TUGAS UTS ROBOTIKA

Diajukan untuk memenuhi tugas pengganti Ujian Tengah Semester (UTS)

pada mata kuliah Robotika

Disusun oleh :

Wening Alfina Rosunika

1103204017

PROGRAM STUDI TEKNIK KOMPUTER

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

2023

```
import os

import pathlib

from launch.substitutions import LaunchConfiguration

from launch.actions import DeclareLaunchArgument

from launch.substitutions.path_join_substitution import PathJoinSubstitution

from launch import LaunchDescription

from launch_ros.actions import Node

import launch

from ament_index_python.packages import get_package_share_directory,
get_packages_with_prefixes

from launch.launch_description_sources import PythonLaunchDescriptionSource

from launch.actions import IncludeLaunchDescription

from webots_ros2_driver.webots_launcher import WebotsLauncher

from webots_ros2_driver.utils import controller_url_prefix



def get_ros2_nodes(*args):

    package_dir = get_package_share_directory('webots_ros2_turtlebot')

    use_nav = LaunchConfiguration('nav', default=False)

    use_slam = LaunchConfiguration('slam', default=False)

    robot_description = pathlib.Path(os.path.join(package_dir, 'resource',
'turtlebot_webots.urdf')).read_text()

    ros2_control_params = os.path.join(package_dir, 'resource', 'ros2control.yml')

    nav2_params = os.path.join(package_dir, 'resource', 'nav2_params.yaml')

    nav2_map = os.path.join(package_dir, 'resource', 'turtlebot3_burger_example_map.yaml')

    use_sim_time = LaunchConfiguration('use_sim_time', default=True)


    # TODO: Revert once the https://github.com/ros-controls/ros2_control/pull/444 PR gets into
the release

    controller_manager_timeout = ['--controller-manager-timeout', '50']

    controller_manager_prefix = 'python.exe' if os.name == 'nt' else ''
```

```
use_deprecated_spawner_py = 'ROS_DISTRO' in os.environ and os.environ['ROS_DISTRO'] ==
'foxy'


    diffdrive_controller_spawner = Node(
        package='controller_manager',
        executable='spawner' if not use_deprecated_spawner_py else 'spawner.py',
        output='screen',
        prefix=controller_manager_prefix,
        arguments=['diffdrive_controller'] + controller_manager_timeout,
    )


    joint_state_broadcaster_spawner = Node(
        package='controller_manager',
        executable='spawner' if not use_deprecated_spawner_py else 'spawner.py',
        output='screen',
        prefix=controller_manager_prefix,
        arguments=['joint_state_broadcaster'] + controller_manager_timeout,
    )


    mappings = [('/diffdrive_controller/cmd_vel_unstamped', '/cmd_vel')]
    if 'ROS_DISTRO' in os.environ and os.environ['ROS_DISTRO'] in ['humble', 'rolling']:
        mappings.append(('/diffdrive_controller/odom', '/odom'))
turtlebot_driver = Node(
        package='webots_ros2_driver',
        executable='driver',
        output='screen',
        additional_env={'WEBOTS_CONTROLLER_URL': controller_url_prefix() + 'TurtleBot3Burger'},
        parameters=[
            {'robot_description': robot_description,
             'use_sim_time': use_sim_time,
             'set_robot_state_publisher': True},
            ros2_control_params
```

```
],
        remappings=mappings
    )

    robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        output='screen',
        parameters=[{
            'robot_description': '<robot name=""><link name=""/></robot>'
        }],
    )

    footprint_publisher = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        output='screen',
        arguments=['0', '0', '0', '0', '0', '0', 'base_link', 'base_footprint'],
    )

    nav_nodes = []
    # Navigation
    os.environ['TURTLEBOT3_MODEL'] = 'burger'
    if 'turtlebot3_navigation2' in get_packages_with_prefixes():
        turtlebot_navigation = IncludeLaunchDescription(
            PythonLaunchDescriptionSource(os.path.join(
                get_package_share_directory('turtlebot3_navigation2'), 'launch', 'navigation2.launch.py')),
            launch_arguments=[
                ('map', nav2_map),
                ('params_file', nav2_params),
                ('use_sim_time', use_sim_time),
            ],
            condition=launch.conditions.IfCondition(use_nav))
```

```
    nav_nodes.append(turtlebot_navigation)


  # SLAM
  if 'turtlebot3_cartographer' in get_packages_with_prefixes():
    turtlebot_slam = IncludeLaunchDescription(
      PythonLaunchDescriptionSource(os.path.join(
        get_package_share_directory('turtlebot3_cartographer'), 'launch',
'cartographer.launch.py')),
      launch_arguments=[
        ('use_sim_time', use_sim_time),
      ],
      condition=launch.conditions.IfCondition(use_slam))
    nav_nodes.append(turtlebot_slam)


  # Wait for the simulation to be ready to start navigation nodes
  nav_handler = []
  if nav_nodes:
    nav_handler.append(
      launch.actions.RegisterEventHandler(
        event_handler=launch.event_handlers.OnProcessExit(
          target_action=diffdrive_controller_spawner,
          on_exit=nav_nodes
        )
      )
    )


  return [
    joint_state_broadcaster_spawner,
    diffdrive_controller_spawner,
    robot_state_publisher,
    turtlebot_driver,
    footprint_publisher,
  ] + nav_handler
```

```
def generate_launch_description():
    package_dir = get_package_share_directory('webots_ros2_turtlebot')
    world = LaunchConfiguration('world')
    mode = LaunchConfiguration('mode')

    webots = WebotsLauncher(
        world=PathJoinSubstitution([package_dir, 'worlds', world]),
        mode=mode,
        ros2_supervisor=True
    )

    # The following line is important!
    # This event handler respawns the ROS 2 nodes on simulation reset (supervisor process ends).
    reset_handler = launch.actions.RegisterEventHandler(
        event_handler=launch.event_handlers.OnProcessExit(
            target_action=webots._supervisor,
            on_exit=get_ros2_nodes,
        )
    )

    return LaunchDescription([
        DeclareLaunchArgument(
            'world',
            default_value='turtlebot3_burger_example.wbt',
            description='Choose one of the world files from `/webots_ros2_turtlebot/world`
directory'
        ),
```

```
    DeclareLaunchArgument(
      'mode',
      default_value='realtime',
      description='Webots startup mode'
    ),
    webots,
    webots._supervisor,

    # This action will kill all nodes once the Webots simulation has exited
    launch.actions.RegisterEventHandler(
      event_handler=launch.event_handlers.OnProcessExit(
        target_action=webots,
        on_exit=[
          launch.actions.UnregisterEventHandler(
            event_handler=reset_handler.event_handler
          ),
          launch.actions.EmitEvent(event=launch.events.Shutdown())
        ],
      )
    ),

    # Add the reset event handler
    reset_handler
  ] + get_ros2_nodes())
```

