Inertia Wheel Pendulum

MECA 482-02 Group 5: *Victor Alvarez, Xavier Andrade, Sean Murray, Samuel Kelly, John Voter*

Introduction

1. Background

The inertia wheel pendulum is a system that seeks to stabilize a pendulum in the inverted upright position. This is to be acheived by calculated accelleration of an inertia wheel at the end of the pendulum arm. In order to achieve the desired functionality, one end of the pendulum arm is attached to a rotating base that is fixed to a table-end. Incorporated into the system are two encoders, one serves as the rotating base in order to track the pendulum arm's angle of rotation while the other monitors wheel speed/position. Figure 1 below captures the operational viewpoint for the system described above.

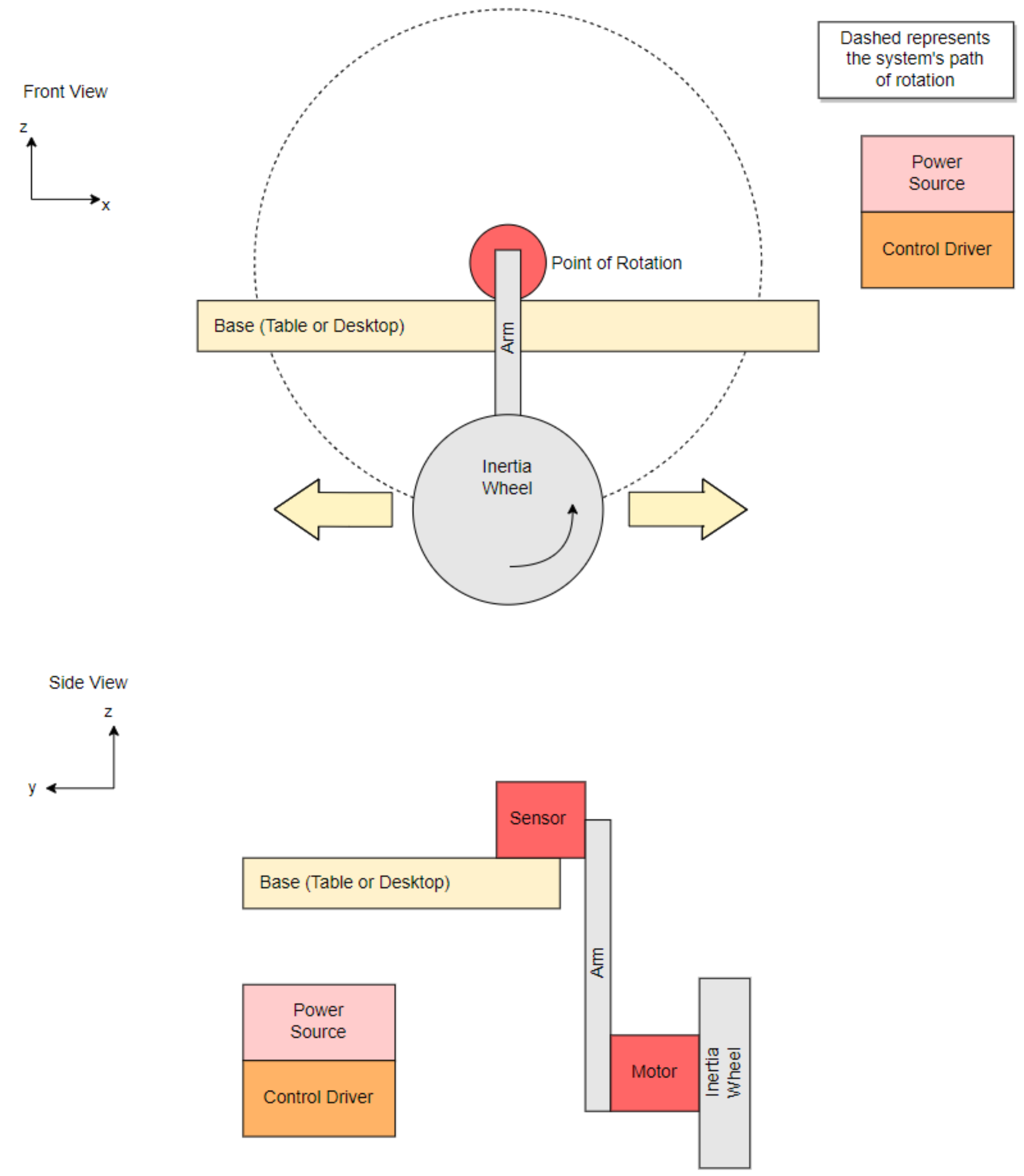

Figure 1: Operational viewpoint for the inertia wheel pendulum.

