MSE 312: Mechatronics Design II

Summer 2021

Electronic Drive Circuit and Control System Design for a Throwing Robotic Arm:

Student Name	Student ID
Sepehr Rezvani	301291960
[redacted]	
[redacted]	
[redacted]	

Prepared by: Long March

Submission Date: July 25th, 2021

Table of Contents

Introduction

In this project we develop controllers for a Throwing Robotic Arm, which throws a ball from a reference point (zero position) to a target point. The target point is in three different ranges, and the ball must land withing 2cm of the specified location.

In Mechanical portion of this course, Long March developed three mechanical designs in Solidworks, and integrated them in Simulink for simulation. Once mechanical part the project was completed, Long March has been focused on designing power electronics drive circuitry for our electronic system. Additionally, simulations at different levels have been completed as well in order to validate quality of our electronic system.

While some members have been working on electronics design, other have been working on the control portion of the project as well. The DC motor (GM8724S009), H-Bridge (TB6568KQ) and other electronics components power the mechanical side of the design. However, control circuitry is as important as the power systems since the motion of the Throwing Arm must be fully controlled. Otherwise, the ball cannot successfully leave the arm.

In this report, Long March will demonstrate how to design, test, and validate electronics and control components of this course. Readers can follow report's instruction to simulate a similar project in Matlab/Simulink.

Part A: Electronics

Electronics Introduction

In this section the design and power interface circuitry for driving a DC motor carrying an inertial load, in this application our throwing arm. In a real system, a power electronics interface is necessary since the control/command signals from the microcontroller cannot provide enough power to drive the motor and the electromagnet.

Motor Selection

Seeing as the desired system is to be electro-mechanical, a suitable motor is to be selected to propel our ball to its destination. The following criteria will be used to select our motor.

- Must be a DC Brushed motor capable of forward and reverse actuation
- Must be within the 0-12VDC operating range to minimize power
- Must be capable of generating at least 6.004oz*in of torque as calculated to be the minimum torque to drive our mechanical system.
- Must have a gearbox to reduce strain on motor and increase torque
- Gearbox must be of at least 90% efficiency to minimize power loss.
- Must be affordable to be within project budget.

Figure 1: GM8724S009 DC Gearmotor [1]

The following motor was selected for our system. The GM8724S009 DC servo gear motor is a DC motor made with Copper-Graphite brushes that has a nominal voltage of 12VDC. It's gear ratio of 6.3 and stainless-steel shaft will reduce strain on the motor and provide up to 42 oz*in of torque to our system [1].

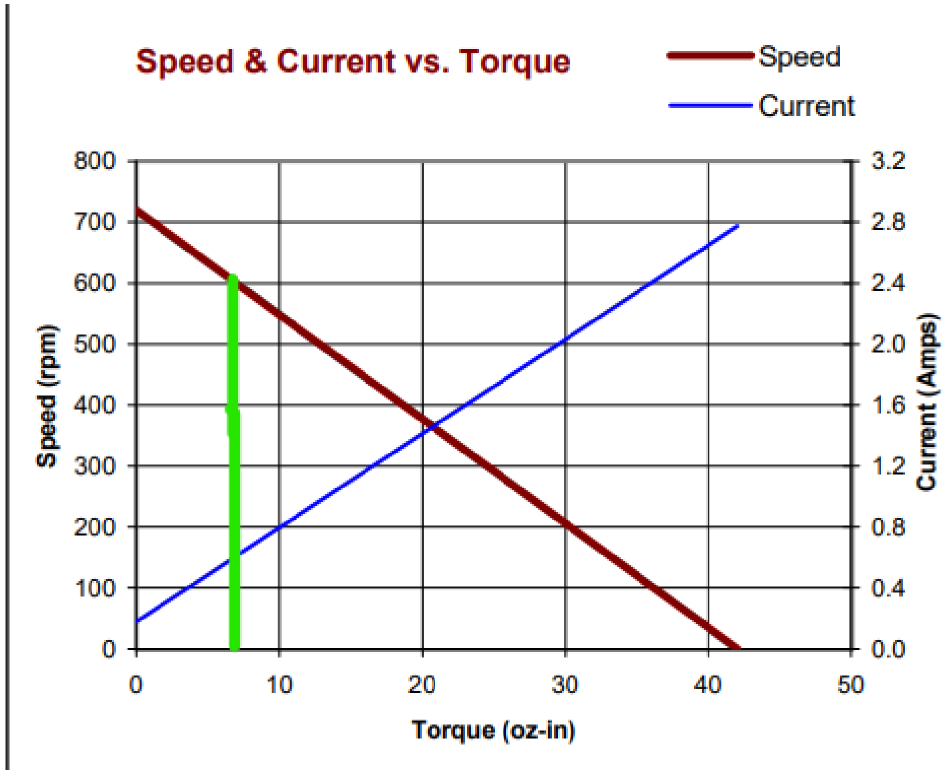


Figure 2: GM8724S009 Torque Speed diagram

In the figure above the vertical green line represents the minimum torque needed to propel the ball in our system. It is clear it is well within the operating range of our system and still provides a lot of operating power to propel the ball to the desired positions.

Design Stages of Power Electronics Drive Circuitry

DC Motor Control

Theory

A simple DC motor uses a stationary set of magnets in the stator, and a coil of wire with a current running through it to generate an electromagnetic field aligned with the center of the coil. One or more windings of insulated wire are wrapped around the core of the motor to concentrate the magnetic field.

The windings of insulated wire are connected to a commutator (a rotary electrical switch), that applies an electrical current to the windings. The commutator allows each armature coil to be energized in turn, creating a steady rotating force (known as torque) [2].

By nature, applying a positive voltage potential across a motor will rotate the motor forward and applying a negative voltage potential across the motor will rotate the motor in reverse. Thus, it is imperative to design a controller that can convert simple control signals into the desired direction control.

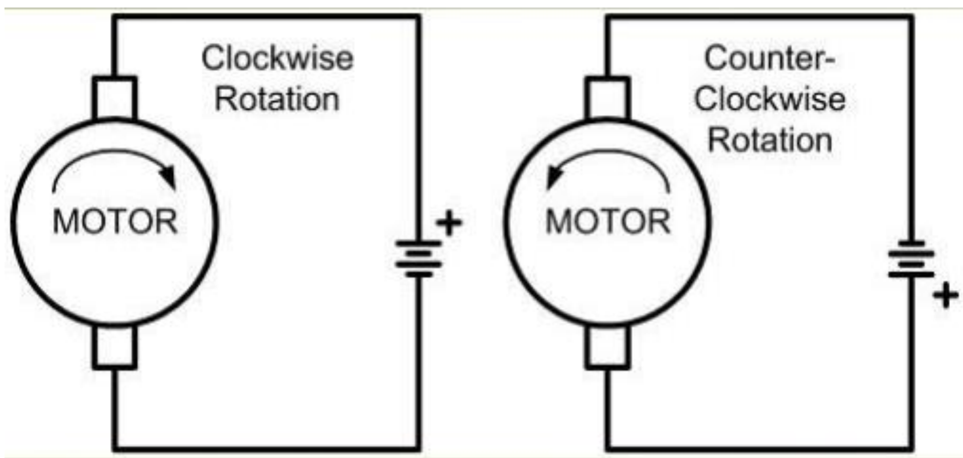


Figure 3: Direction Control of a DC motor

The proposed electrical circuit is a H-Bridge circuit. In general, an H-bridge is a rather simple circuit, containing four switching elements, with the load at the center, in an H-like configuration:

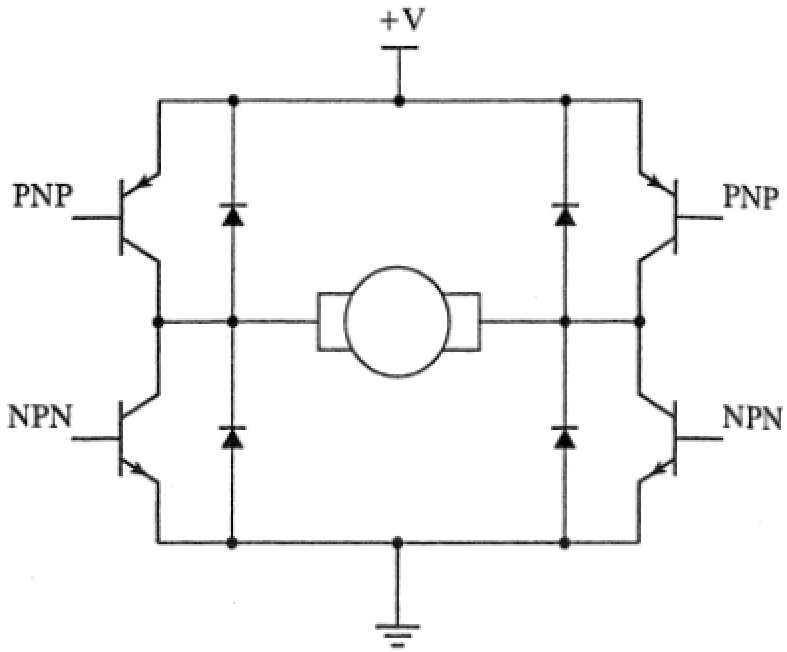


Figure 4: H-Bridge Circuit Topology

The switching elements (Q1...Q4) are usually bi-polar or FET transistors, for our application we are using MOSFETs. The diodes (D1...D4) are called kick-back or snubber diodes. The top-end of the bridge is connected to a power supply (battery for example) and the bottom-end is grounded. The operation table can be seen below according to Figure 5.

Table 1: H-Bridge Sequence

Motor Operation	S1	S2	S3	S4
Off	Open	Open	Open	Open
Clockwise	Closed	Open	Open	Closed
Counter-clockwise	Open	Closed	Closed	Open
Invalid	Closed	Closed	Closed	Closed

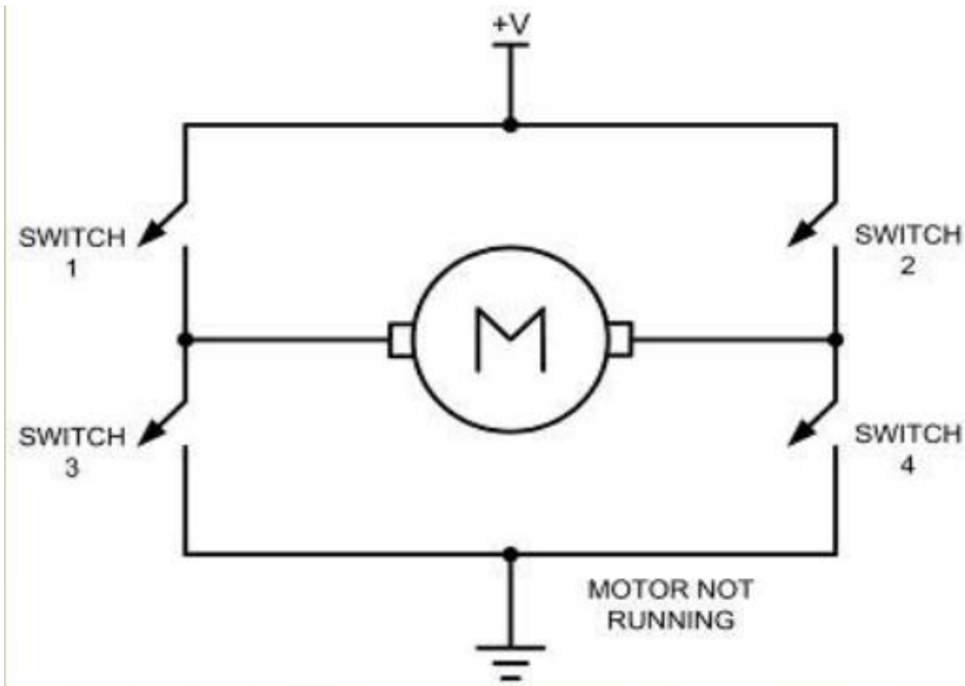


Figure 5: H-Bridge Circuit

The speed can be controlled using PWM. Changing (modulating) the width of the pulse applied to the motor can increase/decrease the amount of power provided increasing/decreasing the motor speed.

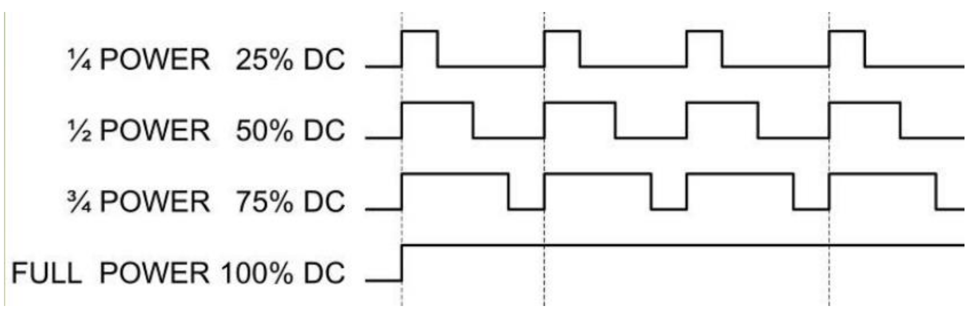


Figure 6: PWM Diagram

Implementation

A Toshiba TB6568KQ Full-Bridge DC Motor Driver IC was selected to be our H-Bridge device. The basic electrical implementation in Simulink can be shown below. This model shows how to use the Controlled PWM Voltage and H-Bridge blocks to control a motor. The DC Motor block uses manufacturer datasheet parameters found on our H-Bridge and motor datasheets respectively.

