

December 15, 2017

Dr. Barry Fussell and Dr. Alireza Ebadi

Dept. of Mechanical Engineering

University of New Hampshire

33 Academic Way

Durham, NH 03824

Dear Professor Fussell and Dr. Ebadi,

With the anticipation of the UNH’s LunaCats competing at the NASA Robotics Mining Competition our team has decided to develop PID control for both the velocity of the Mars Bot and the Mars Cat’s auger velocity.

For the purpose of this experiment a theoretical model for a smaller former battlebot called ‘Another Brick in the Wall’ is developed to compare theoretical results qualitatively to experimental results for different control values. Another model is made for the Mars Cat’s auger to be controlled by a PID controller based on the robot’s position from a wall at 90° to simulate the Mars Bot’s approach to the mining area. Both system models are created in Simulink, and MATLAB is used to for the analysis of their system constants and evaluating proper controller values.

Regards,

Team Velocity and Position Control of a Robot Vehicle and Auger Motor

Members:

Zhangxi ‘Jesse’ Feng

Michael Locke

Simon Popecki

James Skinner

Matthew Westbrook



|  |  |
| --- | --- |
| Course Number and Name:  ME 747  Experimental Measurement and Modeling of Complex Systems | |
| Semester and Year:  Fall 2017 | Name of Lab Instructor:  Barry Fussell  Alireza Ebadi |
| Lab Section and Meeting Time:  Lab Section 2B/3B  Tuesdays/Wednesdays 2:00-5:00PM | Report Type:  External Group Report |
| Title of Experiment:  Velocity Control of a Robot Vehicle and Auger Motor  -Final Laboratory- | |
| Date Experiment Performed:  15 November – 15 December 2017 | Date Report Submitted:  15 December 2017 |
| Names of Group Members:  Zhangxi ‘Jesse’ Feng  Michael Locke  Simon Popecki  James Skinner  Matthew Westbrook | Grader's Comments: |
| Grade: |

TABLE OF CONTENTS

[OBJECTIVES 1](#_Toc501027183)

[EXECUTIVE SUMMARY 2](#_Toc501027184)

[THEORY AND EXPERIMENTAL METHODS 3](#_Toc501027185)

[RESULTS AND DISCUSSION 5](#_Toc501027186)

[CONCLUSIONS 5](#_Toc501027187)

[REFERENCES 5](#_Toc501027188)

[APPENDIX 5](#_Toc501027189)

[C: Equipment List 5](#_Toc501027190)

LIST OF FIGURES

LIST OF TABLES

# OBJECTIVES

The goal of this project is to determine the appropriate constants of two PID controllers for a robot’s DC brush motors and an auger motor. The controller for the robot receives the position data from an ultrasonic sensor and produces a velocity output to the motors. The controller for the motor receives a step voltage input and produces an output that optimizes both fast response and minimal oscillation from overshoot.

# EXECUTIVE SUMMARY

# 1 THEORY AND EXPERIMENTAL METHODS

The UNH LunaCats designs, builds, and competes with a mining robot at NASA’s Robotics Mining Competition in Kennedy Space Center at Cape Canaveral every May. The robot needs to traverse through terrain with random obstacles to reach a mining area. The material must be mined from the mining area only, therefore it is beneficial for the robot to reach the area as quickly as possible with no overshoot such that the robot does not waste time traveling any extra distance. This year’s LunaCats robot will use an auger as the mining apparatus. The auger consists of a rotary motor connected to auger tip that acts like an Archimedes’ screw. For this project, the drive train is modeled using the robot named Another Brick in the Wall as shown in **Figure 1**; the auger motor is modeled using a VEX robotics gearbox motor as shown in **Figure 3**.



