ECSE/CSDS 376/476
Assignment 4: Mobot path execution
Assigned: 2/8/22
Due: 2/18/22

This assignment is a **group** assignment. Your objective is to control both the simulated mobile robot ("mobot") and the physical robot to follow a path with hard-coded subgoals.

Your software architecture should include the following nodes (which can be modified from existing code in learning_ROS, Part 4).

Nodes:
current_state_publisher:
This node will later combine absolute pose information (e.g. from GPS or LIDAR/map-based localization) with high-speed Odom information. For this assignment, this node should merely subscribe to the Odom topic and republish Odom on the topic "current_state".

lidar_alarm: This node should be an upgrade of the STDR LIDAR alarm. It should subscribe to the mobot's LIDAR topic and interpret safe paths towards a goal pose with sufficient look-ahead for graceful braking.

modal_trajectory_controller:
This node will be upgraded to perform "lane-drift" correction, heading control and path progress control. For this assignment, implement this equivalent to the open-loop controller, which merely copies a desired state twist to cmd_vel. The controller should be modified to prepare for different control modes: spin-in-place, straight-line-motion, and halt.

des_state_publisher_service:
This node should use functions from the traj_builder library (Part4/traj_builder) to construct triangular and trapezoidal trajectory plans for either forward travel or spin-in-place motions. It should receive a goal pose as a service request. It should attempt to stream sequential desired states in accordance with the request, resulting in returning either success or failure. Reasons for failure would include: encountering a lidar_alarm prior to reaching the goal pose, and failure to converge on the goal pose within some tolerance. In response to a lidar_alarm, this service should dynamically construct and publish (stream) a graceful braking trajectory.

The des_state_publisher should include a "mode" code for spin-in-place, forward-travel, or halt (at specified final state).

navigation_coordinator:
This node should contain a plan for a sequence of path vertices, and send these as requests one at a time to the des_state_publisher_service. The coordinator will suspend while waiting for a response from the des_state_publisher_service. If the response is "false", the coordinator should pause briefly, then resend the last, unsuccessful goal vertex. (This could be adequate if a pedestrian blocks the robot, then subsequently walks away). The navigation_coordinator will grow in sophistication to incorporate path planning and path replanning (e.g. to circumvent unexpected obstacles).

Run the above nodes concurrently together with the mobot simulator:
roslaunch mobot_urdf mobot_in_pen.launch

Your goal is to have the robot execute some interesting path that can be run both in simulation and in the lab on the physical robot.

Deliverables:
Submit a (group) report. It should describe your theory of operation of each of your nodes and include a github pointer to each of the code implementations. Include a discussion section, in which you describe your code edits/additions and your observations—particularly anything unexpected. Include a video of your solution running (may be a link to a YouTube), both in simulation and on the physical robot.