# FINAL PROJECT

ME190: Mechatronic Systems Design

Winncy Du

San Jose State University

Edwin Navarro

Anthony Peters

Dec. 6, 2022

# Table of Contents

# 1 Summary

In this Capstone Project we were called to design and tune a controller for the MinSeg Robot such that it balances on its own. The MinSeg Robot, at its foundation, is an inverted pendulum and can be modeled as such. For the MinSeg Robot to balance autonomously, there must be a controller written and flashed onto the MinSeg's MCU. Thankfully MATLAB is a perfect tool for the job as there are various types of toolboxes that can be used to complete this process. However, before using MATLAB and Simulink there must be a mathematical model and Free Body Diagrams that represents the system at hand. After the necessary derivations of the model are completed, we can simulate and plot the system's dynamic response and design toward the desired performance criteria with various inputs that are constrained within our assumption. From here we model the MinSeg hardware through Simulink and build various types of control systems that can take in our simulated gain values calculated with MATLAB. Once we build our Simulink Model we can then test, run, and tune our MinSeg System Response to achieve desired results. Fine-tuning the Controller is an inherent part of controller design as the optimal processes in theory do not work as elegantly as we would hope.

# 2 Derivations of Model Equations

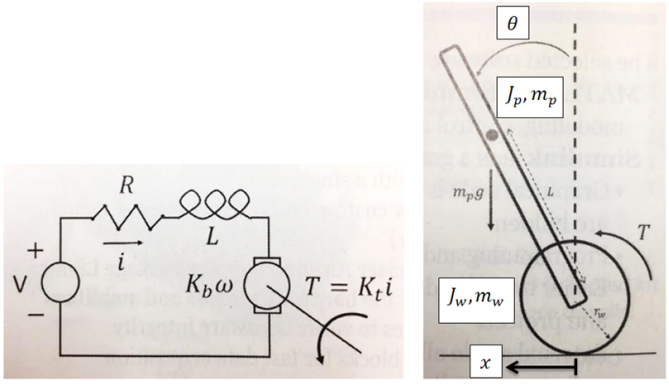


*Figure 1 Electro-Mechanical Diagrams of the MinSeg robot*

The dynamic model of the MinSeg requires an understanding of Rigid body dynamics such as Kinematics and Kinetics. The mindset can be broken down into two coupled rigid bodies. We first find the equations of motion of the wheel and the equation of motion of the pendulum. Then, we use the Kirchhoff equation to reduce or two equations so that the input is the voltage supplied to the motor. Finally, we reduce the equations so that our states are only in terms of the position and angle of the pendulum.

## 2.1 Equations of Motion of the wheel

Equation of motion of the wheel. We start with the moments of the motor shaft which include the moment of inertial, torque, viscous effects, and the reaction forces with the pendulum.

$$\sum M_C = \bar{I}\alpha + m\bar{a}d$$

$$\tau - c(\dot{\phi} - \dot{\theta}) - F_x r_w = J_w\ddot{\phi} + m_w\ddot{x}r_w$$

Assuming No slip condition we can replace $\phi = r_w x$ ; $\dot{\phi} = r_w\dot{x}$…. We then obtain equation 1.1

$$\frac{J_w}{r_w}\ddot{x} + m_w r_w\ddot{x} + c\frac{\dot{x}}{r_w} - c\dot{\theta} = \tau - F_x r_w \qquad \text{Eq 1.1}$$

## 2.2 Equations of Motion of the pendulum

Equations of motion for the pendulum we can find the reaction forces at the shaft.

$$\sum F_x = m_p\bar{a}_x$$

$$F_x = m_p(\ddot{x} + l_p\ddot{\theta}cos\theta - l_p\dot{\theta}^2 sin\theta) \qquad \text{Eq. 1.2}$$

$$\curvearrowleft \sum M_B = \bar{I}\alpha + m\bar{a}d$$

$$m_p g l_p sin\theta - \tau + c(\dot{\phi} - \dot{\theta}) = J_p\ddot{\theta} + m_p\left(l_p^2\ddot{\theta} + \ddot{x}l_p cos\theta + l_p\dot{\theta}^2(0)\right) \qquad \textbf{Eq 2.0}$$

Subbing $\dot{\phi} = \frac{\dot{x}}{r_w}$

$$m_p g l_p sin\theta - \tau + c\left(\frac{\dot{x}}{r_w} - \dot{\theta}\right) = J_p\ddot{\theta} + m_p\left(l_p^2\ddot{\theta} + \ddot{x}l_p cos\theta\right)$$

Combining Eq 1.1 and Eq 1.2 by eliminating $F_x$

$$\tau = \left(\frac{J_w}{r_w} + r_w m_w + r_w m_p\right)\ddot{x} + \frac{c}{r_w}\dot{x} + r_w m_p cos\theta\ddot{\theta} - r_w m_p l_p\dot{\theta}^2 sin\theta - c\dot{\theta} \qquad \textbf{Eq 1.0}$$

## 2.3 Electric Circuit Model Equation

Motor equation can be related to the wheel rotation.

