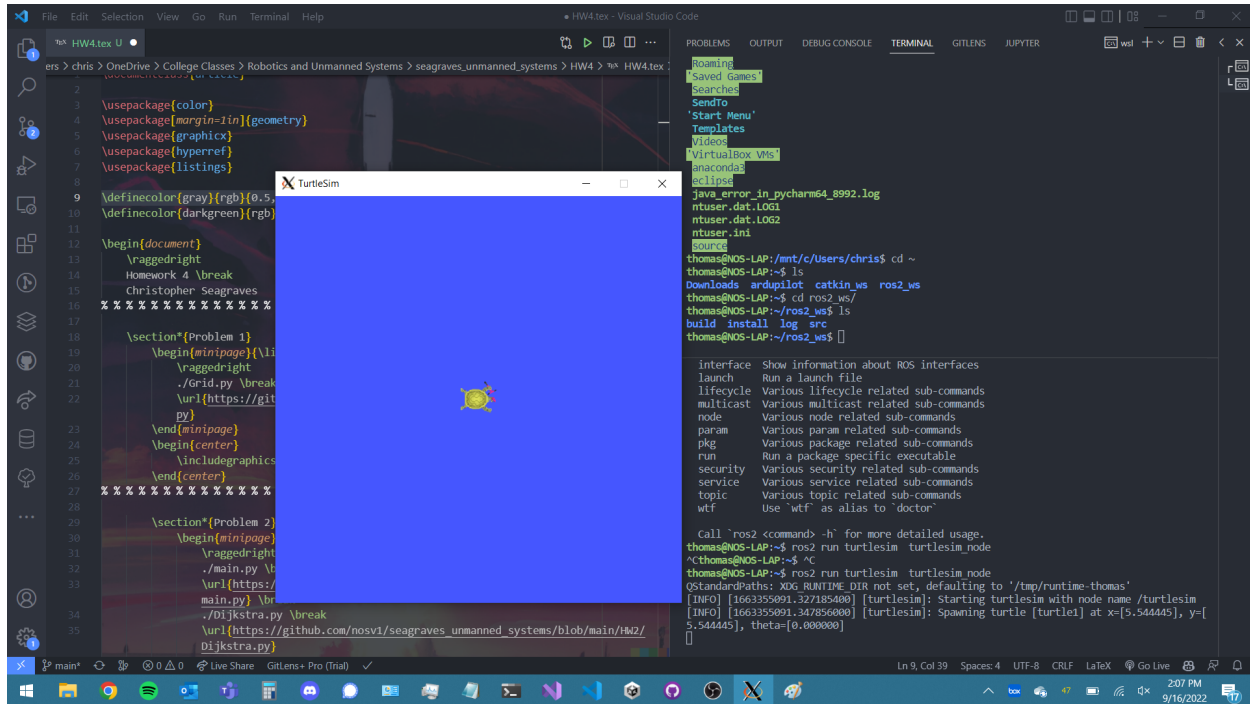


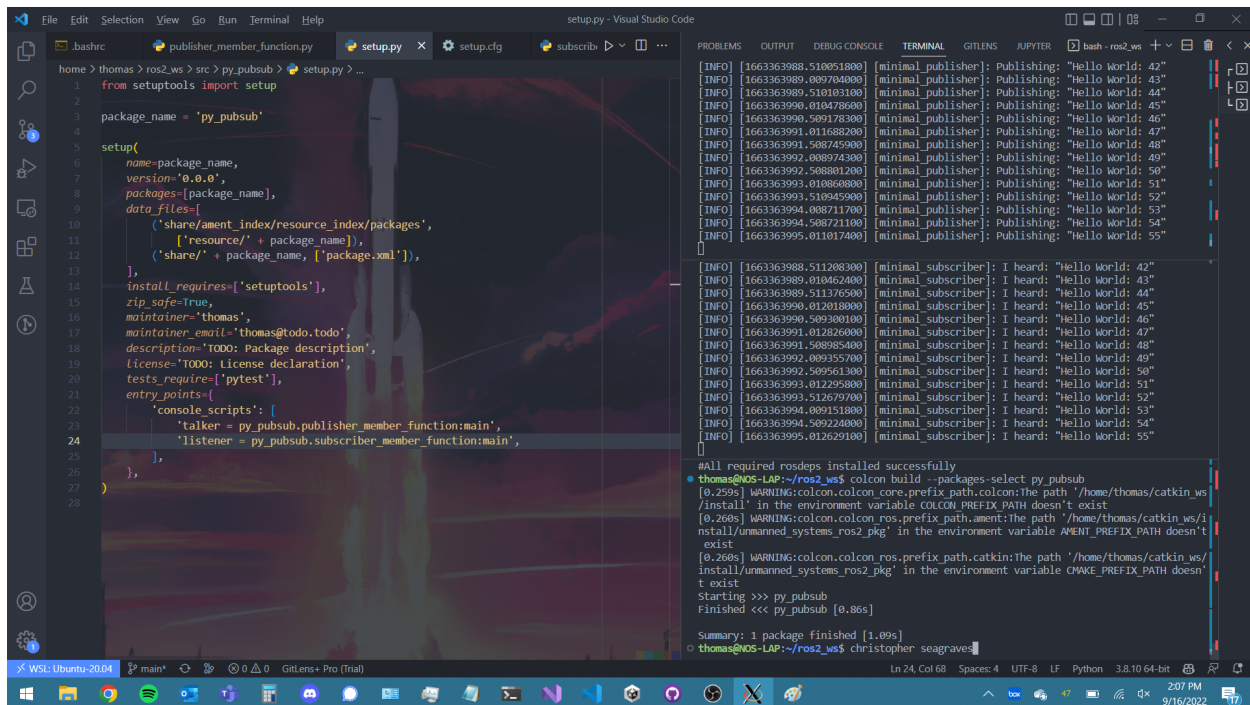
Homework 4 Christopher Seagraves

Problem 1



The screenshot shows a Visual Studio Code editor with a LaTeX document titled "HW4.tex". The document contains a preamble with package imports for color, margin, geometry, graphics, hyperref, and listings. It also includes a document structure with sections for Problem 1 and Problem 2. A TurtleSim window is open in the center, displaying a blue background with a yellow robot icon. The terminal window on the right shows the execution of ROS commands, including `ros2 run turtlesim turtlesim_node`, and the output of the `ros2 run` command, which shows the turtle's position and orientation.

