CS 331 Lab #5
Balancing the Water Seesaw

Spring 2003

Week #1: 4/9/2003 – 4/11/2003
Week #2: 4/16/2003 – 4/18/2003
Report due by 5:00pm: 4/23/2003 – 4/25/2003

1 Overview

In this lab you will enhance your Lab 4 code so that it can dynamically balance the water seesaw. The Display task will now perform a PID computation and send the computed voltages to the I/O task, which will drive the motors.

You may base Lab 5 off of either your threaded (`threads.cc`) or process (`io.cc` and `display.cc`) code from Lab 4. The TAs recommend using the threaded approach, but the choice is up to you.

2 Procedure

1. Make a `lab5` directory to put your code for this lab in. Copy your desired starting point code from Lab 4 to the new directory. Write or modify a `Makefile` to compile this lab.

2. Turn on the op-amp box and execute ~`cs331/lab5/demo` to see an example of the balancing program. The following table describes what the nine displayed values are.

Label	Value
`a`	The seesaw's angle
`ad`	Derivative of the angle
`ai`	Integral of the angle
`fp`	Feedback from proportional control
`fd`	Feedback from derivative control
`fi`	Feedback from integral control
`f`	Total PID feedback
`vd`	Drain motor voltage
`vf`	Fill motor voltage

3. Read the coefficients K_p, K_d, and K_i as floating point command line arguments. You may find the `atof()` function helpful. See the manpage for details. To end up with "nicer" numbers on the command line, divide each of the specified arguments by 50 to get the actual K_p, K_d, and K_i values.

4. Update the Display task to perform a PID computation, calculate a voltage for each motor, and send the resulting raw 12-bit values to the I/O task as `unsigned char`s.

   (a) After computing the angle, numerically differentiate and integrate it. Recall that if $\theta(t)$ is the angle at time t, then the derivative is $\frac{d}{dt}\theta(t)$ and the integral is $\int \theta(t)dt$ where dt is the period of the I/O task.

   (b) Perform a PID computation to find the feedback F. Note that the task should be filled if $F > 0$ and drained if $F < 0$.

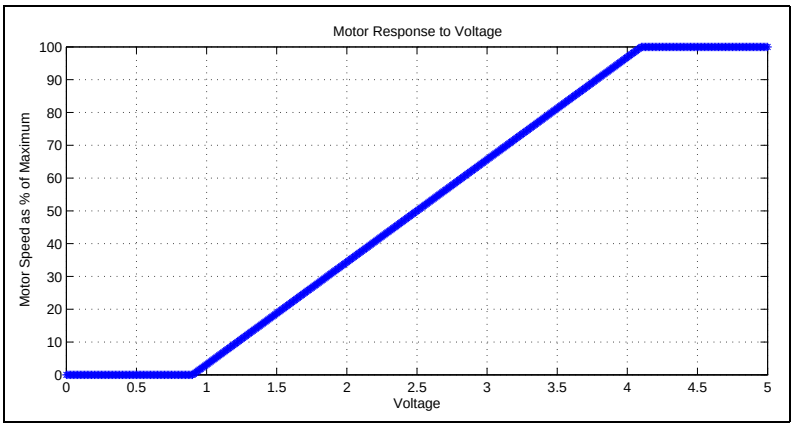

Figure 1. An approximate graph of motor speed vs. the applied voltage. Note that the response is approximately linear in the $1.0-4.0$ volt range.

(c) The PID control model assumes that actuators respond linearly to feedback. Our motors, however, do not respond linearly for all input voltages. For small voltages, the motor will not turn at all. And beyond a certain point, increasing the voltage will not increase the speed of the motor. See Figure 1 for a graph that you can assume approximately models our motors.

To make a linear system, we therefore use two such motors and operate each only within the $1.0-4.0$ volt linear region. The following piecewise equations show how to compute the fill motor voltage v_f and the drain motor voltage v_d from F.

$$v_f = \begin{cases} 1.0 + |F| & \text{if } F \geq 0 \\ 1.0 & \text{if } F < 0 \end{cases}$$

$$v_d = \begin{cases} 1.0 & \text{if } F \geq 0 \\ 1.0 + |F| & \text{if } F < 0 \end{cases}$$

(d) Once you have computed v_f and v_d as described above, ensure that $v_f \leq 4.0$ and $v_d \leq 4.0$.

