

Lab 2 - 2003-10-16

This lab has two parts, which lab partners can work on simultaneously. One part is to build an double-integrator analog circuit. The other part is to get your QNX computer working for you. Both lab partners should do the "prior to Thursday" parts, though you are very welcome to collaborate. On Thursday, you may specialize in the analog or digital part, but be sure to fully teach each other as well.

ANALOG –

In this lab you will build an analog simulation of a motor and flywheel. This week, you will control the input signal *manually* to try to accomplish position control of the flywheel. Next week you will build an analog and a digital controller for the same purpose.

I will get the electronics parts you need.¹

The command input to a motor is a torque τ . (Really, it's a current i , to which torque is proportionate). If the motor is connected to a flywheel with moment of inertia I , then the velocity of the flywheel $\omega = 1/I \int \tau$. The position of the flywheel is $\theta = 1/I \iint \tau$. In other words, the flywheel is a double integrator of the command torque.

PRIOR TO THURSDAY: Attached is a circuit for an analog double integrator. Be sure you understand it. Pick the remaining component values such that it is proper analog computer for the motor/flywheel, with $I=1$, τ in volts as the input, and θ in volts as the output. In other words, if we give the circuit a constant input $\tau = 1$ volt, after 3 seconds we will have a $\omega = 3$ volts rising linearly as t^1 , and a $\theta = 4.5$ volt, rising quadratically as $1/2 t^2$.

The input is provided manually by turning the 10K potentiometer. We will impose an actuation limit of +/- 2 volts for input τ , so pick side-resistors for the potentiometer such that this is the case.

The relay shorts out the integrating capacitors, thus instantly resetting the analog computer to $\omega=0$ and $\theta=0$, a great advantage over a real motor & flywheel. Be sure you see why this works. (You can use FETs instead of relays if you really want to.)

THURSDAY. Build the circuit and monitor ω and θ on a scope. **I suggest you build it on solderboard, not protoboard. BUILD IN STAGES, get them working, then add the next** Leave space for your analog controller next week. Check that it works. Then hook up the reset line to a function generator with a period of about 4 seconds (and a short duty cycle - there is no need to reset for more than a moment). Trigger the scope on the reset signal.

Your challenge: control τ input (manually) to bring the θ output to 1 volt as quickly as possible, with as little overshoot as possible. You can practice over and over because the task repeats every 4 seconds. You are the controller! Good luck! Document your performance.

NEXT WEEK you will interface this to QNX and write a digital controller for this plant.

¹ Get – Pots, 741s, relays, capacitors, solder boards, zener diodes, 555's , 1uF capacitors, sockets

DIGITAL –

Please see <http://www.mech.nwu.edu/courses/433/>

PRIOR TO THURSDAY:

(1) If you aren't familiar with QNX, print out and read Unix (flavor: QNX) quickstart. This will tell you how to use the text editor, how to compile a program, how to list a directory and how to move around between directories, etc. When you use a QNX computer, login as "root" with password "xxxx". User "root" is very powerful and can do bad things like delete the operating system, so don't mess around. Make a directory for your work, named after you.

(2) If you aren't familiar with C, read a few pages of C programming quickstart or any C primer. C is much like other procedural languages, such as matlab or fortran. Write some simple programs, such as in the quickstart. It will help a great deal if you can get to a UNIX or QNX computer and try your programs. Don't worry about learning all of C - just getting a simple program to run, defining variables, making calculations, and printing results.

THURSDAY:

Edit, compile, and run a C program -- any C program at all

Write a program with a loop, and that can write to a file. Write a tab-delimited table of some calculation to a file.

Use ftp (e.g. cuteftp) from a Windows PC to retrieve your program & the file it wrote (QNX: inetd starts the ftp & telnet daemon). Windows: read in the data with matlab, and plot it.

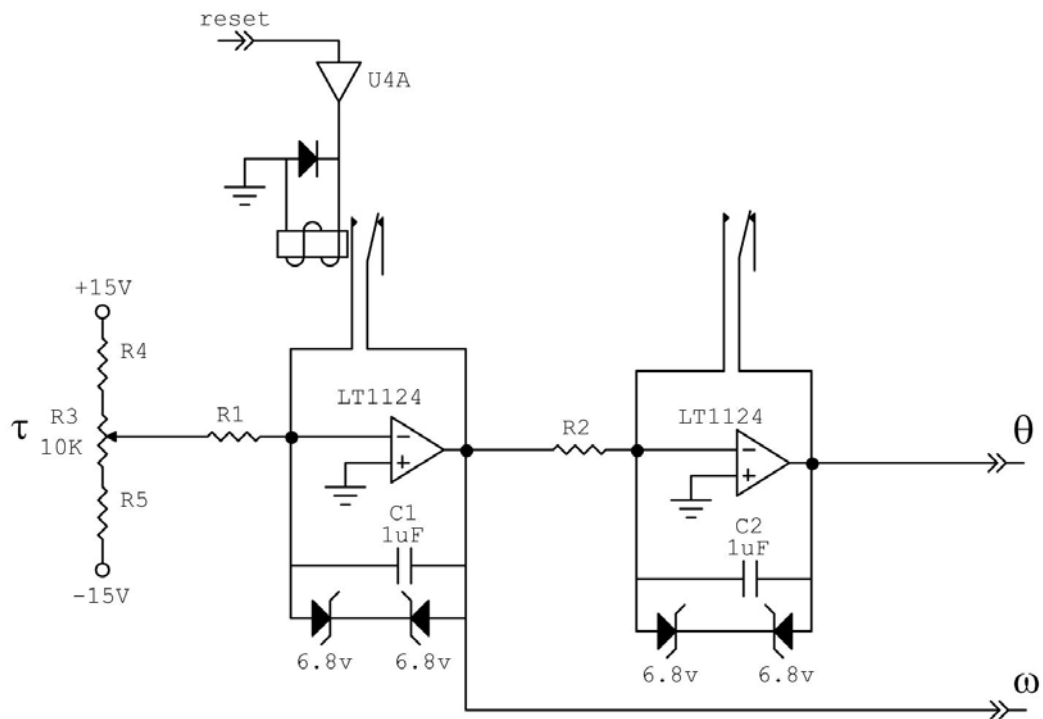
Connect an LED to one of the digital-out lines of the servotogo card, and write a program that turns it on and off

Connect a digital-out line to a digital-in line and show you can write/read digital signals.

Connect a DAC channel to an ADC channel and show you can write/read analog levels.

Connect two digital-out lines to an encoder channel, and show you can simulate the rotation of an encoder and read the counts. Make use of the timer-based delay somewhere.

Measure the current-voltage characteristic of an LED, and plot it.



Circuit notes:

The zener diodes are there to limit the velocity and the position so they saturate at about 7 volts. Zeners act like ordinary diodes in the forward direction, but they conduct in the reverse direction as well, if the voltage across them approaches their “zener voltage”. If you understand the relay, you'll understand the zeners too.

The buffer is there because your function generator, responsible for the reset pulses, probably cannot produce enough current to drive the relay directly. The 555 timer chip makes a good buffer.

The diode across the relay coil is there because when the relay coil is de-energized, its inductance can create a large "kickback" voltage spike which can damage electronics. The diode shorts this spike out.