SENIOR DESIGN PROJECT FINAL REPORT: ADAPTING A PARKER PROPORTIONAL VALVE FOR USE WITH A FESTO HYDRAULIC TRAINER

SPONSORED BY:

EDISON STATE COMMUNITY COLLEGE

TEAM MEMBERS:
JAMES CLEMENTS
AUDREY FIELDS
LUKE HOUSE
DAVID POEPPELMAN

ENT 497/498
SENIOR DESIGN
PROFESSOR ROBERT SPECKERT
May 7, 2020

STATEMENT OF PURPOSE (EXECUTIVE SUMMARY)

The intent of this project is to create a lab for demonstrational purposes in Edison's Hydraulics & Pneumatics course. The lab will clearly demonstrate to students the function of a proportional valve, and how a proportional valve can be used in conjunction with a PID controller to maintain an RPM for a hydraulic motor under varying loads.

Goals:

- Control a proportional directional control valve, using a PID controller, to adjust the valve's position, resulting in the change of the hydraulic fluid flow rate to the motor.
  - The tachometer system will measure the RPM of the hydraulic motor and feed this information back to the PID controller.
- Demonstrate compensation of the hydraulic system when a load is applied to the hydraulic motor.
  - A physical braking system will be implemented to apply a load to a disk attached to the motor's shaft.
- Record the data of system input and output responses to provide a method for the class to analyze lab results.
- Provide diagram schematics for future users to easily be able to set up, disassemble, or add to the system in the future.
- Produce a system that is safe to use with no exposed moving parts or 'pinch points'. The wiring will be properly insulated and directed, as to not interfere with the function of the mechanical components.
- Build a project with the lowest cost possible.

In pursuit of this objective, we will also attempt to create a troubleshooting manual and lesson plan centered around our system. The troubleshooting manual will allow our system to be easily fixed, if necessary. The lesson plan will help with the integration of our system into the classroom environment; and it will assist the Hydraulics & Pneumatics professor, Dave Barth, in teaching with our system. This addition will also allow students participating in the lab to broaden their knowledge and understanding of hydraulic components and controls used in industry.

CONTENTS

TABLE OF FIGURES

SCOPE AND METHODOLOGY

The purpose of this project is to adapt the Parker D1FPE50BA9NB00 Proportional Valve (Appendix B) for use with the Festo Hydraulic Trainers at Edison State Community Colleges for future use of students taking the Hydraulics & Pneumatics course (1). This project will consist of a proportional valve controlled by a variable voltage source made from a constant 24V power supply. The valve will control a hydraulic motor, which will use an encoder to monitor the motor RPM. Our group will set up data acquisition to display a command signal and a system response using an Arduino paired with a connected PC. The project will use a braking system to add load to the motor to induce drag, which will be displayed on a screen connected to the sensors for the project. We will add a closed loop control into the system by allowing the user to set PID parameters. This will allow us, for example, to maintain RPM of the motor under load by calculating and raising the voltage delivered to the valve to an appropriate amount.

Goals:

- The proportional valve will be mounted to the trainer via the slotted mounting board.
- The proportional valve will utilize closed loop feedback.
- The valve will drive a hydraulic motor.
- The hydraulic motor will have friction-based caliper-style brakes for creating drag. The brakes will be used on the flywheel, which will be attached to the end of the motor shaft.
- A sensor will be used to determine the RPMs of the motor and will be utilized to provide feedback. This will allow the voltage supplied to the valve to be shifted in order to maintain RPMs under variable motor loads and vice-versa. The RPM sensor will be Arduino-based and will be built and tested by the group.
- A custom housing for the RPM sensor will be designed and made by the group members.
- An Arduino will be used to analyze sensor output and communicate with the GUI program, located on a separate PC.
- A program will be created to display voltages applied to the valve, RPM of the motor, the load applied to the motor, etc. This GUI program will be modeled after the ServoStar program, as requested by Dave Barth.
- The program will use a PID controller to create a closed feedback loop. (Arduino, Raspberry PI, etc.).
- The program will display gathered/calculated data in graph form within the GUI program.

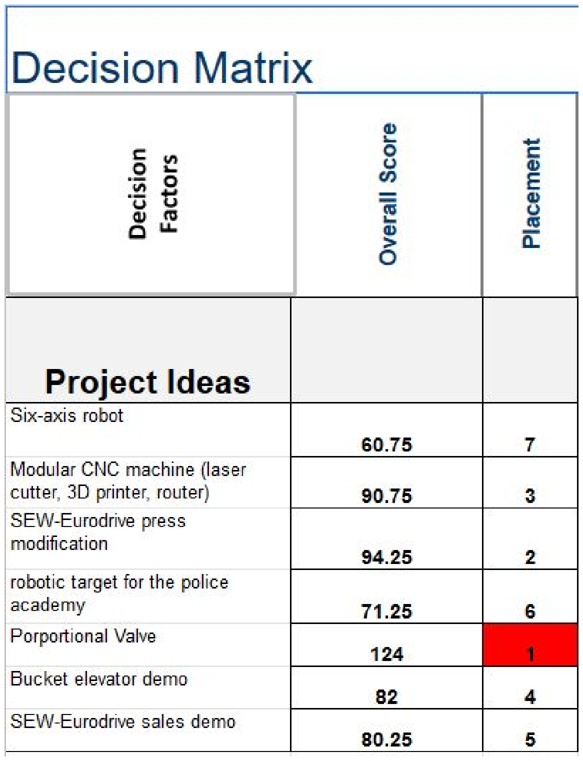
- A Diagram/Schematic/Troubleshooting Manual will be created for future users to easily be able to set up and tear down the system.
- Safety will be paramount - no exposed moving parts or ‘pinch points’. Wiring will be properly insulated and directed as to not interfere with the function of the mechanical components.

To accomplish the goals listed above, a number of factors had to be considered to deliver a quality result, while keeping costs at a minimum. In order to come up with a more accurate project timeline, a variety of research had to be conducted.

<u>RESEARCH</u>

DECISION MATRICES

See "Appendix E" for individual group member Decision Matrices.



Decision Matrix

Decision Factors / Project Ideas	Overall Score	Placement
Six-axis robot	60.75	7
Modular CNC machine (laser cutter, 3D printer, router)	90.75	3
SEW-Eurodrive press modification	94.25	2
robotic target for the police academy	71.25	6
Porportional Valve	124	1
Bucket elevator demo	82	4
SEW-Eurodrive sales demo	80.25	5

Figure 1: Averaged Decision Matrix

Our group began the semester unsure of which project to choose. We initially came up with 7 project ideas, but couldn't come to a conclusive answer about which project to choose. As a group, we decided to complete individual decision matrices and met the following week with our decision matrices completed (see Appendix E for individual Decision Matrices). We then averaged our matrices together and concluded that the highest-scoring project would be the one we chose, as shown in Figure 1. The Edison Proportional Valve Project came out on top with 124 points. We can attribute this to the overall low cost of the project, the complexity and wide range of skills required to complete the project, and the scope that would leave all four of us with plenty to work on during the entire course. We communicated our decision to Dave Barth at Edison, our customer representing Edison State Community College for this project, and expressed to him that we would like to take on his project for ENT497 and ENT498.

MICROPROCESSOR

Research began on the microprocessor for our project and was largely conducted using sources from the internet, as well as several labs from ongoing alternate courses. Research initially began based on the control system or controller that the group had designed to use for this project. There were two target subjects: an Arduino Uno and a Raspberry Pi.