Figure : Robot “Another Brick in the Wall” with direct drive DC brushed gearmotors

## 1.1 Auger Motor – Velocity PID Control

For the auger subsystem, proportional-integral-derivative (PID) velocity control was developed in this experiment and tested on a prototype platform to evaluate the control algorithm. The goal of this platform and algorithm is to develop an understanding of P, I, PI, and PID control using sensor feedback and a dedicated microcontroller to process the sensor inputs and motor outputs. To start, a basic process diagram was developed to understand the velocity control process, in which four main parts to this experiment were determined. First, a means of delivering a step input via user input was needed. Second, a microcontroller was needed to take this step input and control a motor accordingly. Third, a motor to be tested was needed that would provide sufficient output speed. Fourth, a sensor that could be used to provide velocity feedback from the motor to the microcontroller was needed to act as the basis for the control data. **Figure 2** (next page) demonstrates this process flow:

Knowing the overall process of the velocity control, components were selected that would satisfy each of the four components shown in **Figure 2** and that were easily accessible. An Arduino Uno served at the microcontroller that would process the inputs, outputs, and velocity control algorithm. A basic push button served as the means of user input to initiate a step input. A VEX robotics gearbox motor served as the tested motor for its desirable working speed of about 350RPM. Lastly, an encoder was chosen to provide feedback to the Arduino. Since the encoder measures position and not velocity, the position data was converted to velocity data by taking the difference in positional encoder ticks (rotational resolution) and dividing by the corresponding change in time measured using an internal timer on the microcontroller.