$$L\frac{di}{dt} + Ri + k_b(\dot{\phi} - \dot{\theta}) = v$$

$$\tau = k_t i = k_t\left(\frac{v}{R} - \frac{k_b}{R}\left(\frac{\dot{x}}{r_w} - \dot{\theta}\right)\right)$$

$$\tau = \frac{k_t v}{R} - \frac{k_t k_b}{R r_w}\dot{x} + \frac{k_t k_b}{R}\dot{\theta} \qquad \textbf{Eq 3.0}$$

### 2.4 Linearization and State Space Representation

Subbing 3.0 into 1.0 and into 2.0 will further reduce or variables so that we can use state space. One needs to note that equations 1.0 and 2.0 are not linear. Therefore, we apply small angle condition to linearize to obtain our linear equations.

$sin\theta \approx \theta \; ; cos\theta \approx 1 \; ; \dot{\theta}^2 sin\theta \approx 0$

$$\left(r_w m_p + r_w m_w + \frac{J_w}{r_w}\right)\ddot{X}(t) + \left(\frac{c}{r_w} + \frac{k_t k_b}{R r_w}\right)\dot{X} + (r_w m_p l_p)\ddot{\theta} - \left(c + \frac{k_t k_b}{R}\right)\dot{\theta} = \left(\frac{k_t}{R}\right)V \qquad \textbf{Eq 4.0}$$

$$(-m_p l_p)\ddot{X}(t) + \left(\frac{c}{r_w} + \frac{k_t k_b}{R r_w}\right)\dot{X} - (J_p + l_p^2 m_p)\ddot{\theta} - \left(c + \frac{k_t k_b}{R}\right)\dot{\theta} + (m_p g l_p)\theta = \left(\frac{k_t}{R}\right)V \qquad \textbf{Eq 5.0}$$

Linearized Matrix Form using equations 4.0 and 5.0.

$$\begin{bmatrix} r_w m_p + r_w m_w + \frac{J_w}{r_w} & r_w m_p l_p \\ -m_p l_p & -(J_p + l_p^2 m_p) \end{bmatrix}\begin{bmatrix}\ddot{X} \\ \ddot{\theta}\end{bmatrix} + \begin{bmatrix} \frac{c}{r_w} + \frac{k_t k_b}{R r_w} & -\left(c + \frac{k_t k_b}{R}\right) \\ \frac{c}{r_w} + \frac{k_t k_b}{R r_w} & -\left(c + \frac{k_t k_b}{R}\right) \end{bmatrix}\begin{bmatrix}\dot{X} \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & m_p g l_p \end{bmatrix}\begin{bmatrix} X \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{k_t}{R} \\ \frac{k_t}{R} \end{bmatrix} V$$

OR

$$[M]\begin{bmatrix}\ddot{X} \\ \ddot{\theta}\end{bmatrix} + [C]\begin{bmatrix}\dot{X} \\ \dot{\theta}\end{bmatrix} + [K]\begin{bmatrix} X \\ \theta \end{bmatrix} = [F]V$$

**State Space Representation**

$$\dot{q}(t) = Aq(t) + Bu(t)$$

$$q(t) = \begin{bmatrix} X \; \dot{X} \end{bmatrix}^T = \begin{bmatrix} x \; \theta \; \dot{x} \; \dot{\theta} \end{bmatrix}^T, \qquad A = \begin{bmatrix} 0_{2x2} & I_{2x2} \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \qquad B = \begin{bmatrix} 0_{2x1} \\ M^{-1}F \end{bmatrix}$$

# 3 Systems Diagrams

We created two body diagrams for our system. One for the wheel and one for the pendulum. Diagrams for control systems were then implemented on MATLAB's Simulink.

