Optimal State Feedback Control (Ball and Beam)

by Carion Pelton

1 OBJECTIVE

In this experiment you will build and control a ball and beam system. Based on the equations of motion, you will build the theoretical nonlinear model in Simulink. The open loop response will be verified through simulation. Using MATLAB, you will linearize the nonlinear model, and simulate the open loop response for the linear model as well. Once the linear and nonlinear models have been verified, an optimal state feedback controller will be designed and simulated. Then you will test your optimal controller on the real system, and compare with the simulation results.

2 SETUP

2.1 REQUIRED MATERIALS

2.1.1 HARDWARE

All of the materials used in the *Sampling and Data Acquisition* experiment will be used in this experiment, except for the 3-D printed load. Additionally, the hardware listed below, and shown in Figure 2.1 and Figure 2.2 will be used.

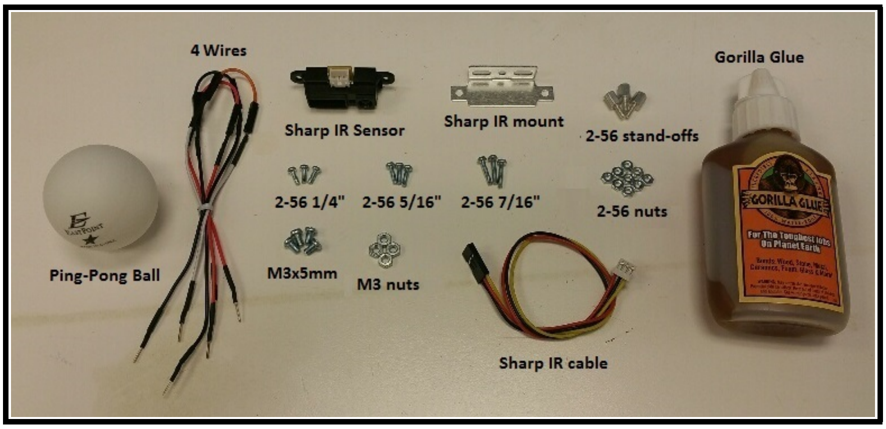


Figure 2.1: Additional Hardware

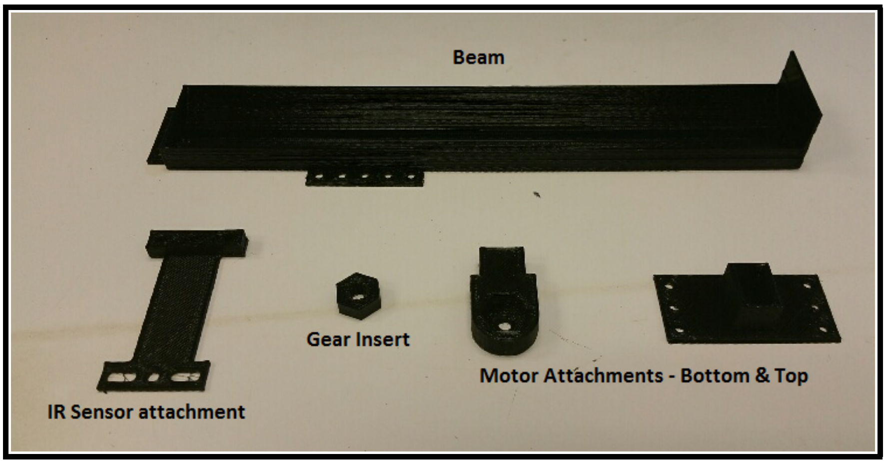


Figure 2.2: 3-D Printed Hardware

- Ping Pong Ball
- Sharp GP2Y0A41SK0F IR Sensor
- Sharp IR Mounting Bracket with 3 to 4 M3x5mm screws and nuts
- Sharp IR Cable
- 4 wires
- 9 - 2-56 Screws (3 - 1/4", 3 - 5/16", and 3 - 7/16")
- 9 - 2-56 nuts
- 3 - 2-56 Stand offs

- 3-D Printed Beam
- 3-D Printed IR attachment
- 3-D Printed Gear insert
- 3-D Printed Motor attachments (Top and Bottom)
- Gorilla Glue

2.1.2 SOFTWARE

- MATLAB/Simulink 2013a or later
  - ⋆ *The steps and images related to MATLAB/Simulink for this experiment were created using MATLAB/Simulink 2013a. Therefore some steps and images may be a little different if you are not using this version. If you are in fact using a different version, make sure you know the steps for running models onto the Arduino for your version of Simulink.*
- MATLAB/Simulink files
- 3-D Printer files

2.1.3 PREREQUISITE EXPERIMENTS

- Sampling and Data Acquisition

2.2 SOFTWARE SETUP

The necessary software files that are needed for this experiment can be downloaded from the Take Home Labs webpage. One method for downloading the files is shown in the steps below.

1. Open your internet browser and navigate to `thl.okstate.edu`
2. On the left side of the Homepage, select "All Experiments"
3. In the middle section of the All Experiments page select "Optimal State Feedback Control (Ball on Beam)"
4. On the Optimal State Feedback Control (Ball on Beam) page select "Software/Code" in the rightmost section. A zipfile named *SoftwareBallBeamOpt* should download.
5. Right-click the file and choose "Extract All...", or any other method of extracting files on your PC.
6. Extract this folder somewhere convenient, and remember the location. This will be the folder where all of the files and plots created for this experiment are saved.

7. Navigate back to the Optimal State Feedback Control (Ball on Beam) page, and select “3-D Printer Files" in the rightmost section.

