Real-Time Systems

Lab 3: Writing a Controller

Due: February 15, 2018 (Before Class)

Introduction

Now that you've had enough experience with the basics of *Ringo* and you have also played with *FreeRTOS*, it's time to write controller tasks for Ringo. You will be writing two controller tasks: **a)** following a straight line, and **b)** turning or rotating movement. You will demonstrate your controllers by making Ringo carve out a square (or rectangle) in its motion. The square must be at least $1\,\text{ft} \times 1\,\text{ft}$. You can include another task that will switch between the controllers at the right time. **Protip:** it will be helpful if you think carefully about your controllers will fit into a larger control architecture in which you have a path planner. What inputs should the rotational controller take? What inputs should the straight line controller take?

You can write a controller of your choice, using any control law of your choice as long as you document and justify why you did what you did. You are not allowed to use PlumGeek's predefined behaviors and functions for this task (those in "Behaviors.h" for example), however you can use their sensor sampling functions to obtain the current heading and rotation values of the robot as well the function to cause the motors to move. Having said that, it is in your best interest to reduce your dependence on PlumGeek's libraries since lab 6 will require you to remove all such dependencies.

Example

One option for a control law is PID control. A continuous PID controller continuously calculates an error value *e(t)* as the difference between a desired set point and a measured point and applies a correction based on proportional, integral, and derivative terms. This law can be mathematically represented as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

The reason to choose such a controller is that it's easier to understand and implement and doesn't require a model. So if you don't have experience with controller development this may be a good choice. For the purpose of this lab, a PID controller is sufficient but you will want to make sure you tune the gains to get an appropriate amount of accuracy. You should recognize that the PID equation above is in continuous time. Obviously, working in the discrete computational world, that equation won't work as is.

What to submit

1. **(35 points)** Zip the entire Arduino project so that we can just unzip and execute the source code.
   - Please include a diff of the source code between this lab and previous one. Please use a "diff" tool of some kind that highlights the changes from previous files (e.g. `https://www.diffchecker.com/`, meld, github, etc.)
2. **(30 points)** Documentation of the task(s) you have implemented for this lab. (0.5 - 1 page long)
   - Clearly state the control law you have implemented and why. Include the equation for the control law.

- Describe each implemented task briefly and how you decided how to break up the behaviors into different tasks (or why you chose to use a single task). Classify any tasks into periodic, aperiodic, or sporadic, and provide justifications on why a task is of a certain type and what were the thoughts while deciding the frequency of the tasks.

3. **(35 points)** Record a short video of the robot going around the square/rectangle doing right turns, and then again doing left turns. This verifies you can rotate in both directions.

    - Please upload the video to YouTube, Vimeo, or something similar and then provide me with a link. Just put the link somewhere in the writeup you did in #2 above.

4. Upload all of this into "handin" at `https://cse-apps.unl.edu/handin`