EE443L Lab 8: Ball & Beam Control Experiment

Introduction:

The ball and beam control approach investigated last week will be implemented on the physical system in this week's lab. Recall the two part controller consisted of an "inner-loop" PD beam controller and an "outer-loop" PD ball controller.

Prelab:

1. Revisit your controller design and simulation from last week's lab. Run the simulation for your chosen gains and plot the motor armature voltage. Ensure that an excessive voltage is not required such that the controller can be implemented with existing lab power supplies. If excessive voltages are present, redesign your controller such that physically feasible values are achieved. This is most likely achieved by slowing down the response, i.e., moving the dominant poles to the right, which should have the effect of lowering the gains. Plot your resulting ball position, beam position, and motor armature voltage.

2. The "inner-loop" and "outer-loop" controllers are both of PD type and require differentiation if direct velocity measurements are unavailable. Beam velocity can be acquired directly through the motor encoder and Altera counters, but ball velocity will have to be determined by differentiating position measurements gathered from Sharp GP2D12 distance sensors. This differentiation will be performed in hardware using the op-amp circuit shown in figure 1. Determine the transfer function $V_{OUT}(s)/V_{IN}(s)$ assuming an ideal op-amp and discuss why it acts like a differentiator at low input frequencies. What happens to the output as the input frequency increases? Is this a good thing?

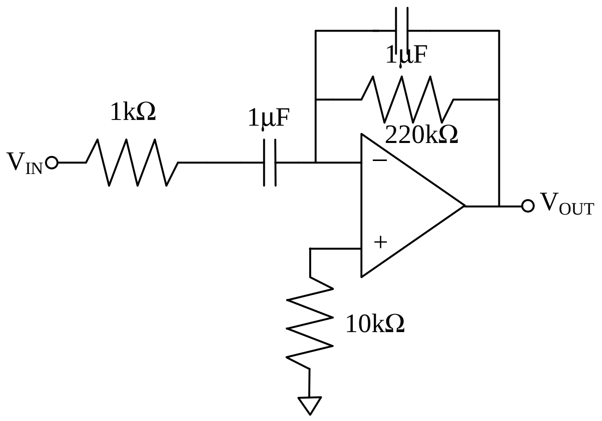

Figure 1: Analog Differentiation

3. The Sharp GP2D12 distance sensors yield a nonlinear distance versus output relationship. Therefore, one of our first tasks in lab will be to take many readings from the sensors and fit a polynomial to the data such that we can determine accurate distance values from sensor voltages. To become familiar with curve fitting, use matlab's *polyfit()* function to fit second and third order polynomials to the following data where *x* and *V* will be polynomial output and input, respectively.

V (V)	x (m)
0.5	0.00
0.6	0.05
0.8	0.10
1.1	0.15
1.5	0.20
1.6	0.25

Write down the resulting polynomials. Plot the raw data (from the table) as points and your second and third order polynomials for V = 0:0.01:2.5 on the same plot to check your curve fit. Comment on which order polynomial provides the better fit.

Laboratory Procedure:

This laboratory requires careful calibration of sensors and skillful implementation of the control approach in order to achieve success. Therefore, attention to details, neatness, and not rushing through the process will be imperative. Two weeks have been allocated for this laboratory to ensure everyone can take his or her time.

Part I: Sensor Calibration and Data Collection

1. **Characterize the Sharp GP2D12 proximity sensors for ball location on the beam.**

   a. Begin by downloading the LabVIEW VI *lab8.vi* from the network directory *N:/ee443l/Lab 8*. This VI contains only hardware-timed sampling of analog signals through the analog-to-digital converter. See the laboratory hardware connections to know which channels are sensor voltages and which are their hardware-computed derivatives.

   b. Place the ball on the beam from –30cm to 30cm in steps of 5cm and record the ball positions and corresponding voltages from both sensors using LabVIEW to display the voltages and compute their mean. Create a table of these values to include in your lab write up.

