LOW VOLUMETRIC FLOW RATE INJECTION SYSTEM

Sponsored by:

National Science Foundation

Dr. Brian Elbing
Mechanical and Aerospace Engineering
Oklahoma State University

Prepared by:

Sarah Bonk, Team Lead
Erin Peterson
Melissa Duncan
Alec Barker

May 10, 2017

CONTACT INFO

Dr. Brian Elbing (Project Sponsor)
Phone: 405-744-5900
Email: elbing@okstate.edu

Sarah Bonk (Team Lead)
Phone: 918-408-7321
Email: sbonk@okstate.edu

Alec Barker
Phone: 405-401-0724
Email: alecsb@ostatemail.okstate.edu

Melissa Duncan
Phone: 405-760-8519
Email: melissa.duncan@okstate.edu

Erin Peterson
Phone: 918-978-0516
Email: erin.peterson@okstate.edu

ABSTRACT

Current research is being studied in polymer drag reduction within a turbulent boundary layer. This research requires precise control of the volumetric flux of solution injected into a developing boundary layer. Because of significant uncertainty in the current system, a critical need exists for an improved injection system that is both mobile and has a wide range of operation. This project includes the design and construction of a mobile syringe pump system that has a mobile platform, digital control of the injection rate, ability to operate over a wide range of volumetric injection fluxes, and can be used with water and polymer solutions. The system design consisted of fluid flow analysis, sizing a motor, design of the system components, and digital controls. The system was tested to verify the flow rates that could be achieved. This calibration found the maximum achievable flow rate to be $8Q_s$. Additional calibration will be done to the system once the system construction is finalized.

TABLE OF CONTENTS

NOMENCLATURE

A	cross-sectional area of pipe
A_r	seal area
d	nominal major diameter
d_m	mean diameter
D	pipe diameter
D_p	diameter of plunging rod
ε	roughness factor
E	modulus of elasticity
f	coefficient of friction
f_c	friction from O-ring compression
f_h	friction from fluid pressure
f^*	Fanning friction factor
F	compressive force
F_T	thrust force
FF	frictional force for O-ring
FC	seal compression force
FH	seal hydraulic compression friction
γ_D	shear rate
h	height
I_y	moment of inertia about the y-axis
K	loss coefficient
l	lead
l_p	length of plunging rod
L	length of tube
L_r	length of seal rubbing surface
MW	molecular weight of polymer
N	threads per inch
P	pressure
p	pitch
ρ	density
Q_{max}	maximum volumetric flow rate
Q_s	discharge per unit width
r_i	syringe tube inner radius
r_o	syringe tube outer radius
r_p	radius of the plunging rod
Re	Reynolds number
σ_t	tangential stress
σ_r	radial stress
t	syringe tube thickness
τ_D	shear stress at onset of degradation
U	velocity
U_{max}	maximum velocity
ν	kinematic viscosity
V	velocity

ACKNOWLEDGMENTS

This project was supported by the National Science Foundation. We thank Dr. Elbing for giving us the opportunity to be a part of his research and for always providing his insight and expertise.

We thank Dr. Taylor, Dr. Kidd, Dr. Conner, and Dr. Delahoussaye for their support of our project, specifically with the digital controls, motor, and Solidworks. We also want to extend a huge amount of gratitude to John Gage, if it weren't for him the system would literally have not been made.

1 INTRODUCTION

1.1 PROBLEM STATEMENT

Current research is being studied in polymer drag reduction within a turbulent boundary layer. This requires precise control of the volumetric flux of solution injected into a developing boundary layer. At this time, the rate is controlled and measured using a scale-stopwatch-bucket system and pressure control to achieve the desired injection condition. This introduces significant uncertainty in the injection rate and makes it nearly impossible to repeat a desired condition. Because of this, there is a critical need for an improved injection system that is both mobile as well as flexible in the range of operation.

1.2 DELIVERABLES

- Design and build a mobile injection system that includes
    - Mobile platform
    - Digital control of the injection rate
    - Operation over a wide range of volumetric injection fluxes
    - Use with water and polymer solutions

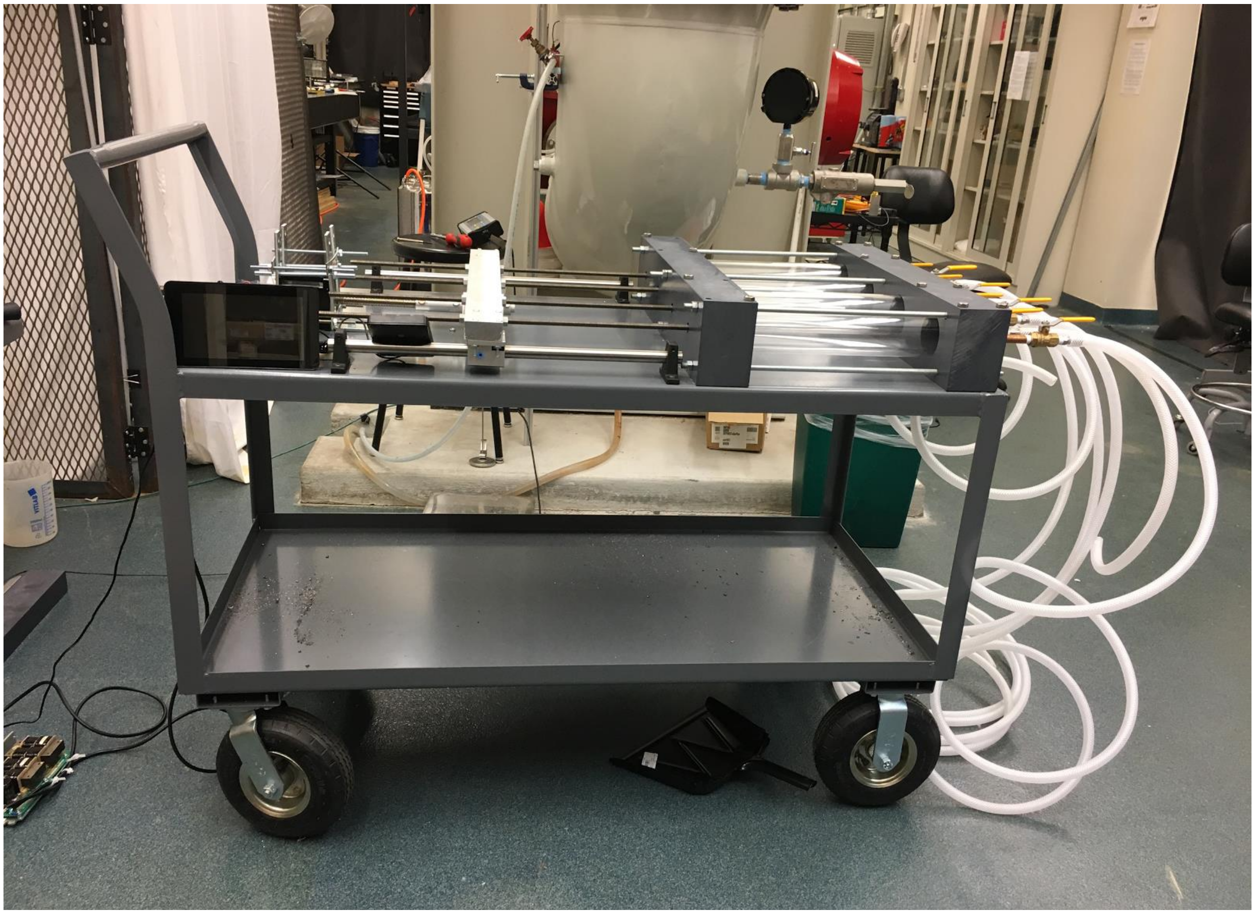


Figure 1 Mobile injection system created by the team.

2 FLUID FLOW ANALYSIS

The maximum volumetric flow rate that can be injected into the water tunnel without degradation of the polymer solution was determined using a series of equations and assuming the use of polymer WSR308 (worst case scenario). The molecular weight (MW) of polymer WSR308 is $8*10^6$ grams per mole. The shear rate (γ_D) was found using the equation

$$\gamma_D = (3.23 * 10^{18}) * MW^{-2.2} \text{ (Winkel, 2009).}$$

The resulting shear rate was calculated to be 2100 liters per second. The density (ρ) and the kinematic viscosity (ν) were based on the average conditions found in the Experimental Flow Physics Laboratory and valued at 998 kg/m^3 and $1.01*10^{-6}$ m^2/s. The shear stress at onset of degradation (τ_D) was found using the equation

$$\tau_D = \gamma_D \rho \nu \text{ (Elbing, 2009).}$$

The Prandtl-von Karman equation is a function of the Reynolds number (Re) and the Fanning Friction factor (f^*). The Reynolds number was found using the pipe diameter (D), the velocity (U), and the kinematic viscosity.

$$Re = \frac{UD}{\nu} \text{ (Munson et al., 2013)}$$

The Fanning Friction factor was found using the shear stress at the onset of drag reduction, the density, and the velocity.

$$f^* = \frac{2\tau_D}{\rho U^2} \text{ (Elbing, 2009)}$$

These equations were then plugged into the equation

$$\frac{1}{\sqrt{f^*}} = 4\log_{10}(Re\sqrt{f^*}) - 0.4 \text{ (Elbing, 2009).}$$

By evaluating each side of this equation at increasing velocities, a convergence was found. The velocity at which the two sides of the equation converge is the maximum velocity before degradation will occur for the polymer solution.

The volumetric flow rate was found by using the maximum velocity (U_{max}) and the tube's cross-sectional area (A) in the continuity equation

$$Q_{max} = U_{max}A \text{ (Munson et al., 2013).}$$

The system was required to be able to inject fluid at the minimum and maximum injection rates of $2Q_s$ and $10Q_s$, respectively. The discharge per unit width (Q_s) was found using the equation

$$Q_s = 67.3\nu \text{ (Wu, 1970).}$$

To size the motor for the system, the force to overcome the initial pressure was needed. This was found by first calculating the major losses using the equation

$$\Delta P = f\frac{L}{D}\frac{\rho U^2}{2} \text{ (Munson et al., 2013).}$$

This pressure drop (ΔP) equation used the length of the tube (L), the friction factor (f) found in the Moody Chart at the roughness factor ε/D, and the Reynolds number. The equation assumes that the syringe and flexible tubes are straight pipes. The minor losses for the system were calculated using the equation

$$\Delta P = K\frac{\rho U^2}{2} \text{ (Munson, 2013).}$$

where K is the loss coefficient for components in the system. For each syringe tube there is a valve ($K = 20$), fittings ($K = 0$), and a carving ($K = 1.7$) (Munson, 2013). The total fluid pressure (P) acting on the brass stopper in the syringe was then determined by summing the major and minor losses. For the fluid system pressure, 5 feet head of pressure was added due to the height of the tubing feeding the fluid into the water tunnel. This sum was multiplied by a magnitude of 4 to account for any pressure losses not considered and to provide a safety factor.

The thrust force (F_T) to overcome this pressure was found using the equation

$$F_T = PA \text{ (Munson, 2013).}$$

The overall force to overcome the initial pressure was the sum of the thrust force and the friction force caused by the O-rings on each stopper. The O-rings frictional force was determined using equations

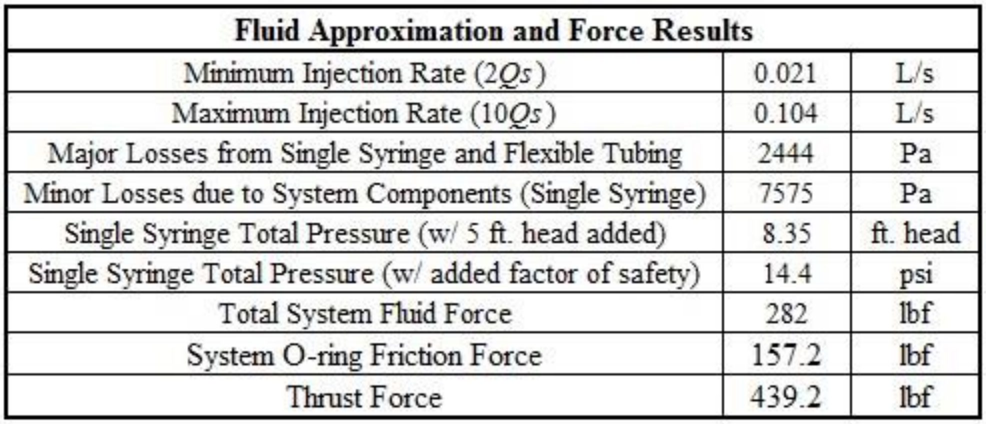
$$FC = f_c L_r$$

$$FH = f_h A_r$$

$$FF = FC + FH \text{ (Parker)}$$

In these equations the frictional force (FF) for a single O-ring is a function of the seal compression force (FC), friction due to O-ring compression (f_c), length of seal rubbing surface (L_r), seal hydraulic compression friction (FH), friction due to fluid pressure (f_h), and seal area (A_r). Each of the components of the frictional force equation were found assuming a 10% compression design, 0.5 O-ring material hardness, 20 psi fluid pressure, and O-ring dash classification 330. The system will use a total of 8 O-rings. The resulting frictional force for the 8 O-rings was added to the linear force to calculate the total force needed to overcome the fluid pressure. The result was the force the motor needed to generate in order to overcome the pressure acting on the system.

The calculated values can be seen in Table 1 and a detailed run through of the calculations can be seen in Appendix D. The values in the table provided the basis for sizing the system. Based on the calculations, the inner diameter of the syringe tubes was determined to be 2.5 inches. These calculations were only done with the assumption that the polymer solution being used would consist of WSR308 at 1000 parts per million. Because of this, calculations would need to be done for additional polymer solutions of different concentrations to check that the force from the motor could still overcome pressure in the syringe tubes.



Fluid Approximation and Force Results		
Minimum Injection Rate ($2Q_s$)	0.021	L/s
Maximum Injection Rate ($10Q_s$)	0.104	L/s
Major Losses from Single Syringe and Flexible Tubing	2444	Pa
Minor Losses due to System Components (Single Syringe)	7575	Pa
Single Syringe Total Pressure (w/ 5 ft. head added)	8.35	ft. head
Single Syringe Total Pressure (w/ added factor of safety)	14.4	psi
Total System Fluid Force	282	lbf
System O-ring Friction Force	157.2	lbf
Thrust Force	439.2	lbf

Table 1 Results from fluid flow calculations. All calculations based on the assumption that the polymer solution used would be WSR308 at 1000 ppm.

3 MOTOR

3.1 MOTOR SIZING AND SELECTION

The torque required to push forward or pull back the four syringes determined the size of the stepper motor needed. From the fluid flow approximations (§2, Table 1), the thrust force and plunger friction force were used as the total compressive force needed to calculate the torque. The thrust force for all four syringes is 282 pound-force and the plunger friction force for 8 O-rings is 157.2 pound-force. Adding the thrust force for the syringes to the friction force for the O-rings determined the overall system force to be approximately 439 pound-force (values based on a syringe with internal diameter of 2.5 inches). Power Screw Equations from *Shigley's Mechanical Engineering Design* were used to determine the raising and lowering torque required. The raising torque was found using the equation

$$T_R = \frac{F d_m}{2}\left(\frac{l + \pi f d_m}{\pi d_m - f l}\right)$$

and lowering torque was found using the equation

$$T_L = \frac{F d_m}{2}\left(\frac{\pi f d_m - l}{\pi d_m + f l}\right)$$

These equations use thrust force (F), mean diameter (d_m), lead (l), and the coefficient of friction (f). The mean diameter was determined using the nominal major diameter (d) and pitch (p) of the lead screw.

$$d_m = d - \frac{p}{2}$$

The pitch was calculated using the threads per inch (N) in the following equation

$$p = \frac{1}{N}$$

Several iterations were done using a range of lead screw sizes to determine the best-fit lead screw for the lowest raising torque required by the motor. A lead screw with a standard 5/8-inch nominal major diameter and a total compressive force of approximately 440 pound-force must receive a minimum torque of 40.3 pound-inches (645 ounce-inches) from the stepper motor. With oversizing and a factor of safety used, the stepper linear actuator needed was a NEMA 34. (An iteration of the calculations can be found in Appendix E).

A NEMA 34 Hybrid Linear Actuator was ordered from Anaheim Automation. The NEMA 34 provides a maximum force of 528 pounds. This force is 89 pounds greater than the calculated thrust force (439 pounds) needed to move the four syringe systems. A lead screw with a length of 18 inches and a diameter of 5/8 inches is included with the motor. Because the motor had an estimated time of arrival of 30 days after purchase a NEMA 17 motor was used as a stand-in. This stand-in motor allowed testing of the digital controls setup and configuration until the NEMA 34 arrived.