Figure 2: Raspberry Pi

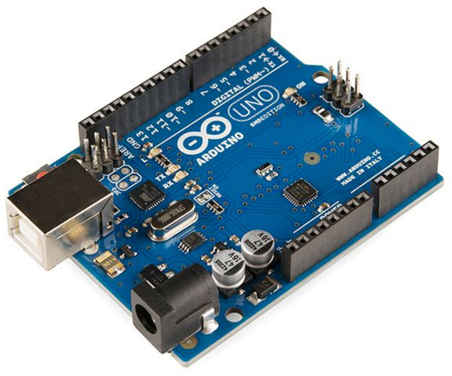


Figure 3: Arduino UNO

The team then spoke to the client to see if either of these devices were readily available. The resulting information was that the client had one Arduino Uno and an older Raspberry PI of an unspecified model. The Arduino Uno is a microcontroller board design, based on an ATmega328 that has 14 digital input and output pins, with 6 pins being Pulse Width Modulation (PWM) enabled. The Uno also contains 6 analog input pins and a USB B connection port for both programming and power connection (9). There was a lack of information about the specific model of Raspberry PI the client possessed; thus, the group was not confident that the necessary functionalities would be available. Upon researching the Arduino Uno and considering the familiarity of all members of the group with said microcontroller, we decided in favor of the Arduino. Since the microcontroller should be protected against the oil present in and on most of the Festo Didactic Hydraulic Trainers, a housing proved necessary for the safety of the system. Fortunately, the client, Dave Barth, possessed a blank housing normally used for electronics (see Figure 4). The box we received could be mounted in the same way as the boxes shown, keeping the Arduino and any other necessary electronics well out of the danger zone.

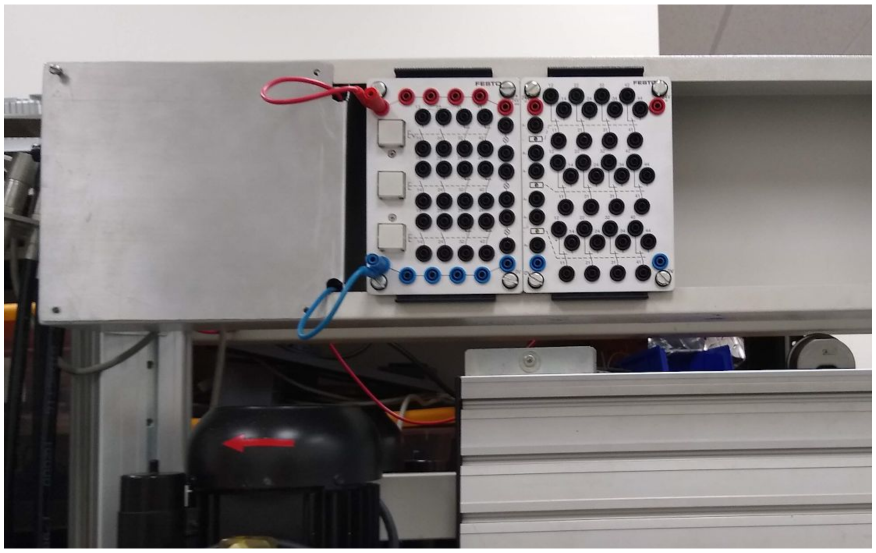


Figure 4: Control Boxes for Festo Hydraulic Trainer

LED AND PHOTODIODE RESEARCH

One of the issues that cropped up during the tachometer research and design was the lack of datasheets and dimensions provided with the components. Despite repeated attempts to communicate with the retailer, no response was ever received. In order to prototype the tachometer housing, dimensions and possible tolerances were needed. A quick search on Google brought up hundreds of images of 5mm LEDs and their associated dimensions. Figure 5 below contained the largest tolerances. Picking the design with the largest tolerances was deemed to be the best option because the parts were low quality. The wavelength of light chosen was 940 nm (in part because of the abundance and low cost of 940 nm wavelength parts). Human eyes alone are not able to perceive whether the infrared LED is working. In order to test the operation of the LED, a photoresistor that had the capability to react to infrared light was used in conjunction with an Arduino Uno, confirming that the LEDs were working. For additional confirmation, a smartphone front-facing camera was used that did not utilize an infrared filter. With confirmation that the components themselves would work, the dimensions given were used to make the prototype of a sensor housing shown in Figure 6. An updated version of the housing in Figure 7 was later modeled with ribs for strength, along with the sensor itself.

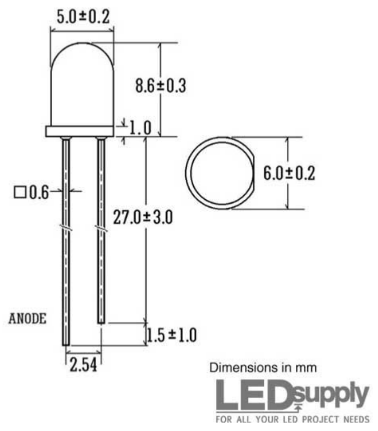


Figure 5: LED Dimensions

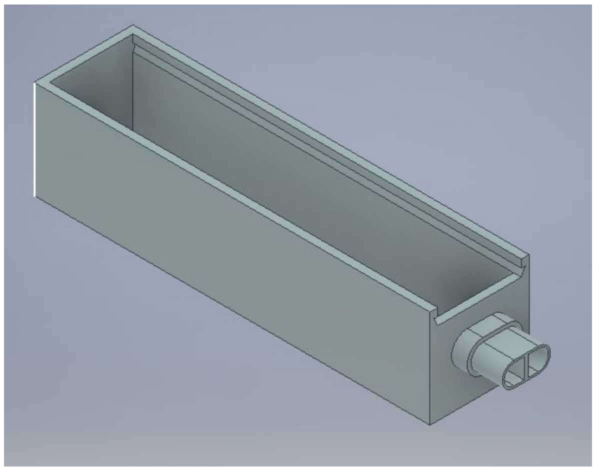


Figure 6: 3D Printed Housings for Tachometer Components

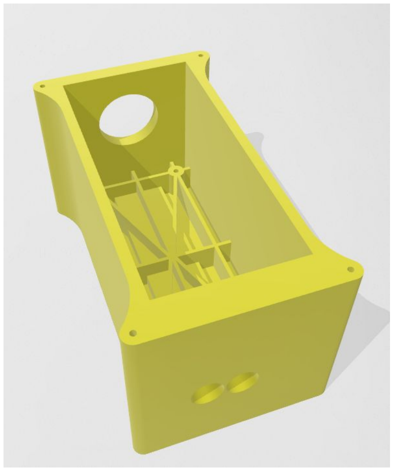


Figure 7: 3D Printed Housings for Tachometer Components with Support Ribs

TACHOMETER

The initial research for the tachometer included two types: contact tachometers and optical tachometers. A third type of tachometer using proximity sensors was also available, but was immediately decided against, due to concerns about collisions, especially in the context that students without knowledge of proximity sensor function would be in contact with the system. Physical tachometers are mechanically connected to a shaft or rotating object, while optical tachometers use light in conjunction with various algorithms and calculations to measure revolutions. The immediate research involving contact and optical tachometers was a simple search on Amazon.com. After deciding that both options could be affordable as handheld devices, research on the integration of handheld tachometers and an Arduino Uno was started. No adequate results were found for handheld tachometer integration. A search on both Amazon and McMaster Carr showed that standalone integratable tachometers are very expensive; however, multiple tutorials and videos on how to create a tachometer using infrared LEDs and phototransistors were available.