The given operational viepoint serves to give the reader a very top-level vizualization of the physical system. However, it is unclear as to how this system will acheive the desired functionality. In response, a logical/functional viewpoint was created to serve as a reference for how components within the system interact with onanother. Seen below in Figure 2 is the logical/functional viepoint for the inertia wheel pendulum.

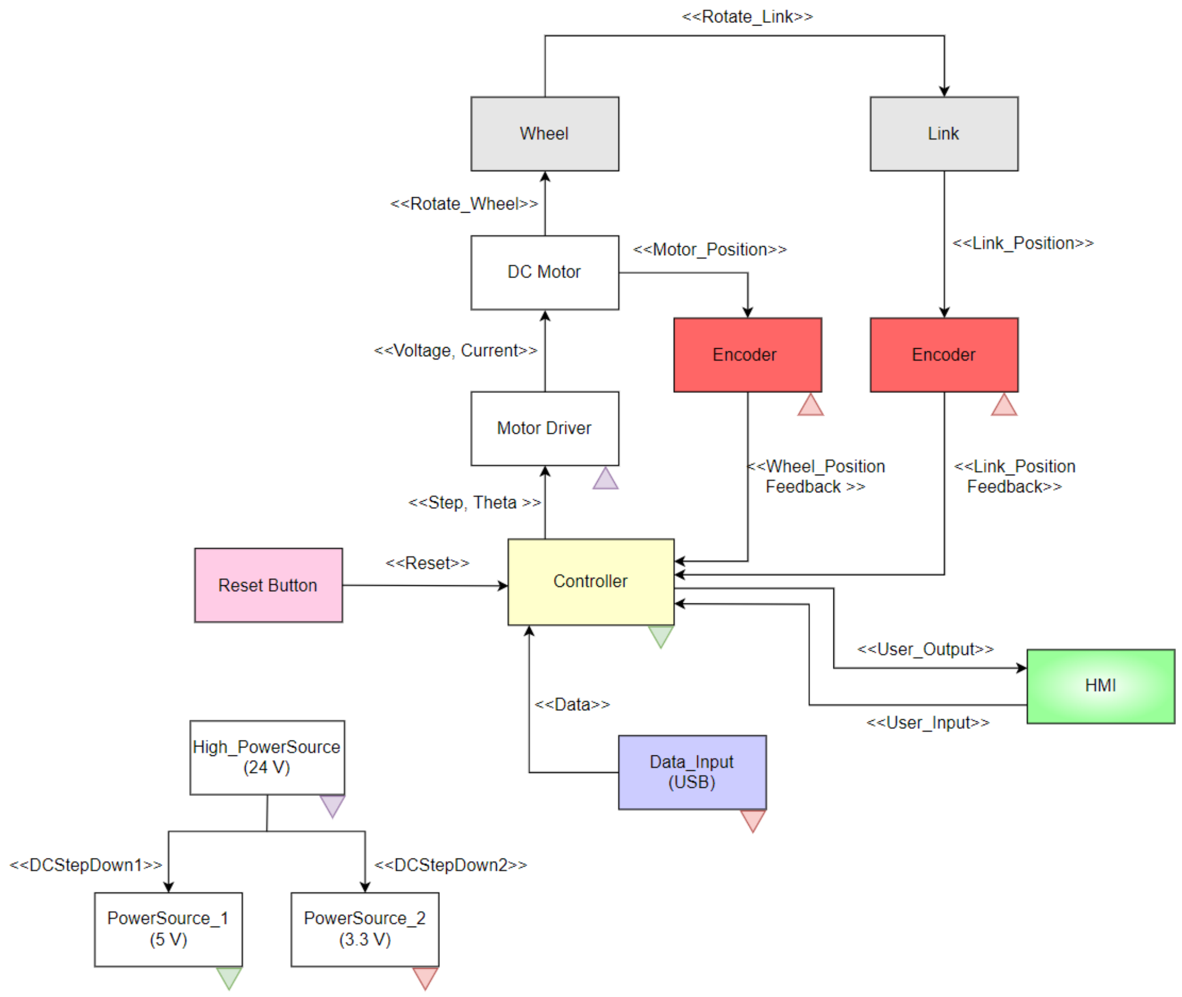

Figure 2: Logical/functional viewpoint for the inertia wheel pendulum.

