AA 448

Lab 2- DC Motor Control with Arduino

Andrew Gorecki, Nick Kano, Leif Wesche

William E. Boeing Department of Aeronautics & Astronautics
University of Washington, Seattle, WA 98195-2400

2018-12-03

Abstract: The physical characteristics of a DC motor were determined through a series of tests. These physical characteristics were used to construct a model of the DC motor system and simulate the system's response to inputs and disturbances. Various control schemes were then implemented on the physical model. The first controller used was an open loop controller, which required no feedback sensing but lacked the ability to respond to disturbances. An optical encoder was used to measure the position of the motor, which was used in several closed loop feedback systems. The first closed loop controller tested was a bang-bang controller, which was found to effectively respond to disturbances, but oscillated around a desired velocity. The bang-bang controller was also found to switch the input voltage between on and off at a very high frequency. To reduce this frequency a hysteresis band was implemented, but this also resulted in an increased oscillation magnitude. The elements of proportional, integral, derivative control were studied individually and in conjunction. It was found that the PID control scheme could be tailored to specific applications through conscious choice of the PID control constants. The PID's ability to approach a steady state voltage was found to place less stress on the physical system than the bang bang controller and increased ability to track the desired position.

Nomenclature

b_m	Resistive Torque	(Nm)
e	Back emf Voltage	(V)
$f_{rotation}$	Frequency	(Hz)
i	Current	(A)
J	Moment of Inertia	(kgm^2)
K_D	PID Derivative Gain	(No Units)
K_I	PID Integral Gain	(No Units)
K_P	PID Proportional Gain	(No Units)
K_T	Mechanical Machine Constant	(Nm/A)
K_V	Electrical Machine Constant	$(Vsec)$
L_a	Motor Inductance	(H)
N	Number of Encoder Slits	(No units)
R	Resistance	(Ω)
T	Torque	(Nm)
t	Time	(sec)
V	Voltage	(V)
α	First Coefficient of the 2nd Order Polynomial Fit	(No Units)
θ	Angle of Rotation	$(radians)$
ω	Angular Velocity	$(radians/sec)$

Subscripts

a	Armature of Motor	(No units)
L	External Loading	(No units)
m	Motor	(No units)
R	Resistor	(No units)
s	Supply	(No units)
ss	Steady State	(No units)

I. Introduction

The purpose of this lab was to investigate techniques used to model and control a DC motor system. The lab was broken up into four weeks. In week one, various motor parameters were determined using a series of tests in order to model the motor. In week two, the optical encoder interface was set up as a means of measuring the motor's speed and position, and the remaining motor parameters were evaluated. The motor was

modeled mathematically in order to predict the behavior when subjected to various voltage inputs or resistance loads. In week three, basic control methods such as open loop and bang-bang closed loop controls were investigated. Finally, in the fourth week, the more complex and robust closed loop PID control method was implemented.

II. Theory

Below lists the equations used to predict and describe the dynamics of the systems in this lab.

The linear relationship between voltage, current, and resistance is given by the following equation:

$$V = i * R. \tag{1}$$

The DC motor electrical equation of motion is described by the following equation:

$$V_a(t) = (R_m + R)i(t) + L_a \frac{di(t)}{dt} e(\omega). \tag{2}$$

With the rotor locked and in a steady state position, The DC motor electrical equation of motion can be simplified to the following equation:

$$V_a = (R_m + R) * i_{ss}. \tag{3}$$

The DC motor mechanical equation of motion is described by the following equation:

$$J_m \frac{d^2\theta}{dt^2} = T(t) + b_m(\omega) - T_L(t). \tag{4}$$

With the rotor locked, the DC motor inductance can be calculated using the following equation:

$$L_a = (R_m + R) * t_c, \tag{5}$$

where t_c represents the time that the motor takes to go from zero to 63.2% of the steady state current.

With the rotor locked, the DC motor inductance can be calculated using the following equation:

$$L_a = \frac{(R_m + R)}{\omega_c}, \tag{6}$$

where w_c represents frequency where the motor transfer function magnitude drops to 3.01dB below the DC gain.

The rotational speed of the DC motor can be calculated using the following equation:

$$\omega_{ss} = \frac{2\pi}{N} * f_{rotation}, \tag{7}$$

given the frequency of the encoder pulses and the number of encoder slits.

The follwoing equation relates the armature voltage to the motor speed in the standard form of y=mx+b, where the slope is the electrical machine constant.

$$V_a = K_V * \omega_{ss} + (R_m + R) * i_{ss} \tag{8}$$

Derived by equating the mechanical and electrical power in the motor system, the following equation states that the electrical and mechanical machine constants are numerically equal in SI units.

$$K_V = K_T \tag{9}$$

The following equation describes the friction force acting on a DC motor in steady state conditions as a function of angular velocity.

$$b_m(\omega) = K_T \frac{V_a - K_V * \omega_{ss}}{R_m + R} \tag{10}$$

The moment of inertia can be determined by subjecting the motor to a short pulse:

$$J_m = \frac{K_T \frac{V_a}{R_m + R} - b_m(0^-)}{2\alpha} \tag{11}$$

where α is the first coefficient determined by curve fitting the step response of the motor.

III. Experimental Apparatus

An Arduino Due was used to input a range of voltages into the circuit. By using Matlab and Simulink software to interface with the Arduino Due , the voltage input could be monitored and controlled. While this can also be accomplished using the Arduino software, Simulink provides a bridge that interfaces Matlab with the Arduino and allows the data to be analyzed in a timely manner.

While the output voltage of the Arduino is strictly positive, many sensors operate at a range of [-5,5]V. The Forward Biasing Circuit (FBC) served to shift the signal voltage such that the output voltages could be negative. A schematic for this circuit is shown in Fig. 1 below.

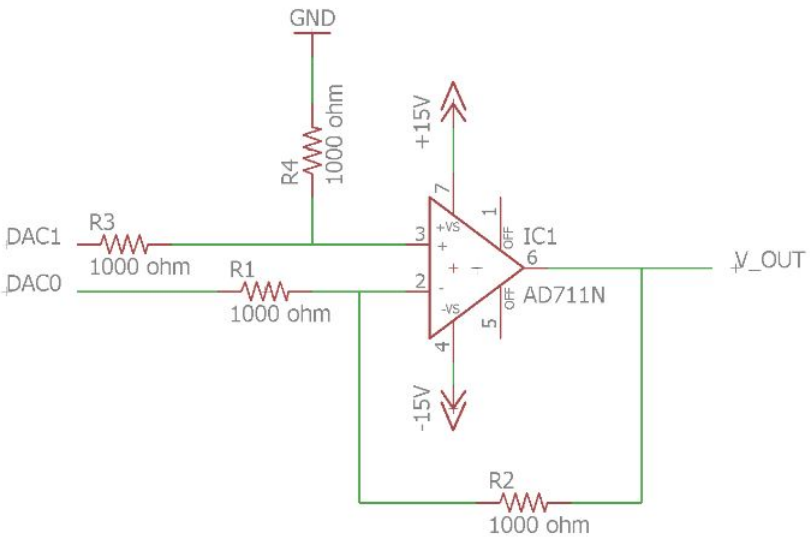

Figure 1: Diagram of the Forward Biasing Circuit. [1]

The Pre-Amplifier increased the range of the voltage output by the FBC and is shown in Fig. 2. Specifically in this experiment, the gain of the pre-amp was 6.

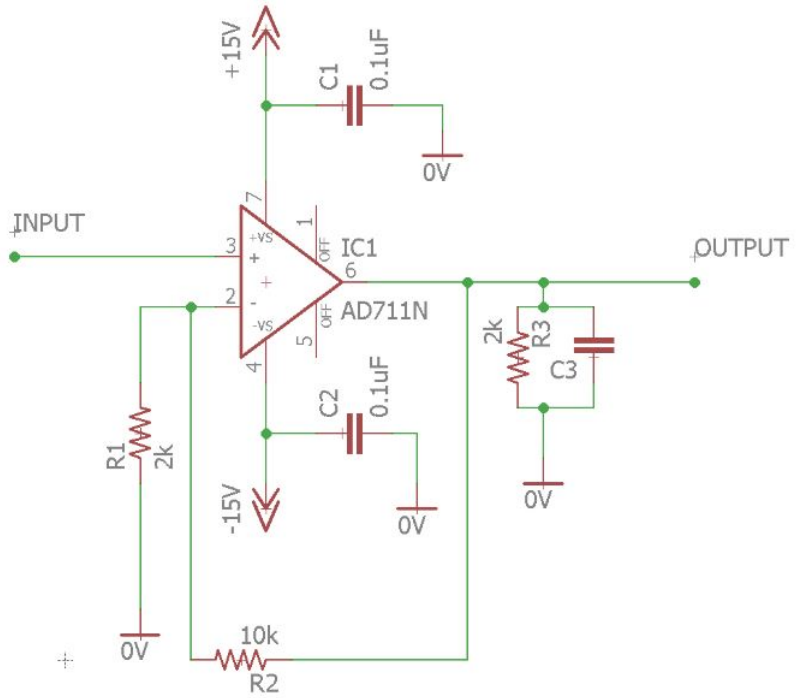

Figure 2: Diagram of the Pre-Amplifier. [1]

A schematic for the DC motor used in this experiment is shown in Fig. 3 while the motor diagram is shown in Fig. 4.

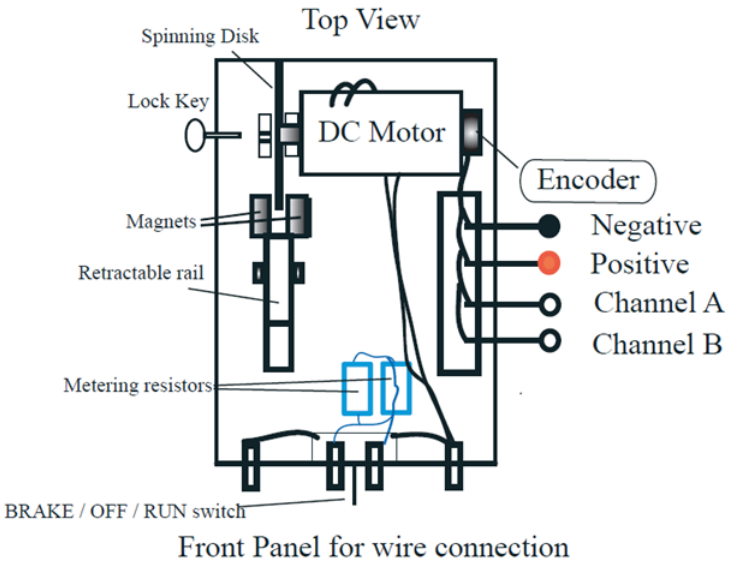

Figure 3: DC Motor Schematic. [1]

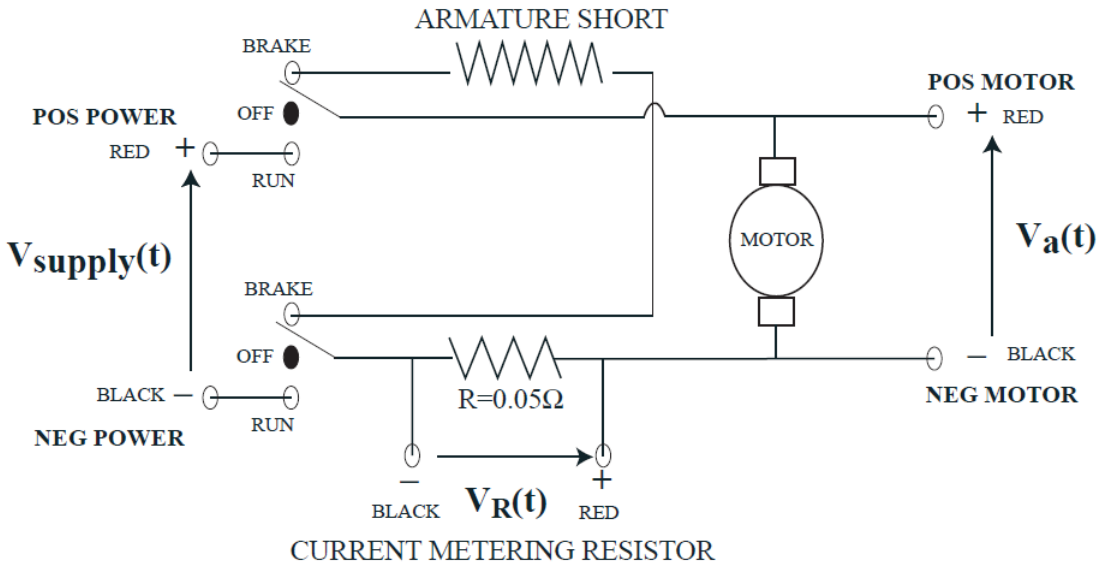

Figure 4: Diagram of the DC Motor. [1]

The panel front of the DC motor is shown below in Fig. 5. The front panel of the motor contained the positive and negative motor and power terminals, a physical power switch, and terminals for measuring the current.

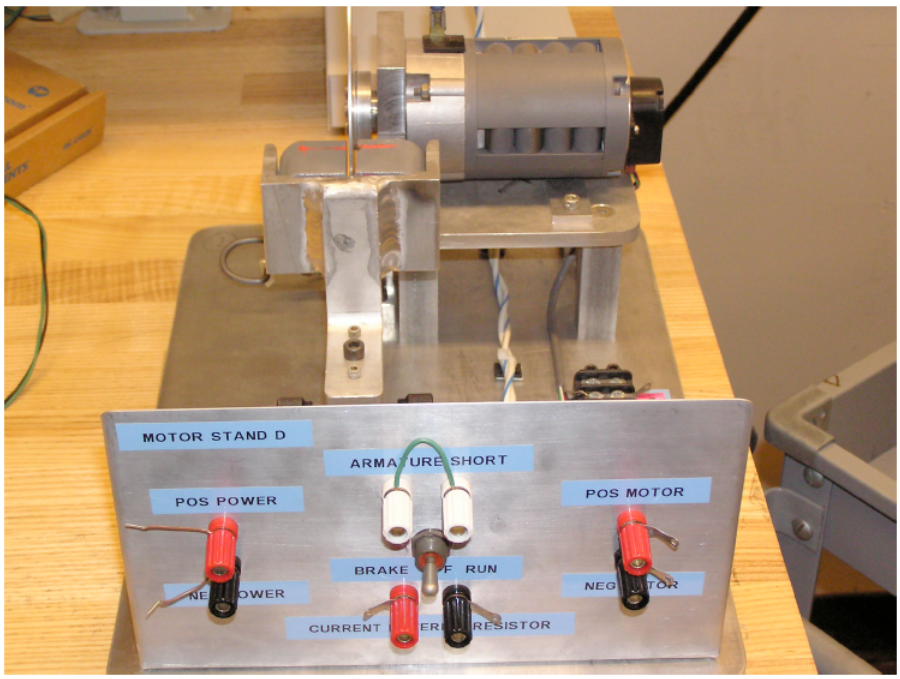

Figure 5: Physical DC Motor. [1]

In order to obtain the position of the motor, a incremental quadrature encoder was used to interface between the Arduino and the LS7366 Chip which counts the number of pulses. This encoder is shown in Fig. 6.