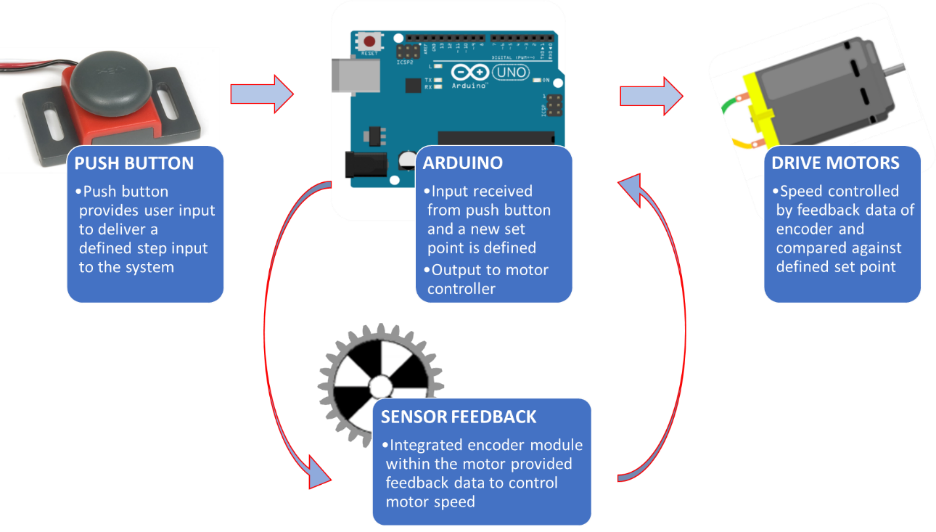


Figure : Velocity Control Process Diagram

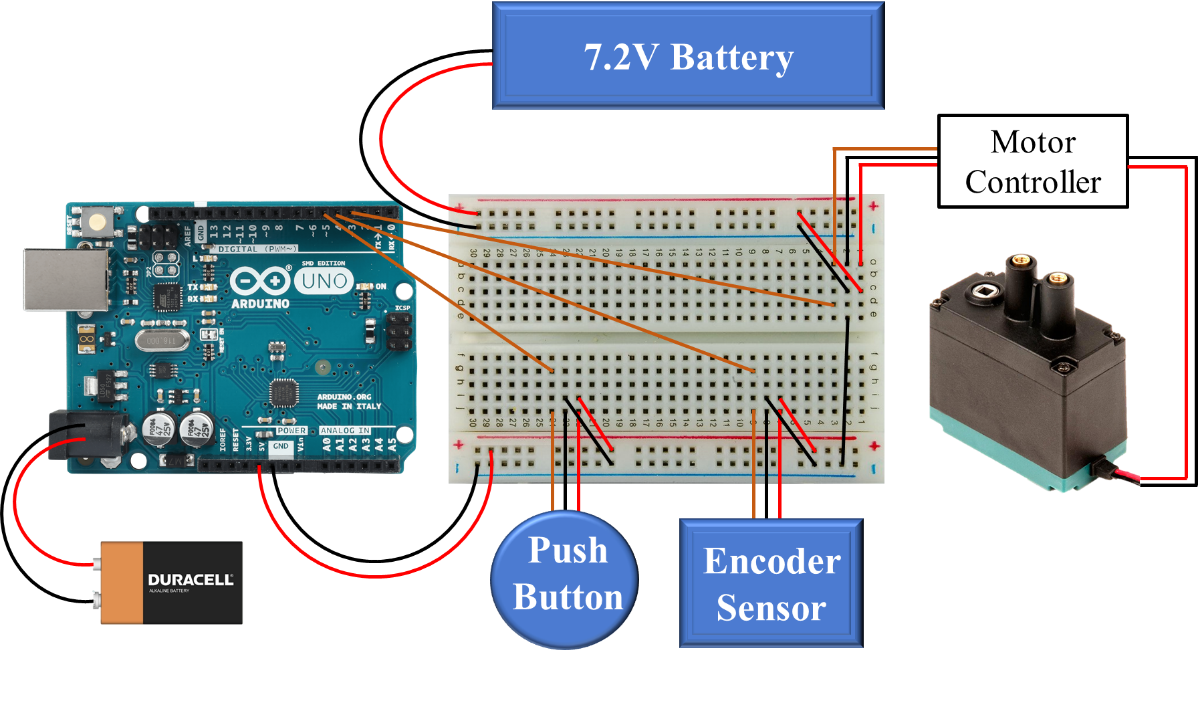


Figure : Wiring Diagram for Experimental Setup

The next step was to combine all the components. A wiring diagram was created to define how each component would be connected and to serve as a reference when creating the circuit. **Figure 3** above reveals the velocity control experimental circuit:

From the wiring diagram, note that an extra 7.2-volt battery was used to provide enough power to the motor controller and motor. Using this layout, the experimental setup could finally be constructed. The first experimental prototype and final experimental prototype can be seen below:

Seen in **Figure 4** below, multiple experimental setups transpired throughout this project. The first version contained an Arduino microcontroller, separate quadrature encoder, a motor, and push button. The final version contained a VEX Robotics microcontroller, integrated encoder module with the motor housing, a motor, and push button. The reason for this change in equipment was due to numerous issues with components burning out and the Arduino not sampling fast enough to accurately count the number of encoder ticks. The VEX Robotics microcontroller has built in encoder features to accurately track encoder position, so it was used in the final setup.