When couple with Figure 1, the logical/functional viewpoint serves to provide a complete top-level unserstanding of system functionality.

The following report consists of documentation pertaining to the solution obtained by the team. In order to acheive the desired functionality a mathematic model wast obtained for the system, from there a controller architechture was devised to simulate and implement the derived mathematical model. The following report consists of documentation of the control system design process.

2. Resources

Listed below are several key resources utilized by the team throughout the control system desing process (for full citation please see "References"):

- *Control System Engineering*: 7th Edition; Norman S. Nice
- *Automatic Control with Experiments*: Victor Manuel Hernández-Guzmán & Ramón Silva-Ortigoza

- *Nonlinear analysis and control of a reaction wheel pendulum - Lyapunov-based approach*: Oscar Montoya & Walter Gonzales
- *Object Heirarchy Relations - CopelliaSim*: Leopoldo Arnesto
- CopelliaSim Forum: *Shared Memory Plugin For VREP /Matlab Communication*

Modeling

1. Schematic

In order to derive the mathematical model for the inertia wheel pendulum, a schematic must first be constructed to define reference axes and system variables. Figure 3 below depicts the system schematic used to define relevant variables.

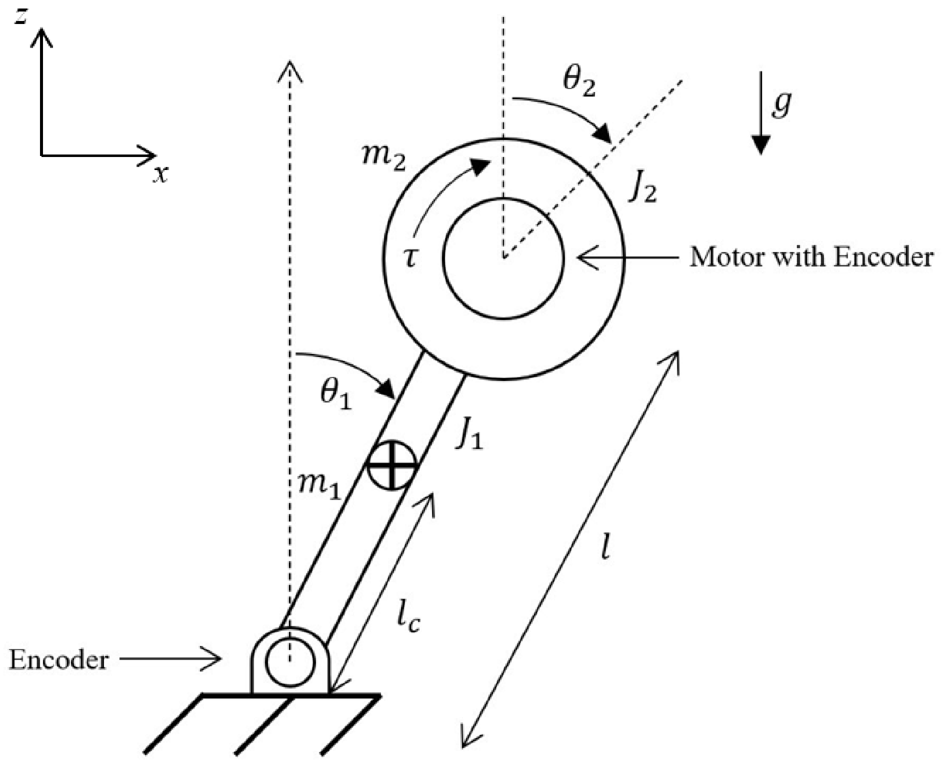

Figure 3: Symbolic schematic of the inertia wheel pendulum.

The following table depicts definitions based on the above schematic for system parameters used in the derivation of the mathematical model.

Table 1: Parameter definitions relevant to preparing the mathematical model for the inertia wheel pendulum.