With a lack of viable methods to integrate handheld tachometers into the Arduino and the unacceptable expense of standalone tachometers, the only options available were using an encoder or an optical sensor to create a tachometer. It was decided that using an optical sensor to detect time between pulses and calculate the number of revolutions was the best choice. Since this sensor was to be used in an educational capacity and exposed to students, the sensor itself is at high risk of breaking. With this in mind, the decision was made to create a sensor from scratch using 5 millimeter infrared LEDs and phototransistors/photodiodes. Several of the team members had a class that had used a phototransistor and an infrared LED to make an optical tachometer, which made the decision lean toward using phototransistors. However, due to a mislabeling issue on Amazon.com for the parts ordered, photodiodes were received and subsequently chosen for the optical tachometer. A tutorial on how to create an optical tachometer using photodiodes could be found on Instructables.com. While there were several ways to create the optical tachometer with a photodiode, the circuit shown below allows the potentiometer to be adjusted to account for lighting differences in the environment (11).

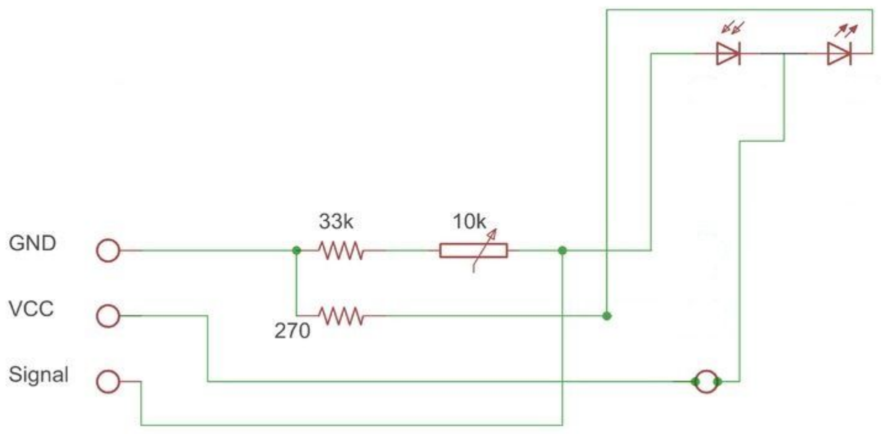


Figure 8: Tachometer Wiring Diagram

However, after verification and completion of a prototype using the circuit in Figure 8, it was discovered that none of the eight remaining photodiodes were viable. With this in mind, the initial reaction was to use an optical sensor that could be hooked directly to the Arduino to avoid further unnecessary costs. The sensor chosen was the Taiss E3F-DS30C4 diffuse photoelectric optical sensor for 6-36 Volts DC, which also had the ability to run at the Arduino's 5V capacity. There were concerns about the sensitivity of the sensor, however, because during testing, the sensor had a difficult time distinguishing reflective and non-reflective surfaces. After further discussion, it was decided that an encoder was the best choice. This would eliminate the uncertainty around lighting conditions and mounting. The wide availability of incremental rotary encoders at varying pricing, made incremental rotary encoders the obvious choice. Due to the cheap component failure while creating the previous iteration of sensor, using a very cheap encoder did not seem like a good idea. With the client running engineering programs at Edison State Community College and receiving donations of all sorts of salvaged electronic components, we approached him about whether he had fitting components. Luckily, there was an adequate encoder available from the selection presented. The encoder chosen was an RSE Optical Incremental Shaft Position Encoder that had an output of 60 pulses per revolution (PPR). While the encoder required more work with regards to mounting and implementation to sense revolutions per minute, the lack of cost and ability to output signals 60 times per revolution with minimal directional change issues was preferred.

PID CONTROLS

One of the specifications from the client was that the proportional valve should have PID controls to compensate for when a load is applied in order to maintain RPMs of the flywheel. PID controls work using loop feedback to calculate the difference between a measured value and target value, then adjust the system to reduce the difference. There are three types of control that can be used depending on the control response requirements, such as maximum allowable overshoot, how quickly the system responds, stability, etc. The three types of control are proportional, integral, and derivative (P,I, and D), which all have their own gain constants within the system. Each type of control responds to a different type of error. Proportional control responds to the amount of error at the current time. Integral control responds to the history of error for error elimination, while derivative control responds to the rate of change of error for the purpose of future error elimination. The general equation for PID control is shown at the top of Figure 9. In this equation P(t) is the output of the controller at a certain time for some error condition. Kp, Ki, and Kd are respectively proportional, integral, and derivative gain constants, which are typically adjusted for system stability and optimal control characteristics. Ep is the change in error that comes from the current system state and the desired system state. P(0) or KpEo is the steady state output of the controller before a change. The PID control block diagram shown in the bottom of Figure 9 demonstrates how each separate control type is summed to equal the adjusted signal to send to the system (12).

The general equation for PID controller operation is:

$$P(t) = KpEp + Kp{*}Ki{*} \int Ep\, dt + Kp{*}Kd \frac{dEp}{dt} + KpEo$$

KpEo is also called P(0).

Also given as:

$$P(t) = Kp(Ep + \frac{1}{Ti}\int Ep\, dt + Td \frac{dEp}{dt}) + P(0)$$

Ti = Integral time
Td = derivative time

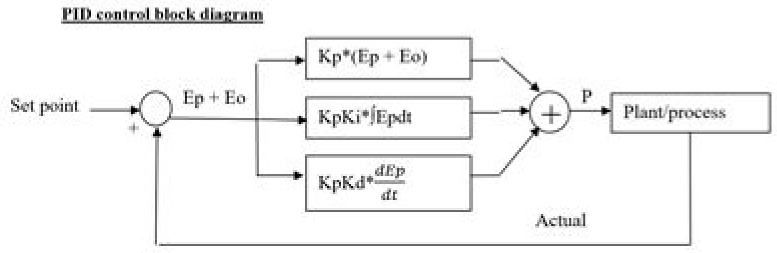


Figure 9: PID Controls

In this particular case, some amalgamation of proportional, integral, and derivative controls will be used to adjust the amount of voltage to a proportional valve based on a RPM measured from a tachometer. This voltage change will cause the valve to open or close, increasing or decreasing the speed of the motor being observed by the tachometer. With the controller in this situation being an Arduino Uno, a library for PID controls was found in the Arduino Playground community that we thought would adequately serve for our purposes, after making a few adjustments. As the Arduino IDE has been updated several times since the library was created in 2017, the library was deemed non-viable. Security was also a concern when deciding whether to use this library, both for the client and the team members, as the original source is unverified (13). The unverified source may cause some issues with getting the library loaded to a computer at Edison State, and the likelihood of a system administrator approving an unverified library is near zero.

BRAKE SYSTEM

When determining how to apply load to the hydraulic motor, a variety of methods were considered. One of the methods considered was implementing a regenerative braking system to apply a load to the hydraulic motor shaft and convert mechanical energy into electrical energy. This was also considered to be an energy efficient way to apply a variable load electronically and recycle some of the energy back into the system. Before finding the cheapest price to implement a regenerative brake, research was conducted to better understand how they work and what components would be needed. This also helped determine whether it would be cost beneficial to design and build a regenerative brake system or purchase a regenerative brake system. After comparing costs online from regenerative braking suppliers, it was determined that building a system from individually purchased components would be cheaper. A list of components were acquired for a cost estimate, and presented to Edison for approval. Through communication with the customer, it was further understood that the cheapest way to apply load to the motor was most desired. Regenerative braking was not the cheapest method we could implement, so more research was conducted on alternative breaking methods to reduce costs. After researching prices for brake systems, a friction-based caliper-style V-brake was found to be the most cost effective method while still being safe to use during labs (4).

Upon further research, we concluded that V-style bicycle brakes would prove an effective means of applying load to our system. Bike brakes were preferred as they came built with the torque load specifications we would need to create significant braking power on the motor, while providing a pre-assembled complete package to easily and securely mount in place to the hydraulic trainer slotted board (4).