### 3.1 Free Body Diagrams

## 3.1.1 Free Body for Wheel

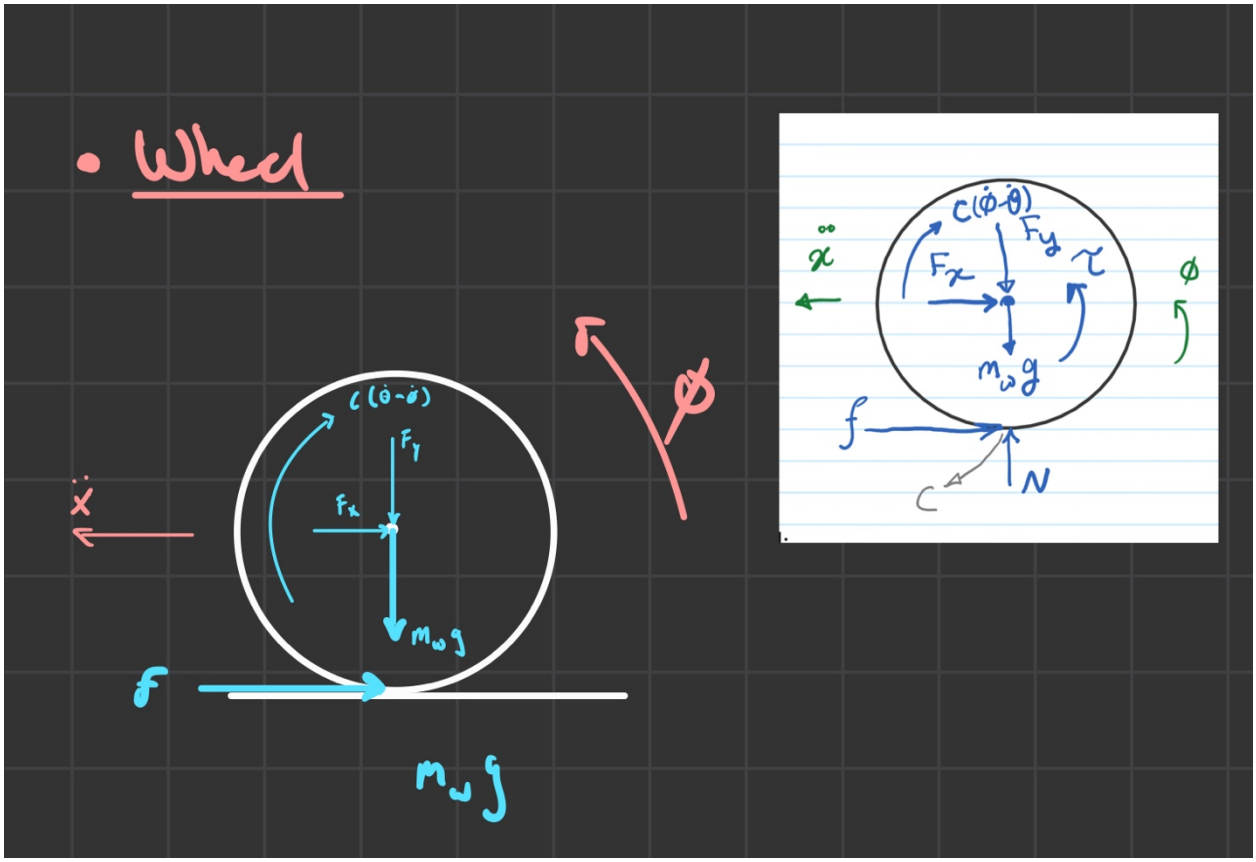


*Figure 2. Wheel FBD*

### 3.1.2 Free Body for Pendulum

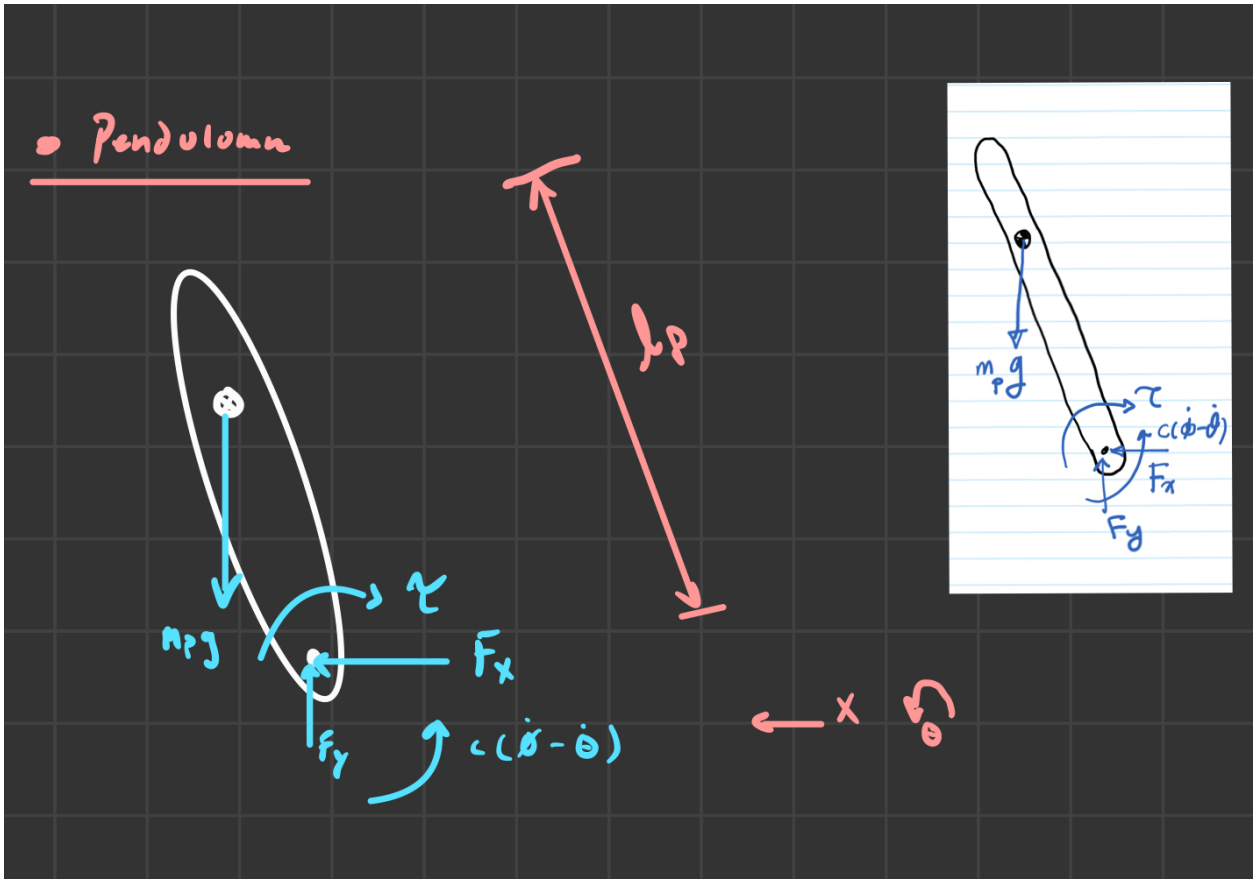


*Figure 3 Pendulum FBD*

## 3.2 Electro-Mechanical

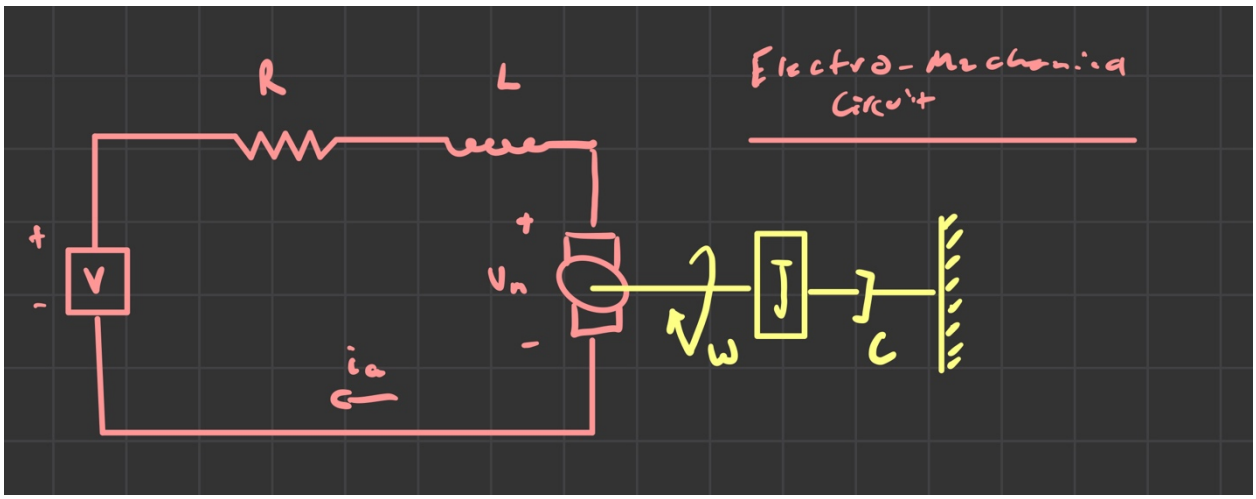


*Figure 4. Electromechanical Schematic*

## 3.3 Simulink Model

To Control the Minseg we created a Simulink Diagram. From left to right, our diagram contains sensor information so that we can apply control gains. Then on the right side the motor diagram is used to send control effort to the motor to maintain the MinSeg at equilibrium.