Parameter	Value	Unit	Definition
g	9.81	m/s^2	Acceleration due to gravity
m_1	2	kg	Mass of pendulum arm
m_2	1	kg	Mass of wheel
l	1	m	Length of pendulum arm
l_c	$0.25\times l$	m	Location of pendulum arm CoM
r	0.5	m	Radius of wheel
J_1	$\left(\frac{1}{3}\right) m_1 l^2$	$kg\times m^2$	Pendulum arm inertia (rotated about end)
J_2	$\left(\frac{1}{2}\right) m_2 r^2$	$kg\times m^2$	Wheel inertia (rotated about center)
θ_1	-	$rads$	Pendulum arm angular position
θ_2	-	$rads$	Wheel angular position
τ	-	Nm	Torque applied at wheel
$\bar{m}$	$m_1 l_c + m_2 l$	kgm	Utilized to simplify mathematical model

2. Equation of Motion

For mechanical systems, a classical method to derive the equations of motion is the Euler-Lagrange approach. As is well known, this approach requires the computation of kinetic and potential energies and the subsequent knowledge of the forces acting on the system. The equations shown below are general definitions for a given system's Lagrangian, along with its kinetic and potential energies.

$$L = K - P \qquad \text{Eq. 1}$$

$$K = K_1 + K_2 \qquad \text{Eq. 2}$$

$$P = P_1 + P_2 \qquad \text{Eq. 3}$$

The first step in obtaining the system's equations of motion is to define the Lagrangian. As seen in the equations above, this is to be done by first quantifying the kinetic and potential energies of the system. The system's total kinetic energy is the summation of the kinetic energy of the pendulum arm rotating on its end, treated as a particle of mass m_1 located at l_c, along with the kinetic energy of the wheel rotating around its center, of mass m_1 translating a radius l. It should be noted that the wheel's angular velocity, as measured by an observer fixed to the mounting table for the system, is θ_1+θ_2. The equations below define the system's total kinetic energy.

$$K_1 = \frac{1}{2} m_1 l_c^2 \dot{\theta}_1^2 + \frac{1}{2} \dot{\theta}_1^2 J_1 \qquad \text{Eq. 4}$$

$$K_2 = \frac{1}{2} m_2 l^2 \dot{\theta}_1^2 + \frac{1}{2} (\dot{\theta}_1 + \dot{\theta}_2)^2 J_2 \qquad \text{Eq. 5}$$

The final step in obtaining the system's Lagrangian equation is to define its potential energies. It has been assumed that when the angular position of the pendulum arm is 0 degrees (normal equlibrium state) the system's potential energy is 0 as well. The equations below define the potential energies within the system.

$$P_1 = g m_1 l_c (1 + \cos\theta_1) \qquad \text{Eq. 6}$$

$$P_2 = g m_2 l (1 + \cos\theta_1) \qquad \text{Eq. 7}$$

Finally, combining Eqs.(4, 5, 6, 7) into Eq.(1) yields the final definition of the Lagrangian for the system shown in Eq.(8) below.

$$L = \frac{1}{2} m_1 l_c^2 \dot{\theta}_1^2 + \frac{1}{2} \dot{\theta}_1 J_1 + \frac{1}{2} m_2 l^2 \dot{\theta}_1^2 + \frac{1}{2} (\dot{\theta}_1 + \dot{\theta}_2)^2) J_2 - g\overline{m}(1 - \cos\theta_1) \qquad \text{Eq. 8}$$

In order to obtain the equation of motion for the system, the Lagrangian is implemented into the Euler-Lagrange equation for both interacting bodies which can be seen below.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1} = 0 \qquad \text{Eq. 9}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} = \tau \qquad \text{Eq. 10}$$

In Eq.(9), the zero on the right hand side indicates that no external torque is being applied to the pendulum at this point. Solving and simpflifying Eqs. (9, 10) yields the results below.

$$\ddot{\theta}_1 (m_1 l_c^2 + m_2 l^2 + J_1 J_2) + J_2 \ddot{\theta}_2 + g\overline{m}\theta_1 = 0 \qquad \text{Eq. 11}$$

$$J_2 \ddot{\theta}_1 + J_2 \ddot{\theta}_2 = \tau(t) \qquad \text{Eq. 12}$$

Utilizing the Euler-Lagrange approach, the above equations represent the fully-defined equations of motion for the inertia wheel pendulum.

3. State Space Representation

With the equations of motion derived for the system, the final step in acquiring the mathematical model is to represent the system in state space. Seen below are the general equations for state space representation. Note the equations have been slightly modified to fit the particular scenario of the inertia wheel pendulum.

$$\dot{\theta} = A\theta + Bu \qquad \text{Eq. 13}$$

$$y = C\theta \qquad \text{Eq. 14}$$

To represent the system in state space the phase variables must first be defined. Eqs.(15, 16) below show the system-particular state variable definitions.

