes
- Consists of: key, value, descriptor
- Available data types: bool, int64, float64, string, byte[], bool[], int64[], float64[] or string[].
- Useful command: `ros2 param`

ROS 2 Bag

- Record and playback ROS topics
- Command line tool
- API for C++ and Python

```
ros2 bag record -o <file_name> <topic_name>
ros2 bag record --all
ros2 bag info <filename.bag>
ros2 bag play <filename.bag>
```

Practice

1: Launch Turtlesim Mimic

1. Create the `launch` folder inside the `ros2_course` package, where the launch files can be stored:

```
cd ~/ros2_ws/src/ros2_course
mkdir launch
```

2. Create the `turtlesim_mimic_launch.py` file in the new `launch` folder a with the following content:

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

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
```

```
    ),
    Node(
        package='turtlesim',
        executable='mimic',
        name='mimic',
        remappings=[
            ('/input/pose', '/turtlesim1/turtle1/pose'),
            ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
        ]
    )
])
```

3. Add the followings to the `setup.py` file:

```
import os
from glob import glob

# ...

data_files=[
    ('share/ament_index/resource_index/packages',
        ['resource/' + package_name]),
    ('share/' + package_name, ['package.xml']),
    # Include all launch files.
    (os.path.join('share', package_name),
        glob('launch/*launch.[pxy][yma]*'))
],
```

4. Add the `ros2launch` dependency to the `package.xml` file:

```
<exec_depend>ros2launch</exec_depend>
```

5. Build the workspace:

```
cd ros2_ws
colcon build --symlink-install
```

6. Launch the launch file:

```
ros2 launch ros2_course turtlesim_mimic_launch.py
```

7. Publish to the topic from the command line, in a new terminal window:

```
ros2 topic pub -r 1 /turtlesim1/turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x:
2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: -1.8}}"
```

8. Let's examine the operation of the system using `rqt_gui` :

```
ros2 run rqt_gui rqt_gui
```

2: Launch Turtlesim Goto

1. Let's make a copy of `turtlesim_mimic_launch.py` from a file named
   `turtlesim_controller_launch.py` .

2. Add the `turtlesim_controller` node written in the previous lesson to the launch
   file. The turtle to be controlled can be set using `namespace` or `remappings` .

3. Build the workspace:

```
cd ros2_ws
colcon build --symlink-install
```

4. launch the new launch file:

```
ros2 launch ros2_course turtlesim_controller_launch.py
```

3: Turtlesim controller params

1. Modify `turtlesim_controller` so that the linear velocity and angular velocity. is
   adjustable via ROS parameters. API example for parameters:

```
import rclpy
import rclpy.node
```

```
class MinimalParam(rclpy.node.Node):
    def __init__(self):
        super().__init__('minimal_param_node')
        # Declare parameter named 'my_parameter' and
        # set default value to 'world'
        self.declare_parameter('my_parameter', 'world')
        self.timer = self.create_timer(1, self.timer_callback)

    def timer_callback(self):
        my_param =
self.get_parameter('my_parameter').get_parameter_value().string_value
        self.get_logger().info('Hello %s!' % my_param)


def main():
    rclpy.init()
    node = MinimalParam()
    rclpy.spin(node)

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

2. Let's run `turtlesim_controller.py` using the previously written launch file. Let's list the parameters.

```
ros2 launch ros2_course turtlesim_controller_launch.py
```

```
ros2 param list
```

3. Change the speed and angular velocity parameters from the command line using the following command:

```
ros2 param set <NODE_NAME> <PARAM_NAME> <NEW_VALUE>
ros2 param set controller speed 100.0
```