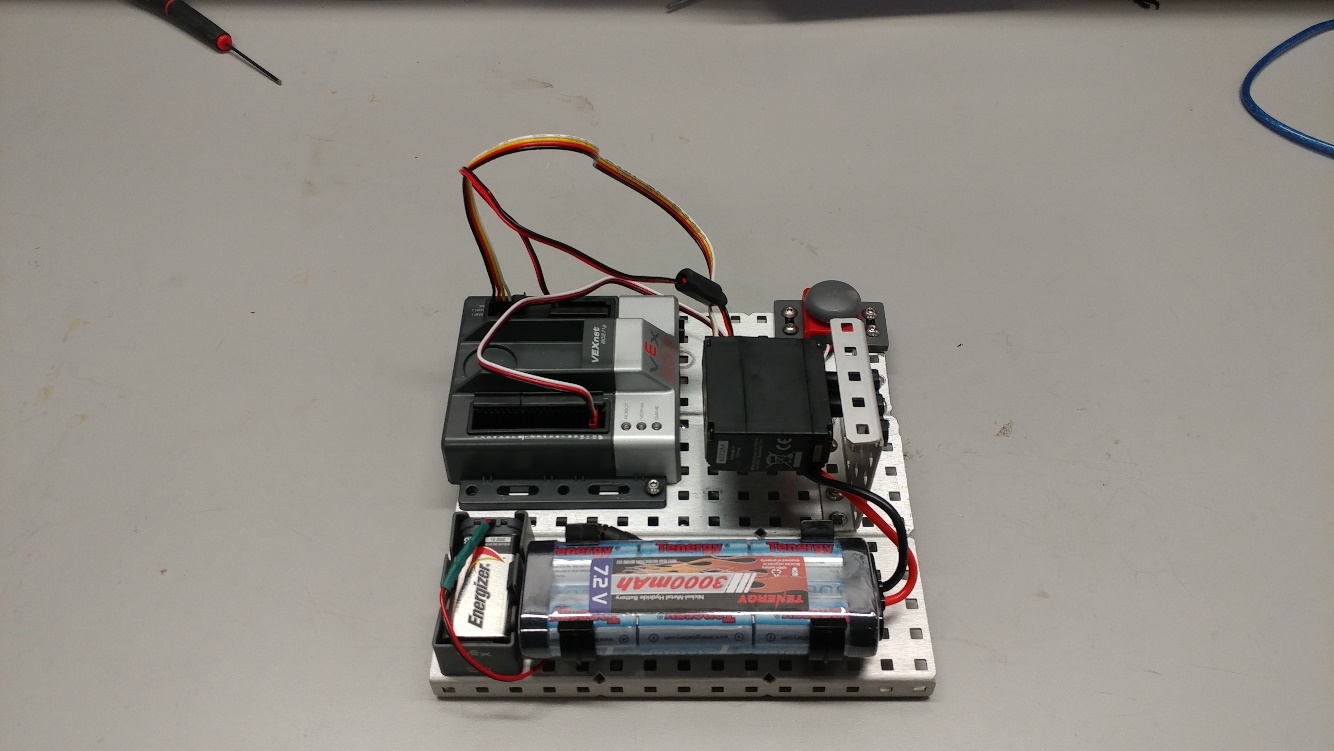
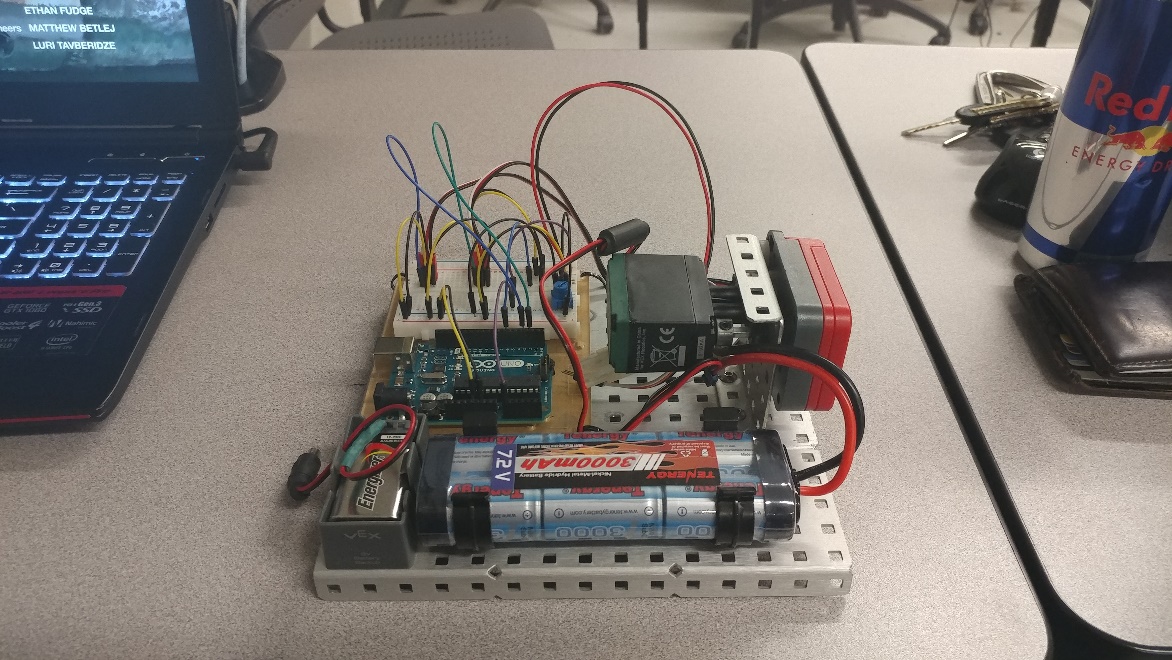


Figure : Experimental Prototypes (Left: Original, Right: Final)

## 1.2 Robot Drive Train – Position PID Control of Velocity

For the drive train subsystem, a proportional-integral-derivative (PID) position-feedback velocity control was also investigated.