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \qquad \text{Eq. 15}$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \qquad \text{Eq. 16}$$

In order to acquire the equations of motion in phase-variable form, Eqs. (11, 12) are utilized to solve for θ_1 double-dot and θ_2 double-dot. From here, the relationships below are applied.

$$\dot{\theta}_1 = \theta_2 \qquad \text{Eq. 17}$$

$$\dot{\theta}_2 = \ddot{\theta}_1 \qquad \text{Eq. 18}$$

$$\dot{\theta}_3 = \ddot{\theta}_2 \qquad \text{Eq. 19}$$

Applying these relationships yields the vector-matrix form for the state space equation. The results can be seen below.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\bar{m}g}{J_2(1 - m_1 l_c^2 + m_2 l^2 + J_1 + J_2)} & 0 & 0 \\ \frac{-\bar{m}g}{J_2(1 - m_1 l_c^2 + m_2 l^2 + J_1 + J_2)} & 0 & 0 \end{bmatrix} \qquad \text{Eq. 20}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{J_2} + \frac{m_1 l_c^2 + m_2 l^2 + J_1 + J_2}{J_2(1 - m_1 l_c^2 + m_2 l^2 + J_1 + J_2)} \\ \frac{-m_1 l_c^2 + m_2 l^2 + J_1 + J_2}{J_2(1 - m_1 l_c^2 + m_2 l^2 + J_1 + J_2)} \end{bmatrix} \qquad \text{Eq. 21}$$

The final step in aquiring the complete state equation is to define the output equation. Given the context of the inertia wheel pendulum, the output equation is represented below.

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad \text{Eq. 22}$$

A step-by-step documentation of the methodology surrounding the acquisition of the inertia wheel pendulum's state space equation can be found here.

Sensor Calibration

No sensor calibration was necessary for implementation.

Controller Design & Simulation

In order to implement the equation of motion into CopelliaSIM and acheive the desired function, an appropriate controller architechture must be devised. To balance the system in the inverted upright position, the motor must act according to feedback from the encoder that tracks the degree of rotation of the pendulum arm.

The first controller used for simulation was a full-state feedback controller seen below in Figure 4.

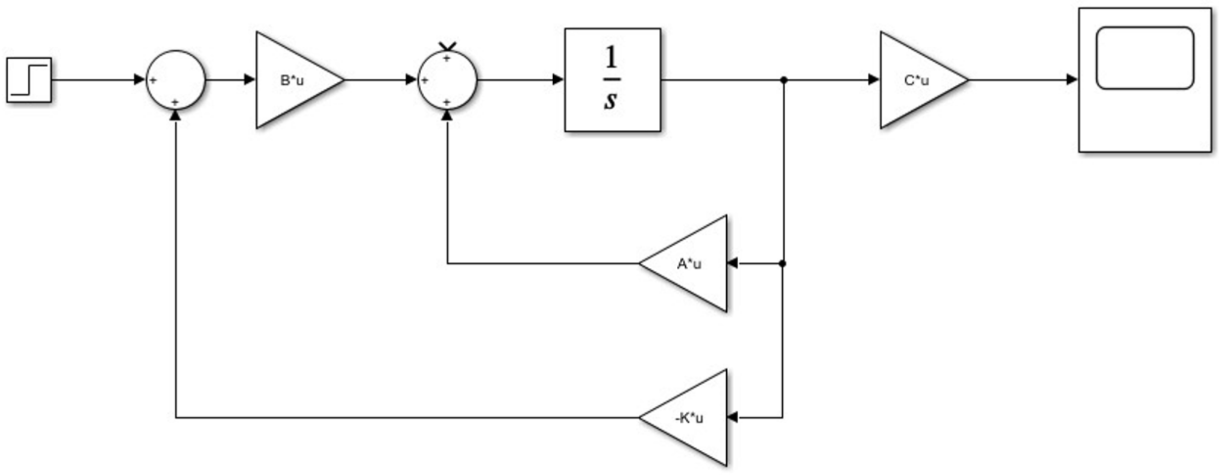

Figure 4: Feedback controller architecture for inertia wheel pendulum.

The step response associated with the controller in Figure 4 can be seen below.

