Course Project Part 2: Hardware Testing Experimental Guidelines

Issued 4/13/'23
Due 4/30/'23

Note: This lab is based on material designed during EE C222/ME C237 in Spring 2022. Speical thanks to Instructors: Prof. Shankar Sastry and Prof. Koushil Sreenath and GSIs: Jason Choi and Kshitij KulKarni.

Disclaimer: This document is based on the Quanser User Manual for the Ball and Beam system. Distribution of this file without the instructors' permission is strictly forbidden.

1 Components

The components of the Ball and Beam module are listed in Table 1 and labeled in Figure 1.

ID	Component	ID	Component
1	Rotary Servo	7	Support arm
2	Lever Arm	8	Support base
3	Coupling screw	9	Ball position sensor connector
4	Steel ball	10	Support arm screws
5	Ball and Beam Potentiometer sensor	11	Calibration base
6	Ball and Beam Steel rod		

Table 1: Listing of Ball and Beam components

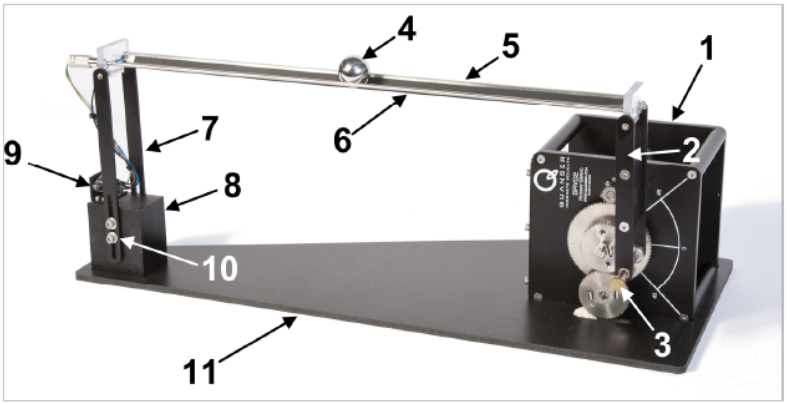

Figure 1: Components on Ball and Beam system

2 System Setup

2.1 Calibration of the support arm height:

The main purpose of this calibration is to make sure that the beam is horizontal when the servo motor angle θ is 0.

1. Tighten the coupling screw into the screw hole of the large 120-tooth load gear as depicted in Figure 2.
2. As illustrated in Figure 2, manually rotate the servo load gear to the 0 degree position. The coupling screw should be aligned with the 0 degree position on the servo.
3. See Figure 3. While holding the load gear at 0 degrees, place the ball in the center of the beam and vary the height of the support arm such that the beam is approximately horizontal and the ball does not move.
4. Tighten the 4x screws on the support arm, as illustrated in Figure 4, to finalize the calibration of the Ball and Beam.

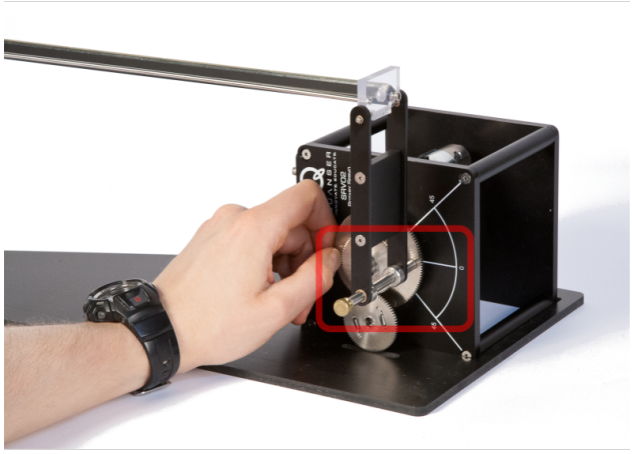

Figure 2: Attach Ball and Beam to Rotary Servo and rotate the gear to 0 degrees.

Figure 3: Adjust height of support column until ball is balanced at middle of beam.

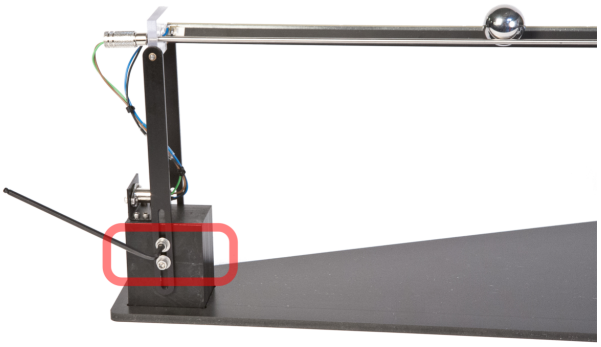

Figure 4: Tighten 4x screws on column to fix the height.

2.2 Wiring Procedure

In order to operate the Ball and Beam, it has to be connected to a data acquisition board and a power supplier. Most of the wiring is already done for you. Before you start the experiment, make sure to go through the following steps to finalize the wiring setup.

