MSE 312 - Mechatronics Design II

SFU

Electronics and Control Report

Development of the Electrical and Control Aspects of a Vertical Ball Launcher

Lab 2 Group #4

Hnat Cheung	-	301362123
Harman Poonian	-	301275750
Ryley McWilliams	-	301360562
Chris Vattathichirayil	-	301328090
Austen Ware	-	301340637
Kalana Wijeratna	-	301282332

I. Abstract

This report details the electronics and control design for our group's MSE 312 project. An overview of the final design and programming is first shown and described in regards to the control of the arm using a motor controller, nand gate, and Simulink. Calculations using linear control system methods and experimental trial results are provided and were found to be fairly similar. Simulations were conducted to initially aid the design process, and then physical experiments were done to further tune design parameters. The integration of the mechanical, electronics, and control systems is also described. Any limitations during the evaluation process are also detailed

II. Introduction

Our group was given the task of designing and creating electronics and control systems that work in conjunction to launch a small wooden ball into a bucket some specified distance away. We were given six weeks to design and optimize the interweaving mechanical, electrical, and code aspects of our project to their full potential. The materials supplied to do this were limited to a Toshiba TB6568KQ integrated circuit motor controller, Fairchild DM74LS00 nand gate integrated circuit, Lo-Cog DM74LS00 DC servo gearmotor with built-in encoder, breadboard, and wires. In order to determine what was the best design, we used Matlab and Simulink to help simulate the stability and behavior of the control system. In the following report, we will describe our electronics and control designs, as well as the integration of all parts of the project.

III. Electronics Design

Our design process began with the consideration of our necessary requirements. Our circuit needed to be able to take necessary signals and send it to a DC motor in a way that we would be able to control the motor. We were given a DC Motor, various logic gates and an H-Bridge. We were allowed to use any other electrical components we saw fit to include in our circuit. For our purposes, our circuit design didn't need to be anything very complex. We could have a choke to limit the current, or a fuse to provide safety. A smoothing capacitor could also be used in this situation, but we knew that the power supply already handles the voltage and the current protection by limiting the maximum output. The motor controller also had built-in overcurrent and overvoltage protection. We also were looking to make this circuit as simplistic as possible. With this in mind, we designed a circuit with as few components as possible. Our initial design can be seen in Figure 1.

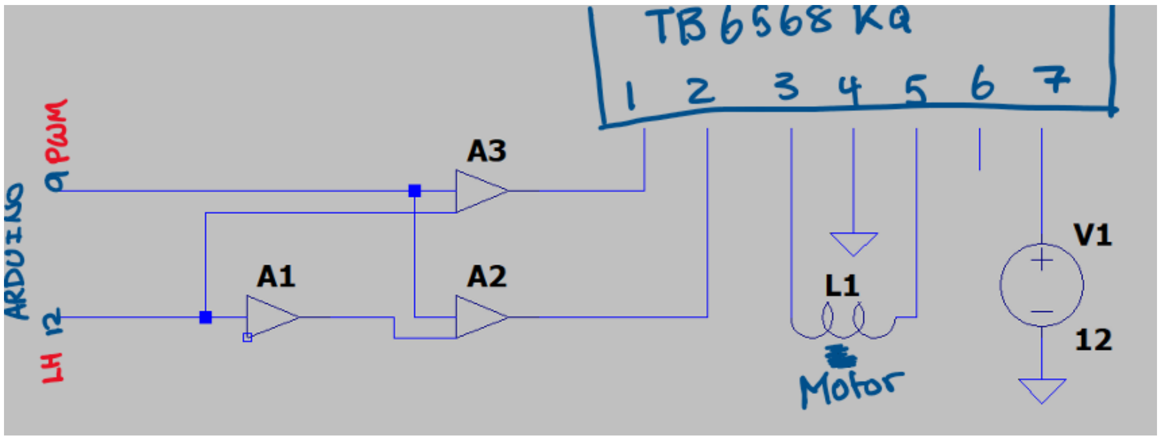


Figure 1: *Schematic of Intended Circuit Design*

Our circuit consists of three nand gates, an H-bridge, as well as an LED light not shown in our schematic to help us know when our system is primed. While the nand gate technically consists of four different gates, we decided that our circuit only required three out of the four gates. Once our schematic had been drawn, we were then able to bring the schematic to life on a breadboard. The first prototype of our circuit can be seen in Figure 2.

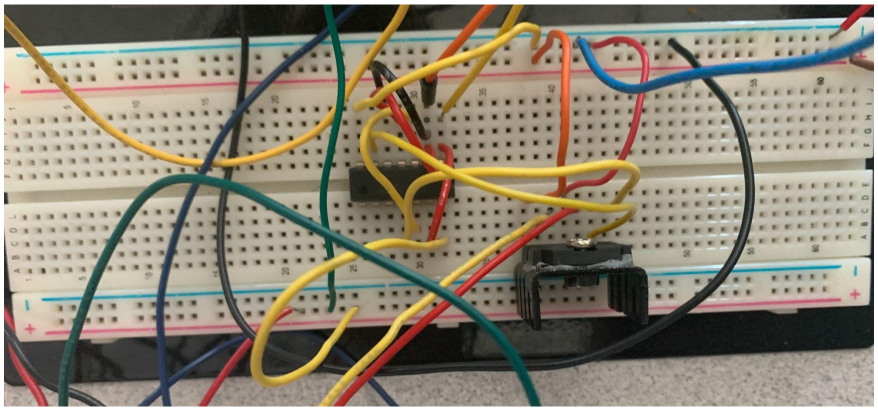


Figure 2: Breadboard Prototype

After making sure that our circuit sent the necessary signals to the necessary stations, we then went ahead and soldered our circuit onto a protoboard. This can be seen in Figure 3 below.