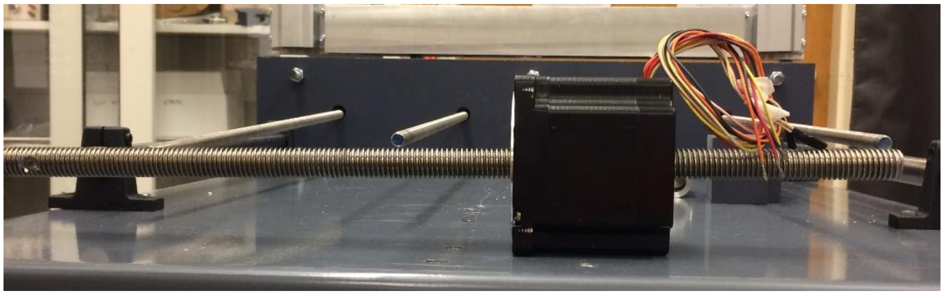


Figure 2 NEMA 34 Non-Captive Linear Actuator purchased from Anaheim Automation.

3.2 LINEAR ACTUATION SYSTEM

Figure 3 shows the linear actuator configuration that was used with the NEMA 34 linear actuator. In the configuration an aluminum bar is attached to two pillow block linear bearings. These bearings are placed on linear mount shafts that are secured to the mobile cart with shaft supports. The aluminum bar is connected to the syringe plunging rods and the motor lead screw via a coupling. This configuration allows for smooth linear movement of the syringe push and pull system.

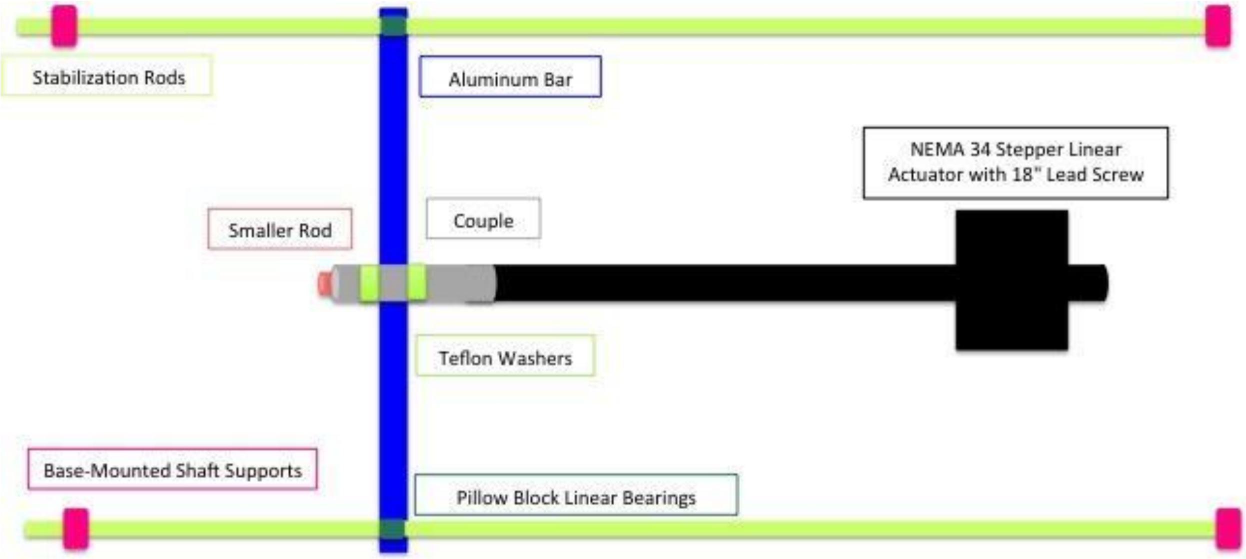


Figure 3 Linear actuator configuration.

The NEMA 34 Linear Actuator is a Non-Captive motor. Because of this, the lead screw must be pinned to provide the appropriate force resulting in forward or backward motion. To compensate for this the coupling rod will be pinned to the push bar using a ¼-inch bolt. Because of the amount of force provided by the motor, it has to be secured to the mobile cart for the system to operate correctly.

3.3 KEY DECISIONS

- To reduce the torque required by the stepper motor, a 5/8”-10 threads per inch 18-inch long lead screw was selected.
- The NEMA 34 stepper motor provides more force than the system is calculated to need, but it was chosen because the oversizing allows for a factor of safety to be included.

- The linear actuation system was designed to mate the motor with the syringe system.
- The lead screw of 18-inches in length was not long enough for the design of the system so a couple was designed to give extra length.
- The NEMA 34 stepper motor chosen is a non-captive stepper motor. Because of this the lead screw must be pinned to allow for actuation. This was done by pinning the couple to the push bar.
- The NEMA 34 did not come with a motor mount so one was designed to secure the motor to the mobile cart.

4 DIGITAL CONTROLS

The design of the digital controls component of the syringe pump system was largely dependent on the limitations and requirements of the mechanical system. The execution of the digital controls was a three-phase process.

4.1 PHASE I

The first phase included the selection of the controller, display interface, and basic peripherals along with developing a basic design for the graphical user interface (GUI). The microcontroller chosen was a Raspberry Pi. This microcontroller is a commonly used, user-friendly device that is compatible with a small-scale stepper motor. It is a series of small single-board computers that operates primarily Linux via Python 2 or Python 3. Because the team had experience with Python as a programming language and Linux is compatible with the Macintosh operating systems (the primary operating system used by the team) the Raspberry Pi was the optimal choice for the microcontroller. To minimize the need for updates and maintenance, a Raspberry Pi 3 Model B with corresponding 2.5 A Micro USB power adapter was purchased.

A small touchscreen was selected in favor of a standard HDMI display, mouse, and keyboard combination because it allowed for the minimization of cost while maximizing functionality. The GUI was designed around the touchscreen capability, thereby eliminating the need for additional peripherals. For development purposes, a mouse and keyboard were used.

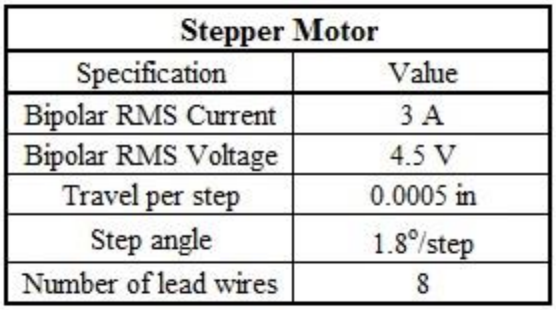
For the GUI draft iterations, a Macbook Pro laptop computer was used in combination with PyQt4 Designer and Spyder software. PyQt4 is a drag-and-drop widget design environment that allows an operator to design a GUI visually rather than via the programming language. Spyder is an integrated development environment (IDE) designed for use with the Anaconda open-source distribution of the Python programming language. Several preliminary drafts of the GUI were developed and modified based on feedback from Dr. Elbing. Draft revisions included: repositioning the system "Exit" button so that it could not be accidentally clicked by the user while adjusting the flow rate, rephrasing the names of the plunger preset positions to clarify their functionality for the user, and resizing and rearranging all GUI components to accommodate for the resolution of the selected hardware.

The touchscreen selected was a 7-inch Touchscreen Display designed by Raspberry Pi. A black case was selected to allow for the Raspberry Pi to be affixed to the back of the selected touchscreen display while neatly containing the display adapter board, jumper wires, and ribbons. Power for the screen is supplied via jumper wires from the Raspberry Pi general-purpose input/output (GPIO) pins to the GPIO pins on the display adapter board. The adapter board converts the signals from the touch inputs received on the display to the display serial interface (DSI) port on the Raspberry Pi via a DSI ribbon cable.

To boot an operating system (OS) on the Raspberry Pi, a 16 GB Micro SD card was selected with the New Out Of Box Software (NOOBS) operating system manager pre-loaded. This manager includes the operating system Raspbian for installation that can provide for the basic programs and utilities for the Raspberry Pi. Raspbian was installed via NOOBS onto the purchased MicroSD card.

Additional advances during this phase included: the use of Bash scripting to install an on-screen keyboard, the use of Nano (the Linux command line text editor) to change the orientation of the display output when booting up, and the development of the initialization code necessary to add functionality to the GUI buttons, spin boxes, and displays.

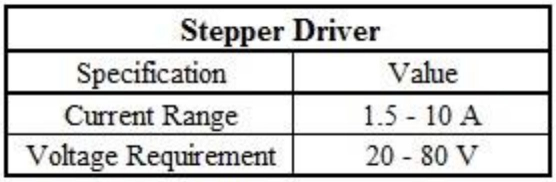
4.2 PHASE II



Stepper Motor	
Specification	Value
Bipolar RMS Current	3 A
Bipolar RMS Voltage	4.5 V
Travel per step	0.0005 in
Step angle	1.8°/step
Number of lead wires	8

Table 2 Specifications of the Anaheim Automation Hybrid Linear Actuator.

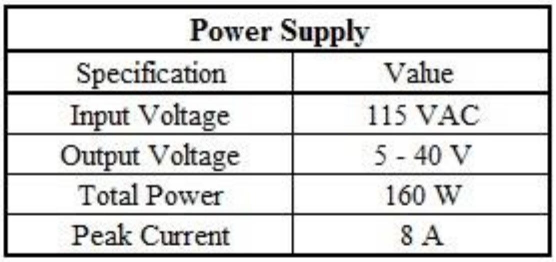
The NEMA 34 stepper motor has specific requirements for the digital control system (Table 2). The manufacturer of the motor (Anaheim Automation) recommended the MBC12101 Stepper Driver (see Table 3 for specifications) to complement the NEMA 34 stepper motor. This stepper driver receives pulses that will be generated by the Raspberry Pi to control the magnitude and direction of the current flow to the motor. This current will dictate the speed and direction at which the motor will turn the lead screw resulting in a specific flow rate from the syringe pump.



Stepper Driver	
Specification	Value
Current Range	1.5 - 10 A
Voltage Requirement	20 - 80 V

Table 3 Specifications for the Anaheim Automation Stepper Driver.

The power for the motor is routed through and modulated by the driver. Because the Raspberry Pi has a maximum voltage output of 5 volts, the DC voltage source for the motor must be externally supplied rather than sourced from the Raspberry Pi. An external power source was selected for the motor (PSA40V4A-1 Unregulated Open Frame Power Supply) based on the recommendations of the manufacturer. This single-phase power supply receives an alternating current at 60 hertz from a standard wall socket and can be set to output up to 40 volts.



Power Supply	
Specification	Value
Input Voltage	115 VAC
Output Voltage	5 - 40 V
Total Power	160 W
Peak Current	8 A

Table 4 Specification of the Anaheim Automation Power Supply.

A 10-key keyboard and lever actuating limit switches were purchased. The 10-key keyboard resolves earlier concern of the practicality of implementing and debugging a full on-screen keyboard while still providing a simple, user-friendly control interface. The lever-actuating limit switches were selected instead of push-button switches because the functionality allows for the switch to be placed out of the plane of motion (Figure 4). Because it is difficult to determine the stopping rate of the motor for various stepping frequencies, it is simplest to remove the limit switches from the plane of motion so that the moving plates of the syringe pump system do not break the switches as the motor decelerates to a stop.

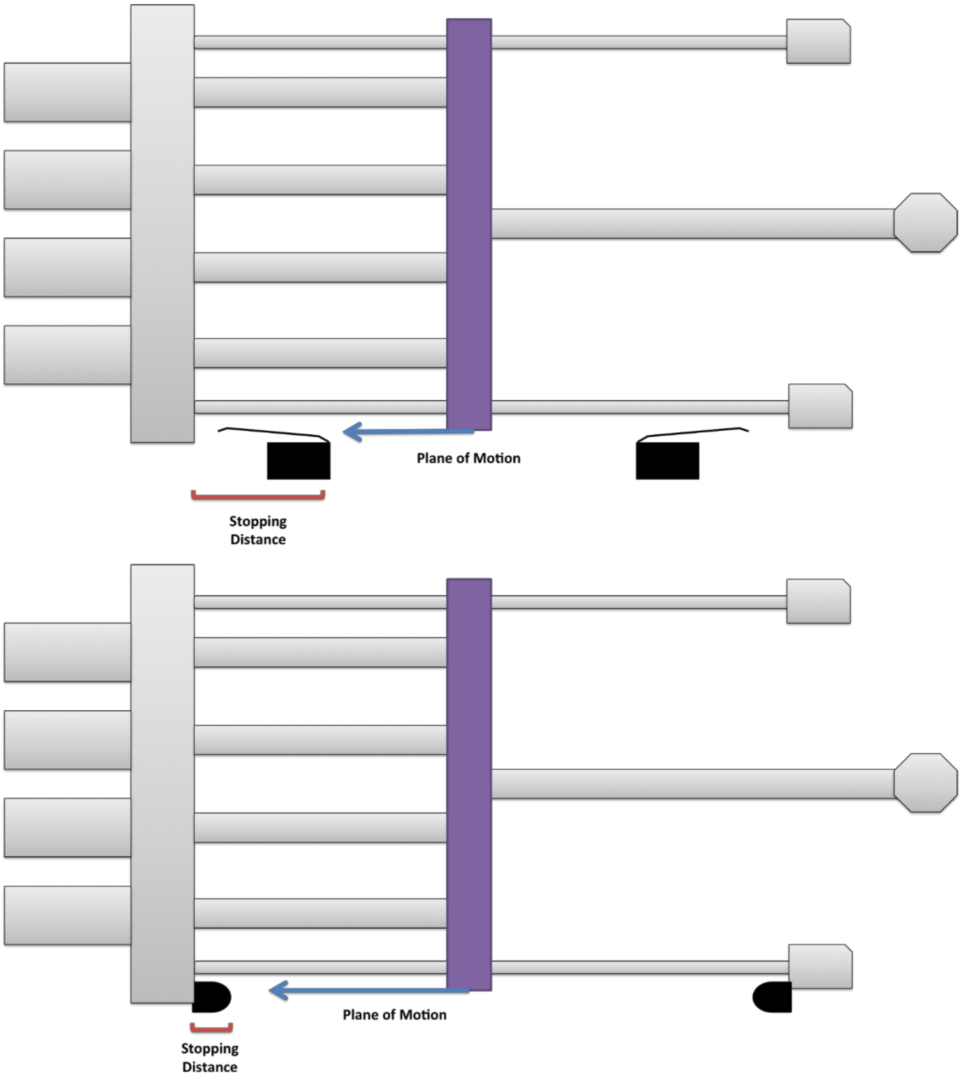


Figure 4 Shows a comparison of the switch configurations for lever-actuating (top) and push-button (bottom) switches to show the positional freedoms of the lever-actuating limit switches.

The finalized schematic for the circuit prototype can be seen in Figure 5. The power supply is routed into the driver board instead of to the motor directly because it allows the driver to modulate the power that the stepper motor receives based on the digital signals received by the driver from the Raspberry Pi.