## 1.3 Proportional-Integral-Derivative Control Theory

Now that the experimental setup has been built, the control algorithm needed to be created to control the motor velocity of the system. The three control methods investigated in this experiment were proportional, integral, and derivative control (and combinations of PI and PID). The basis for each control method can be seen in equations 1 to 3 below:

Equation 3

For proportional control, the difference between a desired set point velocity and the feedback velocity of the motor is considered. For integral control, the summation of error over a time period is considered, so the error will start to increase rapidly over time if the error is large and the control system does not see the input approaching the set point. Finally, for derivative control, the rate at which the feedback velocity is approaching the setpoint velocity is considered. For a full PID control, the sum of these three errors controls the response of the motor. Combining equations 1 to 3 gives:

Equation 2

Equation 4

Equation 1

Where , , and are the respective proportional, integral, and derivative gain values that tune the system response for different responses. Knowing these control algorithms, a block diagram of the system can be developed:

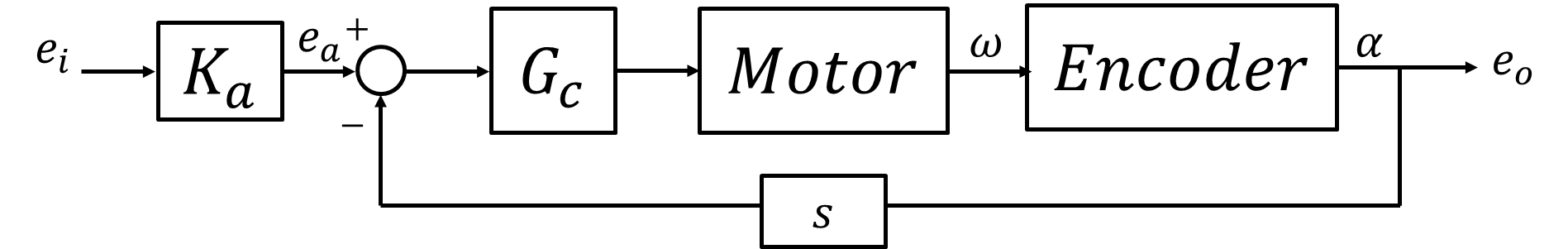


Figure : System block diagram for the auger motor

Where is the motor controller gain, and is the respective controller being used. Note that the encoder signal needed to be differentiated to obtain a velocity signal. Using this block diagram, a transfer function was derived that describes the complete PID control system:

Equation 5

Where and are the motor gain and motor system response, respectively. These values were going to be found experimentally. Once all values of the transfer function were determined, root loci of each control system were determined to better understand the range of system parameters and system responses.

The original setup was first tested, and the initial analysis started with the calibration of the pulse-width-modulation (PWM) input to the VEX motor controller. The motor controller specification sheet called for an input PWM signal of 1.5-2.0ms pulse widths, so this band was tested to discover the optimal operating range. Once the best operating range was found, an initial proportional control step input was examined. The Arduino was not able to accurately sample the data, and lousy data was obtained due to aliasing. Due to this, the Arduino microcontroller was switched to a VEX Robotics microcontroller, and testing resumed. This new platform operated on sending a power signal of -127 to 127 to the motor (with -127 being full reverse, 0 being stop, and 127 being full forward). A desired starting velocity was calibrated to a corresponding motor power input, and a mapping equation was created to proportionally convert future velocities to the correct, corresponding motor power:

The different control algorithms were then implemented into the system and tested against a set step impulse of a desired set point. P, I, PI, and PID control were all examined and compared against theoretical step responses. Resulting settling times were used to evaluate each system.

# 2 RESULTS AND DISCUSSION

## 2.1 Velocity Control of the Auger Motor

Analysis of the velocity controlled auger started with calibration of the DC motor with PWM input. **Figure 6** below shows the velocity of the motor (calculated using the encoder) versus the PWM input from the Arduino. The motor must overcome a stall torque to rotate and after a certain PWM input the motor is at its maximum velocity. From this data, step inputs were used from 184 to 250 rpm which is within the increasing linear region of the calibration plot.