8. Select the Optimal State Feedback Control (Ball on Beam) link to download the files. Open and 3-D print all of the “.STL" files. *Note:* Before printing each part, be sure to orient them correctly in the 3-D printing software. You should not need any support material. The correct orientations are shown in Figure 2.2, but you may have to orient the beam diagonally to make it fit on the print bed.

2.3 HARDWARE SETUP

2.3.1 BEAM CONSTRUCTION

9. The base hardware for this experiment is shown in Figure 2.3. If you do not have the hardware set up this way, follow the hardware setup steps provided in the *Sampling and Data Acquisition* experiment.

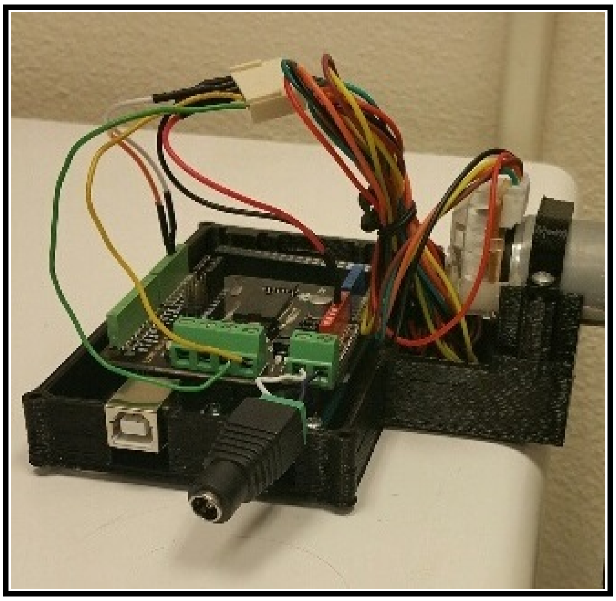


Figure 2.3: Base Assembly

10. Take the gear insert and make sure it fits on the motor’s gear, as in Figure 2.4. Then remove it from the gear. *Note:* You may need to use some force to fit the gear insert onto the gear.

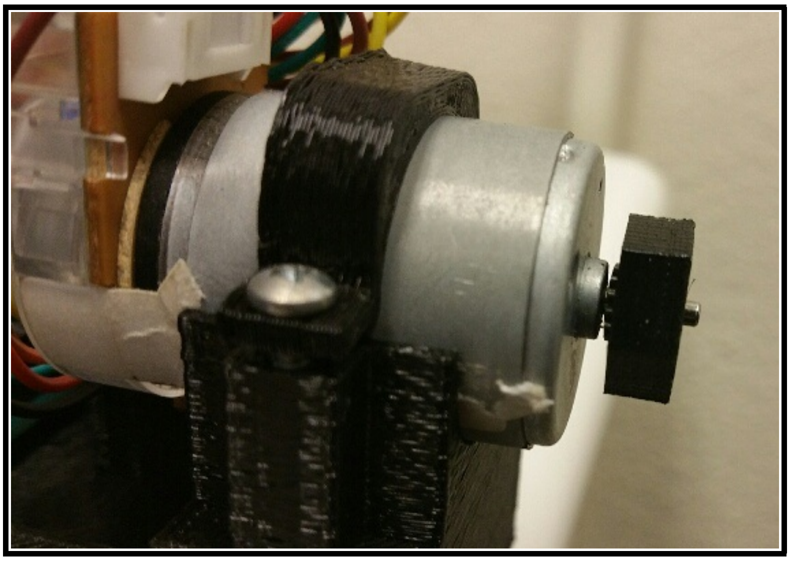
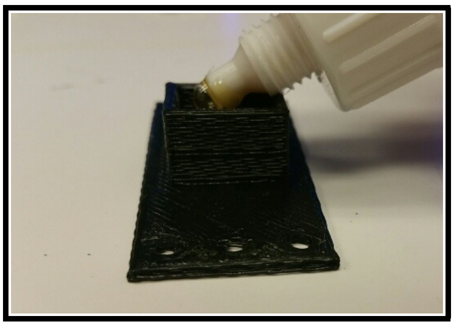


Figure 2.4: Gear Insert

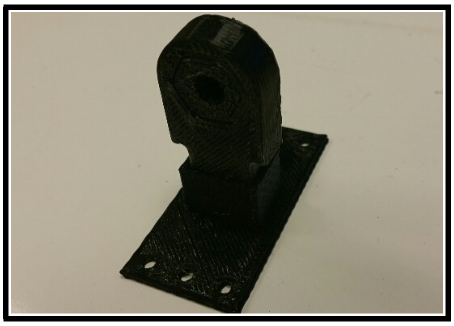
11. Place the gear insert inside the bottom motor attachment, as in Figure 2.5. *Note:* You may need to sand the sides of the gear insert to make it to fit inside the attachment.

Figure 2.5: Gear Insert inside Bottom Motor Attachment

12. Place gorilla glue inside the top piece of the motor attachment, as in Figure 2.6a. Place the bottom motor attachment piece inside the top motor attachment piece, as in Figure 2.6b. Give the glue time to dry before moving the piece. *Note:* You may need to sand the sides of the bottom motor attachment piece to make it to fit inside the top motor attachment piece.