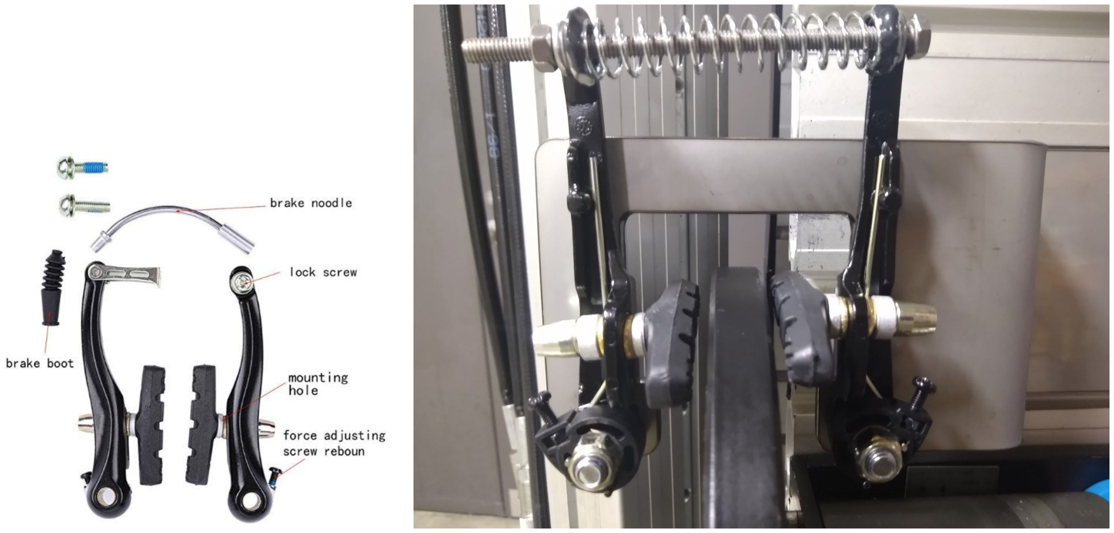


Figure 10: Caliper-Style Bicycle Brakes

The brakes are mounted with screws and a locknut via the bottom hole in each brake. The top brake noodle retainer was removed, and, in its place, two eye bolts were threaded through the lock screw holes to serve as 90° adapters to each V-brake arm. A longer 110mm bolt was threaded through the loop of each eye bolt and is used to tension the V-brakes and apply load by tightening a nut at the end of the bolt.

The V-brakes are mounted to the Festo Trainer board by using a cut stainless steel plate. The plate, donated by Sidney Manufacturing Company, was made using 7ga. stainless steel cut with a plasma machine (See Figure 10). The mounting bracket uses a hole to mount each of the bike brakes with the aforementioned lock nut and bolt, and slots are used to secure the plate to the trainer board with t-screws in the correct position. This design allows for remarkable adjustability and accurate alignment of the brakes with the flywheel.

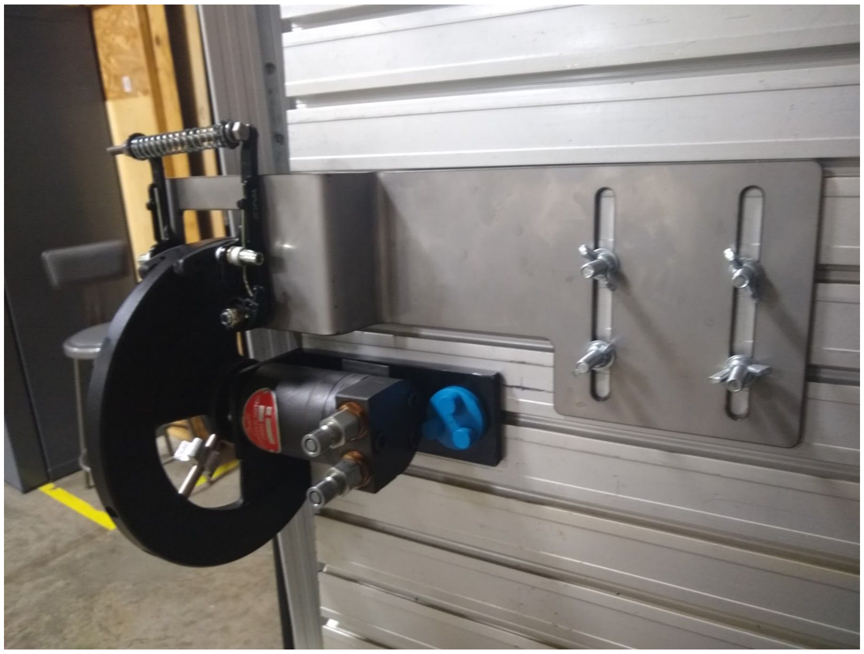


Figure 11: Brake Mounting Plate

Upon testing our braking assembly we were able to apply appropriate load to the flywheel by tensioning the 110mm bolt's nut, and were able to completely stop the flywheel with the braking force from the V-brakes.

PROPORTIONAL VALVE CONTROL

Upon researching several methods of controlling the proportional valve, we decided to utilize a PWM module. This module is optimal because it can be easily controlled through an input voltage sourced from the Arduino, and can direct larger voltages and currents from our main 24VDC power to control the valve's spool position. The command signal will be sent to pins E and D on the proportional valve's Amphenol plug (See Figure 12) (2)(7)(8).

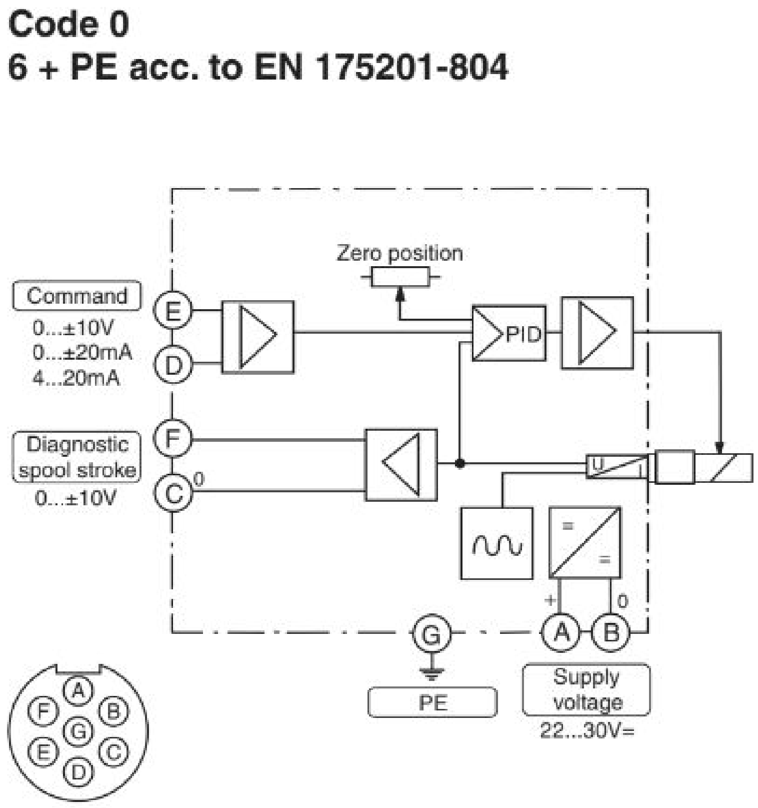


Figure 12: Valve Pin Layout

We first considered using a logic-level MOSFET on recommendation from various Arduino forum users looking to control their own proportional valves. We liked this option, but decided to put in more research in order to open our options and find the most effective solution for the cost.