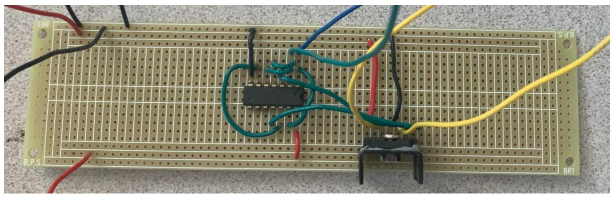


Figure 3: Protoboard Implementation

As mentioned before, we also included an LED light, which isn't shown in our circuits due to it not being an essential component. While our circuit is completed, there were still some modifications that needed to be made to the circuit. One of these modifications was the design of a heat sink for our H-bridge. The H-bridge that we were given can get fried quite easily, therefore we needed to calculate how big our heat sink needed to be. From our calculations, which can be seen below, we found that our heat sink needed to have an area of at least 1.82 cm^2. Once this had been determined, our circuit was ready to be plugged in, but we then needed to come up with some software to control what kind of signals were sent to the DC motor.

In order to make sure that our circuit worked the way that we wanted it to, we decided to run a couple of tests on it. The first test that we ran was to use an oscilloscope to make sure that a pulse was being sent into the circuit, and it was being transcribed into the correct signal that we were wanting to send to the DC Motor. This also allowed us to make sure that our encoder was giving us the values that we were going to use. Using a multimeter, we also measured our output current and our output voltage of the circuit while we ran our control system, in order to make sure we weren't going to short any of our components. From our tests, we saw that it was extremely difficult to maintain repeatable results throughout the entire process. After some investigation, we determined this was due to our power supply having some internal inductance. This means that the power supply can never catch up to the PID demands. The solution to this problem is to get a better power supply or put a limit on our PID requests. Overall, our results from our testing reveal that our circuit operates exactly how it was designed to.

IV. Control Design

Design Criteria

For this project, the goal is to complete 3 trials and our control design criteria will be based on these trials. The first trial is to launch a ball accurately at a target placed at a distance of 75cm. The second trial is to launch the ball into two vessels at positions in a 200cm range. The third trial is a mystery trial that will be revealed at a later date. A total time of 15 minutes is allotted to complete all 3 trials.

Based on the requirements of this project, we judge that accuracy is of the utmost importance as two of the trials require launching the ball at a specified target. Therefore we require our percentage overshoot to be less than 2%. Due to the time limit for completing the trials we have set a settling time of less than 2 seconds as the goal. These are the maximum values that we will accept and will strive to achieve values as low as possible for both of these.

Transfer Function Derivation

To design our controller we first derive the transfer function for the system. Newton's second law gives us the following equation:

$$\Sigma\tau = J\alpha \tag{1}$$

Where τ is the torque, J is the inertia, and $\boldsymbol{\alpha}$ is the angular acceleration of the motor. The total torque consists of the motor torque minus the viscous friction and static friction components. The motor torque is derived by multiplying the current by the torque constant, gear ratio, and efficiency parameters. This results in the following equation:

$$\eta n K_m i - \dot{B}\omega - \tau_s = J\alpha \tag{2}$$

Where η is the efficiency, n is the gear ratio, K_m is the motor constant, i is the current applied to the motor, B is a damping parameter, ω is the angular velocity of the motor, and τ_s is the static friction. Using Kirchoff's Voltage Law with the equivalent resistor, inductor, and back emf voltage circuit representation of a motor gives the following equation:

$$V = L\frac{di}{dt} + Ri + K_m n\omega \tag{3}$$

Where V is the voltage applied to the motor, L is the motor inductance, R is the armature resistance, and $K_m n\omega$ is the back emf voltage. Taking the Laplace transforms of equations 2 and 3 gives the following equations:

$$V(s) = LsI(s) + RI(s) + K_m ns\theta(s) \tag{4}$$

$$Js^2\theta(s) = \eta nK_m I(s) - Bs\theta(s) - \tau_s(s) \tag{5}$$

Equating these two equations through the current and angular position, we can create the following block diagram of the system:

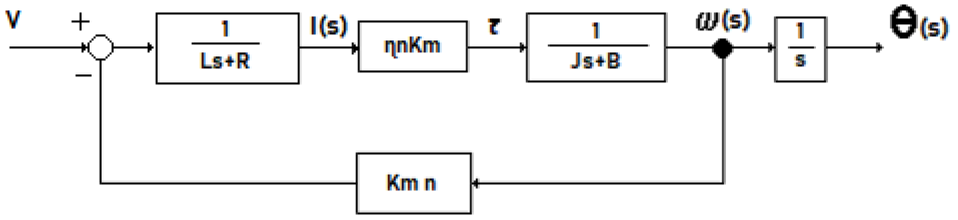


Figure 4: System Block Diagram in S-Domain

From this block diagram we can obtain the transfer functions for speed and position which are as follows:

$$\frac{\omega(s)}{V(s)} = \frac{\eta nK_m}{(Ls+R)(Js+B) + \eta n^2 {K_m}^2} \tag{6}$$

$$\frac{\theta(s)}{V(s)} = \frac{\eta n K_m}{s[(Ls+R)(Js+B) + \eta n^2 {K_m}^2]} \tag{7}$$

Based on feedback from the TAs during lab sessions, we have chosen to create a design that utilizes position and speed control to launch the ball at the desired speed and angle. This is done due to the ease with which the system's parameters can be modified during physical experimentation, which will be necessary as we expect there to be external factors that cannot all be accounted for in our calculations which will impact our results. To implement this design, we will be making use of two PID controllers, one each for position and speed control. The simulink block diagram is shown in Figure 5 below.