4: Turtlesim controller launch and substitutions

1. Let's make a copy of `turtlesim_controller_launch.py` as `turtlesim_controller_param_launch.py` . Modify the new launch file based on the example below so that the velocity and angular velocity parameters of the launch can be specified as file arguments.

```python
from launch_ros.actions import Node
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, ExecuteProcess, TimerAction
from launch.conditions import IfCondition
from launch.substitutions import LaunchConfiguration, PythonExpression

def generate_launch_description():
    turtlesim_ns_value = LaunchConfiguration('turtlesim_ns')
    use_provided_red_value = LaunchConfiguration('use_provided_red')
    new_background_r_value = LaunchConfiguration('new_background_r')
    background_g_value = LaunchConfiguration('background_g')
    background_b_value = LaunchConfiguration('background_b')

    turtlesim_ns_launch_arg = DeclareLaunchArgument(
        'turtlesim_ns',
        default_value='turtlesim1',
        description='Namespace for turtle 1'
    )
    use_provided_red_launch_arg = DeclareLaunchArgument(
        'use_provided_red',
        default_value='False'
    )
    new_background_r_launch_arg = DeclareLaunchArgument(
        'new_background_r',
        default_value='200'
    )
    background_g_launch_arg = DeclareLaunchArgument(
        'background_g',
        default_value='100'
    )
    background_b_launch_arg = DeclareLaunchArgument(
        'background_b',
        default_value='100'
    )

    turtlesim_node = Node(
        package='turtlesim',
        namespace=turtlesim_ns_value,
        executable='turtlesim_node',
        name='sim',
        parameters=[{
            'background_g': background_g_value,
            'background_b': background_b_value,
        }]
    )
    spawn_turtle = ExecuteProcess(
        cmd=[[
            'ros2 service call ',
            turtlesim_ns,
            '/spawn ',
            'turtlesim/srv/Spawn ',
            '"{x: 2, y: 2, theta: 0.2}"'
        ]],
        shell=True
    )
    change_background_r = ExecuteProcess(
        cmd=[[
```

```
        'ros2 param set ',
        turtlesim_ns,
        '/sim background_r ',
        '120'
      ]],
      shell=True
  )
  change_background_r_conditioned = ExecuteProcess(
      condition=IfCondition(
          PythonExpression([
              new_background_r_value,
              ' == 200',
              ' and ',
              use_provided_red
          ])
      ),
      cmd=[[
          'ros2 param set ',
          turtlesim_ns_value,
          '/sim background_r ',
          new_background_r
      ]],
      shell=True
  )

  return LaunchDescription([
      turtlesim_ns_launch_arg,
      use_provided_red_launch_arg,
      new_background_r_launch_arg,
      turtlesim_node,
      spawn_turtle,
      change_background_r,
      TimerAction(
          period=2.0,
          actions=[change_background_r_conditioned],
      )
  ])
```

2. Build the workspace and run `turtlesim_controller_param_launch.py` :

```
cd ros2_ws
colcon build --symlink-install
ros2 launch ros2_course turtlesim_controller_param_launch.py
```

3. Let's list the arguments of the new launch file:

```
ros2 launch ros2_course turtlesim_controller_param_launch.py --show-args
```

4. Run the launch file by setting the arguments:

```
ros2 launch ros2_course turtlesim_controller_param_launch.py speed:=100.0
omega:=60.0
```

5. Using the above example, let's set the background color also using command line argument(s).

5: Rosbag

1. While the program implemented in the previous exercise is running, record the contents of the topics in a rosbag file.

   ```
   ros2 bag record --all
   ```

   > ■ **Syntax**
   >
   > The filename and the topics to record can also be set, e.g.:
   >
   > ```
   > ros2 bag record -o turtle_bagfile_1 /turtle1/cmd_vel /turtle1/pose
   > ```

2. Use the following command to query the properties of the bag file:

   ```
   ros2 bag info <PATH_TO_BAGFILE>
   ```

3. Play back the bag file and plot the `pose/x` value of one of the turtles on a graph using `rqt_gui`.

   ```
   ros2 bag info <PATH_TO_BAGFILE>
   ```

   ```
   ros2 run rqt_gui rqt_gui
   ```

Useful links

- ROS 2 Launch Tutorial
- ROS 2 Parameters
- Using ROS 2 parameters in a Class
- ROS 2 Bag