Figure 6: Incremental Quadrature Encoder. [1]

A GW GPC-30300 power supply was used to supply the necessary ± 15V to power the components.

A Kepco Bipolar Operational Power Supply/Amplifier was used amplify the voltage output by the pre-amplifier and input into the DC motor.

The HP Dynamic Signal Analyzer was used to simultaneously generate input signals while measuring corresponding output signals.

A Tektronix TBS 1052B-EDU Digital Oscilloscope was used monitor the input and output behaviors of the motor.

Two Kiethley 2110 5 1/2 Digital Multimeters were used read voltage values and confirm that each component was working as expected.

IV. Procedure

Week 1

The goal of week one was to determine physical motor parameters by conducting a series of tests to model the DC motor system. The parameters determined were R_m, L_a, K_V, K_T, and $b_m(\omega)$.

First, the hardware for the DC motor was set up and the lab equipment used for the system was recorded. The Kepco amplifier was set to limit the voltage range to $\pm$10V before the motor was powered. The Kepco was connected to the positive and negative power terminals while the first channel of the digital multimeter was connected to the current metering resistor, to measure the voltage drop across the known resistor. The second channel of the digital multimeter was connected to the positive and negative motor terminals to measure the motor armature voltage. The terminals used for all of the motor connections in week one are shown in Fig. 5.

The first parameter, R_m, was determined by locking the rotor in place, applying a DC supply voltage, and recording the armature voltage along with the voltage drop across the current metering resistor. The steady state voltage was determined using the known value of the current metering resistor and V_R in Eq. (1). Then, R_m was determined using Eq. (3). A series of 9 DC supply voltages in the range of $\pm$8V were applied and R_m was calculated at each voltage. The final value of R_m was determined by calculating the average of these values, and is shown in Table 2.

Next, the motor inductance, L_a, was determined using two separate methods. First, L_a was determined by analyzing the transient response using the oscilloscope. The motor was again locked and the oscilloscope was used to record V_a. A constant 5V DC voltage was supplied to the motor while the transient voltage, as a function of time, was recorded by the oscilloscope and saved. Using the data from the scope, the time constant t_c was determined, and with Eq. (5) the motor inductance was calculated.

The second method for calculating the motor inductance used the dynamic system analyzer (DSA) to experimentally determine the frequency response of the motor hardware. The Kepco was connected to channel 1 of the DSA and channel 2 was connected

to the current metering resistor on the motor hardware. First, a sine sweep test was run on the motor hardware, then a white noise test was run. The data from each test was saved and recorded. The sine sweep and white noise bode plots from both methods were plotted, and a break frequency, ω_c, was determined from each graph. Using each break frequency along with Eq. (6), the motor inductance was determined for each case.

The transient response test, sine sweep test, and white noise test together yielded three separate values of the motor inductance. L_a was taken to be the average of these values, and is shown in Table 2.

Next, the electrical and mechanical machine constants, K_V and K_T respectively, were determined. First, the rotor was unlocked and the digital multimeter was set up to measure V_R and V_a. Both channels of the oscilloscope were attached to the optical encoder to view the square wave pulses and measure the frequency of the waves. The motor was turned on and a constant voltage of either 2V or 5V was supplied. The brake was implemented in the quarter, half, three-quarter, and full positions for each voltage. Data was collected once the motor reached a steady state for multiple data points.

The oscilloscope was used to measure the frequency of the square waves pulses output by the encoder and the steady state motor speed, ω_{ss}, was calculated using Eq. (7). The measured values of V_R were used along with Eq. (1) to determine the steady state current. Using Eq. (8) along with the data from the eight measured points, the electrical machine constant K_V was determined by plotting the motor speed versus the value of $V_a - (R_m + R) * i_{ss}$ at each point and fitting a best fit line to the data, where the slope was K_V. According to Eq. (9), the mechanical machine constant K_T is equivalent to K_V. The values of K_V and K_T are shown in Table 2.

To simplify the process of predicting the friction of the motor, an assumption was made so that the friction could be modeled as a function of speed. To characterize the motor friction, the friction at various speeds was recorded and organized into a look-up table which could be used to interpolate between measured friction values in the motor model. To do this, the eddy current brake was completely removed and the oscilloscope was set up to read the frequency of encoder pulses. The supply voltage was adjusted so that the armature voltage ranged from 10V to -10V in steps of 1V. The frequency of encoder pulses was recorded at each voltage for a total of 21 points. Equation 7 was used to calculate the speed of the motor, and Eq. (10) was used to calculate the friction at each point. The friction curve generated is shown in Fig. 17.

Week 2

The goal of week two was to configure the optical encoder and use the encoder to obtain the moment of inertia of the wheel, J_m. The incremental quadrature encoder output pulses in the form of square waves. These square waves were then sent to a LS7366 processor designed to map the square waves to the angular position of the motor.

The encoder worked by rotating a set of two disks with rows of slits aligned along the disks. The slits on each disk are misaligned slightly. One disk corresponded to channel A, while the other corresponded to channel B. A light shines through the slits as the disk spins, and the light signal is received up by a optical sensor. Each disk outputs a binary signal, and the two signals make up four possible states. These four states allow the encoder with 500 slits on each disk to divide one revolution into 2000 steps. To keep track of the motor position, each state change was counted.

The first method used was the "naive" approach, where the Arduino was connected directly to the encoder and used to count the pulses instead of the LS7366 processor. In other words, the digital inputs of the Arduino were used to take in the outputs of the encoder, and the Arduino kept track of the pulse pattern manually. The encoder was connected to the Arduino as shown in Fig. 7

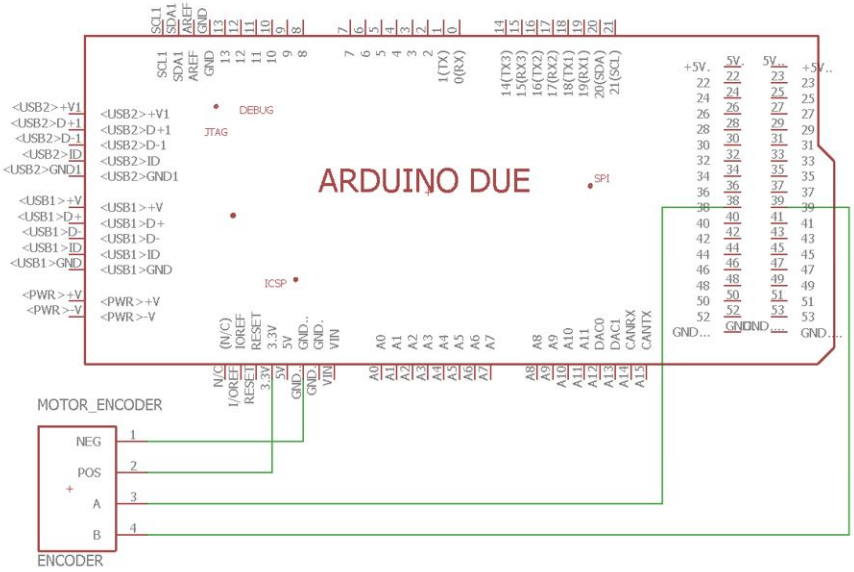

Figure 7: Connections Between the Quadrature Encoder and the Arduino. [1]

The provided "IncrementalRotaryEncoder" Simulink block [1], shown below in Fig. 8, was used to interface with the Arduino and count the outputs from the encoder and return the

angular position of the wheel. The system was tested by gradually increasing the speed and recording the position using the "naive" approach.

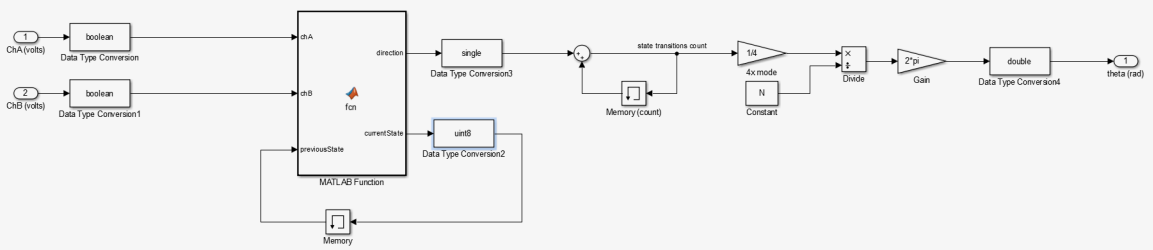

Figure 8: "IncrementalRotaryEncoder" Simulink Block. [1]

To explore the possibility of the system tracking faster angular velocities, a variety of tests were run. The first method was to decrease the sample time until the Arduino could no longer record the velocity. The second method was to determine how much data the system could handle before it could no longer operate. Both methods were tested and the results were recorded.

The main system that would be used to obtain the position of the wheel used a separate LS7366 processor to offload the computation. This chip was used because to record the pulses in place of the Arduino because it was built to specifically handle this type of simple computation much more efficiently than the Arduino. In other words, the LS7366 was theoretically able to record the pulses at much higher speeds than the Arduino. The LS7366 was connected to the Arduino as shown in Fig. 9 below.

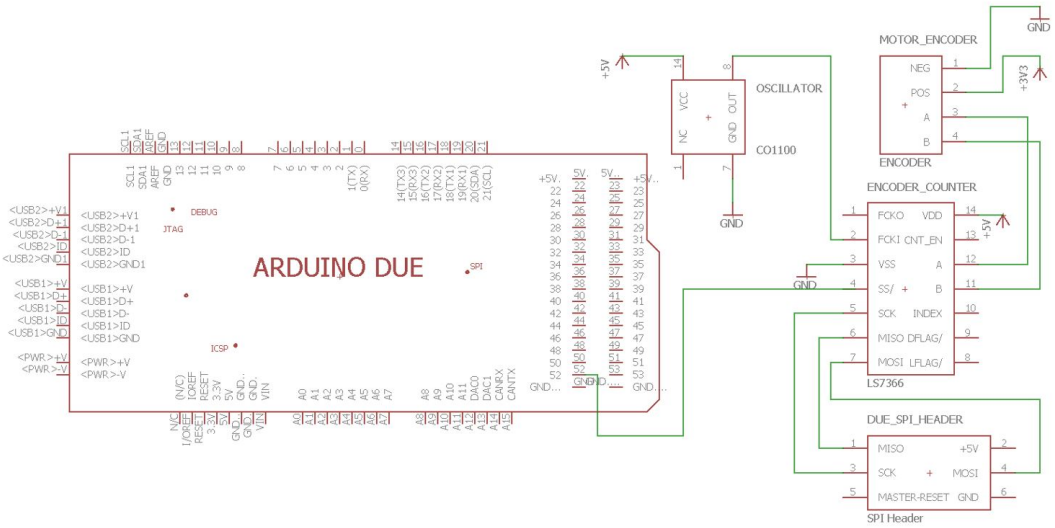

Figure 9: Encoder System Circuit Diagram Using the LS7366 Processor. [1]

First, an Arduino sketch was used to interface with the LS7366. The provided "LS7366SingleEncoder.ino" [1] sketch was pushed to the Arduino Due from the host PC, and the encoders position tracking of the motor was observed in real time by viewing the encoder count through the Arduino serial monitor and spinning the motor by hand. An example of the serial output is shown in Fig. 19.

Next, the provided "Plant (DC Motor) Arduino" Simulink block [1] was used to supply a voltage to the motor and record the position measured by the encoder. This block interfaced with the Arduino shield by receiving a desired motor voltage input and using the FBC, BBC, pre-amp, and Kepco amplifier Simulink blocks from Lab 1 to convert this voltage to the corresponding Arduino output voltage. The block then output the appropriate output voltage from the Arduino. The block also interfaced with the encoder to measure the position of the block. The "Plant (DC Motor) Arduino" [1] Simulink block is shown below in Fig. 10.

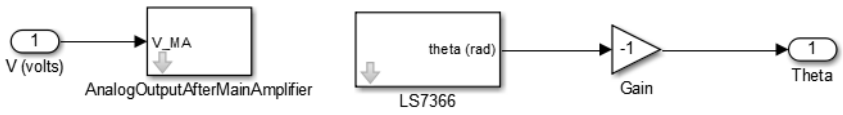

Figure 10: "Plant (DC Motor) Arduino" Simulink Block. [1]

A "Velocity Estimation" [1] block was also provided to calculate the angular velocity of the motor given the measured position. This was done using a pseudo-derivative transfer function block. The "Velocity Estimation" block is shown below in Fig. 11.

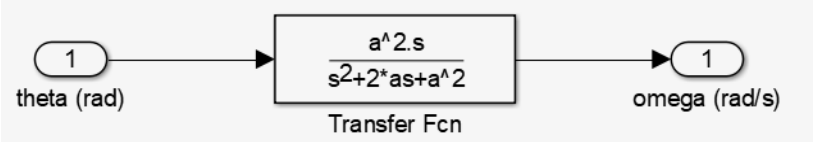

Figure 11: "Estimate Velocity" Simulink Block. [1]

The voltage output of the block was tested by setting the DC motor input to a constant voltage and measuring the voltage using the digital multimeter. The position tracking was tested by setting the input voltage to zero volts, then spinning the motor by hand one revolution and confirming that the recorded displacement matched the physical displacement.