(a) Motor Top Attachment



(b) Connecting Motor Attachments

Figure 2.6: Motor Attachments

13. Connect the standoffs to the end of the beam, as in Figure 2.7.

Figure 2.7: Beam Standoffs

14. Connect the IR sensor attachment to the standoffs using three 2-56 5/16" screws, as in Figure 2.8.

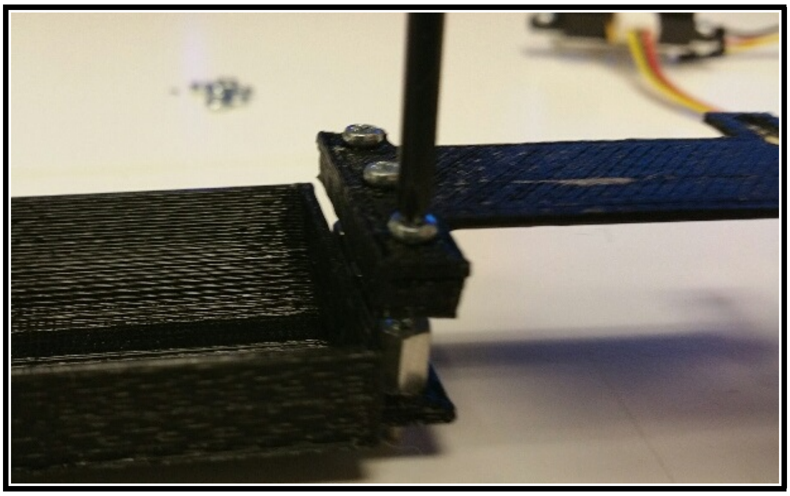


Figure 2.8: IR Attachment

15. Connect the IR sensor to the IR sensor mounting bracket using two M3x5mm screws, as in Figure 2.9a. Connect the IR sensor cable to the IR sensor, as in Figure 2.9b. *Note:* Remember the pin configuration of the IR sensor.

(a) IR Sensor Mount (b) IR Sensor Cable

Figure 2.9: IR Sensor Connections

16. Connect the IR sensor mounting bracket to the IR attachment, using one M3x5mm screw and nut in the middle hole, as in Figure 2.10. *Note:* You can also mount the IR sensor to the attachment using the two adjustable slots, which would require using two M3x5mm screws and nuts.

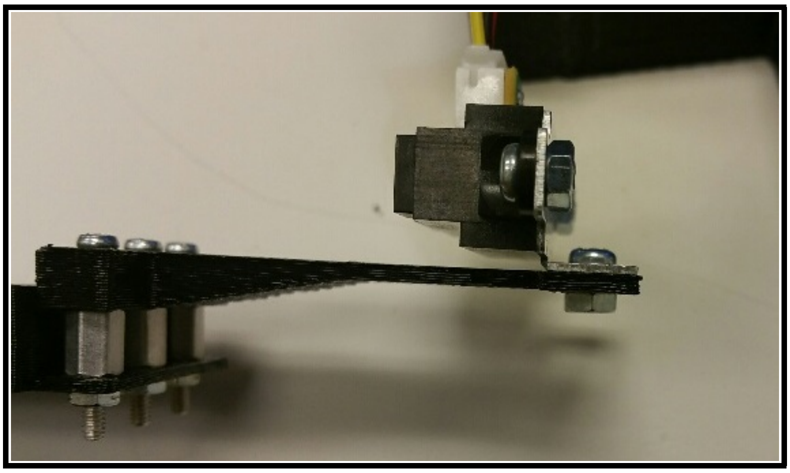


Figure 2.10: IR Connection to the Beam

17. Place the motor attachment, that you glued together, onto the motor's gear, as in Figure 2.11.

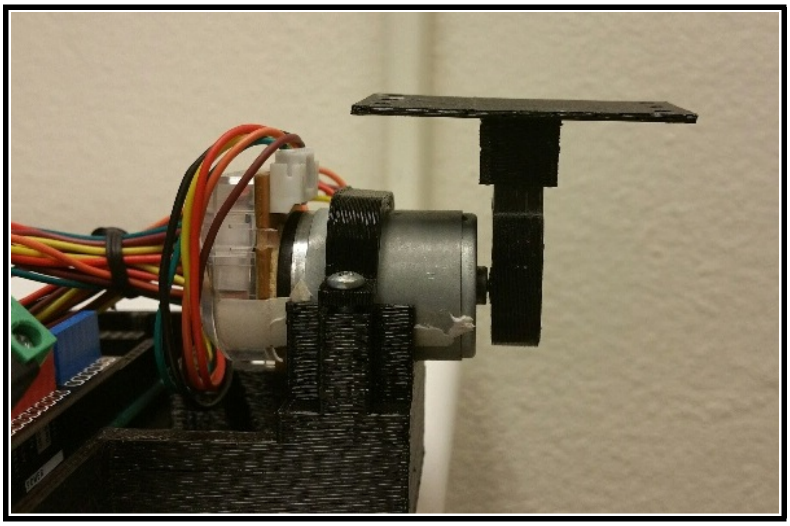


Figure 2.11: Motor Attachment to Motor

18. Take the beam and place it onto the motor attachment. Adjust the beam over the motor attachment's holes, so that the beam balances on the motor attachment, as in Figure 2.12.