We then considered using a buck converter to step down our voltage from 24VDC to variable 0-10VDC for control. This was another appealing option, but we had difficulty finding a suitable buck converter that had all the features we needed (both manual and electronic voltage control). We wanted to avoid wasting money on unneeded features, such as an LCD screen to display voltage values, as we would be displaying voltage in our PC program regardless. Available buck converters were also significantly higher priced than comparable MOSFETS that performed the same job. In addition, buck converters would have performed slower in a system where time was key, so we decided to use a MOSFET module in our project (5)(6).

We eventually settled on using a IRF520 MOSFET driver module. We liked the utility and ease of use of a MOSFET for PWM control, and the cost was significantly lower for 6 modules than for the cost of one buck converter module. Using PWM control is also one of the quickest ways to regulate larger voltages by use of an Arduino, so it seemed most fitting for our project as a whole (5)(6).

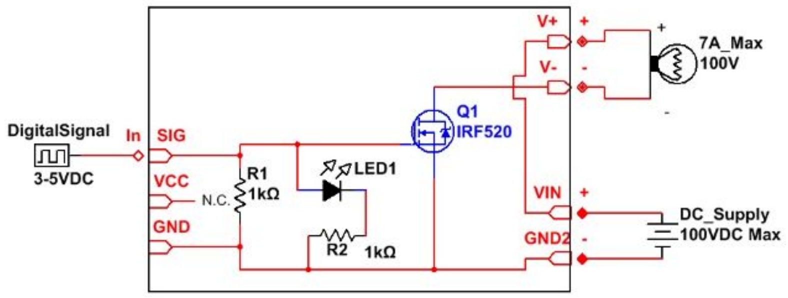


Figure 13: MOSFET Module Schematic

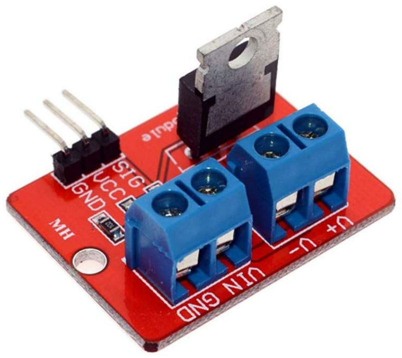


Figure 14: MOSFET Module

Before testing began, we found that the original valve socket plug was proprietary, and the mating connector needed was found to be too expensive. To save costs on an electrical connector, the valve plug was disassembled from the valve and a new plug was soldered to replace the original plug. After the soldering and wiring of the new plug was installed, testing began.

After testing the valve with the IRF520 MOSFET driver module, it was found that the control circuit for the valve needed a 0V to +10V signal and a 0 to -10V signal. With the current IRF520 MOSFET driver module only 0V to +10V could be achieved so other control circuits had to be considered.

The first control circuit that was tested and evaluated integrated a LM741 operational amplifier. This operational amplifier was chosen because it was readily available and would not cost anything.

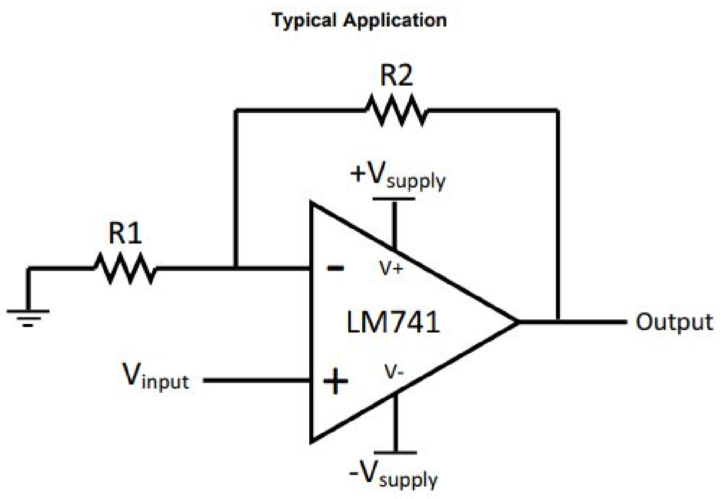
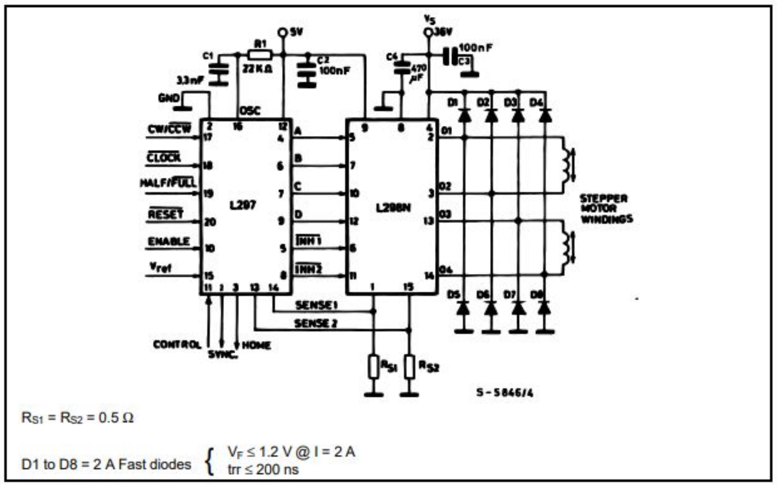


Figure 15: LM741 Electrical Schematic

After measuring the output of this chip, it was found that a -10V to +10V could not be achieved because the buck converter being used did not supply a negative voltage for the V- input. To reduce the complexity of the control circuit and to save time testing, soldering, and wiring the chip, a new electronic module was purchased, which was an H-bridge module.



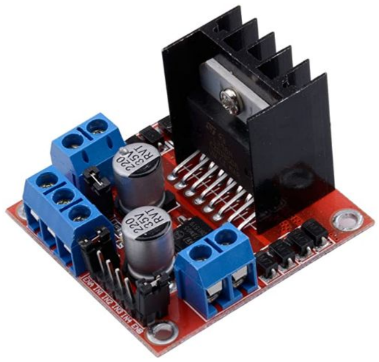


Figure 16: L298 electrical schematic and picture of L298 module board

The final module that was purchased was a module that had a L298 dual full-bridge driver integrated into a printed circuit board. Theoretically, this would have been able to supply both a 0V to +10V and a 0V to -10V signal to the control circuit of the valve. This module was never tested and confirmed to control the valve correctly due to the COVID-19 epidemic and its impact on our team's accessibility to the valve.