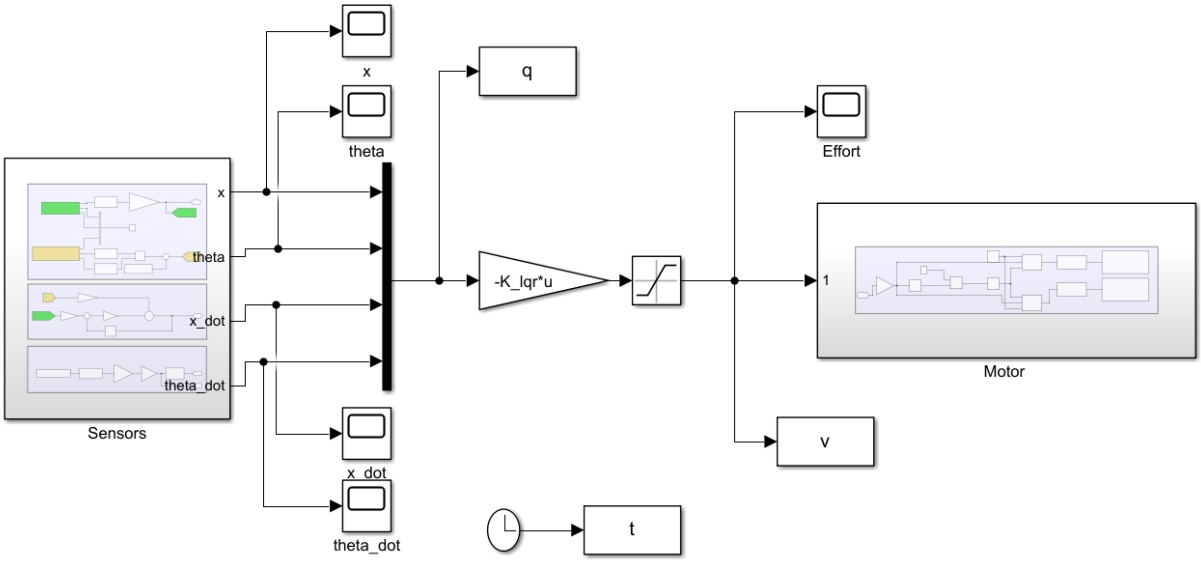


*Figure 5. Simulink of our system*

The sub systems called sensors (left side of figure 2) shows all the states required to control the system. To obtain displacement data, we the encoder. And to get angle of the pendulum we create a complimentary filter so that our states is rather stable. By having a Gyro scope, we guarantee that we are integrating the error.

