Lab #5: Get Loopy

In this one-week exercise, you will learn to write code to implement a motion control loop.

You have created drivers for motors and optical encoders. Now you can put these together and make a closed loop position control system. This week's exercise is to make a modular motor position controller which uses your motor control class and your encoder driver class. The modularity is so that you will be able to make two or more independent position controllers for the pan and tilt axes of...something.

Your controller must be at least a proportional controller. If you would like to implement a PI, PD, PID, or state feedback controller that's even better. You should be careful about noise affecting derivative control and reset windup affecting integral control if you implement them. There are some PID controller implementation documents on the Poodle site; even if you are an old hand at PID control as practiced in controls courses, these documents will be helpful in implementing all that $\vec{x}(t) = e^{\mathbf{A}(t-t_0)}\vec{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\vec{u}(\tau)d\tau$ stuff on a real microcontroller.

There should be a motor control task which can control any of two or more motors, an encoder reading task which reads the encoders, and a controller task which implements your control algorithm. Data should be passed between all these tasks in a thread-safe manner.

Make sure to test your position controller thoroughly. Example tests include:

- Making the motor move several revolutions, then stop at the same angle at which it began
- Making the motor move back and forth between two setpoints several times in quick succession
- Grabbing the motor shaft when it is holding one position, making sure that when you twist the shaft away from the setpoint the controller causes the motor to push back
- Making the motor track a moving setpoint, for example by following the position of a potentiometer which the user turns

You should be able to think of other ways to try to make your controller fail (and hopefully fail at doing so).

Deliverables

Next week you should turn in a report including the following:

- A one-page memo describing your controller and test program. Be sure to discuss how well the program worked in tests.
- A printout of the Doxygen manual for your encoder driver only. This manual should be complete enough so that someone can use your driver using the manual as a guide (and not having to read your source code). Please print it double sided.
- A printout of your task class and test program code. Please print it double sided.

Please email a .zip file of your code to `me@me.me` as usual, and also as usual, you may be asked to demonstrate the operation of your program in lab on the due date.