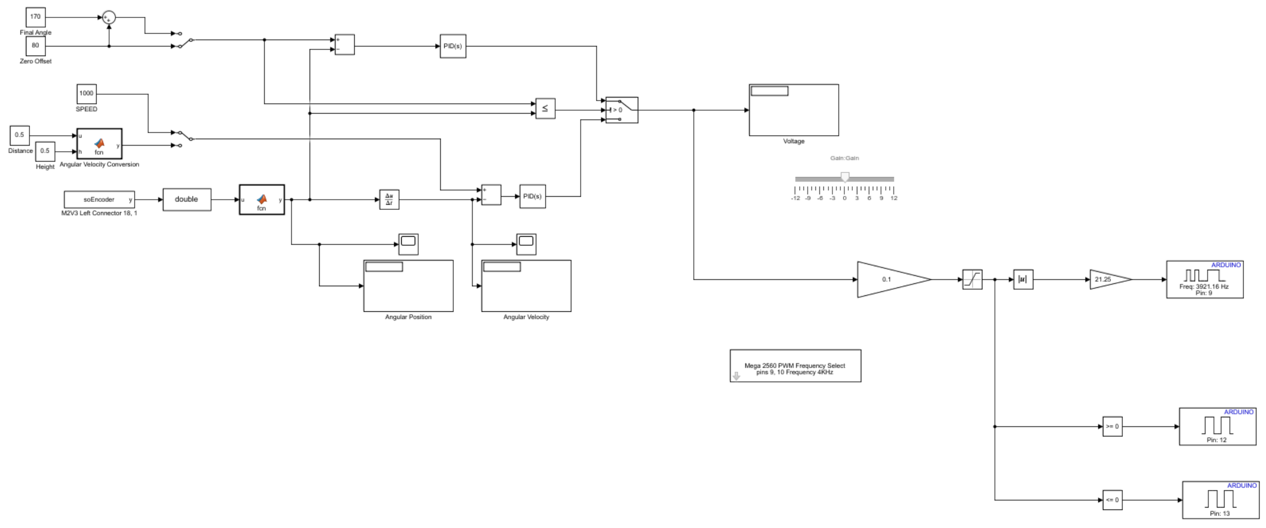


Figure 5: Simulink Block Diagram

The simulink file is also attached in a zip file provided with this report for further reference.

Controller Design

In order to effectively design our PID controller, we make use of transient analysis and root locus techniques in order to determine PID values for both position and speed control. These techniques are also used to study results and support our design process. The block diagram for the system is shown below.

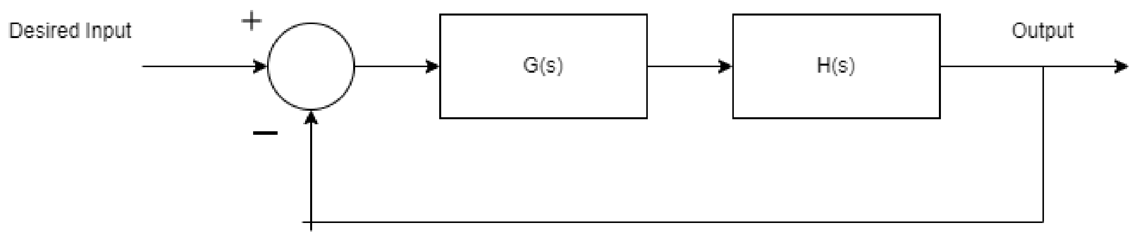


Figure 6: High Level System Block Diagram

Where the desired inputs and output will be either angular velocity or angular position, G(s) is the controller transfer function, and H(s) is the transfer functions from speed and position as obtained from equations 6 and 7.

The PID gains can be determined by the placement of zeroes as seen in equations 8 through 10 below in the manipulated form of the PID transfer function as a product of PD and PI controller gains. In other words we can choose values of Zc1 and Zc2 and this would result in a particular set of PID controller gain values. As a result by placing zeroes at different locations the shape of the root locus would also change. Each point in a root locus plot corresponds to a different gain, overshoot and settling time and in this way the root locus plot can be observed to see which set of gain, overshoot and settling time is ideal.

$$G_{PID}(s) = K_p + K_d s + \frac{K_I}{s} = K_d \frac{s^2 + \frac{K_p}{K_d} s + \frac{K_I}{K_d}}{s} = K_d\left(s + z_{c_1}\right)\left(\frac{s + z_{c_2}}{s}\right) = G_{PD}(s) G_{PI}(s) \tag{8}$$

$$z_{c_1} + z_{c_2} = \frac{K_p}{K_d} \tag{9}$$

$$z_{c_1} z_{c_2} = \frac{K_I}{K_d} \tag{10}$$

Equation 8 can be modified into proportional, PI, or PD versions by removing either one or both of the derivative and integral components of the equation.

The following root loci were plotted for position control and speed control by placing a zero at s = -20.