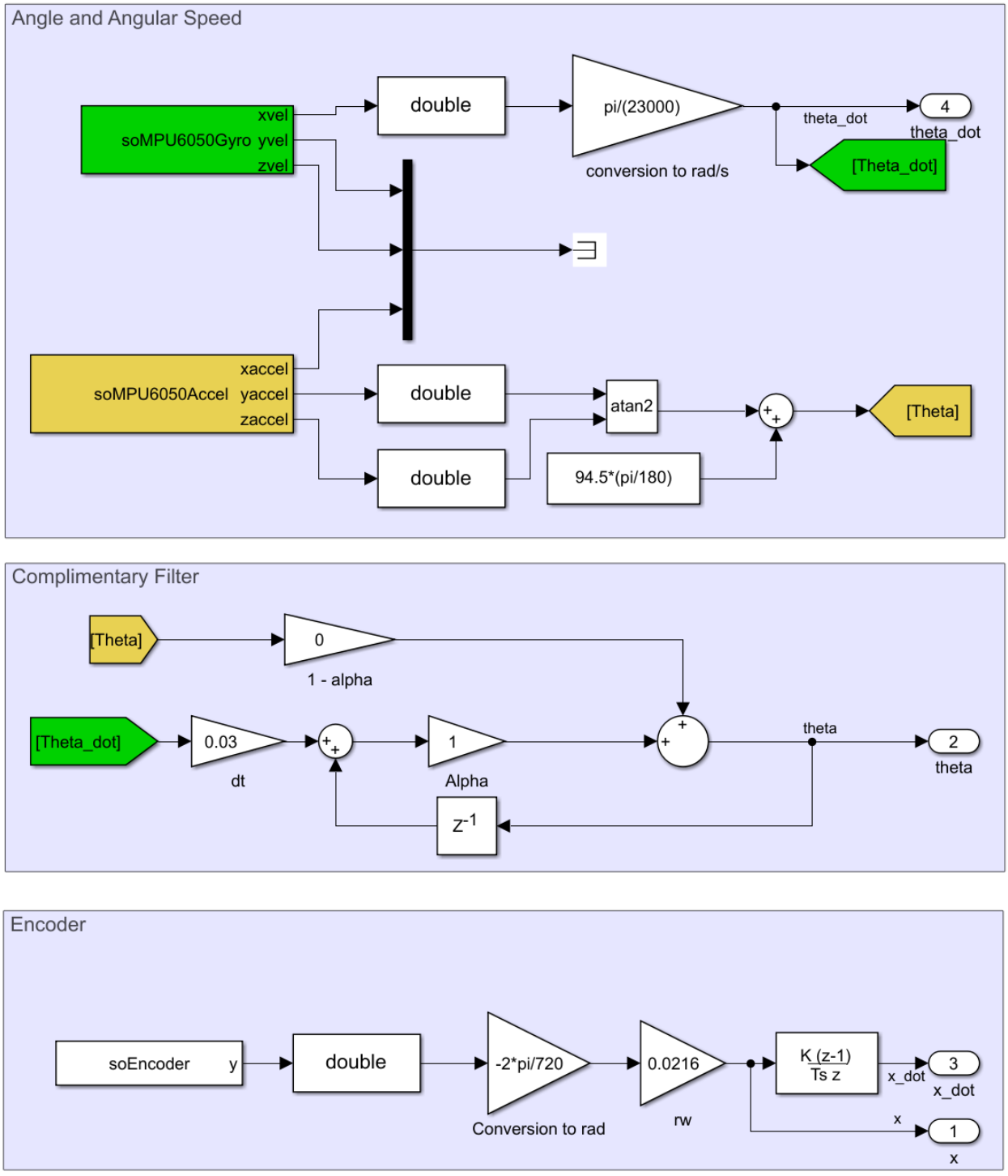


*Figure 6. 5Sensors to collect the states of our system.*

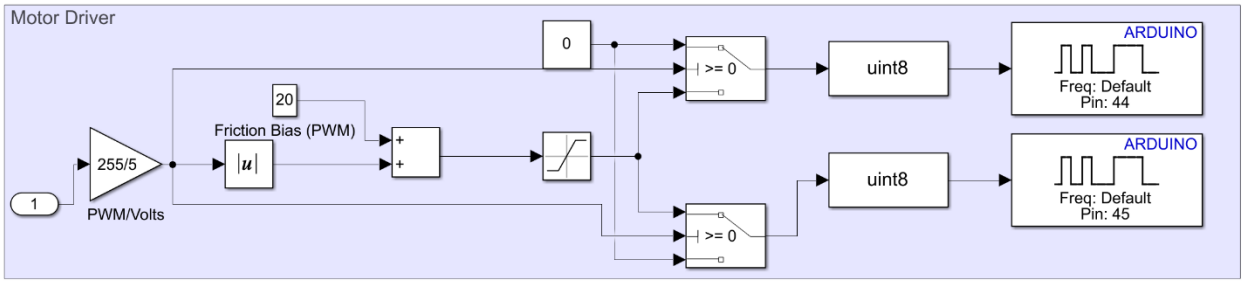


*Figure 7. 6Motor Driver*

The "magic" numbers shown in figure 3 represent values specific to our system. For example since we are sampling at 0.03s

# 4 Implementing the Controller

## 4.1 Open Loop system (No Controller)

The plant alone does not have any control and figure

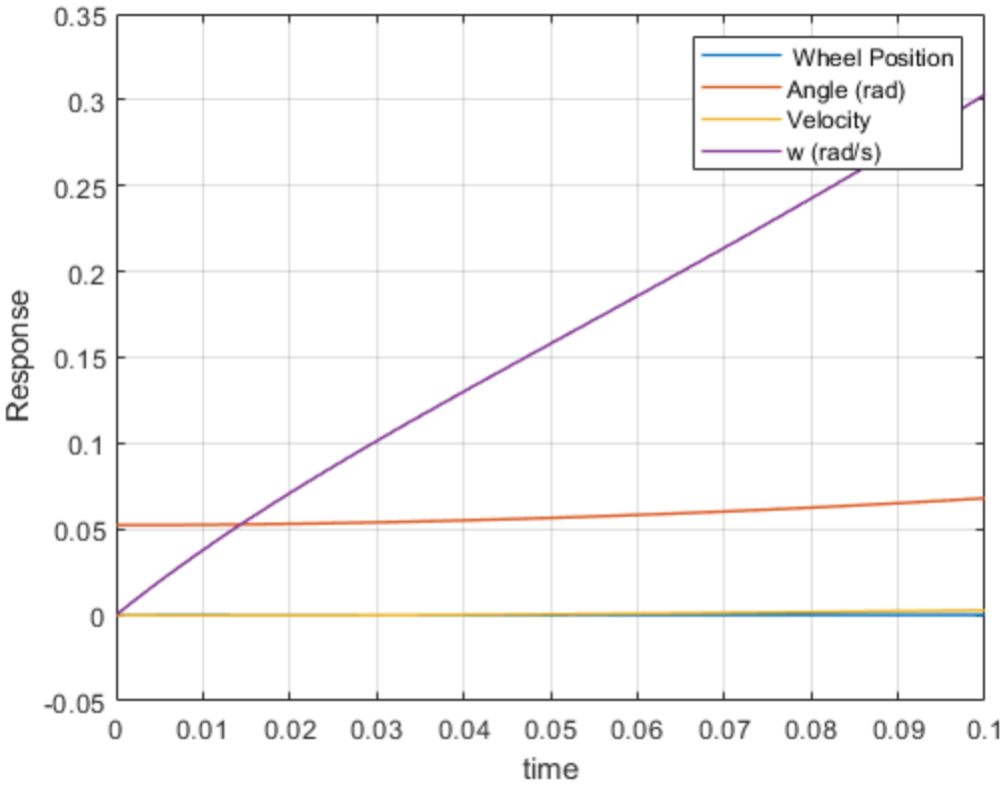


*Figure 8.7 Open Loop system Response*

We see in figure 5 that our system explodes and does not seem to go to a steady state. This was expected as we don't have any controller to specify our performance criteria.

## 4.2 Full State Feedback Controller

To implement the feedback controller, we used acker method using MATLAB. Acker solves for the gains of our controller. The controller feedback law is $u = -Kx$ where K is a Nx1 vector meaning that there is a gain value associated with each state. By using acker method, one can arbitrarily select where we want our closed loop poles for the system according to stability criterion; however, one must note that acker only works if our system is controllable. This could be checked by using the following command and matrixes from the state space representation.

$$rank\ (B\ AB\ A^2B\ A^3B) = length(A) = 4$$

Only if the above is true can we continue to use this function on MATLAB.

$$K = acker(A, B, \lambda_{desired})$$

Ultimately, our state space representation becomes. And we are able to simulate this on MATLAB to find eigen values as well as provide initial conditions.

