# 

# 

# 

# 

# 

# **ENEE 461 Laboratory 4: Stabilizing the Inverted Pendulum**

# 

# **Name: Elick D’Rozario**

# **Partner: Adrian Fernandez De Leon**

# **Section: 0101**

# **Lab date: April 3, 2023**

# 

# 

# 

# 

# 

# 

# **Introduction**

In this lab, we will be conducting an experiment to stabilize an inverted pendulum. This pendulum is connected to a servo. The point of this experiment is to move the servo in such a way so that the pendulum goes against gravity and is always pointing upwards. An open-loop controller cannot be used in this experiment because it is clearly unstable.

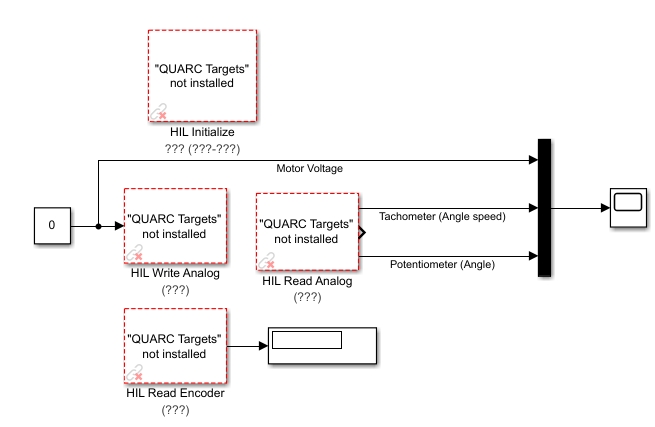
# **Hardware/Software Used**

1. Q8-USB or PCI
2. Desktop computer with MATLAB
3. Quanser Universal Power Module
4. Quanser Linear Voltage Amplifier (LVA) (depending on which bench you are using)
5. Quanser Q8-USB Data Acquisition Board (DAB)
6. Quanser SRV-D2 Plant

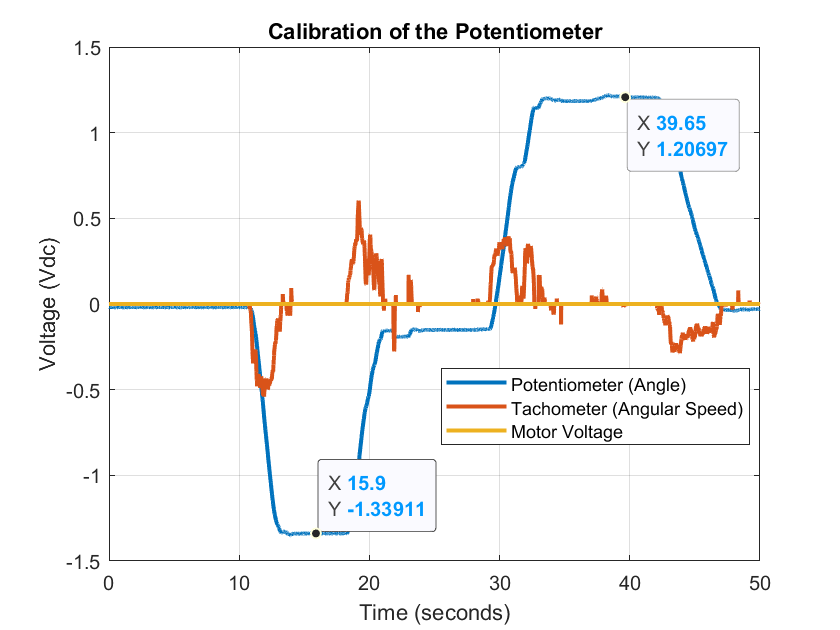
**Results**

Calibrating the Potentiometer:

To begin the experiment we created an open-loop controller whose input was zero in order to calibrate the potentiometer. However, before doing so we ensured that the angle was as close to zero as possible. This can be seen in figure 2 where the angle is initially very close to zero. We then manually moved the control arm to measure the potentiometer’s voltage readings at 45 degrees and -45 degrees. The voltage at 45 degrees was -1.34 volts and the voltage at -45 degrees was 1.21 volts. Additionally, the voltage at 0 degrees was 0.02 volts. These voltages are displayed in figure 2. Overall, this portion of the experiment allowed us to analyze what voltage readings are associated with angles 45 and -45 degrees.



*Figure 1: Open-loop controller of the system used to calibrate the potentiometer. The motor voltage was set to zero in order to manually turn the control arm. This was done to measure the voltage readings at 45 degrees and -45 degrees.*



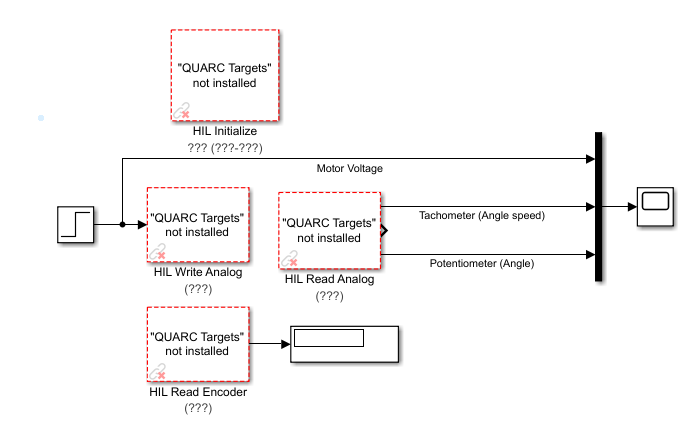
*Figure 2: Graph showing the different potentiometer voltages associated with angles 0, 45, and -45 degrees. This was done by manually turning the control arm to the marked angles of the SRV-D2 housing.*

Calibrating the Tachometer:

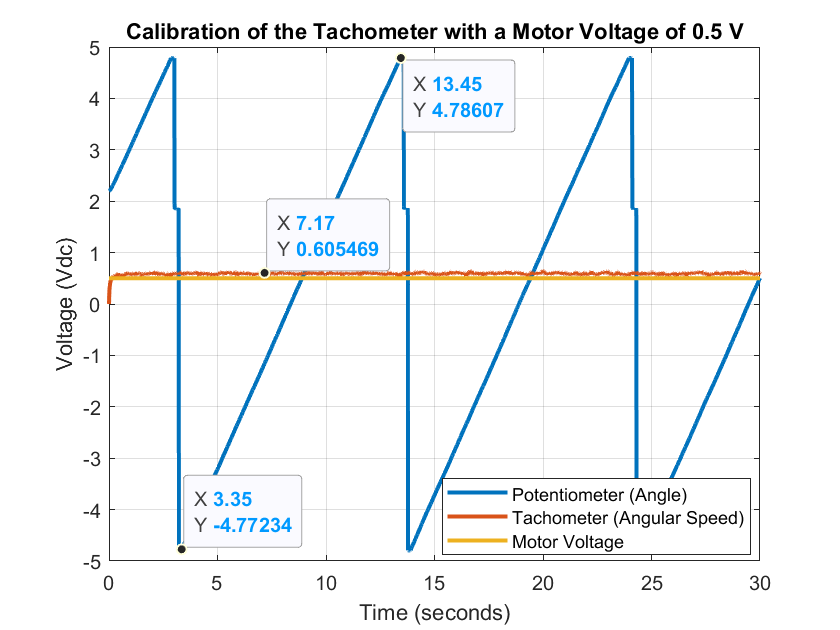
Next, we focused on calibrating the tachometer by removing the pendulum which allowed us to apply a constant voltage to the motor. This constant motor voltage allowed the control arm to turn at a constant velocity. We executed this experiment for multiple different motor voltages including: 0.5, 1.0, 1.5, and 2.0 volts. Doing this allowed us to analyze both the potentiometer voltage and the tachometer voltage. The slope of the potentiometer voltage corresponds to the velocity of the angle and the tachometer voltage corresponds to the angular speed. To calculate the slope of the potentiometer we simply use the formula . We perform this calculation and compare it to the measured tachometer voltage. An example of this calculation is shown below for when the motor voltage was 0.5 volts. We find that the slope of the potentiometer voltage and the tachometer have a percent difference of around 55% no matter the motor voltage. The results are summarized below in table 1.

| **Motor Voltage** | **Slope of Potentiometer Voltage** | **Tachometer Voltage** | **Percent Difference** |
| --- | --- | --- | --- |
|  |  | 0.60 | 54.33 |
|  | 2.173 | 1.40 | 55.21 |
|  | 3.447 | 2.20 | 56.68 |
|  | missing | 3.12 |  |