Problem 2

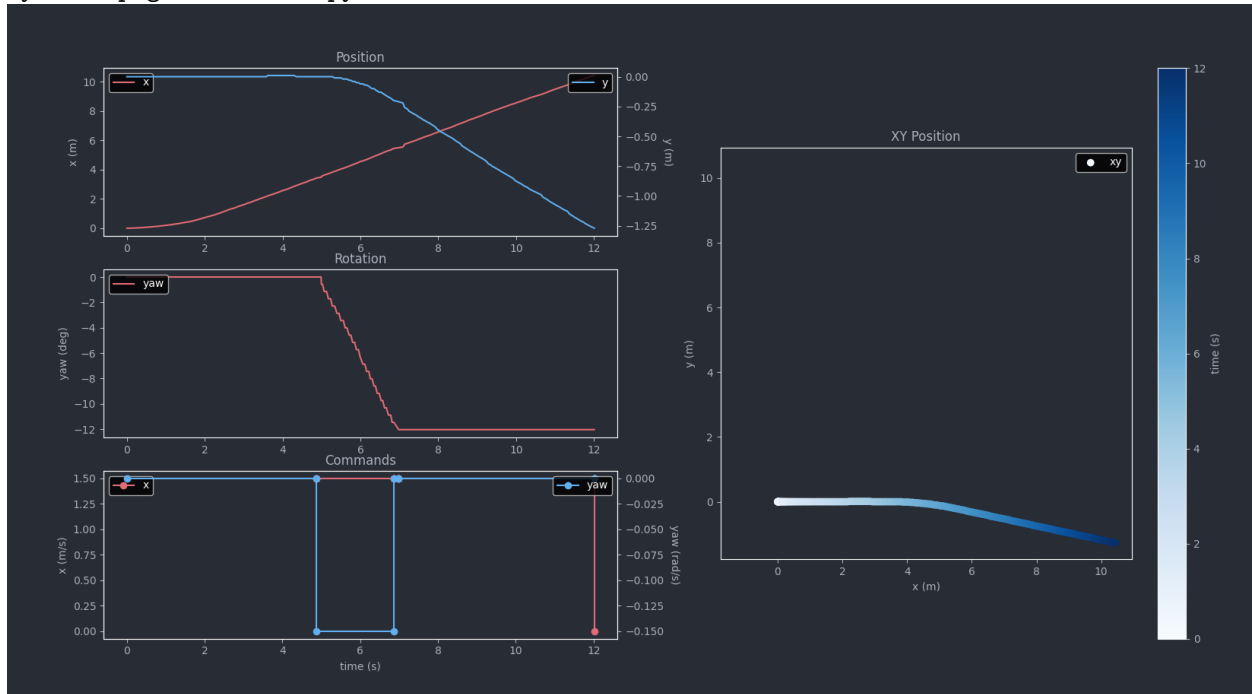


The screenshot shows a Visual Studio Code editor with a Python script titled "setup.py". The script defines a ROS2 package named "py_pubsub" and sets up the package's metadata, including the name, version, author, and entry points. The terminal window on the right shows the execution of the `colcon build` command, which builds the package and its dependencies. The output of the build command shows the package's name, version, and the path to the built package. The summary of the package build shows that the package was built successfully.

Problem 3

controller.py

https://github.com/nosv1/seagraves_unmanned_systems_pkg/blob/master/seagraves_unmanned_systems_pkg/controller.py

