Lab 6 Report

Justen Stall <stallj2@udayton.edu>
Keenan Griffin <griffink5@udayton.edu>
CPS 592 Intro to Robotics
Assignment: Lab 06
Dr. Nick Stiffler

Task 1:
Challenges:

Sensor issues. We were sleeping 100ms on each iteration to allow the robot's sensors to update. While every other sensor could reliably be observed at that interval, the infrared sensors would only receive an opcode once every ten or so readings. This made a PID controller much less effective for the docking approach, so we had to zero out the integral and derivative terms since they had such little valid data stored. Another challenge was the docking system built into the Roomba. Lots of diagram drawing and sensor checking helped, but the force field was not drawn to scale in the Roomba manual. We anticipated the robot would get much closer to the dock before sensing the force field. Initially this made repositioning in front of the dock a little harder, but using the infrared omni sensors helped. We were able to detect the buoys with the omni sensor to let us know when we were in front of the dock. It was a challenge to figure out how to go from wall following to aligning to the dock, but we ended up figuring out a pretty robust solution.

What didn't work as anticipated:

We did not anticipate the robot's charging state detection to work the way it did. We spent more time than we should on getting the robot to detect charging because of our misreading of some imprecise language in the Open Interface specification. Our interpretation of the Create iRobot manual left us wondering if the robot needed to be in passive to get a charging state. During the majority of our tests for charging state we assumed the robot could get a charging state of '2' even in full mode. We knew the robot wouldn't charge in safe or full mode, but we did not need the robot charging necessarily, just stopping in the charging state.

How we overcame challenges:

We spent lots of time trying each possible sensor and tweaking the velocities, error weights, and PID gains. To make the approach to the dock easier we slowed the robot way down to a crawl. This allowed the robot to make more fine tune adjustments to its position to the dock, and gave the sensors plenty of time to report the current position of the robot.Our dock approach also had an issue missing the buoys, and continuing past the dock. By incorporating the infrared omni sensor to detect when it was receiving just the red buoy or the green buoy we overcame the challenges of the robot missing the dock completely. Whenever the omni sensor got just one buoy the robot knew it was getting way off and changed its approach accordingly. Tinkering with velocities, error weights and PID gains meant that we needed to keep the code as simple and reusable as possible so we could iterate fast. We designed helper functions, which broke down the overall behavior of the dock approach into more manageable code. Rotations and drives were handled in their own functions. This allowed us to simply call the rotation or drive function again when that action was needed.

What we would do differently to improve:

If we had more time, we would greatly benefit from having an on-robot controller. The communication delay and its effects on our robot's reaction time were an issue for the past two labs. Given more time we would have thrown out the invalid IR opcode sensor reads, and only run PID when a valid read was given. There is a chance it could have worked fine if we factored in the change in time between readings. It was not worth the effort though because our approach worked well enough. We also could have tried different approaches to the initial alignment with the dock. Our robot journeyed away from the wall at 30 degrees until the far buoy was detected, but from our experimenting with the built-in docking feature, it seems that we could have found a way to drive in a radius around the dock. I would have liked to try out a "wall follow"-like behavior but based on the dock's force field, which could have resulted in similar behavior to the robot's built-in docking. We also could have added the switch to Passive mode into our program. We left it to ourselves to switch to passive mode to check if it was charging, but we could have taken the time to have the robot check, and if it was not on power, retry the last portion of the dock approach.

What went well:

Tuning our PID for the docking approach went much smoother than tuning the PID for wall following, partially because of a class we implemented and also because it was a simpler and smaller error value. Additionally, infrared sensors were more exact with their readings than the light bump sensors. Specifically, force field detection was more apparent then figuring out what the light bump sensor reading needed to be for wall following.