Finally, tests were run to determine the moment of inertia of the system. The sampling time was set to be 0.001, hardware and software voltage limits were set to ± 10V, and it was ensured that a positive voltage produced a positive displacement of the motor. A Simulink block was created to apply a $3V$ pulse to the motor for 0.25 seconds using "Plant (DC Motor) Arduino" Simulink block, and the response of the system was measured and recorded. The leading coefficient α was determined by fitting a 2nd order polynomial to the portion of the position curve that occurred during the $3V$ pulse. The friction value at low speed $b_m(0^-)$ was taken from the measured values in week one. Equation (11) was used to calculate the motor moment of inertia, J_m.

The plant interface block was also used to record the motors response to a variety of inputs, including a ramp and sin wave. The input voltages and displacements of these inputs were recorded and compared to the expected results.

Week 3

The goal of week three was to implement and test simple open and closed loop control systems using the model of motor behavior that was developed previously in weeks one and two of the lab. This was accomplished by creating a simple open loop controller and comparing its behavior to that of "Bang/Bang" closed loop control.

The open loop controller was constructed using a one dimensional lookup table to calculate the appropriate voltage to supply to the motor given a desired velocity. This was calculated using previously measured motor speeds at a variety of voltage inputs. The controller accounted for the non linear behavior of motor friction by using motor speed and voltage data measured over a range of voltages and speeds. The lookup table calculated appropriate input voltages by interpolating between measured points. The "Plant (DC Motor) Arduino" shown in Fig. 10 was used to input the desired voltage through the Arduino and hardware shield and measure the position output of the block. The "Estimate Velocity" block shown in Fig. 11 was used to approximate the velocity of the motor given the angular position. The Simulink model of this controller can be seen in Fig. 12.

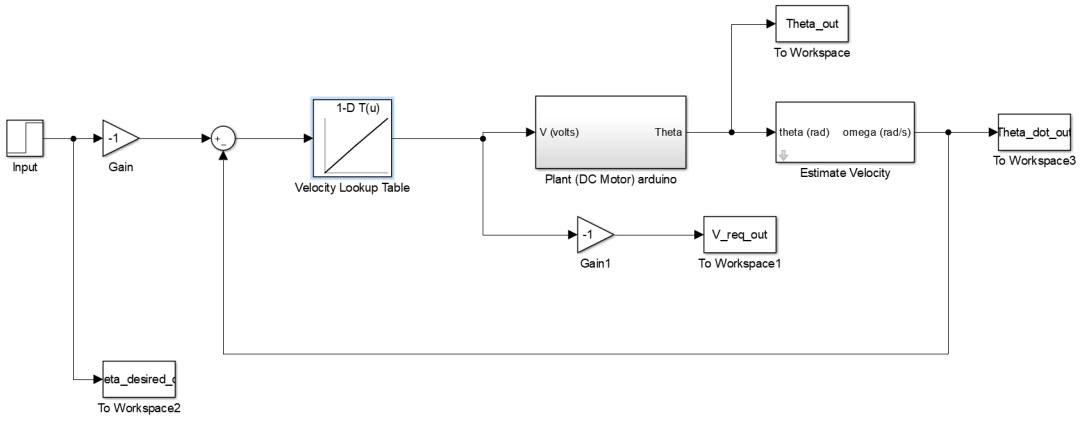

Figure 12: Simulink Open Loop Velocity Controller.

The open loop controller was then commanded to produce a velocity step from 20 rad/s to 40 rad/s after 5 seconds. The motors desired speed, actual speed, and input voltage were recorded. This process was then repeated while the motor was subjected to a torque load in the form of the eddy brake set to its halfway-in position.

Next, a bang-bang closed loop controller was constructed and tested. This control scheme was programmed to apply 5V to the motor whenever the angular velocity of the motor dropped below the target velocity, and apply 0V when the motor speed exceeded the target velocity. This was accomplished using a simple "if" statement. The "Plant (DC Motor) Arduino" and "Estimate Velocity" blocks, shown in Fig. 10 and Fig. 11 respectively, were used in the bang bang controller similarly to the open loop controller. A Simulink model of the controller can be seen in Fig. 13.

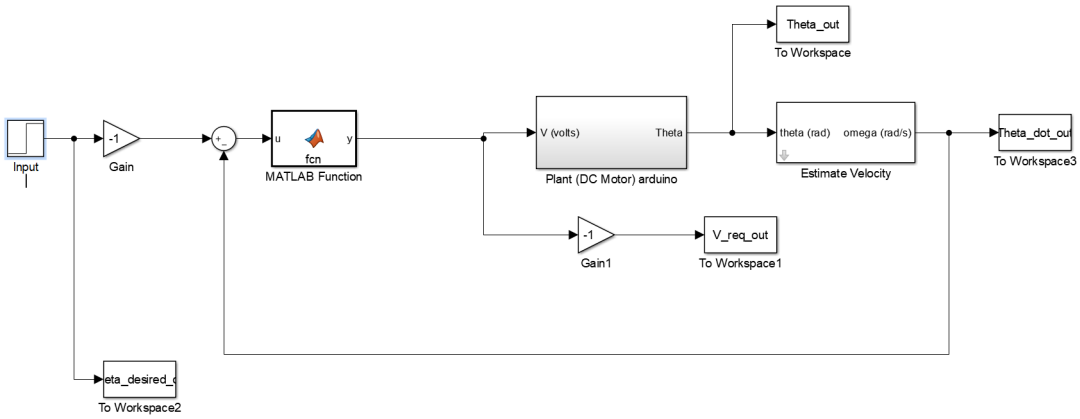

Figure 13: Simulink Bang-bang Velocity Controller.

The bang-bang closed loop controller was then subjected to the same 20-40 rad/s step function as the open loop controller. The process was repeated with the eddy brake halfway in. The resulting behavior of the motor and controller was recorded and compared to the open loop controller.

Next, a version of the bang bang controller which implemented a hysteresis band of +/- 2 rad/s was implemented. The "Plant (DC Motor) Arduino" and "Estimate Velocity" Simulink blocks were again used to interface with the Adruino, motor, and encoder hardware. The hysteresis band essentially adds a buffer space around the desired speed where the voltage input stays constant. In other words, if the desired speed is 20 rad/s, instead of switching between on and off rapidly at that exact speed, the voltage will switch on at 18 rad/s and switch off at 22 rad/s. This results in a larger range of oscillation around the desired speed, but allows for a more steady voltage input. This was modeled in Simulink using a if statement and a memory block to recall the last voltage setting. This controller Simulink model can be seen in Fig. 14 below.

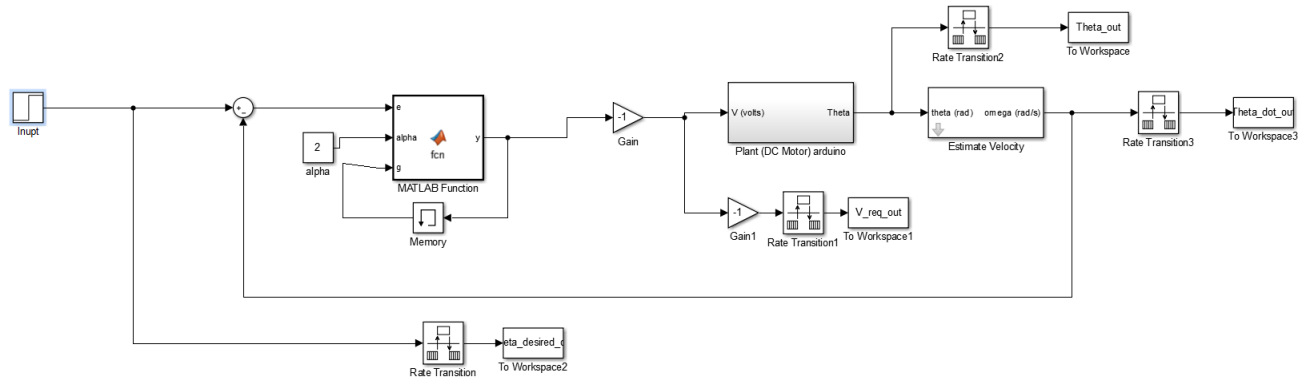

Figure 14: Simulink Bang-bang Velocity Controller With Hysteresis Band.

Again, the bang-bang controller with the hysteresis band was subjected to a desired input of in the form of a step function from 20-40 rad/s. The desired speed, actual speed, and input voltage were again recorded.

Week 4

The goal of week 4 was to implement a PID controller to control the position and speed of the DC motor. A PID controller Simulink model was constructed using the provided "Plant (DC Motor) Arduino" shown in Fig. 10, which input an appropriate voltage to the motor through the Arduino, shield, and Kepco amplifier, and interfaced with the encoder

to measure the position of the motor. The "Estimate Velocity" block shown in Fig. 11 used a pseudo derivative technique to estimate and record the motors speed.

In the PID controller block, the motor position was set as the input, as opposed to velocity control which was used in week 3. The calculated the position state error by subtracting the actual position from the desired position. Then the controller calculated took the derivative and integral of the error signal. The error, error derivative and error integral were multiplied by proportional, integral, and derivative gains respectively (K_P, K_I, K_D). These components were then summed, and used as a voltage input signal. The PID controller Simulink block is shown below in Fig. 15.

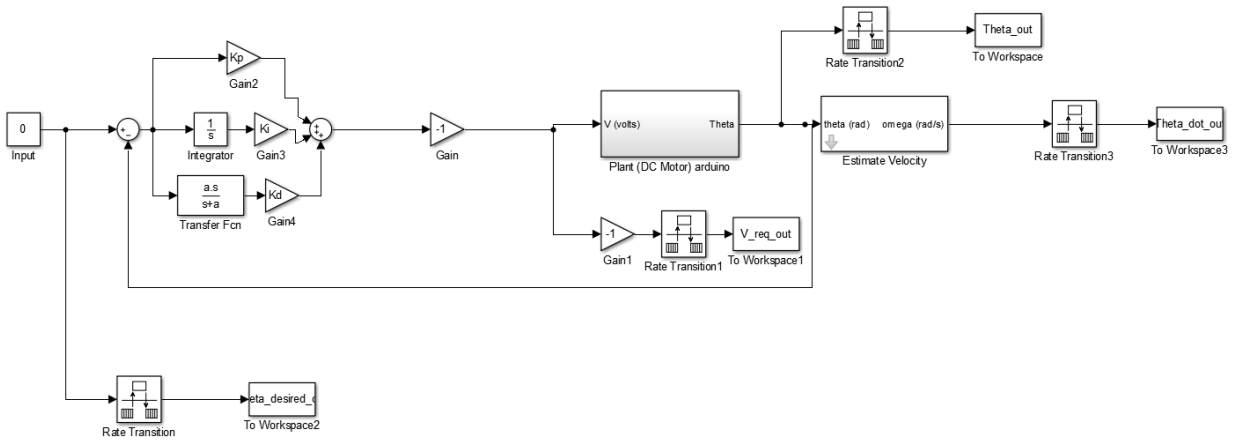

Figure 15: Simulink PID Position Controller.

The effects of the proportional, integral, and derivative components were examined first. To test each component individually, the gain of the component under test was set to one while the other two gains were set to zero. For example, to test the proportional component, K_P was set to 1 while K_I and K_D were set to zero. The proportional and integral control together were also tested. For each test, the desired position was set to zero, and disturbances were applied by moving the motor by hand. The desired position, actual position, motor speed, and input voltage was recorded in each test.

Next, all elements of the PID controller were tested together. To do this, the input was instead set to a step function. The initial position was set to 3.14 rad, or one rotation, and the initial desired position was also set to 3.14 rad. At T=5 seconds, the desired position changed to 0. This was done to simulate a consistent "disturbance" in order to best compare the various gain combinations tested. The gain combinations tested are shown below in Table 1.

Table 1: Experimentally Determined Motor Parameters.

PID Test #	K_P	K_I	K_D
1	1	1	1
2	5	1	1
3	5	1	5
3	5	5	5
4	5	0.5	2

The desired position, actual position, motor speeds, and input voltages were recorded for each combination of gains tested.

In the primary PID controller, a pseudo derivative block was used because a derivative block on its own is extremely susceptible to noise corruption. Noise typically occurs at high frequencies, and high frequencies are amplified by time derivatives. To get around this problem, a low pass filter transfer function in the form of $\frac{a}{s+a}$ was applied to the derivative block, resulting in a pseudo derivative transfer function of the form $\frac{as}{s+a}$.

A "naive" version of the PID controller was tested by using a standard derivative block instead of the pseudo derivative transfer function used in the original PID controller. The gains were all set to 1, and the desired position, actual position, speed, and input voltage of the "naive" PID controller were all tested and compared to the pseudo derivative PID controller.

V. Discussion of Results

Week 1

Before the experiment was preformed, the voltage drop across a known resistor was confirmed to be 0.05Ω.

The motor parameters R_m, L_a, K_V, K_T were successfully determined, and are listen in Table 2 below.

Table 2: Experimentally Determined Motor Parameters.

Physical Parameter	**Value**
R_m	1.468Ω
L_a	$0.0112\ henries$
K_V	$0.0821\ V * s$
K_T	$0.0821\ \frac{N*m}{amp}$

These parameters along with the electrical and mechanical equations of motion, Eq. (4) and (2) respectively, were used to develop a Simulink model of the DC motor system. This model was used to predict the DC motors response to various inputs as well as various control schemes later in the lab. The Simulink motor model is shown below in Fig. 16.