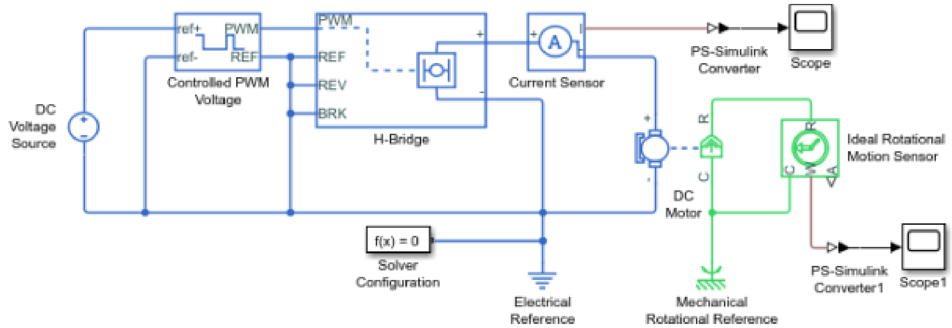


Figure 7: Electrical Control Simulink Circuit

To implement direction control, the H-Bridge block by default runs in the forward direction and if the voltage applied at the 'REV' port is larger than a pre-defined voltage, the motor will run in reverse. Thus, if the voltage is greater than zero volts, 5V is applied to the 'REV' port and the motor is ran CW, if it is less than 5V, 0V is applied to the 'REV' port and the motor is turned CCW.

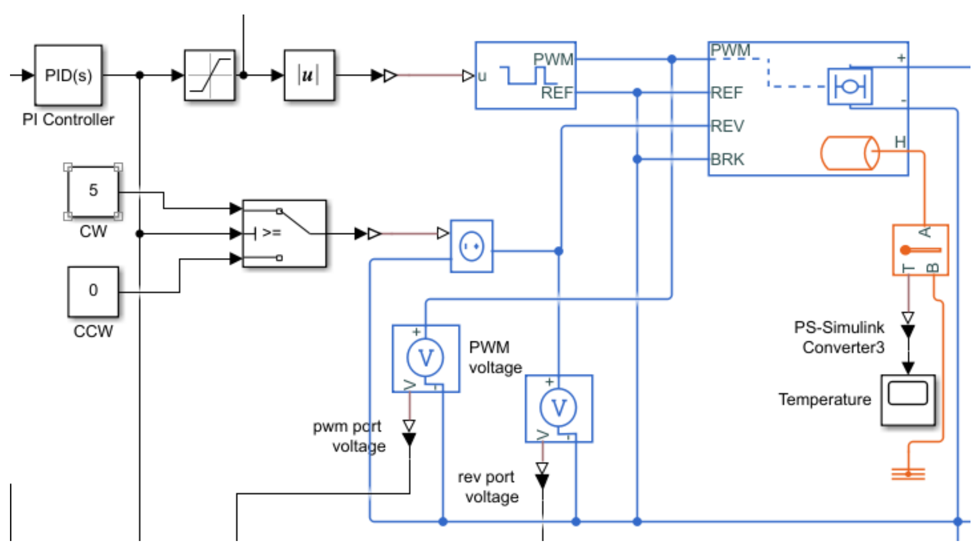


Figure 8: Simulink Circuit of Direction Control

The simulation was populated with the parameters obtained from the respective datasheet. This can be found in Appendix: Part A. The simulation results can be found in the sections below.

Thermal Design for PWM Chip

Using the properties from the TB6568KQ datasheet [4] the selection criteria for a heatsink can be obtained. The temperature difference between the junction and the ambient depends on the power dissipation (P_D) and the thermal resistance between them as shown in the equation below. The P_D was extracted from the datasheet to be 7.80W. Seeing as the power dissipated was less than 10W, a passive heatsink was selected.

$$T_{Chip} - T_{Ambient} = \theta_{ChipAmbient} P_D$$

The thermal resistance between the Chip and the Ambience can be broken down into a pair of series resistances as shown below.

$$\theta_{ChipAmbient} = \theta_{ChipCase} + \theta_{CaseAmbient}$$

Combining the two equations listed above and rearranging to solve for the unknown thermal resistance of the Chip to the Case can be shown as follows where $\theta_{ChipCase}$ was extracted from the datasheet to be 6°C/W. T_{Chip} is a known value for silicon devices to be 150°C.

$$\theta_{CaseAmbient} = \frac{T_{Chip} - T_{Ambience}}{P_D} - \theta_{ChipCase} = \mathbf{10.026°C/W}$$

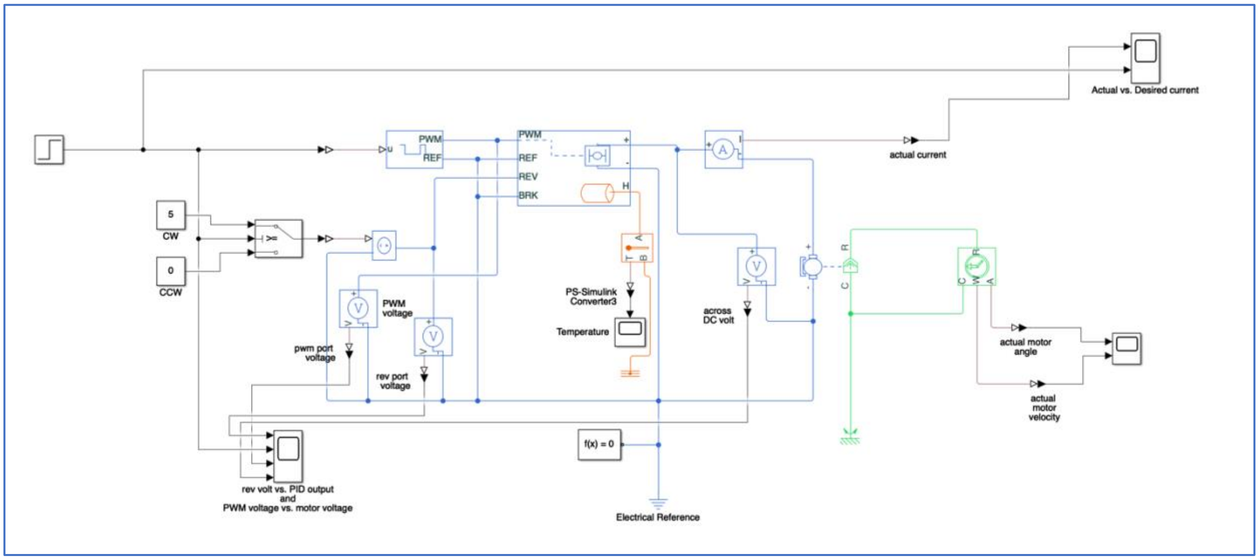
A rough formula for designing a passive heat sink can be shown below where the area units are in cm^2 and the thermal resistance is measured in °C/W.

$$A = (\frac{50}{\theta})^2 = \mathbf{24.87}\boldsymbol{cm^2} * \mathbf{1.5} = \mathbf{37.5}\boldsymbol{cm^2}$$

Applying a 50% safety factory yields an aluminum heat sink area of **37.5cm²**.

Simulation Results for Evaluation Power Drive Circuit Operation

Long March has implemented lab 1 design in the project. Many factors such as nonlinearities and uncertainties from the electrical section can affect the performance of the system. This section explains how different parameters (e.g. motor or H-bridge values) can change simulation results. For the electrical simulation, Long March developed the following design to test electrical components. Shown in the figure below.



Note that this design was modified to integrate into the model, and a PID block was added to implement PID for the current control of the control project. At the center of this design, we have a DC motor and H-Bridge that controls the direction of current in the motor. Values are based on the motor and bridge selected for this project that is with the specification values included in Appendix A. Controlled PWM block creates the Pulse-Width Modulation voltage signal across its PWM and REF ports, which are connected to PWM and REF ports of the H-bridge, respectively.

In this circuit, a step input was used for testing usage to create a predictable desired current. The purpose of this specific structure was to help us with signal tuning for the controller project. Further details will be discussed in the portions of the control of the report. Table 4 in Appendix A shows settings for the Controlled PWM voltage. Note that Output Voltage Amplitude must match PWM Voltage Amplitude in the H-bridge. Voltage and current sensors connect to different scopes to help us understand how values change in different parts of the circuit. For example, figure 10 compares voltages at 4 different stages of the circuit, which are the following: the h-bridge's REV port voltage, the desired input current to the system, the h-bridge's PWM port voltage, and the voltage across the DC motor. These scopes were not required for lab 1, however, they helped with the analysis of errors

in the integration of our system.

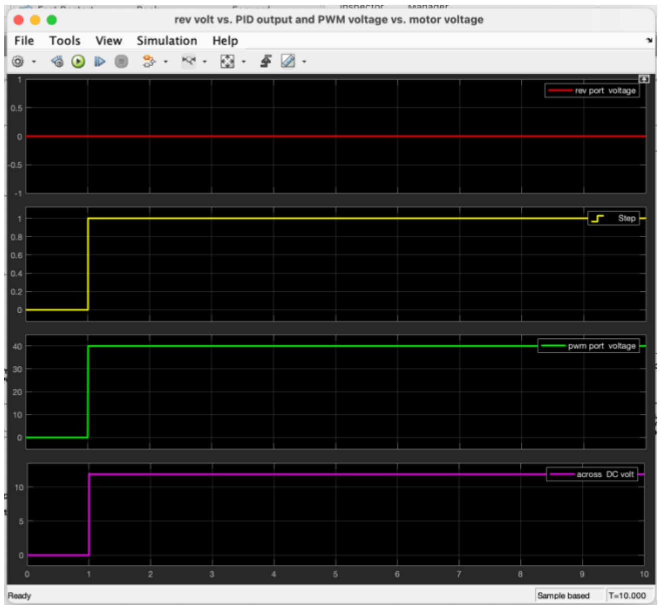


Figure 10 – "rev volt vs. PID output AND PWM voltage vs. motor voltage"

"rev port voltage", is directly affected by the value obtained from desired input current, and condition of the switch. For integration portion of the project, the switch was used to change voltage value entering REV port in the h-bridge to change polarity of the output voltage. This is how direction of rotation of DC motor is controlled. The following figure shows the input (step) current vs. actual current. There is a slight overshoot at t=1, and the actual current drops down to about 1.4A. This error can be fixed by implementing proper PID values, which will be done in part B.

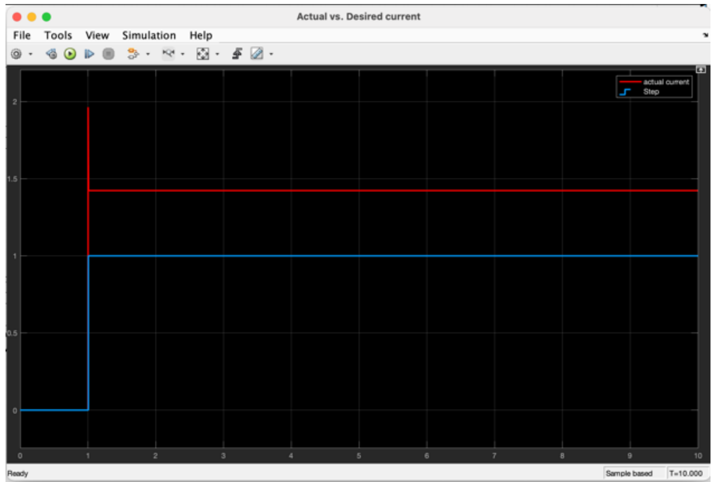


Figure 11 – Actual vs. Desired Current of the Power Drive Circuit

Another important part of the power drive circuit is position and velocity of the motor, and they are included in the figure 12. Note that DC motor “Model Parameterization” must be set to “By stall torque and no-load speed”, since the mechanical portion is not connected yet.