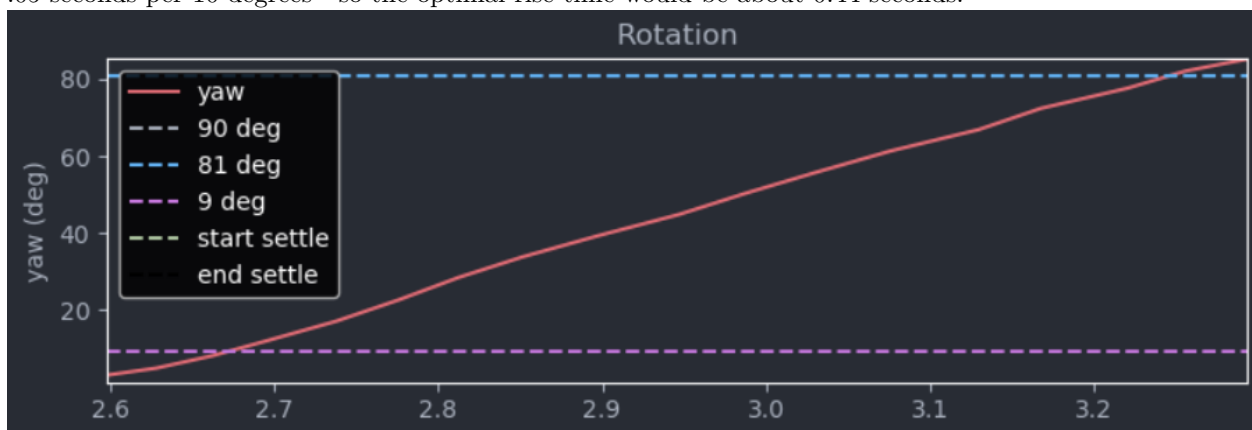


Problem 4

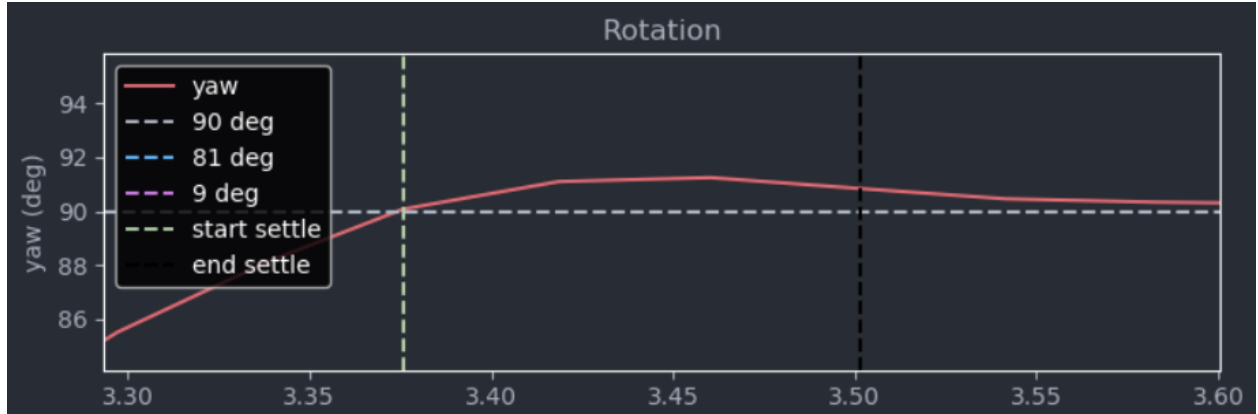
$K_p = 6.5$, $K_i = 0.0$, $K_d = 0.0$

Rise time = 0.6 seconds

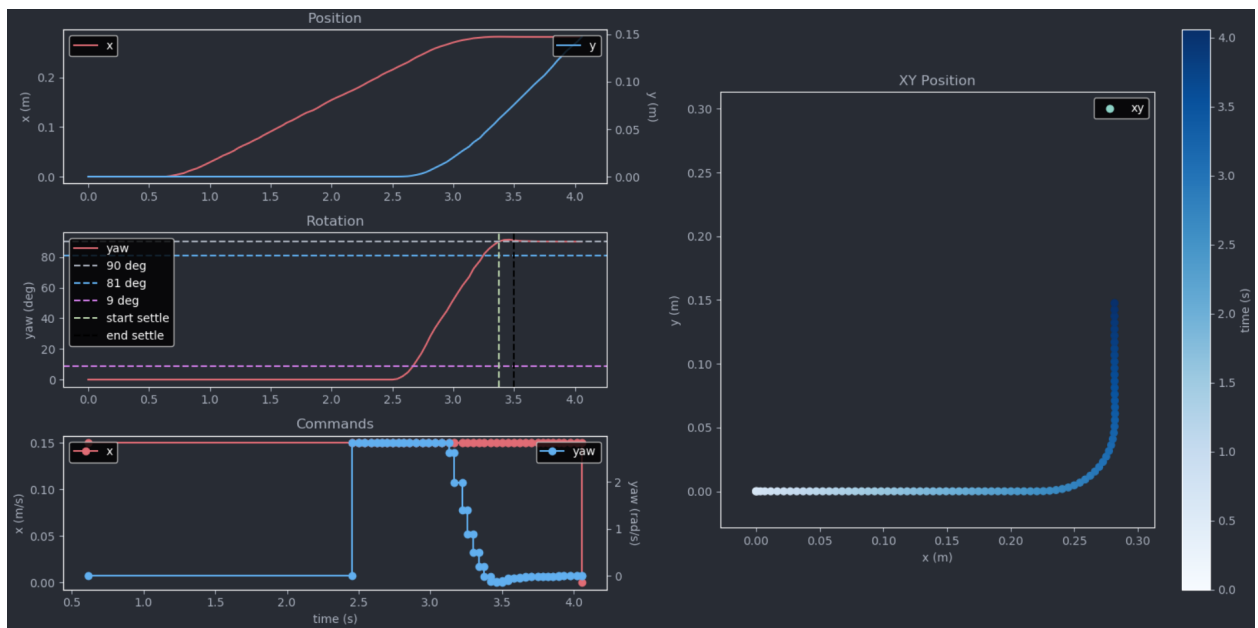