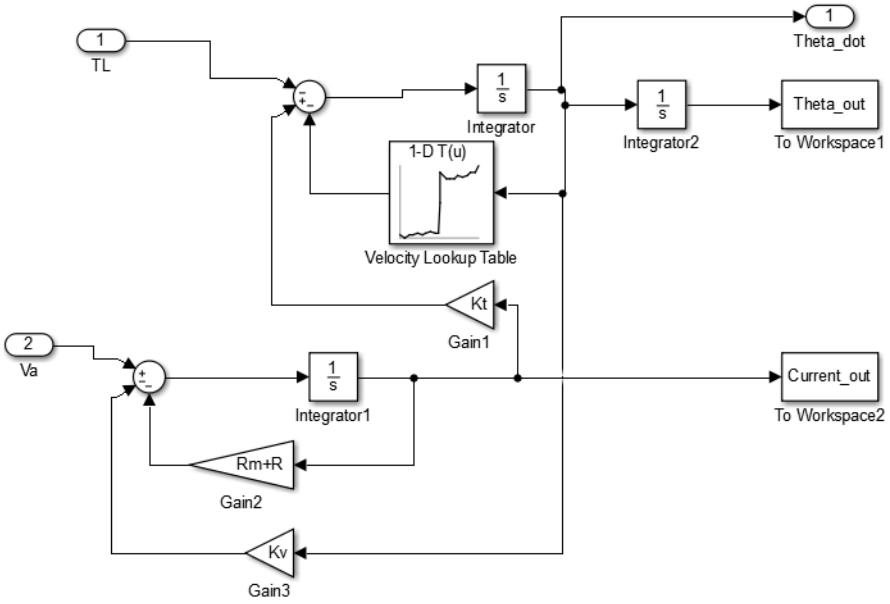

Figure 16: Simulink Model of the DC Motor System.

The motor friction was also characterized as a function of motor speed. This friction data was collected for use in designing and running an open control system later in the lab. The friction data as a function of speed is shown below in Fig. 17.

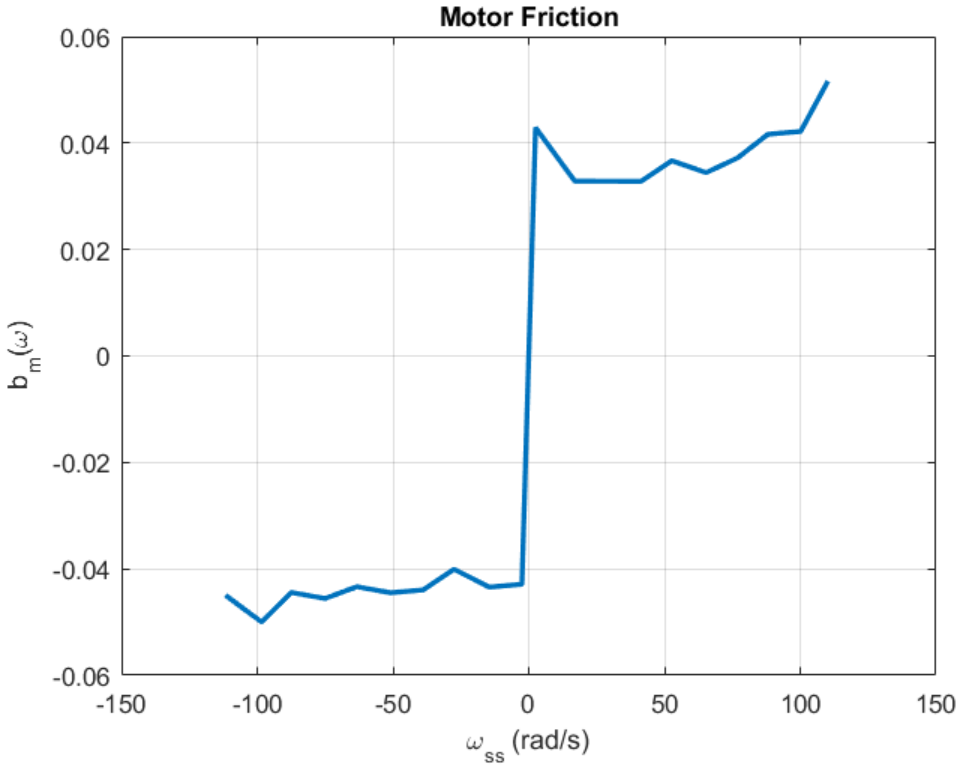

Figure 17: Motor Friction b_m as a Function of Steady State Speed ω_{ss}.

According to Fig. 17, the motor experienced higher friction when moving in the positive direction than in the negative direction. This measured outcome fit the expected result. Since the motors are most often run in the positive direction, the friction associated with positive movement was lower than friction experienced when spun in the negative direction.

Week 2

First, the encoder was connected directly to the Arduino which counted pulses and record the position of the motor. The "IncrementalRotaryEncoder" Simulink block shown in Fig. 8 was used to interface between the Arduino and the encoder and record the position of the motor. A voltage was applied to the motor as the position of the motor was recorded, and the voltage was increased steadily. The results of this test are shown below in Fig. 18.

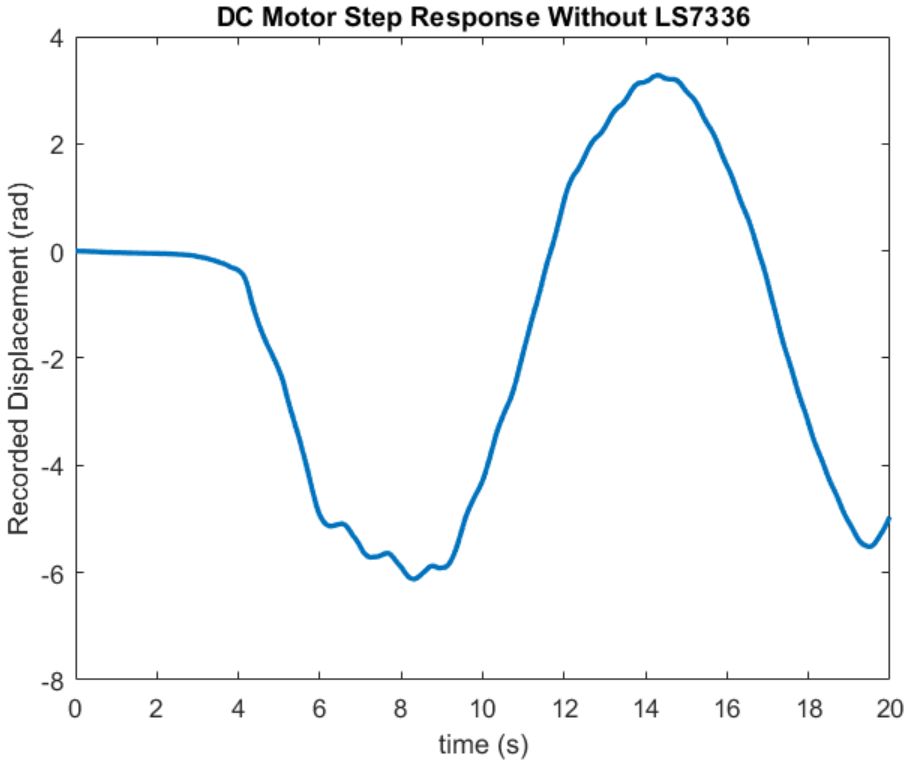

Figure 18: Encoder Output as the Voltage Was Increased.

As the input voltage increased, the position recorded became inaccurate around 6 seconds into the test. Although the motor spun only in one direction, the recorded position oscillated in a sinusoidal fashion. This test made it clear that it was necessary to offload the encoder computations to a dedicated processor.

Next, the LS7366 processor was tested by viewing the encoder count through the Arduino serial monitor as the motor was spun. The "LS7366SingleEncoder.ino" sketch [1] was pushed to the Arduino using and the motor was spun quickly by hand. Figure 19 below shows a screen shot of the Arduino serial output.

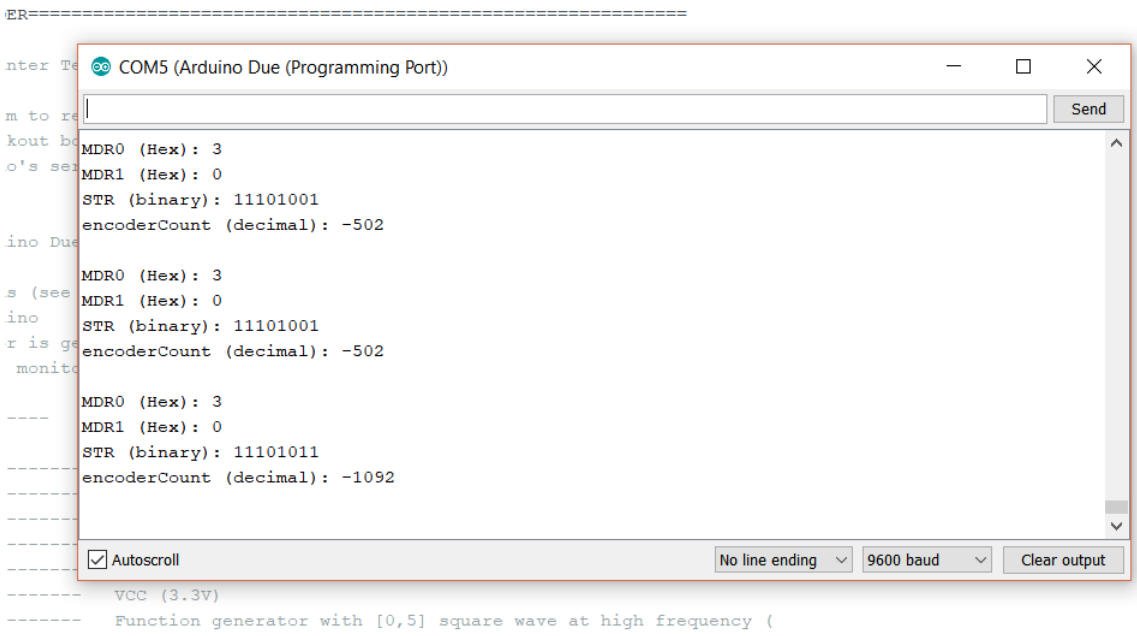

Figure 19: Encoder Output Viewed Through the Arduino Serial Monitor.

The output of the encoder was confirmed to be accurate by viewing the position through the serial output. The "Plant (DC Motor) Arduino" Simulink block [1], shown in Fig. 20 was used from this point forward to interface with the rotary apply a desired input voltage to the DC motor and record the motor position using the LS7366 processor. A series of voltage signals were input to the motor, and the position response of the motor was recorded.

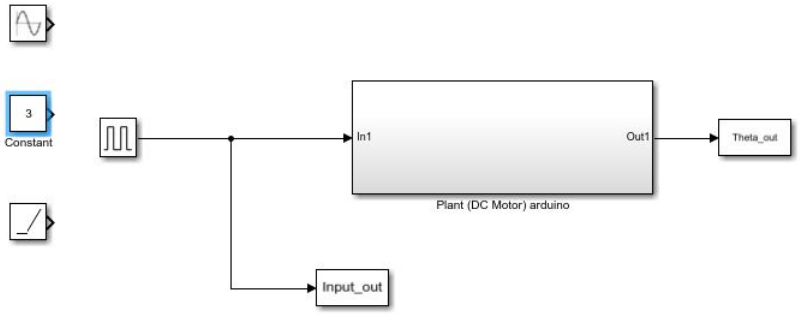

Figure 20: Plant (DC Motor) Arduino Simulink Block. [1]

The first input voltage tested was a step function with a magnitude of 3V and a duration of 0.25 seconds. The full position and time curve is shown below in Fig. 21. By curve fitting the portion of the θ(t) curve, corresponding to the duration of the pulse, a

second-order polynomial was found. This portion of the curve along with the curve fit to it is shown in Fig. 22. Using the first coefficient of the polynomial, α, and assuming that the angular acceleration was constant, $\ddot{\theta}_{ss}$ was found to be $80.2\frac{rad}{sec^2}$.

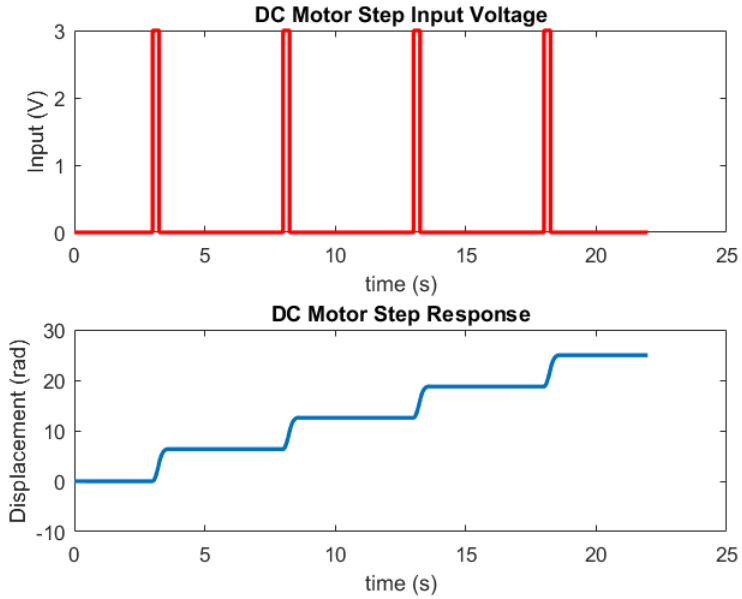

Figure 21: Voltage Input to the Motor and the Step Response.

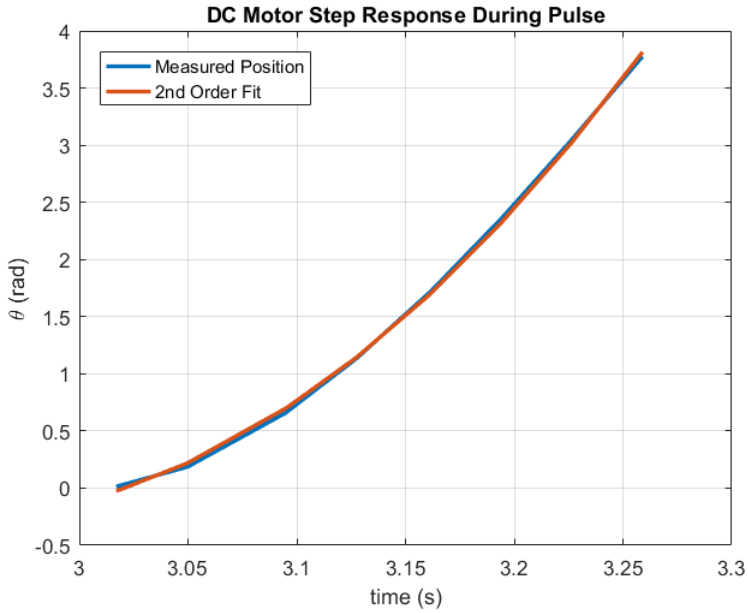

Figure 22: Best Fit Curve of the Motor Pulse Response.