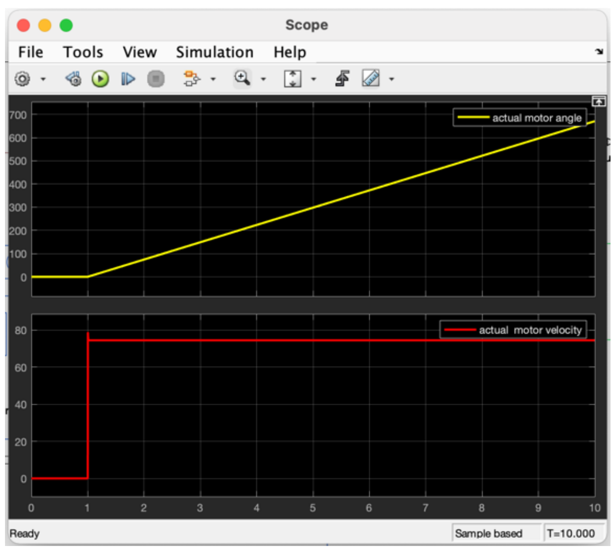


Figure 12 – DC Motor Position and Velocity

Electrical Conclusions

In conclusion the theory behind the operation control of a DC motor was introduced and relates to the context of the throwing a ball. The selection criteria for our desired motor were discussed and a motor was chosen accordingly. DC Motor direction control was introduced and applied to our electrical system in the form of PWM controlled H-Bridge circuit. A switch in the circuit was used to implement the reversal of direction feature on the DC motor allows the throwing arm to both accelerate in the positive and negative direction to first accelerate the ball, then release the ball and return the arm to its resting position. To maintain suitable operating temperature, a heatsink was introduced and implemented into our system based on values extracted from the datasheet. This increases the heat transfer coefficient of the system. The passive heatsink was designed to be adequate and sized accordingly. Once satisfactory results were obtained, individual parameters (rotor inertia, damping, PWM frequency, and Vm) were adjusted to view the effects on the electrical system. Each parameter was discussed thoroughly.

Part B: Control

Control Introduction

The main objective of the control system is to create a position, velocity and current control of the motor such that the ball follows trajectory. For this purpose, PI controllers are considered by getting feedback from the actual values and comparing them to the desired values.

Control Design Criteria

The performance of the PI controllers is evaluated by checking the quantities such as error overshoot and settling time. The settling time is referred to as the time it takes for the output response to reach acceptable final values from the time the simulation starts. Respectively the error is defined as the steady state value which would be the difference between the desired value and the value attained in the experiment. The simulation file has been extracting both the desired and the actual value under the same scope to clearly compare and observe the difference between both values.

Each variable ”P” or “I” would modify certain characteristics of the scope when the value of P=1 is assigned it would entail that no effect on the scope is made and as the value of “P” decreases it would entail and actual result with a greater error value therefore it would be considered that a tune “P” value would lead to an outcome with a smaller steady stater error. As we increase

“P” the there would have greater overshoot before reaching steady state in the system. Further tuning of the system can be achieved with the integral part of the controller

The tuning of the variable “I” the similarly increase the overshoot value of the experimental result but helps decrease any offset from the desired steady state output, I is gradually increased it is suitably close to the step input response. Because in this portion of the project simulation of the “Derivative” is not the focus, an overshoot within the system is inevitable

In our experiment for this project both speed and current desired and actual values are compared. Settling time is referred to the time it takes for the actual experimental value to settle in the standardized range of +-5% [5] of the steady state value

The PI controller parameters are tuned manually for current and speed by observing the actual values of current and speed and comparing them to the desired values. The PID values are colored in yellow in the following shot for your reference

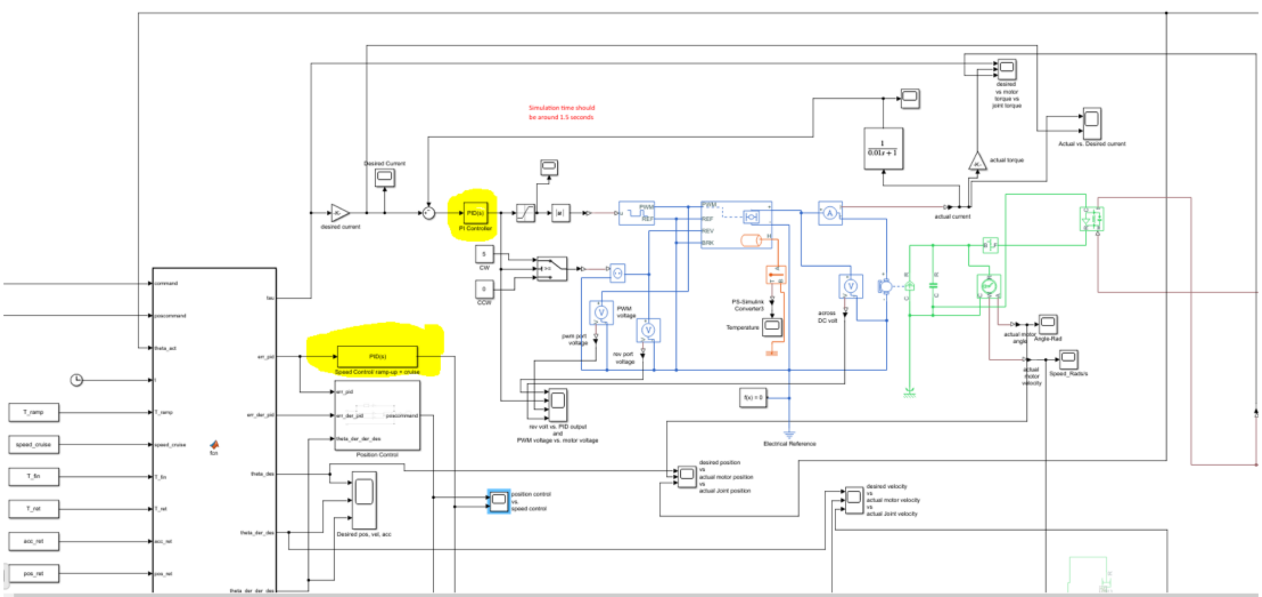


Figure 13 – Highlighted Portion of the PID controllers from the whole system

Design Stages of Feedback Motion Control System

Plant Transfer Function and Stability

In order to design a controller for our electro-mechanical system a transfer function must be obtained using first principles.

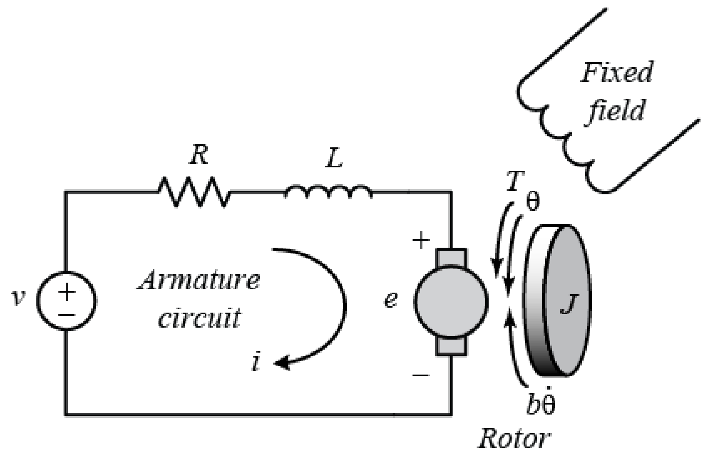


Figure 14: Electromechanical Diagram of a DC Motor

Beginning with a KVL loop performed on the electrical side of the electro-mechanical diagram we can obtain the following equation where K_m is the motor constant, n represents the gear ratio, 'ω' is the angular velocity of the motor shaft.

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

Where:

$$V_{emf} = K_m n\omega$$

Now, on the mechanical side of the diagram we can examine the system from first principles, using Newton's 2nd law.

$$\Sigma\tau = J\alpha$$

The summation of torque can be broken down into the torque produced by the DC motor subtract the friction torque as shown below. Note that η is equal to the efficiency of the gearbox, 'n' is equal to the gear ratio, 'i' is equal to the current passing through the DC motor, B is equal to the viscous friction, τ_s is the static friction and J is the rotational inertia of the system.

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

Obtaining the Laplace transform of KVL equation and the equation above yield the following.

$$V(s) = LsI(s) + RI(s) + K_m n s\theta(s)$$

$$Js^2\theta(s) = \eta n K_m I(s) - Bs\theta(s) - \tau_s(s)$$

Knowing that the input into our system will be a voltage source and the output will be the angular position of the motor shaft we can develop a state diagram.

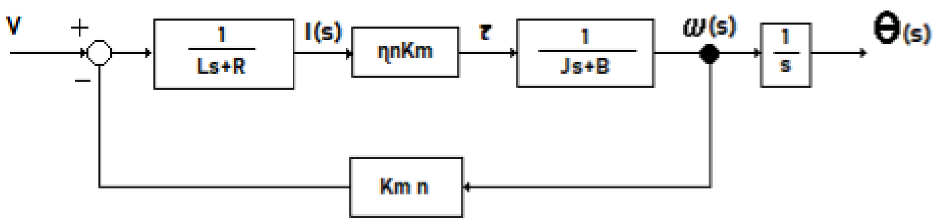


Figure 15: State Diagram of a Geared DC Motor

Where;

$$\theta(s) = \frac{\eta n K_m I(s) - \tau_s}{Ls + R}$$

Using the equation and state diagram above and if $\tau_s = 0$ we can obtain the transfer function of the plant.

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

Now that the transfer function has been obtained, it is worthwhile viewing the stability of the plant through the root locus poles and zeros. Solving the characteristic equation, above yields the following poles.

Table 2: Poles of Characteristic Equations

Pole 1	0
Pole 2	-0.067
Pole 3	$-1.85\text{x}10^3$

Therefore, due to the location of the poles it can be shown that the system is stable under certain conditions. It is noteworthy that Pole 3 is approximately –R/L which is also known as the inverse of the electrical time constant. Pole 2 is within the order of B/J which is the inverse of the mechanical time constant. Thus, the feedback system can be viewed within two-time systems. The fast time system (electrical) and the slow time system (mechanical).

Using this understanding we can develop a cascade controller to implement current and position feedback on our DC motor system.

Current Controller

Theory

Observing the transfer function of the current and voltage system we can find the transfer function.

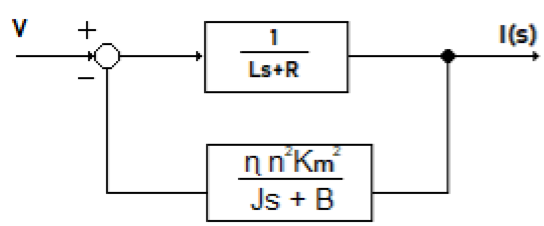


Figure 16: Transfer function for the Current and Voltage system

Which yields for the following transfer function:

$$\frac{I(s)}{V(s)} = \frac{J(s+B/J)}{(Ls+R)(Js+B)+\eta n^2 K_m^{\;2}}$$

Seeing as a pole and zero are in the same spot it is sufficient to say that they can be removed from the system which yields the remaining transfer function.

$$\frac{I(s)}{V(s)} = \frac{1/L}{s + R/L}$$

Thus, the only pole is located at –1850. The system can be determined to be stable.

Integration

To tune our current controller a Simulink file was generated as shown below. A step input of 1 was applied and PI values were modified until a satisfactory system was created.

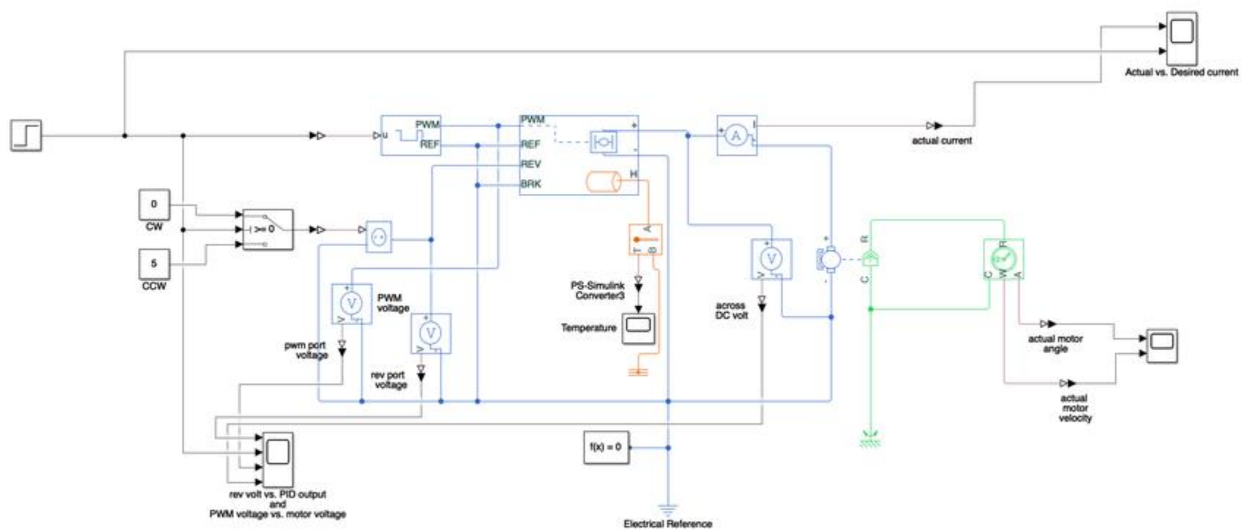


Figure 17: Current Controller Simulink File

In our first iteration PI values were selected arbitrarily as P=0.7 and the I = 1 and the results can be found below. The settling time can be shown as 122.965ms. The percentage overshoot can be observed to be approximately 15.

