ROS

Robot Operating System

Course of

Robot Programming with ROS 2

Day 2

5. Exercise: Bump&GO BT

ikerlan

Universidad Rey Juan Carlos

Intelligent Robotics Lab

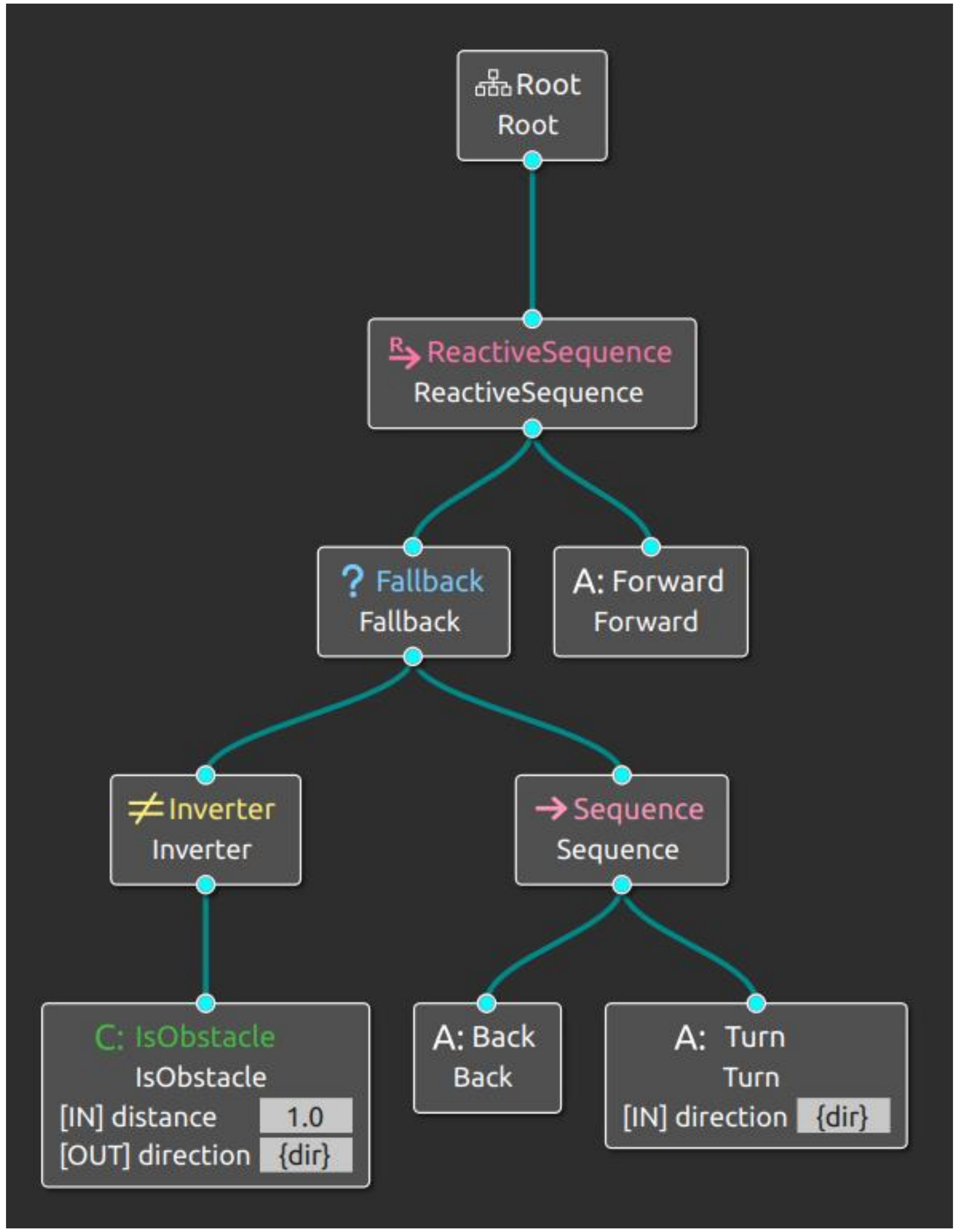
Objectives

- Bump and go behavior using BT
- When the obstacle is on the right side of the robot, turn left, and vice versa
- Use ports to communicate BT nodes