The friction value at low speed, $b_m(0^-)$ was inferred from the measured friction data shown in Fig. 17. $b_m(0^-)$ was determined to be 0.03 by finding the y-intercept approaching from $-\omega$. Using α, $b_m(0^-)$, and Eq. (11), the moment of inertia was found to be 0.0018 kg m^2.

Using the "Plant (DC Motor) Arduino" Simulink block shown in Fig. 20, the plant was subjected to ramp and sinusoidal input voltages. Figures 23 and 24 show the results of the plant being subjected to a ramp and sine wave, respectively.

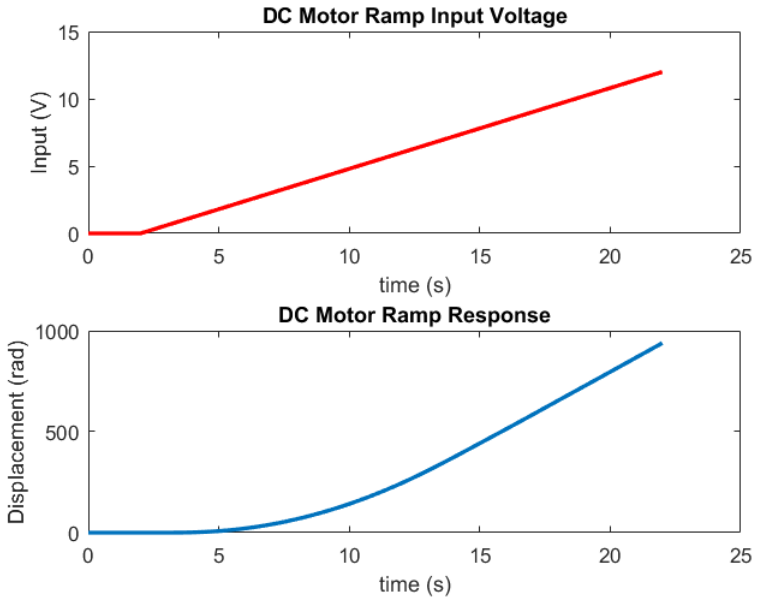

Figure 23: Voltage Input to the Motor and the Ramp Response.

subjected to a linear ramp voltage, the motors position increased in a quadratic fashion, which fit the expected results. The voltage is proportional to the motor speed, so as the voltage increased linearly the motor speed increased linearly as well.

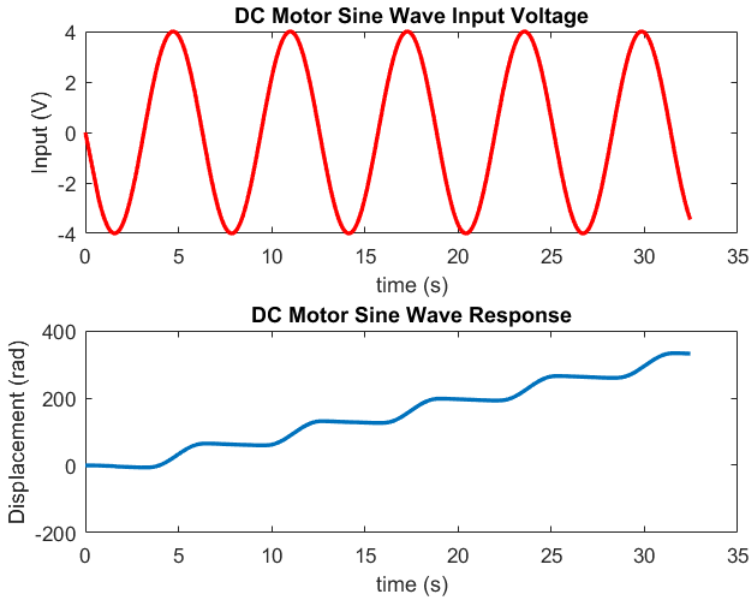

Figure 24: Voltage Input to the Motor and the Sine Wave Response.

When the motor was subjected to a sinusoidal input voltage, the motor was expected to oscillate around the starting position given equal friction in each direction. However, due to uneven friction in the two directions, the motor spun considerably faster in the positive direction than the negative direction, which led to an increasing overall position. This result matches the expected result deduced from the measured friction values shown in Fig. 17.

Week 3

Prior to week three, the open loop and bang-bang control schemes were simulated. The simulation used the Simulink DC motor model shown in Fig. 16, which employed the physical parameter data determined in weeks one and two. The control schemes were each simulated with and without the brake applied. Since the exact force output from the brake was unknown, the torque exerted by the eddie brake was assumed to be 0.5 Nm. The desired speed input was set to a step function, beginning at 20 rad/s and ending at 40 rad/s. The simulation results are shown below in Fig. 25.

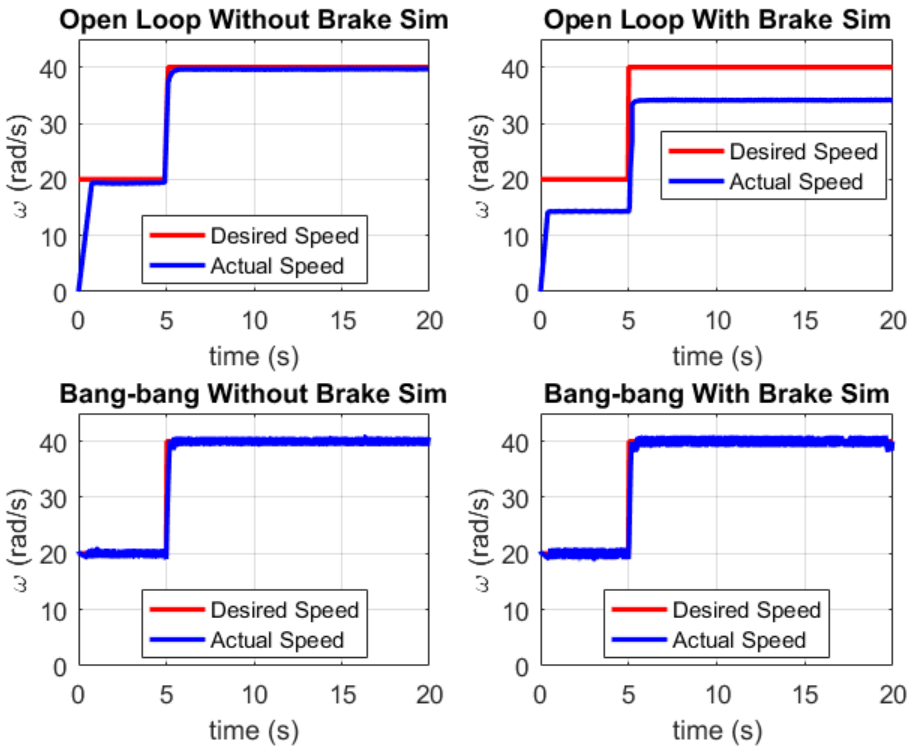

Figure 25: Simulated Open Loop and Bang-bang Controlled Motor Responses.

According to the simulation, the actual speeds came close to matching the desired speeds in the open loop simulation without the brake, but were slightly lower. These slight differences were likely due to linear interpolation errors, since the open loop controller selected the input voltage based on a linear interpolation between measured points. The speeds reached by the open loop controller with the brake were significantly lower than the input voltage, which matched the expected results. The bang-bang controller reached the desired speeds in both cases, and oscillated around the desired speeds at very high frequencies. In the simulation, there were no hardware limitations in place to dictate how frequently the voltage could switch from on to off. It was expected, and later confirmed, that the voltage would switch states slower in the physical test, which would result in a larger oscillation amplitude.

The first control scheme tested in week 3 was the open loop controller. The Simulink model of the open loop controller is shown in Fig. 12. The desired speed input was set to an initial value of 20 rad/s, then increased to 40 rad/s after 5 seconds of run time. The first test was run with the brake all the way out, so there was no resisting load on the motor. The desired motor speed, actual motor speed, and voltage input for the first test is shown below in Fig. 26.

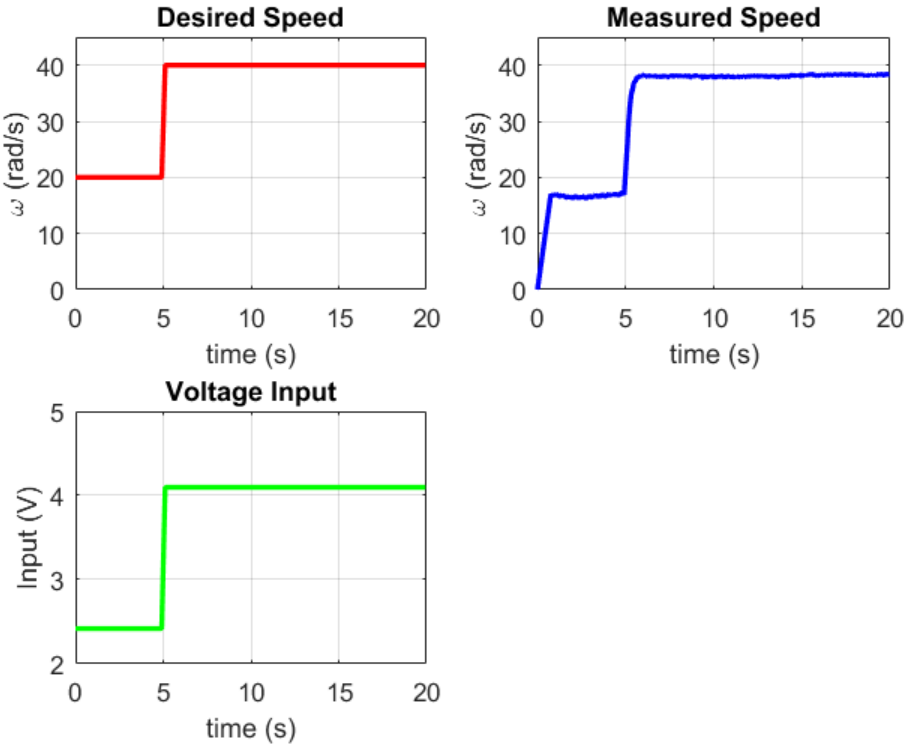

Figure 26: Open Loop Controller Without Brake.

The open loop controller performed fairly well with no applied load, although the motor speeds achieved were slightly lower than the desired speeds in both cases. These slightly lower speeds suggest that the voltage supplied to reach a requested speed was consistently too low for the speed ranges requested. This seems like a likely outcome based on the fact that the friction of the motor is not a linear function of speed, which means that the voltage required as a function of desired speed is not a linear function either. Since the lookup table used to determine the voltage input interpolated between measured points in a linear fashion, slight errors in the input voltages were expected. The input voltage stayed at a constant value for each speed, which matched the expected result.

Next, a similar test was run using the same open loop controller and desired speed input, but this time a resisting torque was applied to the motor. The eddie brake was moved in to the halfway position. The desired speed, measured speed, and voltage input measurements are shown below in Fig. 27.

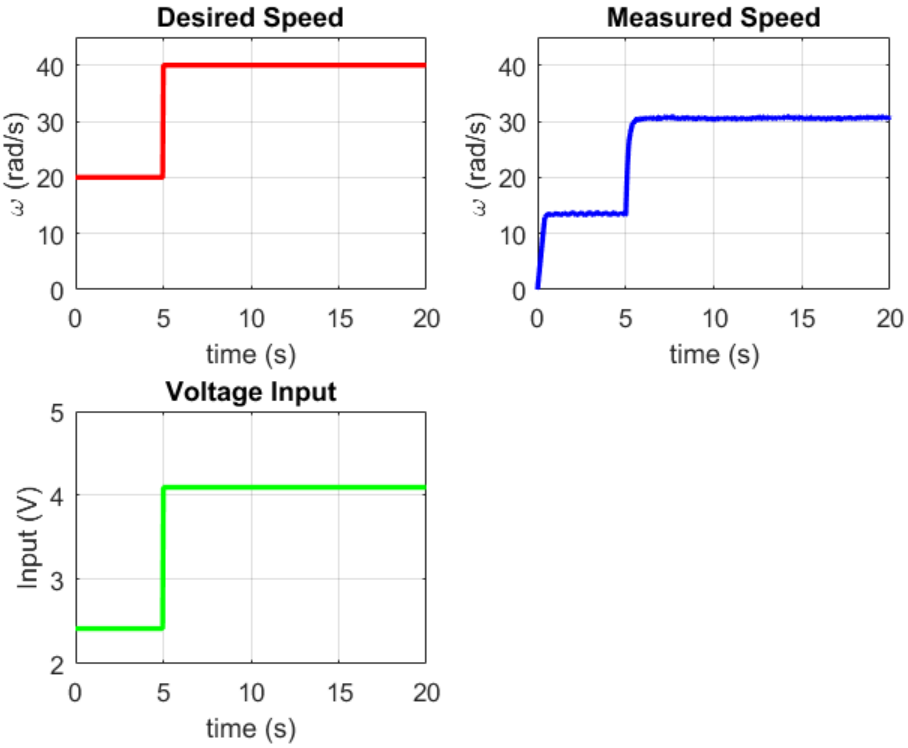

Figure 27: Open Loop Controller With Brake at Halfway Position.

The motor reached much lower final speeds with the brake applied. This fit the expected result because without open loop feedback, nothing signaled the motor to supply a higher input voltage. The voltage supplied to the motor was identical in the test with the brake and without the brake.

The biggest drawback to open loop controllers is that they lack the ability to respond to unaccounted for disturbances. The open loop controller was able to over come the friction of the motor fairly well, since that disturbance was accounted for and measured. However, without feedback, it was unable to respond to an additional resistance load, which resulted in a speed that was much lower than the desired speed. The benefit of an open loop control scheme is that it is simple. To run the motor, no active sensing or feedback was necessary. The drawback is that open loop controllers are not robust.

Next, the bang-bang closed loop controller was tested. The model of the bang-bang controller is shown in Fig. 13. The same desired speeds were input as in previous tests, and the brake was removed. The results of this test are shown below in Fig. 28.