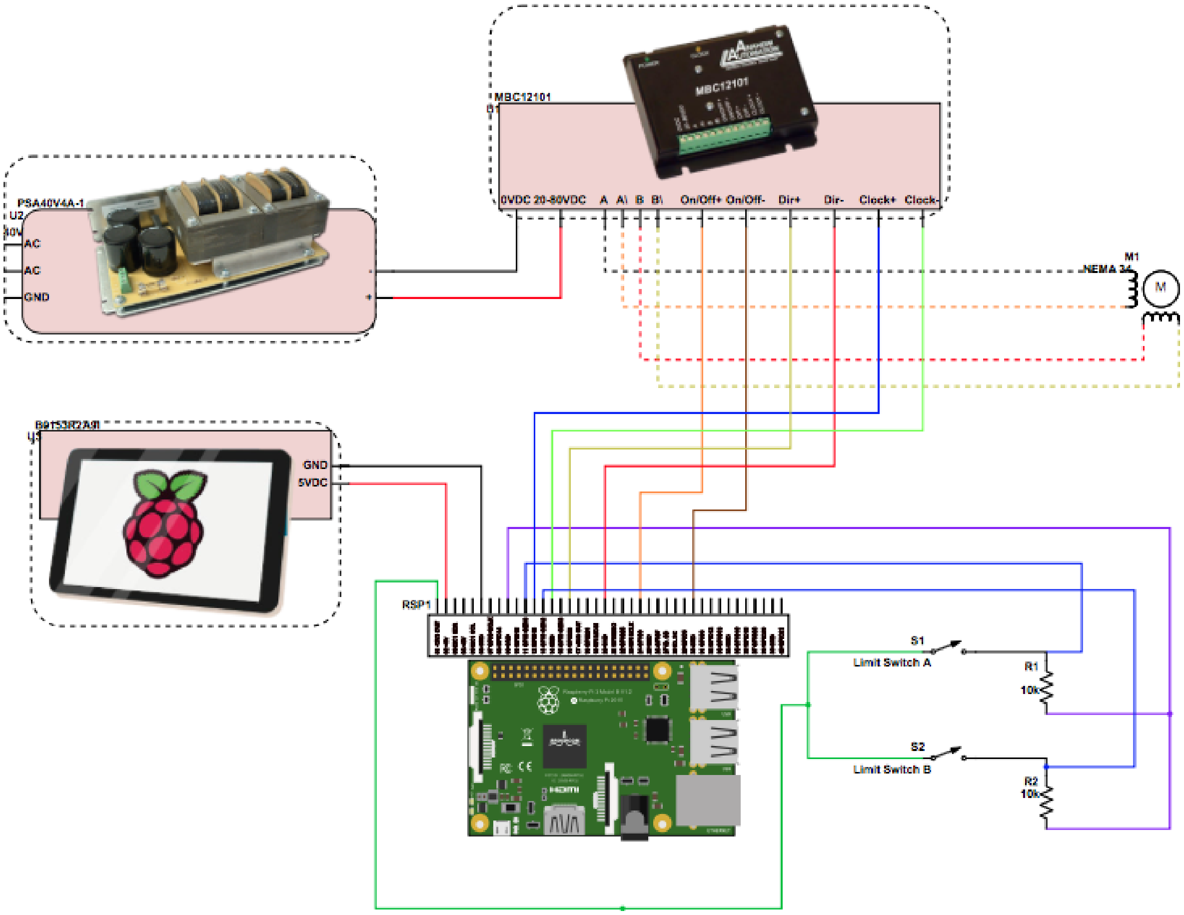
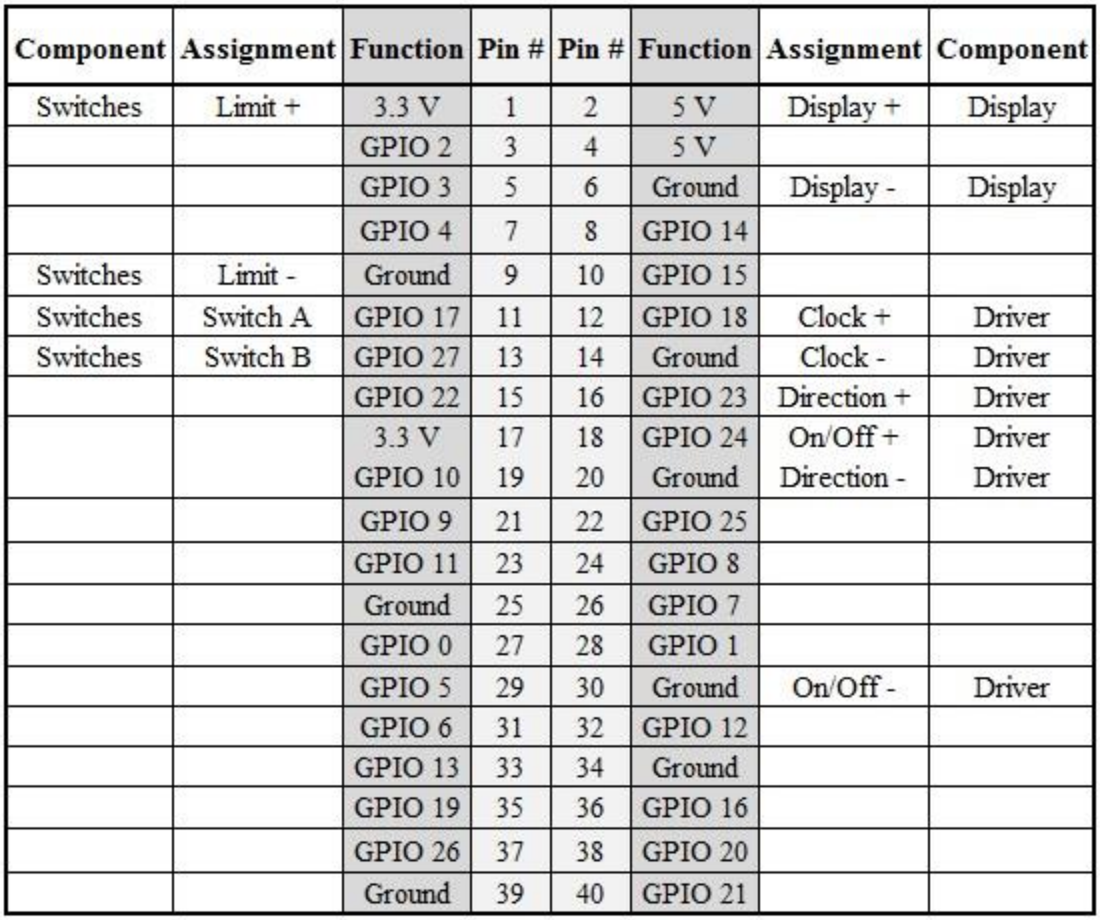


Figure 5 Finalized circuit schematic for the digital controls system. The schematic illustrates how the inputs and outputs of each component interface with the other components in the system.

When developing the circuit schematic, the allotment of the general-purpose input/output (GPIO) pins on the Raspberry Pi was the first step. The Raspberry Pi 3 Model B has 40 GPIO pins that can be utilized as lines of digital control. These pins generally have no predefined purpose and are set by the user to have either input or output functionality. For the design of the system, it was advantageous to give consideration to some of the special functions of particular pins because they could prove useful for interfacing with certain components. For example, pin 12 on the pin-out of the Raspberry Pi 3 Model B is known as GPIO 18 and was designed to have special output functionality that designates this pin as ideal for pulse width modulation (PWM) applications.

PWM is a digital control technique that utilizes pulses of square waves to get analog results out of a digital signal. The high end of the square wave effectively signifies “on” and the low end signifies “off.” By varying the amounts of time that the signal spends in each position, the square wave pattern can simulate voltages in between the high and low positions. The “pulse width” is typically defined as the length of time that the signal spends in the “on” position. This pin is ideal for sending the step pulses required to drive the motor. Other pins provide specific voltage outputs (at 3.3 or 5 volts) or serve as grounds for the circuit design.

See Table 5 for a summarization of the allotment of pins from the Raspberry Pi to the various components.



Component	Assignment	Function	Pin #	Pin #	Function	Assignment	Component
Switches	Limit +	3.3 V	1	2	5 V	Display +	Display
		GPIO 2	3	4	5 V		
		GPIO 3	5	6	Ground	Display -	Display
		GPIO 4	7	8	GPIO 14		
Switches	Limit -	Ground	9	10	GPIO 15		
Switches	Switch A	GPIO 17	11	12	GPIO 18	Clock +	Driver
Switches	Switch B	GPIO 27	13	14	Ground	Clock -	Driver
		GPIO 22	15	16	GPIO 23	Direction +	Driver
		3.3 V	17	18	GPIO 24	On/Off +	Driver
		GPIO 10	19	20	Ground	Direction -	Driver
		GPIO 9	21	22	GPIO 25		
		GPIO 11	23	24	GPIO 8		
		Ground	25	26	GPIO 7		
		GPIO 0	27	28	GPIO 1		
		GPIO 5	29	30	Ground	On/Off -	Driver
		GPIO 6	31	32	GPIO 12		
		GPIO 13	33	34	Ground		
		GPIO 19	35	36	GPIO 16		
		GPIO 26	37	38	GPIO 20		
		Ground	39	40	GPIO 21		

Table 5 Raspberry Pi pin-out allotments by function and component.

The motor driver requires three major input signals from the Raspberry Pi to provide control of the stepper motor. These three inputs are referred to as “Clock,” “On/Off,” and “Direction.” Each of these three inputs have a corresponding output that needs to be grounded by one of the ground pins on the Raspberry Pi. The “Clock” input on the motor driver will receive the PWM signal that will tell the motor the frequency of rotation. For example, a signal with frequency of 1000 hertz corresponds to a motor output of 1000 steps per second (a single step will result in a step angle of 1.8 degrees).

For the other two inputs, standard GPIO pins will be used to modulate the signals. The “On/Off” input enables the motor or disables the motor depending on the presence of an input signal. The “Direction” input tells the motor which direction to turn based on the input signal i.e. if the presence of a signal tells the motor to turn the lead screw clockwise the absence of a signal would result in a counter-clockwise rotation. See Figure 6 for an illustration of the input signals for the “On/Off” and “Direction” inputs represented as square waves being received from the GPIO pins

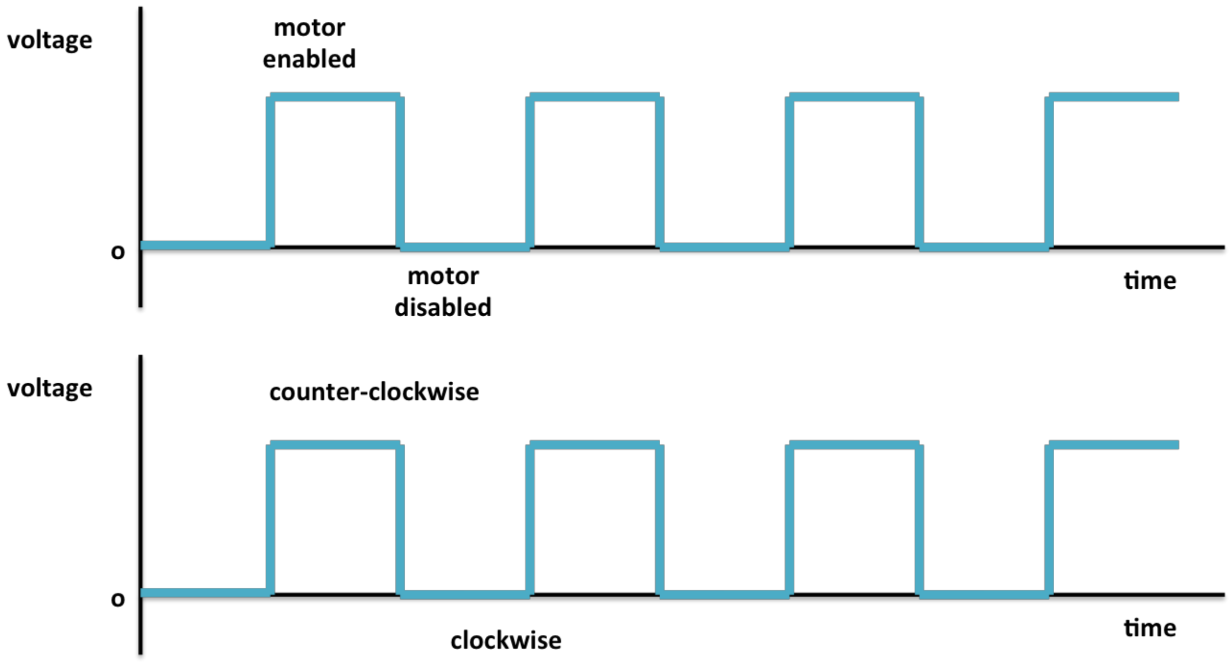
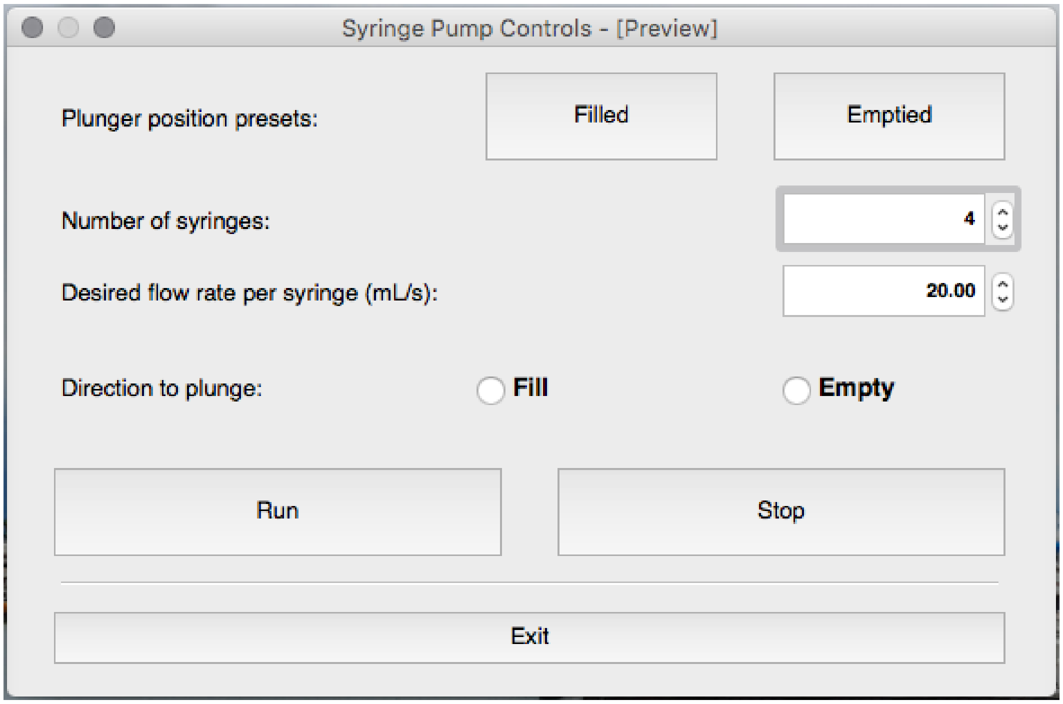


Figure 6 Shows a square wave representation of the input signals for "On/Off" (top) and "Direction" (bottom).

The configuration of the limit switch circuit was such that a 3.3 V source is applied to two switches in parallel. Each switch then proceeds to a GPIO pin on the Raspberry Pi. As can be seen in Figure 4, each line is routed through a resistor before proceeding to ground. These resistors serve as pull-down resistors to keep the logic signal near zero when the switch is open. When the switch is closed, the input voltage is able to send a signal (in the form of a non-zero voltage “high” of approximately 3.3 V) to the GPIO pins. This indicates that the moving plates affixed to the lead screw have reached a boundary and the motor can then be disabled to stop the motion of the linear push bar. In the design of the Python code, the limit switches circuit would have to be checked for a non-zero voltage signal iteratively as the system was in operation.

They Python code written to serve the digital controls system runs an infinite while-loop that continuously evaluates the circuit for alterations that would require a change in the system behavior. A Python code with all system functions can be found in one script, while the layout for the GUI is called in from another. For the GUI to work properly, both scripts have to be located in the same folder. Figure 7 depicts a preview of the GUI that deploys on the Raspberry Pi to provide the user with a simple interface by which to direct to actuation of the syringe pump system. The GUI has a text input box that allows the user to specify the flow rate in milliliters per second, a scroll box that can be used to designate the number of active syringes, two radio buttons that define the direction of actuation, a “Run” button to activate the system based on the inputs, a “Stop” button to deactivate the system, two present position options that will automatically set the plungers to system extremes at a low displacement rate, and an “Exit” button to terminate the program and cleanup the GPIO pin assignment. The two Python scripts can be found in Appendices G and H.



Syringe Pump Controls - [Preview]

Plunger position presets:

Number of syringes: 4

Desired flow rate per syringe (mL/s): 20.00

Direction to plunge: ○ Fill ○ Empty

Figure 7 The GUI design that displays via the Raspberry Pi.

4.3 PHASE III

The 8-lead wiring configuration needed to be shorted appropriately to accommodate for the 4-lead configuration of the motor driver terminal. Figure 8 depicts the wiring diagram for the NEMA 34 hybrid linear actuator selected for the system. To short the wires to achieve the 4-lead configuration, the White/Black wire was connected to the White/Orange wire and the White/Red wire was connected to the White/Yellow wire. The black wire then corresponded to the “A”-phase terminal on the motor driver while the orange wire corresponded to the “A\”-phase terminal, the red wire corresponded to the “B”-phase terminal, and the yellow wire corresponded to the “B\”-phase terminal.

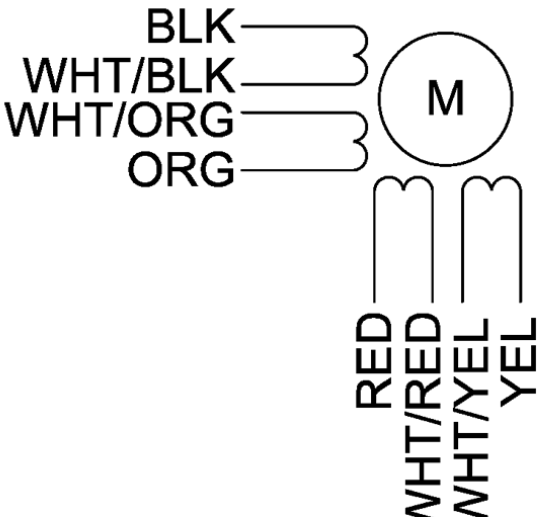


Figure 8 The 8-lead wiring diagram for the Anaheim Automation Non-Captive Hybrid Linear Actuator.

The motor was connected to the digital controls circuit assembly as indicated in Figure 5 and tested independently of the syringe pump mechanical system to ensure that all wiring was done correctly. While it appeared that the directional polarity of this motor behaved inversely to that of the stand-in motor, the NEMA 34 operated as directed otherwise. The directional polarity was resolved by switching the relationship between the high and low voltage signals and corresponding output directions within the Python code. Further calibration will be performed to define the relationship between a given input pulse frequency and the output flow rate.