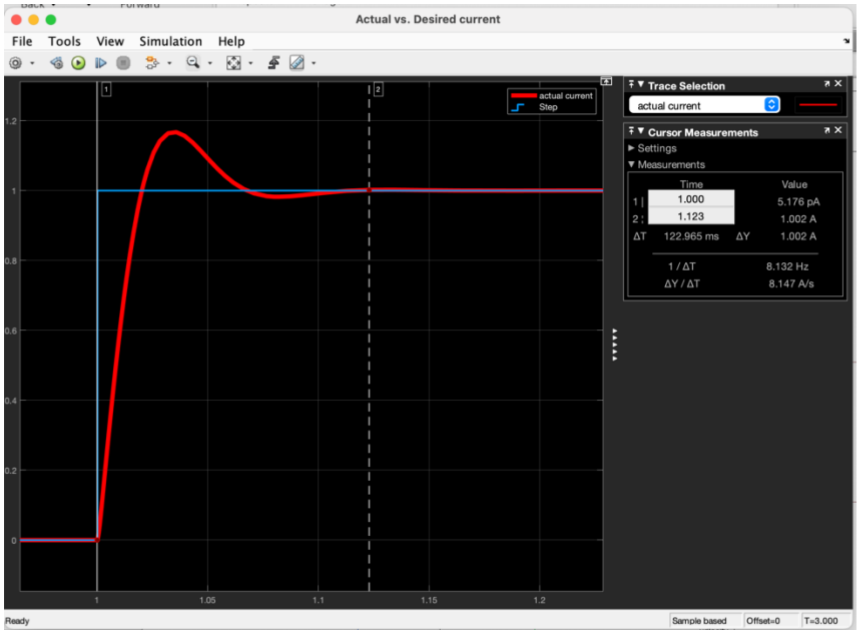


Figure 18: PID values of 0.7/1/0 respectively

To reduce the overshoot and improve the settling time, a much lower P value was selected and a much higher I value was selected. The results can be shown below.

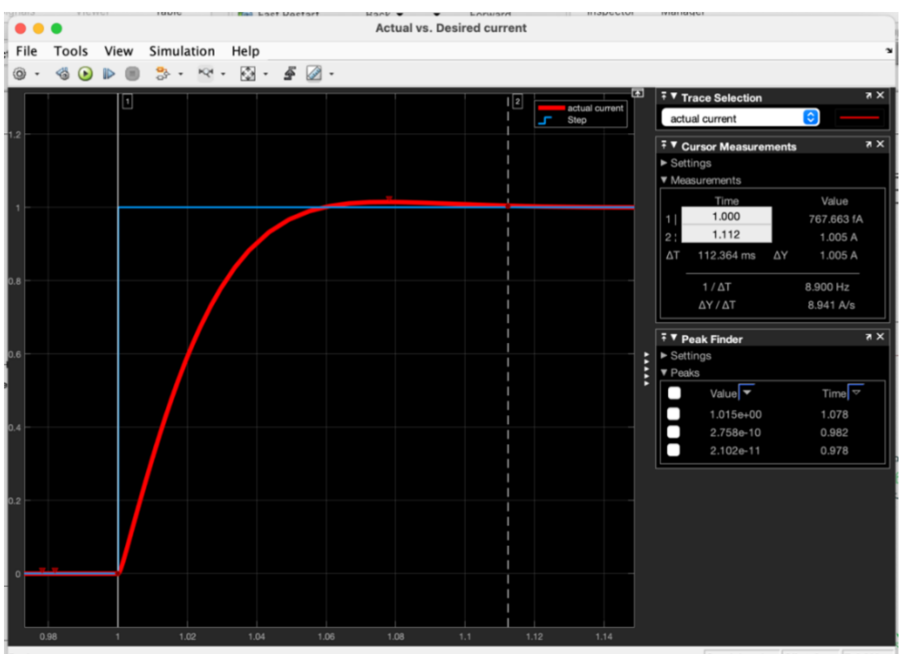


Figure 19: PID values of 0.0001/10.5/0 respectively

It is clear this system is drastically improved as the settling time has been reduced to 112ms and the overshoot percentage was reduced to 1.5%. These are the values we will proceed with in our cascade controller.

Speed Controller

Background

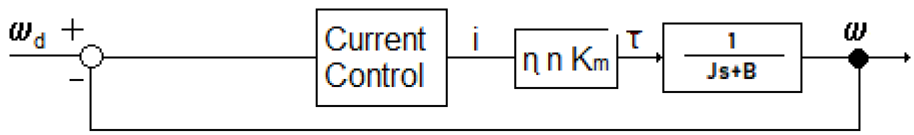


Figure 20: Transfer Function of the with the implemented cascade

After recognizing the relationship between current and torque and torque and speed it is evident that the cascade motor controller can be implemented to yield a transfer function in figure 20.

Implementation

The Simulink topology can be found below that was used to tune the speed controller.

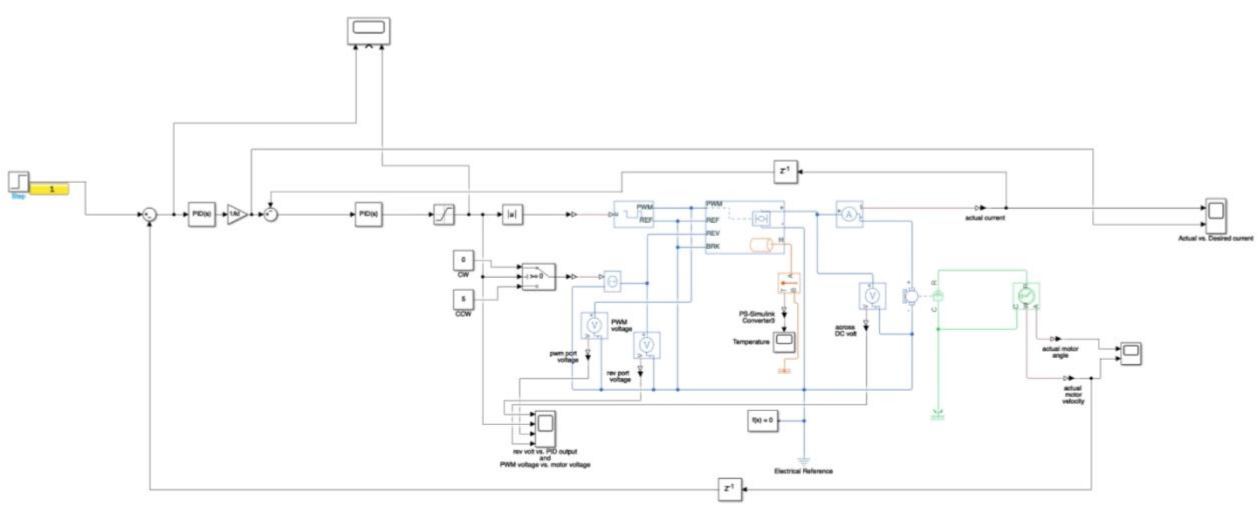


Figure 21: Speed Controller Simulink File for Tuning

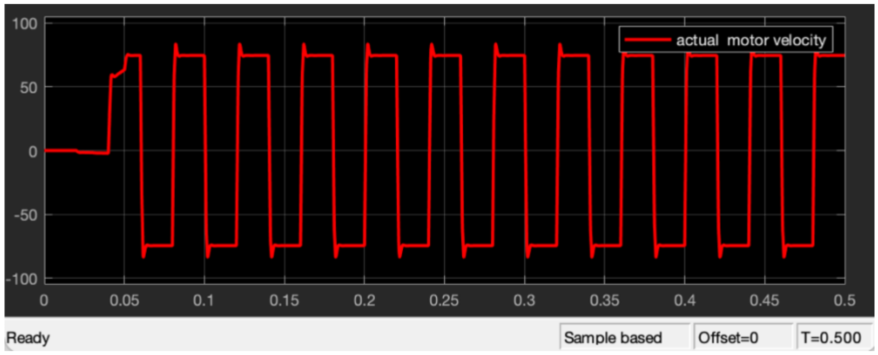


Figure 22: Actual Motor Velocity

Kp was selected to be approximately 120*L and Ki was selected to be 120*R as the default values as determined in the control design. The error magnitude is incredibly large, and the output is simply oscillating around the desired value of 1. Thus, a proportional constant was implemented with the intention to reduce the absolute value and to converge much sooner on the desired value of 1.

The system is much more stable than the initial design. Though the system stabilizes on a value of 0.8 for our final design the proportional parameter will be increased until the output stabilizes on 1. The overshoot % is still quite high and could be reduced by increasing the integral parameter.

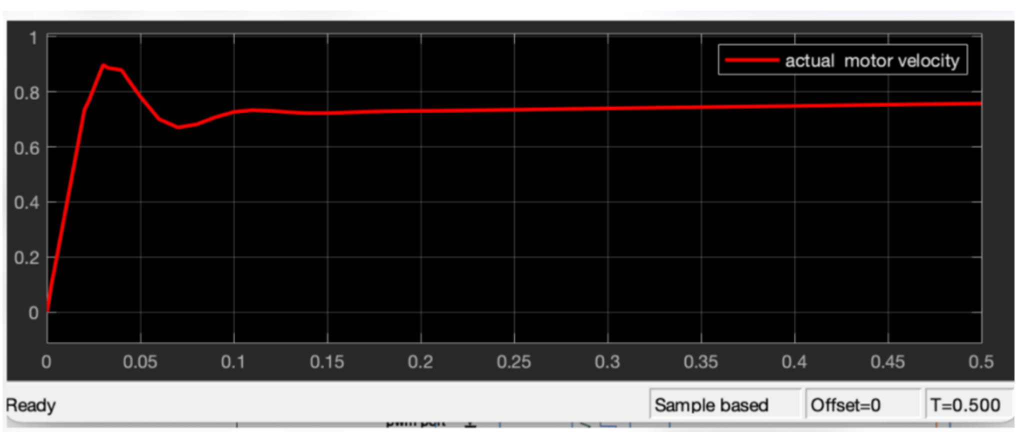


Figure 23: Actual Motor Velocity graph with P=0.8

Position Controller

Background

The position controller can be obtained using the above Speed controller and applying and additional PI controller complete an additional cascade loop. No additional gain is required. Tuning this system was the most sensitive as it with cascaded recursively with the speed and current controllers. Performing simulations on this system was extremely tedious. Top layer of the cascade control loop is the position control feedback loop. Position control is created by adding a step input signal, and an additional PID block to the speed controller loop. Actual position of motor is subtracted from step input signal, and the result is fed into the previous system and considered as desired position.

Implementation

The system was modelled as shown below.