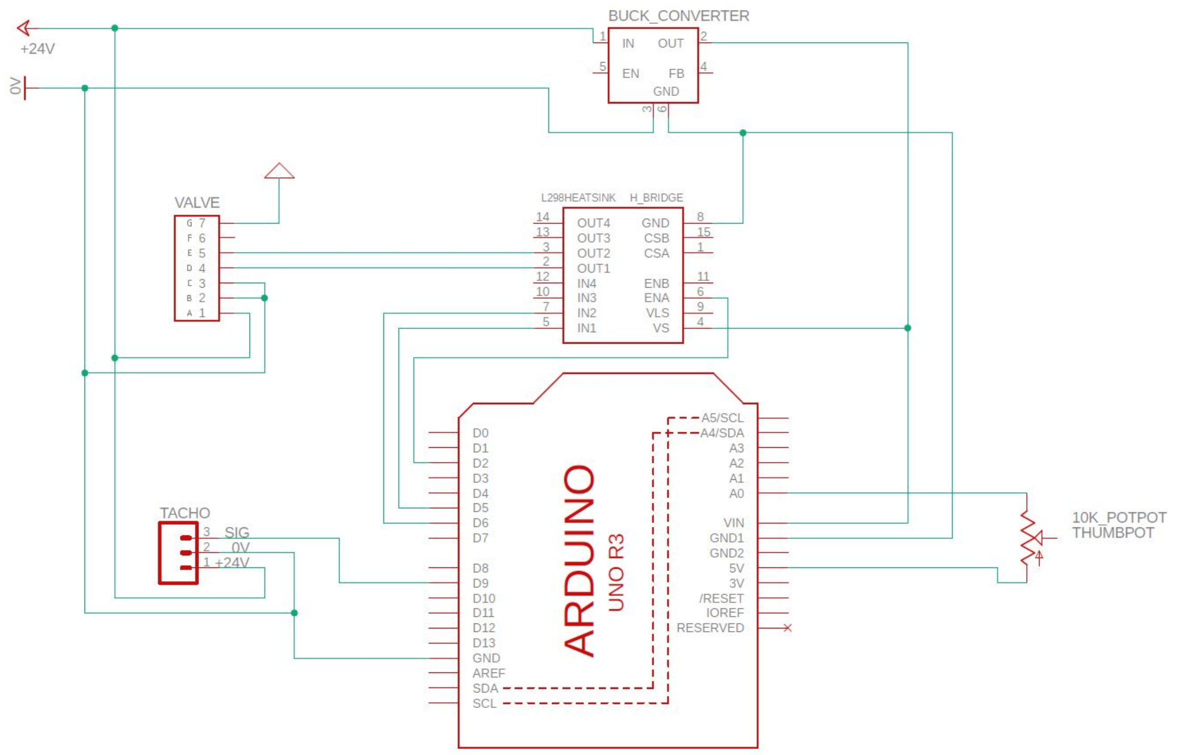


Figure 17: Electrical Schematic for the control box

VALVE BLOCK / MOUNTING

Another obstacle we faced as a group was how to adapt the proportional valve for use with our training unit. Upon investigation, we were relieved to find that the proportional valve would allow for more than enough fluid flow to properly power the hydraulic motor. We were also relieved to find that the proportional valve could be controlled with an adapted variable control voltage sampled from our main 24VDC bus line, and that it would be functional in our system with a bit of adaptive work. We then faced the problem of how to secure the proportional valve to the Hydraulic Trainer as it was running.

We knew from the start of the project that we would need to design a custom valve block to direct hydraulic fluid to hose nipples for use with the Quick-Connect hoses the hydraulic trainer uses. We considered purchasing a separate valve block from Festo that matches the ISO 4401 specifications needed for this valve, but upon inquiring about the price of this block, we realized that machining our own block with an Edison-provided piece of scrap aluminum would be vastly more time and cost effective (2). After several discussions about mounting the valve to the trainer board, we realized that we could use excess space on the valve block for mounting holes. We could drill holes through the block on either side of the valve to use with t-screws and wingnuts to secure the valve to the trainer board. This would save on manufacturing costs for making separate block and mounting systems, and would provide an ergonomic and intuitive mounting solution for the valve. As such the below valve block was designed.

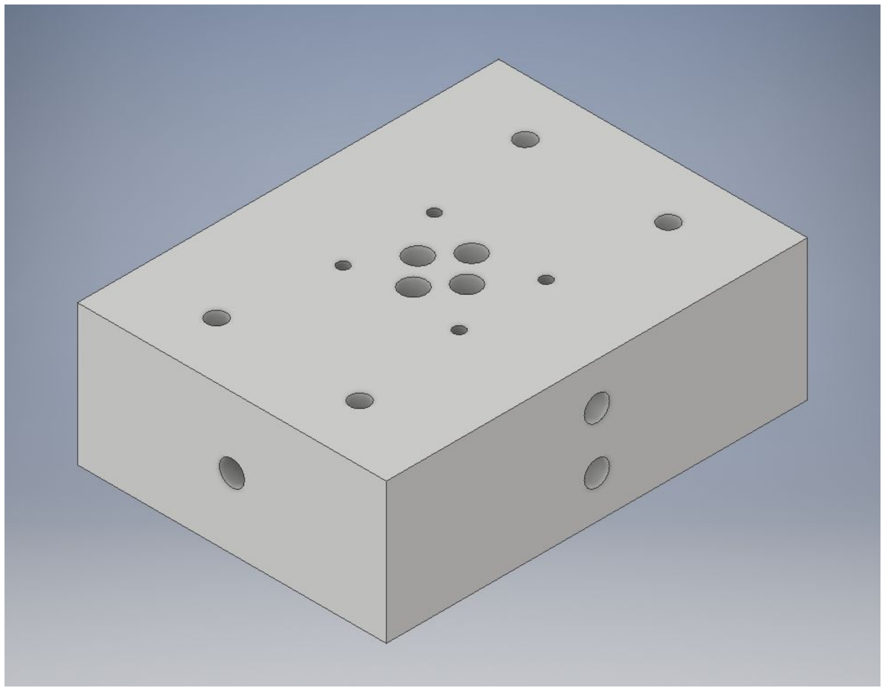


Figure 18: Finished Valve Mounting Block

ARDUINO PROGRAM

In order to measure RPMs and apply PID controls, the Arduino first needed to obtain the primary and secondary desired RPM values, the time to oscillate between the RPM values, proportional gain constant Kp, integral gain constant Ki, and the derivative gain constant Kd. This information is sent to the Arduino when the program on the computer is started. The Arduino program, however, must be uploaded and started to run the setup section of the programming. After the Arduino program is uploaded or started, the serial monitor is initialized using a baud rate of 500,000, which sets the maximum transferable bits per second to 500,000.

Next, the program waits on the required data to be sent from the computer application by utilizing the Serial.available() method inside an empty while loop. The result is the program stopping when no incoming data is detected in the serial monitor. Once data is detected, a string variable reads the data in the serial monitor until the newline character ‘\n’ is read. Since the string read from the computer application uses commas as delimiters to separate the data, the index of each comma is found. These indexes are used to separate the six required values mentioned above, and assign them to their respective variables, which marks the end of the setup portion of the Arduino program.

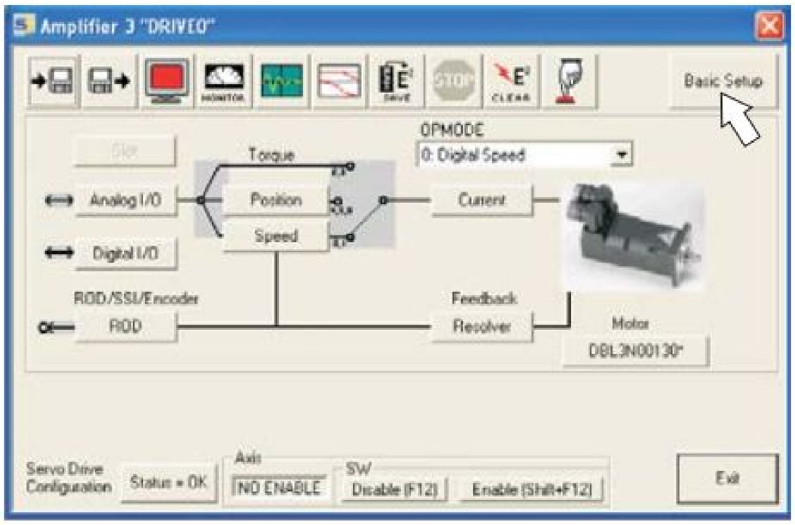
The main portion of the program, which loops continuously, measures RPMs, applies that RPM to a rolling average, then adjusts the signal sent to the proportional valve based on the rolling average result. The first step of the main portion is to assign the value returned by calling the pulsein() function to a variable. This returned value is the time in microseconds between pulses from the encoder. With this value being read only once per iteration of the main portion of the program, an array with an index rotating each iteration is assigned the RPM calculated using the formula $1/(\textit{pulsein}()\ \textit{returned value} / 1000000)$. This formula converts the microsecond return of the pulsein() function to seconds, and gets the number of revolutions per second. Then, the revolutions per second are multiplied by 60 seconds per minute, and divided by the 60 pulses per revolution the encoder transmits. As these two steps cancel each other out, they are not apparent in the formula. After adding the RPM value to the array for calculating the rolling average, the rolling average is calculated by summing all values in the array and dividing by the number of elements in the array. The arbitrary number of values chosen was 10, which significantly reduces the effect of false signals.