4.4 KEY DECISIONS

- The Raspberry Pi was selected as the control device for this project because the team had previous experience with the programming language Python 2 and Linux is highly compatible with the Macintosh operating system.
- A touchscreen was selected in place of a HDMI display, mouse, and keyboard combination.
- A 10-key keyboard was selected to provide a simple, user-friendly control interface.
- Lever-actuating switches were selected instead of push-button limit switches because their functionality allows them to be placed outside of the plane of motion.
- The allotment of GPIO pins on the Raspberry Pi was developed base on the input/output signal requirements of each component’s terminals and their functions.
- An infinite while-loop structure was employed via the GUI to evaluate the system for changes in the inputs to the controls while executing previous commands from the user.

5 MACHINING

Parts that were machined include the syringe tubes, compression rods, brass stoppers, PVC front/back blocks, linear push bar, and switch mounts. Prior to machining, tolerances were calculated to ensure there would be enough clearance between parts for assembly. Because the syringe tube, compression rods, and bolts diameters did not change, the tolerances for the brass stoppers and PVC blocks were calculated based on the need to fit around or inside of those diameters. Tolerances, nominal sizes, basic sizes, and actual sizes for each part machined can be found in Appendix C.

5.1 SYRINGE TUBES

The syringe tubes were purchased as 2 clear cast acrylic tubes of 3 feet in length. Using a DeWalt Single Bevel Sliding Compound Miter Saw with a standard carbide circular saw blade, the 2 tubes were cut into 4 tubes of 15 inches in length.

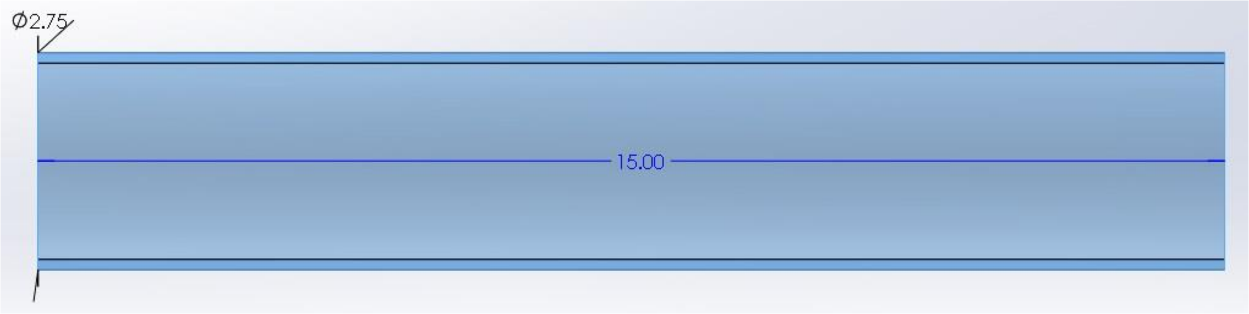


Figure 9 Dimensions of syringe tubes.

5.2 BRASS STOPPERS

The brass stoppers were purchased as a foot-long piece of brass stock with a starting diameter of 2.5 inches. This brass stock was cut into approximately 1.8-inch-long pieces using a DoALL Horizontal Band Saw. The brass stoppers were faced down to have a final width and diameter of 1.5 and 2.4 inches, respectively. This was done using a South Bend 14” Engine Lathe with a Single Point Brazed Carbide Tipped Tool Bit. Each side of the stopper was faced off approximately 0.10 inches to give it a smooth finish, and then the remaining length was taken off one side until the 1.5-inch width was reached. The diameter was faced off using the same tool.

Each stopper has two grooves for O-rings to sit in. To create these grooves, a Retaining Ring Grooving Tool was used. This tool had to be grinded down to fit the needed groove dimensions for depth (radius of 0.15 inches) and width (0.21 inches).

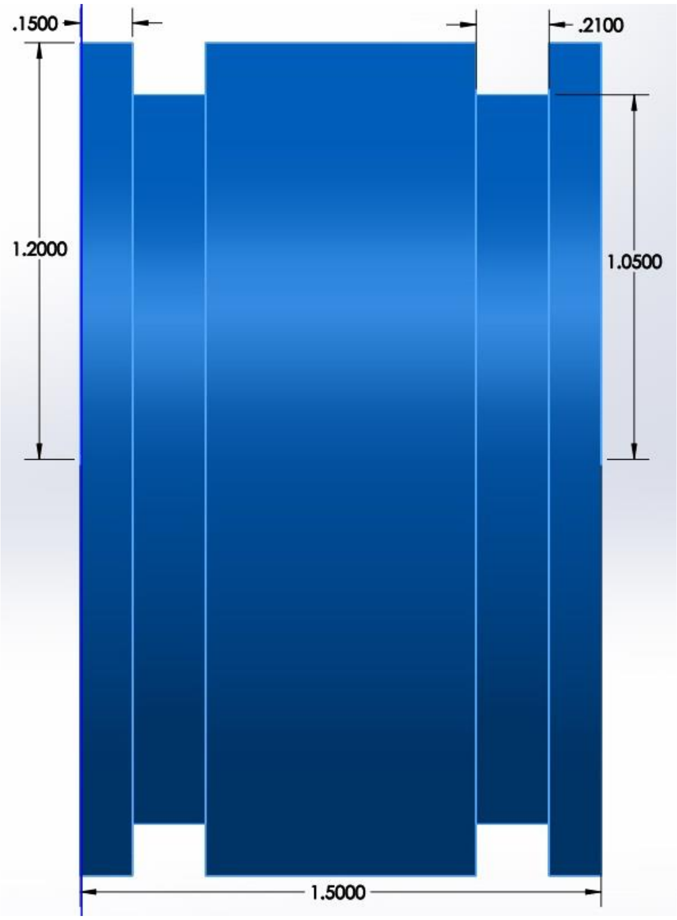


Figure 10 Outer dimensions of the brass stopper after machining.

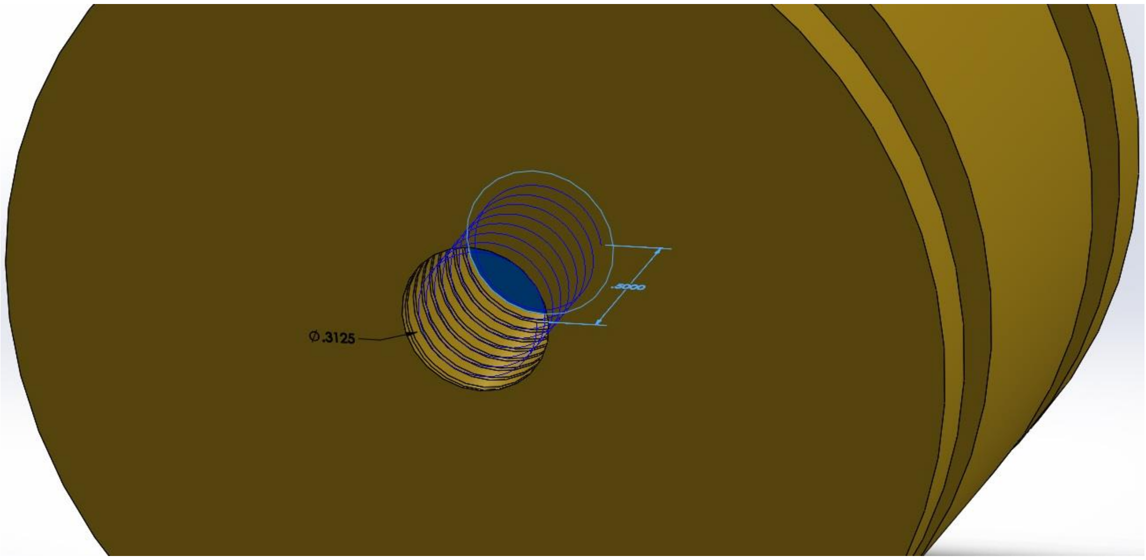


Figure 11 Dimensions of the threaded hole cut in the center of the front face of the brass stopper.

A hole was drilled into the front face of the stopper to allow it to connect to a plunging rod. The plunging rod is a 3/8 – 16 thread per inch stainless rod. To correctly drill a hole for the 3/8 – 16 thread per inch tap, a drill bit with a diameter of 5/16-inch was used to drill a hole with a diameter of 0.3125 inches and a depth of 0.5 inches. A 3/8 – 16 thread per inch tap was used to create the threads in the hole. The tap was inserted into the hole while the stopper was secured to the lathe in neutral gear. Then the stopper was placed in a vice and the remaining threads were created by manual rotation.

Figure 126 Finished brass stoppers.

5.3 PVC BLOCK FRONT/BACK

The PVC was purchased as one block with dimensions of 12x24x3 inches. It was cut into two blocks with dimensions of 5x24x3 inches using a table saw. Each block was placed on the work table of a Birmingham Milling Machine parallel to its x-axis. Using an edge finder, the x and y axes were zeroed out from the top and left of the blocks. The block was then moved into the correct (x,y) position and a 3/8-inch drill bit was used in the z-direction to cut 8 holes on each PVC block. These holes allowed for bolts to attach the PVC block to the mobile cart.

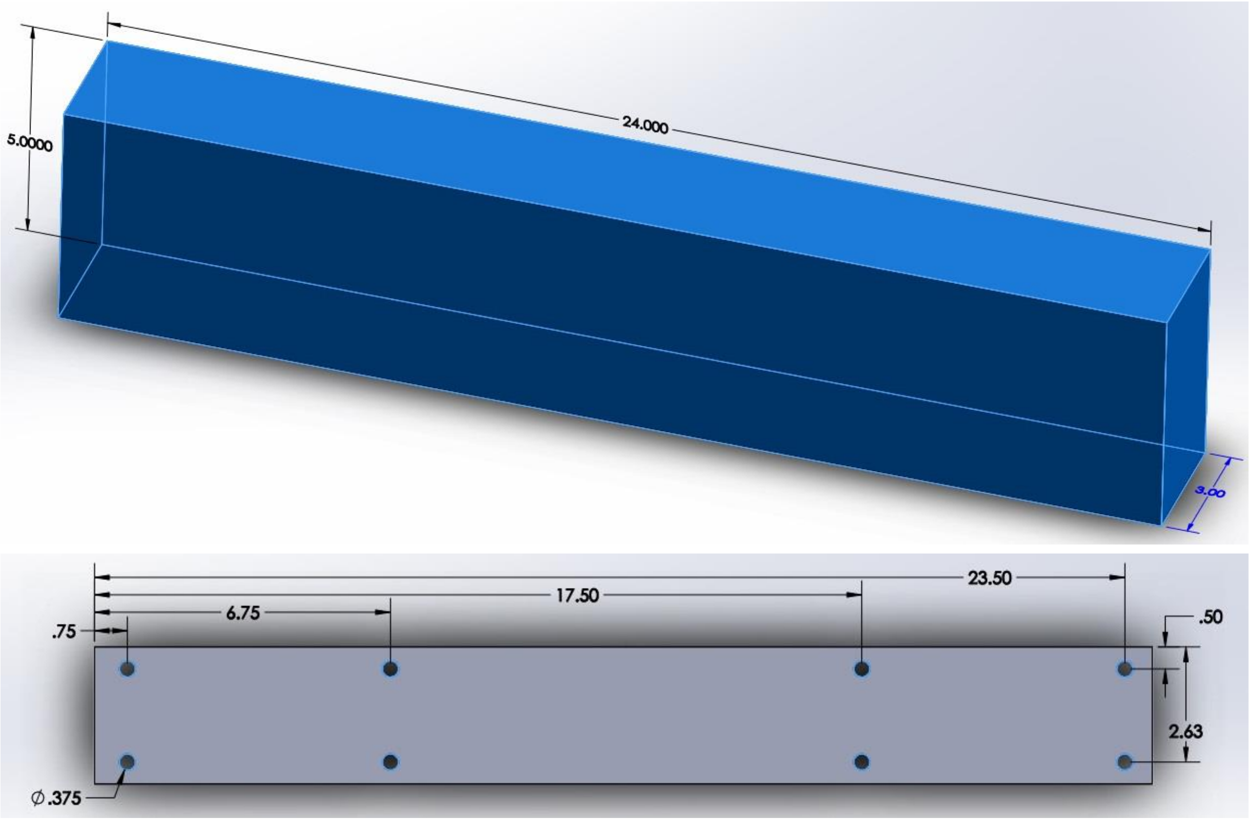


Figure 13 Dimensions of the 3/8 inch holes cut out from top to bottom on the PVC blocks.

The same method and drill bit were used to make 6 holes of diameter 0.375 inches from front to back. These holes were for the compression rods that were fixed with a nut on the front side of the front block and the back side of the back block causing the PVC blocks to compress and secure the syringe tubes.

Figure 7 Dimensions of the 3/8 inch holes cut from front to back on the PVC blocks.

Holes with a diameter of approximately 2.77 inches and depth of 1.5 inches were cut into the back side of the front block and the front side of the back block to hold the syringe tubes. The holes were started by using a ¾ inch end mill to drill down in the z-direction (at the correct (x,y) location) to a depth of 1.5 inches. A boring bar was then used to widen the diameter of each hole. Each pass of the boring bar widened the diameter by 0.01 inches. The (x,y) location for the center point of each syringe hole was found by zeroing out the x and y axes from the top and left of the block.

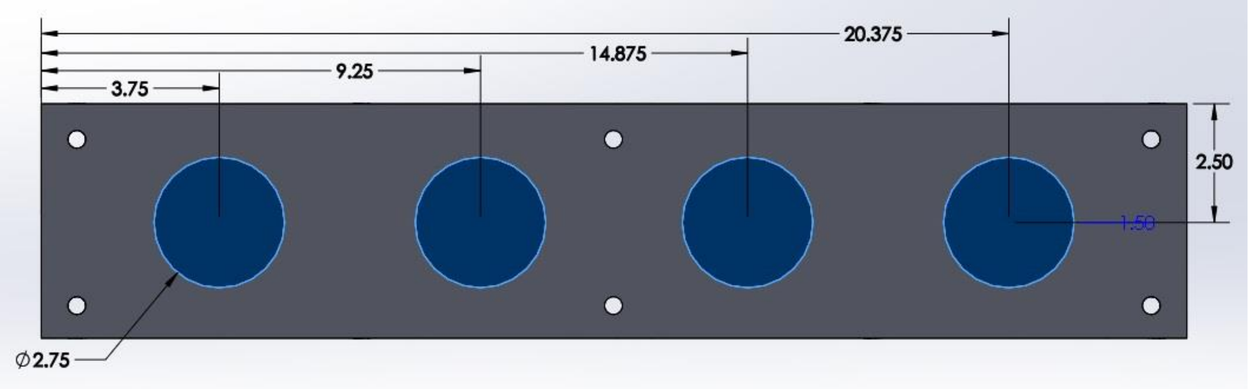


Figure 15 Dimensions of the holes cut to fit the syringe tubes.

On the Front block, 4 holes were cut out to allow for the plunging rods to easily slide through when pushing and pulling the brass stoppers. The same method was used to find the correct (x,y) location and a 5/8-inch end mill was used to cut the hole through all material from front to back.

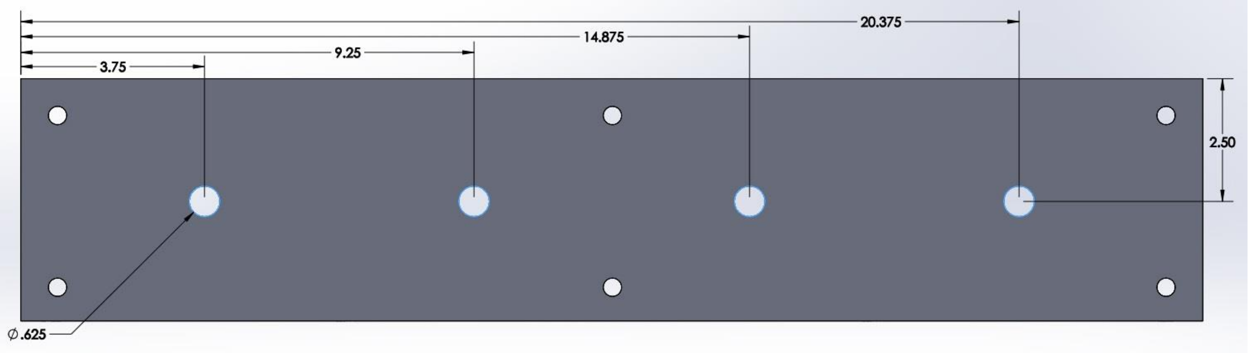


Figure 106 Dimensions for the holes cut on the Front PVC block to allow for the back and forth movement of the plunging rods.

Holes were cut out on the Back PVC block to allow for the attachment of ¼ NPT Valves. This was done by drilling through the block on the front face using a 7/16-inch drill bit and tapping them using a ¼ NPT threaded tap to the depth of approximately 2/5-inch. On the front opening of the valve holes, a ½-inch drill bit was used to smooth the edges allowing for a smoother transition of the fluid from the syringe into the valve openings.