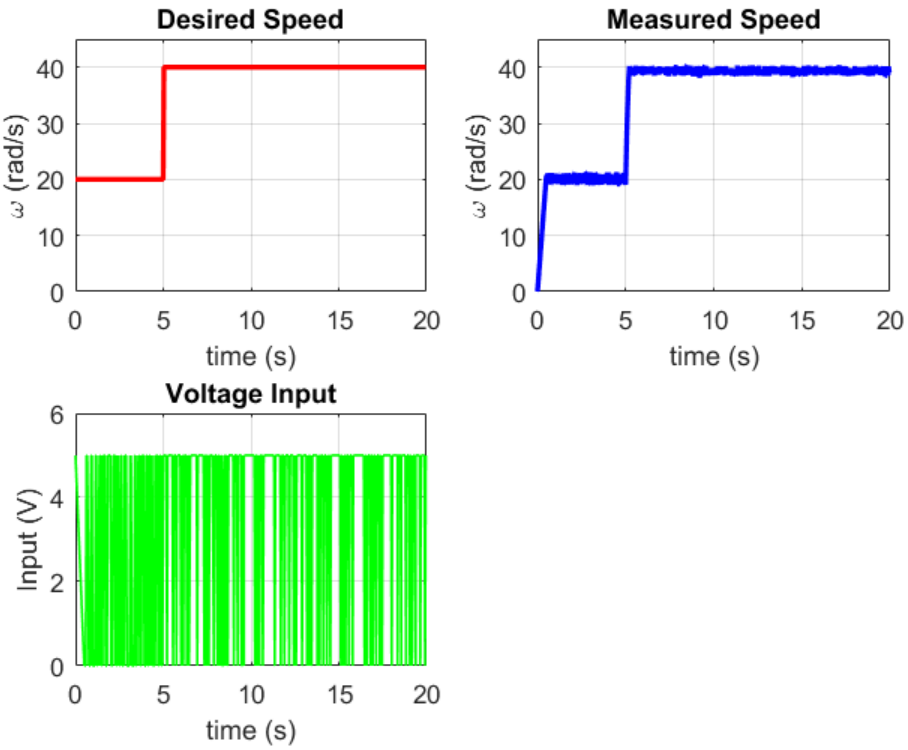

Figure 28: Bang-bang Controller Without Brake.

The bang-bang controller on average achieved the desired speeds. More precisely, the motor oscillated between about ± 1 rad/s above and below the desired speeds. This fit the expected result, since the motor only had two voltage input states it was constantly either accelerating or decelerating. This quality is clear in the voltage measurement, which jumped between 0 and 6 volts frequently during the test. Overall, with no brake the bang-bang controller reached the desired speeds better than the open loop controller.

Next, the test above was repeated with the bang-bang controller, except this time the brake was situated in the halfway position. The results of the bang-bang controller with a resisting torque are shown below in Fig. 29.

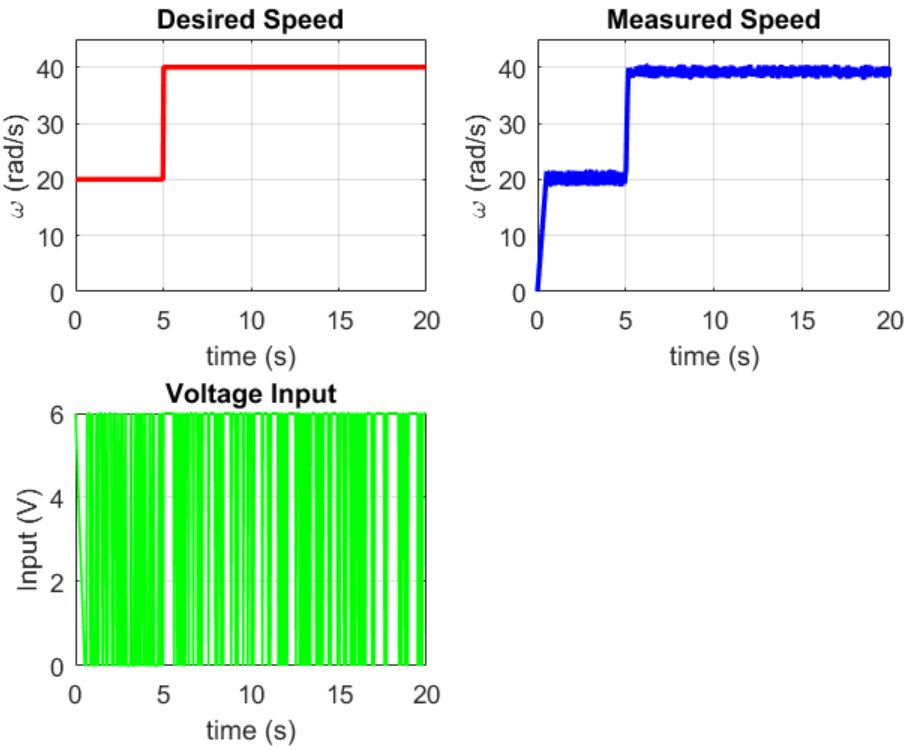

Figure 29: Bang-bang Controller With Brake at Halfway Position.

The first time this test was run, it was found that the voltage input maximum of 5V was not enough to reach the full 40 rad/s. To fix this, the maximum voltage was raised to 6V. With the brake applied and the maximum voltage raised, it was found that the motor speed matched the desired speed just as well as it did with no resistance. This matched the expected result due to the fact that the motor was able to use feedback to increase the amount of time voltage was supplied to the motor and reach the desired speeds. The motor oscillated around the desired speeds with the same amplitude as the previous test.

The biggest advantage to the bang-bang closed loop controller was that it was able to react to disturbances. In both the tests with the brake and without, the bang-bang controller matched the desired speed better than the open loop controller. The bang-bang controller also required no data pertaining internal resistance, such as friction data.

The largest downside of bang-bang controllers is the fact that they require sensors, which increases the complexity and cost of the system. The DC motor required the use of not only the optical encoder, but also the estimated velocity obtained from position data. This velocity estimation represented the largest possible source of error for the system, since accurately calculating the derivative of a position signal signal using only data from

previous time measurements is not an easy task. Another downside of the bang-bang controller is that the voltage is constantly being switched on and off when the motor speed is close to the desired speed. This results in oscillation around the desired speed, rather than maintaining the desired velocity with a constant voltage. With a bang-bang controller, the only way to bring the actual speed closer to the desired speed is to increase the frequency which the voltage input shuts on and off. Realistically, the voltage input can only oscillate so fast, which limits how close to the desired speed the motor can get. Constant oscillation of the input voltage is also not efficient, and may be damaging to some systems.

Next, a hysteresis band was implemented into the bang bang control scheme. The width of the hysteresis band was set to ±2 rad/s, and the same desired speed input that starts at 20 rad/s and steps up to 40 rad/s was input to the controller. This created a range from 18-22 rad/s where the input voltage would stay constant. The desired speed, measured speed, and voltage input is shown below in Fig. 30.

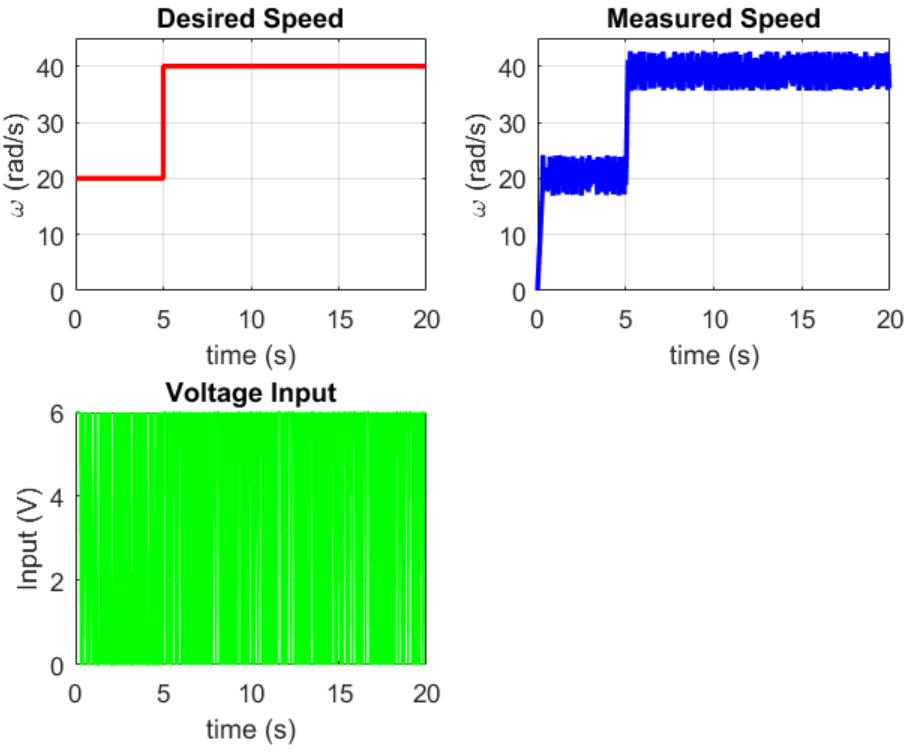

Figure 30: Bang-bang Controller With Hysteresis Without Brake.

The average measured speed still matched the desired speed. The measured speed oscillated around the desired speeds with a higher amplitude than they did without the hysteresis

band, which was expected. The oscillation amplitude with the hysteresis band was about $\pm$ 3 rad/s. From Fig 30, its clear that the voltage still oscillates between 0 and 6 volts rapidly. However, Fig. 31 below shows a close up view of the voltage over a brief interval.

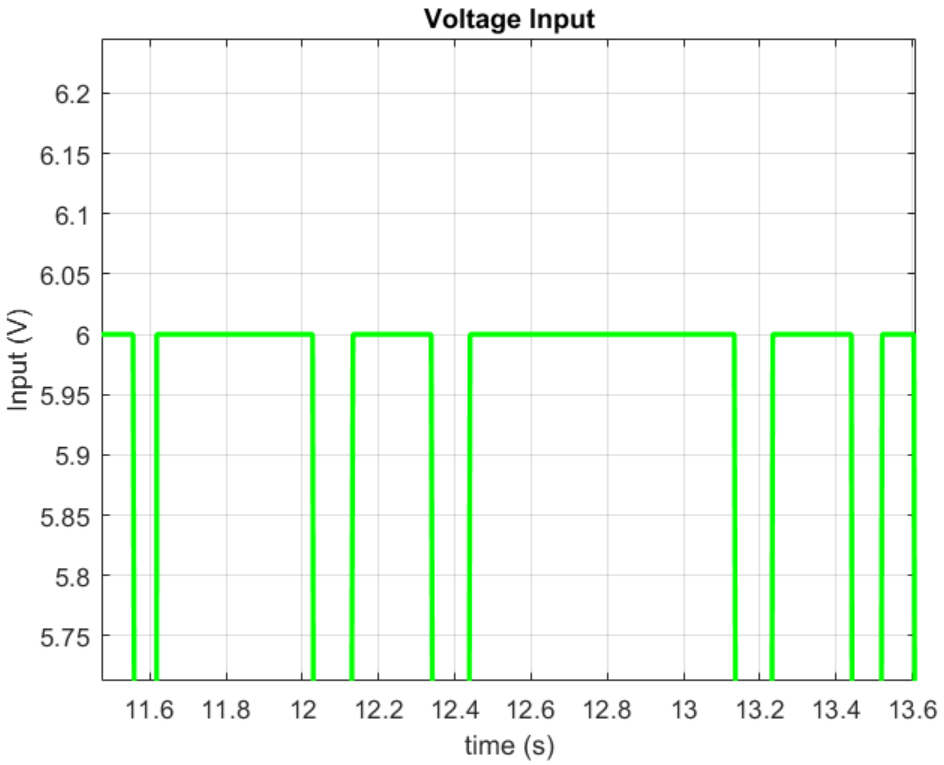

Figure 31: Hysteresis Band Effect on the Input Voltage.

Figure 31 provides a clearer view of how the input voltage changed over a brief period of time with the addition of the hysteresis band. It shows that the voltage stayed constant for up to half a second at a time. Compared to the voltage without the hysteresis band, the voltage stayed constant for much longer periods of time than it did without the addition of hysteresis control. The hysteresis band was seen to be an effective method of ADD/MOVE

Finally, the bang-bang controller with the hysteresis band was tested again with the same input, except now the eddie brake was moved halfway in to add a resisting torque. The desired speed, measured speed, and voltage input plots are shown below in Fig. 32.

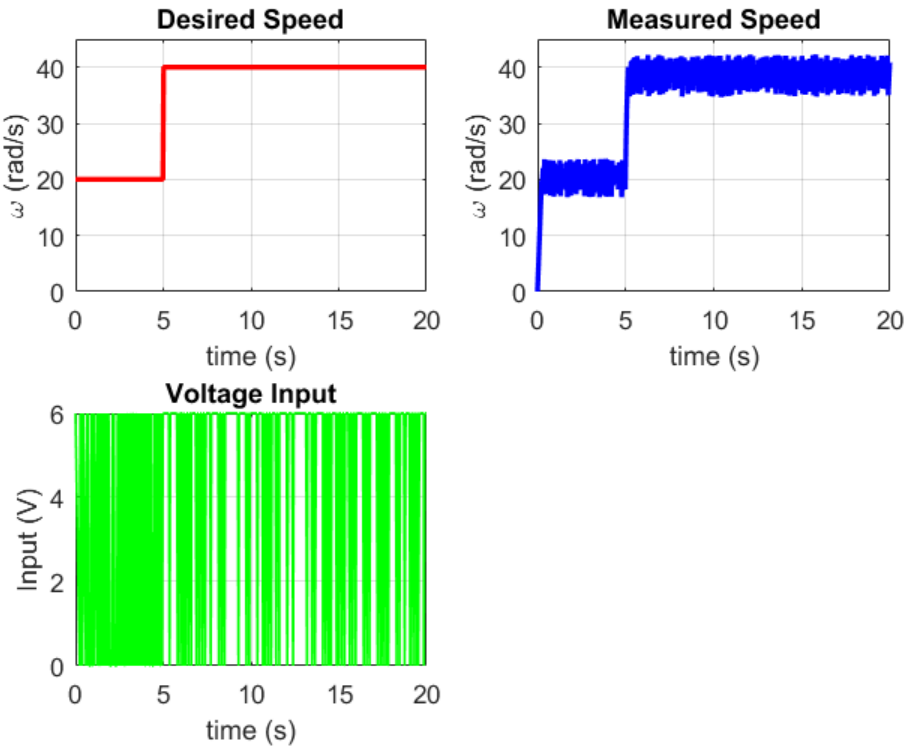

Figure 32: Bang-bang Controller With Hysteresis and Brake at Halfway Position.