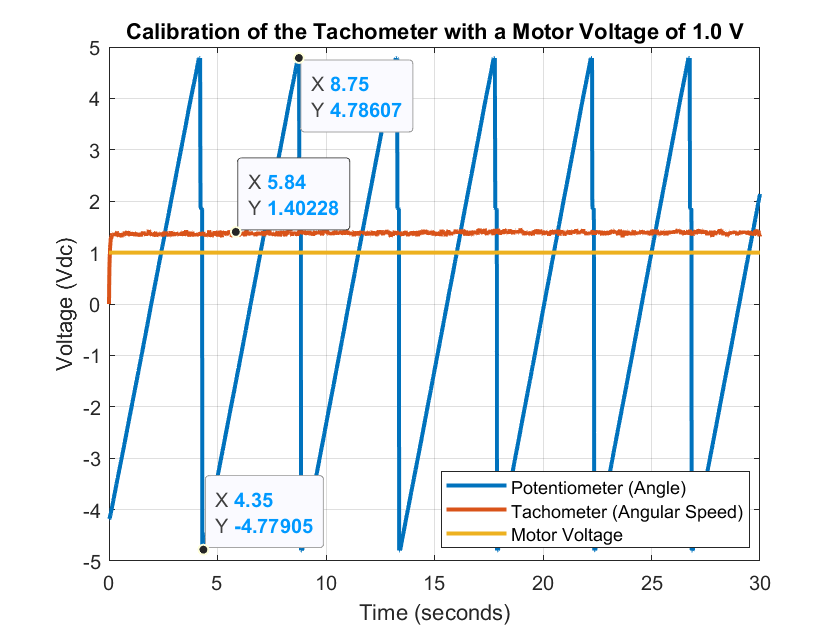
*Table 1: Table which displays the slope of the potentiometer voltage and the tachometer voltage in the face of a changing motor voltage. As can be seen the percent difference between the two voltages is roughly 55% for all four motor voltages tested.*

**

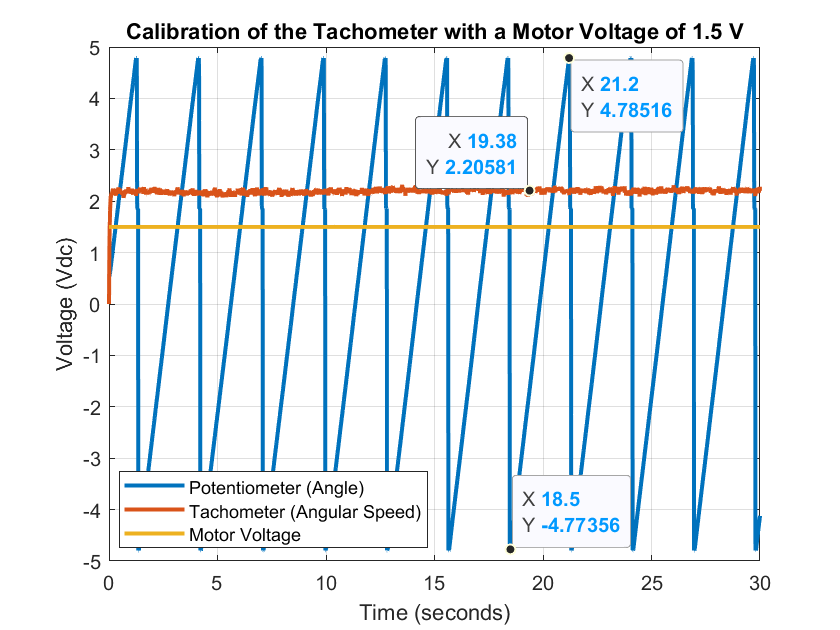
*Figure 3: Open-loop controller of the system used to calibrate the tachometer. The motor voltage was set by a step input for different voltages including: 0.5, 1.0, 1.5, and 2.0 volts. This was done to analyze the slope of the potentiometer voltage and the tachometer voltage.*



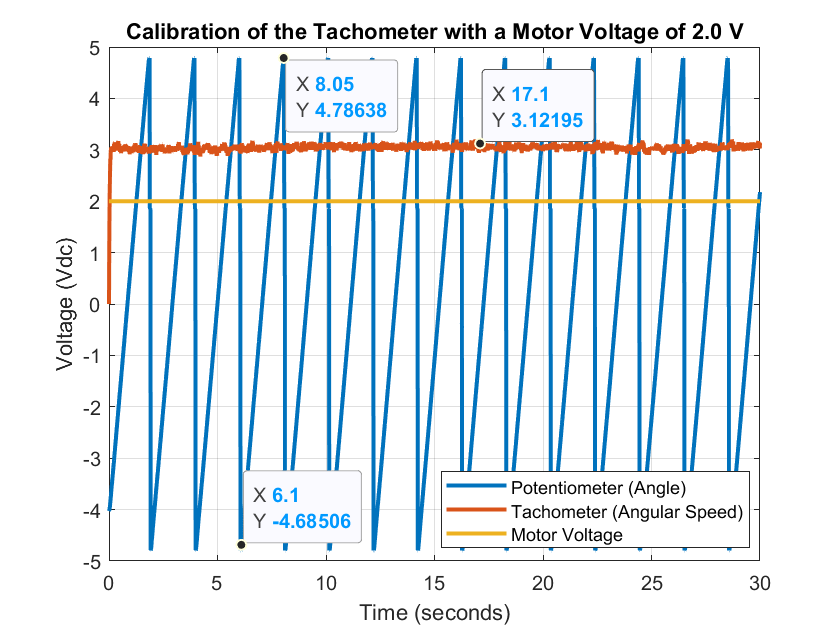
*Figure 4: Graph showing the potentiometer voltage and the tachometer voltage when the motor voltage is 0.5 volts. The slope of the potentiometer voltage is calculated by dividing the rise over the run. This is compared to the measured tachometer voltage in table 1.*



*Figure 5: Graph showing the potentiometer voltage and the tachometer voltage when the motor voltage is 1.0 volts. The slope of the potentiometer voltage is calculated by dividing the rise over the run. This is compared to the measured tachometer voltage in table 1.*



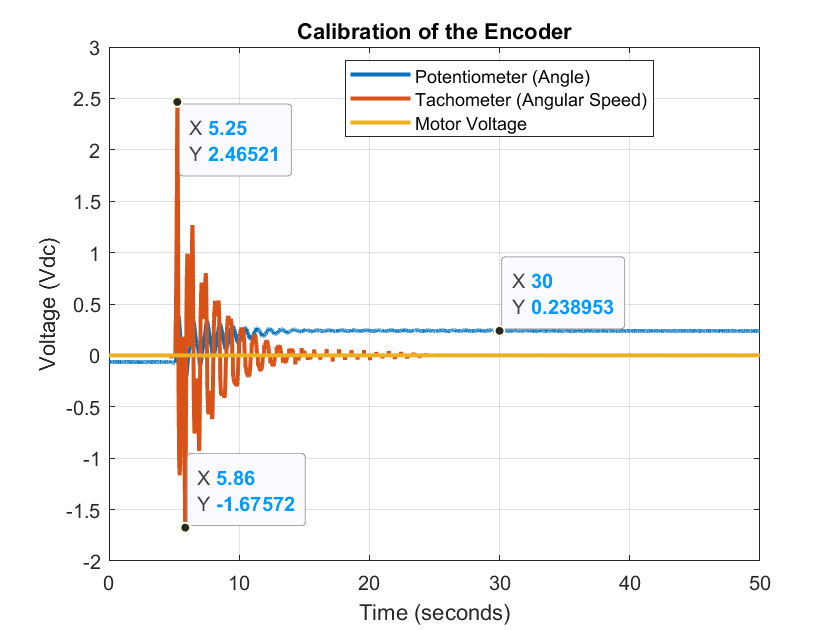
*Figure 6: Graph showing the potentiometer voltage and the tachometer voltage when the motor voltage is 1.5 volts. The slope of the potentiometer voltage is calculated by dividing the rise over the run. This is compared to the measured tachometer voltage in table 1.*



*Figure 7: Graph showing the potentiometer voltage and the tachometer voltage when the motor voltage is 2.0 volts. The slope of the potentiometer voltage is calculated by dividing the rise over the run. This is compared to the measured tachometer voltage in table 1.*

Calibrating the Encoder:

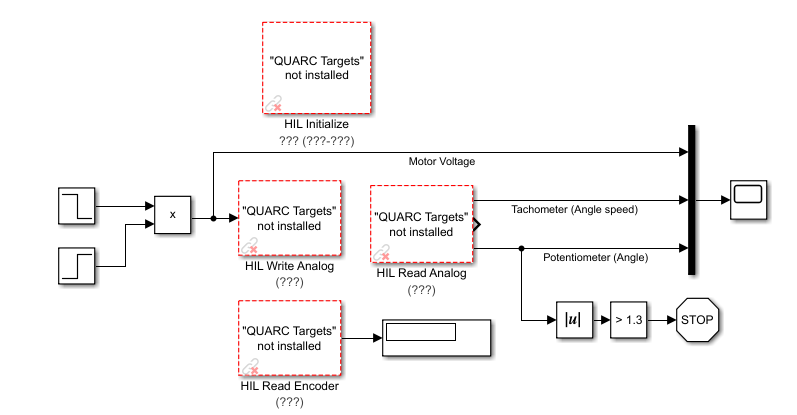
We now calibrated the encoder by setting the motor voltage to zero and holding the pendulum at its unstable equilibrium position. The unstable equilibrium position is when the pendulum is pointing upwards in an upright position. The stable equilibrium position is when the pendulum is pointing downwards to the floor. We started the system by holding the pendulum in an upright position and then let the pendulum go so it could point downwards. This was done to measure the encoder signals at its stable and unstable equilibrium positions. The encoder at 0 radians read -5 and at π radians read 2050. The gain needed to convert the encoder signal to radians involves . Figure 8 shows the potentiometer, tachometer, and motor voltages during this experiment.