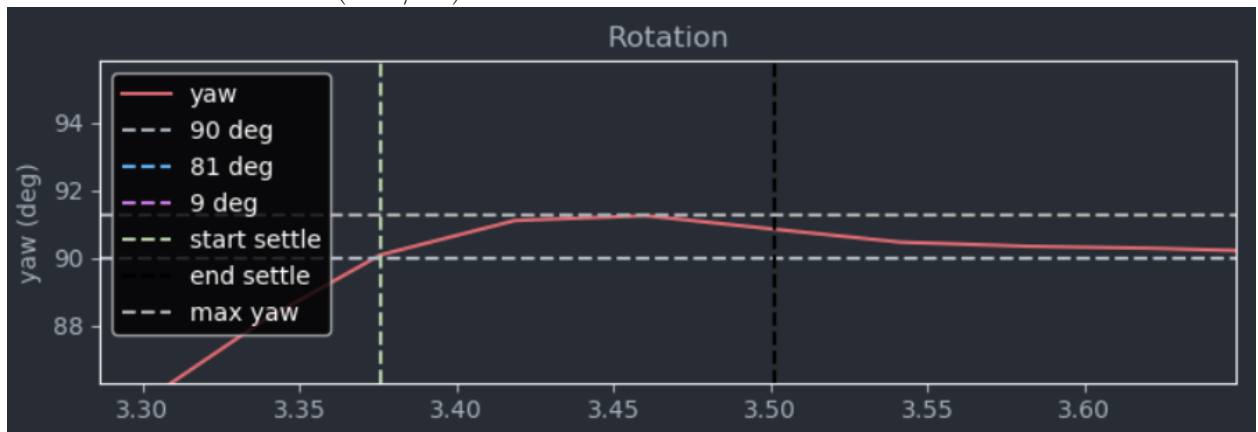
Given Burger can turn at 2.84 rad/s (162.4 deg/s), it can complete a 90 degree turn in 0.55 seconds - about .05 seconds per 10 degrees - so the optimal rise time would be about 0.44 seconds.