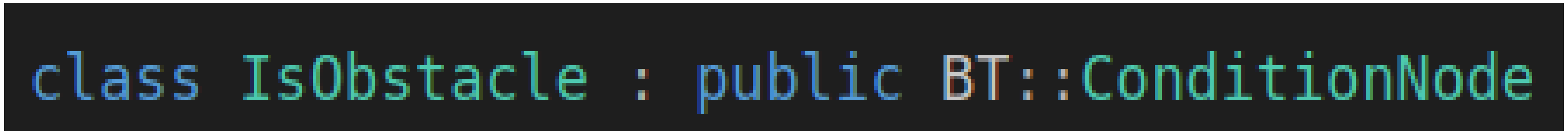
Steps

1. Design the BT in XML using *Groot*
2. Implement BT nodes
3. Create an executable that starts the BT
4. Compile and launch it!

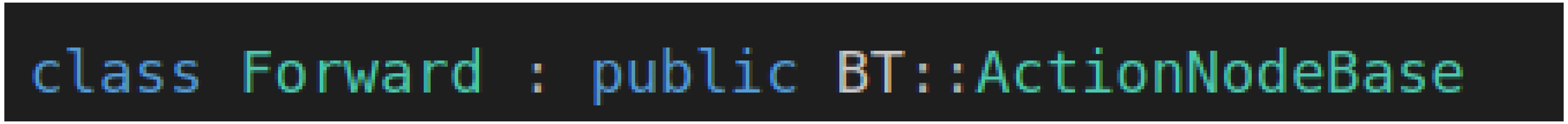
Step 1: XML


Root
Root
ReactiveSequence
ReactiveSequence
? Fallback
Fallback
A: Forward
Forward
Inverter
Inverter
Sequence
Sequence
C: IsObstacle
IsObstacle
[IN] distance 1.0
[OUT] direction {dir}
A: Back
Back
A: Turn
Turn
[IN] direction {dir}

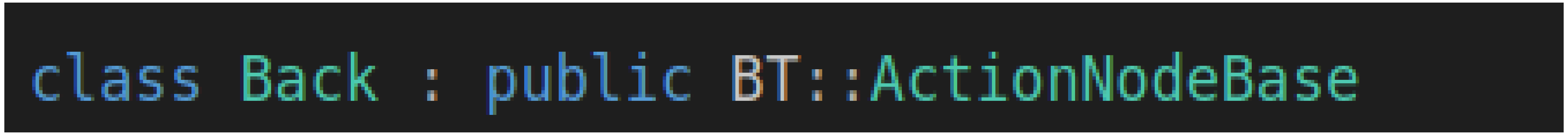
Step 2: BT nodes



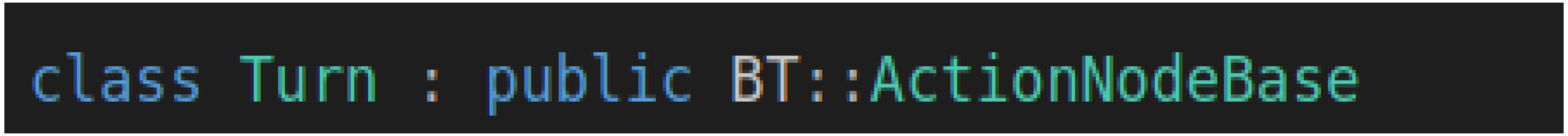
```
class IsObstacle : public BT::ConditionNode
```



```
class Forward : public BT::ActionNodeBase
```

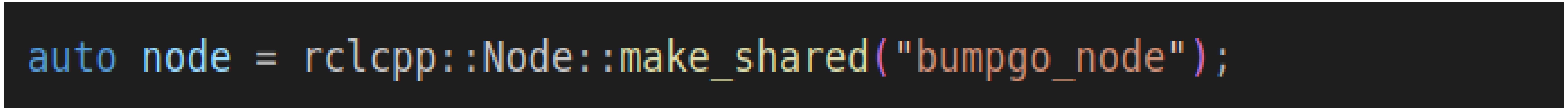


```
class Back : public BT::ActionNodeBase
```