*Figure 8: This graph displays the calibration of the encoder. The pendulum was held upright and let go at approximately 5 seconds. This was done to measure the encoder signals at the pendulum’s unstable and stable equilibrium positions. The encoder read -5 at 0 radians and 2050 at π radians. To convert the encoder signal to radians we use the following gain: .*

Safety:

The final preparation of our main experiment was the implementation of certain safety stops in the *Simulink* system model. The first safety stop involved forcing the motor voltage to be zero one second before the computer turns off the controller. This involved placing a step input which would have an initial value equal to the final value of our motor voltage. The final value of this step input would then be zero. The second safety stop involved having the system completely stop if the absolute value of the angle was greater than 45 degrees. Both of the safety stops can be seen in figure 9. Finally, we made sure to place the apparatus far away from anything that could obstruct the pendulum’s path.

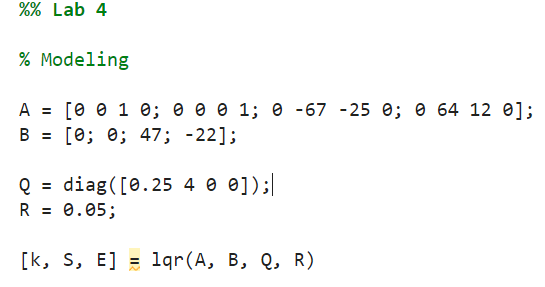


*Figure 9: Simulink model with two safety stops implemented. The first safety stop involved a step input which drives the motor voltage to zero a couple of seconds before the computer turns off the controller. The second safety stop involved having the system completely stop if the absolute value of the angle was greater than 45 degrees.*

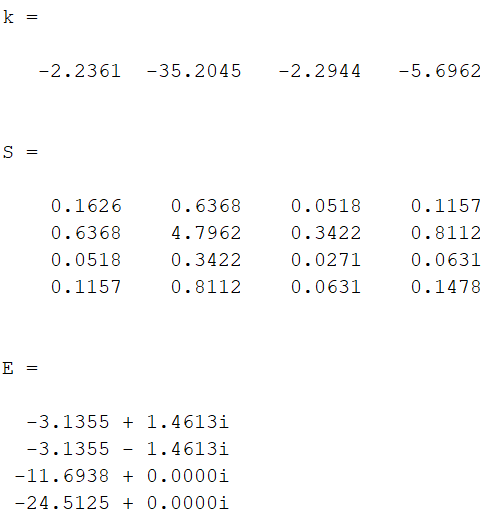
Modeling and Linear Controller Design:

We now advance to the design of the first controller for this experiment. This design involves a closed-loop system whose goal is to maintain the pendulum upright when the pendulum is initialized in an upright position. We derive the design of this controller through *MATLAB* and the code is shown in figures 10 and 11. Matrices A and B were obtained from the lab manual and the belong to the state space equation for the linearized system of the form . We now define a diagonal matrix Q and a constant R equal to 0.05, both of which are suggested by Quanser to find the necessary feedback gains of the closed-loop system. The result for the feedback gain vector is displayed in figure 11 where the vector k represents gains k1, k2, k3, and k4. These gains were calculated using *MATLAB’s* linear-quadratic regulator design which used matrices A, B, and C as inputs in addition to the constant R. Finally, the lab manual helps us to design the closed-loop system by telling us that the input to the motor is given by: . The gains k1, k2, k3, and k4 were all previously calculated through *MATLAB*. In addition, represent the cart angle and its derivative, respectively. Finally, represent the pendulum angle and its derivative, respectively. In conclusion, now that we calculated all of the feedback gains and knew their respective inputs this meant we could implement our closed-loop system into *Simulink*.

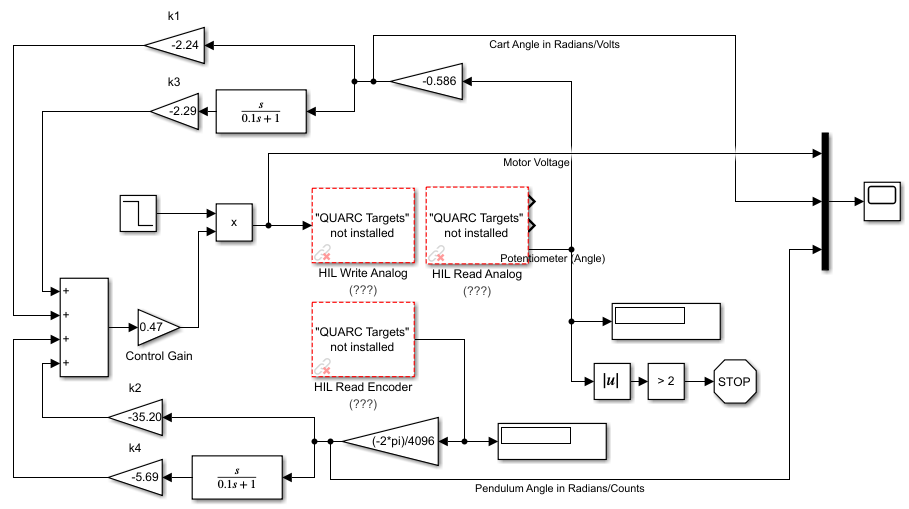
The actual implementation of the controller is shown in figure 12. Thi implementation was achieved with two key pieces of information. The first key piece of information included the feedback gains calculated in *MATLAB*. The second key piece of information included the motor voltage equation . Since we knew the values of k1, k2, k3, and k4 all we simply had to connect the necessary inputs to the gains. This was possible since we knew exactly what x1, x2, x3, and x4 represented. The derivatives were calculated by using the following transfer function . Finally, the pendulum was held in an upright position and was let go after starting the simulation. The controller successfully maintained the pendulum in its initial upright position for the duration of the simulation. Figure 13 shows a plot of the pendulum being maintained upright successfully. The plot displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage and the cart angle oscillate since the cart was moving from side to side to balance the pendulum.



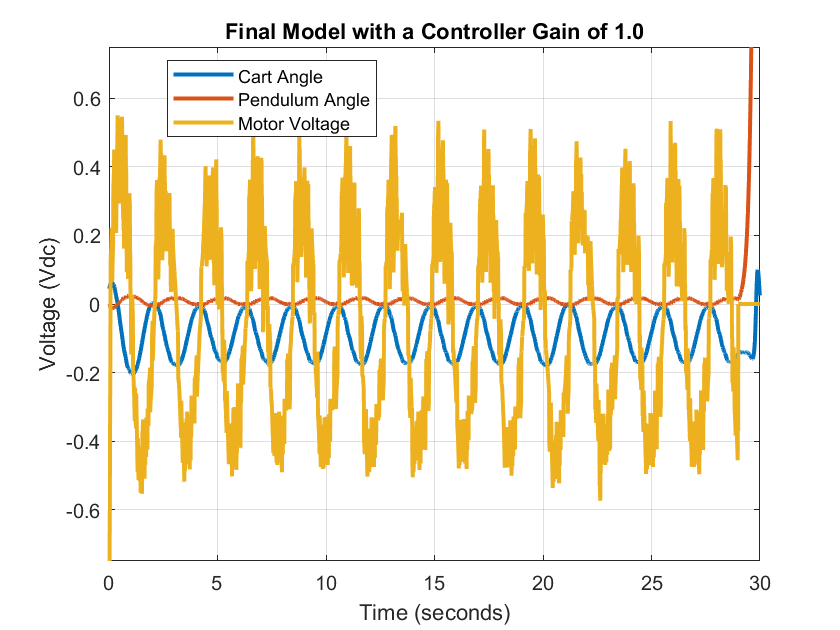
*Figure 10: MATLAB code used to calculate the feedback gains of the controller. Matrices A and B were obtained from the lab manual while matrix Q and constant R were suggested by Quanser. Finally, MATLAB’s linear-quadratic regulator function was used to calculate the feedback gains.*



*Figure 11: Results from the code in figure 10. We see here that the feedback gains k1, k2, k3, and k4 are equal to -2.24, -35.20, -2.29, and -5.70 respectively.*

**

*Figure 12: Closed-loop controller which stabilizes an upright pendulum by using four different feedback gains, the pendulum angle and its derivative, and the cart angle and its derivative. The outputs of this system included the pendulum angle converted to radians/counts, the cart angle converted to radians/volts, and the motor voltage.*



*Figure 13: This plot shows the successful implementation of the closed-loop controller which allowed the pendulum to maintain its initial upright position. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage and the cart angle oscillate since the cart was moving from side to side to balance the pendulum.*

Testing:

Now that we have successfully implemented the inverted pendulum, starting from an upright position, we need to test its limits. The first test involves altering the gain of the controller to analyze the sensitivity of the closed-loop system to changes in the controller parameters. We test the system for different values of the controller gain including: 1.00, 0.90, 0.50, and 0.47. We compare the performance of the system in the case of the four different controller gains in table 2 below. In addition, the figures corresponding to the four controller gains are figures 13, 14, 15, and 16.

