Boot Camp Part 1: Task 1: Send a Command to the Robot

Send a Command to the Robot

The robot receives a duty cycle signal on the topic **/motor/duty_cycles**

Your task is to send a command on that topic such that the robot moves.

Before writing your node as described below, you can try and publish directly in the terminal

```
ros2 topic pub /motor/duty_cycles robp_interfaces/msg/DutyCycles '{duty_cycle_left: 1.0, duty_cy
cle_right: 1.0}' -r 10
```

Remember to also run

```
ros2 launch robp_boot_camp_launch boot_camp_part1_launch.xml
```

in a different terminal/tab.

You should see the same behavior when you run your open loop controller node (if implemented correctly and you give the same left/right duty cycle).

Create an Open Loop Controller Node

Create a package called **controller** and a node called **open_loop_controller** in it

```
cd ~/dd2419_ws/src
ros2 pkg create controller --build-type ament_python --node-name open_loop_controller --dependen
cies geometry_msgs rclpy robp_interfaces robp_boot_camp_interfaces --license MIT
```

Make sure all the dependencies are installed

```
cd ~/dd2419_ws
rosdep install --from-paths src -y --ignore-src --as-root pip:false
```

Build the new package

```
cd ~/dd2419_ws
colcon build --symlink-install
```

Source the workspace so the new package is included in your path

The **open_loop_controller** node should publish full power commands (**duty cycle 1.0**) to the DC motors through the **/motor/duty_cycles** topic for the wheels **every 100 milliseconds (10 Hz)**. Edit the file `~/dd2419_ws/src/controller/controller/open_loop_controller.py` and take a look at the **ROS 2 Humble publisher/subscriber tutorial (https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html)** . The following is a skeleton of the file

```
#!/usr/bin/env python

import rclpy
from rclpy.node import Node

from robp_interfaces.msg import DutyCycles


class OpenLoopController(Node):

    def __init__(self):
        super().__init__('open_loop_controller')

        # TODO: Implement

    # TODO: Implement


def main():
    rclpy.init()
    node = OpenLoopController()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass

    rclpy.shutdown()


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

When you have implemented your open loop controller you can run it

```
ros2 run controller open_loop_controller
```

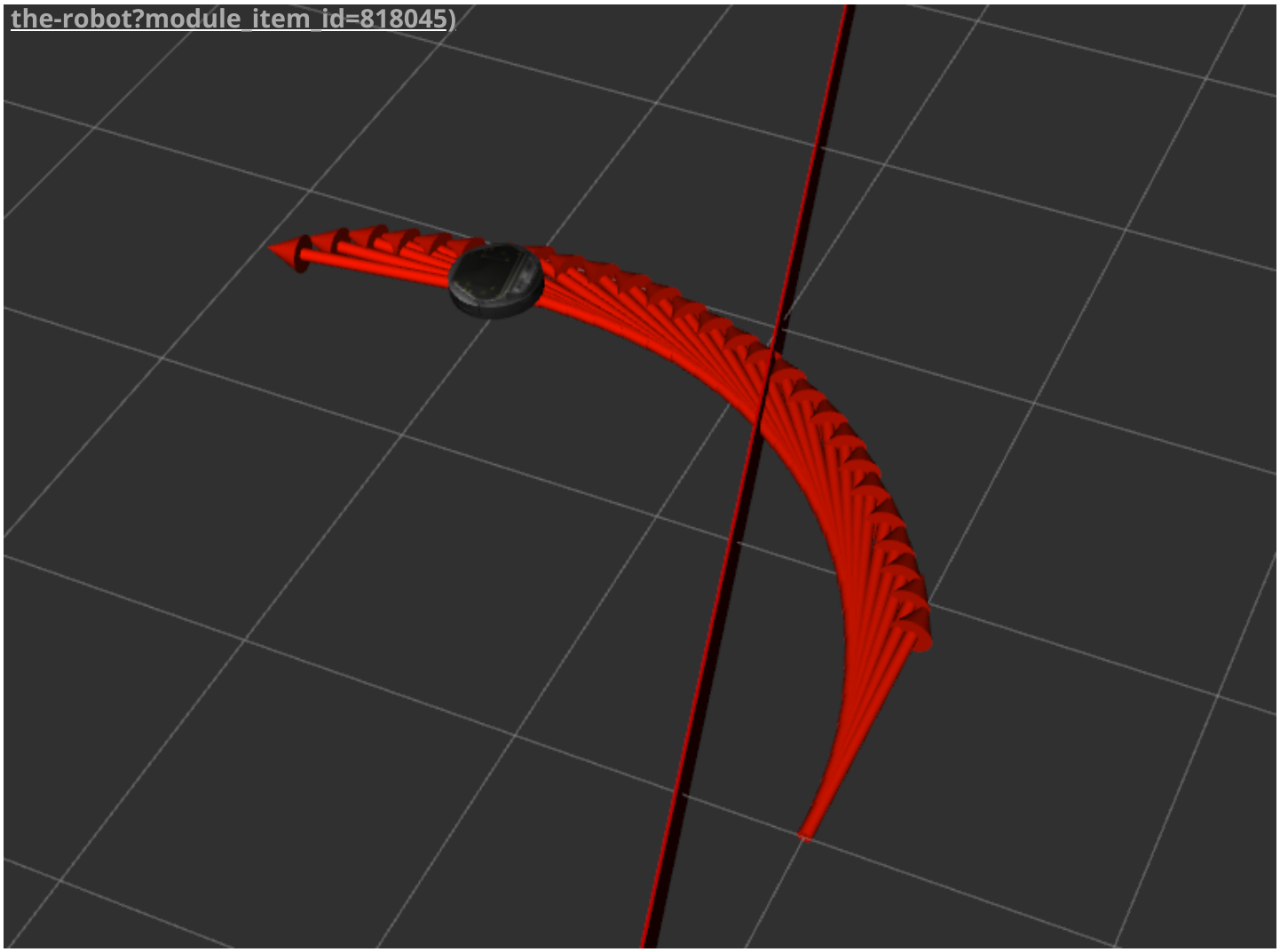
Remember to also run

```
ros2 launch robp_boot_camp_launch boot_camp_part1_launch.xml
```

in a different terminal/tab.

You will notice that even though you are sending the same power to both motors the robot drifts towards the left because the left motor is actually slower than the right one.

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-task-1-send-a-command-to-the-robot?module_item_id=818045)



This will also happen with real robots, given that two DC motors will never have exactly the same response. This is why we need some form of feedback on the output angular velocity/position of the motors to regulate the power that we send to the motors so that they rotate at the speed that we desire.

Play around with your open-loop controller node and try to send different **duty cycle** values (between [-1.0, 1.0]) to the motors to see what happens to the trajectory of the robot.