   c. Use matlab's *polyfit()* function to fit third order polynomials to both the right and left sensor data independently. Plot both the raw data and polynomial fit for both sensors to verify that the polynomials yield good position results. Utilize a higher order polynomial if the fit needs to be improved. Record the polynomials and include them as well as their plots in the lab write up.

   d. Enter both polynomials into LabVIEW using *formula blocks* to convert sampled sensor voltages to ball position. Place the ball on the beam at a variety of positions and display the ball position from both polynomials to verify their correct operation.

   e. Average the two ball positions from the two polynomials in LabVIEW to get one final ball position measurement *x* (this is the value we'll use for our ball position throughout the lab). Place the ball on the beam from –30cm to 30cm in steps of 5cm and record this ball position in your previously created table of values. Comment on the accuracy of your measured values (values within ±2cm should be possible).

2. Determine ball velocity from hardware differentiated sensor voltages.

   a. A transfer function representation of the op-amp differentiation circuit was found in the prelab. Compute the signed scaling factor that occurs in the differentiation process and reverse it in LabVIEW to get good representations of the proximity sensors' true voltage derivatives.

   b. Plot both voltage derivative measurements for a fixed ball position. Determine any offsets (biases) from zero in the values and add compensation in your VI to remove them.

   c. The relationship between proximity sensor voltage derivatives and ball velocity now needs to be obtained. Recall the polynomials found above relating ball position to sensor voltages. These polynomials can be written as *x(t) = P(V(t))* where *x(t)* is the ball position, *V(t)* is the sensor voltage, and *P()* represents the polynomial. Differentiating this expression with respect to time yields *dx/dt = [¶P/¶V][dV/dt]* which relates sensor voltage derivative to ball velocity. Write the two expressions for ball velocity *dx/dt* from both sensors.

   d. Place the velocity expressions into your VI using *formula blocks*. Plot both ball velocities *dx/dt* for a fixed ball position and check that they're zero.

   e. Take the average of these two computed velocities. This average value will be considered our ball velocity *dx/dt* from now on.

f. Clean up your VI (diagram and panel) such that only ball position and ball velocity are plotted on the same graph and verify that velocity does look like the derivative of position by moving the beam (by hand, with power supply off) so that the ball rolls back and forth. Show me plots of ball position and velocity and then place both the position and velocity calculations into a SubVI. This SubVI with have four inputs (two proximity sensor readings and their hardware-computed derivatives) and two outputs (ball position and velocity).

3. Determine beam angle from DAQ card counters.

a. Beam angle *q(t)* is the same as the motor position we've used in previous labs; therefore the same counter mechanism can be used for its measurement. Add the associated counter setup and read blocks and conversion factors to your VI such that beam angle is available in radians.

b. Change the trigger parameter in the counter setup from *rising edge* to *falling edge*. Check with me if you're not sure how to do this.

c. Plot beam angle *q(t)* and move the beam (by hand, with power supply off) to verify correct values.

4. Determine beam angular velocity from Altera counters with new reset rate.

a. Beam angular velocity *dq/dt* is the same as motor velocity we've used in previous labs except for a change in the Altera counter reset rate. In order to achieve better velocity measurement resolution, the Altera counter reset rate has been lowered by a factor of four to 122.07Hz. Add the associated port setup and read blocks and conversion factors to your VI such that beam velocity is available in radians/second.

b. Plot beam velocity *dq/dt* on the same graph as beam angle *q* and move the beam (by hand, with power supply off) to verify correct values.

5. Finalize data collection VI.

a. Take care to clean up your VI (diagram and panel) as this will be the starting point for Part II (controller implementation) of the lab. Display two graphs on the panel: one with ball position *x* and ball velocity *dx/dt*, and the other with beam angle *q* and beam angular velocity *dq/dt*.

b. For one last check of all measurements before implementing the controller, run your VI with the ball on the beam and rotate the beam by hand to verify measured values.