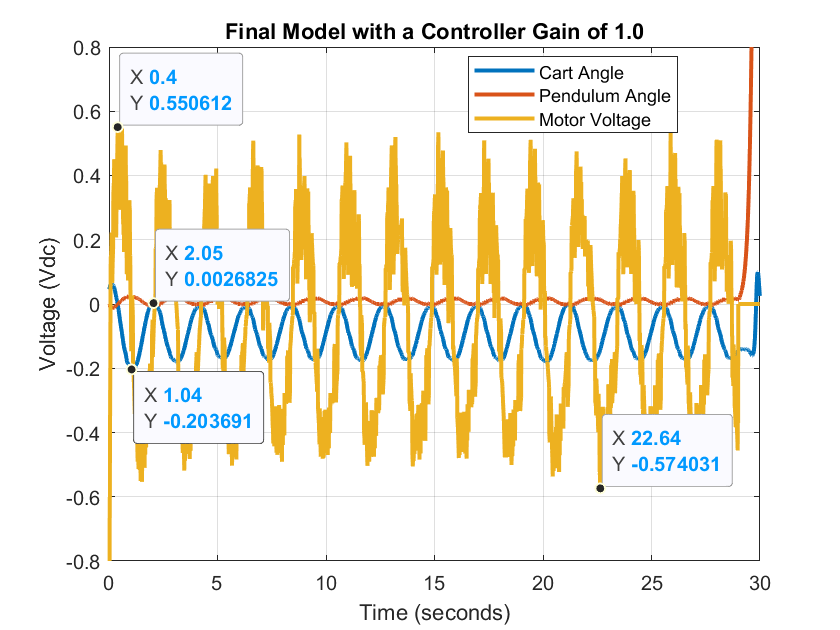
The second test involves adding a small step input to the motor after the closed-loop system has been running for a while. We test the system for different values of the step input including: 0.2, 0.3, 0.4, and 0.5. We compare the performance of the system in the case of the four different step inputs in table 3 below. In addition, the figures corresponding to the four step inputs are figure 17, 18, 19, and 20.

| **Controller Gain** | **Maximum Motor Voltage** | **Minimum Motor Voltage** | **Maximum Cart Angle** | **Minimum Cart Angle** | **Maintained Upright Pendulum?** |
| --- | --- | --- | --- | --- | --- |
| 1.00 | 0.551 | -0.574 | 0.003 | -0.204 | Yes |
| 0.90 | 0.745 | -0.558 | -0.208 | -0.453 | Yes |
| 0.50 | 2.228 | -2.316 | 0.937 | -0.925 | Yes |
| 0.47 | 2.407 | -2.163 | 0.814 | -1.080 | No |

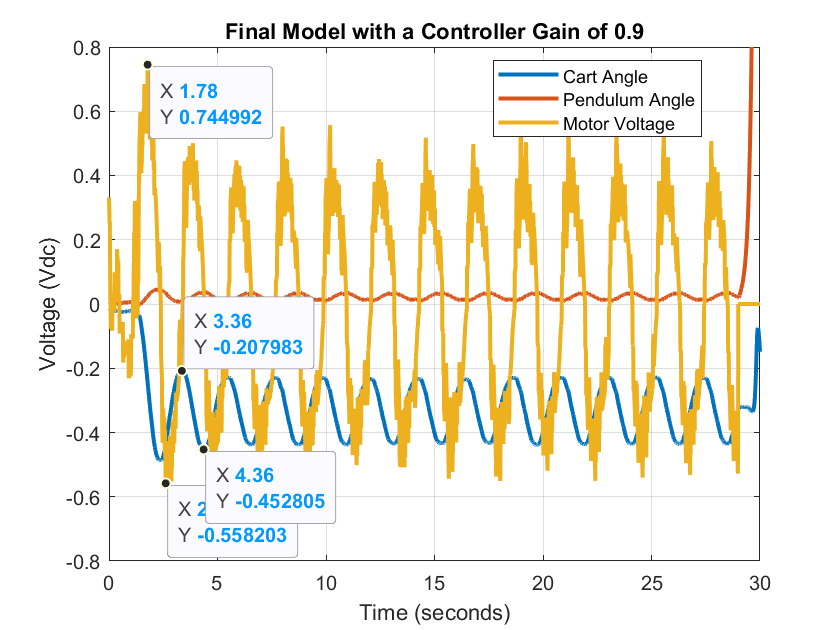
*Table 2: Table which compares the performance of the closed-loop controller with different controller gains. The following controller gains are compared: 1.00, 0.90, 0.50, and 0.47. The maximum and minimum of the motor voltage and cart angle are compared. In addition, we note if the controller achieved its goal of maintaining the pendulum in an upright position.*

| **Step Input** | **Maximum Motor Voltage** | **Minimum Motor Voltage** | **Cart Angle Midline Before Step** | **Cart Angle Midline After Step** | **Maintained Upright Pendulum?** |
| --- | --- | --- | --- | --- | --- |
| 0.00 | 0.551 | -0.574 | -0.091 | No Step | Yes |
| 0.20 | 0.566 | -0.667 | -0.112 | -0.049 | Yes |
| 0.30 | 0.494 | -0.629 | -0.012 | 0.087 | Yes |
| 0.40 | 0.435 | -0.732 | 0.113 | 0.232 | Yes |
| 0.50 | 0.649 | -0.660 | 0.007 | 0.156 | Yes |

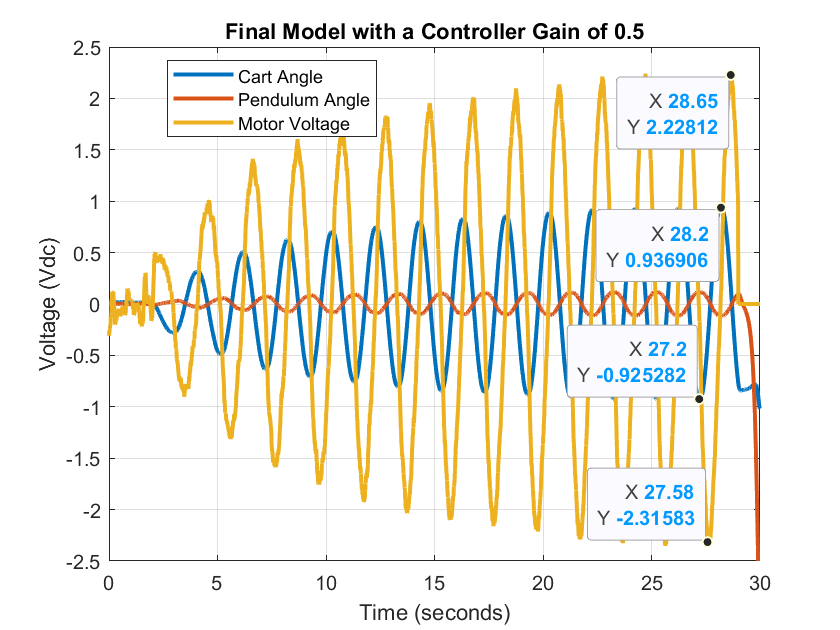
*Table 3: Table which compares the performance of the closed-loop controller with different step inputs. The following step inputs are compared: 0.00, 0.20, 0.30, 0.40, and 0.50. The maximum and minimum motor voltages are compared. The cart angle midline before and after the step are also compared. In addition, we note if the controller achieved its goal of maintaining the pendulum in an upright position.*

**

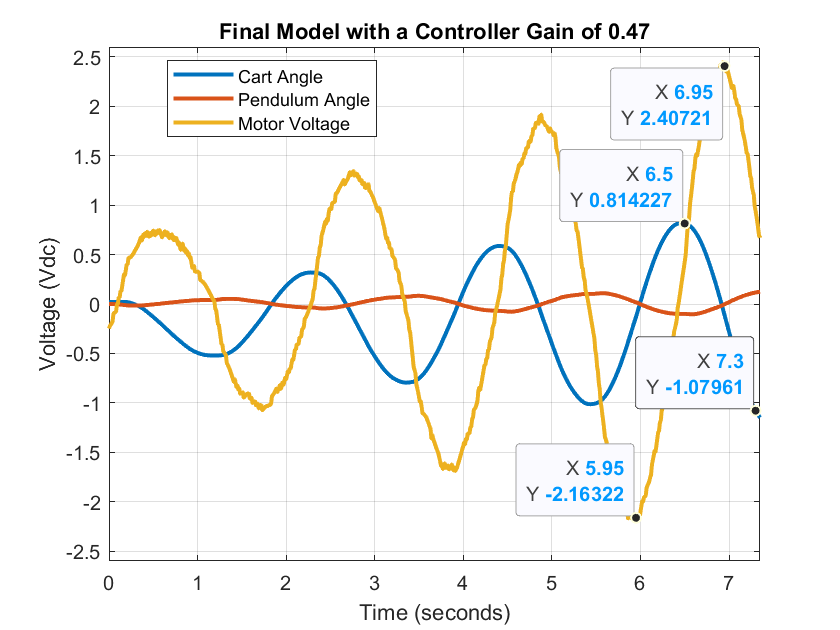
*Figure 14: This plot shows the implementation of the closed-loop controller with a normal controller gain of 1.00. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage spikes initially before stabilizing for the rest of the simulation. The cart angle’s midline is consistent throughout the entire simulation.*