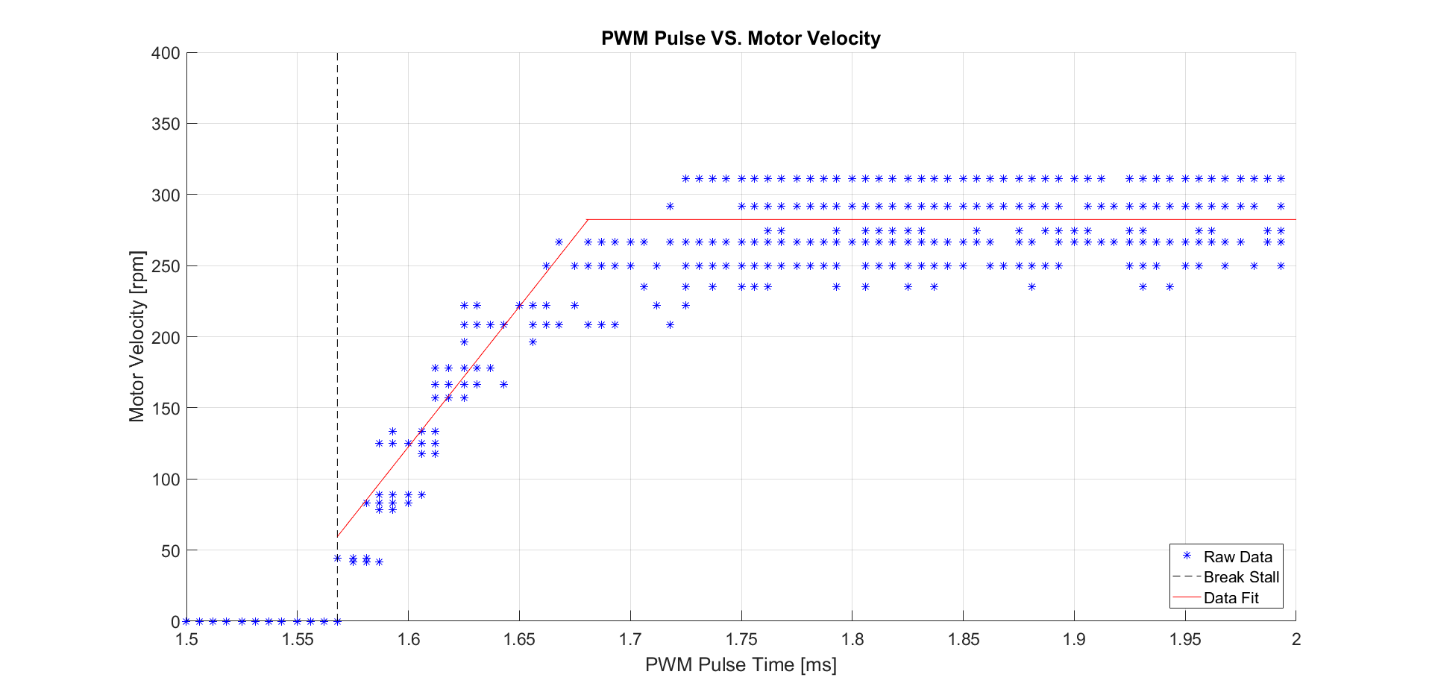


Figure : Calibration of DC Vex 393 motor with varying PWM inputs from Arduino microcontroller

Step inputs were initially done using an Arduino with PWM. The step response of the motor with proportional gain is shown in **Figure 7**. The PWM input combined with the Arduino’s inability to sample fast enough resulted in very noisy data. It was not feasible to analyze this response which led to using the Vex controller.

The motor was then analyzed with a step input to the Vex controller using proportional control. This resulted in **Figure 8** below. This data was used to find the gain of the motor which is 0.7729. The time constant of the motor is 0.35s which was used for theoretical control analysis. The steady-state error of the motor with P-control is 16% of the target speed.