Once a rolling average is calculated, the proportional, integral, and derivative error are calculated. Proportional error is found by taking the setpoint of the system minus the rolling value, and dividing the result by the setpoint of the system. Integral error is found by adding the error multiplied by the time (in seconds) used to calculate the RPM value to all the previous values found the same way, representing the accumulation of error. Derivative error is calculated by subtracting the previous error from the current error, and dividing by the time (in seconds) used to calculate the RPM value. This shows the rate of change, or derivative, of the error. With each aspect of the PID control calculated (proportional, integral, and derivative), each of the errors are multiplied by their respective gain value and divided by 1000. 1000 was an arbitrary number chosen to keep the gain constants manageable and to allow for more leeway in tuning the system. Then, each of the errors with applied gain is added to the initial output of the controller, which is the setpoint divided by the maximum estimated RPMs of the motor. The maximum estimated RPMs were previously calculated using characteristics pulled from the datasheets of the pump and motor that are included with the hydraulic trainer. To find the maximum RPMs, the equation $N = (1000*Qt) / Vm$ was utilized, where N is the RPM value, Qt is the pump delivery rate, and Vm is motor displacement. Qt for the motor is 2.3 l/min, while the motor displacement is 8.3 cubic centimeters per revolution. This leads to the maximum RPM value being 2300 / 8.3, or just over 277 RPMs.

The result of all the previous calculations will be the output percentage that the controller should give, or the duty cycle of the pulse width modulation. Arduino pulse width modulation has a maximum value of 255. With this in mind, the output percentage as a decimal multiplied by 255 gives the value to be output. Once the output value is found, the analogWrite() function is called, using the output value as the argument. This function sends the desired signal to the proportional valve to adjust the valve position. Finally, the rolling RPM value and time are delimited by commas within a string and output through the Arduino serial monitor to the HMI, and the Arduino program begins the loop again.

HMI / COMPUTER APPLICATION

At the start of this project, the design and coding of the PC application had not yet started (as it was largely depending upon real-world data gathered from the completed working project), but we still wanted to consult with Dave Barth on suggestions for HMI design, along with what he would like this program to look and perform like. Dave Barth recommended modeling our program after the ServoStar program used at Edison for servo motor control in the Robotics class.



OPMODE: 0: Digital Speed

Motor: DBL3N00130

Status = OK

Axis: NO ENABLE

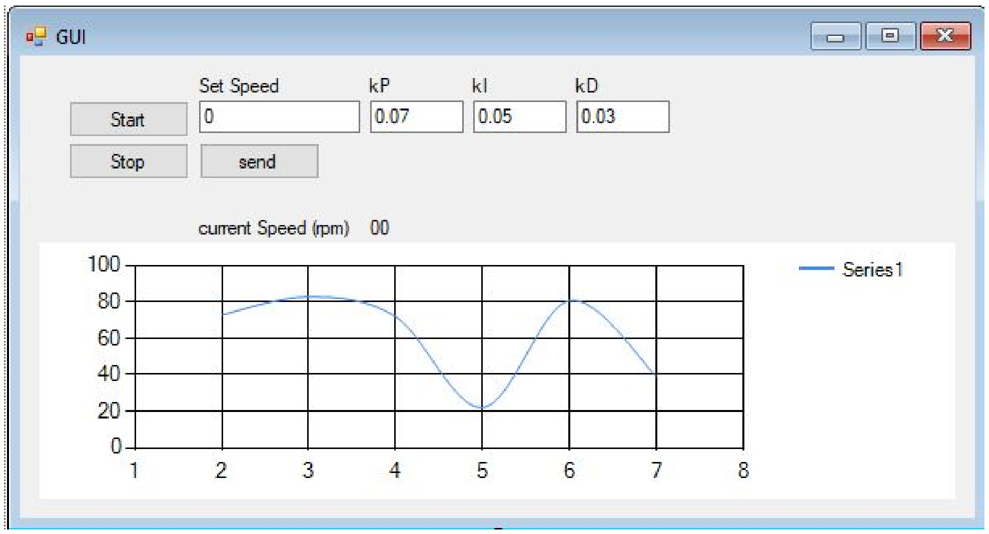
Figure 19: Servo Star UI

The ServoStar program was recommended because it uses PID controls in controlling smaller servo drives. Dave is also very comfortable with the program, and expressed that he would appreciate a program modeled after the ones he already uses for similar applications. The program also performs similar functions to what we intend our program to accomplish - both

ServoStar and our program will use closed-loop feedback and PID controls to control the position of a moving part. We decided to create a program that navigated and functioned similarly to ServoStar, but adapted to our application with an updated modern interface. We also decided to improve upon several of the functions that the ServoStar program offers, such as providing real-time data and graphical representation of motor RPMs and valve supply voltage over time.

The HMI was created using the Visual Basic.NET language within Microsoft Visual Studios. While Visual Studio Enterprise is available for free for engineering students, Visual Studio Community was chosen as the preferred development environment. Along with the Arduino IDE, Visual Studio Community is open source, meaning both applications can freely be updated at no cost to the client. In addition, the client can make adjustments to either program at will.

The first step in creating the HMI was to create a form that fit the design requirements of the client by requesting two RPM setpoints, time to oscillate between setpoints, along with Kp, Ki, and Kd. Additionally, the serial port the Arduino is utilizing needs to be selected in order to have proper communication. The GUI of the HMI accomplishes the port selection with a drop down box (combo box) of available ports; and the rest of the inputs are via editable text boxes. The graph to show the motor RPMs and supply voltages over time was put on the form, as shown in the image below.



Set Speed: 0
kP: 0.07
kI: 0.05
kD: 0.03
current Speed (rpm) 00

Figure 20: Initial Graphical User Interface Layout

Upon starting the HMI application, the program cycles through each serial port available on the computer. If the combo box listing the serial ports does not already contain the port, the port name is added to the combo box for selection, which was necessary to prevent duplicate port names. The first available port found is selected by default and assigned to the port name variable. When each editable text box has the text changed, the respective variable for that box

(Kp, Ki, etc.) is assigned the text. Once all required data is entered, the client can click the start button. When this button is clicked, a custom function to set up the serial port for communication is called, using a concatenated string of all data with comma delimiters. This function sets the parameters of the serial port such as the port to use and baud rate, which must be identical to what the Arduino utilizes. Then, inside of a try statement that will show an error should any issues arise when trying to contact the serial port, the port is opened, the initialization string is sent, the port is closed, and a clock is started. If there is an error, the error message will be displayed and the clock is stopped.

The purpose of the clock is to signify when the HMI should read serial data from the Arduino, and 10 millisecond intervals were chosen arbitrarily. Inside a try statement checking for port timeout, the serial port is opened. If the serial port contains data, check using the BytesToRead function of the serial port, assigning the data string to a variable. The data string is then split using the comma delimiter into the RPM value, valve control voltage value, and time. The current RPM value is then assigned to a label showing the current RPMs and all values are graphed.

SAFETY

Safety has been a top priority for our group and has been discussed throughout the project. The biggest concerns of our project, with regards to safety, have been the hydraulic motor and spinning flywheel. After much discussion, it was clear to our group that we needed to develop some sort of guarding to protect users while the flywheel is spinning.