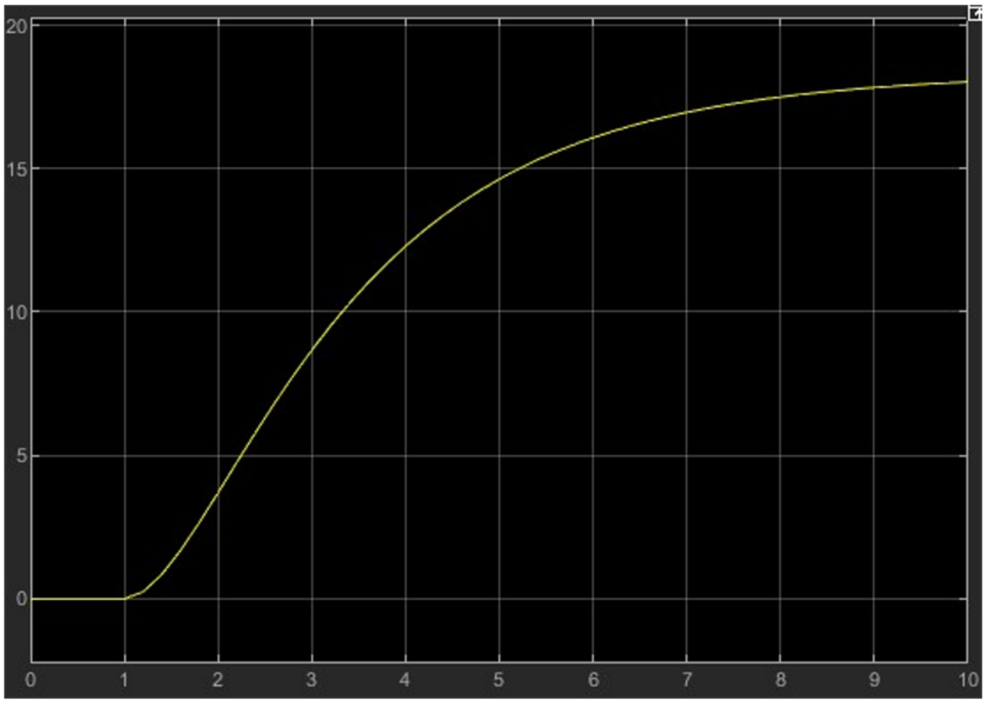

Figure 5: Step response of feedback controller.

A PID controller was also implemented and simulated. This controller tracks system error and regulates input accordingly - this architechture can be seen below.

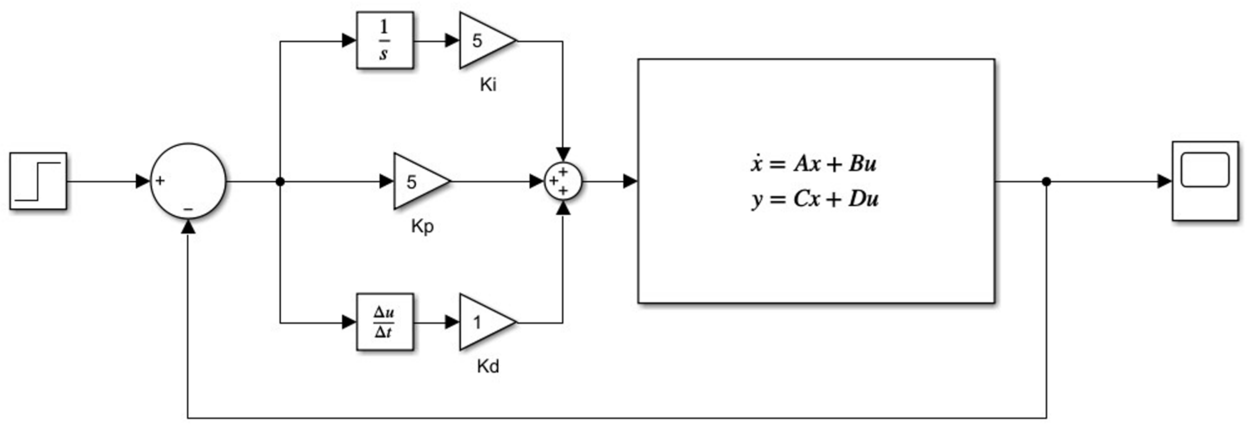

Figure 6: PID controller architecture for inertia wheel pendulum.

The step response associated with the controller in Figure 6 can be seen below.