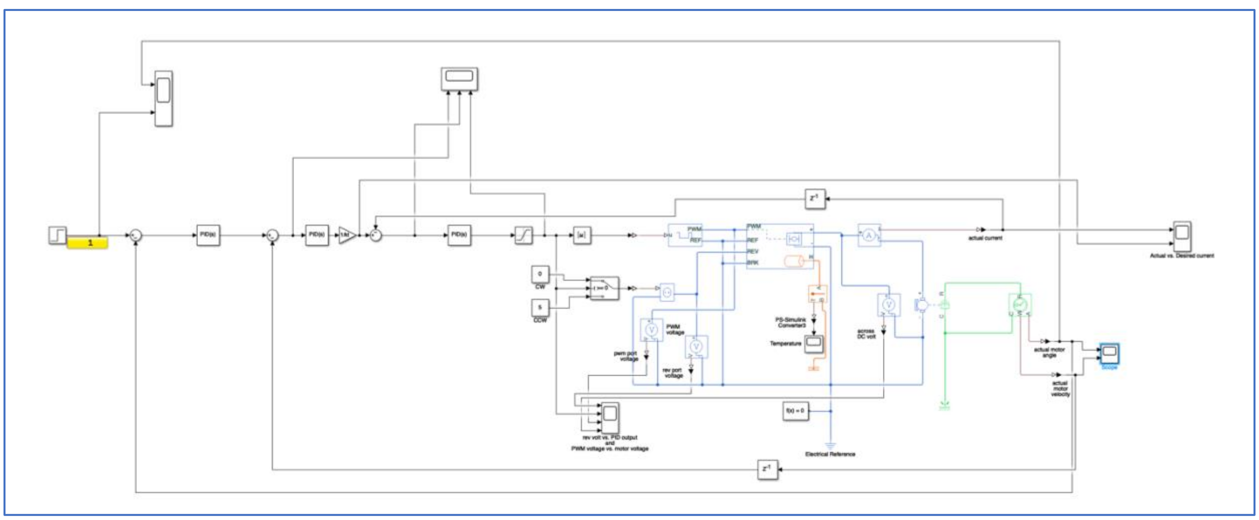
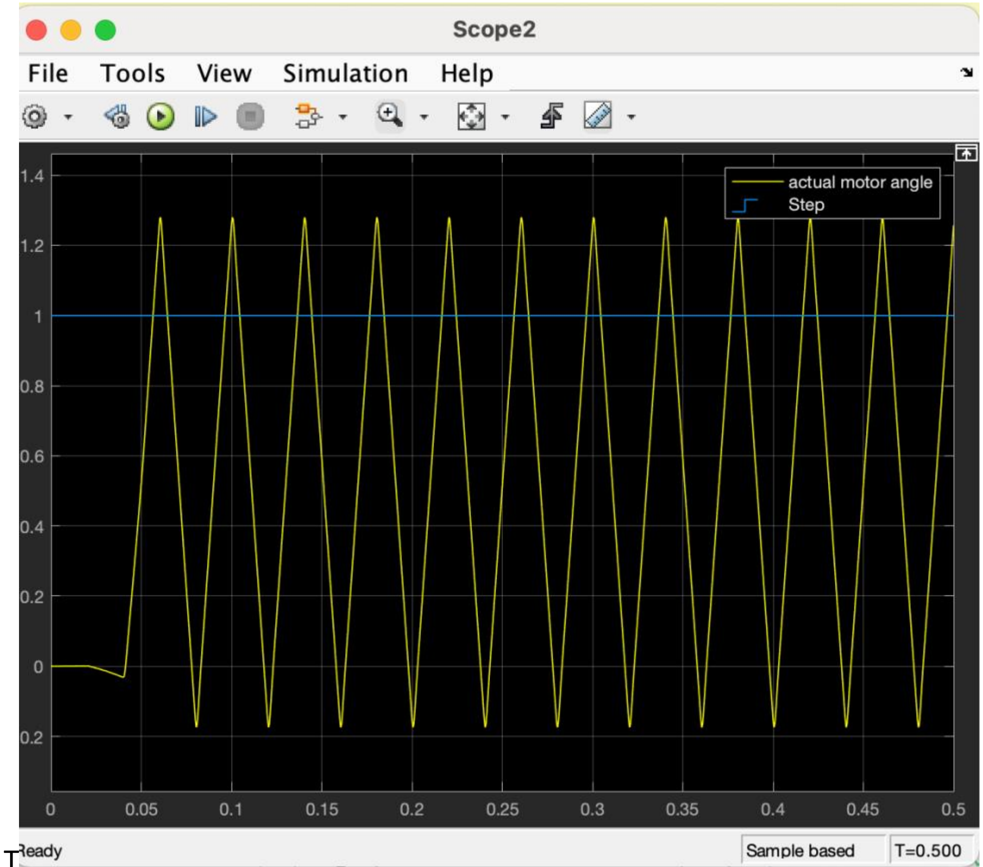


Figure X24 – Position Controller Simulink File

The first iteration of tuning the position controller was awful as it did not converge on our desired value and oscillated on a value less than ours.



T

Figure 25: First Iteration of Position Controller

Performance Evaluation in Presence of nonlinearities

Nonlinearities can be caused by multiple factors such as friction, slip condition and electrical or mechanical component values such as DC motor, H-bridge, gears etc. For this project, Long March is working on simulation of a Throwing Robotic Arm, that was initially designed in Solidworks, and simulations are done in Matlab Simulink. So, this section focuses on nonlinearities in simulations.

Mechanical load variations can be caused by gear or revolt joint values. The following figure of our Simulink design, and different mechanical components can be observed.

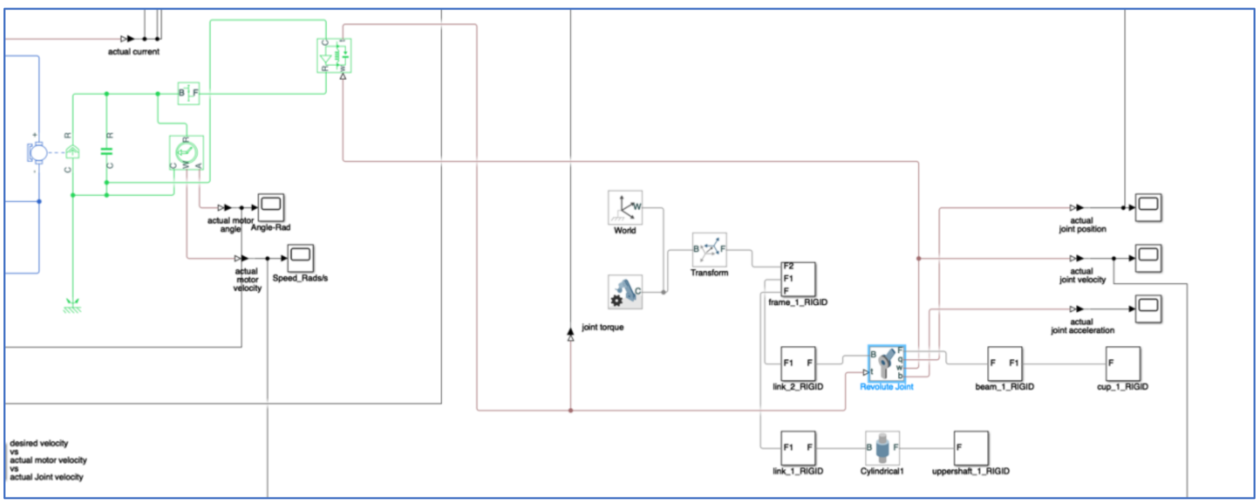


Figure 26 – Integration Mechanical Simulink Design with the Control and Electrical System

The following figure is scope results of position and velocity (desired vs. motor vs. joint) for the original settings of mechanical loads. The following figure is like the previous one, except changes in Simple Gear changing from 1 to 2 values have been implemented.

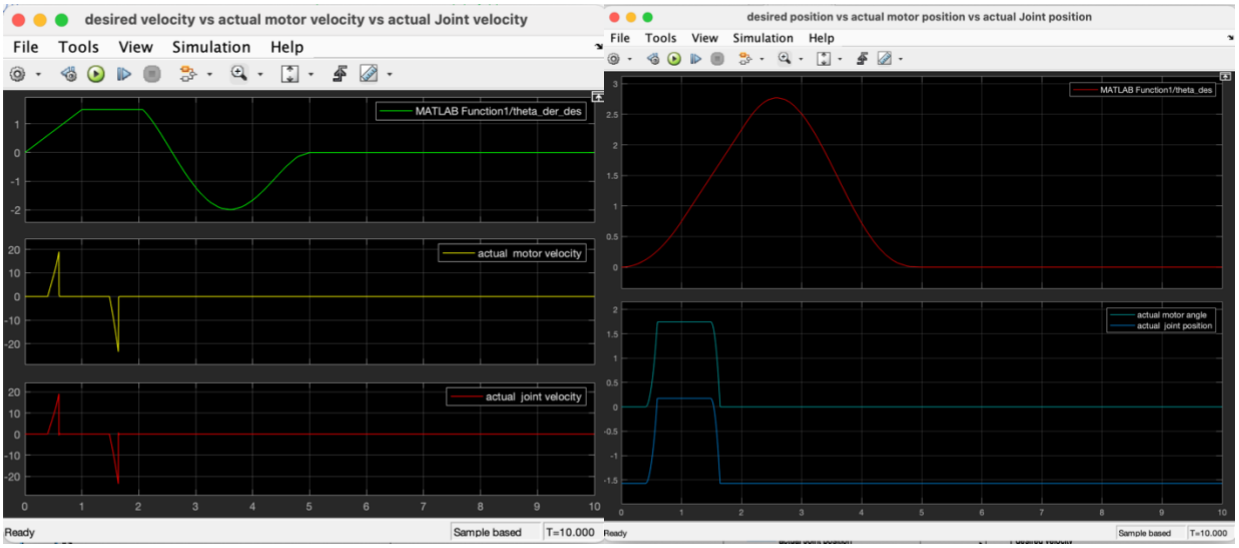


Figure 27 - Desired vs. motor vs. joint position and velocity with Simple Gear Ratio 1

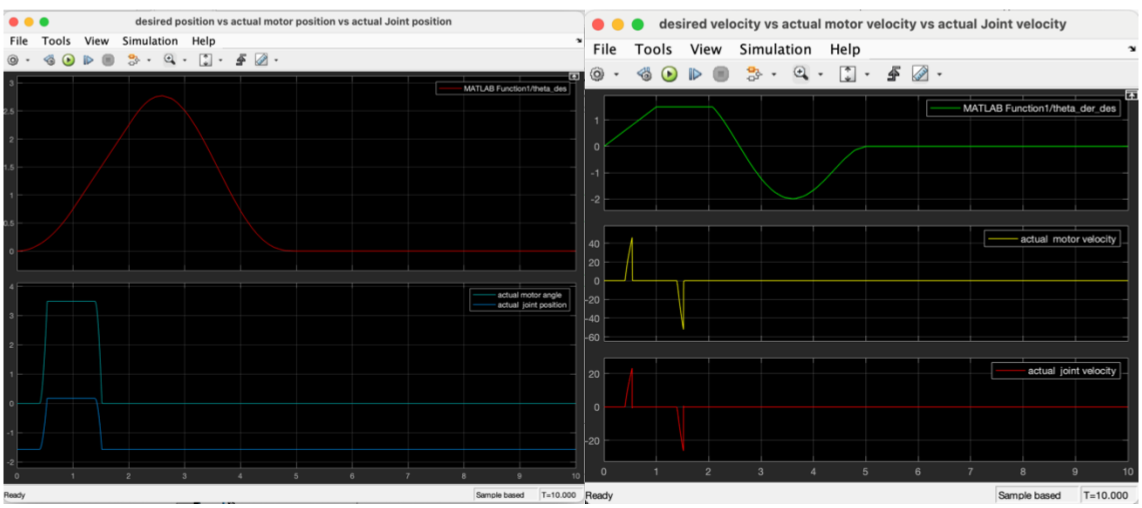


Figure 28 – Changing Simple Gear follower to base ratio to 2

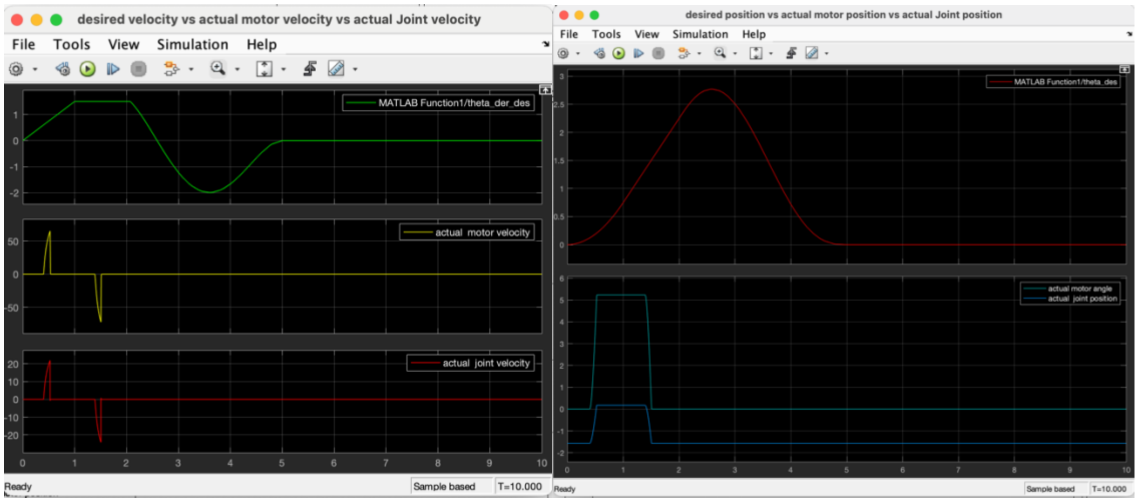


Figure 29 - Changing Simple Gear follower to base ratio to 3

As expected, “actual motor angle” and “actual motor velocity” have been expanded vertically (by factor of 2) compared to the previous figure. Similarly, “actual motor angle” and “actual motor velocity” will be expanded or compressed vertically depending on value of Simple Gear follower to base ratio. Similarly, the following figure shows results when follower to base ratio is changed to 3.