Part II: Controller Implementation

Note: For your safety and the well being of the equipment, keep a hand on the power supply switch at all times when running your controller(s). A lot of action can happen in a short amount of time, so be prepared to quickly turn off the power.

6. Add PWM generation and limiter.

a. PWM generation from specified armature voltage and overflow protection can be implemented similarly to that of previous labs. Add the blocks needed to convert armature voltage to a ±127 duty cycle assuming a supply voltage of 15V.

b. Remove any h-bridge dropout voltage compensation added in the armature voltage to PWM conversion. Our beam will be moving in both directions, so the (only) positive dropout voltage compensation will cause a less than desired amount of voltage to be applied when a negative voltage is required.

c. Set the power supply to 15V and send out a small PWM value (< 15) for a short time (< 0.5 second) to ensure proper operation, i.e., the motor and beam move.

7. **Implement PD beam angle control.**

a. Add PD beam angle control as shown in figure 2. Since beam velocity *dq/dt* measurements are available, use them to avoid taking derivatives in your controller (see figure 2).

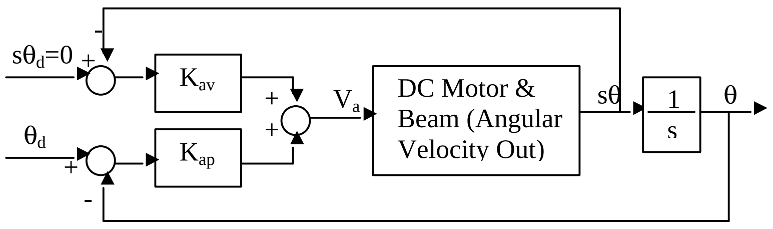

Figure 2: Beam Angle Control System

b. Begin with low gains (say $K_{ap} \le 20$, $K_{av} \le 1$) using fractions of those determined in the prelab and/or previous simulation lab. Vary gains (if needed) such that a reasonable beam step response ($T_s \le 2$sec, POS $\le 50\%$) is achieved for a desired angle of $q_d(t) = 0.3u(t)$ radians (approximately 17 degrees). Since we're performing a step response, let the desired beam velocity be zero, i.e., $dq_d/dt=0$.

c. Demonstrate this response to me and print out the recorded response from LabVIEW.

8. **Implement PD ball position control.**

a. Add PD ball control as shown in figure 3 to the PD beam angle control system of figure 2. Note ball velocity *dx/dt* is available, so differentiation is not required (see figure 3). Also, we'll use the common assumption that desired beam velocity is always zero ($dq_d/dt=0$) to avoid differentiation of the very noisy desired beam angle q_d. Theoretically, this will diminish performance, but practically it should prove better than if we differentiate a noisy q_d.

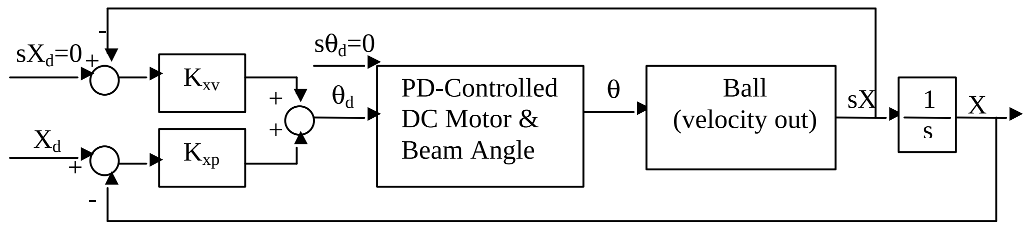

Figure 3: Ball Position Control System

b. Start your ball position controller with small gain values (say $K_{xp} \leq 1$, $K_{xv} \leq 0.5$) and gain ratios determined in the prelab. Tune the gains such that the ball can be positioned approximately 5cm from its initial value, i.e., $x_d(t) = 0.05u(t)$ meters, demonstrate the response to me, and print out the recorded LabVIEW response.