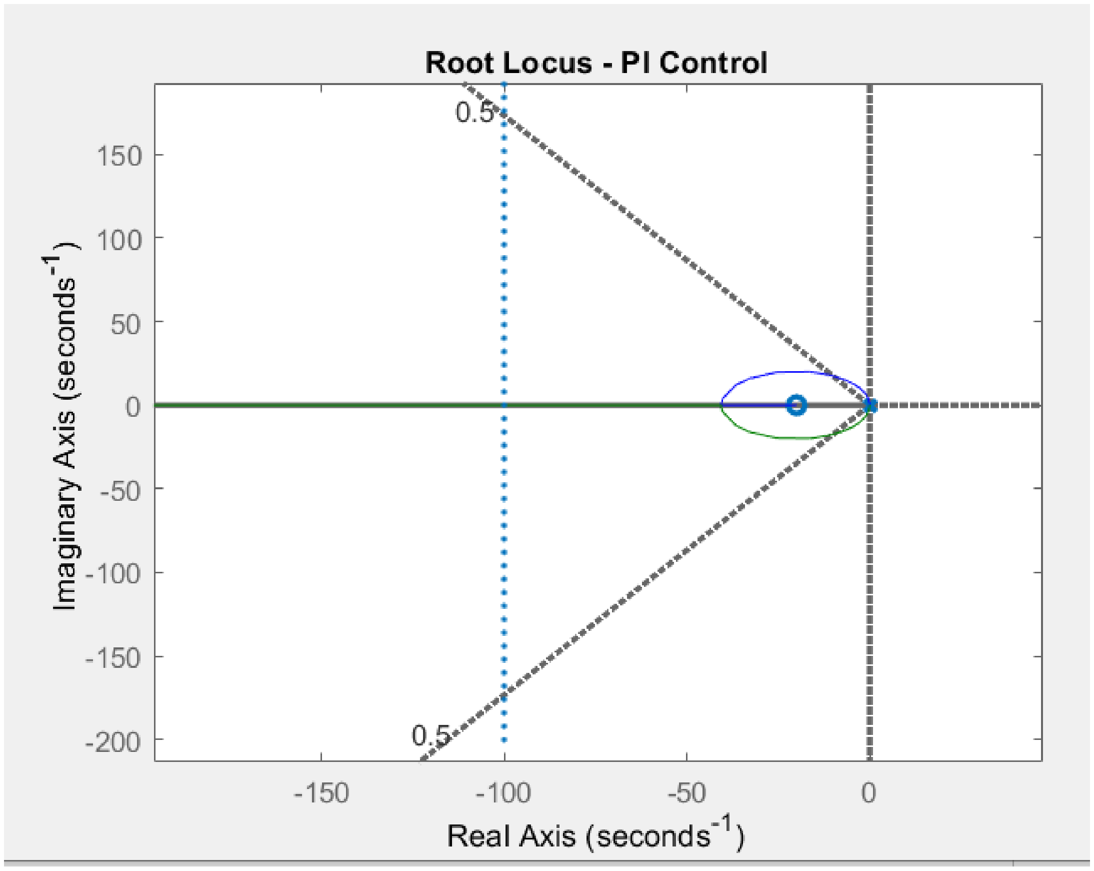


Figure 7: Graph of Root Locus for Position Control

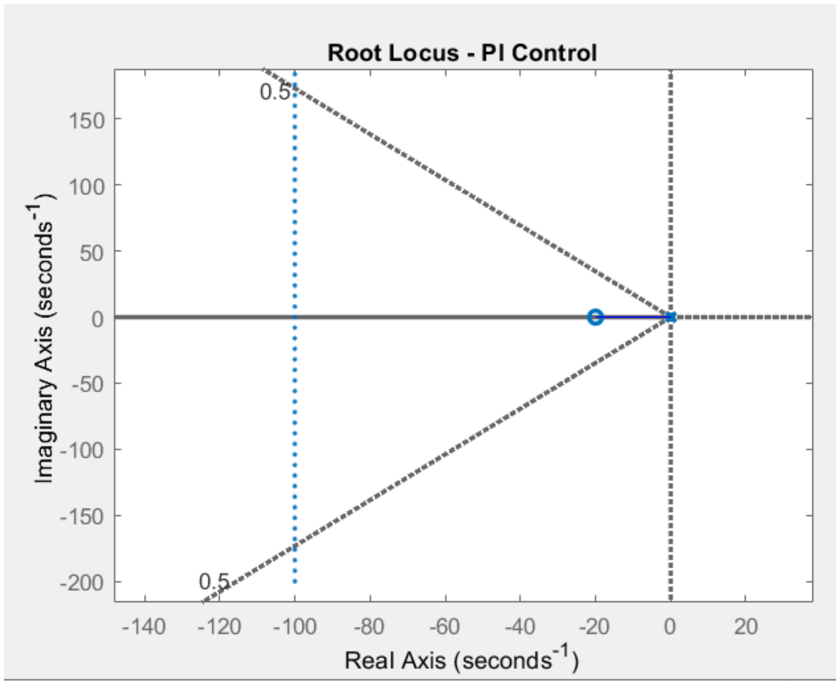


Figure 8: Graph of Root Locus for Speed Control

For transient analysis, we first obtained the step response for a proportional control system with varying gain parameters in order to observe the results on the step response for angular velocity.

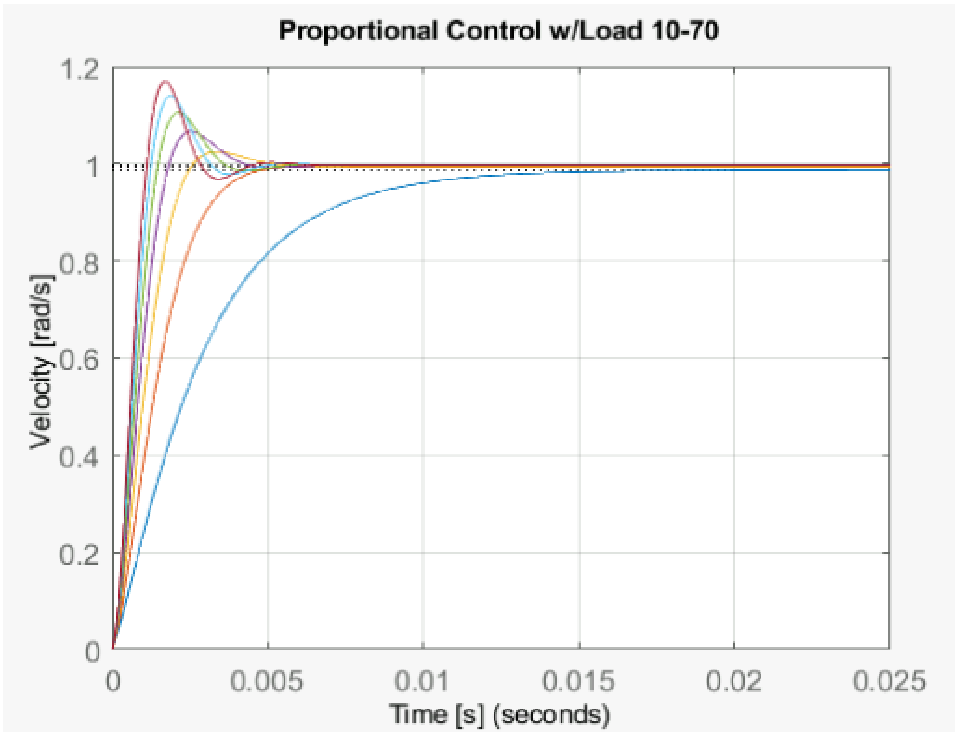


Figure 9: Proportional Speed Control Step Response

We then found the step response for the position control based on the determined gain parameters from the root locus analysis and equations 8 through 10, the results of which are shown in Figures 10 through 12 below.