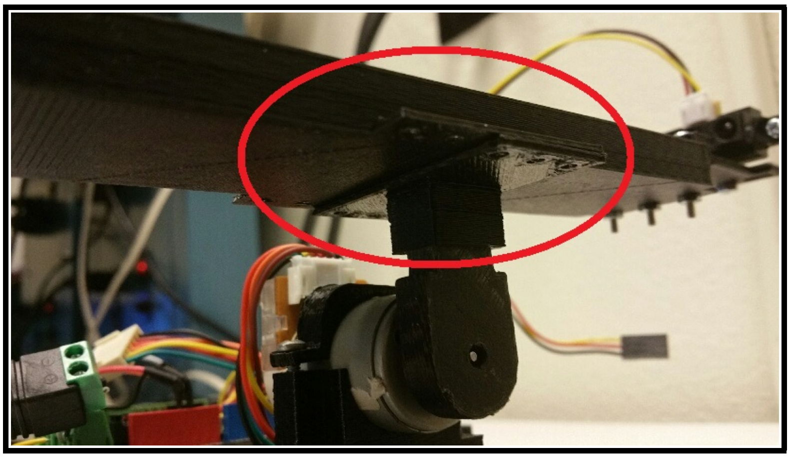
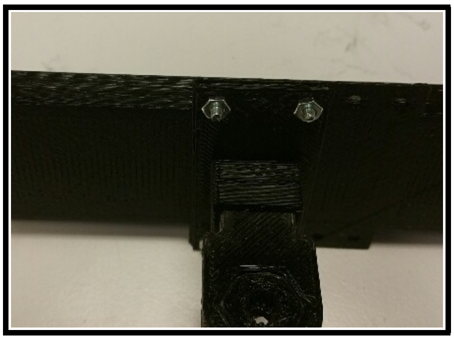
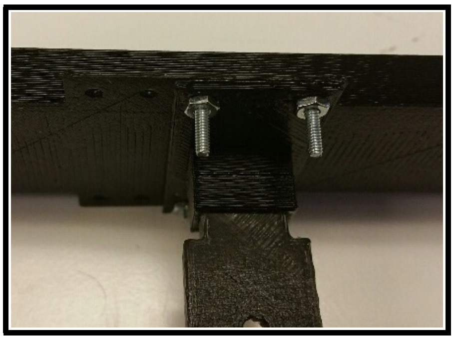


Figure 2.12: Attaching the Beam

19. When you find a position where the beam is able to balance on the motor attachment, screw the beam and motor attachment together, as in Figure 2.13a and Figure 2.13b. Use as many screws as you desire, but make sure to use the 2-56 1/4" screw on the side closest to the motor, and use the 2-56 7/16" screw on the opposite side.



(a) Shorter Screw Side



(b) Longer Screw Side

Figure 2.13: Beam Connection

20. Stop and ensure your setup resembles Figure 2.14.

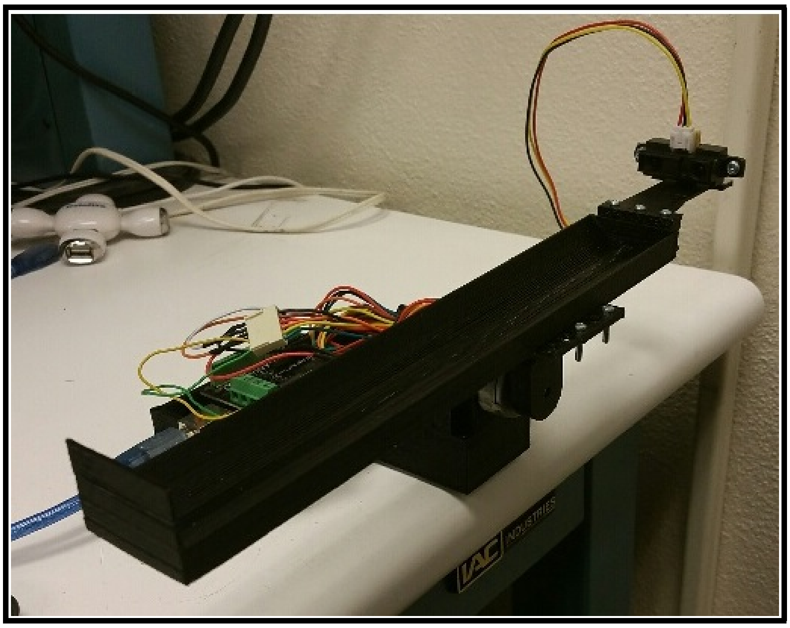


Figure 2.14: Complete Beam Construction

2.3.2 IR Sensor Connection

21. Place 3 wires into the end of the IR connector, as in Figure 2.15.

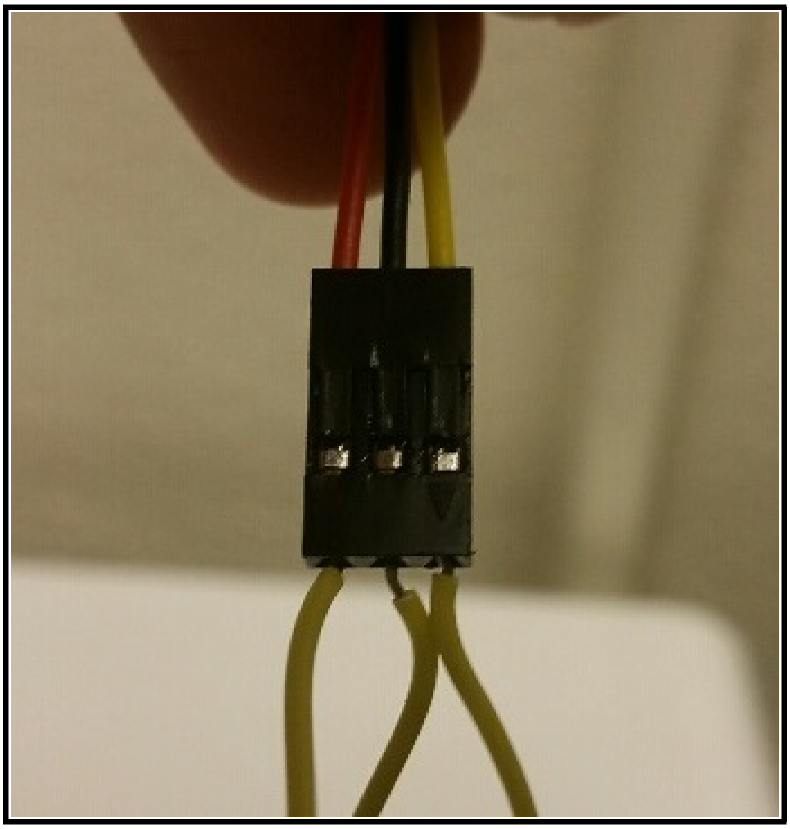
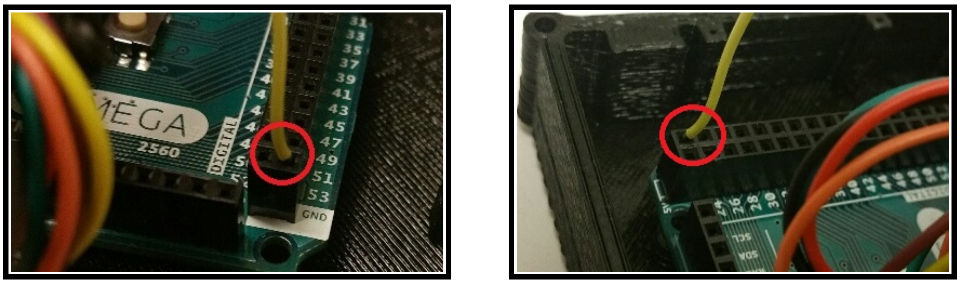


