Experiment C8
Ball-and-Beam Experiment
Procedure

Deliverables: Checked lab notebook, In-Lab Demonstrations

Overview

The Ball and Beam is a classic mechatronics and controls lab experiment. Illustrated in Fig. 1, a ball rolls along an inclined plane. A servo motor is used to adjust the angle θ via a four-bar linkage, and feedback from an optical distance sensor are used to move the ball to a desired position x_s.

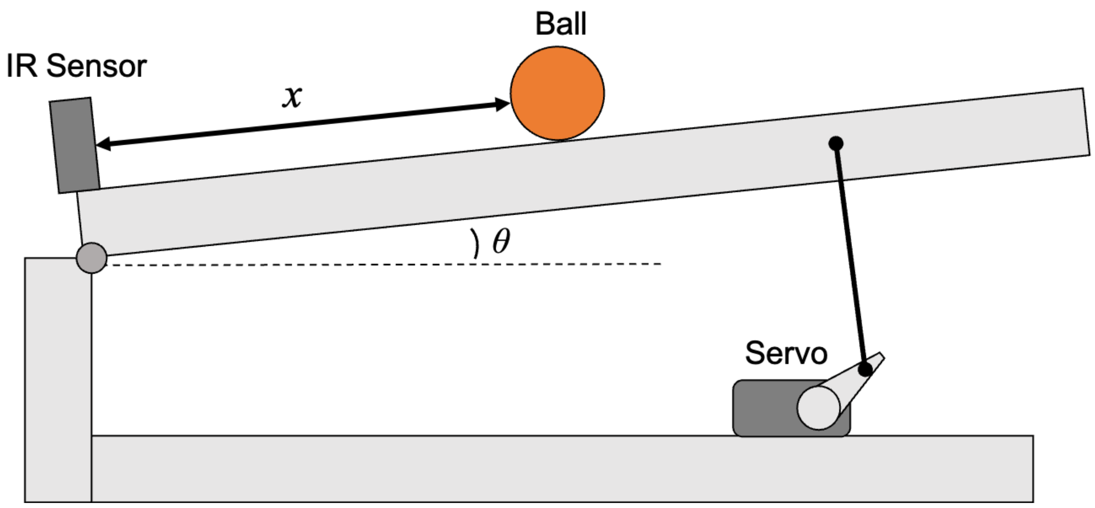

Figure 1 – The classic Ball and Beam setup uses feedback from an infrared optical distance sensor to move the ball at a desired position x_s by adjusting the angle θ.

Subsystem A: Mechanical Structure

You will begin by creating the 4-bar mechanism illustrated in Fig. 4.

Procedure

1. Attach the servo motor to the GoBilda servo block.
   a. All GoBilda hardware has both through holes and threaded holes for M4 screws. However, the servo disc is held to the servo shaft with an M3 screw.
   b. Various examples and ideas for mounting the servo blocks can be found on the product page linked below.

      https://www.servocity.com/compact-servoblock-43mm-width-for-standard-size-h25t-spline-servo-hub-shaft/

2. Use the second longest U-channel to create a sturdy base, and mount the servo to it as illustrated in Fig. 1. Note that its location will influence the kinematics of the 4-bar mechanism, with certain locations giving greater range and less sensitivity, and other locations giving less range with more sensitivity in θ.
3. Use the GoBilda hardware on the counter to attach the shortest U-channel to the base, creating the grounded "L" structure shown in Fig. 1.
4. Use a hinge to mount the longest U-channel to the structure, as shown in Fig. 1.
5. Complete the 4-bar mechanism using a large servo horn or rod coupled to a ball-end linkage.
6. Test the mechanical sub-system to make sure it works.
   a. You should be able to attain a range of $\theta = \pm 5°$ by rotating the servo horn between its limits.
   b. The mechanism should not pass through any singularities when rotating over this range. (None of the linkages should ever be parallel during rotation.)
7. **Demonstrate the working mechanism to the lab instructor or TA to receive credit on your score sheet.**

Subsystem B: Servo Motor

You will now electronically interface the servo motor with the Arduino microcontroller and find the quiescent position of the servo that levels the beam to $\theta = 0$.

Procedure

1. Use 3M command strips to mount the Arduino to the base of the mechanical apparatus. Peel the wax paper off the bottom of the small breadboard and adhere it adjacent to the Arduino.
2. Create Ground and +5V bus lines on the breadboard.
   a. Connect the 5V battery pack to the vertical bus lines. Per usual, red is +5V, and black is ground.
   b. Connect the Arduino 5V and GND pins to the bus lines.
   c. Connect the black and red wires of the servo to the bus lines. Per usual, red is +5V, and black is ground.
3. Connect the white or yellow signal wire on the servo to pin 9 on the Arduino.
4. The servo reads a 50 Hz digital pulse train, where the pulse width η corresponds to the servo angle.
   a. $\eta = 1000$ μs corresponds to the minimum angle
   b. $\eta = 2000$ μs corresponds to the maximum angle
5. Download the Servo Example code from the C8 webpage. Replace the *** in the code with different values between 1000 and 2000. Observe how this affects the position of the inclined plane.

6. Make sure the servo can adjust the inclined plane over a range of angles $\theta = \pm 5°$. You may need to remove the M3 screw holding the servo disc to the servo shaft and mechanically tune it to provide this range.
7. Determine the servo value η_0 that levels the inclined plane to a level quiescent state where $\theta = 0$. Test it with a bubble level tool. Test it again with the ball. Write the corresponding servo value down in your lab notebook
8. **Demonstrate the last two steps to the lab instructor or TA to receive credit on your score sheet.**

Subsystem C: Optical Distance Sensor

You will now implement an infrared (IR) optical distance sensor to measure the balls position *x*.