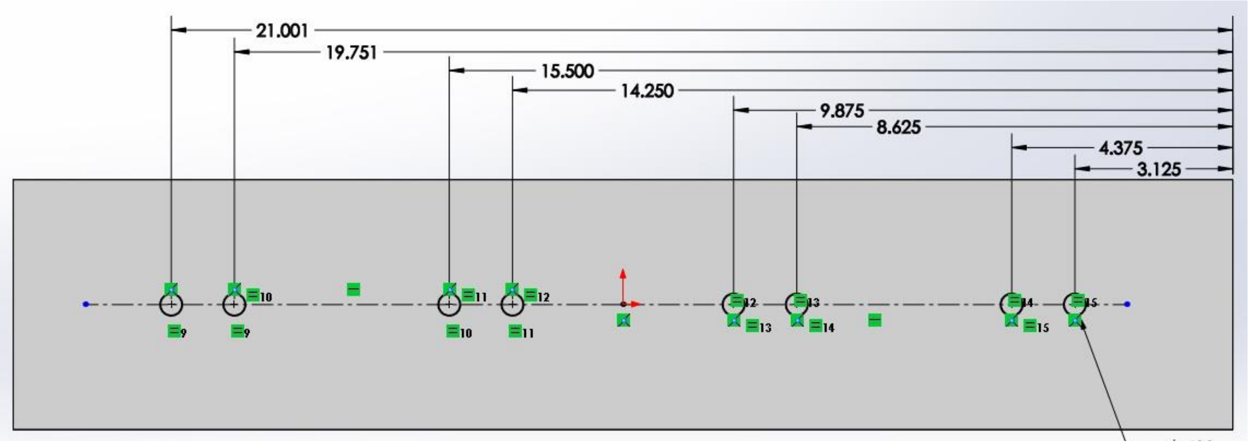


Figure 87 Dimensions for the tapped holes for valve attachment on the Back PVC block.

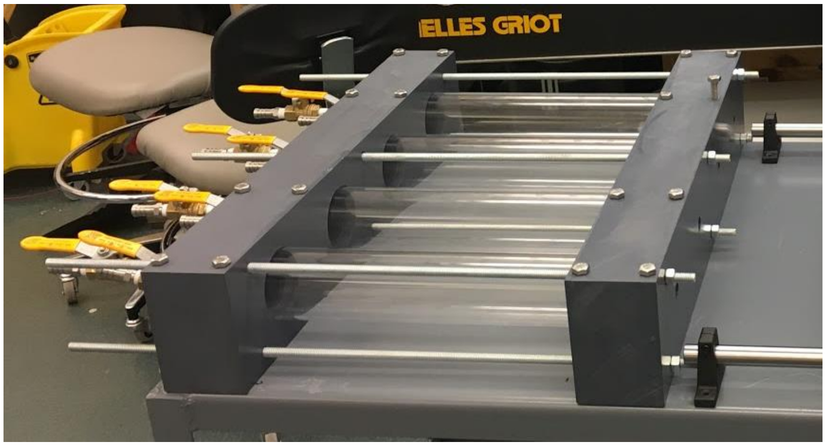


Figure 98 Finished Front and Back PVC blocks.

5.4 LINEAR PUSH BAR

A push-bar for the linear actuation system was created from a bar stock of aluminum of the size 3x24x3 inches. This was cut down first using a 1-inch end mill rougher to get it close to the correct dimensions and then each side was finished off with a fly cutter. The fly cutter was more precise in how much material it would shave off each pass giving each face a smooth finish.

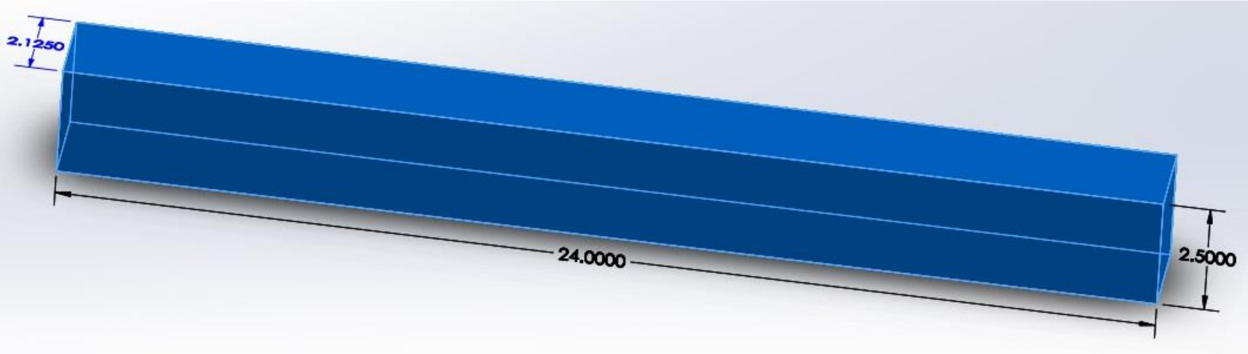


Figure 119 Dimensions for the height, width, and length of the push bar.

The push-bar was designed to sit on top of linear ball bearings that would allow it to move forward and backward with little to no impedance. A ½-inch end mill was used to carve out the ends of the bar to the correct x and z dimensions that would allow the bar to sit appropriately on the bearings. Four ¼-inch holes were drilled on each end to secure the bearings to the bar.

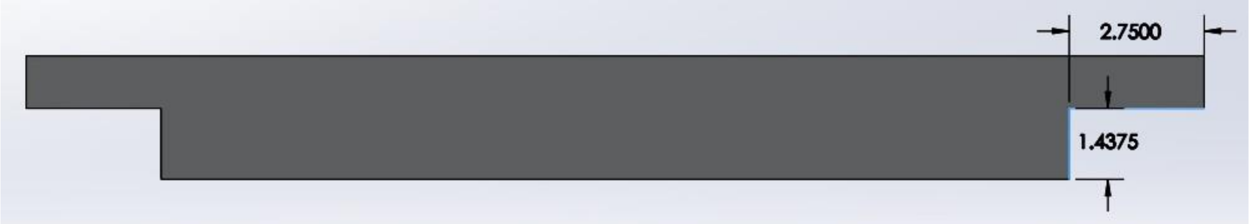


Figure 13 Dimensions for ball bearing cutouts.

Figure 131 Dimensions for ball bearing attachment.

The linear push-bar connects the motor's lead screw with the plunging rods of the syringes. Holes for the lead screw and four syringe plungers were drilled from front to back using a 7/16-inch and ¾-inch drill bit, respectively.

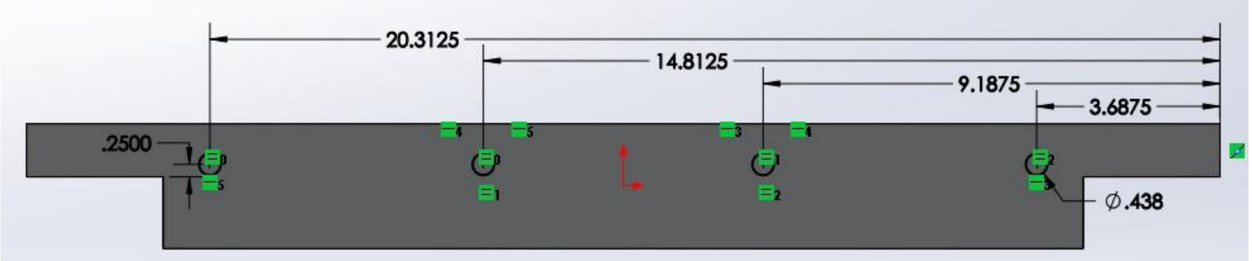


Figure 162 Dimension for syringe holes.

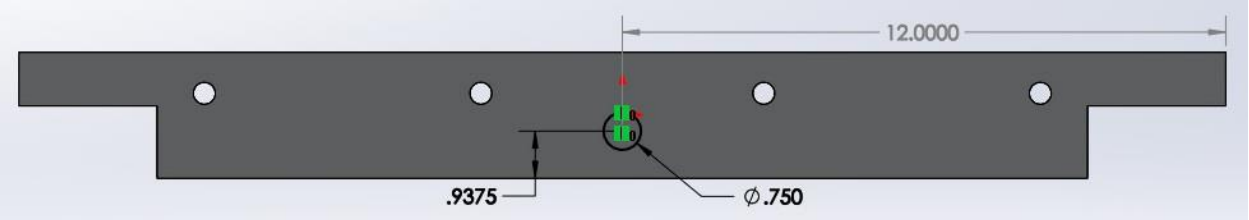


Figure 153 Dimension for lead screw hole.

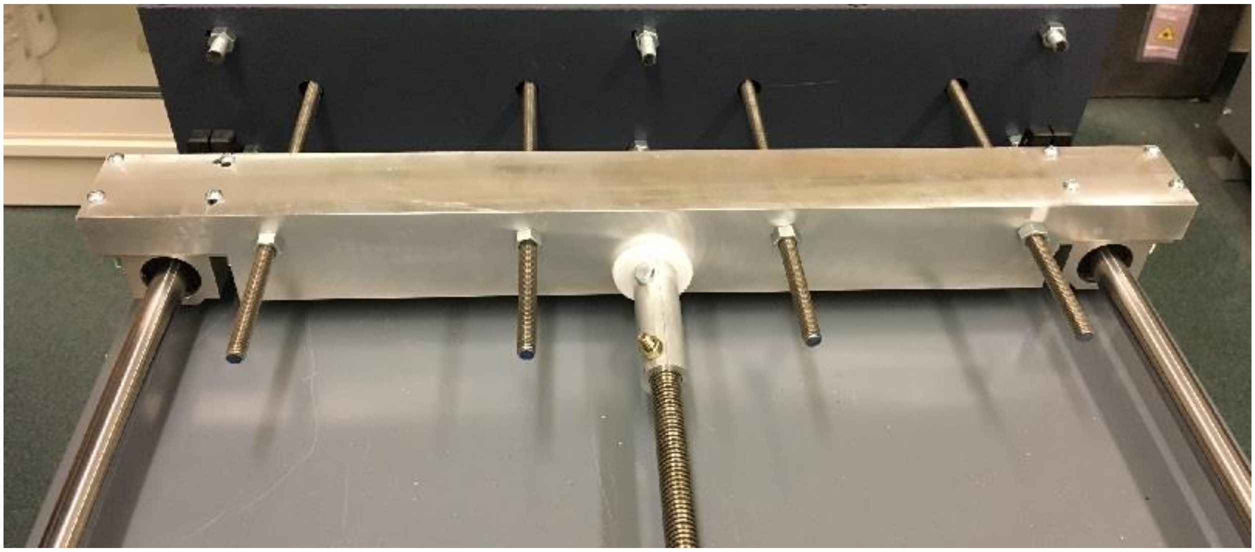


Figure 144 Finished linear push bar.

5.5 SWITCH MOUNTS

Mounts for the switches were made out of PVC. Using a saw each mount was cut to the dimensions of approximately 1.5x2.5x1.5 inches. An end mill was used to carve out part of the PVC to allow for clearance of the linear push bar and a 3/8-inch drill bit was used to drill a hole into the platform to allow for attachment to the cart.

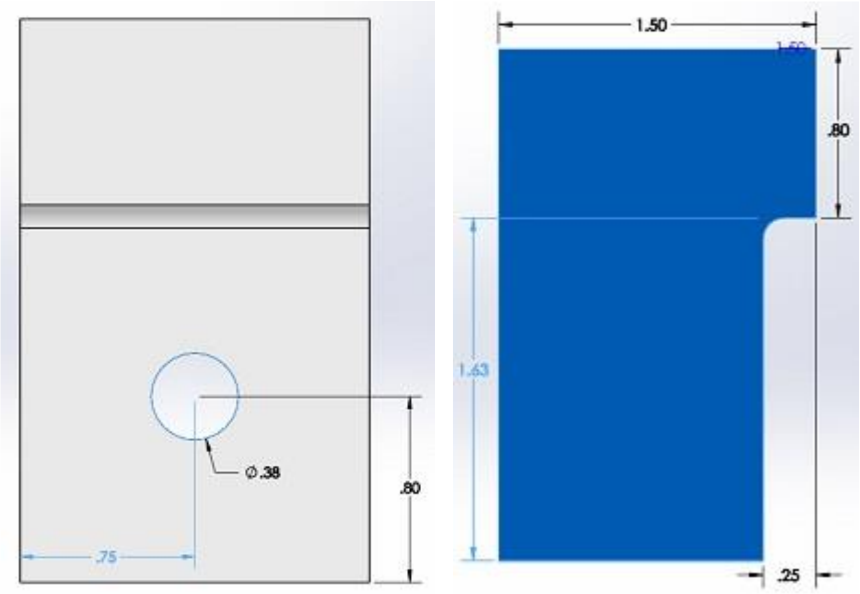


Figure 175 Front and Side views of the dimensions of the switch mounts.

5.6 COUPLING

To elongate the lead-screw a coupling was made. This coupling consisted of three parts: the back coupling, front coupling, and coupling rod. The front and back coupling were made from aluminum bar stock with an original length and diameter of 6 inches and 1 inch, respectively. This was cut into two pieces using a band saw. The front couple was cut to the dimensions of 3 inches and the back was cut to 0.5 inches. Using a lathe each was faced down to the exact length.

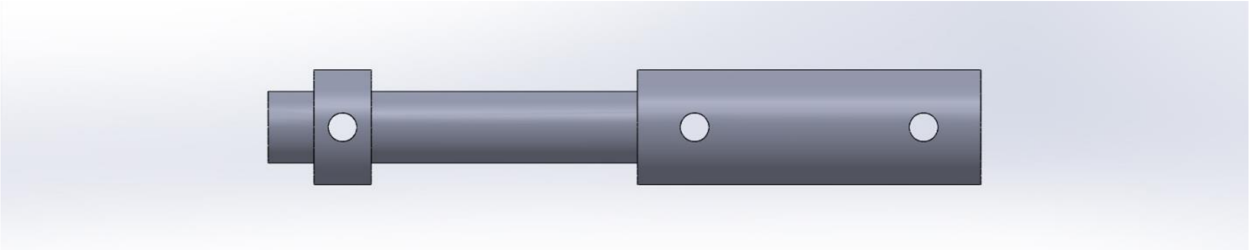


Figure 186 The coupling consisting of the back couple, coupling rod, and front couple (from left to right).

To allow for the coupling rod to fit inside of the front and back couple, a 21/32-inch drill bit was used to hollow the inside to an internal diameter slightly larger than the diameter of the coupling rod. The coupling rod was cut to 5 inches in length using the band saw. All parts of

Figure 197 Finished coupling. (Front piece on the left and Back piece on the right).

the coupling had holes drilled using a 9/32-inch drill bit in the vertical direction. These holes lined up so that the coupling pieces could be connected.

5.7 COMPRESSION RODS

The rods used to compress the front and back PVC blocks together were originally 3/8-inch – 16 threads per inch by 2 feet long threaded rods. Using a single bevel sliding compound miter saw with a high performance aluminum oxide blade, they were cut down to 20 inches in length.

5.8 ADDITIONAL MACHINING

Additional machining needs to be completed to allow the system to be fully operational for future use. This machining includes:

- Mount to secure the motor.
- Update linear push bar to allow for the couple to be pinned internally to it.
- Mount to hold the digital controls under the top shelf of the mobile cart.

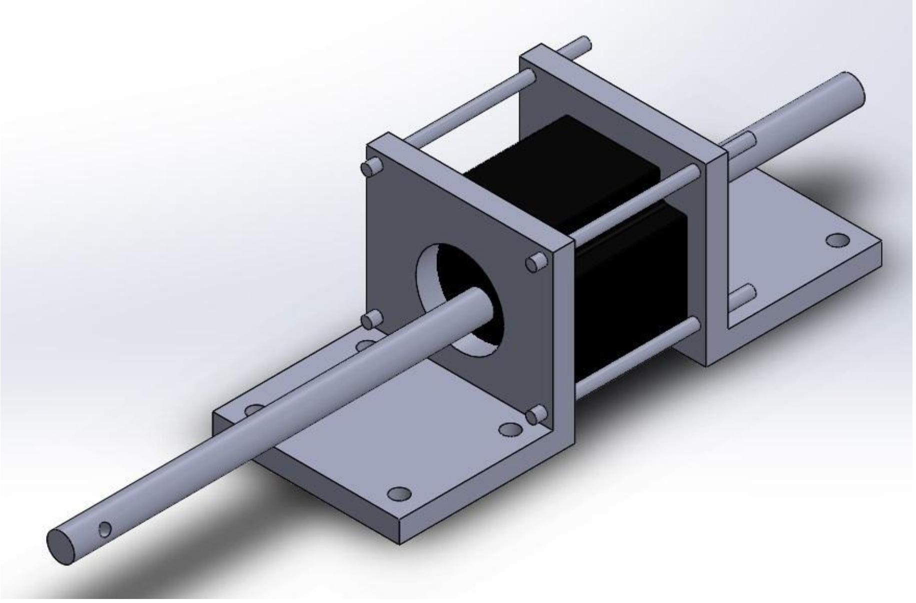


Figure 20 Newly designed motor mount.