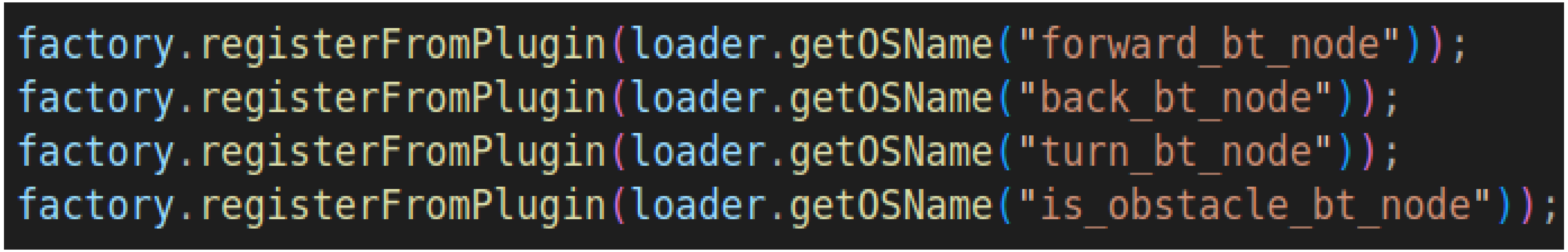


```
class Turn : public BT::ActionNodeBase
```

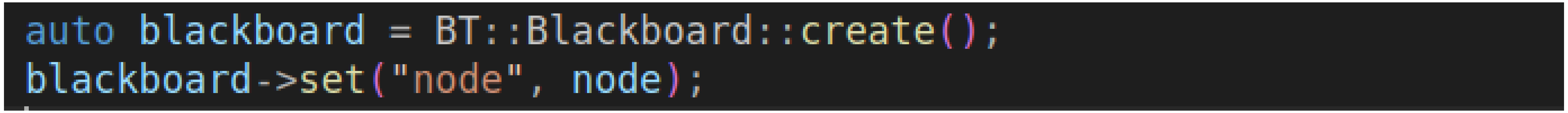
Step 3: Main



```
auto node = rclcpp::Node::make_shared("bumpgo_node");
```

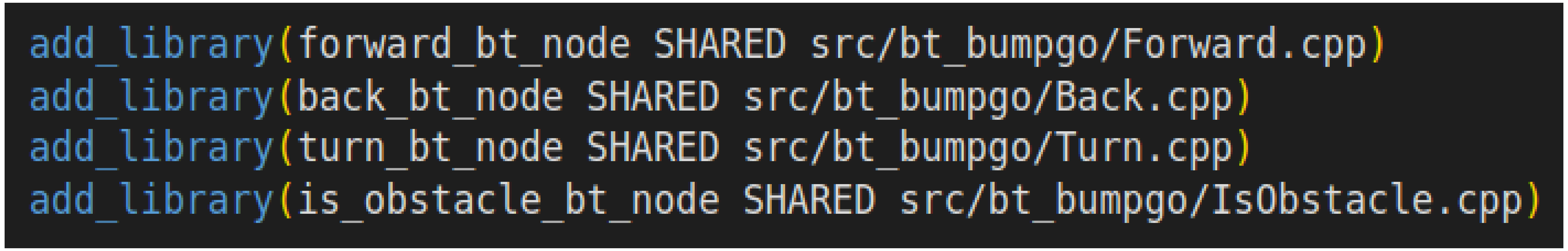


```
factory.registerFromPlugin(loader.getOSName("forward_bt_node"));
factory.registerFromPlugin(loader.getOSName("back_bt_node"));
factory.registerFromPlugin(loader.getOSName("turn_bt_node"));
factory.registerFromPlugin(loader.getOSName("is_obstacle_bt_node"));
```