Even with the addition of the eddie brake, the the bang-bang controller with the hysteresis band maintained an average speed that was equal to the desired speed. The motor oscillated around the average value with about the same amplitude as it did without the brake. However, stretches of time that the voltage input stayed constant were much longer with resistance. This is due to the fact that with the brake in place, the motor decelerated at a much greater rate than it accelerated. The brake assisted the motor while when decelerating and resisted the motor while accelerating. This relatively slow acceleration resulted in long enough periods of time in between voltage change that the gaps are visible without zooming into the voltage plot in Fig. 32.

The hysteresis controller was seen to effectively decrease the frequency which the voltage input switched from on to off. This effect is practical in many application where higher efficiency is desired. However, while the average speed of the motor didn't change, the hysteresis band also increased the rate of oscillation. The hysteresis band effectively increased the percent overshoot of the controller.

Week 4

The goal of week four was to design position controllers for the motor using a proportional, integral, derivative controller (PID). The behavior of each element of the controller was first tested individually. This was achieved by setting controlling the PID gains such that there was only one non zero gain at a time. Henceforth the control coefficients corresponding to the proportional, integral, and derivative gains will be referenced in vector form as $[K_P, K_I, K_D]$.

The first control system tested was a simple proportional controller of the form [1,0,0]. This system was instructed to hold the motor's position at zero radians. An external torque was then applied to the motor by hand. The behavior of the motor can be seen in Fig. 33 below.

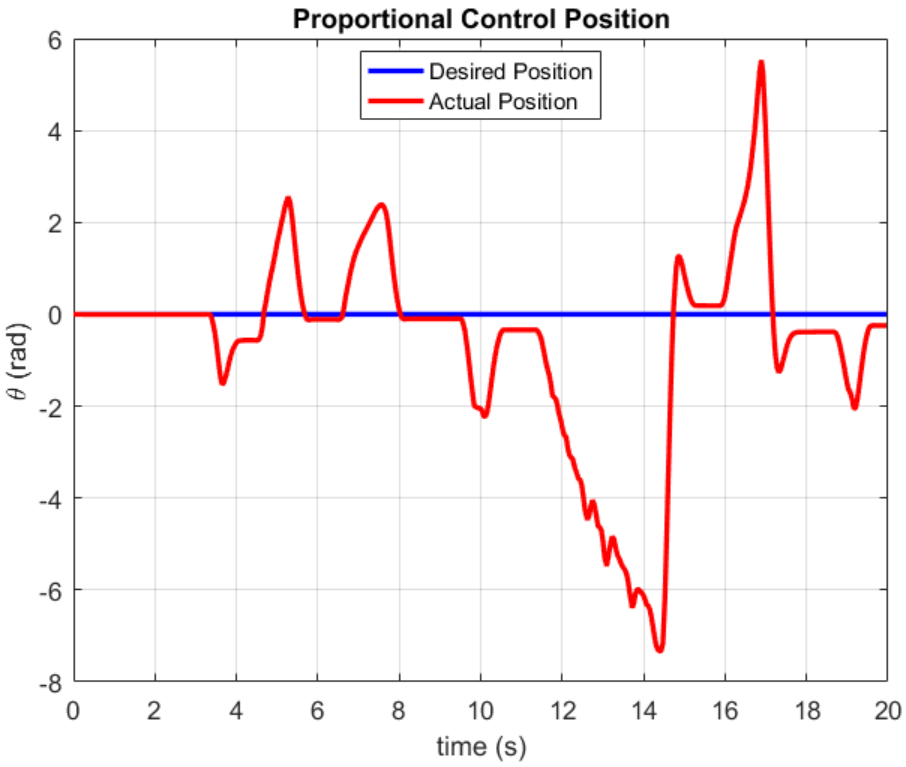

Figure 33: Proportional Controller Test.

It can be that the motor is able to accurately return to its initial position in a relatively efficient manner. It is important to note that the motor routinely overshoots its intended position when subjected to higher magnitude disturbances. This is a result of the large voltage applied to the motor which is proportionate to the distance the motor is from its

intended position. This result of this proportional signal is a voltage input which tapers to zero as the motor approaches its desired position. Due to the momentum of the physical system, the motors position does not stop at zero even though the voltage applied at this point is zero. This results in the tendency of the motor to overshoot the target. It should be noted that the motor returns to a non-zero position due to the steady state error of the proportional control system. It can be concluded that the proportional control system excels at quickly returning a system to its desired state regardless of overshoot and steady state error.

Next, the PID controller was configured to test the performance of the integral control coefficient. This control system can be denoted by the vector [0,1,0]. The above procedure for the testing of the proportional controller was repeated on this new configuration. The The results of this test are shown in Fig. 34.

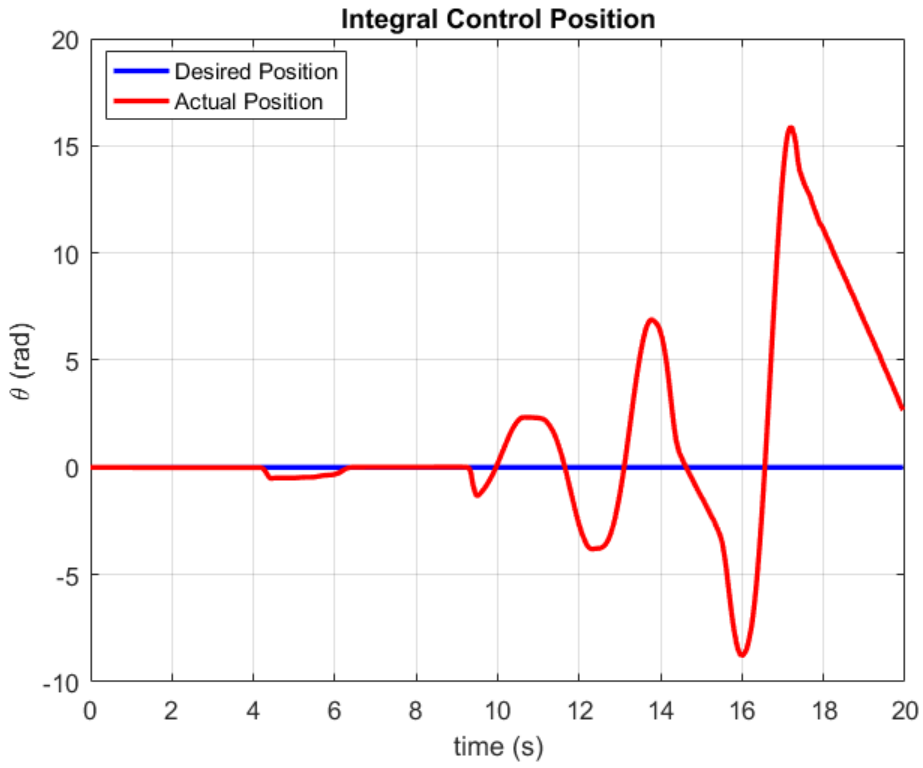

Figure 34: Integral Controller Test.

Upon slightly disturbing the motor, the controller began to send voltages to the motor which resulted in an unstable system in which the magnitude of the oscillation of motor position grew steadily with time. This is a result of the behavior of the integral controller.

The integral controller sends a voltage to the motor which is proportional to the integral of the error term over time. For this reason, the controller continues to send a voltage to the motor even after it has reached its desired position because the integral of the error continues to grow until this time. When the motor has reached its desired position, the integral of the error term reaches a local maximum. This causes the system to continue to accelerate past the target position only ceasing to apply voltage once the integral of the error reaches zero at some position on the opposite side that the motor started on. The momentum of the system causes the resultant error integral produced by the overshoot of the system to be greater than the initial cycle. This process repeats continuously overshooting the target more and more each time due to the inertia of the system. From this it can be concluded that if the integral control term is left unchecked, the system is likely to go unstable.

The derivative control coefficient was isolated and its behavior tested individually through the use of a controller of the form [0,0,1]. The results of this test can be seen in Fig. 35.

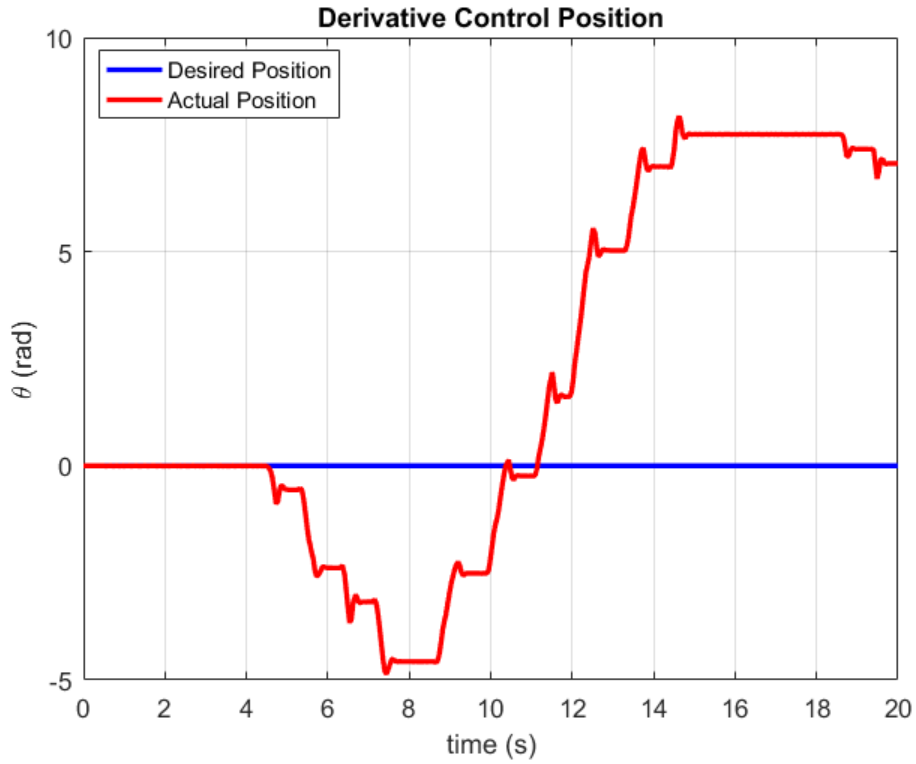

Figure 35: Derivative Controller Test.

It can be observed that the derivative controller was effective in holding the position of the motor constant at whatever position it was at. It can also be seen that the controller was

totally ineffective at achieving the specified position. This is due to the derivative controller's tendency to oppose change in motor position. The derivative of the error term can be thought of as the rate of change of motor position with respect to the target motor position. If the motor begins at rest, the derivative controller will not send a signal to the motor and therefore will not produce a change in the motor's position. This explains the motors tendency to hold a given position once it has settled. By examining a case in which the motor begins in a negative position with a positive velocity a useful discussion of the dampening effects of this coefficient can be had. In this case, the derivative of the error term is negative as the motor is approaching its desired position. The controller then scales the negative derivative and sends a negative voltage to the motor opposing its positive motion until the velocity of the system reaches zero. In the absence of other controller inputs, the analysis performed on this case of motor motion can be applied to other cases of divergent and static behavior with similar results. The conclusion that can be drawn in this case is that the derivative controller opposes change in velocity and tends to hold the motor in any given position.

Having tested each of the PID control coefficient's behavior individually, the next step was to test the behavior of the controller when used in conjunction with each other. In order to understand the behavior of a proportional integral controller the coefficients of the PID controller were set to [1, 1, 0]. The system was then subjected to an external disturbance as before. The motor's behavior can be observed in Fig. 36 below.

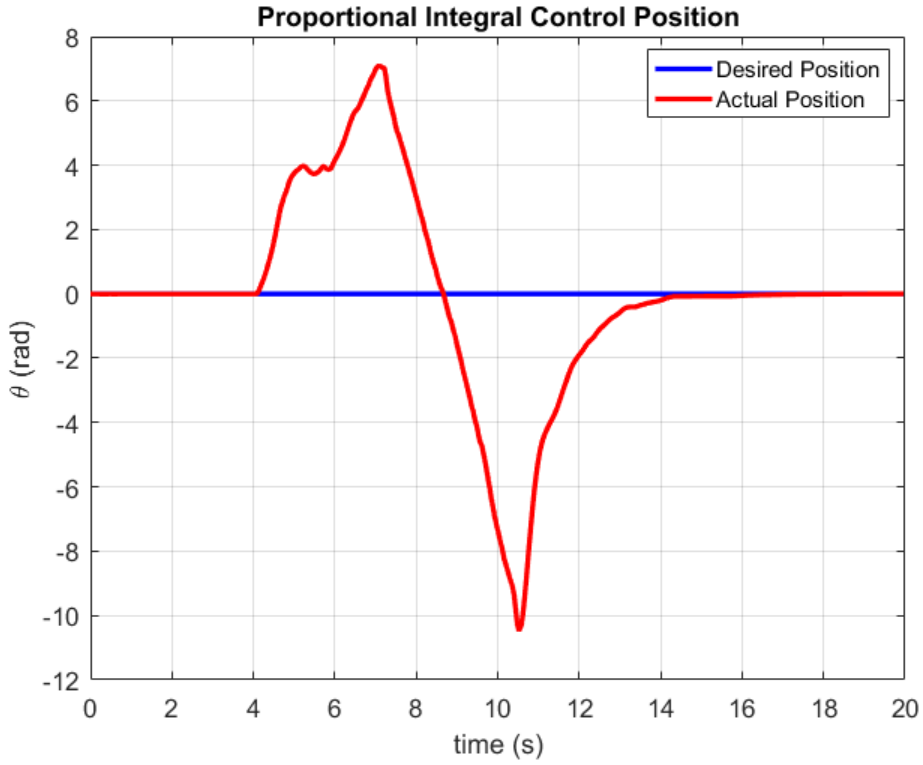

Figure 36: Proportional Integral Controller Test.