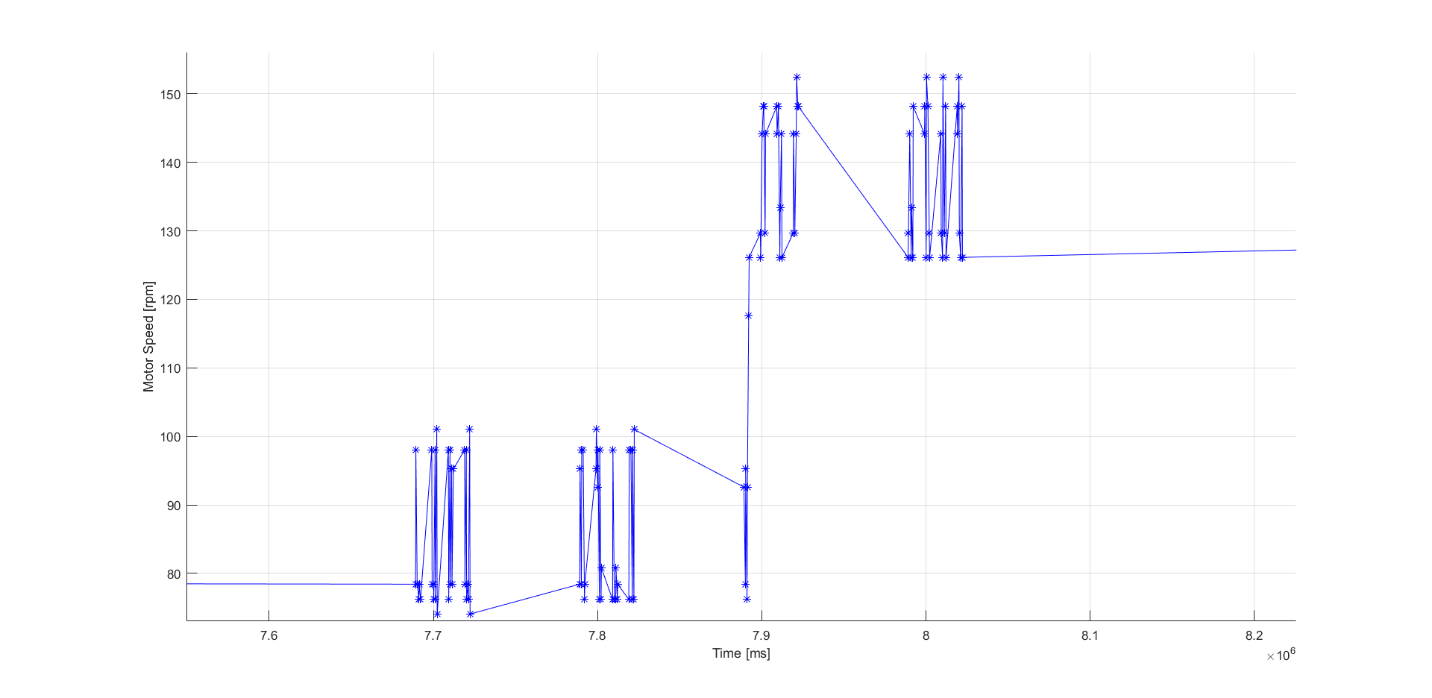


Figure : Velocity of motor with PWM step input



Figure : Step input with proportional gain of 0.5

The motor was tested with integral control shown in **Figure 9**. There was no steady-state error in the response, but the oscillations cause an increased settling time of approximately 5.6s. This is greater than the theoretical settling time of 3.1s with integral control.



Figure : Step input to motor with I-control



Figure : Step input to motor with PI-control

Proportional and integral control were implemented on the motor as shown in **Figure 10**. Proportional-Integral control had no steady-state error and a much faster settling time than the integral control at approximately 3.1s experimentally and 2.2s theoretically.

Derivative control was added to the system as shown in **Figure 11** for Proportional-Integral-Derivative control. The response had nearly no steady-state error and settled faster experimentally at approximately 2.4s. The theoretical response was very close to experimental with a settling time of 2.7s.



Figure : Step input to motor with PID-control

Root loci were also performed for the motor with each velocity control shown in **Figures 12, 13, 14, and 15**. These root loci show how the gains can be altered to change the response of the system. The proportional control can be adjusted for a faster response, but this will change the steady-state error. The integral control cannot be adjusted for a faster response time. Both PID and PI control can be adjusted to respond faster but PID will be better assuming the system is accurate in calculating the derivative.



Figure : Root locus of motor with P-control



Figure : Root locus of motor with I-control.



