Zach Janice, znj
PS6

To begin, I was unable to fully implement the solution to this assignment; while an emergency halt was in place and did bring the robot to a stop, it doesn't fully work as intended (will explain later).

The approach behind the changes in the code was to bring the robot to a slow stop over the course of the halt. The maximum-allowed deceleration for linear and angular movement - corresponding to the negative values of the maximum acceleration values respectively - were used to iteratively construct a series of braking poses that slowed the robot's linear and angular velocity until both were equal to zero. These changes were made in the "traj_builder.cpp" file.

The other changes were made in "pub_des_state.cpp". In meeting resistance with trying to compile alterations to the DesStatePublisher class, I otherwise attempted to set up a subscriber for an alarm topic. When the topic was tripped, the emergency-stop trigger was raised, and when the topic was cleared the reset trigger was raised.

The collision alarm used is provided in the provided package "e_stop_alarm".

The only exact outstanding issue is within the halting process in "traj_builder.cpp". Without subscribing to some degree of an odometry topic (or Gazebo's output of the states of its models), as no node handler exists within the trajectory builder class, I do not have a way of determining the robot's current linear and angular velocity. As such, the halting process defaults to assuming a forward velocity equal to the maximum velocity and an angular velocity of zero. In terms of the simulation, the halting process makes the robot speed up and then slow down to a stop as a result. I'm sure there is a way to gather this information to address this issue, but I am unaware as to what that solution is.