1. There are mainly two type of setups (Figure 5):
   (a) **Type 1** consists of VoltPAQ power supplier and Q2 Data Acquisition (DAQ) board.
   (b) **Type 2** consists of UMP_1503 power supplier and Q4 DAQ board.
2. Before you touch anything, always make sure that the power supplier is turned off.
3. Plug in the encoder cable with the tag 'Encoder0' to the Encoder connector of the Rotary Servo (Figure 6).
4. Plug in the motor power cable with the tag 'Motor' to the Motor connector of the Rotary Servo (Figure 6).
5. Turn on the power supplier.

2.3 Setting up the PC environment and Testing

You can double-check that the hardware and the wiring is arranged properly by checking if the PC can correctly receive sensor measurements from the hardware:

1. Set up your Windows instructional accounts via WebAcct and log into the PC. Open Matlab.

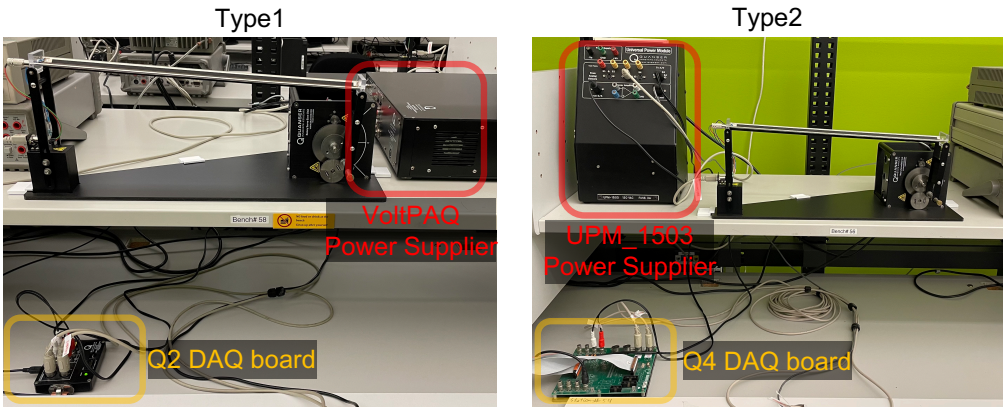

Figure 5: Two different type of hardware setup.

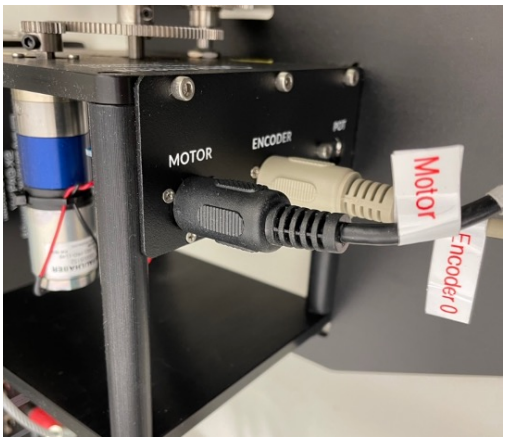

Figure 6: Motor power cable and encoder cable connected to the Rotary Servo.

2. Download the starter code from https://github.com/HybridRobotics/ball-and-beam-project. Replace **studentControllerInterface.m** with your controller. Run **setup.m** or add the repository and the subfolders to the matlab path.

3. Open **setup/setup_simulink_experiment.m** in the editor and run it. Make sure **setup_type** in line 3 matches the type of the machine you are using.

4. Open **model/simulink_experiment_test_type#.slx**, with # being the setup type of your machine.

5. Click the icon **Build Model**. Once the build is done without any error, click the icon **Connect To Target**. Finally, click the icon **Run** to start the test. (Figure 7).

6. Open the **Ball Position** plot and check if you are receiving the correct ball position measurements by manually moving around the ball position. The value should be -20cm when the ball is closest to the Rotary Servo, and 20cm when the ball reaches the other end of the beam.

    - **Note:** The ball position sensor often gives erroneous values when the ball is

attached to the servo-side end of the beam. This is one reason why you should prevent the ball from hitting the edges of the beam.

7. Open the **Servo Angle** plot and check if you are getting the servo angle measurements from the encoder. There will be no voltage applied to the servo so manually move around the Coupling screw (within the servo motor's operation range indicated on the base) and check if the received values are correct.

8. Click the icon **Stop** if you want to stop the test or if you run into any issue.

9. **Troubleshooting:**

   - If you are not receiving any measurements, make sure that your wiring is correct (Sec.2.2) and the power supplier is turned on.
   - If the data transmission is working properly, the green LED in the DAQ board will blink when you are running the test.
   - If there is an offset in the measured ball position, consider calibrating it by adjusting the **Position Bias** value in **Ball and Beam Hardware Interface** block. Note that changes made in one model (.slx) file does not affect the other models.

