CSCE 274 Project 3

Team 05

Chris McKinney
Josh Benton
Nick Wrobel

Description

The main library for this project is contained in lib3.[ch], and the main function is in proj3.c. The other files, which reside in the utils directory, provide the rest of the functionality for both parts of the project.

This first part of this assignment required development of a PID controller to facilitate wall following. To achieve this, we first developed a PID step function which also required us to develop a fixed queue to hold values for an integral term (the queue is fixed because it is backed by a circular array rather than a linked list, which gives some performance enhancements), and an extra variable for the previous error value. This allowed us to implement a simplified integral and derivative function. We also created numerous constants in lib3.h to control various aspects of the PID controller, including the proportional, integral, and derivative gains.

Other utilities such as setting the queue up to restrict the scope of the integral part were created as the project developed. One such utility was the fixed queue used in integral calculations. Chris also improved the serial input program on the computer to handle the data from serial dump (It just removes null characters and outputs to a file). We also made a spreadsheet to mathematically model the PID controller and experiment with different values and techniques for modifying the values.

Once all the PID controller code was written, we added a serial output to verify the code is working utilizing the `printf` functionality developed for the last project. Afterwards we created an update motors function which uses the results of the PID step function to actually drive the robot keeping the wall signal at a set point.

The remainder of the project is just utilizing the already implemented sensor reading code and timer based execution code developed in the first project to handle such things as bump and wheel drop sensors and modal overturning with continued sensor polling and handling. Wall bumps were responded to by turning counter-clockwise until the bump sensor was no longer depressed, and then turning ten degrees more. This “overturning,” co-opted from the previous project's code, *greatly* improved the response of the PID controller.

SELECTING THE PID VALUES
From prior knowledge and experience working with PID controllers, we knew the P term was supposed to be the most influential, followed by the D term, followed by the I term. From there, we followed a guide on-line, which essentially had you increase the gains in the aforementioned order until the desired effects emerged. Afterwards, we did a lot of trial and error and landed values that worked well. A lot of this later testing utilized the spreadsheet in addition to actually testing the robot.

Evaluation (and reflection)

The PID controller seems to handle wall following rather well. After a reduction in forward speed, a coarser time granularity of the controller, and the addition of “overturning,” the robot does not oscillate much at all going around the test setup. The only complaint that remains for this writer is that on convex corners, the robot bumps the corner before rounding it. This, however, can be viewed as a limitation of the placement of the wall sensor, and solving it would likely require a significant reduction in the reactivity of the system.

In the spec for this project, the question, “would [you] recommend PID control to someone else that wanted to complete this task,” is asked. I would. It seemed at first that the I term of the controller was solely detrimental to the operation of the controller, and that a PD controller would be superior to a PID controller, but given the final results and the workings of the mathematics, I am convinced that a PID controller is suitable for this task, and that even at higher speeds, finer time granularity and other recalibration could enable the same kind of result (keeping in mind the limits of the hardware, of course, especially the sensor polling time).

Allocation of Effort

Chris, Nick, and Josh all contributed to the design decisions and debugging and documentation of the code. Chris was the primary programmer. Josh was the primary author of the paper, and Chris and Nick proofread it for errors. Chris rewrote the evaluation and reflection.