1. Read the IR sensor description on the AdaFruit product page.

   https://www.adafruit.com/product/1031
2. Connect the IR sensor to the breadboard +5V and GND bus lines, and use a DMM to test the analog output voltage. The voltage should decrease as the object gets further away.
   a. BLACK = GND
   b. RED = +5V DC power to sensor
   c. YELLOW/WHITE = Analog output voltage
3. Mount the IR sensor to one end of the ramp using any hardware of your choosing along with 3M double-sided command strips. Extra male/female dupont pin wires are available, if you find the sensor cables are too short.
4. Interface the IR sensor with the Arduino UNO.
   a. Connect the yellow/white analog voltage signal to pin A0 on the Arduino.
   b. Write a short script to read in the sensor data. Use the analogRead() to read in the sensor voltage as a 10-bit integer, and print the values to the serial monitor.
5. Use a tape measure to perform a 2-point calibration of the IR sensor. Put the ball on the track and measure the 10-bit sensor value at two known distances. Use the two data points to determine the linear calibration equation that relates distance *x* [cm] to sensor output *R*.
6. In your Arduino code, use the linear calibration formula to convert the 10-bit sensor output *R* to the distance *x* in centimeters. Print the distance values to the serial monitor. **Do NOT use the map() function**—it only outputs integers and will not give you enough resolution.
7. Use the tape measure to verify the system is measuring the correct distance to the ball in centimeters.
8. **Demonstrate the last two steps to the lab instructor or TA to receive credit on your score sheet.**

Design Challenge 1 – Proportional Feedback

You will now use the measured ball position *x* to compute the servo position with just proportional feedback $\eta = \eta_0 + k_p(x - x_s)$. There is not much natural damping in this system, so it will oscillate with only proportional feedback.

- Combine your codes from the previous sub-systems into a code that will measure the ball position *x* and compute the servo pulse width η using proportional feedback.
- Use a set-point $x_s = 50$ cm and the quiescent servo signal η_0 that you determined earlier.
- Estimate a value for the proportional gain k_p to be the maximum pulse range value of 1000 divided by the maximum displacement Δx_{max}.
- Start the ball near the setpoint $x_s = 50$ cm. Tap the ball and observe the dynamic response.
- Try different values of k_p to see how they affect the dynamic response of the system.
- **IMPORTANT:** When you are satisfied with the system performance, save the angle vs. time data from the Arduino serial monitor or PuTTY. **Demonstrate the system to the TA to receive credit on the score sheet.**

Design Challenge 2 – Proportional-derivative Feedback

Use the measured ball position *x* and speed $v = \dot{x}$ to compute the servo position with proportional-derivative feedback $\eta = \eta_0 + k_p(x - x_s) + k_d(v - v_0)$.

- Use a first order finite difference to compute the speed as the numeric derivative of position $v \approx (x - x_{prev})/(t - t_{prev})$.
- Numeric derivatives always amplify noise and uncertainty. Making the time step Δt too small will greatly exacerbate this problem. Use the formula $\Delta x = \frac{5}{14} g \sin\theta\, \Delta t^2$ to estimate the size of the time step Δt, where $\Delta x = 1$ cm is the approximate uncertainty in the sensor and $\theta = 20°$ is the maximum angle of the beam.
- You can also make the numeric derivative less noisy by computing a moving average of speed, where you save the speed from the previous iteration and average it with the speed from the current iteration.
- Estimate a value for the derivative gain k_d to be the maximum pulse range value of 1000 divided by the maximum speed you observe.
- It may be tricky getting the sign convention correct for the feedback gains. Think about it. Trial-and-Error it. Make it work.
- Try different values of k_p and k_d until you find values that quickly dampen the oscillations.
- Start the ball near the setpoint $x_s = 50$ cm. Observe the dynamic response.
- When you are satisfied with the system performance, save the angle vs. time data from the Arduino serial monitor or PuTTY. **Demonstrate the system to the TA to receive credit on the score sheet.**

Clean-up

To receive full credit, you must return the lab bench to its initial state:

- Disconnect all cables and sensors.
- Disassemble the ball and beam apparatus.
- Return all tools to their proper storage place.

Data Analysis and Deliverables

Using LaTeX or MS Word, make the following items and give them concise, intelligent captions. Make sure the axes are clearly labeled with units. Plots with multiple data sets on them should have a legend. **Additionally, write one or two paragraphs describing the plots/tables. Any relevant equations should go in these paragraphs.**

IMPORTANT NOTE: Add a horizontal line denoting the set-point whenever it is applicable.

1. A plot with the following distance vs. time traces (both traces on the same plot):
    a. The oscillation of the ball position with only proportional feedback.
    b. The oscillation of the ball position with well-tuned proportional-derivative feedback.

Appendix A

Equipment

- Standard Mechatronics lab kit in small tote (above lab bench)
- 40-20 Rails
- C-clamp
- 6mm Shaft Rotary Encoder 1024 P/R w/ mounting bracketSKU: RB-Spa-1042
- RoboGaia 3 Axis Encoder Counter Arduino Shield (SKU 00021)
- Cytron MD10C motor driver board
- 5202 Series Yellow Jacket Planetary Gear Motor, 19.2:1 Ratio, 24mm Length 6mm D-Shaft, 312 RPM
- GoBilda 1121 Series Low-Side U-Channel (5 Hole, 144mm Length)
- 1309 Series Sonic Hub (6mm Bore) – SKU 1309-0016-0006
- 1140 Series Aluminum Baseplate (6mm Thickness, 144mm Diameter)
- 1400 Series 1-Side, 2-Post Clamping Mount (12mm Bore)
- 4100 Series Aluminum Tube (10mm ID x 12mm OD, 300mm Length)
- Sky-Watcher S20540 Star Adventurer Counter Weight Kit