Figure 2.15: Wires for IR Connector

22. Connect pin 2 of the IR sensor to a GND pin on the Arduino, as in Figure 2.16a. Connect pin 3 of the IR sensor to a 5V pin on the Arduino, as in Figure 2.16b.



(a) GND Connection

(b) Vcc Connection

Figure 2.16: IR Sensor Connections

23. Connect pin 1 of the IR sensor to the Analog IN 0 pin of the Arduino, as in Figure 2.17.

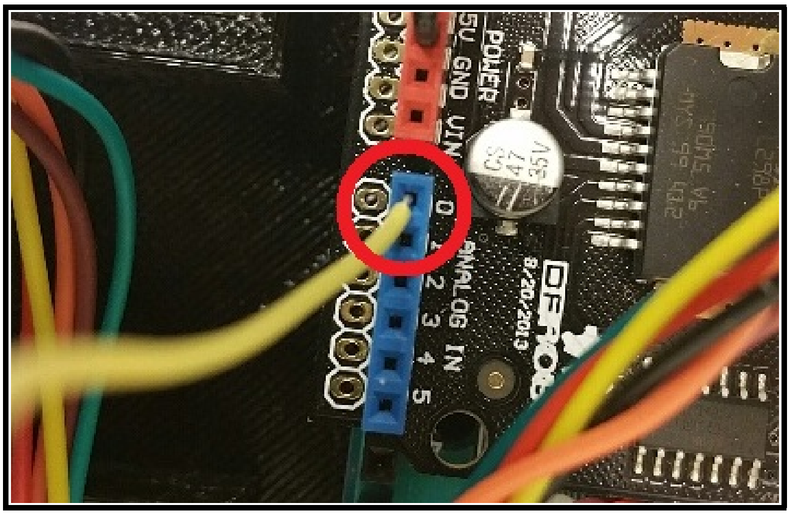
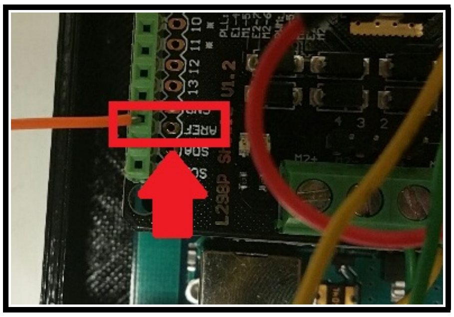
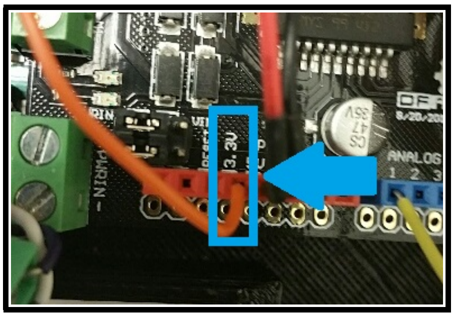


Figure 2.17: IR Sensor Vo Connection

24. Take another wire and connect it from the AREF pin on the Arduino to the 3.3V pin on the Arduino, as in Figure 2.18a and Figure 2.18b.



(a) AREF Pin



(b) 3.3V Pin

Figure 2.18: Arduino Analog Reference Voltage

25. Place the ball on the beam above the motor attachment, as in Figure 2.19.

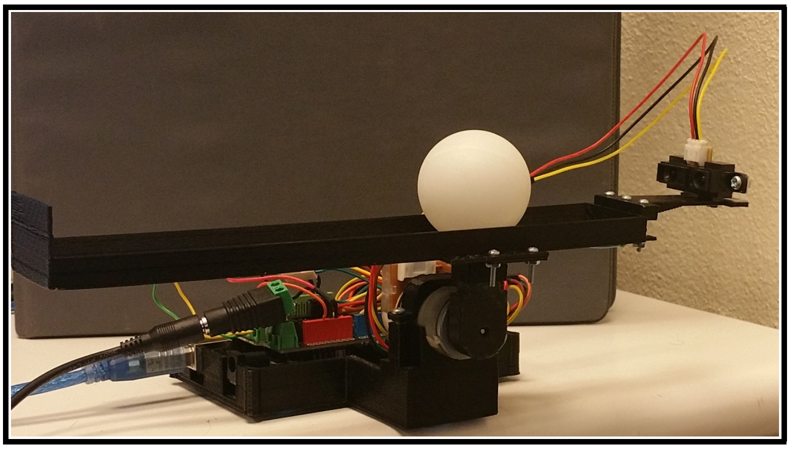


Figure 2.19: Ball and Beam

26. After adding the IR connection wires and the ping pong ball, the beam's center of mass point may be off. If this happens, you can add sticky tack to the bottom of the beam to adjust the beam's weight on either end, as in Figure 2.20.