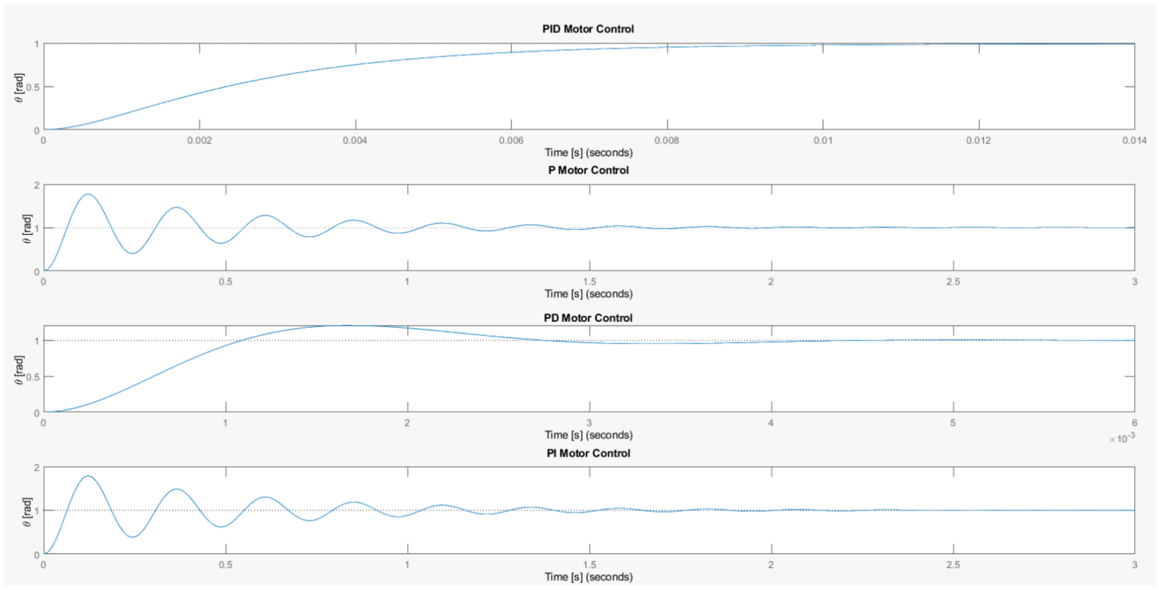


Figure 10: Proportional, Integral, Derivative, PID Position Control Step Response

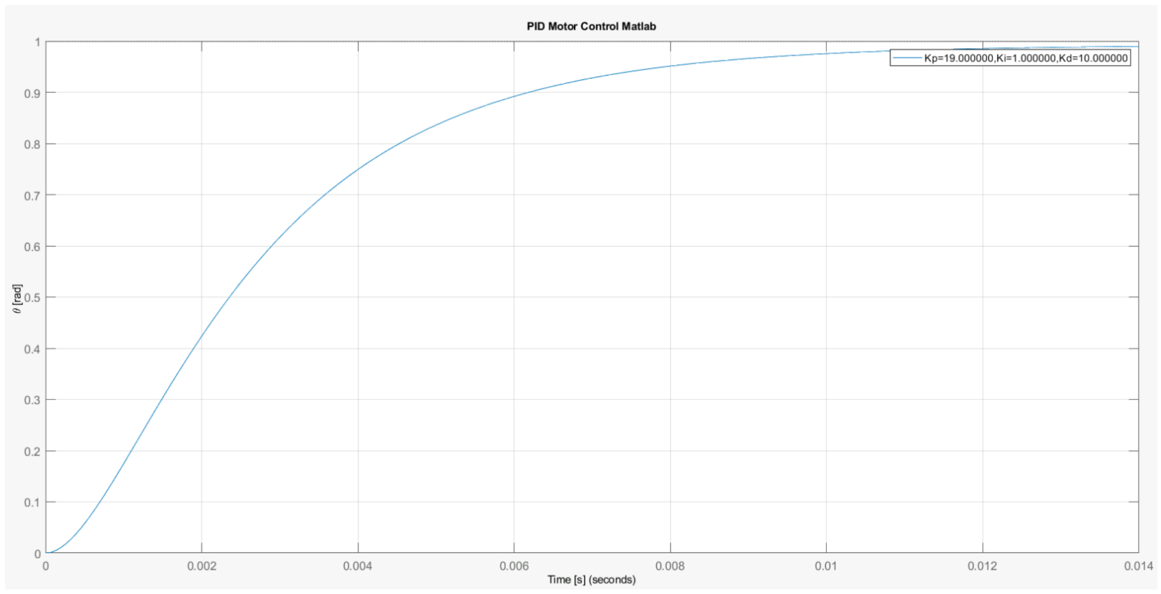


Figure 11: Position Control PID Step Response

From these results we can see that the PID controller has the best percentage overshoot and settling time.

The results of the transient analysis for speed control of various controllers with gain parameters based on the root locus analysis are shown in table 1 below.

Table 1: Performance of various controllers for speed control

Control method	*Gains (Kp, Ki, Kd)*	*Percentage Overshoot*	*Settling Time (s)*
P	*20, 0, 0*	*0.9659*	*0.0036*
PI	*20,10,0*	*0.3669*	*0.0037*
PD	*20,0,10*	*0.2663*	*6.0218e-06*
PID	*20,10,10*	*0.0000*	*6.5435e-06*

As with position control, these results show that a PID controller has the best percentage overshoot and settling time.

Evaluation of Nonlinearities

The system nonlinearities are the variation in load due to the truss structure at different angles during rotation and the power drive saturation. The power drive saturation occurs when the value from the controller exceeds the maximum power of the power supply so the duty cycle becomes clipped at 100% duty cycle for the duration of that time. Because the maximum voltage from the power supply is 12V and the PID controller adjusts the duty cycle between 0-100% duty cycle the average voltage to the motor can be 0-12V. The PID controller with proportional and integral gains being multiplied by the error signal is the control signal. This control signal should be between -12 and +12 which controls the average voltage sent to the H-Bridge using PWM and the direction of that voltage. In figure 12 below, the yellow line is the position vs time and the green line is the Voltage. Until the position reaches close to 45 degrees, the 100% duty cycle is used due to the nature of this PID controller. Once it reaches 45 degrees, the voltage averages to zero with some fluctuations to maintain the position setpoint.