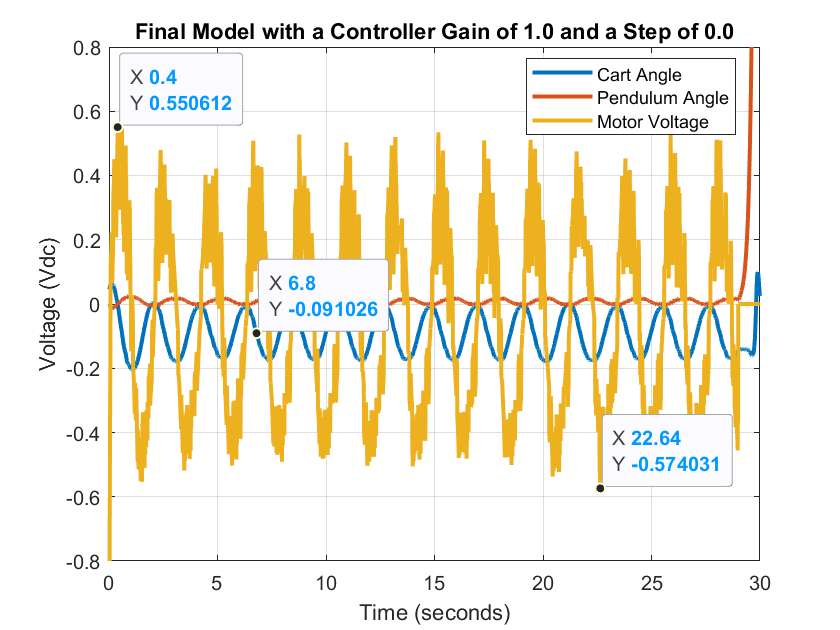
*Figure 15: This plot shows the implementation of the closed-loop controller with an altered controller gain of 0.90. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage spikes initially before stabilizing for the rest of the simulation. The cart angle’s midline is noticeably different from when the controller gain was 1.00. The midline is now around -0.33.*



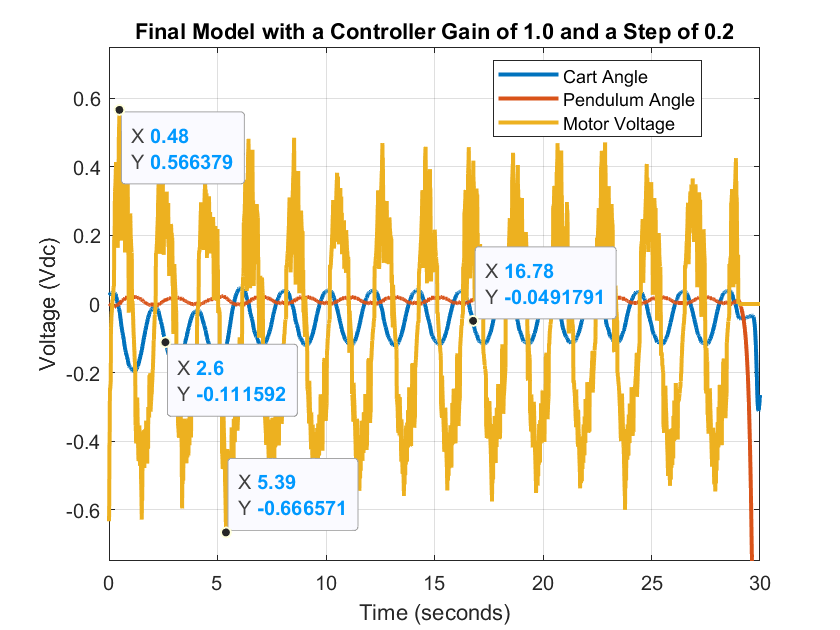
*Figure 16: This plot shows the implementation of the closed-loop controller with an altered controller gain of 0.50. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage spikes initially and becomes increasingly unstable for the rest of the simulation. The cart angle also follows a similar trend where its oscillations become larger and larger. Although the controller succeeded in maintaining the pendulum upright it almost failed due to the huge fluctuations in the motor voltage and cart angle.*



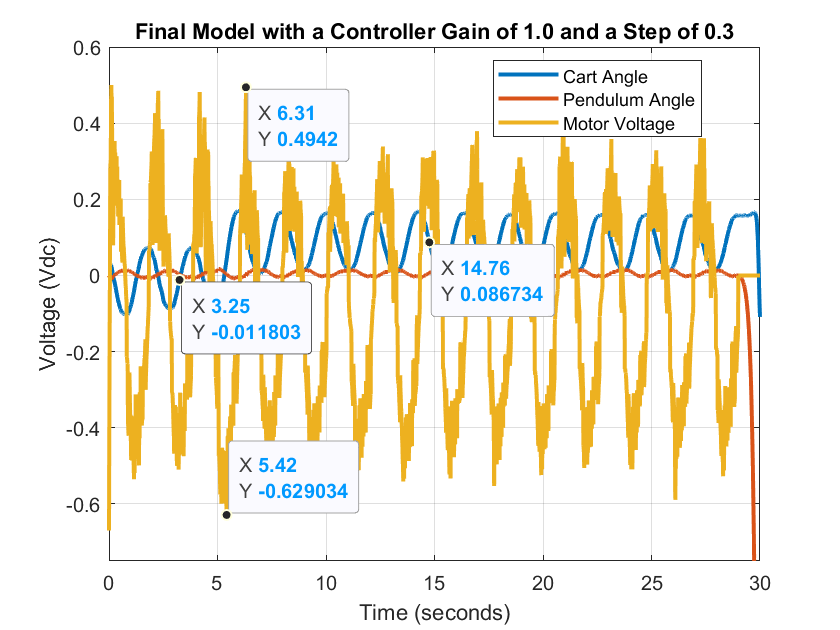
*Figure 17: This plot shows the implementation of the closed-loop controller with an altered controller gain of 0.47. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the motor voltage starts becoming increasingly unstable as soon as the simulation starts. The cart angle also follows a similar trend where its oscillations become larger and larger. The controller is cut short at around 7.3 seconds since it failed to maintain the pendulum upright. This means that a controller gain of 0.47 was the limit for a successful implementation of the controller.*

**

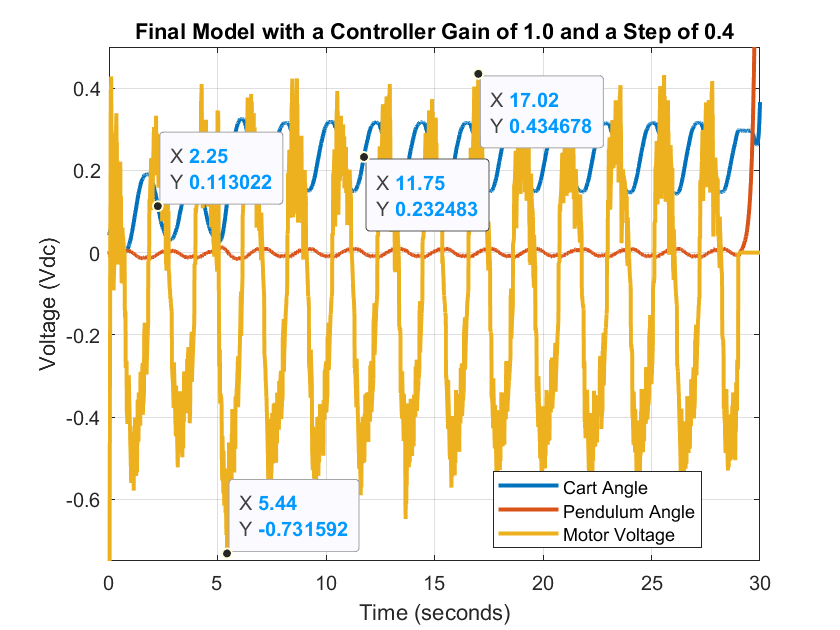
*Figure 18: This plot shows the implementation of the closed-loop controller with an additional step input of 0.20. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the cart angle’s midline is noticeably different before the step input is added. The midline before the step is -0.112, but after the step it is -0.049.*



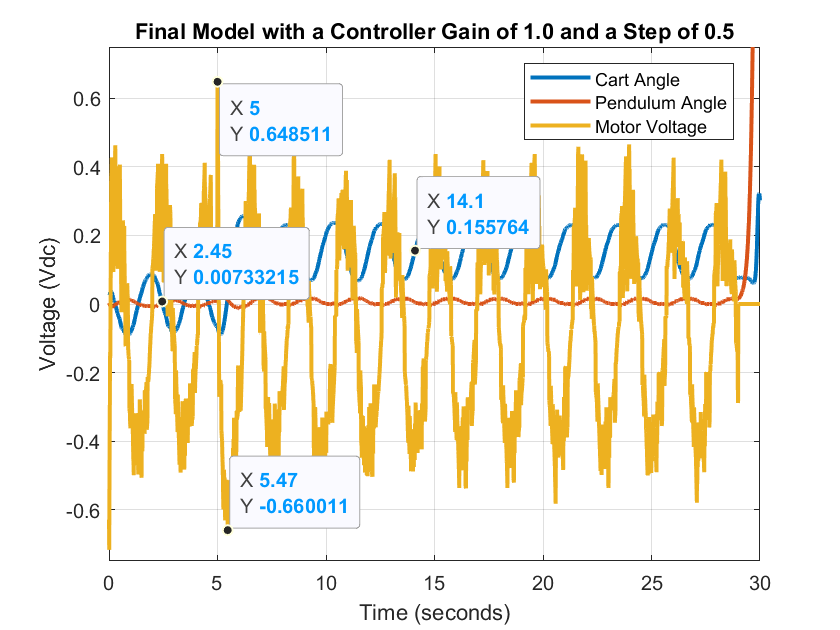
*Figure 19: This plot shows the implementation of the closed-loop controller with an additional step input of 0.20. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the cart angle’s midline is noticeably different before the step input is added. The midline before the step is -0.112, but after the step it is -0.049.*