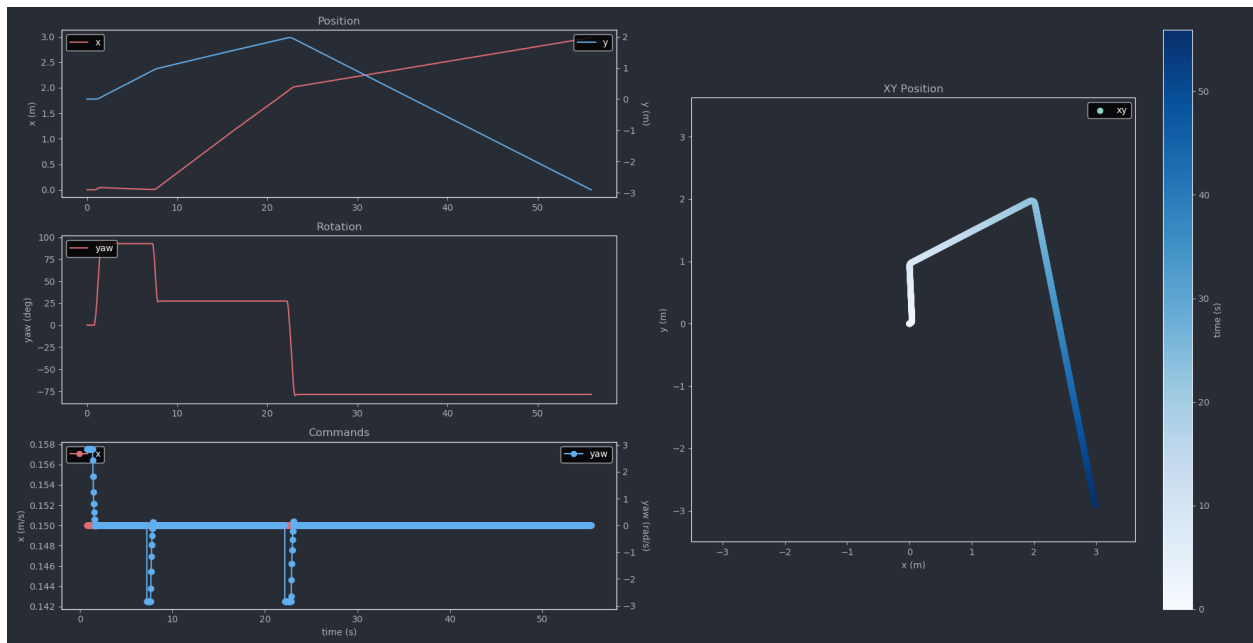
Settle time = 0.2 seconds



Percent overshoot = 1% $1 - (91.2 / 90)$



Problem 5



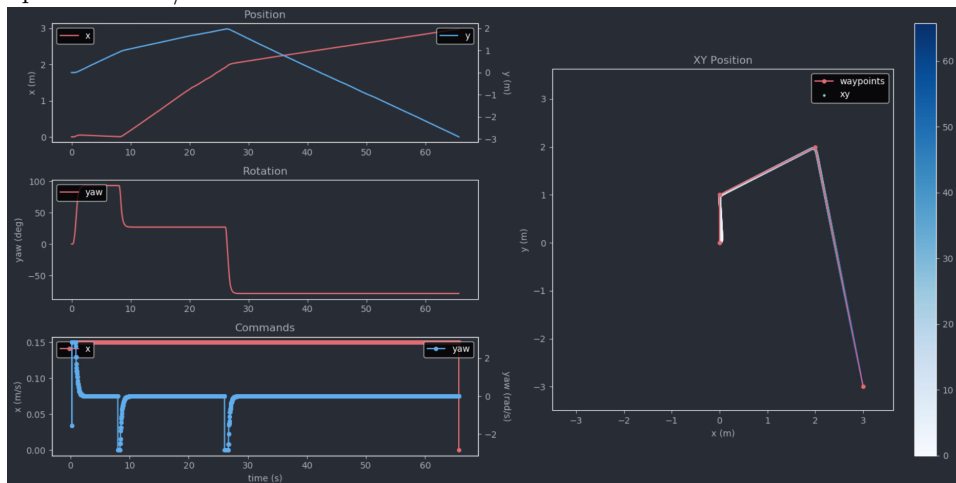
Problem 6

All runs below use the PID: $k_p=4.5$, $k_i=0$, $k_d=0.25$

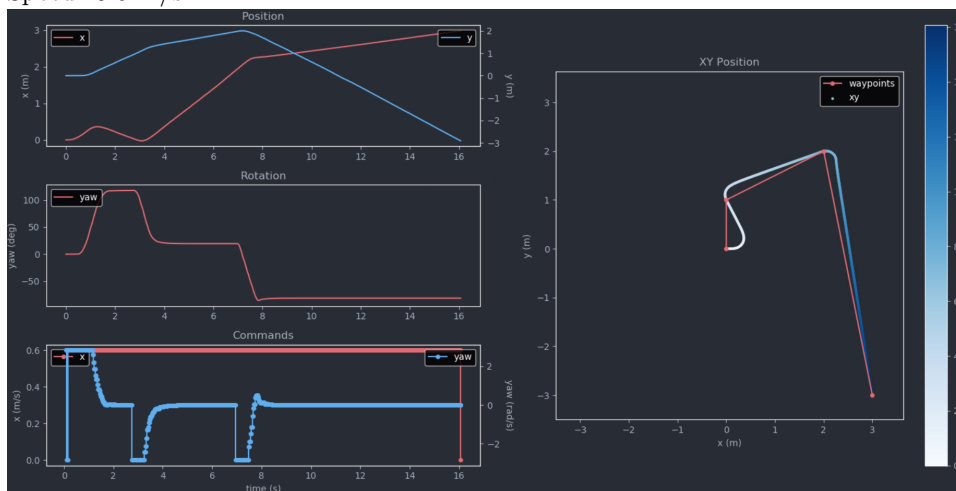
Clearly, as the speed increases, the PID controller is put under more stress; at 1.5m/s, this PID controller fails completley. Anything over 1 m/s and your pushing your luck, in my opinion.

