For this problem set, use the Gazebo mobile-robot model in package "mobot_urdf"

You can start up the simulated robot with:
roslaunch mobot_urdf mobot_in_pen.launch

You should use the code in packages “mobot_pub_des_state” and “traj_builder,” to control the Gazebo mobot model in package "mobot_urdf"
See the text and the README files for details of theory of operation and running tests, and Section 9.1 of the text. With the robot simulation running, try running:
rosrun traj_builder traj_builder_example_main

In “traj_builder.cpp”, the function “TrajBuilder::build_braking_traj()” is not completed. You
should edit this to construct a valid braking trajectory to bring the robot to a halt, whether moving
forward, spinning or both.

In “pub_des_state.cpp” you should add code to listen for e-stop conditions, including a “lidar
alarm” (using your own code to respond to LIDAR sensor data). Graceful braking should be
performed when there is a detected obstacle. Graceful recovery should also be performed, picking
up from the halted state and resuming progress towards the next (or previously unobtained) subgoal.
Desired states should stop being published—and motion is re-enabled, the robot should start up gracefully to resume its journey to its next (or previously unobtained) subgoal.
You should recommend values for speed limits (translational and rotational) and acceleration limits
(translational and rotational). You should demonstrate proper functioning of the following
behaviors:
• graceful halt from LIDAR alarm
• graceful recovery from LIDAR alarm
• ability to execute open-loop control corresponding to a prescribed polyline path
• ability to flush a path plan and replace it

Submit: a pointer to your code on github, a screencast movie of your robot responding to the LIDAR with a
graceful halt, a screenshot of rqt_plot showing your commanded and actual forward velocity while halting
due to a LIDAR alarm, and a brief write-up of your solution approach.