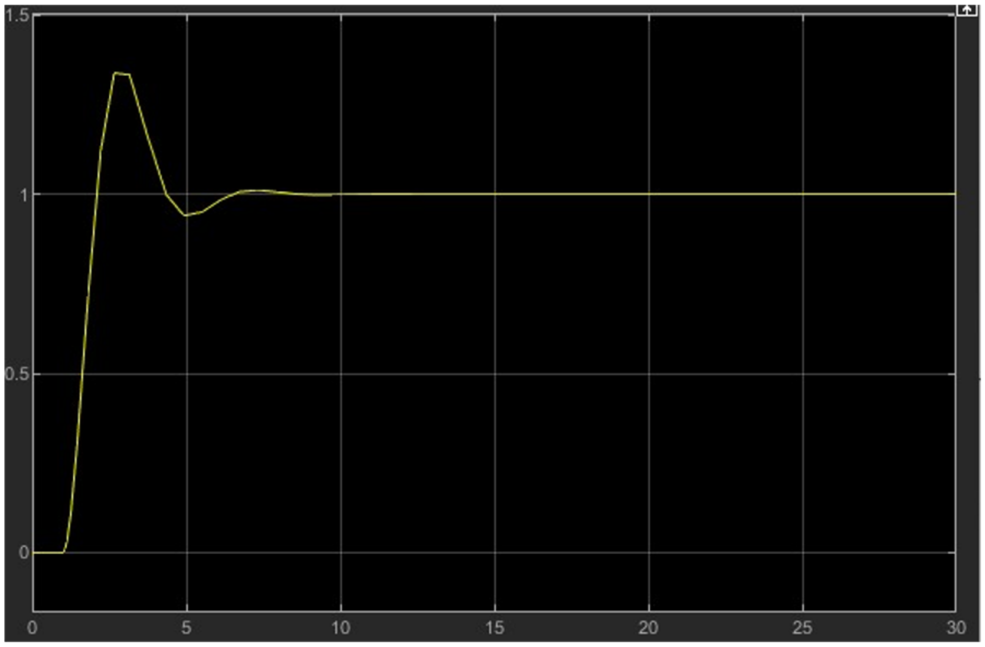

Figure 7: Step response for PID controller.

The inertia wheel pendulum was also modeled in CopelliaSIM. This model can be seen below.

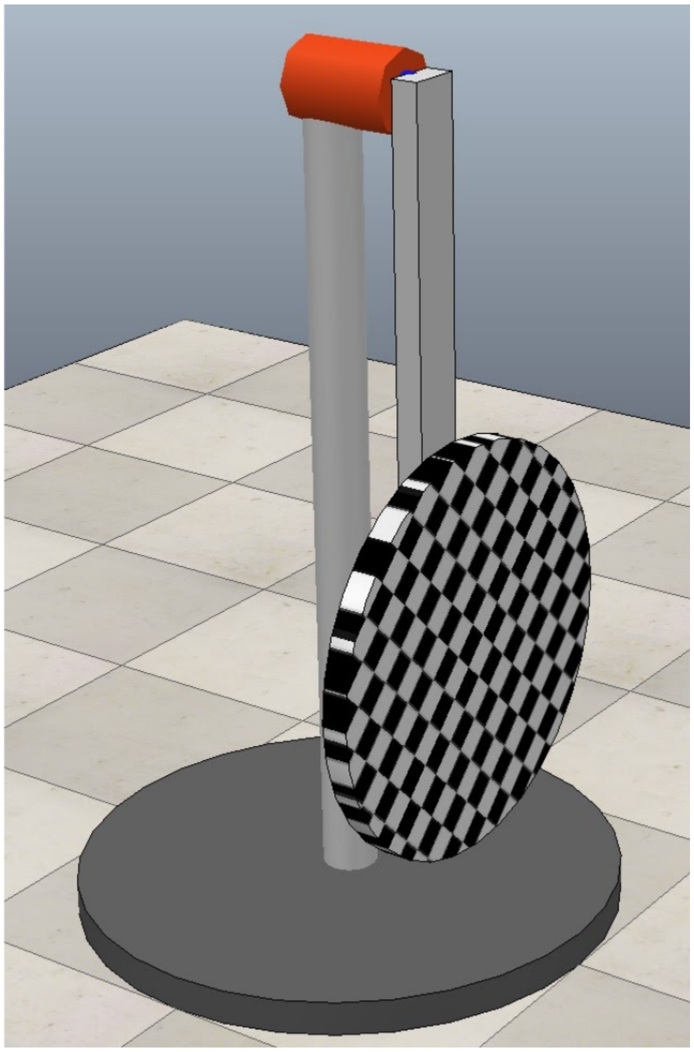

Figure 8: CopelliaSIM model for inertia wheel pendulum.

