


Lab 3

Due Feb 23 by 7:30pm **Points** 40 **Submitting** a file upload
File Types zip, tgz, and bz2 **Attempts** 1 **Allowed Attempts** 1
Available until Feb 26 at 7:30pm


This assignment was locked Feb 26 at 7:30pm.

In this assignment you will use both turtlesim and the Gazebo simulator with a Pioneer P2 DX robot. P2 DX is a common (albeit old) mobile robot used both for learning robotics and research. If you are interested, [here](https://people.cs.ksu.edu/~dag/Activmedia/p2opman4.pdf)  (<https://people.cs.ksu.edu/~dag/Activmedia/p2opman4.pdf>) you can find more details, though they are not necessary to solve this assignment. Also note that if you were to use ROS to control a real world P2 DX, you would use exactly the same methods used to control it inside Gazebo. So, from a programming perspective you are facing no simplified assumptions.

The overall idea is the following:

- You start the nodes `turtlesim_node` and `turtle_teleop_key` to control the turtle with the keyboard -- this is something you have already done in the past.
- You start Gazebo with the simulated P2 DX robot (see details below on how to do that). The P2 DX robot can be controlled with the same velocity commands used to control the turtle.
- Your task is to write a node that subscribes to the pose topic publishing the velocity commands received by the turtle and passes those commands to control the P2 AT robot. So, once you use the keyboard to move the turtle, you will see the P2 DX doing the same motions because your node functions as a bridge between the two.

Here are the details to setup your working environment (seems a lot, but it is very simple and you have done most of this already in the past).

1. Make sure you get the latest version of the repository <https://github.com/stefanocarpin/MRTP>  (<https://github.com/stefanocarpin/MRTP>). **This is important. If you do not get the latest version, the following instructions will not work.** If you use the git command line interface you just need to run `git pull`, whereas if you downloaded the code as a zip file you have to erase the previous version and download it again. If you are using the lab machines it is advised that you put the MRTP repository in `/var/tmp` otherwise it will take a very long time to compile (see bullet point #4 in lab 0).
2. Build the workspace in the MRTP repository with `colcon build.`
3. From a separate shell, start `turtlesim_node` as follows:

```
ros2 run turtlesim turtlesim_node
```

4. From a separate shell, start `turtle_teleop_key` as follows:

```
ros2 run turtlesim turtle_teleop_key
```

5. Optional: to test that everything is working correctly, try moving the turtle with the keyboard.

6. From a separate shell, source the overlay for the MRTTP workspace and start the Gazebo simulation as follows:

```
ros2 launch gazeboenvs gazebo_p2dx.launch.py
```

Note: the first time you launch this environment Gazebo needs to download some files from the web and this may take some time. However, these will be cached and subsequent executions will be faster.

7. If you now run `ros2 topic list` you will see that the the P2 DX robot accepts velocity commands on a topic called `/p2dx/cmd_vel`

Your task:

- Either reuse the CSE180 workspace you used in previous weeks, or create one from scratch, also called CSE180.
- Inside CSE180, create a package called `lab3`. This package will contain the solution to this lab.
- Write a node `replicate` that subscribes to `/turtle1/pose`, extracts the velocity of the turtle and publishes the velocity commands to `/p2dx/cmd_vel`. This node is both a publisher and a subscriber.
Important: use `ros2 interface show` to determine the structure of messages sent on the topic `/turtle1/pose`. You will see that those messages include both the pose (x,y, theta, that we used last week), as well as the translational and rotational velocity of the turtle. All you need to do is retrieve those values and pass them to the P2 DX robot.
- Update the manifest file and `CMakeLists.txt` as needed.

Notes:

1. even if you solve this exercise correctly, you may observe that the P2 DX robot does not necessarily follow exactly the same motion as the turtle. This is not a problem, but rather a consequence of different physical constants between the two (mass, friction, etc.). As long as your code correctly passes data to the P2 DX robot, you will get full credit.
2. You must subscribe to `/turtle1/pose`, not to `/turtle1/cmd_vel`. If you get the velocity commands from `/turtle1/cmd_vel`, it will be considered an error.
3. You might run into a turtlesim bug where *translational* velocity data coming from the `/turtle1/pose` topic is always positive (and thus, the down-key doesn't make the P2 DX sim-bot move backwards). This is fine, as long as the program logic is correct, you will receive full marks for that portion of the lab assignment.

Submission: submit your entire workspace as a single zipped file. Before zipping the file, remove the folders build, install, log, otherwise your file will be way too big. You can zip your workspace with any

program generating the file formats listed below (zip, tgz, bz2). No other formats will be accepted. If you do not follow these instructions we will apply a 10% penalty to your grade.

Administrative notes:

- This assignment **must** be solved individually. If we determine you have copied from other students or from the web without proper attribution, all involved parties will receive a 0 for the assignment, and will be reported as per the CSE academic honesty policy discussed during the first lecture with all the associated consequences.
- If you wish, is ok to include the packages from previous weeks in your submission, and we will not consider it during grading. However, whatever you submit must be correctly built using `colcon build`. Please doublecheck before submitting, and if previous weeks' lab causes problems, remove them from this week's submission.
- This assignment can be submitted only once. Please check your solution carefully before hitting "Submit".
- Pay attention to the posted deadlines, as they will be enforced by CatCourses. This assignment is due seven days after it is released to your lab session. Assignments marked "late" with respect to the due deadline will receive a 50% penalty. Assignments submitted more than 72 hours after the posted deadline will receive a 0 grade. Do not wait until the last minute to submit your solution.