Section 5: Running the Navigator

Overview

The goals for this section:

1. Apply navigator in hardware

Setup

Task 1.1 — Cleanup. Remove directory `~/autonomy_ws` if it exists. This may be leftover from a previous section

Task 1.2 — Directories. Create the base directory structure for your group's ROS workspace, `~/autonomy_ws/src`.

Task 1.3 — Clone and Branch. Clone your group's GitHub repo to the `src/` directory, and create a new branch called `s5_inperson`.

Task 1.4 – Copy homework navigator. We will be adding the code you wrote in HW2 to your autonomy stack to control the TurtleBot. Copy the file `navigator.py` to the folder `~/autonomy_ws/src/<group_repo>/autonomy_repo/scripts` and the file `navigator.launch.py` to the folder `~/autonomy_ws/src/<group_repo>/autonomy_repo/launch` on the AA274a laptop.

Task 1.5 — Build. Navigate back to the root of your workspace, `~/autonomy_ws/`, and build your ROS workspace using `colcon build` .

Testing your HW

Task 2.1 — Launch simulator. Firstly, we will test out your HW2 code before you can test it out on the actual robot. Start your TurtleBot simulator by:

```shell
ros2 launch asl_tb3_sim root.launch.py
```

Task 2.2 — Launch your navigator. In a new terminal, source your workspace, and then launch the navigator:

```shell
source ~/autonomy_ws/install/setup.bash
ros2 launch autonomy_repo navigator.launch.py
```

Task 2.3 – Navigate to goal. Set a new goal pose in RViz by selecting the `2D Goal Pose` button on the top toolbar and then clicking and dragging in the environment to set a goal pose.

Checkpoint — Show the CA your TurtleBot navigating in sim.

Full TurtleBot Bring Up

Time to test this out on the actual robot! Determine the ROS_DOMAIN_ID of the laptop you are using for this section using this command:

```shell
echo $ROS_DOMAIN_ID
```

Call a CA over to assign a robot to you based on your domain ID!

CAs will move your TurtleBot to the map on the ground, and turn on your TurtleBot for you.

Then, use the following workflow to start up the TurtleBot.

```shell
# TODO: close any terminals running the sim in Gazebo before
using the hardware

# Use SSH to connect to the Turtlebot
ssh aa274@<name_from_ca>.local

# Enter the password given by the CA

# Start the ROS environment
micromamba activate ros_env

# Bring up the Turtlebot
ros2 launch asl_tb3_driver bringup.launch.py
```

Task 3.1 — Available Topics. Like last time, run `ros2 topic list` from a new terminal window on the laptop to verify that your TurtleBot is running!

Task 3.2 — Sim2Real. Launch your `navigator.launch.py` with `use_sim_time` set to `false`

```shell
ros2 launch autonomy_repo navigator.launch.py use_sim_time:=false
```

You can then specify goal poses in RViz, and watch your robot navigate through the environment!

Checkpoint — Call a CA over to demonstrate this.

Clean Up

Clean up the workstation.

1. Push your code (`add`, `commit`, and `push`) to your `s5_inperson` branch!
2. Create a PR either from the GitHub web page or by running `gh pr create` in your terminal.
3. Verify that you've pushed correctly by checking if the code you wrote is in the repository on GitHub.com
4. Delete the `~/autonomy_ws/` .
5. Log out of GitHub on the workstation.
6. Close all terminals and windows!
7. Shutdown laptop

Checkpoint — Call a CA over to sign you out!