Lab 4

Due Mar 9 by 7:30pm **Points** 40 **Submitting** a file upload
File Types zip, tgz, and bz2 **Attempts** 1 **Allowed Attempts** 1
Available until Mar 12 at 7:30pm

This assignment was locked Mar 12 at 7:30pm.

In this assignment you continue to use the Gazebo simulator with a Pioneer P2 DX robot. Your task is to modify the `drawsquarefb` example we saw in class to make the Pioneer robot draw a square. The provided example (available in the MRTP GitHub) needs to be modified because the Pioneer does not provide its pose through a topic as the turtle does.

Here are the details to setup your working environment (very similar to last week).

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`, wheres 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, source the overlay for the MRTP 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.
4. If you now run `ros2 topic list` you will see that the P2 DX robot accepts velocity commands on a topic called `/p2dx/cmd_vel` and publishes to a topic `/p2dx/odom` of type `nav_msgs/msg/Odometry.`

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 `lab4`. This package will contain the solution to this lab.
- Write a node `drawsquarefb` that mimics the functionality of what we saw in class by subscribing to

`nav_msgs/msg/Odometry`. This node is both a publisher and a subscriber.

- **Important**: use `ros2 interface show` to determine the structure of messages sent on the topic `nav_msgs/msg/Odometry`. The pose of the robot is in the field `pose`.
- Important: the orientation of the robot is provided as a quaternion. To extract the orientation of the robot (yaw) from the quaternion see the example `convert.cpp` available **here (https://github.com/stefanocarpin/MRTP/blob/main/MRTP/src/examples/src/convert.cpp)** .
- Update the manifest file and `CMakeLists.txt` as needed. Your code now depends on the package `nav_msgs`.

Notes:

1. even if you solve this exercise correctly, you may observe that the P2 DX robot does not necessarily follow the desired trajectory. This is not a problem, but rather a consequence of different physical constants between the two (mass, friction, etc.). The logic of your solution is more important than the path followed by the robot.
2. You can copy/paste or reuse as much as you wish from the code samples provided in the GitHub.

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.