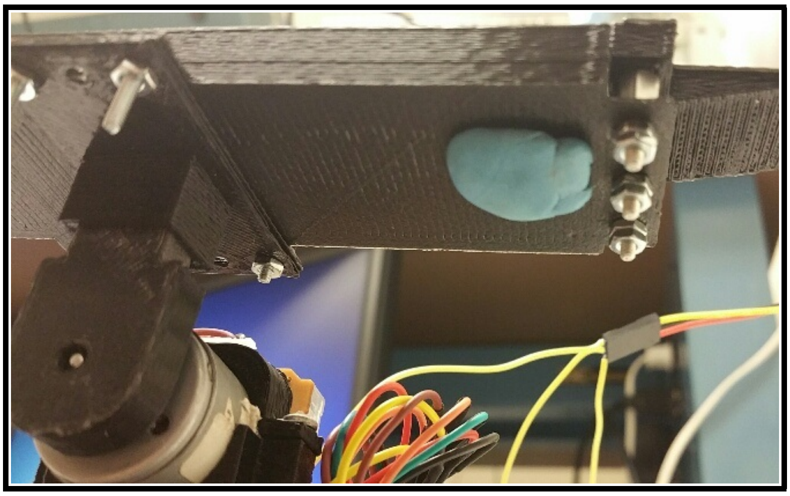


Figure 2.20: Adding Sticky Tack

27. If you added sticky tack to make the beam balance, with the ball above the motor attachment, the final assembly should look like Figure 2.19.

3 Experimental Procedures

In this section, you will build a Simulink model of the system, and simulate the open loop response to verify performance. Then you will develop a linear model of your system, simulate the open loop response, and compare the open loop response of the linear model to the open loop response of the nonlinear model. After verifying the open loop responses for both the linear and nonlinear model, you will design an optimal state variable feedback controller for the linear model. You will simulate the linear and nonlinear system responses with the added optimal state variable feedback controller. Then you will use the optimal state variable feedback controller to control the physical ball and beam system you built, by using the Arduino and Simulink.

3.1 Exercise 1: System Model

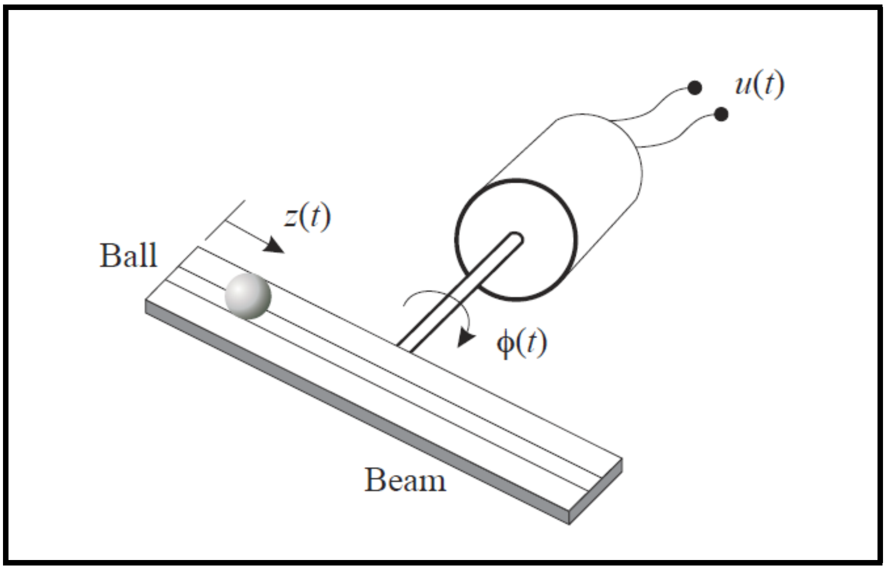


Figure 3.1: Ball and Beam Model

The equations of motion for the Ball and Beam are shown below.

$$\dot{x}_1 = \frac{\tau - mgcos(x_2)x_4 - 2mx_1x_3x_4}{J_{beam} + J_{ball} + mx_4^2} \tag{3.1}$$

$$\tau = \frac{K_t}{R_a}(u - K_b x_1) \tag{3.2}$$

$$\dot{x}_2 = x_1 \tag{3.3}$$

$$\dot{x}_3 = \frac{mx_1^2 x_4 - mgsin(x_2)}{\frac{J_{ball}}{r^2} + m} \tag{3.4}$$

$$\dot{x}_4 = x_3 \tag{3.5}$$

where

$$x_1 = \text{Beam Velocity}$$
$$x_2 = \text{Beam Position}(\phi)$$
$$x_3 = \text{Ball Velocity}$$
$$x_4 = \text{Ball Position}(z)$$
$$u = \text{Motor Voltage}$$
$$\tau = \text{Torque}$$

The constants are

$$K_b = 0.01 \text{ (Back EMF)}$$
$$K_t = 0.01 \text{ (Torque Constant)}$$
$$R_a = 30 \text{ ohms (Armature Resistance)}$$
$$J_{beam} = 6\times 10^{-5} \text{ Kg-}m^2 \text{ (Beam Inertia)}$$
$$m = 0.0027 \text{ Kg (Ball Mass}$$
$$r = 0.02 \text{ m (Ball Radius)}$$
$$J_{ball} = 7.2 \times 10^{-7} \text{ Kg-}m^2 \text{ (Ball Inertia)}$$
$$g = 9.81\ m/s^2 \text{ (Gravitational Constant)}$$

28. Make a Simulink model of the system.

29. Simulate the open loop response for 1 second, using an initial ball position of *0.1*m.

30. Plot each system state vs time.

31. Is the response what you expected? Explain.

3.2 Exercise 2: System Linearization

32. Create a linear model of the system using the MATLAB command “linmod". Linearize the model about the equilibrium point where all states equal 0.