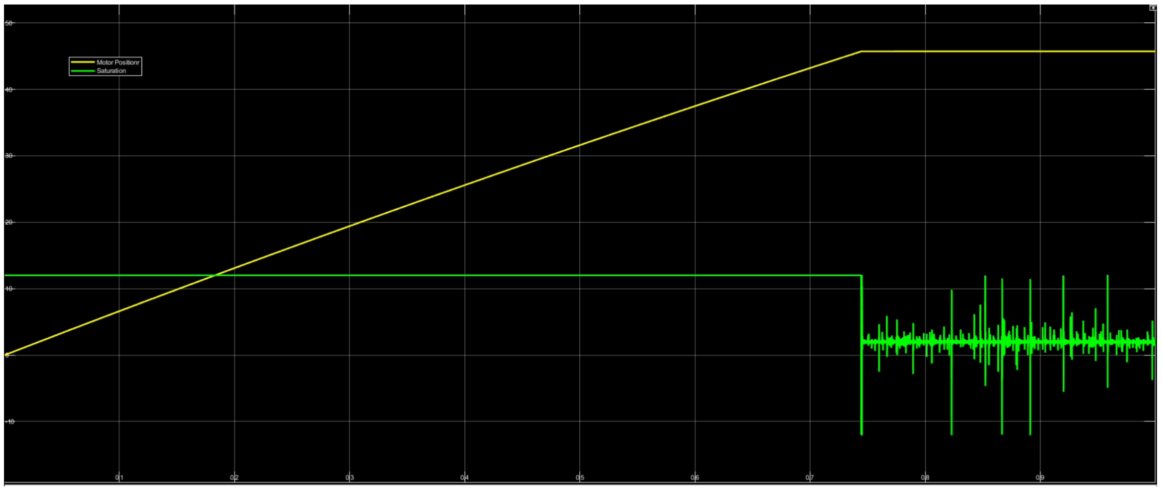


Figure 12: Graph of Position Control with Voltage to Motor

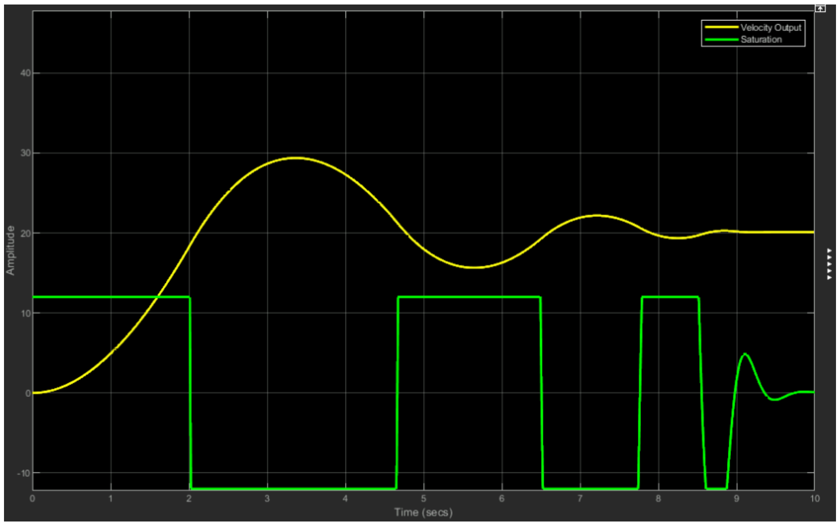


Figure 13: Graph of Speed Control with Voltage to Motor, P=100, D=10

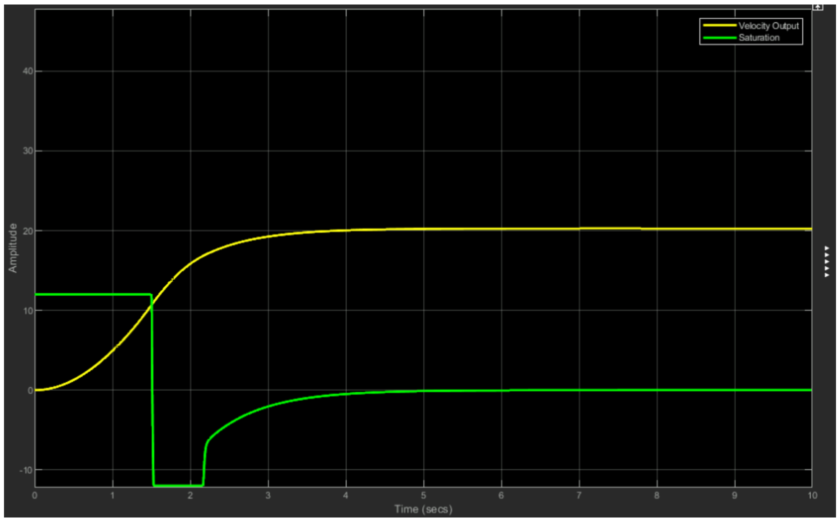


Figure 14: Graph of Speed Control with Voltage to Motor, P=100, D=70

Non-linear load on the shaft is a function of angle. As the angle of the arm changes, the moment on the shaft due to the weight of gravity on the truss arm is shown by the following equation.

$$\tau - N_b - B \times \widehat{\theta} - m_b \times g(r \times sin(\theta) = J \times \overline{\theta} \qquad (11)$$

V. Integration

The entire system consists of a DC motor, nand gate for the digital logic, H-bridge, as well as an Arduino board. The Arduino is used to interface Simulink with the breadboard, where the circuit is designed. The nand gate itself consists of 4 different nand gates but only three of which we had to use in our final circuit design. The nand gate is shown below. The top rightmost nand gate with inputs B3 and A3 were not used in the circuit design.