*Figure 20: This plot shows the implementation of the closed-loop controller with an additional step input of 0.30. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the cart angle’s midline is noticeably different before the step input is added. The midline before the step is -0.012, but after the step it is 0.087.*



*Figure 21: This plot shows the implementation of the closed-loop controller with an additional step input of 0.40. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the cart angle’s midline is noticeably different before the step input is added. The midline before the step is 0.113, but after the step it is 0.232.*



*Figure 22: This plot shows the implementation of the closed-loop controller with an additional step input of 0.50. It displays the pendulum angle, the cart angle, and the motor voltage. We see that the cart angle’s midline is noticeably different before the step input is added. The midline before the step is 0.007, but after the step it is 0.156.*

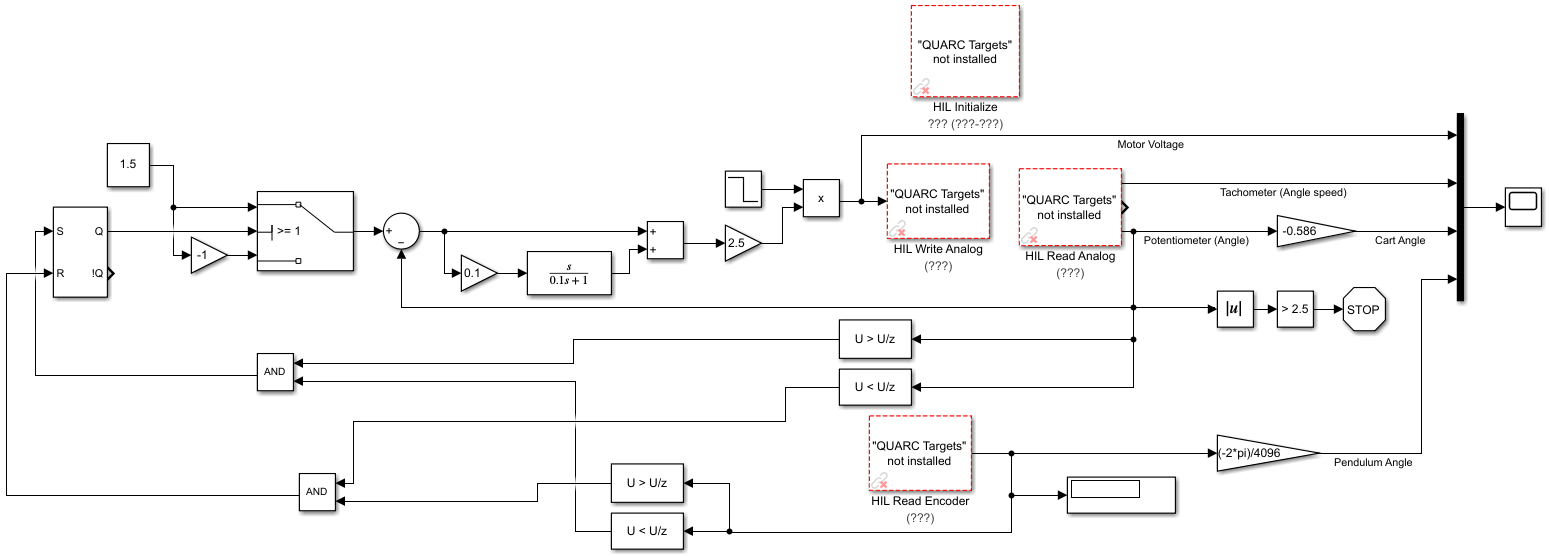
Swing-Up Controller:

We now enter the second phase of this laboratory which involves achieving the same goal as before, but with the pendulum starting from a downwards position. This means that we first have to force the pendulum to swing upright and from there balance the pendulum so it does not fail. The first step to accomplish this involved creating a swing-up controller which moved the pendulum from a downward position to an upright one. Figure 23 displays the complete *Simulink* model which was able to accomplish this task. The controller works in such a way that the cart moves 60 degrees to the right which makes the pendulum swing. As soon as it is detected that the pendulum stops swinging the cart moves 60 degrees to the left. This makes the pendulum swing even farther. This process continues until the pendulum swings overhead and falls downwards. There is no mechanism in place which balances the pendulum, but this will be implemented soon enough.

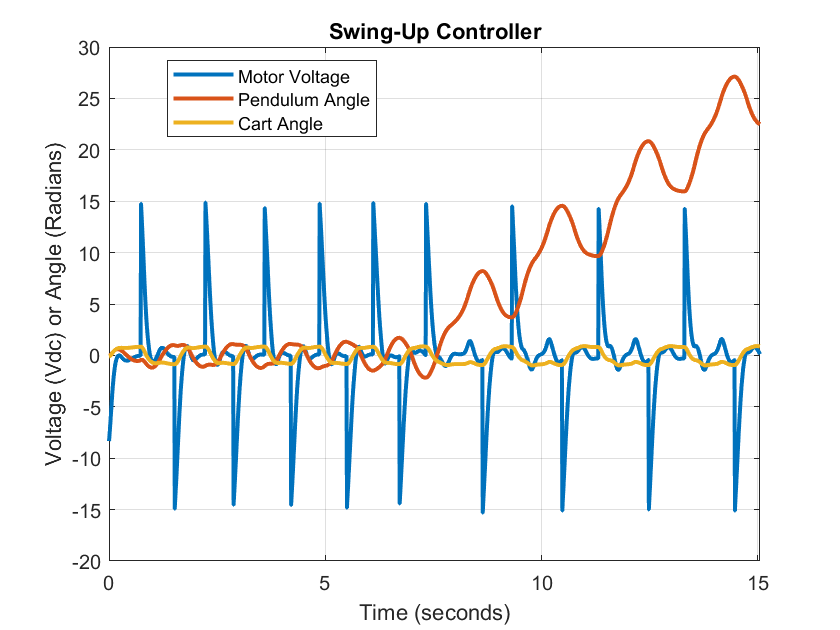
The *Simulink* model was created through a combination of logic gates, flip flops, switches, and blocks that detect increases and decreases. The model starts with two blocks which detect increases and two blocks which detect decreases. The two inputs used in these blocks are the potentiometer voltage and the HIL encoder values. The output of these detection blocks are boolean values which are inputted into two AND gates. The first AND gate boolean inputs include: is the potentiometer voltage increasing and are the HIL encoder values decreasing. The second AND gate boolean inputs include: is the potentiometer voltage decreasing and are the HIL encoder values increasing. The two outputs are used as inputs for an SR flip-flop. This SR flip-flop controls whether the cart moves 60 degrees to the right or to the left. The output of the SR flip-flop is used as an input for a closed-loop proportional-derivative controller. Figure 24 displays the successful implementation of this controller since the pendulum angle continually increases which represents the pendulum continuously rotating. This controller was successful in swinging the pendulum to an upright position, but it does not balance the pendulum. This will be addressed in the next step for this controller.

The next obstacle includes zeroing the pendulum angle when it reaches an upright position. Figure 24 shows that as the pendulum reaches the upright position the angle reaches π which does not work with the balancing controller previously made. The controller needs the pendulum to be zero when it is upright. We can adjust the pendulum to be zero by using the modulo operation. The pendulum angle is divided by 2π and the remainder of this division is subtracted π. This means that when the pendulum is upright the pendulum angle will be zero. In addition, a new switch is added which allows the controller to switch from the swing-up phase to the balancing phase.

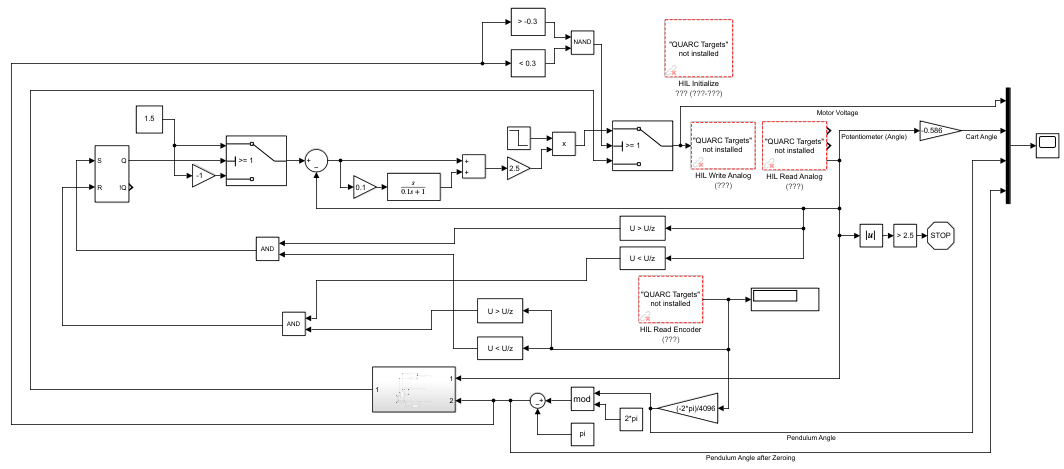
The final swing-up controller which swings the pendulum from a downwards position to an upright one and balances that position is shown in figure 25. The sub-controller in figure 26 has the following inputs: the potentiometer voltage and the pendulum angle after adjustment. The pendulum angle is manipulated in such a way that it reads 0 when the pendulum is upright. This contrasts with the pendulum angle seen in figure 24. The zeroed out pendulum angle is obtained by converting the encoder values to an angle by multiplying by . After this is done we use the modulo operation to obtain the remainder of the angle after being divided by . Next we subtract from this result to obtain a new pendulum angle which is never more than and its new zero is the upright position. We then use a NAND gate and the new zeroed out pendulum angle to implement the mode-switching of the system. This determines whether the system is trying to swing the pendulum upwards or if it is trying to balance it.