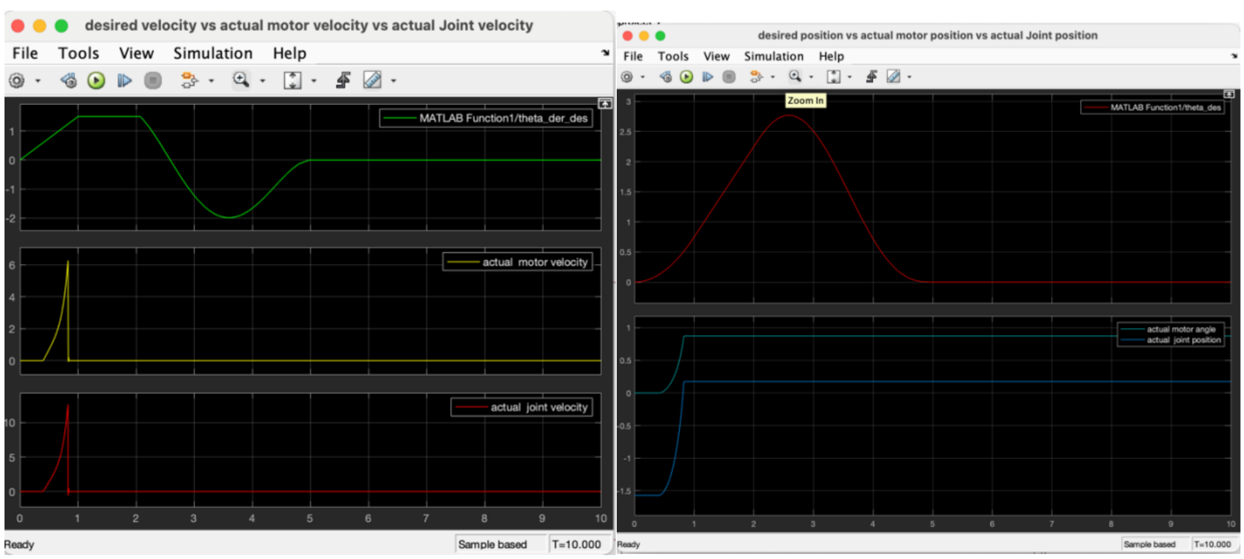


Figure 30 - Changing Simple Gear follower to base ratio to 0.5

The above figures show results of changing follower to base ratio to 0.5. The issue is that the velocity stays non-negative, therefore, position of the joint never goes back to the original position when simulating the design. We were unable to find the exact reason in why this occurs, but we believe that it could be either related to the edge cases of the calculations of supplied torque, or the input threshold of the system to activate is not high enough to help overcome the static friction in a high torque output system. These problems can be fixed by revisiting equations in the PI controller, or adjusting the motor to help match the specifications needed from the system

Table 3 – Rotational Friction Block parameters (original values)

Parameter Name	Parameter Value	Unit
Rotational Friction		
Breakaway Friction Torque	0.00000000000001	Nm
Breakaway Friction Velocity	0.00000000000001	rad/s
Coulomb Friction Torque	0.00000000000001	Nm
Viscous Friction Coefficient	0.00000000000001	Nm/(rad/s)

Another factor contributing to nonlinearities is frictions. In our design, we have Rotational Friction block, that lowers power transition from motor to joint. The following figure shows original settings or Rotational Friction block, which is realistically simulates zero friction.

Table 4 – Rotational Friction first iteration of test

Parameter Name	Parameter Value	Unit
Rotational Friction		
Breakaway Friction Torque	2.5e-6	Nm
Breakaway Friction Velocity	5e-6	rad/s
Coulomb Friction Torque	2e-4	Nm
Viscous Friction Coefficient	5e-3	Nm/(rad/s)

Two other runs were tested, and results can be seen in Figure X and Figure X.

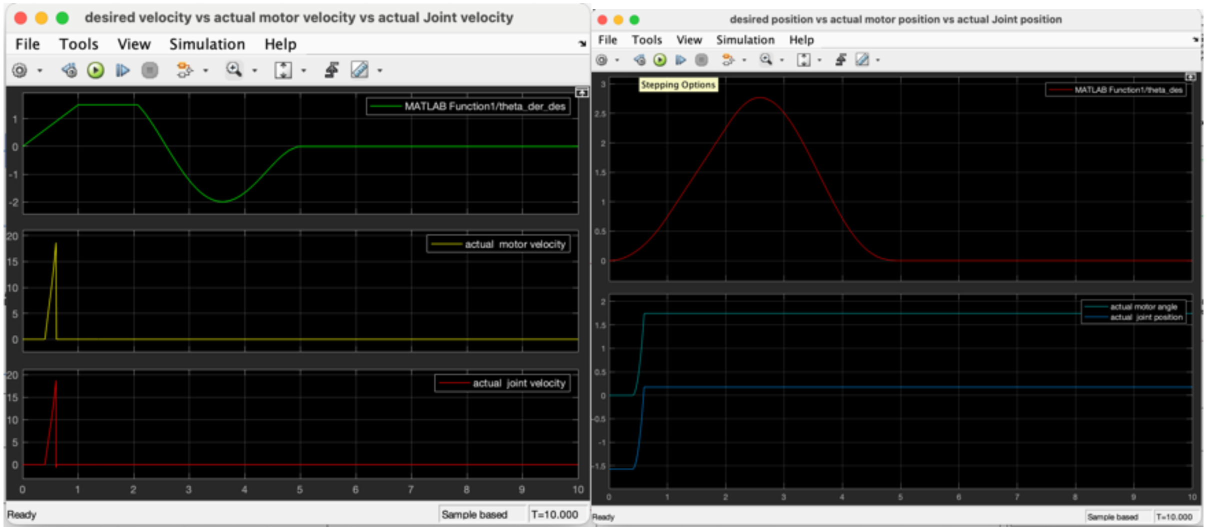


Figure 31 – Too Much Friction, no Reverse Motion

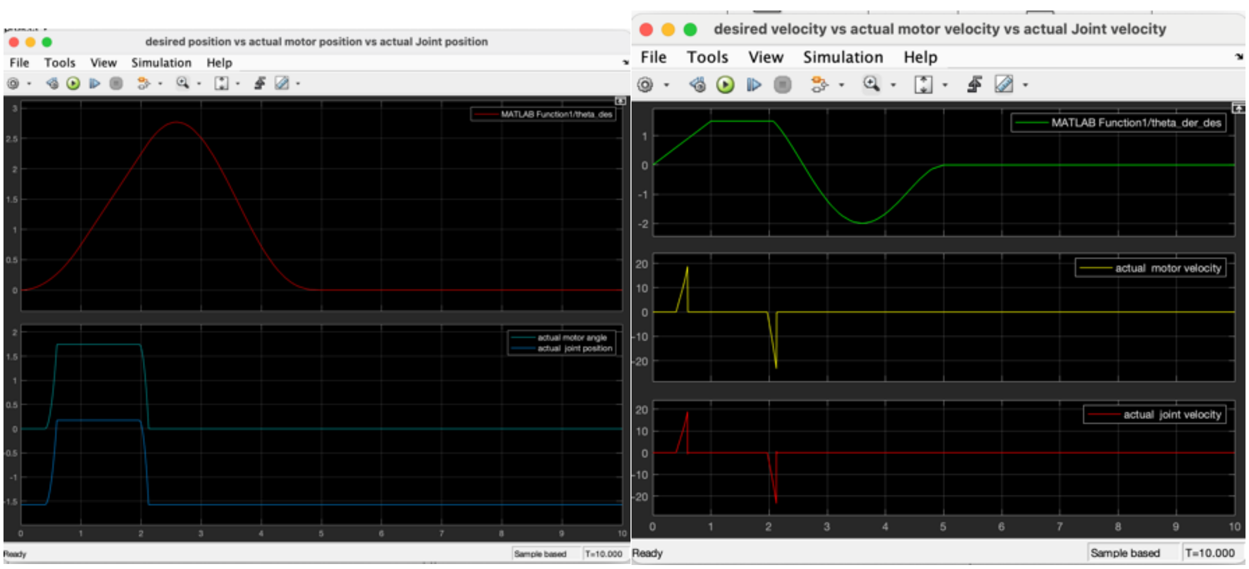


Figure 32 – Limit Friction Where Arm Operates Normally

Table 5 – Rotational Friction Third iteration of test

Parameter Name	Parameter Value	Unit
Rotational Friction		
Breakaway Friction Torque	1.25e-7	Nm
Breakaway Friction Velocity	5e-6	rad/s
Coulomb Friction Torque	1e-7	Nm
Viscous Friction Coefficient	5e-7	Nm/(rad/s)

In addition to these causes that can contribute to nonlinearities in the system, parameter uncertainties play an important role. For example, settings in the above figure (rotational friction) will not be exactly like what has been demonstrated. The reason for that is physical components have uncertainties that can affect overall performance of the system. To have better simulations of the system, it could be beneficial to be able to measure different frictions in the system and compare it to realistic values and revaluate motor selection in the future. It could also be possible that the system is already performing at a “good enough” state where it is able to perform all the necessary objectives given in the initial project description.

When studying the performance regarding Saturation Limits, our main concern is typically what happens to our feedback control system when it begins reaching the physical limits of the system [6]. This type of nonlinearity error occurs in the simulation when our control engineering signal tries to apply an unrealistic signal in which case our H-bridge is not able to keep up with the demands. To simulate the situation in which this would happen within our Simulink simulation we will adjust different parameters within the system. Our goal is to create a situation where we are making the PI controller wants a supplied output voltage greater than what it is rated for. Without changing the motor specifications, we have decided to change the K_p constant in the PI controller. What this will do is cause magnitude of the control signal inputted into the system to change in relation to the constant. As we increase the K_p we should see an increase in the desired voltage into the system, which at one point should go past the values that the H-bridge can supply. From previous testing and block parameters, we know that the system is limited to being able to supply a voltage of about ±11.6V, because torque and voltage should have a positive relationship in our system, we should extrapolate when there is saturation when desired torque increases but not actual torque. In figure [33] we can see when comparing our K_p = 5 and K_p = 10, The actual torque values do not increase anymore despite the desired torque still increasing.

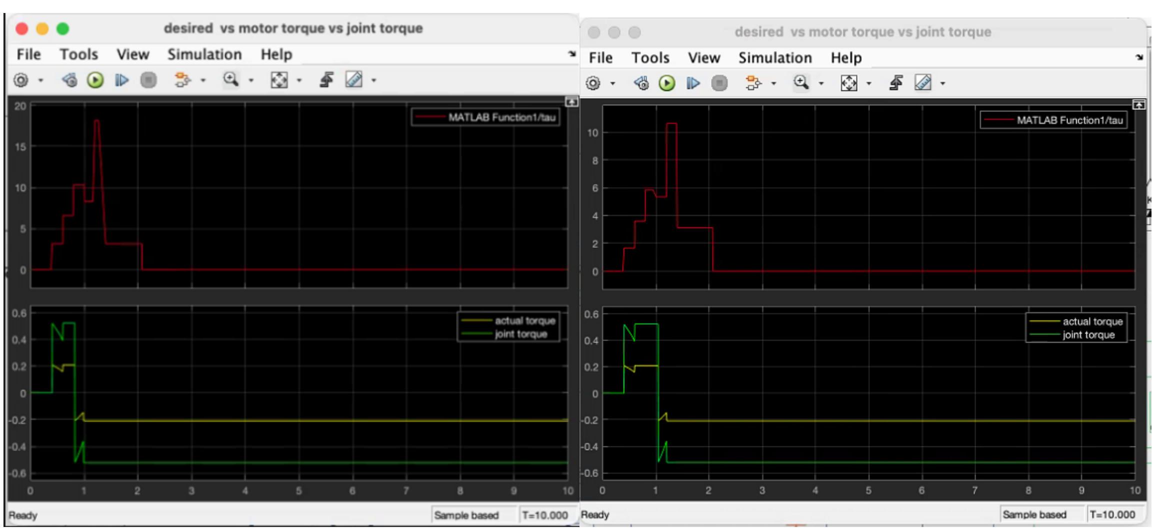


Figure 33 – Desired Torque Graphs when Changing the K_p = 5, 10values

This could be demonstrating a case where the desired torque of the controller cannot be matched by the physical limits of the system. Further testing would be required to find the exact different limits of the system. From additional research, we have found that there are three different methods that can be taken to help with dealing with saturation limits. These methods are the following:

1. Consider the effects of the saturation in our design procedure of the controller
2. Modeling Predictive Control (MPC), which is known as a form of moving horizon control
3. By using a two-step system, where a nominal linear control system is construct ignoring saturation (ex. Standard Linear design tool), and then applying an "anti-windup compensator" which is designed to handle saturation constraints. [7]

As of right now, we believe that these methods exist outside the scope of this course and project, so they have not yet been implemented. If required for performing the actions of the system, they will be investigated for further detail and implemented.

Control Conclusions

The PI controller's parameters are chosen for the best performance of the system. Seeing as the design criteria was to implement an accurate, repeatable, and reliable system each respective controller must be as precise as possible. Particularly when using a cascade controller where each value is used in the next controller it is imperative that each layer has been tuned correctly. According to the simulation results, the following parameters are selected for current and speed controllers respectively: Current: Kp= 0.0001, Ki=10.5 which lead to a 122ms settling time and a 1.5% overshoot., Speed: Kp=, ki=0.1 (for which the actual speed follows the desired speed almost perfectly during the ramp).

Part C: Integrating Part A and Part B

Long March selected Design II from the Mechanical Project to be integrated with electrical and control systems project. Initially, a sperate Simulink file was created to solely simulate the mechanical design. For this project, that Simulink code was integrated with mechanical section of the current design, as it can be seen in the following figure.

