Industrial Automation

Project #1: T-RECS Continuous-Time (C.T.) Controller: Design; Simulate; Implement

Due: See Canvas "Assignments"

For this project, you will design a linear feedback controller around a linearized model of the T-RECS (you may need to redo some Sys ID from Project #0), then simulate the linear controller around the non-linear T-RECS to make sure it doesn't fail, you will then implement the controller on the Arduino Due via Simulink, and finally you will present all of that in a report form (written in Latex).

Your report should include, but not limited to, the following aspects:

- Report Writing
  - Tell a good story
  - Quality Figures
  - Take Pride in your work
  - There should be a sufficient amount of information and directions so that someone can replicate this work
- Linearized Model
  - You will either derive a linearized model from the non-linear model (which may need some updated Sys ID from Project #0) **or**
  - You can first implement a simple controller (say just P maybe) around the T-RECS by trial-and-error and then do some new Sys ID experiments, on the new system which is under basic control, to derive the the transfer-function of that closed-loop system (it still may exhibit some non-linear features, so you may need to do some linearization of this system too). You will now pretend that this new system is your open-loop system and then you will design a controller around this "open-loop" system to enhance performance. This a good trick for systems that are unstable and make it hard to get an open-loop transfer-function. So, yes, you will technically have two controllers (an inner and an outer, where the inner was chosen by trial-and-error and the outer was designed using feedback control techniques).
- Hardware
  - As with Project #0, you need to describe the Hardware that you are using and especially give details on hardware that you haven't previously described in Project #0. Feel free to use some figures from Project #0 if they useful for this project, but don't directly copy sections of text and images and paste into this new project one-to-one. **One Example:** I mentioned in class that the Due's A2D has a certain number of bits and that will be important to reading in the potentiometer... make sure you explain that in some detail.

  - It is crucial that you get the Simulink communicating with the Arduino Due as soon as possible, because there may be some hiccups for you on this and the earlier you figure it out the better. So this means you should be able to read the potentiometer and make the motor spin. **Do not wait to get Simulink working with the Due for the "Implementation" section.**

- Controller Design and Simulation
  - In this section, you will design a Controller (PID sufficient as a controller structure, but you may design more advance controllers if you are inspired) for the linearized "open-loop" system that you got from the "Linearized Model" section.
  - Don't forget to analyze your control design via plots like "root-locus" and "bode" (both closed-loop and open-loop).
  - Once you have a controller ($C(s)$), you should simulate this controller on the non-linear model and verify that it doesn't go unstable.

- Implementation
  - Now you will implement the $C(s)$ in Simulink (use the C.T. Transfer-Function block) and run it on the Arduino Due. You may not use any "PID" blocks in Simulink... you must only use the Transfer-Function block (even if you Controller is PID)
  - Make sure you collect experiment results of the controller implementation and compare them to your design and simulation results.
  - You must define a sampling rate (even though we are ignoring the fact that this is discrete-time problem for this project), so use a 1 millisecond sampling period.

- References

- Appendices (for example, code portions or derivations that you think shouldn't be in the body of the report but should be in the appendices).