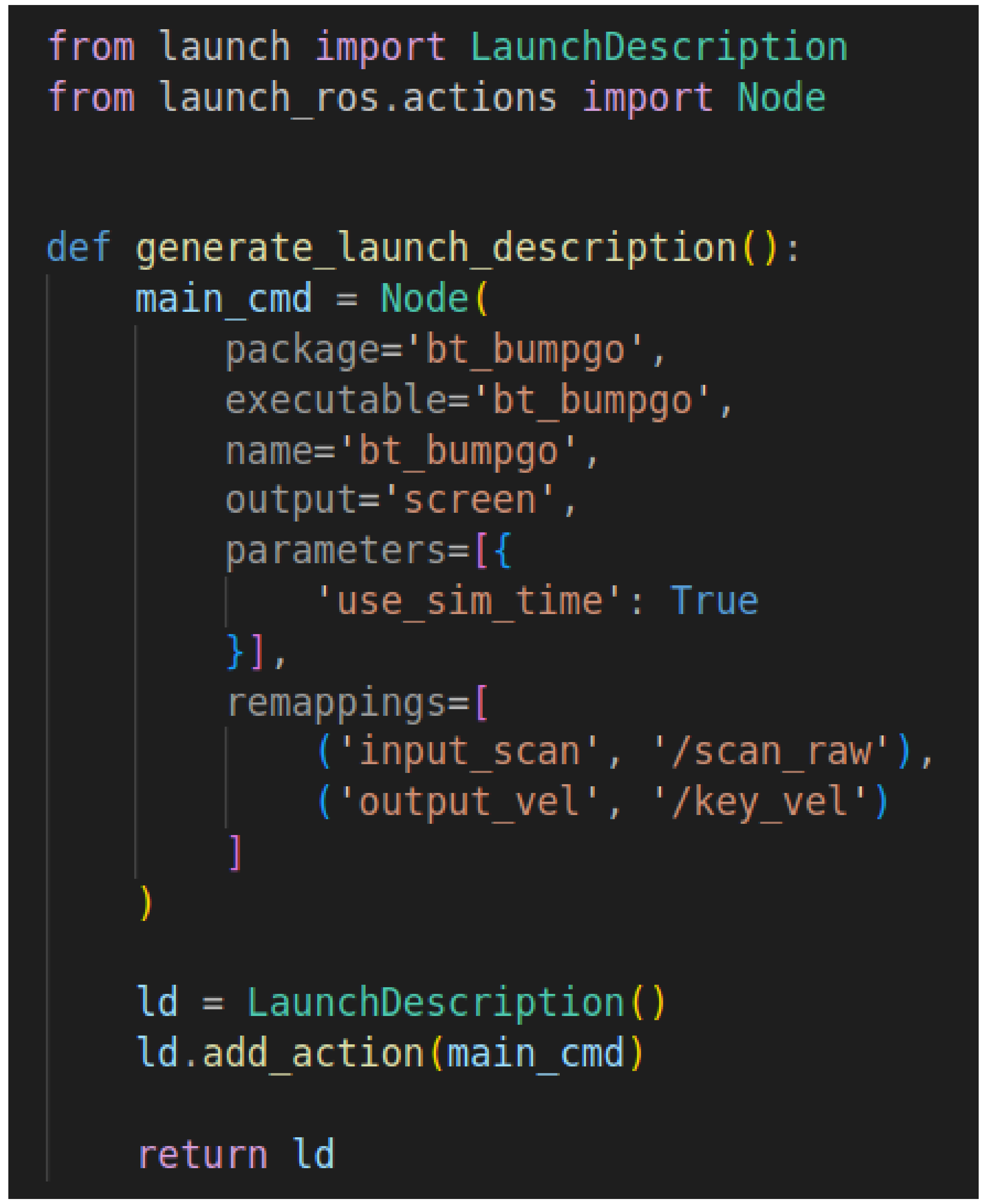


```
auto blackboard = BT::Blackboard::create();
blackboard->set("node", node);
```

Step 4: CMake & launcher



```
add_library(forward_bt_node SHARED src/bt_bumpgo/Forward.cpp)
add_library(back_bt_node SHARED src/bt_bumpgo/Back.cpp)
add_library(turn_bt_node SHARED src/bt_bumpgo/Turn.cpp)
add_library(is_obstacle_bt_node SHARED src/bt_bumpgo/IsObstacle.cpp)
```



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


def generate_launch_description():
    main_cmd = Node(
        package='bt_bumpgo',
        executable='bt_bumpgo',
        name='bt_bumpgo',
        output='screen',
        parameters=[{
            'use_sim_time': True
        }],
        remappings=[
            ('input_scan', '/scan_raw'),
            ('output_vel', '/key_vel')
        ]
    )

    ld = LaunchDescription()
    ld.add_action(main_cmd)

    return ld
```