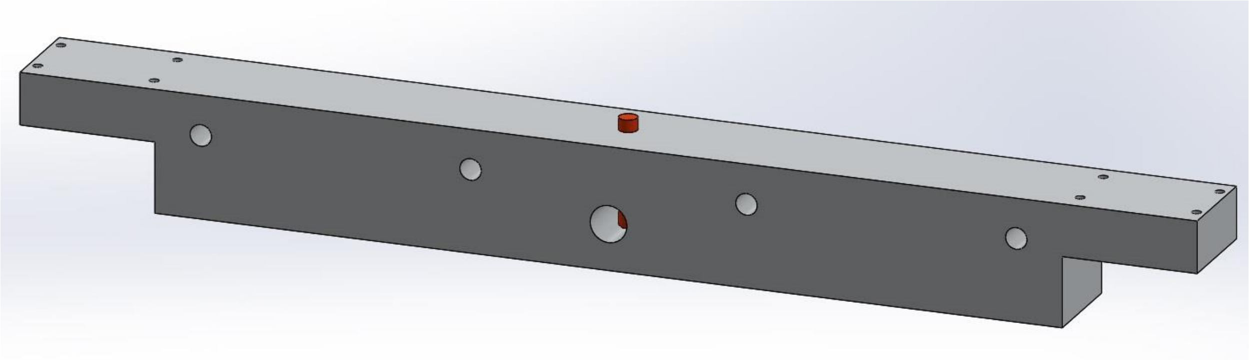


Figure 29 The updated pinning design for the push bar.

6 ASSEMBLY

All assembly took place in the Experimental Fluids Physics Laboratory at Oklahoma State University (ATRC 150). The assembly occurred in four steps: syringes, linear actuation system, motor, and digital controls.

Once components for each system were completed, securing them to the cart occurred. Holes were drilled throughout the cart based on where each component needed to be placed through the use of a DeWalt hand drill.

The first portion to be assembled was the syringe system. Gasket material was cut out and placed on the interior syringe tube-hole sections in the PVC blocks, allowing for sealing of the syringes. The front PVC block was secured to the cart initially, allowing for placement of the syringe tubes. Stoppers with O-rings placed into their trenches were set at an equal position into the tube so fluid would evenly be injected. A thin layer of Silicon grease was placed on the stopper in between the two O-rings to provide assistance in the stopper's ability to move throughout the syringe. Plunger rods were secured into their threaded holes. Syringe tubes with inserted plungers and plunger rods were placed into their perspective holes, allowing for securing of the back PVC block. Valves were sealed using threaded seal tape along each male and female sections, and secured into their threaded holes in the front PVC block.

The second portion assembled was the linear actuating system. Supports for the rod the ball bearings move linearly through were secured so that the motion paralleled the syringe tubes. These were placed in a position close to the PVC block so that the motor would have room to be mounted, as well as be able to achieve the full discharge of fluid from the syringe tubes. Once the push-bar and ball bearings were placed into position, the coupling system for the linear actuator was secured using nuts, bolts, and washers.

The next two sections assembled were the motor and digital controls. The motor mount was secured to the motor and cart, allowing for testing to begin. Digital controls were previously assembled, but were paired with the secured motor. Once all sections were assembled, initial testing could begin.

Note: A user manual for the system will be created, in which will be detailed instructions on how to assemble and disassemble the system.

7 CALIBRATION

An initial calibration of the system was done to evaluate the limits of the motor and verify that the relationship between motor input frequency and plunging velocity was linear. The calibration was done using a stopwatch and tape measure to measure the amount of time it would take the plungers to move a distance of 1-inch. No fluid was used during this calibration.

The cylindrical volume per inch was found to be 0.3217 liters using the equation

$$V = \pi r_i^2 h.$$

This volume per inch was then divided by the time it took the plunger to move an inch to get the injection rate in liters per second. Table 6 shows the resulting injection rates based on various input motor frequencies and Figure 30 illustrates the relationship between the two.

Initial System Calibration			
Frequency (Hz)	Time Elapsed (sec)	Injection Rate (L/s)	Injection Rate (Q_s)
2000	14	0.023	2
4000	8.5	0.038	≈ 4
8000	5.75	0.056	< 6
10000	4.7	0.068	> 6
15000	4	0.08	8

Table 5 Calibration results from the testing of different motor input frequencies.

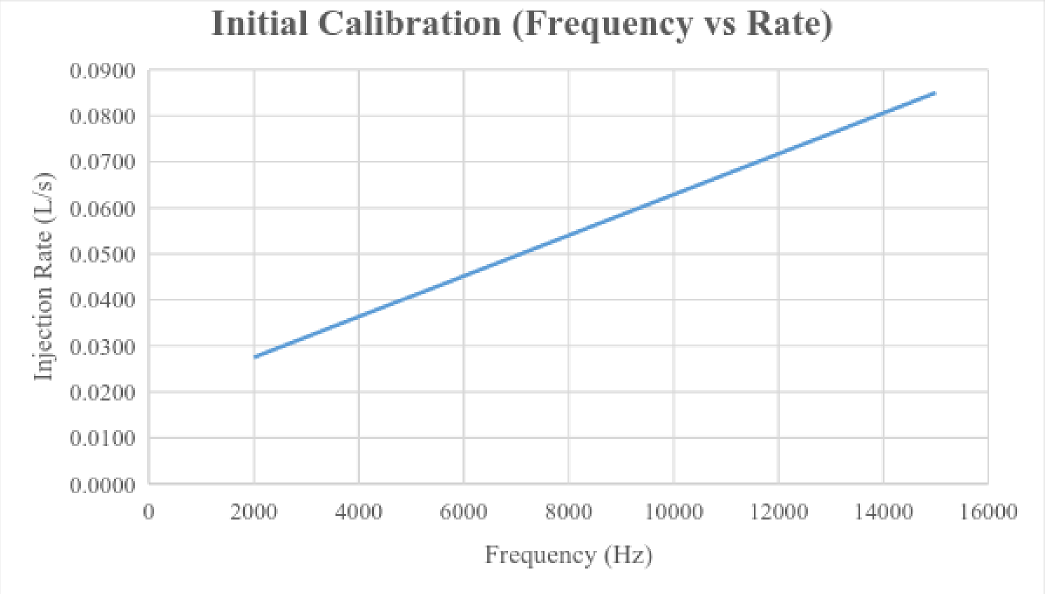


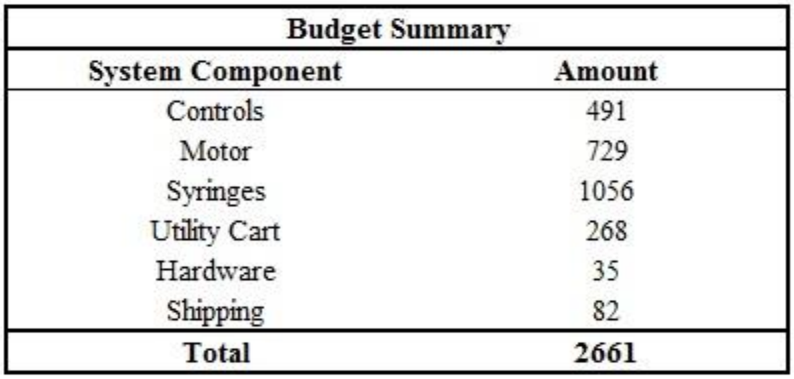
Figure 3021 Graphical representation of the relationship between input motor frequency and the resulting injection rate.

Tests run with motor frequencies over 15,000 hertz caused the motor to stall and not actuate the lead screw. Based on the results, the maximum attainable injection rate is $8Q_s$. Possible changes to the system will be implemented to allow the maximum injection rate to meet the $10Q_s$ requirement.

8 MANAGEMENT PLAN

8.1 COST

The budget for this project increased from $1500 to approximately $2600. This increase was approved by Dr. Elbing. A summarized budget can be seen in Table 7 and a more detailed itemized budget can be found in Appendix A.



Budget Summary	
System Component	**Amount**
Controls	491
Motor	729
Syringes	1056
Utility Cart	268
Hardware	35
Shipping	82
Total	**2661**

Table 6 Summarized budget for the major components of the system.

8.2 SCHEDULE

The revised schedule can be seen in Figure 31. All tasks with the exception of calibrating the system were complete by the deadline of May 10, 2017. The green light was given by Dr. Elbing to calibrate the system after the overall due date for project completion. The calibration is scheduled to be finished by May 17, 2017.

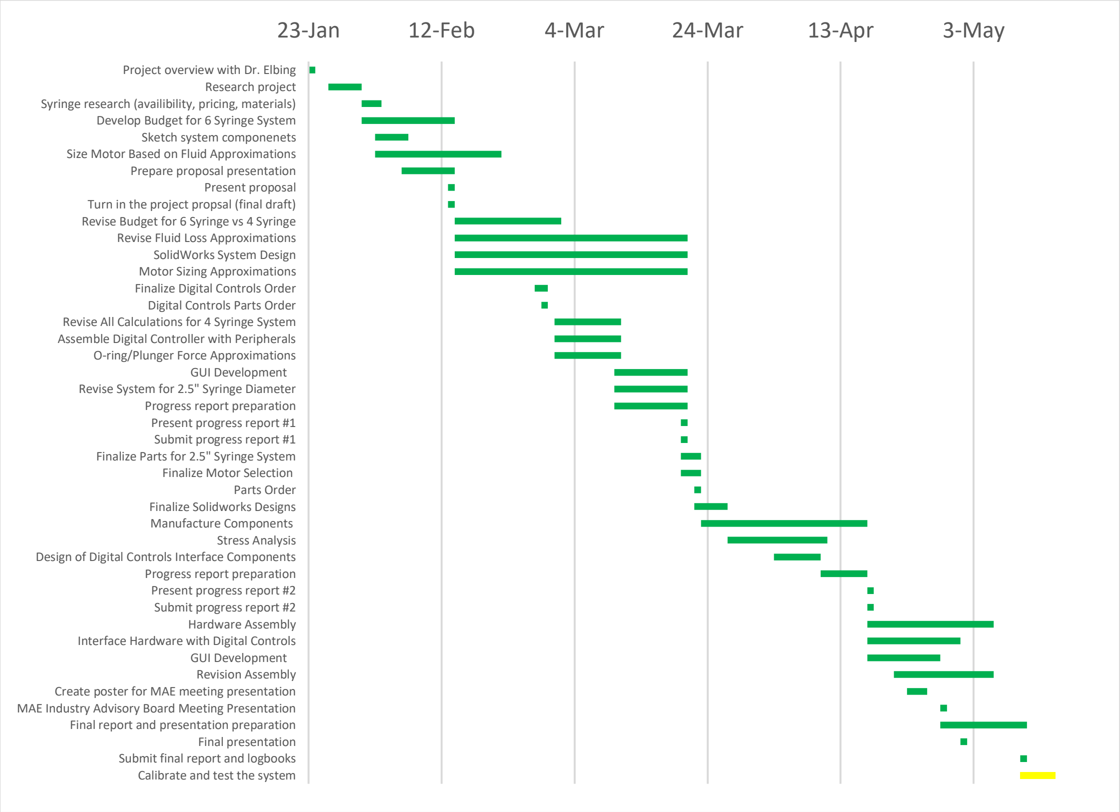


Figure 31 Schedule for the entirety of the project.

9 CONCLUSIONS

- The system was placed on a mobile cart to allow for mobility.
- The digital controls run through the Raspberry Pi were able to actuate the motor correctly.
- The system was assembled and an initial calibration was done to see the relationship between the input frequency and injection rate. Further testing needs to be done to better define this relationship.
- Through the calibration the maximum injection rate was found to be $8Q_s$.
- Changes will be made to the assembly and digital controls to complete the system.
- A user manual will be created to allow for ease of use for future users.

REFERENCES

Budynas, Richard G., J. Keith. Nisbett, and Joseph Edward. *Shigley's Mechanical Engineering Design*. 10th ed. New York: McGraw-Hill, 2011.

Elbing, et al. "Degradation of Homogeneous Polymer Solutions in High Shear Turbulent Pipe Flow". *Experiments in Fluids*. Pgs. 1033-1041. 2009.

http://www.americanfastener.com/tap-and-drill-size-chart/

Munson, Okiishi, Huebsch, Rothmayer. Fundamentals of Fluid Mechanics 7th Ed. Pgs. 200-728. John Wiley & Sons Inc. 2013.

Parker Engineering. "Friction Estimate". https://www.parker.com/literature/O-Ring%20Division%20Literature/Static%20Files/frictionestimation.pdf (Accessed March 3, 2017).

Virk, P.S., and H. Baher. "The Effect of Polymer Concentration on Drag Reduction." *Chemcial Engineerin Science*, 25(7): 1183-1189.

Winkel, et al. "High-Reynolds Number Turbulent Boundary Layer Friction Drag Reduction from Wall-Injected Polymer Solutions". *Journal of Fluid Mechanics*. Pgs. 259-273. 2009.

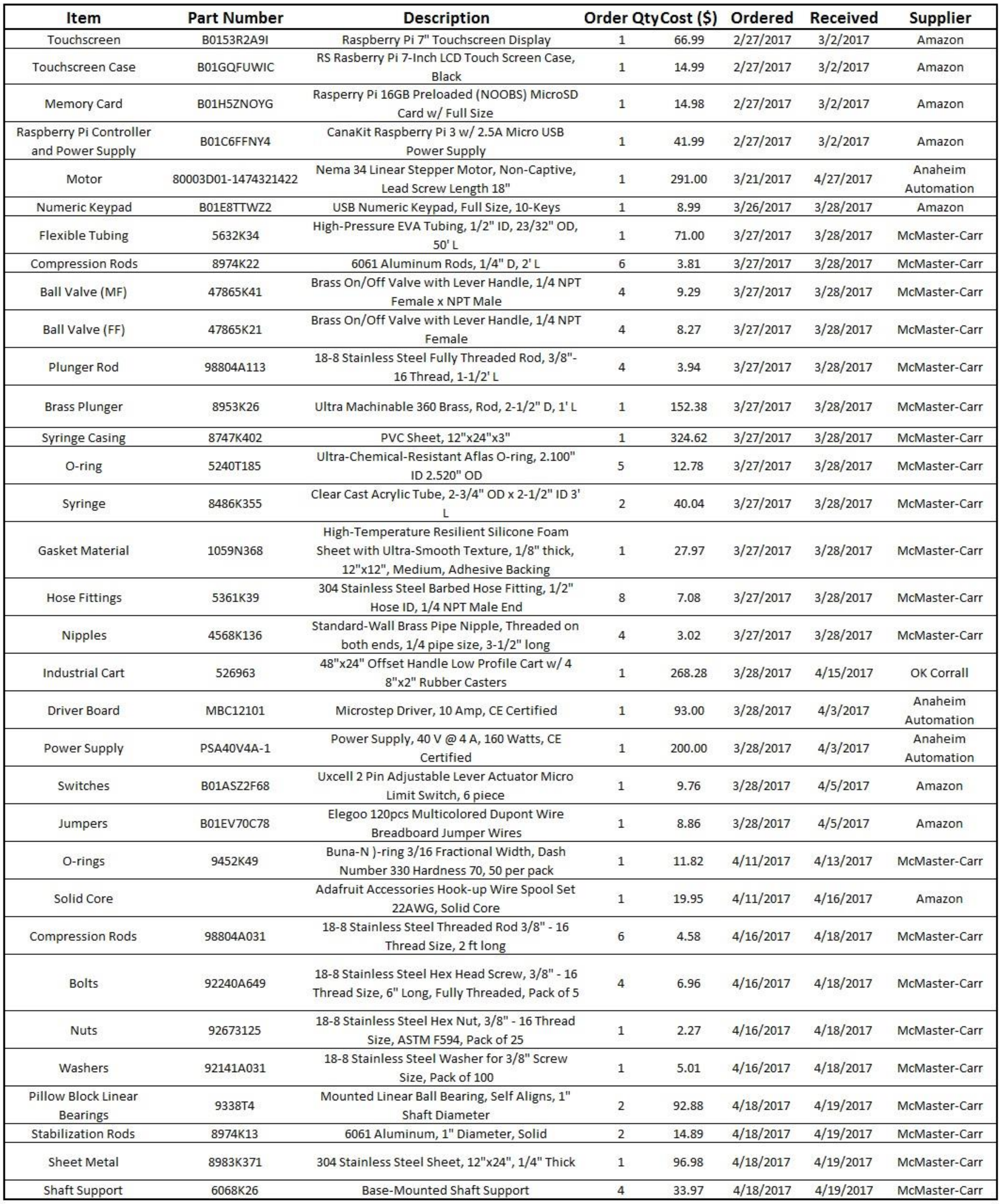
Wu & Tulin. "Drag Reduction by Ejecting Additive Solutions into a Pure-Water Boundary Layer. Hydronautics Inc. Pg. 8. 1970.