33. Simulate the open loop response of the linear model, for 0.1 second and using initial ball positions of *0.1*m and *0.01*m. Compare with the responses of the nonlinear Simulink model. Plot each state.

34. How well do the linear model’s responses resemble the nonlinear model’s responses? How do they differ as the initial condition moves further away from the equilibrium point?

3.3 EXERCISE 3: OPTIMAL STATE VARIABLE FEEDBACK - SIMULATION

35. Assuming there is no noise in the system, and all states are measurable, design an optimal state variable feedback controller for your linear ball and beam model using MATLAB's "lqr" command. You will need to select the weighting matrices in the performance index. Your intial settings are not critical, as you can make changes later.

36. Simulate the linear and nonlinear closed loop systems for 5 seconds, with the added controller. Use an initial ball position of *0.1*m.

37. How does the response for the linear model differ from the response for the nonlinear model?

38. Do the input voltages for each response seem reasonable?

39. If needed, make any adjustments to your controller design by changing the weighting matrices to provide a fast response without exceeding physical limitations.

40. If all of the states were still exactly measurable, but there were disturbances in the system, how would the optimal gains change? Explain.

3.4 EXERCISE 4: OPTIMAL STATE VARIABLE FEEDBACK - EXPERIMENT

41. Open MATLAB and set the MATLAB "Current Folder" location to the *Experiment_OSFC* folder you extracted in the software setup section. The files *SerialPlotData.m*, *OSFC_Constants.m*, and *OSFCExpModel.slx* should show in the "Current Folder" section.

42. Open *OSFCExpModel.slx*. The model should now resemble the model in Figure 3.2.

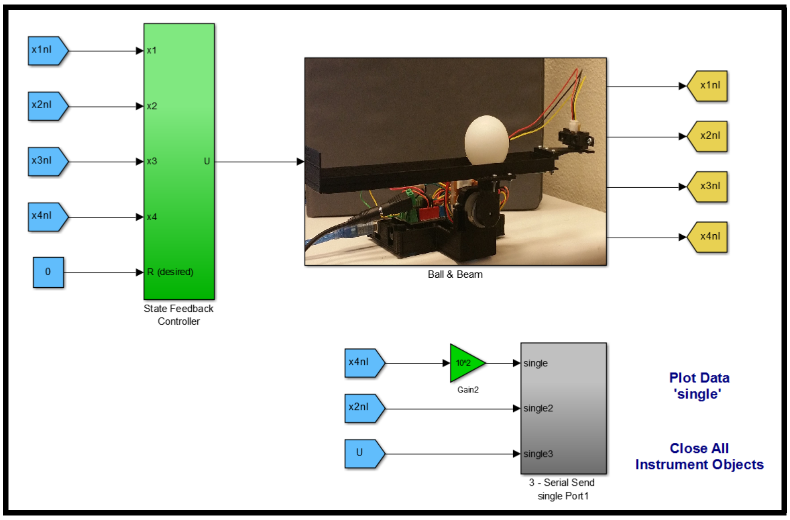


Figure 3.2: Experimental Model

43. Open the *OSFC_Constants.m* file, and add your optimal gain values into the variable “K".

44. Run  the file.

45. Set up the system so that it is balanced, with the beam parallel to the floor, and with the ball at the center of mass point, as in Figure 3.3.

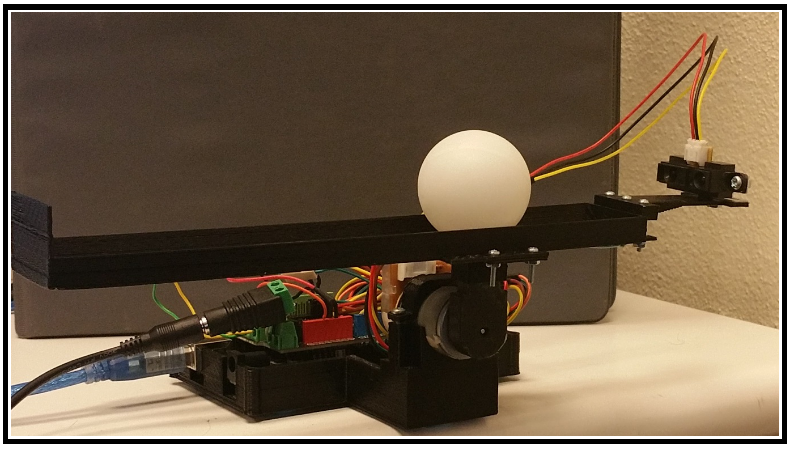


Figure 3.3: Ball and Beam

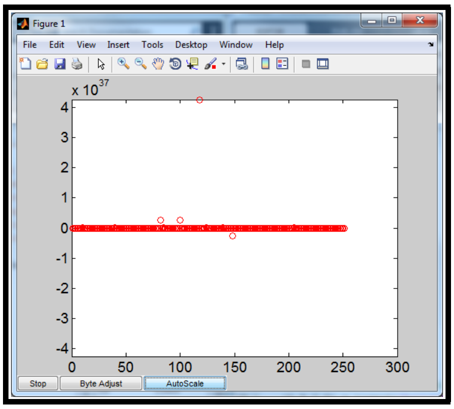
46. Connect the Arduino to the PC, but do not plug in the power supply. Download the Simulink model to the Arduino. The model will already be set up to run on the Arduino. If you receive an error about being unable to connect to the Arduino, you may need to set the COM port manually in the Configuration Parameters .