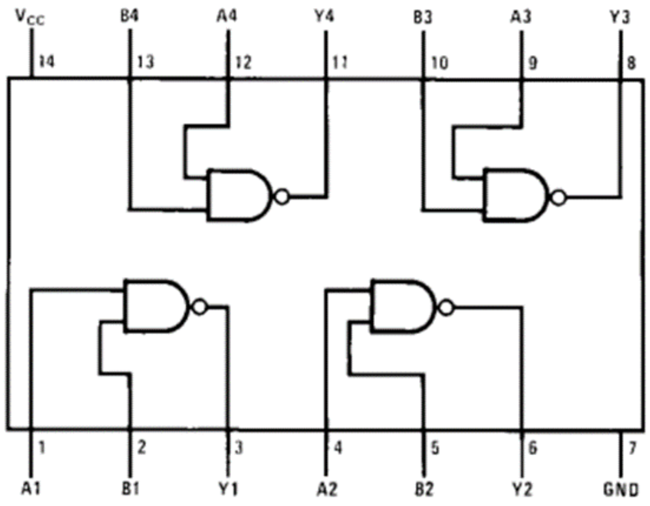


Figure 15: Quad 2-input NAND gates

The PWM signal is connected to pin 1 of the nand gate, which is connected to pin D9 of the Arduino. Pin 2 is the L/H pin, which is connected to D12 of the Arduino. Pin 3 is an output of the nand gate, which is connected to pin 1 of the H-bridge. The outputs of the nand gate were connected to the inputs of the H-bridge. So the power flows from the Arduino to the nand gate to the H-bridge. The power supply powers the motor with 12 Volts and the motor power wires are connected to pins 3 and 5 of the H-bridge. Pin 7 of the H-bridge is connected to the positive terminal of the breadboard.

The encoder wires of the motor are connected to Arduino pins 18 and 19 for channels A and B respectively. The encoder data is fed into Simulink to determine the angular position and velocity. Two PID controllers are also implemented in Simulink for angular position and speed to get the motor shaft to rotate a particular amount of degrees at a particular angular speed. Figure 20 below shows the interconnection of various components of our system.

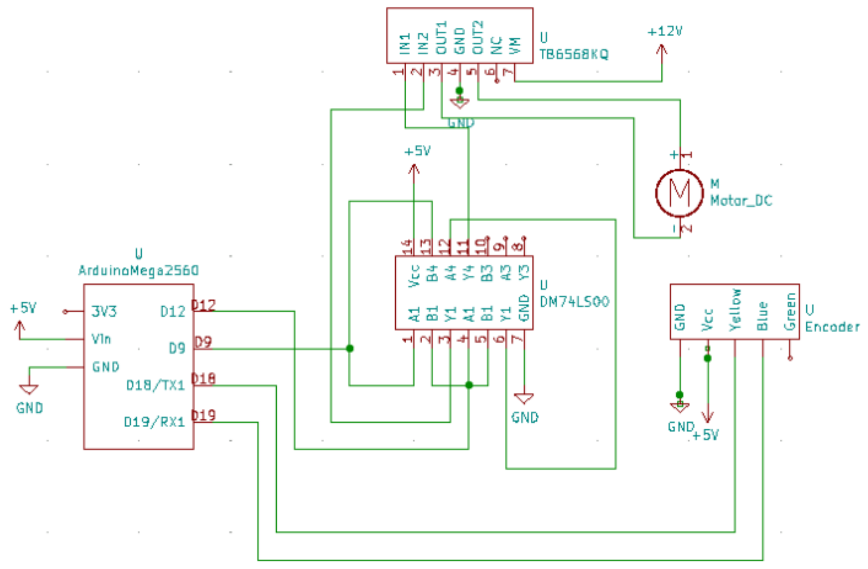


Figure 16: Systems Interconnection

Having created the electrical and control design, we set up our system by attaching our launch arm to the provided station, and connecting the supplied motor and encoder at the station to our Arduino and electronic circuit which are controlled by the simulink program.

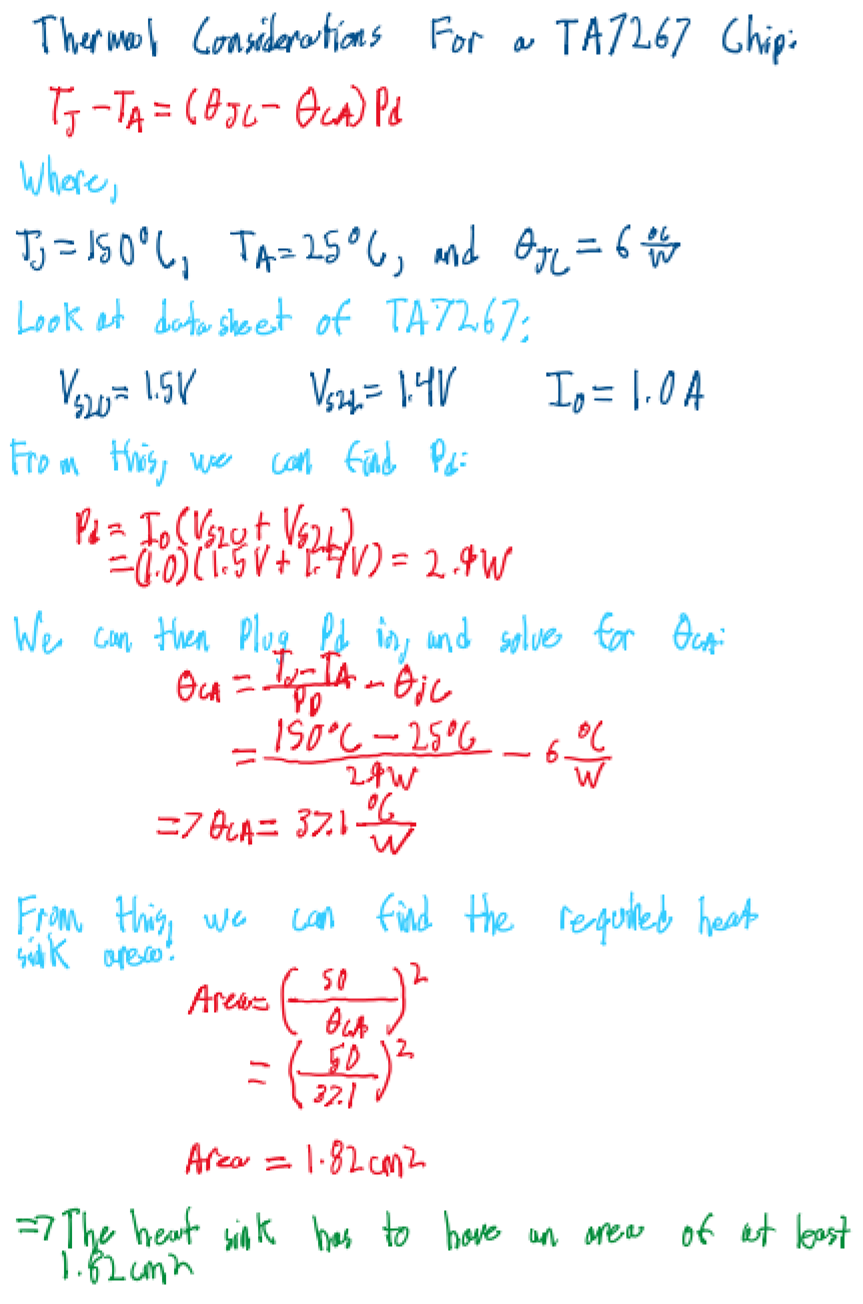
Due to the issue with the power supply inductance and its effect on the speed control, the priority of the position control, and the likelihood of other factors that could significantly affect the speed control, we first tested the position control on our physical system and adjusted the PID parameters as necessary to minimize the overshoot. Once the overshoot had been minimized to 1 degree, we moved onto adjusting the speed control PID parameters. To do this, we started with a test to simply launch the ball. In this test, the angular velocity was set to 1000 degrees per second, and the desired launch angle was set to 170 degrees. The test was conducted by recording results and slowly adjusting the PID values in order to create successful launches. The most recent 20 throw results were considered and adjustments to PID values continued until the ball was capable of being launched in 18 out of 20 throws. The position overshoot was found to be 5 degrees or greater for 7 of these trials and so we adjusted the position control PID parameters slightly and repeated the throwing test. The results of these throwing tests are shown in tables 2 and 3 in Appendix B. In the process of the PID tuning for the position control, we had to switch to a different station and noted that the results were slightly different even though the PID parameters had not been changed. It is suspected that potential differences in wear and tear on the motors in different stations and other differences between stations can cause variance in our system's performance. This is something that will be kept in mind for the final demonstration and a plan will be made to mitigate the impact of this.