It would be cool to have a PID controller for throttle that also takes into consideration the heading error so it could maybe slow down if it detected error and pedal to the medal otherwise... but this still wouldn't solve the overshoot problem compeltley; you would need a method to know how fast you could turn and slow down for the next 'big step' in optimality.

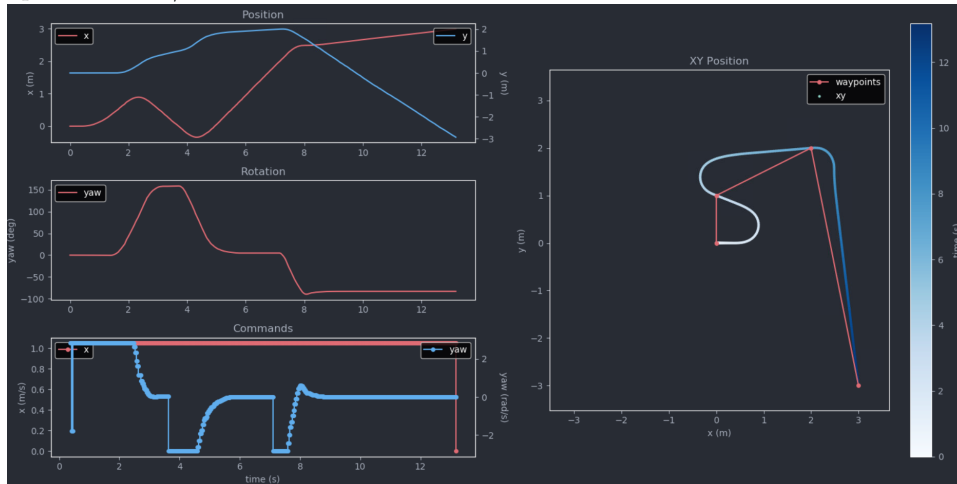
Speed: 0.15 m/s



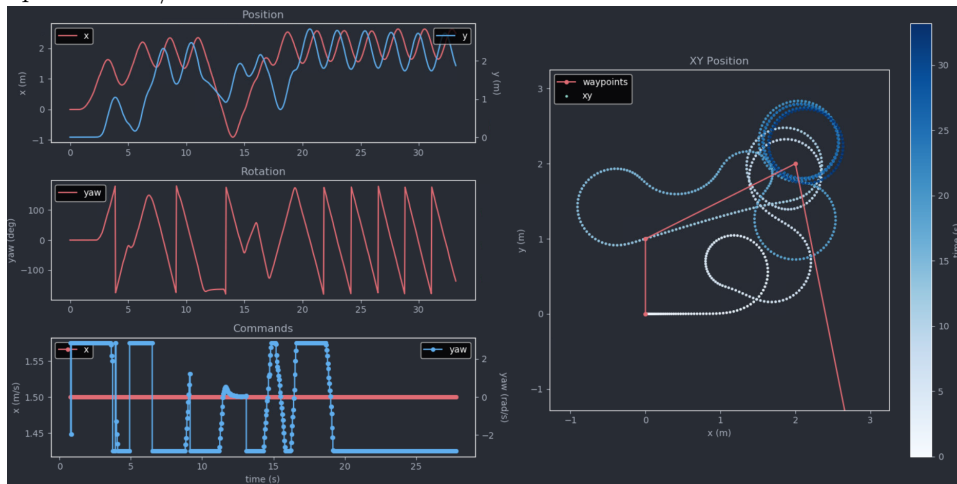
Speed: 0.6 m/s



Speed: 1.05 m/s



Speed: 1.5 m/s



Notes

Personal imports can be found in the 'support_module' directory

Log plotter can be found in 'extra' directory

You may notice the additional package directory, 'turtlebot3_gazebo'. I edited the original worlds to run in headless mode and/or speed up simulations, had to edit /worlds/empty_worlds/burger.model and /launch/empty_world_no_x_launch.py