Figure : Root locus of motor with PI-control



Figure : Root locus of motor with PID-control

## 2.2 Position Control of the Robot’s Velocity

In order to determine the system parameters of the robot the Arduino’s code was set to go at maximum speed, ignoring any obstacles sensed by the ultrasonic sensor. **Figure 12** below shows output voltage from the ESC PWM signal before it reaches the motors.



Figure : The resulting output voltage of the robot driving along the table

Using this response the time constant and gain of the system were calculated for the measured constant voltage from the DC battery source measured as 7.41 V and a steady state voltage of 7.20 V. The motor’s resistance was determined by tapping the motor leads (disconnected from the ESC) with an ohmmeter. Using the specifications for the Beetle B16 Gearmotor 16:1 the stall torque is interpolated for the input voltage from the battery, which then allowed the torque constant to be calculated. By rearranging equation (**INSERT EQ # HERE**) the damping and moment of inertia for the system is calculated. The ‘Another Brick in the Wall’ system parameters are tabulated in **table 1** below.

|  |  |
| --- | --- |
| **Time Constant, τ (sec)** | 0.0244 |
| **System Gain, K** | 0.9719 |
| **Motor Resistance, R (Ω)** | 4.0000 |
| **Stall Torque, Td (oz-in)** | 31.2300 |
| **Torque Constant, Kt (oz-in/A)** | 20.8200 |
| **Motor Voltage Constant, Ke (V/rad/s)** | 0.1470 |
| **Damping, B (oz-in/(rad/s))** | 4.5905 |
| **Moment of Inertia, J (oz-in-s2)** | 0.1306 |

Table : Another Brick in the Wall system parameters

The time constant was expected to be a small value from qualitative experimentation of the robot going to full speed. The time constant was limited electronically by the ESC so that the motors would not exceed certain current limits.

Using the system parameters above for the robot, the system can be simulated with Simulink. The Simulink model is displayed in figure (**INSERT FIGURE NUMBER OF SIMULINK MODEL HERE**).

# 3 CONCLUSIONS

## 3.1 Zhangxi ‘Jesse’ Feng

## 3.2 Michael Locke

## 3.3 Simon Popecki

## 3.4 James Skinner

## 3.5 Matthew Westbrook

The goal of the analyzing the auger motor was to find its defining system parameters and implement velocity control to quickly reach the desired input. The initial use of an Arduino posed problems with PWM and a slow sampling time affecting the data. We were able to find the best range to operate the motor and then used a Vex controller for further analysis. Parameters of the motor were found which allowed for experimental and theoretical comparisons of velocity control.

From analysis of the motor with velocity control, PID was the best combination of response time and no steady-state error. To use PID control the system needs to be very faset and precise in its calculations of velocity. The initial setup with the Arduino would not have been reliable with PID because of noise in the system and slow sampling time. With the vex controller it made the system settle faster than using PI control alone.

In the future the system could be optimized to have a better response with PID control. Our data showed the PID performing better than other control methods but with proper optimization from the root locus it could be altered to have less overshoot and oscillations before settling to the steady state value.

# 4 REFERENCES

[1] Ogata, K., *System Dynamics,* 4th ed*.,* Upper Saddle River, NJ.: Person Prentice Hall, 2004.

[2] Figliola, R., and Beasley, D., *Theory and Design for Mechanical Measurements,*5th ed., Hoboken, NJ.: John Wiley & Sons, 2011.

# 5 APPENDIX

## C: Equipment List

* Station #2 – Kingsbury S221
* Arduino Uno
* Arduino Mega
* Sabertooth 2x12 ESC
* Beetle B16 gearmotors (Henkwell)
* Robot Chassis Platform with Motors
* Ultrasonic Sensor HC-SR04
* Vex Robotics Microcontroller
* VEX 393 Motor with Integrated Encoder Module
* NI Eight Slot Chassis (NI PXIe-1062Q)
* Control Board (NI PXIe-8360)
* Function Generator (NI PXI-5412)
* Digital Multimeter (NI PXI-4065)
* Digital Oscilloscope (NI PXI-5142)
* DC Power Supply (NI PXI-4110)
* DMM Soft Front Panel Software
* LabVIEW 2016
* MATLAB R2016B