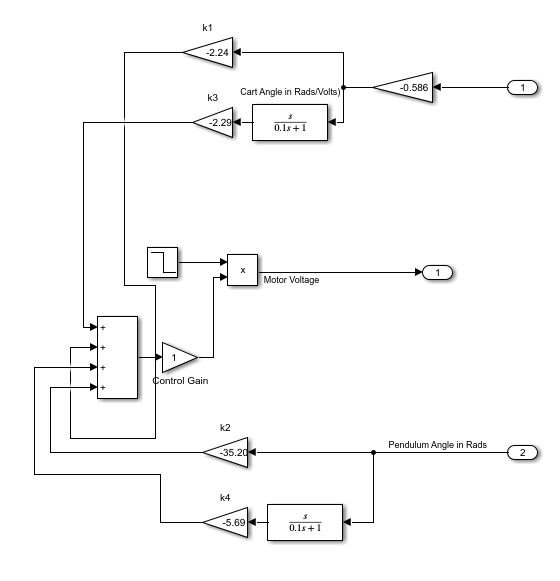
*Figure 23: Swing-up controller created through a combination of logic gates, flip flops, switches, and blocks that detect increases and decreases. The controller starts with two blocks which detect increases and two blocks which detect decreases. The two inputs used in these blocks are the potentiometer voltage and the HIL encoder values. The output of these detection blocks are boolean values which are inputted into two AND gates. The first AND gate boolean inputs include: is the potentiometer voltage increasing and are the HIL encoder values decreasing. The second AND gate boolean inputs include: is the potentiometer voltage decreasing and are the HIL encoder values increasing. The two outputs are used as inputs for an SR flip-flop. This SR flip-flop controls whether the cart moves 60 degrees to the right or to the left. The output of the SR flip-flop is used as an input for a closed-loop proportional-derivative controller.*

**

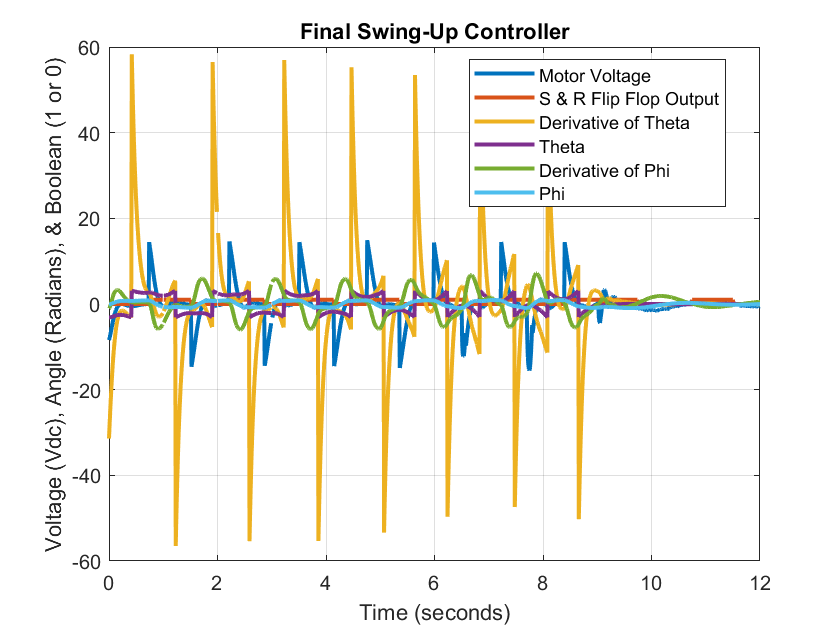
*Figure 24: This plot shows the successful implementation of the swing-up controller since the pendulum angle continually increases which represents the pendulum continuously rotating.*

**

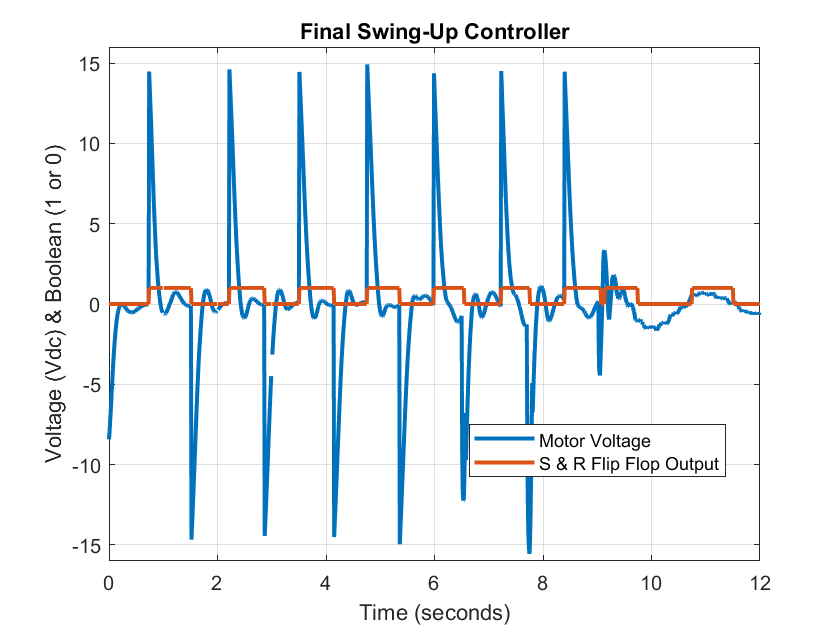
*Figure 25: The final swing-up controller which swings the pendulum from a downwards position to an upright one and balances that position using the controller shown in figure 26. This sub-controller has the following inputs: the potentiometer voltage and the pendulum angle after adjustment. The pendulum angle is manipulated in such a way that it reads 0 when the pendulum is upright. This contrasts with the pendulum angle seen in figure 24. The zeroed out pendulum angle is obtained by converting the encoder values to an angle by multiplying by . After this is done we use the modulo operation to obtain the remainder of the angle divided by . Next we subtract from this result to obtain a new pendulum angle which is never more than and its new zero is the upright position. We then use a NAND gate and the new zeroed out pendulum angle to implement the mode-switching of the system. This determines whether the system is trying to swing the pendulum upwards or if it is trying to balance it. This is executed by checking if the zeroed pendulum angle is less than or greater than 0.3 radians. These two inputs are used in a NAND gate to switch from the swing-up portion to the stabilization portion of the controller.*

**

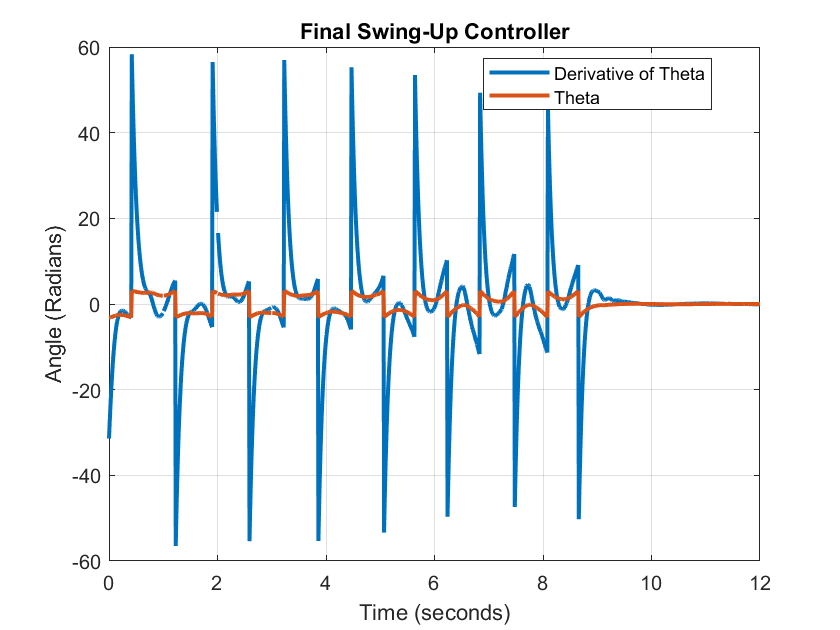
*Figure 26: Modified controller from figure 12 which is used in the final swing-up controller displayed in figure 25. This controller stabilizes the pendulum when it is in an upright position. The controller uses two inputs and generates one output. Input 1 is the car angle and input 2 is the pendulum angle. Output 1 is the motor voltage which changes to stabilize the pendulum in an upright position.*