(e) Convert the final v_f and v_d to 12-bit raw values and send the four resulting `unsigned chars` to the I/O task.

(f) Display the following six values:

- Angle (from Lab 4)
- Derivative of the angle
- Integral of the angle
- PID feedback F
- Fill motor voltage v_f
- Drain motor voltage v_d

5. Update the I/O task to output the motor voltages received from the Display task.

(a) To avoid jitter, perform all interaction with the A/D–D/A card together close to the top of the I/O task's periodic body.

(b) The fill motor is connected to D/A channel 0 and the drain motor to D/A channel 1. The corresponding port addresses are shown in the following table.

	Channel 0 (Fill)	**Channel 1 (Drain)**
Low Byte	0x300 + 4	0x300 + 6
High Byte	0x300 + 5	0x300 + 7

6. At the top of your program's `main()` function, send zero volts to both motors. This way, if your program somehow dies without turning off the motors, you can run it again quickly to stop everything.

7. Detect when Ctrl+C is pressed and send zero volts to both motors before exiting. This can be done by adding SIGINT to the `signals_to_wait_on` set and then checking the value that `sigwait(&signals_to_wait_on, &sigval)` writes into the integer `sigval`. If `sigval` = SIGINT, then Ctrl+C was pressed. Be sure to modify both `sigwait()` calls in this way.

8. Once the code has been updated, find "good" values for the coefficients K_p, K_d, and K_i that balance your seesaw reasonably quickly and without excessive thrashing. Note that all of the seesaws are physically different (friction, marble mass, motors, etc.), so your good values may be significantly different from those found by other groups.

9. Once your seesaw is balancing nicely, record its angle over time for the duration of one full good run. This may be done any way you wish. Two suggested possibilities are:

   (a) Redirect the regular output of your program to a file by appending "`> myfile.txt`" to the command line. Then use a tool such as Perl to parse out just the angle values.

   (b) Open a file in your program and write the angle to it directly from within the Display task. The functions `fopen()`, `fwrite()`, and `fclose()` may be used to do this. See the manpages for details.

   The reason for recording the angle over time is so that you can plot it for the Lab 5 report. See the end of section 4 for details.

3 Demonstration

Notify the TA when you are done with the assignment and ready to demonstrate your code. The TA may randomly pick one of you to do the demonstration and ask another to explain how it was done. *Make sure that everyone in your group understands how all of the code works before your demonstration.* Some lab problems will appear on the final exam.

4 Lab Report

The Lab 5 report is due 4/23/2003 – 4/25/2003, a week when there are no labs. The deadline is 5:00pm one week after the end of your Lab 5. You may hand it in to a TA during office hours or slide it under the lab door (1102 DCL). The report should include the following items:

1. An organized header that, at the minimum, contains your names and NetIDs, as well as your lab section (day and time), workstation, and username. For example:

   CS 331 Lab Report #5

   Tim Eriksson (eriksson@uiuc.edu)
   Xiaolei Li (xli10@uiuc.edu)

   Section: Friday at 10:00am
   Workstation: emb8
   Username: group38

2. An overview of your program implementation.

3. A discussion of any problems you encountered while working on this assignment, and how you overcame them.

4. Responses to the following questions.

   (a) What "good" values for K_p, K_d, and K_i did you arrive at?

   (b) What was your algorithm for finding these "good" values?

   (c) When varying the K_p, K_d, and K_i values, does the real seesaw respond like the Lab 3 MATLAB model did? How well do you think the MATLAB simulation models reality?

5. Use the angle over time data that you collected in the last step of the Procedure to generate an Angle vs. Time graph. Time $t = 0$ should be when you start running the program, and the plot should continue until the seesaw is balanced. The horizontal axis of the graph should be time in *seconds* and the vertical axis should be the angle in degrees. You may use any graphing tool such as MATLAB or Microsoft Excel to generate the plot. Include a full-page printout of your graph with the report.

6. A printout of your commented code.