After the throwing tests, we moved onto accuracy testing. The purpose of this test was to set the angular velocity to a predetermined value, calculate the distance that such a launch would result in, and launch the ball and compare the error results after 10 attempts. At this point PID values would be adjusted and 10 more launch attempts would be repeated. However, this test was unable to be conducted. While the ball throwing tests had been completed with no issues on July 22, 2022, when we went in to run the accuracy test and evaluate the performance of our system further, we found that our system was behaving erratically. After much troubleshooting and swapping out components, it was determined that the system was for some reason only reading either a 0 or a 1 from the encoder, rather than reading the full range of encoder values. We attempted to resolve this issue with the help of Osama and Dr. Palmer but were unable to do so, which prevented us from running further tests on our system. The intent was to complete the accuracy test, and then use scopes and other tools to obtain graphs and other data of our system for analysis. Going forward, the goals are to attempt to resolve this issue with the encoder and then complete the remaining system analysis.

VI. Conclusion

Throughout the entire past six weeks, our group has worked tirelessly to create the best integrated electrical and control system to complete the task of throwing a ball accurately into a cup at a given distance. To help with our design process, we made use of calculations done both by hand as well as Simulink and MATLAB. Our design succeeded in every experimental test. We also gained practical experience in applying transfer function models in MATLAB/Simulink to accurately model and simulate the response of the motor. By using a feedback controller we were able to produce a response with the desired characteristics of low overshoot, settling time, and stability for the control applications with a load. The most crucial section was the derivation of the transfer function because, with it, we proved that we can simulate any physical system by applying the physics and applying the transfer function in the control system.

Appendix A



Thermal Considerations For a TA7267 Chip:

$T_J - T_A = (\theta_{JC} - \theta_{CA}) P_d$

Where,

$T_J = 150°C$, $T_A = 25°C$, and $\theta_{JC} = 6 \frac{°C}{W}$

Look at datasheet of TA7267:

$V_{S2U} = 1.5V$ $V_{S2L} = 1.4V$ $I_0 = 1.0A$

From this, we can find Pd:

$$P_d = I_0(V_{S2U} + V_{S2L}) = (1.0)(1.5V + 1.4V) = 2.9W$$

We can then Plug Pd in, and solve for θ_{CA}:

$$\theta_{CA} = \frac{T_J - T_A}{P_D} - \theta_{JC} = \frac{150°C - 25°C}{2.9W} - 6\frac{°C}{W}$$

$$\Rightarrow \theta_{CA} = 37.1 \frac{°C}{W}$$

From this, we can find the required heat sink area:

$$\text{Area} = \left(\frac{50}{\theta_{CA}}\right)^2 = \left(\frac{50}{37.1}\right)^2$$

$$\text{Area} = 1.82 cm^2$$

$\Rightarrow$ The heat sink has to have an area of at least 1.82 cm²

Figure 17: Heat Sink Calculations

Appendix B

Table 2: First Throwing Test Results

Attempt	Ball Launch Status	Position Overshoot (Degrees)
1	Launched	1
2	Failed	180
3	Launched	3
4	Launched	3
5	Launched	4
6	Launched	3
7	Launched	5
8	Launched	5
9	Failed	2
10	Launched	0
11	Launched	7
12	Launched	5
13	Launched	3
14	Launched	3
15	Launched	5
16	Launched	4
17	Launched	3
18	Launched	5
19	Launched	2
20	Launched	4

Table 3: Second Throwing Test Results

Attempt	Ball Launch Status	Position Overshoot (Degrees)
1	Launched	3
2	Launched	4
3	Launched	5
4	Launched	2
5	Launched	3
6	Launched	2
7	Launched	0
8	Launched	-3
9	Failed	4
10	Launched	6
11	Launched	3
12	Launched	1
13	Launched	2
14	Launched	4
15	Launched	4
16	Launched	4
17	Launched	3
18	Launched	4
19	Launched	3
20	Launched	1