APPENDIX A: ITEMIZED BUDGET

Table 7 Itemized budget for each system component.

POLYMER INJECTION SYSTEM BUDGET		
Controls		
Raspberry Pi 3 (w/ power supply)	1	$45.00
Touchscreen display (7")	1	$70.00
Touchscreen case	1	$15.00
MicroSD card	1	$15.00
Jumpers/Resistors	1	$10.00
Solid Core, 22 AWG Wire Spools	1	$19.91
Motor Power Supply	1	$200.00
Switches	6	$15.00
Numeric Keypad	1	$8.99
Driver Board	1	$93.00
Controls Subtotal		**$491.90**
Motor		
Stepper Motor/Lead Screw	1	$291.00
Teflon Washers	4	$17.64
Pillow Block Linear Bearings	2	$133.20
Stabilization Rods	2	$42.14
Aluminum Bar	1	$105.55
Bolts for Bearings	1	$8.43
Nuts for Bearings	1	$3.50
Washers for Bearings	1	$5.04
Shaft Support	4	$122.92
Motor Subtotal		**$729.42**
Syringes		
Cast acrylic tube	2	$120.12
O ring	50 per pack	$11.82
PVC	1	$324.62
Brass Plunger	1 ft	$152.38
Plunger/Metal rod	4	$3.94
Ball Valve (FF)	4	$8.27
Ball Valve (MF)	4	$9.29
Nipples	4	$3.02
Hose Fittings	8	$7.08
Compression Rods	6	$4.58
Flexible tubing	1	$70.00
Gasket material	1	$27.97
Syringes Subtotal		**$1,056.51**
Utility Cart		
Cart	1	$268.28
Cart Subtotal		**$268.28**
Hardware		
Bolts	4	$6.96
Nuts	1	$2.27
Washers	1 bag of 50	$5.01
Hardware Subtotal		**$35.12**
Shipping & Handling		
Anaheim Automation		$16.30
McMaster-Carr		$44.04
McMaster-Carr		$6.32
Anaheim Automation		$15.37
Shipping Subtotal		**$82.03**
Returns		
Compression Rods	6	$3.81
Plunging Rods	4	$3.94
O-rings	5	$12.78
Returns Subtotal		**-$102.52**
Initial Budget		$1,500.00
Total Cost		$2,560.74

APPENDIX B: PURCHASE ORDER



Item	Part Number	Description	Order Qty	Cost ($)	Ordered	Received	Supplier
Touchscreen	B0153R2A9I	Raspberry Pi 7" Touchscreen Display	1	66.99	2/27/2017	3/2/2017	Amazon
Touchscreen Case	B01GQFUWIC	RS Rasberry Pi 7-Inch LCD Touch Screen Case, Black	1	14.99	2/27/2017	3/2/2017	Amazon
Memory Card	B01H5ZNOYG	Rasperry Pi 16GB Preloaded (NOOBS) MicroSD Card w/ Full Size	1	14.98	2/27/2017	3/2/2017	Amazon
Raspberry Pi Controller and Power Supply	B01C6FFNY4	CanaKit Raspberry Pi 3 w/ 2.5A Micro USB Power Supply	1	41.99	2/27/2017	3/2/2017	Amazon
Motor	80003D01-1474321422	Nema 34 Linear Stepper Motor, Non-Captive, Lead Screw Length 18"	1	291.00	3/21/2017	4/27/2017	Anaheim Automation
Numeric Keypad	B01E8TTWZ2	USB Numeric Keypad, Full Size, 10-Keys	1	8.99	3/26/2017	3/28/2017	Amazon
Flexible Tubing	5632K34	High-Pressure EVA Tubing, 1/2" ID, 23/32" OD, 50' L	1	71.00	3/27/2017	3/28/2017	McMaster-Carr
Compression Rods	8974K22	6061 Aluminum Rods, 1/4" D, 2' L	6	3.81	3/27/2017	3/28/2017	McMaster-Carr
Ball Valve (MF)	47865K41	Brass On/Off Valve with Lever Handle, 1/4 NPT Female x NPT Male	4	9.29	3/27/2017	3/28/2017	McMaster-Carr
Ball Valve (FF)	47865K21	Brass On/Off Valve with Lever Handle, 1/4 NPT Female	4	8.27	3/27/2017	3/28/2017	McMaster-Carr
Plunger Rod	98804A113	18-8 Stainless Steel Fully Threaded Rod, 3/8"-16 Thread, 1-1/2' L	4	3.94	3/27/2017	3/28/2017	McMaster-Carr
Brass Plunger	8953K26	Ultra Machinable 360 Brass, Rod, 2-1/2" D, 1' L	1	152.38	3/27/2017	3/28/2017	McMaster-Carr
Syringe Casing	8747K402	PVC Sheet, 12"x24"x3"	1	324.62	3/27/2017	3/28/2017	McMaster-Carr
O-ring	5240T185	Ultra-Chemical-Resistant Aflas O-ring, 2.100" ID 2.520" OD	5	12.78	3/27/2017	3/28/2017	McMaster-Carr
Syringe	8486K355	Clear Cast Acrylic Tube, 2-3/4" OD x 2-1/2" ID 3' L	2	40.04	3/27/2017	3/28/2017	McMaster-Carr
Gasket Material	1059N368	High-Temperature Resilient Silicone Foam Sheet with Ultra-Smooth Texture, 1/8" thick, 12"x12", Medium, Adhesive Backing	1	27.97	3/27/2017	3/28/2017	McMaster-Carr
Hose Fittings	5361K39	304 Stainless Steel Barbed Hose Fitting, 1/2" Hose ID, 1/4 NPT Male End	8	7.08	3/27/2017	3/28/2017	McMaster-Carr
Nipples	4568K136	Standard-Wall Brass Pipe Nipple, Threaded on both ends, 1/4 pipe size, 3-1/2" long	4	3.02	3/27/2017	3/28/2017	McMaster-Carr
Industrial Cart	526963	48"x24" Offset Handle Low Profile Cart w/ 4 8"x2" Rubber Casters	1	268.28	3/28/2017	4/15/2017	OK Corrall
Driver Board	MBC12101	Microstep Driver, 10 Amp, CE Certified	1	93.00	3/28/2017	4/3/2017	Anaheim Automation
Power Supply	PSA40V4A-1	Power Supply, 40 V @ 4 A, 160 Watts, CE Certified	1	200.00	3/28/2017	4/3/2017	Anaheim Automation
Switches	B01ASZ2F68	Uxcell 2 Pin Adjustable Lever Actuator Micro Limit Switch, 6 piece	1	9.76	3/28/2017	4/5/2017	Amazon
Jumpers	B01EV70C78	Elegoo 120pcs Multicolored Dupont Wire Breadboard Jumper Wires	1	8.86	3/28/2017	4/5/2017	Amazon
O-rings	9452K49	Buna-N)-ring 3/16 Fractional Width, Dash Number 330 Hardness 70, 50 per pack	1	11.82	4/11/2017	4/13/2017	McMaster-Carr
Solid Core		Adafruit Accessories Hook-up Wire Spool Set 22AWG, Solid Core	1	19.95	4/11/2017	4/16/2017	Amazon
Compression Rods	98804A031	18-8 Stainless Steel Threaded Rod 3/8" - 16 Thread Size, 2 ft long	6	4.58	4/16/2017	4/18/2017	McMaster-Carr
Bolts	92240A649	18-8 Stainless Steel Hex Head Screw, 3/8" - 16 Thread Size, 6" Long, Fully Threaded, Pack of 5	4	6.96	4/16/2017	4/18/2017	McMaster-Carr
Nuts	92673125	18-8 Stainless Steel Hex Nut, 3/8" - 16 Thread Size, ASTM F594, Pack of 25	1	2.27	4/16/2017	4/18/2017	McMaster-Carr
Washers	92141A031	18-8 Stainless Steel Washer for 3/8" Screw Size, Pack of 100	1	5.01	4/16/2017	4/18/2017	McMaster-Carr
Pillow Block Linear Bearings	9338T4	Mounted Linear Ball Bearing, Self Aligns, 1" Shaft Diameter	2	92.88	4/18/2017	4/19/2017	McMaster-Carr
Stabilization Rods	8974K13	6061 Aluminum, 1" Diameter, Solid	2	14.89	4/18/2017	4/19/2017	McMaster-Carr
Sheet Metal	8983K371	304 Stainless Steel Sheet, 12"x24", 1/4" Thick	1	96.98	4/18/2017	4/19/2017	McMaster-Carr
Shaft Support	6068K26	Base-Mounted Shaft Support	4	33.97	4/18/2017	4/19/2017	McMaster-Carr

Table 9 Purchase order for system.

APPENDIX C: TOLERANCES

		Nominal Size*				Basic Size					
Part	Dimension	Size	Tolerance	MMC	LMC	Size	Tolerance	MMC	LMC	Actual Size**	Additional Notes
Syringe Tube	Outer Diameter	2.75"	±0.02"	2.77"	2.73"					2.749"	No change
Syringe Tube	Wall Thickness	0.125"	±0.019	0.144"	0.106"					0.130"	No change
Syringe Tube	Inner Diameter	2.50"	±0.019"	2.481"	2.519"					2.489"	No change
Syringe Tube	Length	36.0"	±1.00"	37.0"	35.0"	15.0"	±0.10"	15.10"	14.90"	15.05"	
PVC	Thickness	3.0"	+0.600"	3.600"	3.00"					3.125"	No change
PVC	Length	24.0"	+0.25"	24.25"	24.0"					24.125"	No change
PVC	Height	12.0"	+0.25"	12.25"	12.0"	5.0"	+0.125"	5.125"	5.0"	4.9375"	
PVC - Syringe	Diameter					2.75"	+0.05"	2.75"	2.80"	2.77"	
PVC - Syringe	Depth					1.50"	±0.10"	1.40"	1.60"	1.53"	
PVC - 3/8" Compression Rods	Diameter					0.375"	+0.006"	0.369"	0.381"	0.375"	
PVC - 3/8" Bolts	Diameter					0.375"	+0.02"	0.375"	0.377"	0.375"	
PVC - Plunger Rod	Diameter					0.625"	±0.05"	.575"	0.675"	0.625"	
PVC - Valve Fittings	Diameter					0.4375"	±0.005"	0.4325"	0.4425	Not Complete	Threaded opening
Brass	Outer Diameter	2.50"	±0.006"	2.506"	2.494"	2.40"	±0.05"	2.45"	2.35"	2.398"	
Brass	Length	12.0"	±0.500"	12.5"	11.5"	1.5"	±0.05"	1.55"	1.45"	1.498"	
Brass - O-ring Indention	Width					0.210"	+0.005"	0.210"	0.215"	0.205"	
Brass - O-ring Indention	Inner Diameter					2.100"	+0.005"	2.100"	2.105"	2.108"	
Brass - Threaded Hole	Depth					0.50"	±0.025"	0.475"	0.525"	0.515"	
Brass - Threaded Hole -Drill Bit	Diameter					0.3125"				0.3125"	Threaded opening
Compression Rods	Outer Diameter	0.375"									Threaded rod - 16 threads/inch
Compression Rods	Length	24.0"	±1.00"	25.0"	23.0"	20"	±0.25"	20.25"	19.75"	20.125"	
Gasket Material	Thickness	0.125"	±0.02"	0.145"	0.105"						
Gasket Material	Width	12.0"	±0.313"	12.313"	11.687"						
Gasket Material - Syringe	Outer Diameter					2.75"				2.75"	

*Measurements given by McMaster-Carr

**Averaged between number of part

Table 10 Tolerances for machined parts.

APPENDIX D: FLUID FLOW CALCULATIONS

Onset of drag reduction was found using the equation

$$\gamma_D = (3.23 * 10^{18}) * MW^{-2.2}$$

The molecular weight of WSR308 is 8,000,000 grams per mole.

$$\gamma_D = (3.23 * 10^{18}) * (8 * 10^6)^{-2.2} = 2100 \; liters \; per \; second$$

$$\tau_D = \gamma_D \rho \nu = 2100(998)(1.01 * 10^{-6}) = 2.1 \; Pa$$

Multiple iterations for the velocity were done by examining at which velocity the Prandtl-Von Karman equation would converge.

$$\frac{1}{\sqrt{f^*}} = 4 \log_{10}\left(Re\sqrt{f^*}\right) - 0.4$$

$$Re = \frac{UD}{\nu} = \frac{0.73(0.0127)}{1.01 * 10^{-6}} = 9.18 * 10^3$$

$$f^* = \frac{2\tau_D}{\rho U^2} = \frac{2(2.1)}{998(0.73)^2} = 0.00796$$

The maximum injection rate for the polymer was found by

$$Q_{max} = AU_{max} = (1.267 * 10^{-4})(0.73) = 0.0925 \; liter \; per \; second$$

The minimum and maximum injection rates desired were as follows

$$Q_s = 67.3\nu = (67.3)(1.01 * 10^{-6})(0.1524) = 0.0104 \; liter \; per \; second$$

$$Injection \; Rate_{minimum} = 2Q_s = 0.0208 \; liter \; per \; second$$

$$Injection \; Rate_{maximum} = 10Q_s = 0.104 \; liter \; per \; second$$

The force to overcome the fluid pressure loss was solved by

$$\Delta P = f\left(\frac{L}{D}\right)\frac{\rho U^2}{2}$$

For syringe tube major losses at maximum injection rate $10Q_s$, volumetric flow rate is 1.04×10^{-4} m^3/s, diameter is 2.5 in or 0.06235 m, length is 0.3048 m, area is 0.00317 m^2, viscosity is 1.01×10^{-6} m^2/s, and density is 998 kg/m^3

$$\rightarrow V = \frac{Q}{A} = \frac{1.04x10^{-4}}{0.00317} = 0.033 \text{ m/s}$$

$$\rightarrow Re = \frac{0.033 * 0.0635}{1.01x10^{-6}} = 2077$$

$$\rightarrow f = \frac{64}{Re} = \frac{64}{2077} = 0.0308$$

$$\rightarrow \Delta P = f\left(\frac{L}{D}\right)\frac{\rho U^2}{2} = 0.0308 * \left(\frac{0.3048}{0.0635}\right) * \frac{998 * 0.033^2}{2} = 0.079 \text{ Pa}$$

For flexible tube major losses at maximum injection rate $10Q_s$, diameter is 0.5 in or 0.0127m, length is 3m, area is 0.00013 m^2, and absolute roughness of 0.07mm

$\rightarrow V = \frac{Q}{A} = \frac{1.04x10^{-4}}{0.00013} = 0.82 \text{ m/s}$

$\rightarrow Re = \frac{0.82*0.0127}{1.01x10^{-6}} = 10386$

$\rightarrow \frac{\varepsilon}{D} = \frac{0.07}{12.7} = 0.0055$

→ From Moody Chart at Reynolds number and roughness ratio: $f = 0.031$

$\rightarrow \Delta P = f\left(\frac{L}{D}\right)\frac{\rho U^2}{2} = 0.031 * \left(\frac{3}{0.0127}\right) * \frac{998*0.82^2}{2} = \frac{2444\ Pa}{(2990\frac{Pa}{ft})} = 0.817$ ft

$$\Delta P = K\frac{\rho U^2}{2}$$

For minor losses in the system at the maximum injection rate $10Q_s$, K=20 for valves, K=1.7 for carving, and K=0 for fittings

$\rightarrow \Delta P = K\frac{\rho U^2}{2} = (20 + 1.7) * \frac{998*0.82^2}{2} = 7575 \text{ Pa} = \frac{7575\ Pa}{(2990\frac{Pa}{ft})} = 2.533$ ft

$$F = PA$$

For total pressure in the system, total pressure drop due to major and minor losses is 3.35 ft, pressure drop due to elevation change is 5 ft, sum of these two is 3.59 psi, and the safety factor included multiplied 3.59 psi by 4, resulting in 14.36 psi; cross-sectional area is 4.91 in^2

$\rightarrow F = 14.36 * 4.91 = 70.5$ lbf

For four syringes F = 282 lbf.

Solving for force required to overcome O-ring friction

$$FC = f_c L_r$$

From Parker Engineering for 10% compression and 50-hardness, friction due to O-ring compression is 0.5; for 2.5” OD and 2.125” ID Dash 330 O-rings, length of seal rubbing surface is 6.67

$\rightarrow FC = f_c L_r = 0.5 * 6.67 = 3.335$ lbf

$$FH = f_h A_r$$

From Parker Engineering for 20psi fluid pressure, friction due to fluid pressure is 12; for 2.5” OD and 2.125” ID Dash 330 O-rings, seal area is 1.36

$\rightarrow FH = 12 * 1.36 = 16.32$ lbf

$$FF = FC + FH$$

$\rightarrow FF = 3.335 + 16.32 = 19.655$ lbf

For 2 O-rings per syringe or 8 O-rings for the entire system FF = 157.24 lbf.