$$\dot{q}(t) = (A - BK)q(t)$$

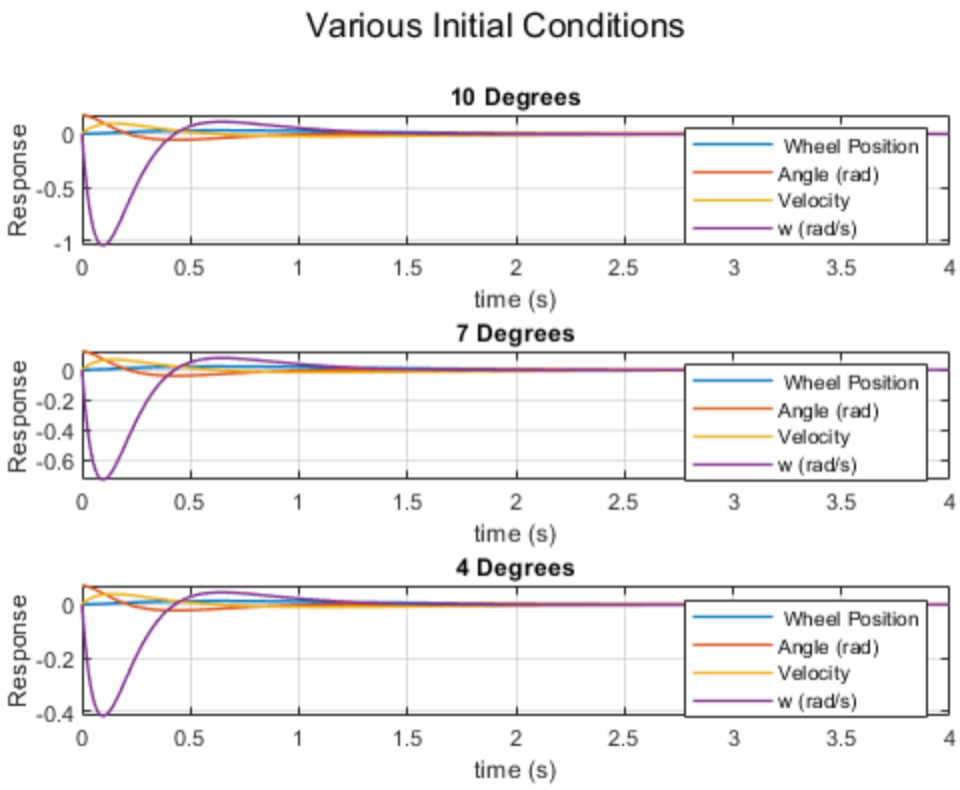


*Figure 6 Respond of our system with Full State feedback controller*

Figure 6 shows the response to initial conditions. For example at an angle of 10 degrees our system is able to balance itself.

## 4.2 Linear Quadratic Regulator

For a Linear Quadratic Regulator (LQR) we change weights on the states on Q try to minimize the cost function by adjusting performance and control effort.

$$J = \int_{t_0}^{T} (q^T Q q + u^T R u) dt$$

For the MinSeg we selected Q and R so that the control effort (voltage input) is limited to 5V as this was the voltage available. While making sure that our angle ($\theta$) was the most weighted so that the controller would prioritize keeping the error minimal for the pendulum.

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 5200 & 0 & 0 \\ 0 & 0 & 69 & 0 \\ 0 & 0 & 0 & 77 \end{bmatrix}$$