A video of the actual simulation can be seen here.

Appendix A: Simulation Code

▼ MATLAB Full State Feedback

```
clc, clear all, close all;

% Define system matrices
g = 9.81
r = 0.5
m1 = 2
m2 = 1
l = 1
lc = 0.25*l
mbar = m1*lc+m2*l
J1 = (1/3)*m1*l^2
J2 = (1/2)*m2*r^2
```

```
A = [0 1 0; mbar*g/(J2*(1-m1*lc^2+m2*l^2+J1+J2)) 0 0; -mbar*g/(J2*(1-
m1*lc^2+m2*l^2+J1+J2)) 0 0]
B = [0; 1/J2+(m1*lc^2+m2*l^2+J1+J2)/(J2*(1-m1*lc^2+m2*l^2+J1+J2)); -
(m1*lc^2+m2*l^2+J1+J2)/(J2*(1-m1*lc^2+m2*l^2+J1+J2))]
C = [1 0 0];
D = 0;


check1 = A(2,1)
check2 = A(3,1)
check3 = B(2,1)
check4 = B(3,1)
% Create state space object
sys = ss(A,B,C,D);

% Check open-loop eigenvalues
E = eig(A);

% Desired closed-loop eigenvalues
P = [0, -0.5, -1.5];

% Solve for K using pole placement
K = place(A,B,P);

% Check for closed-loop eigenvalues
Acl = A-B*K;
Ecl = eig(Acl);
detAcl = det(Acl)

% Closed-loop system
syscl = ss(Acl, B, C, D);

Kr = 1/dcgain(syscl);
syscl_scaled = ss(Acl, B*Kr, C, D);

% Step response of the system
%step(syscl);
step(syscl_scaled);
%step(sys)
```

▼ MATLAB API

```
wheel=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
wheel.simxFinish(-1); % just in case, close all opened connections
```

```
clientID=wheel.simxStart('127.0.0.1',19999,true,true,5000,5);

w = 30;

 if (clientID>-1)
        disp('Connected to remote API server');
            %output

[returnCode,motor_encoder]=wheel.simxGetObjectHandle(clientID,'motor_encoder',wheel.

            %input

[returnCode,encoder1]=wheel.simxGetObjectHandle(clientID,'encoder',wheel.simx_opmode



[returnCode,encoder]=wheel.simxGetJointPosition(clientID,encoder1,wheel.simx_opmode_




%
[returnCode,encoder]=wheel.simxGetIntegerParameter(clientID,encoder,wheel.simx_opmod

             %Execute This
        %Moves forward

        while (1)

[returnCode,encodernum]=wheel.simxGetJointPosition(clientID,encoder1,wheel.simx_opmo


  %
[returnCode,encoder]=wheel.simxGetIntegerParameter(clientID,encoder,wheel.simx_opmod


 encoderout = (encodernum * (180 / 3.14))

%          data = '{}\n'.format(encoderout)

[returnCode]=wheel.simxSetJointTargetVelocity(clientID, motor_encoder, w
,wheel.simx_opmode_blocking);
             pause(0.5)

        end

        wheel.simxFinish(-1);

 end
```

```
wheel.delete(); % call the destructor!
```

Other simulation files can be found here.

Appendix B: Project Documents

- Presentation Slides
- Presentation Video
- GitHub .pdf version

References

[1] D. Dixon, “Shared Memory Plugin For VREP /Matlab Communication,” CopelliaSim. [Online]. Available: https://forum.coppeliarobotics.com/viewtopic.php?f=9&amp;t=4157&amp;hilit=simulink.

[2] Hernández-Guzmán Victor Manuel and Silva-Ortigoza Ramón, Automatic control with experiments. Cham, Switzerland: Springer, 2019.

[3] L. Arnesto, “Object hierarchy relations | Coppeliasim (V-rep),” YouTube, 31-Jan-2020. [Online]. Available: https://www.youtube.com/watch?v=HyqD140boOw.

[4] N. S. Nise, Control Systems Engineering. Wiley, 2015.

[5] O. D. Montoya and W. Gil-González, “Nonlinear analysis and control of a reaction wheel pendulum: Lyapunov-based approach,” Engineering Science and Technology, an International Journal, vol. 23, no. 1, pp. 21–29, 2020.