**

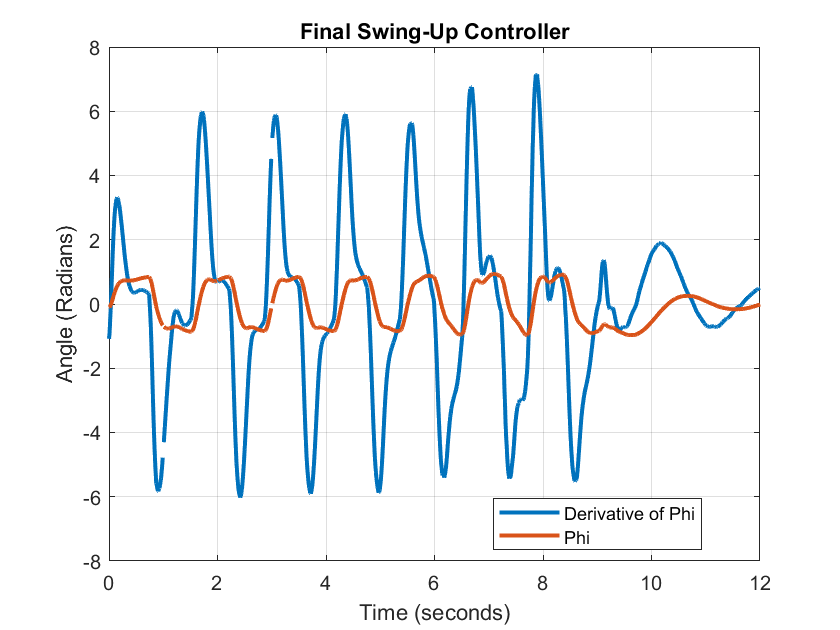
*Figure 27: Response of the final swing-up controller displayed in figure 25. The S and R Flip-Flop output is in terms of a Boolean (1 and 0).Theta and its derivative are in terms of Rads/Counts. Phi and its derivative are in terms of Rads/Volts.*

**

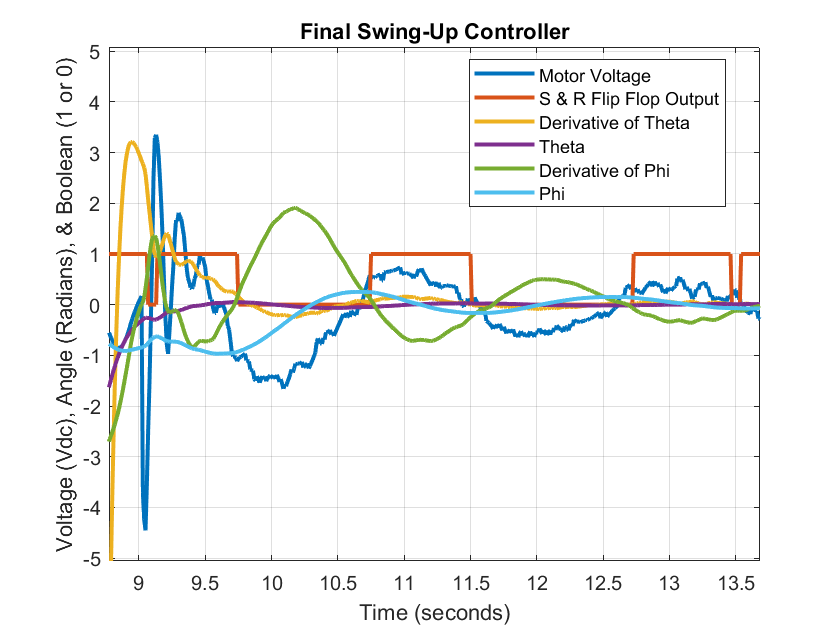
*Figure 28: This figure displays a filtered response from figure 27 which only displays the S & R flip flop output along with the motor voltage. The motor voltage is measured in volts and the S & R flip flop output is in terms of a boolean. When the S and R output, Q, is 1 the motor spins counter clockwise and when it is 0 the motor spins clockwise.*

**

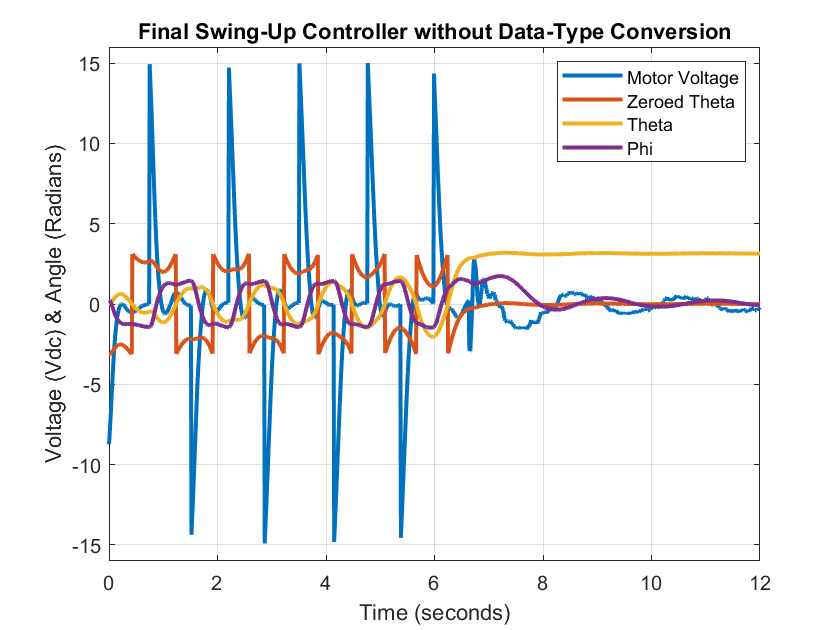
*Figure 29: This figure displays a filtered response from figure 27 which only displays theta and its derivative. Theta and its derivative are both measured in radians. This displays theta (the pendulum angle) as the cart swings it back and forth to get it to an upright position. Once it is upright, the pendulum stabilizes. Note: the cart angle is not listed here.*

**

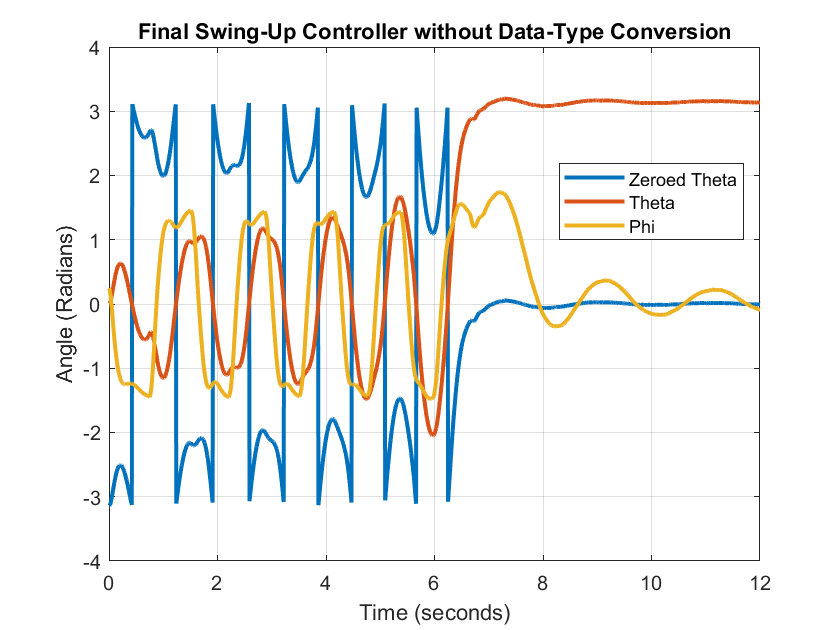
*Figure 30: This figure displays a filtered response from figure 27 which only displays phi and its derivative. Phi and its derivative are both measured in radians. This displays phi (the cart angle) as the cart moves clockwise and counterclockwise to get the pendulum to an upright position. Note: the pendulum response is not listed here.*

**

*Figure 31: This figure displays a zoomed-in response from figure 27 when the pendulum stabilizes. The movement of the cart and the pendulum become much smaller when compared to the swing-up phase. This is because only very mild movements are necessary to keep the pendulum balanced.*

**

*Figure 32: Response of the swing-up controller from figure 25 before a Simulink data type conversion. The motor voltage is measured in volts, the theta outputs are in terms of radians/counts, and phi is in terms of radians/volts.*

**

*Figure 33: This figure is a zoomed-in version of figure 32. It displays the transition the system underwent from the swing-up phase to the stabilization phase. It displays theta (the pendulum angle) before and after zeroing and phi (the cart angle). Note: The response of the motor voltage is not listed in this graph.*

**Conclusion**

This experiment consisted of making a controller that would use a servo cart to swing an attached pendulum upwards and keep it in its upright position. For the first part, we had to make a controller that would only stabilize the pendulum in the upright position. The basic mechanics of this controller are to move the cart counterclockwise and clockwise repeatedly to counteract the gravity acting on the pendulum. The cart would change angles very rapidly, but also in very small increments to keep the pendulum stabilized. The controller we used to achieve this stabilization is a closed-loop controller shown in figure 12. The controller takes in inputs of the pendulum angle and the cart angle. Then, the controller changes the angle of the two to keep the position of the pendulum upright.

The second part of the experiment involves swinging the pendulum to its upward position by swinging it using the servo cart. Before, we would place the pendulum upwards initially using our hands, now the cart does it for us without any external contact. We built this controller by also implementing the controller from the first part. Some differences in the second controller consisted of using a switch, an SR flip-flop and logic gates. The logic gates and SR flip-flops were used to analyze the position of the pendulum as the cart was going back and forth to create a swinging motion. When the controller read that the pendulum was pointing upwards, it then activated the first controller that we made and kept it in that position in a stable way.