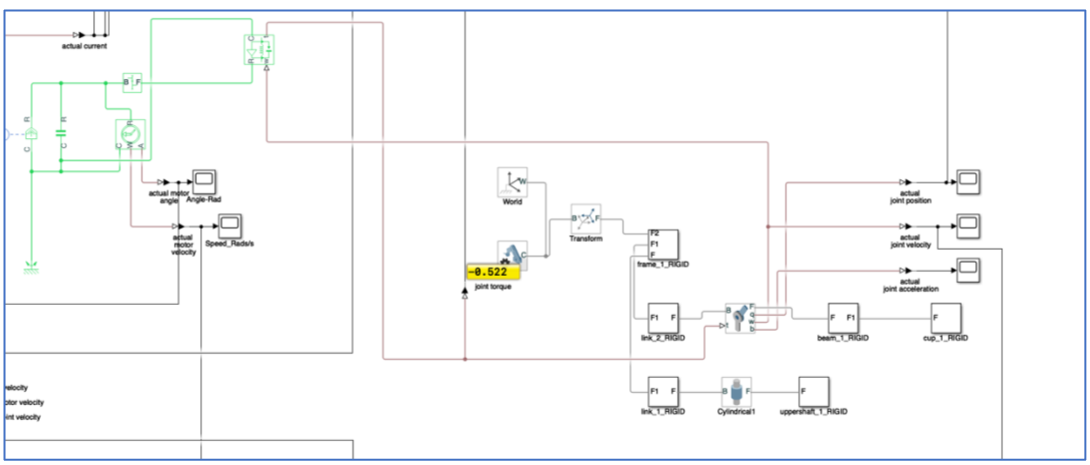


Figure 34 – Mechanical section of current Simulink model

Long March completed two separate labs that prepared us for this project. The first lab, which was focused primarily on electronics components of this design (e.g. DC motor and H-bridge). Purpose of lab 1 was to design and implement a bi-directional DC motor that is controlled by Controlled PWM Voltage Source in Simulink. Overall, the main goal was to become familiar with Simscape electrical by constructing the DC motor (GM8724S009) and H-bridge (TB6568KQ) to drive the motor. Additionally, our team used the model created in lab 1 for the project. Purpose of lab 2 was to design and test feedback controllers in Simulink by using the circuit created in lab 1. In addition, three separate loops were created to control the current (actual current), speed and trajectory (position) of the Throwing Robotic Arm. Similar to lab 1, lab 2 was also used for the main project. Finally, Long March integrated the Simulink file created for Design II

Of the mechanical project into the file that was created for lab 2. Our group faced multiple warnings and errors, so we had to test different sections of our Simulink program (electrical, control or mechanical) to troubleshoot. For example, when the current control loop was not working properly, a step input was used (instead of the desired current input) to test that specific loop. In addition, multiple scopes were connected to each section of the program in order to understand how values are changing and debug the problem. Similar to testing for the current loop, a step signal was also used for speed or position feedback loops, and also the input to mechanical portion.

Table 6: Rotational Friction Simulation Parameters

Parameter Name	**Parameter Value**	**Unit**
	Rotational Friction	
Breakaway Friction Torque	1.25e-7	Nm
Breakaway Friction Velocity	5e-10	rad/s
Coulomb Friction Torque	1e-7	Nm
Viscous Friction Coefficient	5e-7	Nm/(rad/s)

Once the program was running without any errors, the simulation of the Robotic Arm was unsuccessful. So, we had to move back from mechanical portion of the system to control and electrical parts. We realized the problem was with the Rotational Friction block, since the default values in Simulink are implemented for much larger systems, and the power applied for this project is much smaller than those values, hence effects of friction are much more noticeable. Therefore, we tested our model with values seen in the following figure, which is basically zero friction. For more details, refer to “Performance Evaluation and Presence of Nonlinearities” section.

Final integration and Next Steps

References

[1] *Pittman Express GM8724S009 Datasheet*

[2] “The Complete Guide to DC Motors,” *RS*. [Online]. Available: https://ie.rs-online.com/web/generalDisplay.html?id=ideas-and-advice%2Fdc-motors-guide.

[3] “Mathworks,” *DC Motor Model - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/physmod/sps/ug/example-modeling-a-dc-motor.html. [Accessed: 25-Jul-2021].

[4] *Toshiba TB6568KQ Full-Bridge DC Motor Driver IC M*anual

[5] *“Settling Time,” RS.* [Online]. Available: https://en.wikipedia.org/wiki/Settling_time

[6] Haidekker, M. A. (n.d.). “*Saturation Effect*.” Saturation Effect - an overview ScienceDirect Topics. https://www.sciencedirect.com/topics/engineering/saturation-effect. [Accessed: 25-Jul-2021]

[7] Souad Bezzaoucha, Benoît Marx, Didier Maquin, José Ragot. Linear feedback control input under actuator saturation: a Takagi-Sugeno approach. 2nd International Conference on Systems and Control, ICSC 2012, Jun 2012, Marrakech, Morocco. pp.CDROM. ffhal-00687739f

Appendix

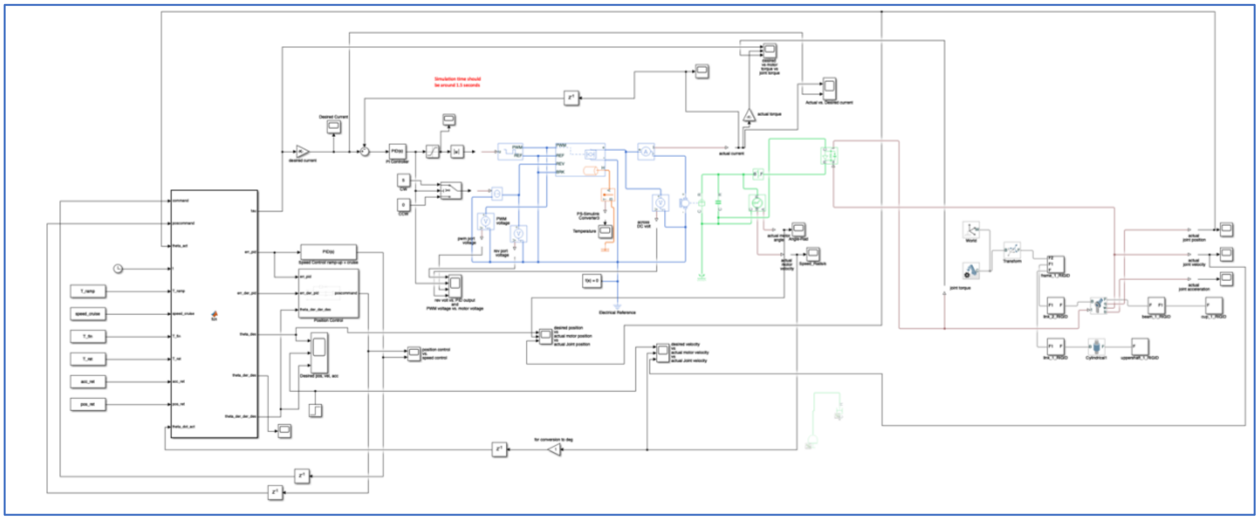


Figure 35 – Overall View of Long March Design

Part A: Electronics Project

Table 7: Simulink Parameters from TB6568KQ Full-Bridge DC Motor Driver Datasheet

Parameter Name	Parameter Value	Unit
Input Threshold		
Enable Threshold Voltage	2.5	V
PWM Signal Amplitude	18	V
Reverse Threshold Voltage	2.5	V
Braking Threshold Voltage	2.5	V
Bridge Parameters		
Output Voltage Amplitude	12	V
Total Bridge on Resistance	0.1	Ohm
Freewheeling Diode On Resistance	0.05	Ohm
Freewheeling Diode Off-state Conductance	1e-6	S
Measurement Temperature	25	°C

Table 8: GM8724S009 DC Gearmotor Simulation Parameters

Parameter Name	Parameter Value	Unit
Electrical Torque		
Field Type	Permanent Magnet	N/A
Model Parameterization	By Rated load and Speed	N/A
Armature Inductance	2.34	mH
No Load Speed	720	rpm
Rated Speed (At rated load)	400	rpm
Rated Load (Mechanical Power)	10	W
Rated DC Supply Voltage	12	V
Rotor Damping Parameterization	By Damping Value	N/A
Mechanical		
Rotor Inertia	1.6e-6	Kg*m^2
Rotor Damping	1.1e-4	Nm/(rad/s))
Initial Rotor Speed	0	rpm

Table 9: Controlled PWM Voltage

Parameter Name	Parameter Value	Unit
Pulse Width Modulation		
PWM Frequency	4000	Hz
Simulation Mode	Averaged	N/A
Output Voltage Amplitude	40	V

Part B: Control Project

Control Function Code

```
function [tau, err_pid, err_der_pid, theta_des, theta_der_des, theta_der_der_des] ...
    = fcn(command, poscommand, theta_act, t, T_ramp, speed_cruise, T_fin, T_ret, acc_ret, pos_ret, theta_dot_act)

    if (t <= T_ramp)
        theta_des = 0.5 * speed_cruise * t^2/T_ramp;
        theta_der_des = speed_cruise * t/T_ramp;
        theta_der_der_des = speed_cruise/T_ramp;
        err_pid = theta_der_des - theta_dot_act;
        tau = command;
        q_0 = []; q_1 = []; q_2 = []; q_3 = []; err_der_pid = 0;

    elseif (t > T_ramp & t < T_ret)
        theta_des = 0.5 * speed_cruise * T_ramp + speed_cruise * (t-T_ramp);
        theta_der_des = speed_cruise;
        theta_der_der_des = 0;
        err_pid = theta_der_des - speed_cruise;
        tau = command;
        q_0 = []; q_1 = []; q_2 = []; q_3 = []; q_4 = []; q_5 = []; err_der_pid = 0;

    elseif (t >= T_ret & t < T_fin)
        del = T_fin - T_ret;
        q_2 = acc_ret * 0.5;
        q_1 = speed_cruise;
        q_0 = pos_ret;
        A = inv([del^3, del^4, del^5; 3*del^2, 4*del^3, 5*del^4; 6*del, 12*del^2, 20*del^3])...
            * [-q_0 - q_1*del - q_2*del^2; -q_1 - 2*q_2*del; -2*q_2];
        q_3 = A(1); q_4 = A(2); q_5 = A(3);
        del_T = t - T_ret;

        theta_des = q_0 + q_1*del_T + q_2* del_T^2 + q_3* del_T^3 + q_4* del_T^4 + q_5* del_T^5;
        theta_der_des = q_1 + 2*q_2* del_T + 3*q_3* del_T^2 + 4*q_4* del_T^3 + 5*q_5* del_T^4;
        theta_der_der_des = 2*q_2 + 6*q_3* del_T + 12*q_4* del_T^2 + 20*q_5* del_T^3;

        err_pid = theta_des - theta_act;
        err_der_pid = theta_der_des - theta_dot_act;
        tau = poscommand;

    else
        theta_des = 0; theta_der_des = 0; theta_der_der_des = 0;
        err_pid = theta_des - theta_act;
        err_der_pid = theta_der_des - theta_dot_act;
        tau = poscommand;
    end


end
```