Pendahuluan :

Webots TurtleBot adalah model simulasi dari robot mobile bernama TurtleBot yang dikembangkan oleh Willow Garage. TurtleBot didesain untuk mempermudah pengembangan aplikasi robotika, seperti navigasi, pengenalan objek, dan tugas-tugas lainnya. Dalam simulasi Webots, TurtleBot diwakili oleh model yang sama dengan spesifikasi dan perilaku yang sama dengan aslinya. Webots menyediakan lingkungan simulasi 3D yang lengkap, yang memungkinkan pengguna untuk menguji dan mengembangkan program robotika secara virtual sebelum diimplementasikan pada robot fisik.

Analisis Code :

Fungsi generate_launch_description membuat file peluncuran yang meluncurkan node WebotsLauncher untuk memulai simulasi Webots, dan kemudian memanggil get_ros2_nodes untuk memulai node ROS 2 yang mengontrol dan mengarahkan robot.

Fungsi get_ros2_nodes membuat dan mengembalikan daftar node ROS 2 yang diluncurkan menggunakan paket peluncuran. Node-node ini termasuk diffdrive_controller_spawner, joint_state_broadcaster_spawner, turtlebot_driver, robot_state_publisher, dan footprint_publisher.

Node diffdrive_controller_spawner dan joint_state_broadcaster_spawner digunakan untuk memunculkan kontroler untuk roda dan status sendi robot, masing-masing. Node turtlebot_driver adalah node pengendali yang berinteraksi dengan Webots untuk mengontrol motor robot dan membaca data sensor. Node robot_state_publisher mempublikasikan status sendi robot dan transformasi antara frame-link robot. Node footprint_publisher mempublikasikan transformasi statis antara frame base_link robot dan base_footprint.

Fungsi get_ros2_nodes juga meluncurkan node untuk navigasi dan pemetaan, jika diaktifkan oleh konfigurasi peluncuran use_nav dan use_slam. Node-node ini termasuk turtlebot_navigation dan turtlebot_slam, yang meluncurkan node-node navigasi dan pemetaan dari paket turtlebot3_navigation2 dan turtlebot3_cartographer, masing-masing.

Daftar nav_handler berisi pengolah acara yang menunggu node diffdrive_controller_spawner keluar sebelum memulai node navigasi. Ini dilakukan untuk memastikan bahwa kontroler robot sudah siap sebelum node navigasi mulai mempublikasikan perintah.

Secara keseluruhan, file peluncuran ini dapat digunakan untuk memulai simulasi robot TurtleBot3 di Webots, dan menjalankan berbagai node ROS 2 untuk kontrol dan navigasi.