It can be seen that the motor wildly overshot its target in its first attempt to reach the zero position. On the motors second approach the position of the motor hyperbolically approached the zero position. This is a beautiful demonstration of the proportional integral signal at work. After being released from rest at approximately seven seconds, both the integral and proportional controllers were sending negative voltages to the motor resulting. This resulted in a rapid return to the zero position. At this point the proportional controller ceased sending negative voltages to the motor, yet the integral controller was still commanding the motor to move in the negative direction due to the large integral of error that had been built up to this point. This resulted in the immense overshoot that can be seen in the system. Eventually, as the integral of the error signal approached zero, the magnitude of the proportional control signal was enough to stop the motor and begin to bring it back towards the desired position. With a decreasing error integral and a decreasing proportional control signal the system was able to smoothly approach and settle at exactly the desired position where neither control element was commanding an input voltage.

The PID controller was then set up to control the motor through the use of all three control schemes by setting the proportional, integral, and derivative gain coefficients to one

resulting in a controller of the form [1,1,1]. In order to achieve a consistent standard by which the performance of different control schemes could be compared, the motor was commanded to follow a step input in position after two seconds of magnitude π radians. This corresponds one half of a revolution of the motor, a fairly modest displacement. This process was repeated using different control coefficient vectors. Five unique control schemes were tested. The performance of each configuration can be seen in Fig. 37 below.

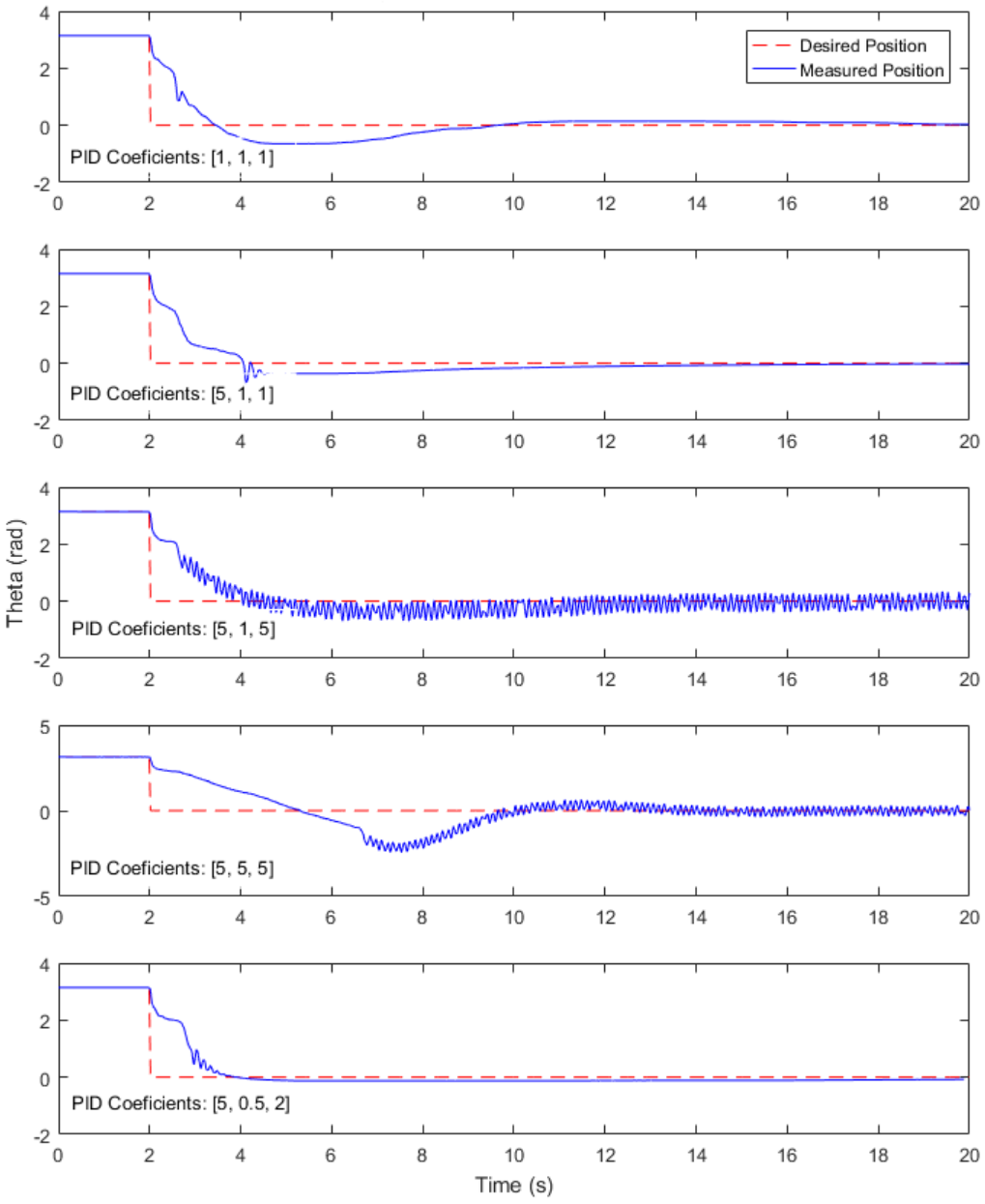

Figure 37: PID Controller Response to Step Input

Upon examining the performance of the simple ([1,1,1]) PID controller it can be seen that the controller tracked its desired position fairly adequately. The controller overshoots slightly towards either direction before settling at the desired position after about 20 seconds. By comparing this system's performance with that of the proportional integrating controller it can be seen that the controller overshoots its target far less while taking slightly longer to settle. This is due to the competing nature of the different signals. This is a result of the derivative controller seeking to impede the motors motion allowing for smaller overshoot, and less dramatic motor velocities.

The next control scheme tested was the [5,1,1] configuration in which the proportional control coefficient was set to dominate the PID control signal. In this case, it can be seen that the motor briefly moved rapidly towards the zero position. This was quickly counteracted by the growing derivative term due to the large angular velocity of the motor. At high speeds it can be seen the derivative coefficient dominated the other two resulting in a dramatic decrease in velocity of the system. Once the motor had been slowed, the proportional signal once again drove the motor rapidly towards the zero position. This behavior results in a somewhat bumpy motor trajectory towards the zero position due to the oscillatory behavior of the derivative term. After reaching the zero position, the integral controller continued to drive the motor past the desired position. This resulted in a derivative constrained overshoot and subsequent slow approach to the desired motor position as dictated by the proportional controller. The overshoot of this system indicates that the derivative control coefficient chosen for this scheme is too low.

The performance of the [5,1,5] controller was similar to that of the previous control scheme with one profound difference. The [5,1,5] controller can be seen to oscillate rapidly in position over the duration of the test. This is a result of the competing interests of the proportional controller and the derivative controller. The proportional controller commands the motor to move to the zero position. The motor begins to move, this results in a proportional opposing signal from the derivative controller due to the acquired velocity of the system. The sum of these two signals results in a voltage that changes rapidly from positive to negative depending on which term, the derivative or proportional, is dominating at any given time. Due to the motor's inertia, the motor is carried towards its destination and away from from it in alternating fashion due to its tendency to continue moving even after the applied voltage has changed. When this happens, both the proportional and the derivative terms work together to drive the system towards the goal position until once again the derivative controller puts a stop to the controller's progress. This oscillatory motion leads naturally to the conclusion that the derivative control gain is too high relative

to the other terms to lead to an ideal result.

Upon comparing the performance of the [5,5,5] controller to that of the [5,1,5] controller several differences can be noted. First, rapid motor oscillations were not introduced until after the first overshoot of the zero position. This is a result of the combined efforts of the integral and proportional control signals to dominate the derivative control term and not allow for any reverse motion of the the motor due to inertia during its first zero approach. Having past the zero mark, the integral controller and the proportional controller began to oppose each other, allowing for the derivative control input to dominate the position of the motor at times. The motor under these conditions followed closely with the behavior of the previous system and can be explained using a similar analysis. One further difference that should be noted, is the system's relatively tendency to overshoot its target. This is a result of the high voltages applied to the system in reaction to small disturbances. Large voltage inputs result in higher motor velocities. Due to the inertia of the system, higher motor velocities require more time to slow, hence, the tendency of the motor to overshoot its desired position.

Using the information that was gathered from the performance of the previous control schemes a [5, 0.5, 2] control scheme was implemented in the hopes of creating a system that settles quickly, with minimal overshoot, and little oscillation. This result was confirmed by the impressive performance demonstrated by the system. The system approached the zero point rapidly with minimal overshoot and very slight steady state error. The behavior of this system warrants an in depth discussion of the reasoning behind its performance. The motor began out of position with zero velocity. In this case the relatively large magnitude of the proportional controller caused the system to accelerate rapidly towards the zero point. After acquiring a finite velocity the derivative term began to oppose this motion and eventually was able to stop the motion of the motor. Having stopped, the derivative controller again made negligible contributions to the resultant signal. This allowed the proportional and integral signals to once again dominate the signal that was sent to the motor and in effect the motors position. This process was able to repeat several times with a smaller proportional control signal due to the motors proximity to the desired position. This process can be seen in the section of high frequency positional oscillation at a time of approximately three seconds before approaching zero in a controlled manner with a balance of the two relevant control parameters. Due to the small but finite contribution of the integral controller, the system can be seen to overshoot its desired position slightly due to the tendency of an integral controller to continue to provide a signal even as the motor passes its desired position. The motor then settled slowly to zero

over the next twenty seconds as the integral of error was able to build to a point where the integral term could overcome the impeding effect of the derivative controller.

While the [5,0.5,2] controller was able to perform remarkably well, the system could be further optimized using techniques such as root locus and through the use of tools like Sisotool. These methodologies would allow the user to define certain design goals such as a settling time, and an allowable percent overshoot. The output of these calculations would give the range of possible control coefficients which satisfy the requirements and result in a stable system. This would allow for further optimization of the system towards an ideal control scheme for the application with minimal overshoot, and rapid settling.

Previously, the Simulink model of the PID controller was implemented using a pseudo-derivative block as opposed to a strict derivative in order to lessen signal noise through the implementation of a low pass filter. Now in order to quantify the impact of the psuedo-derivative on the system it was necessary to test the response of the PID controller using the derivative block as provided by Simulink. With coefficients of [1,1,1] the response of the motor to a step input under control of this naive control system can be seen in Fig. 38 below.

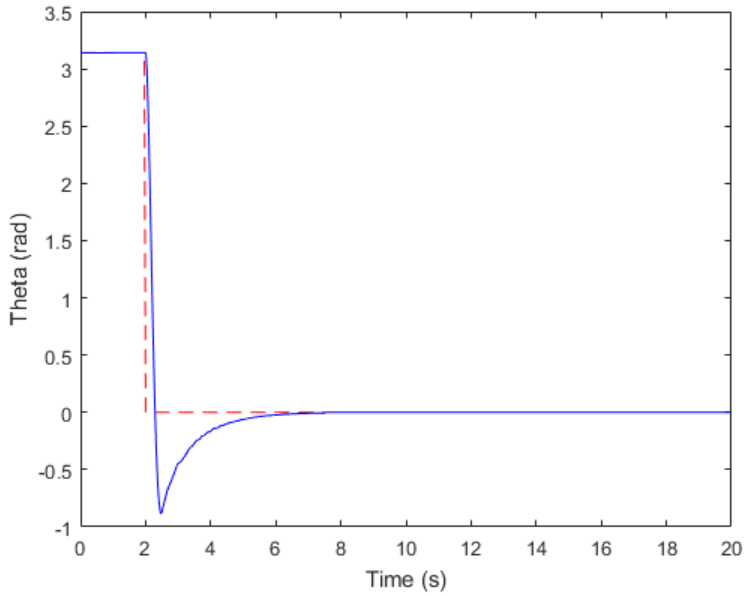

Figure 38: Naive PID Controller Response to Step Input

In this case the system was able to perform remarkably well in response to the commanded change in position of the motor. Its performance is comparable to that of the

psuedo-derivative control system. This is indicative of the lack of high frequency noise in the signal due to the competing control inputs.

Having demonstrated the use of both PID and Bang-Bang controllers to achieve motor position control it is important to discuss the relative pros and cons of both systems. In a bang-bang control scheme the system is subjected to a rapid fire series of pulses in order to achieve a steady motor velocity. This results in high stresses within the system due to the rapid variation of input signals from low to high. In the case of a PID controller, the system settles over time to an appropriate constant input voltage to achieve the desired motor velocity. This places far less stress on the system as the variation in input voltage is relatively low once the system has settled. This constant voltage results in less steady state error, allowing the PID control system to more closely track the desired position. For this reason it is generally more advantageous for the user to use an appropriately tuned PID control scheme over a bang-bang controller.

VI. Conclusions

Through a series of tests, physical motor parameters were determined to create a theoretical model of the motor. This model was used to simulate the motors response to a series of different control schemes and disturbances. A series of control schemes were implemented on the physical motor. The first controller tested was a closed loop controller, which was simple and required little instrumentation but lacked the ability to respond to disturbances. A simply bang-bang closed loop controller was tested, which was found to effectively reach a correct average state and respond to disturbances but continually overshot and undershot the desired state. A hysteresis band was implemented to the bang-bang controller, which effectively reduced the frequency that the voltage oscillated but increased the amount the controller overshot and undershot the desired range. Next, the PID control scheme was examined. The proportional control element was found to quickly bring the system to the desired position with small steady state error and non negligible overshoot. The integral controller was found to be unstable and resulted in a sinusoidal signal that increased in magnitude with time. The derivative controller was successful in holding the position of the motor steady and proved effective in slowing the velocity of the motor. The derivative system alone was ineffective in producing a desired output. When proportional control was implemented in conjunction with integral control it was found that the system would overshoot its target significantly but return to the desired state over a period of time. This performance was improved via the addition of a non zero derivative control scheme which resulted in a complete PID controller. When properly

tuned it was found that the PID controller was able to settle to the desired position quickly and with minimal overshoot. The PID's constant output voltage proved advantageous over that of a bang bang control scheme due to its ability to track a signal with minimal oscillation and smaller magnitude and lower frequency voltage alterations. By altering the PID gains using programs like sisotool the PID controller could be modified to produce an even better result than was found experimentally during this lab.

VII. References

[1] Lum, C., *Lab 2 - DC Motor Modeling and Control*, William E. Boeing Department of Aeronautics and Astronautical Engineering, 2018.