47. Once the model is successfully downloaded to the Arduino, select the “Plot Data ‘single’" text.

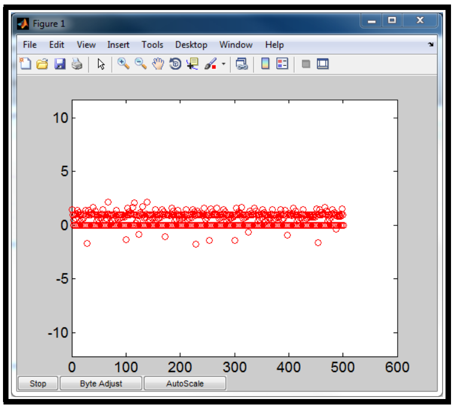
48. In the pop-up window, change the COM port to the port your Arduino is using, and set the number of samples to plot to *500*. Press “OK" to start plotting the data. The beam position, ball position, and input voltage are all being sent over the serial port, so there will be three plots showing in the figure.

49. Press “Autoscale". The beam position and ball position should be close to zero initially, and if the data is in sync, a small input voltage may be showing in the plot. If the data looks like Figure 3.4a, it is out of sync so press the “Byte Adjust" button. After the out of sync data passes, press “Autoscale" again. When the data becomes in sync it will be similar to Figure 3.4b. If your plot does not look similar to this, press “Byte Adjust" and “Autoscale" a few more times, if needed. After a few attempts, if the data is still out of sync, press “Stop" and close the plot window. Initiate the plotting process again by selecting the “Plot Data ‘single’" text and entering the number of samples to plot. Repeat

this step until the data is correct. Additional techniques for troubleshooting the serial data can be found in steps 79-82 of the *Sampling and Data Acquisition* experiment.



(a) Out of Sync Data



(b) In Sync Data

Figure 3.4: Experimental Serial Data

50. Once the data is in sync, ensure that the initial ball position is close to 0. If the ball position is not close to 0, rotate the IR sensor left or right, as in Figure 3.5, until the ball position is close to 0. After adjusting the sensor, be sure to tighten the screw again, and ensure the system is still balanced.

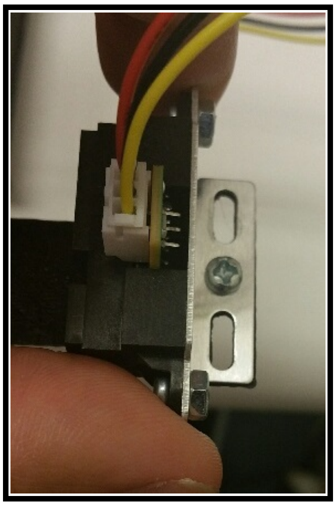


Figure 3.5: Adjust IR Sensor

51. Plug in the power supply.

52. The beam will likely not move since the ball is in the desired position initially, so lightly tap the ping pong ball in either direction. Try not to block the IR sensor as it may read

the distance to your hand instead of the ball. *Note:* Tapping the ball with too much force may cause it to bounce off either end of the beam, which may cause the system to become unstable.

53. If the ball does not balance, press “Stop" and redo your optimal state variable feedback simulation using different choices of the performance index. Repeat this until you are satisfied with the response.

54. Press “Autoscale" once the ball and beam start moving, if the values are not visible. Let the experiment run until the ball comes to rest after tapping it in either direction. Do this a few times.

55. Manually move the ball to the longer end of the beam, and then let the ball go. Press “Stop" once the ball balances.

56. After pressing stop in the serial plot window, open the *SerialPlotData.m* file in the MATLAB Current Folder section.

57. Press Run .

58. Describe the response of the system. Does the ball settle at the desired position exactly every time? If not, explain possible reasons for this.

59. Run the simulation again, but before plugging in the power, set the ball position close to the initial position used in the simulations (0.1m). Keep the beam around 0 radians, and hold it as steady as possible. An example of this is provided in Figure 3.6.

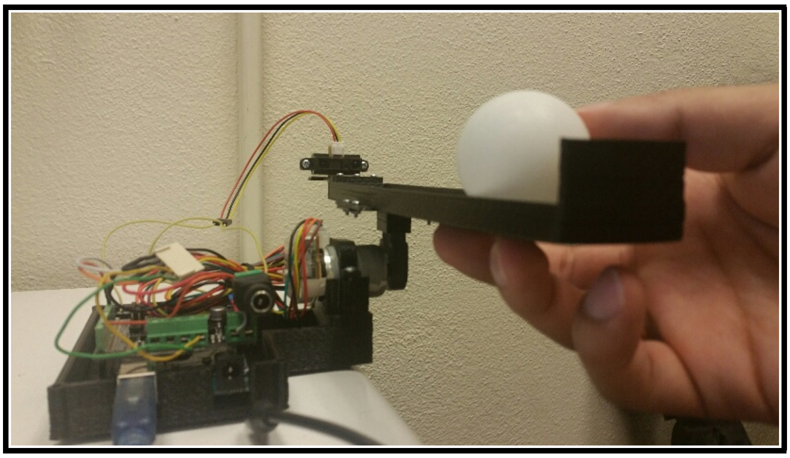


Figure 3.6: Ball Position Example

60. Plot and compare the output positions with the simulation results.

4 Conclusion

This experiment consisted of designing an optimal state feedback controller for a ball and beam system. Before testing anything on the actual system, the open loop response was verified for the nonlinear model. The linear model was found from the nonlinear model using MATLAB, and the open loop response for the linear model was found and compared to the nonlinear open loop response. An optimal state feedback controller was designed and simulated for the linear and nonlinear models. Then the controller was tested on the actual system and compared to the simulations.