Figure 36 – Control Function Code

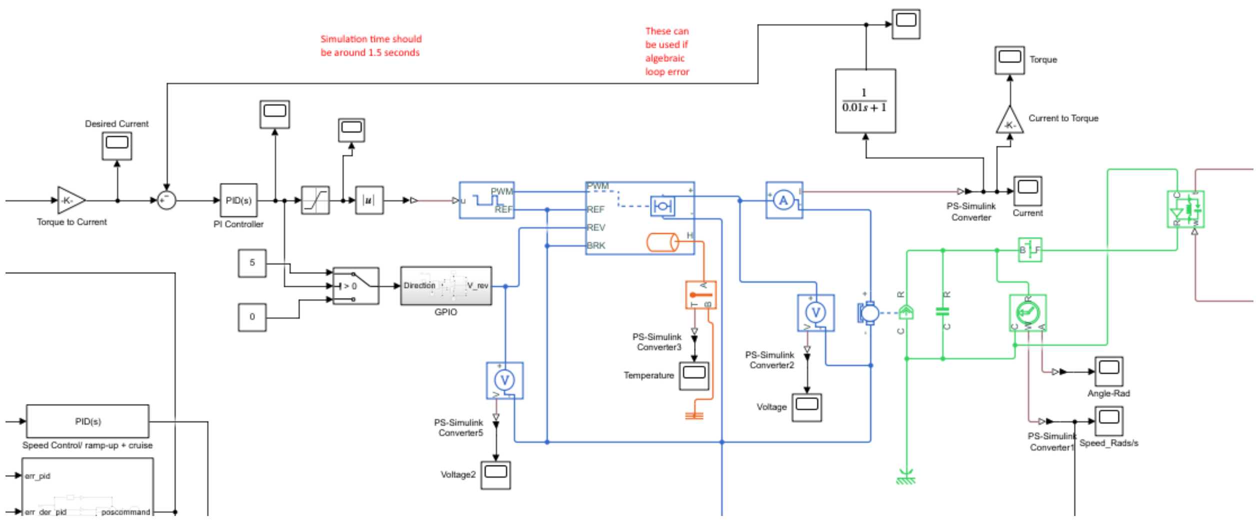


Figure 37 – MATLAB Simulink blocks for Electrical

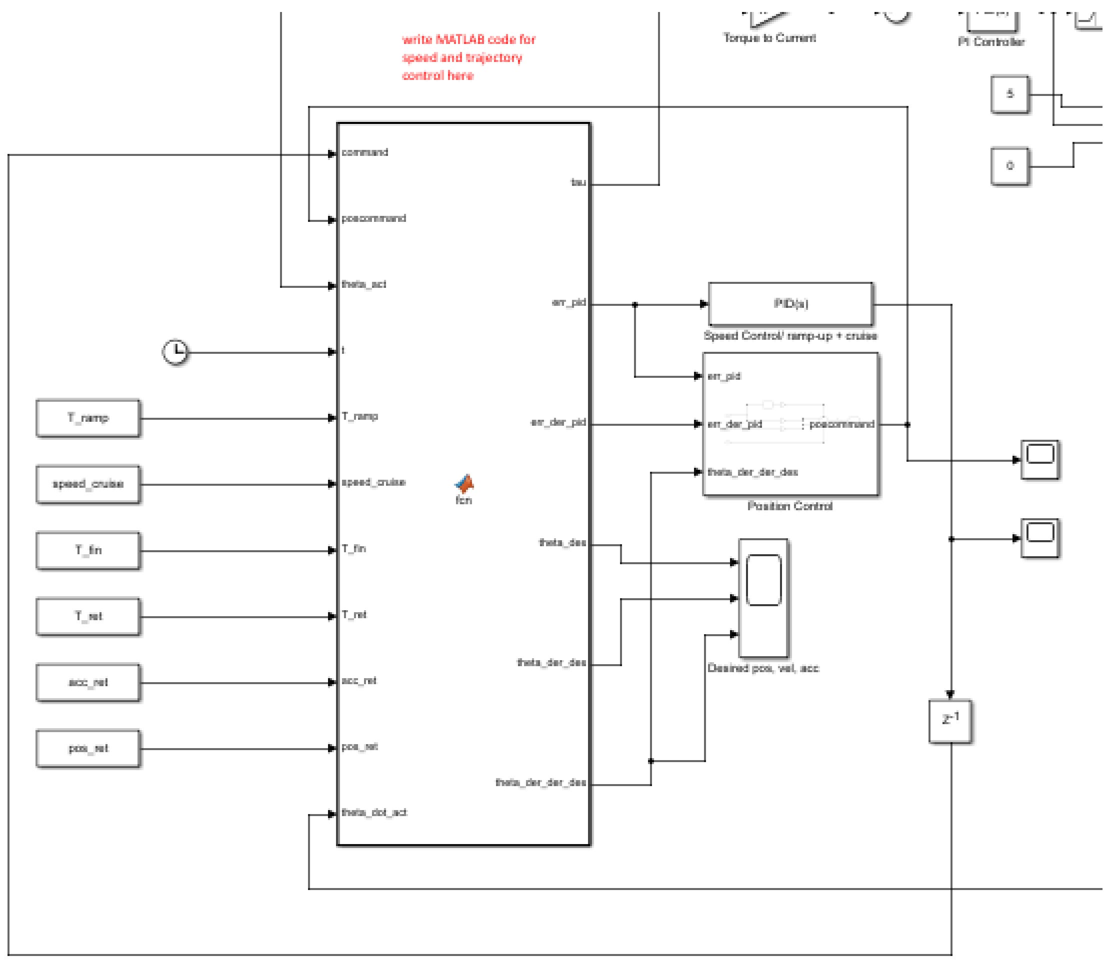


Figure 38 – MATLAB Simulink blocks for control

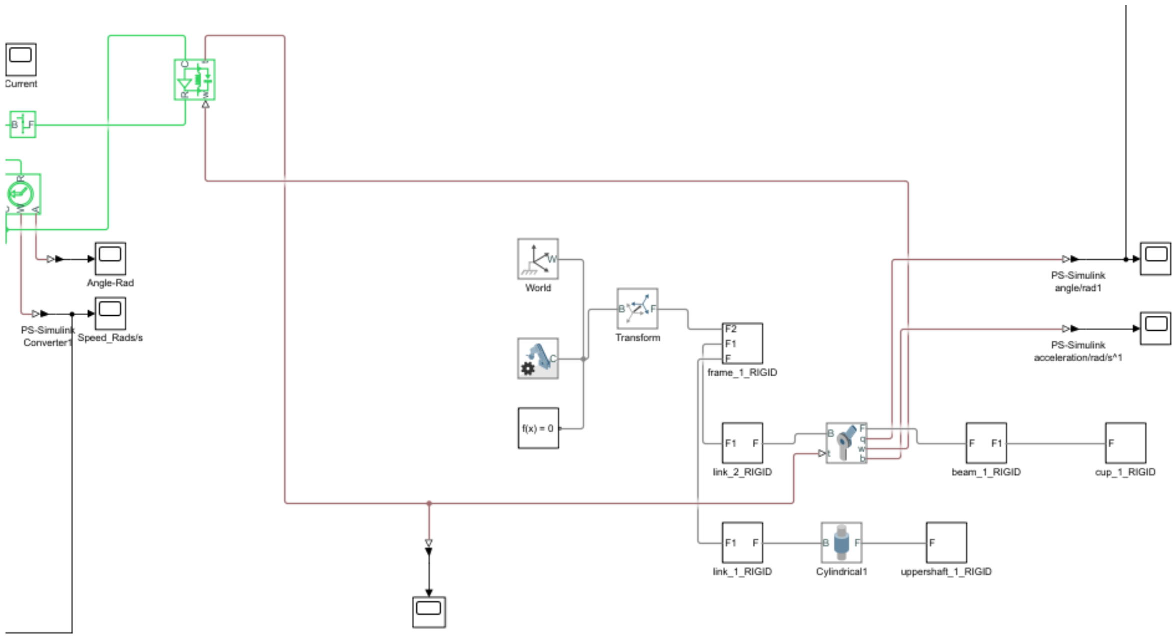


Figure 39 – MATLAB Simulink blocks for Mechanical Integration

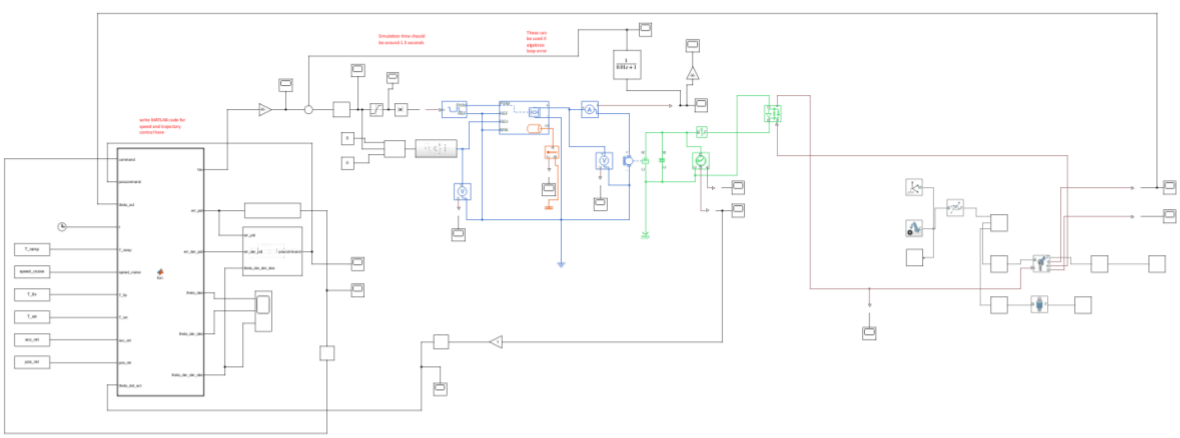


Figure 40 – MATLAB Simulink Blocks Integrated Controls, Electrical, and Mechanical

Project – Electrical/Control Report Contribution

Note: During the first two meetings we discussed task allocation and were not certain as a team of how to move forward. Therefore, no serious tasks were completed prior to Meeting 3.

Meeting 3 - 18, July, 2021

Attendees: Everyone

During this stage of the project, we did the following tasks:

1. Design and complete hand calculations for the H-bridge
2. Complete DC motor and H-bridge calculations based on data sheets
3. Complete current controller for lab 2

Contribution of each member from the previous items:

- Sepehr: Task 2 and task 3
- [redacted]: Task 1 and task 2
- [redacted]: -
- [redacted]: - Task 2 Task 3

Meeting Notes

- Ask TA whether we need to design the h-bride or hand calculations are sufficient.
- Asked [redacted] to integrate Simulink design from Mechanical section of the course to our current design.

Meeting 4 - 19, July, 2021

Attendees: Everyone

During this stage of the project, we did the following tasks:

1. Test the current controller and make sure the graph looks realistic
2. Update design 2 Solidworks so it matched Matlab simulation
3. Update the Mechanical project simulation
4. Continue working on heat sink hand calculations
5. Prepare presentation for TA meeting.

Contribution of each member from the previous items:

- Sepehr: Task 1 and task 5
- [redacted]: Task 4
- [redacted]: Task 2
- [redacted]: Task 2, Task 3

Meeting Notes

- How to resolve the issue of importing files from Solidworks to Matlab (we get an error)

Meeting 5 - 20, July, 2021

Attendees: Everyone

During this stage of the project, we did the following tasks:

1. Integrate the Simulink blocks from mechanical project's simulation to our current file for elec/control project, and test whether simulation is created in the new file
2. Complete the function in the function block that updates desired value for position/vel/acc continuously
3. Complete the speed control and position control
4. Design a gpio block (based on instructor's notes) to faciliate direction control
5. Update cup of the mechanical design

Contribution of each member from the previous items:

- Sepehr: Task 1, task 2 and task 3
- [redacted] Task 4
- [redacted]: Task 5
- [redacted]n: Task 1, Task 5

Meeting Notes

- Create a "What's left" file on team's Google drive so that tasks are more organized and clear
- Make sure everyone is using the most recent version of MATLAB

Meeting 6 - 20, July, 2021

Attendees: Sepehr - [redacted]

During this stage of the project, we did the following tasks:

Troubleshoot Matlab errors

Contribution of each member from the previous items:

- Sepehr:
- [redacted]:

Meeting Notes

Meeting 7 - 22, July, 2021

Attendees: [redacted] – Sepehr

During this stage of the project, we did the following tasks:

1. Test the system with different parameters/ variables and observe results
2. Figure out the problem occurred while integrating Solidworks files into Simulink
3. Calculate pid block's values and use most suitable values for our design

Contribution of each member from the previous items:

- Sepehr: Task 1 and task 2
- [redacted] Task 1
- [redacted] Task 1 and task 3
- [redacted]:

Meeting Notes

- Create a "What's left" file on team's Google drive so that tasks are more organized and clear
- Make sure everyone is using the most recent version of MATLAB

Meeting 8 - 24, July, 2021

Attendees: All members were present

During this stage of the project, we did the following tasks:

- Assigned responsibilities for the report:

1. Introduction
2. Electronics Introduction
3. Motor Selection
4. Design Stages of Power Electronics Drive Circuitry
5. DC Motor Control
6. Thermal Design for PWM Chip
7. Simulation Results for Evaluation Power Drive Circuit Operation
8. Electronics Conclusion

9. Control Introduction
10. Control Design Criteria
11. Design Stages to Support Control Design Process
12. Performance Evaluation in Presence of Nonlinearities
13. Control Conclusions
14. Integration
15. General Formatting

Contribution of each member from the previous items:

- Sepehr: Tasks 7, 12, 14, 15
- [redacted] Task 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13 15
- [redacted]: task 1 ,9
- [redacted]: 1,7, 12,13,14,15

Meeting Notes

- Meet at 7pm on due date to do final grammar checks prior to submission.

Grading Scheme

Design stages of the power electronics drive circuitry (e.g., motor/electromagnet for virtual projects), thermal design for PWM chip, MOSFET drive, etc	**15%**
Simulation results to evaluate the power drive circuit operation. Numerical results must be based on parameters obtained from real datasheets (DC motor, electromagnet, PWM chip, MOSFET)	**15%**
Determining control design criteria such as settling time and error performance	**5%**
Design stages of feedback motion control system by utilizing linear control system methods. Application of concepts in control system design such as transfer functions, root-locus, frequency response, and transient analysis.	**10%**
Study of simulation results to support the control design process such as the following: • Time domain and/or frequency domain response and evaluation of the system performance (e.g., root-locus, frequency response) • Transient analysis of closed-loop behavior • Comparison of performance and justification for using the proposed scheme	**10%**
Time domain response and performance evaluation of the system in the presence of nonlinearities such as power drive saturation (e.g., ±12V saturation limits), friction, mechanical load variations, and parametric uncertainties. Justification for any discrepancies and suggestions for improving performance.	**15%**
Integration of the electronics circuit, feedback control, and mechanical system and performance evaluation of the complete system.	**10%**
Presentation: Quality graphs, formulas, flow of concepts, citations, discussion of results, conclusions, attachments	**20%**