APPENDIX E: MOTOR CALCULATIONS

The raising torque used to size the motor for the system was found using the equation

$$T_R = \frac{Fd_m}{2}(\frac{l + \pi f d_m}{\pi d_m - fl})$$

For the NEMA 34 Hybrid Linear Actuator, the lead and friction variables were based off dimensions listed on the Anaheim Automation's website. The lead (l) was given as 0.1 inches per revolution. The lead screw is made out of stainless steel and has a coefficient of friction (f) of approximately 0.25. The nominal diameter for the lead screw is 15.88 millimeters or 0.625 inches.

The threads per inch (N) was calculated by counting the number of threads per inch of the lead screw, it was approximately 10 threads per inch. Pitch was then calculated using the equation

$$p = \frac{1}{N} = \frac{1}{10} = 0.1\frac{inches}{thread}$$

Using the nominal major diameter and the pitch, the minor and mean diameters could be calculated.

$$D_{minor} = nominal\ major\ diameter - pitch = 0.625 - 0.1 = 0.525\ inches$$

$$D_{mean} = d_m = nominal\ major\ diameter - \frac{p}{2} = 0.625 - \frac{0.1}{2} = 0.575\ inches$$

The total thrust force (F) calculated in the fluid flow approximations was used to calculate the raising and lowering torques.

$$T_R = \frac{439.2(0.575)}{2}\left(\frac{l + \pi(0.25)(0.575)}{\pi(0.575) - (0.25)(0.1)}\right) = 39.10\ lb\ in = 626\ oz\ in$$

$$T_L = \frac{Fd_m}{2}\left(\frac{\pi f d_m - 1}{\pi d_m + fl}\right) = \frac{439.2(0.575)}{2}\left(\frac{\pi(0.25)(0.575) - 1}{\pi(0.575) + 0.25(0.1)}\right) = 24.24\ lb\ in = 388\ oz\ in$$

The raising and lowering torques were converted into ounce-inches and were used with the total thrust force to size a motor for the system.

APPENDIX F: SYRINGE CALCULATIONS

Calculating syringe tube thickness:

$$P = 14.4\, psi$$

$$\frac{thickness}{radius_{internal}} = \frac{t}{r_i} = \frac{0.25}{1.25} = 0.2$$

$$(\sigma_t) = P\left(\frac{r_o^2 + r_i^2}{r_0^2 - r_i^2}\right) = 14.4\left(\frac{1.375^2 + 1.25^2}{1.375^2 - 1.25^2}\right) = 151.54 \text{ psi}$$

$$(\sigma_r) = -P = -14.4\, psi$$

The chosen material was clear cast acrylic plastic. It has an outer diameter of 2.75 inches, an inner diameter of 2.5 inches, and a tensile strength of 8,000 psi.

$$8{,}000\, psi > 151.54\, psi$$

Because of this the 0.25-inch thickness tube satisfied the design requirements.

Calculating diameter of syringe plunging rods:

$$F = 439.2\, lb_f$$

$$l_p = 24\, in$$

$$E_{stainless\ steel} = 28e6$$

$$D_p = \frac{3}{8}\, in$$

Using the equation

$$F = \frac{n\pi^2 E I_y}{l_p^2} = 439.2 = \frac{4\pi^2(28 * 10^6)\left(\frac{\pi r_p^4}{4}\right)}{24^2}$$

Where

$$I_y = \frac{\pi r_p^4}{4}$$

$$r_p = 0.1306\, in$$

$$D_{p_{min}} = 2r_p = 0.2613\, in$$

Because 0.375 inches is greater than 0.2613 inches, the 3/8-inch diameter threaded rod of 18-8 Stainless Steel satisfied the design requirements.

APPENDIX G: DIGITAL CONTROLS CODE – GUI

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'GUI_Draft05.ui'
#
# Created by: PyQt4 UI code generator 4.11.4
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName(_fromUtf8("Form"))
        Form.resize(583, 361)
        font = QtGui.QFont()
        font.setPointSize(11)
        font.setBold(True)
        font.setWeight(75)
        Form.setFont(font)
        self.DepressButton = QtGui.QPushButton(Form)
        self.DepressButton.setGeometry(QtCore.QRect(420, 10, 141, 61))
        self.DepressButton.setObjectName(_fromUtf8("DepressButton"))
        self.DrawBackButon = QtGui.QPushButton(Form)
        self.DrawBackButon.setGeometry(QtCore.QRect(260, 10, 141, 61))
        self.DrawBackButon.setObjectName(_fromUtf8("DrawBackButon"))
        self.label_2 = QtGui.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(30, 30, 241, 21))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.label_3 = QtGui.QLabel(Form)
        self.label_3.setGeometry(QtCore.QRect(30, 90, 241, 16))
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.syringe_spinBox = QtGui.QSpinBox(Form)
        self.syringe_spinBox.setGeometry(QtCore.QRect(430, 80, 131, 31))

self.syringe_spinBox.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.
AlignVCenter)
        self.syringe_spinBox.setMinimum(1)
        self.syringe_spinBox.setMaximum(4)
        self.syringe_spinBox.setProperty("value", 4)
        self.syringe_spinBox.setObjectName(_fromUtf8("syringe_spinBox"))
        self.label_4 = QtGui.QLabel(Form)
```

```
        self.label_4.setGeometry(QtCore.QRect(30, 130, 241, 16))
        self.label_4.setObjectName(_fromUtf8("label_4"))
        self.RunButton = QtGui.QPushButton(Form)
        self.RunButton.setGeometry(QtCore.QRect(20, 230, 261, 61))
        self.RunButton.setObjectName(_fromUtf8("RunButton"))
        self.StopButton = QtGui.QPushButton(Form)
        self.StopButton.setGeometry(QtCore.QRect(300, 230, 261, 61))
        self.StopButton.setObjectName(_fromUtf8("StopButton"))
        self.ExitButton = QtGui.QPushButton(Form)
        self.ExitButton.setGeometry(QtCore.QRect(20, 310, 541, 41))
        self.ExitButton.setObjectName(_fromUtf8("ExitButton"))
        self.line = QtGui.QFrame(Form)
        self.line.setGeometry(QtCore.QRect(30, 290, 521, 16))
        self.line.setFrameShape(QtGui.QFrame.HLine)
        self.line.setFrameShadow(QtGui.QFrame.Sunken)
        self.line.setObjectName(_fromUtf8("line"))
        self.doubleSpinBox = QtGui.QDoubleSpinBox(Form)
        self.doubleSpinBox.setGeometry(QtCore.QRect(430, 120, 131, 31))

self.doubleSpinBox.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.Al
ignVCenter)

self.doubleSpinBox.setCorrectionMode(QtGui.QAbstractSpinBox.CorrectToNearestValue)
        self.doubleSpinBox.setMinimum(00.01)
        self.doubleSpinBox.setMaximum(20000.0)
        self.doubleSpinBox.setSingleStep(0.01)
        self.doubleSpinBox.setObjectName(_fromUtf8("doubleSpinBox"))
        self.Fill_radioButton = QtGui.QRadioButton(Form)
        self.Fill_radioButton.setGeometry(QtCore.QRect(260, 180, 121, 20))
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.Fill_radioButton.setFont(font)
        self.Fill_radioButton.setObjectName(_fromUtf8("Fill_radioButton"))
        self.Fill_radioButton.setChecked(True)
        self.Empty_radioButton = QtGui.QRadioButton(Form)
        self.Empty_radioButton.setGeometry(QtCore.QRect(430, 180, 121, 21))
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.Empty_radioButton.setFont(font)
        self.Empty_radioButton.setObjectName(_fromUtf8("Empty_radioButton"))
        self.label_5 = QtGui.QLabel(Form)
        self.label_5.setGeometry(QtCore.QRect(30, 180, 241, 21))
        self.label_5.setObjectName(_fromUtf8("label_5"))

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        Form.setWindowTitle(_translate("Form", "Syringe Pump Controls", None))
        self.DepressButton.setText(_translate("Form", "Emptied", None))
        self.DrawBackButon.setText(_translate("Form", "Filled", None))
        self.label_2.setText(_translate("Form", "Plunger position presets:", None))
        self.label_3.setText(_translate("Form", "Number of syringes:", None))
```

```
        self.label_4.setText(_translate("Form", "Desired flow rate per syringe (mL/s):",
None))
        self.RunButton.setText(_translate("Form", "Run", None))
        self.StopButton.setText(_translate("Form", "Stop", None))
        self.ExitButton.setText(_translate("Form", "Exit", None))
        self.Fill_radioButton.setText(_translate("Form", "Fill", None))
        self.Empty_radioButton.setText(_translate("Form", "Empty", None))
        self.label_5.setText(_translate("Form", "Direction to plunge:", None))


if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    Form = QtGui.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

APPENDIX H: DIGITAL CONTROLS CODE – PROGRAM

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  6 15:45:48 2017

@author: erinpeterson
"""





"""
import modules
"""

import sys
import RPi.GPIO as GPIO
import time
from PyQt4.QtGui import QDialog, QApplication
from PyQt4 import QtGui, QtCore
from GUI_Draft06 import Ui_Form
#import numpy as np

"""
import I/O pin masks
"""
global flow, syringes, direction, clock, directionPin, enable, limitA, limitB

#all pin numbers based on setmode = BCM (not board)

clock = 18 #step signal to driver - "clock+"
directionPin = 23 #direction signal to driver - "direction+"
enable = 24 #enable signal for driver - "on/off+"
limitA = 17 #limit A GPIO hookup
limitB = 27 #limit B GPIO hookup

"""
Set up general purpose input/output
"""

GPIO.setmode(GPIO.BCM) #configure pin layout as BCM not board (personal preference)
GPIO.setwarnings(False) #ignore warnings and proceed with the program

GPIO.setup(clock, GPIO.OUT)
GPIO.output(clock, GPIO.LOW) #GPIO.LOW sets the present output to 0V
                            #will later need to initialize this pin as PWM

GPIO.setup(directionPin, GPIO.OUT)
GPIO.output(directionPin, GPIO.LOW) #GPIO.LOW sets the present output to 0V

GPIO.setup(enable, GPIO.OUT)
GPIO.output(enable, GPIO.LOW) #GPIO.LOW sets the present output to 0V
                                #the motor is now enabled (on)

GPIO.setup(limitA, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #GPIO.PUD_DOWN pulls the voltage
down if no signal
```

```
GPIO.setup(limitB, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #GPIO.PUD_DOWN pulls the voltage
down if no signal
                                                         #GPIO.PUD_DOWN is a redundancy on
the pull-down resistors in the circuit

"""
Call up GUI for inputs
"""

class main_window(QDialog):

    #initialize the variables for the GUI
    global flow, syringes, direction, clock, directionPin, enable, limitA, limitB, freq,
pwm

    #set BCM pin numbers within class
    clock = 18 #step signal to driver - "clock+"
    directionPin = 23 #direction signal to driver - "direction+"
    enable = 24 #enable signal for driver - "on/off+"
    limitA = 17
    limitB = 27

    #initialize variables based on GUI default design settings
    flow = 0.01 #initial flow from GUI design (max is 200 for now)
    syringes = 4 #initial number of syringes from GUI design
    freq = flow #initialize a relationship between frequency and flow
    direction = GPIO.HIGH #set default direction to match default checked radio button
    pwm = GPIO.PWM(clock, freq) #initialize the pwm variable for reference


    def __init__(self):
        super(main_window,self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)
        self.assign_widgets()
        self.show()

    def assign_widgets(self):
        self.ui.ExitButton.clicked.connect(self.close_application)
        self.ui.RunButton.clicked.connect(self.run)
        self.ui.StopButton.clicked.connect(self.stop)
        self.ui.DrawBackButon.clicked.connect(self.fill)
        self.ui.DepressButton.clicked.connect(self.empty)
        self.ui.doubleSpinBox.valueChanged.connect(self.flowset)
        self.ui.syringe_spinBox.valueChanged.connect(self.syringe_num)
        self.ui.Fill_radioButton.clicked.connect(self.CCW)
        self.ui.Empty_radioButton.clicked.connect(self.CW)

    def close_application(self):
        GPIO.cleanup()
        sys.exit()

    def flowset(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB,
freq, pwm
        flow = self.ui.doubleSpinBox.value()
        #***need to define relationship between output flow and frequency setting***
```

```
        print(flow)

    def syringe_num(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB, 
freq, pwm
        #***can later calibrate this based off of flow rates from NEMA 34***
        syringes = self.ui.syringe_spinBox.value()
        print(syringes)

    def CW(self): #CW rotation corresponds to draw-back plunger motion
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB, 
freq, pwm
        direction = GPIO.HIGH
        print("CW")

    def CCW(self): #CCW rotation corresponds to forward plunger motion
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB, 
freq, pwm
        direction = GPIO.LOW
        print("CCW")

    def run(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB, 
freq, pwm
        print("run")

        #disable all "Run"-function buttons
        self.ui.RunButton.setEnabled(False)
        self.ui.DrawBackButon.setEnabled(False)
        self.ui.DepressButton.setEnabled(False)

        if (GPIO.input(limitA) == GPIO.LOW and direction == GPIO.HIGH):
            #if not already fully drawn back, and intended motion is to draw back (CW)
            GPIO.output(enable, GPIO.LOW) #enable the motor
            GPIO.output(directionPin, direction) #set the direction
            freq = flow #set the frequency as it corresponds to the flow given
            dutycycle = 50 #leave duty cycle at 50%
            pwm = GPIO.PWM(clock, freq) #redefine pwm output with set freq
            pwm.start(dutycycle) #start the pwm output
            print(freq)

        elif (GPIO.input(limitB) == GPIO.LOW and direction == GPIO.LOW):
            #if not already fully depressed, and intended motion is to plunge forward 
(CCW)
            GPIO.output(enable, GPIO.LOW) #enable the motor
            GPIO.output(directionPin, direction) #set the direction
            freq = flow #set the frequency as it corresponds to the flow given
            dutycycle = 50 #leave duty cycle at 50%
            pwm = GPIO.PWM(clock, freq) #redefine pwm output with set freq
            pwm.start(dutycycle) #start the pwm output
            print(freq)

        else:
            print("Error: Improper input configuration.")
            self.stop

        #check to see if limit switches have been tripped
```

```
##        GPIO.add_event_detect(limitA, GPIO.RISING, callback = self.limA, bouncetime =
300)
##        GPIO.add_event_detect(limitB, GPIO.RISING, callback = self.limB, bouncetime =
300)

    def limA(self, event, names=None):
        print("Limit A tripped.")
        self.stop()

    def limB(self, event, names=None):
        print("Limit B tripped.")
        self.stop()

    def stop(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB,
freq, pwm
        print("stop")

        #reactivate all "Run"-function buttons
        self.ui.RunButton.setEnabled(True)
        self.ui.DrawBackButon.setEnabled(True)
        self.ui.DepressButton.setEnabled(True)

        pwm.stop()

    def fill(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB,
freq, pwm
        print("Fill")

        #disable other "Run"-function buttons
        self.ui.RunButton.setEnabled(False)
        self.ui.DrawBackButon.setEnabled(False)
        self.ui.DepressButton.setEnabled(False)

        if (GPIO.input(limitA) == GPIO.LOW): #if not already fully drawn back
            direction = GPIO.HIGH #set rotation to CW to draw back plungers
            GPIO.output(enable, GPIO.LOW) #enable the motor
            freq = flow
            dutycycle = 50 #leave duty cycle at 50%
            pwm = GPIO.PWM(clock, freq)
            pwm.start(dutycycle)

        else:
            print("Error: Limit A already tripped.")
            self.stop

        #check to see if limit switches have been tripped
        GPIO.add_event_detect(limitA, GPIO.RISING, callback = self.limA, bouncetime =
300)
        GPIO.add_event_detect(limitB, GPIO.RISING, callback = self.limB, bouncetime =
300)

    def empty(self):
        global flow, syringes, direction, clock, directionPin, enable, limitA, limitB,
freq, pwm
        print("Empty")
```

```
        #disable other "Run"-function buttons
        self.ui.RunButton.setEnabled(False)
        self.ui.DrawBackButon.setEnabled(False)
        self.ui.DepressButton.setEnabled(False)

        if (GPIO.input(limitB) == GPIO.LOW): #check if already in fully depressed
position
            direction = GPIO.LOW #set direction to CCW to drive plungers forward further
            GPIO.output(enable, GPIO.LOW) #enable the motor
            dutycycle = 50 #leave duty cycle at 50%
            pwm = GPIO.PWM(clock, 20) #push plunger forward at constant 20 Hz
            pwm.start(dutycycle)

        else:
            print("Error: Limit B already tripped.")
            self.stop

        #check to see if limit switches have been tripped
        GPIO.add_event_detect(limitA, GPIO.RISING, callback = self.limA, bouncetime =
300)
        GPIO.add_event_detect(limitB, GPIO.RISING, callback = self.limB, bouncetime =
300)


if __name__ == "__main__":
    app = QApplication.instance()
    if not app:
        app = QApplication(sys.argv)
    app.aboutToQuit.connect(app.deleteLater)
    main_win = main_window()
    app.exec_()
```