$$R = 1$$

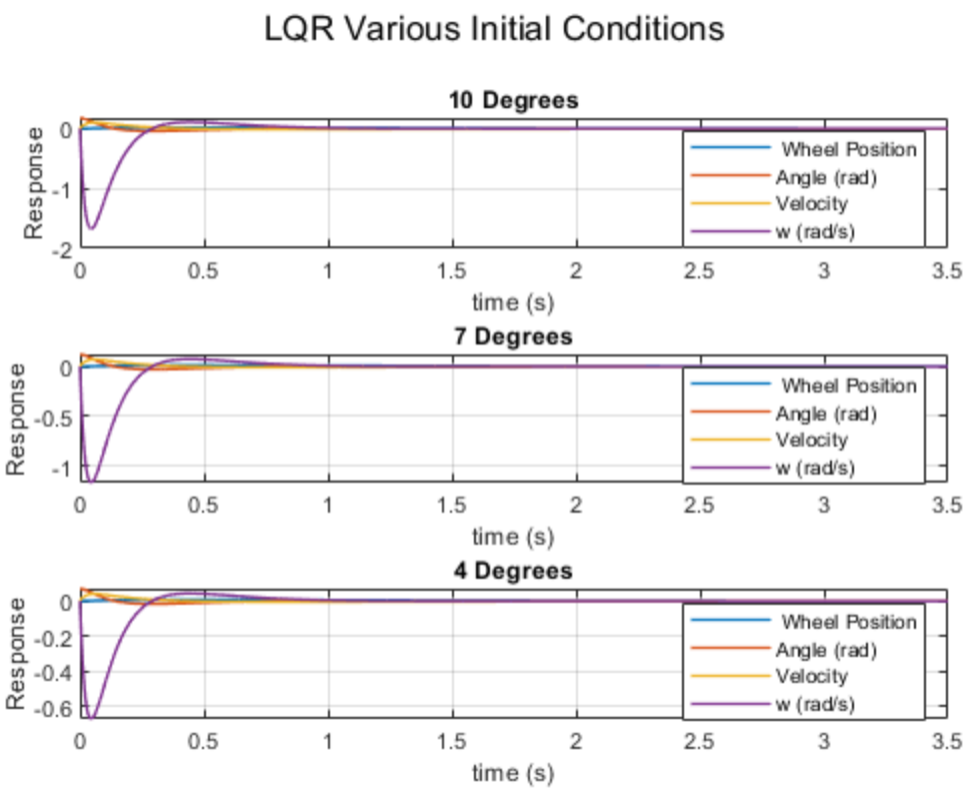


*Figure 7 LQR Controller system Response*

To implement the controller, we needed to adjust Q and R. The general rules are:

1. If Q>R then states decay to zero quickly
2. If Q< R then system will have slow response and the control effort is small

## Discussion

During the Project, we ran into multiple problems while tuning the dynamics of the system. One of the problems that we encountered was that the MinSeg would stop running during the Monitor and Tune Process. We debugged this issue by changing the cables and found that the system reacted faster and much more crisply when using a thicker gauge wire. We reasoned this improvement to come from an increase in voltage and bandwidth supplied by a wider wire that allowed more data to be transferred at the required speed. Another problem that arose was the lack of understanding of LQR and the Matrixes that we were supposed to iterate to improve system performance. We approached this problem from intuition learned from ME 280 where we were taught how the Q and R matrix relates to the systems modeled variables. Knowing which scalar weights to which variable is very useful when adjusting your Q matrix because one can evaluate current system performance in real time and make an educated guess in how to change the dynamics of the system through Q. If we were not to take ME 280 in unison with this class, it would be difficult to understand this concept and interaction process.

## Discussion Questions

Q. Is the open-loop system stable? Set the simulation time to 0.1 s to see how the states are evolving at the beginning:

**A.** Open loop system is not stable as the plant emulates a unit ramp function

**Q.** Can you relate the directions of x and θ to the physics of the system when you leave the robot from a small initial angle?

A. Yes, you can relate x and θ to the physics of the system when you leave the robot from a small initial angle because there is potential and kinetic energy that comes into play, when the robot tilts at an angle the potential energy decreases but the kinetic increases, tocounteract the kinetic energy, the motor engages and rotates in the direction of the tilt to catch itself from falling.

**Q.** Check to see what happens if you use the same weights for x and Θ. Can your controller still be stable? How can you justify the results?

A. The controller is not stable when we set both x and theta values to the same. In the system, we need to account for angle much more than we need to account for the x position so that when a change in theta occurs the system can react immediately

**Q.** Try to use the accelerometer sensor only for Θ. Change its gain from 0.01 to 1, and change the gyro gain from 0.99 to 0 and disconnect the delay to prevent accumulation of the gyro signal. What difference do you observe in the behavior of the system? Now try to use the gyro sensor only for measuring the angle by disconnecting the accelerometer and changing the gyro gain back 1. What difference do you observe now? Is the complementary filter the savior of the robot?

**A.** The complementary filter is the savior of the system as the system achieves zeros steady-state error as time goes on. Without it the system would become more and more unstable as time increases.

**Q.** In your report, summarize what you learned through this project and whether you could connect the dots between theory and practice. Just a piece of food for thought: Think about how else you could have controlled the robot. Could you stabilize the robot without modeling, system identification, and optimal control? The answer is probably yes, with a lot of luck guessing the right values for the control gain vector, K!

A. This is seen in the above discussion

**Q.** Also in your report, list the possible sources of mismatch between the model and the actual system, and explain how they might have affected your system's response. Provide suggestions that can help mitigate them if you had more time to spend on the project. Some sources of error between the ideal model and our actual system come from 1: the physical location and solder quality of the sensors, and 2: fine-tuning certain parameters to achieve the desired response. Some values such as the gyro's angle bias had to be edited as in our case, the gyro/accelerometer did not sit perfectly flush with the Arduino board at 90 degrees. This resulted in the robot moving continuously in the forward direction as the sensor was slightly offset from the Arduino.

A. The model assumes that the Robot is at 90 degrees but because it's not we have to change this in our model. If I had more time on this project we can learn how to integrate the q and r values of our LQR so that we can get a crisper response in our systems.

# Conclusion

Completing this capstone project teaches us the skills of designing, modeling, and tuning a dynamic system. As we began deriving the equations for the design, we began to build significant intuition on the components of the system and their physical impact on the system. Because the differential equations of this system are cumbersome and have multiple inputs and outputs, we need to represent the Model in State Space. This mathematical representation is modeled in the continuous time domain. Hence, it is necessary for us to Discretize the system when we input our dynamics in Simulink so that the digital computer can read and manipulate the data in the control system.

One important thing to note is that our complimentary filter saves this system. One could not just run just the accelerometer or just the gyroscope. When attempting this our system becomes unstable.