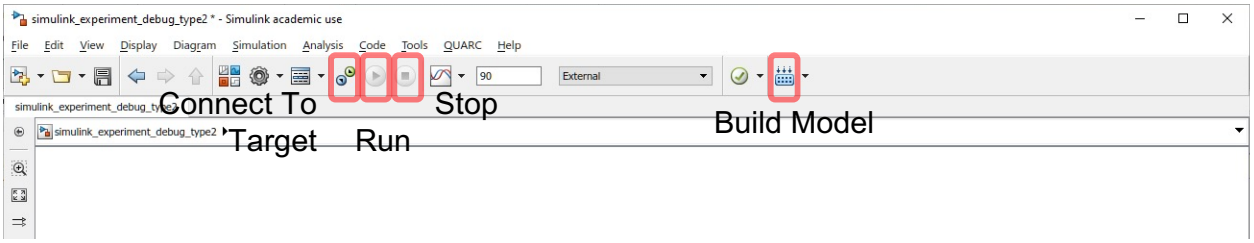

Figure 7: Simulink GUI.

3 Experiments

3.1 Safety Rules

In order to make sure that your experiment is conducted safely, please abide by the following rules. Violation of any of the rules will be subjected to a severe penalty in the project evaluation.

1. You are not allowed run the experiment when the GSIs are not present in the lab.

2. At least two students have to be present in the lab for each team when running the experiment. It is strictly forbidden for an individual team member to run the experiment without a partner.

3. The role of the first member is to monitor and operate the PC, and the role of the other is to monitor the hardware while the experiment is running.

4. Under an emergency (hardware malfunction, fire hazard, etc.), the one who is monitoring the hardware should immediately shut down the power of the power supplier.

5. Always make sure to turn off the power supplier when you are done with the experiments.

6. Any hazardous cases or hardware breakdown must be directly reported to the GSIs.

3.2 Running experiments for debugging your controller

1. Open **setup/setup_simulink_experiment.m** in the editor. Make sure **setup_type** in line 3 matches the type of the machine you are using.

2. Open **model/simulink_experiment_debug_type#.slx**, with # being the setup type of your machine.

3. Click the icon **Build Model**. Once the build is done without any error, click the icon **Connect To Target**. Finally, click the icon **Run** to start the test. (Figure 7).

4. Click the icon **Stop** if you want to stop the experiment or if you run into any issue.

5. After the stop, the recorded data (ts, ps, thetas, ref_ps, us) will be saved to the matlab workspace.

6. Run the following script in the matlab command window to evaluate the score of your controller.

```
score = get_controller_score(ts, ps, thetas, ref_ps, us);
```

7. Run the following script in the matlab command window to plot the data.

```
% Plot outputs.
plot_outputs(ts, ps, thetas, ref_ps);
% Plot output errors.
plot_tracking_errors(ts, ps, ref_ps);
% Plot control input history.
plot_controls(ts, us);
```

8. Similar to how you used simulations, play around with different reference trajectory profile under various values of amplitude and period by modifying **get_ref_traj.m**, and debug and improve the controller.

3.3 Running experiments for reporting the score

1. Clear the matlab workspace (type 'clear all' in the matlab command window).

2. Before running this experiment, remove **get_ref_traj.m** or change the file name temporarily. **Your controller will not use the proper reference trajectory for evaluation if you skip this part.**

3. Run the following command in the matlab command window, with **SETUP_TYPE** being your machine type, **STUDENT_ID** being your student id, and **TEAM_NAME** being your team name. The command will run the experiment after building the simulink model for evaluation (this might take a while).

```
run_experiment_to_leaderboard(SETUP_TYPE, STUDENT_ID, TEAM_NAME)
```

   - **Note:** This command will give you an error if you run it in your local PCs.

4. If anything goes wrong during the experiment, turn off the power supplier immediately. Close the matlab and relaunch it.

5. If the experiment is done successfully, you will get the recorded data (ts, ps, thetas, us) and the score saved in your workspace. Also, the score will be reported to the leader board online.

6. Run the following script in the matlab command window to plot the data.

```
% Plot outputs.
plot_outputs(ts, ps, thetas);
% Plot control input history.
plot_controls(ts, us);
```

Deliverable

As well as loading your lowest score to the leaderboard, please compile a short report (2-3) pages under the following section headings

- **Controller that achieved the best score** Please compare the two (or more) controllers you designed in simulation and reason why one may have performed better on hardware.
- **Model/plant mismatch** Please discuss the performance of each of your controllers on hardware relative to their performance during simulation (for a common reference trajectory). What were the main issues you faced implementing your controller on hardware? How much tuning was necessary?
- **Lessons learned** How did you find the hardware testing? Were you surprised by the difference between actual vs simulated performance? Is there anything you would do differently if you were starting this lab again? Do you have any feedback for repeating this lab again in the future?

Notes

An updated version of run_experiment_to_leaderboard is being prepared for this class and will be available shortly. The current function will return an error if used.