At first, we wanted to use something robust such as perforated steel or aluminum; however, in the end, it seemed too costly and too heavy to be mounted easily. Speaking with the customer, Dave Barth, a sheet of plexi was offered to the group. The plexi sheet can be formed when heat is applied so it seemed like a perfect solution without having to find a machine shop with a water jet and brake press. The group also liked the fact that it was transparent and the students can observe their labs more easily when working.

EXPECTED FINDINGS

MICROCONTROLLER

There were two primary options that were considered for the controller in the system: the Raspberry PI and the Arduino Uno. The team was familiar with the Arduino. Only one member had used the Raspberry PI before. The Arduino IDE allows for programming in C (10), while the language for the Raspberry PI is dependent on several variables, such as the operating system and physical components present. As such, the Arduino was chosen as the controller for our project. In terms of performance, the Arduino had plenty of processing power to control this system, with adequate response times and stability for an educational setting. The Arduino IDE allows for integration with a PC, which allows for some communication and storing of data (9). The Arduino proved to be an adequate controller up to the ending point of our project.

TACHOMETER

Several options for sensing the revolutions made by the motor were explored. There were three main ideas presented for the creation of an RPM sensor at the start. These three ideas were contact, optical, and proximity tachometers. Initially considering the three, proximity switches seemed dangerous since they created both a pinch point and a crash hazard. The contact sensors would likely be vulnerable to wear and hard to replace. The optical tachometer had weaknesses as well, such as being vulnerable to the environment. The initial choice was the optical tachometer. Further research led to the initial decision to create a tachometer from scratch using infrared LEDs and photodiodes. After component failures and a brief testing of pre-built optical sensors, the tachometer was ultimately chosen to be built using an encoder. The previous concern of wear vulnerability was not applicable to the encoder, as the bearings of the encoder allow for rotation at very low forces. However, the encoder would still be difficult to replace. The encoder works up to about 6000 RPMs, which is well within the maximum RPM rating of the motor in use. There were several points of concern with the encoder. The first was the possibility of false data points, leading to improper RPM calculation and subsequent PID adjustments. The coupling was also a point of concern, since most of the cheaper forms of coupling are rigid and require precise alignment of shafts. As previously expressed, both of these concerns are heightened due to the system being utilized by students.

While there is a chance for some false data points, the calculation of the RPMs from the encoder pulses and timings uses a rolling average that minimizes the impact of data outliers. While the mounting for the encoder is rigid, bolts with springs were used to make fine alignment adjustment simple, and to absorb force should the coupling not be properly aligned. While the coupling is currently well-aligned, it is highly likely that students will run the system without proper alignment. If the coupling is no longer viable, or the client does not want to adjust the alignment, the coupling can be replaced by a section of quarter inch tubing. Edison State already owns large amounts of this tubing, and the tubing will not be subject to enough torque to damage the tubing, while providing flexible coupling of the encoder and motor.

PID CONTROLS / PROPORTIONAL VALVE CONTROLS

There are typically three or four separate variations on PID control that are considered for a system. However, the client specifically requested that PID controls be used to control the proportional valve, so there was no decision to be made. Though convincing the client would probably have been possible, PID controls offered the highest accuracy and stability when compared to other PID variants. The expectations the team had of the PID controls is simple. The PID controls should take the feedback from the RPM sensor reading, and compare the received value to the target value. When the sensed value deviates, the PID controls are utilized by the Arduino to adjust the values closer to target by application of the general equation for PID controls (12).

GUI PROGRAM

When the GUI program was finished, it was expected to be a simple program that would have the capability to show the users how changing the three PID parameters (Kp, Ki, Kd) and target RPMs would affect the valve. The graphs that would be displayed would be a RPM versus time and voltage versus time graph. In the background, the program would be communicating with the Arduino board by sending the user inputs (target RPM,Kp, Ki, Kd) which would then tell the Arduino it was ready to receive data. The data that it would receive would be the command voltage to the valve and the RPM value of the flywheel. Once that data was received, the data would be displayed in the GUI window in real time. The program would also give the user the capability to choose what data to display in the graph and also the option to adjust the X and Y intervals for data viewing purposes.

BRAKE SYSTEM

The braking system sketch (Figure 21) and concept of the brake system design is made with the intent to have the ability to adapt and adjust the brake, as needed. This maneuverability is necessary given that we will not know how much clearance will be available between the motor (Figure 22), flywheel disk (Figure 22), and trainer’s mounting board (Figure 23) each time the system is set up, as it will vary depending on the time, place, and individual assembling of the system. Our bracket mount is adjustable to accommodate for clearance, but is also adaptable so that the brake pads will contact the flywheel's flat surface for equal friction distribution. The braking system works as intended, and is able to apply varying levels of friction to the flywheel.

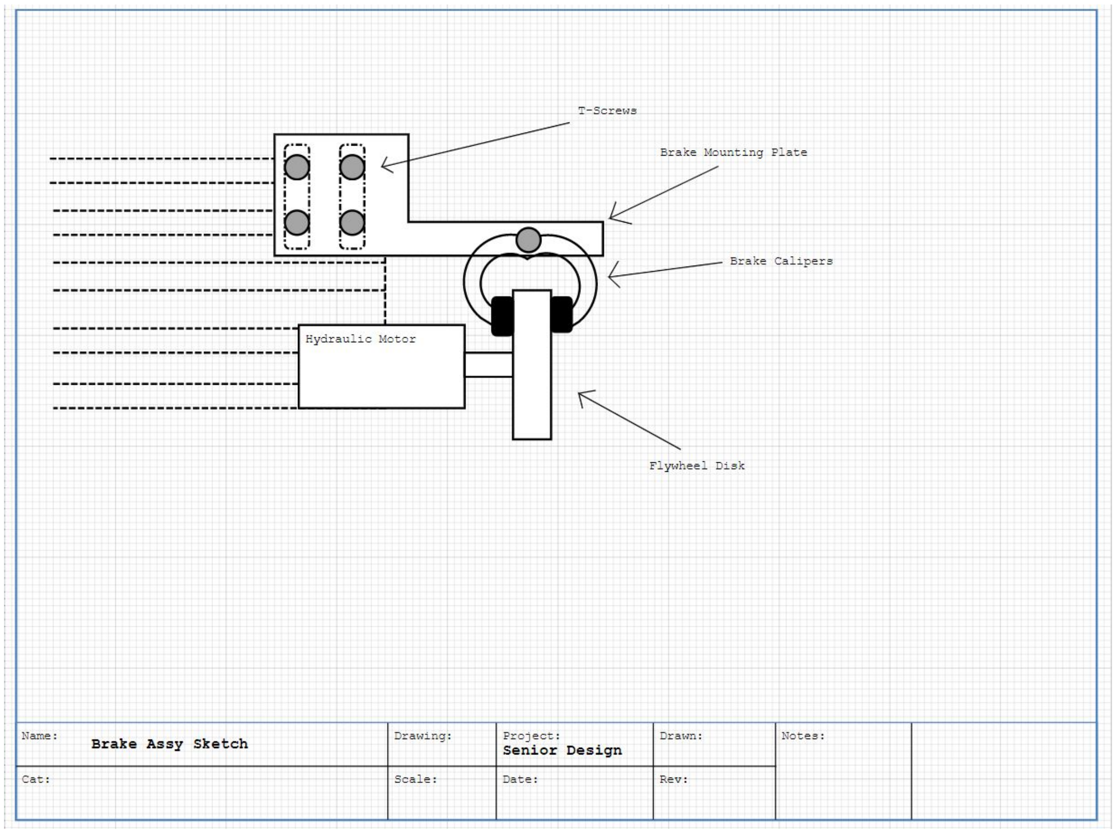
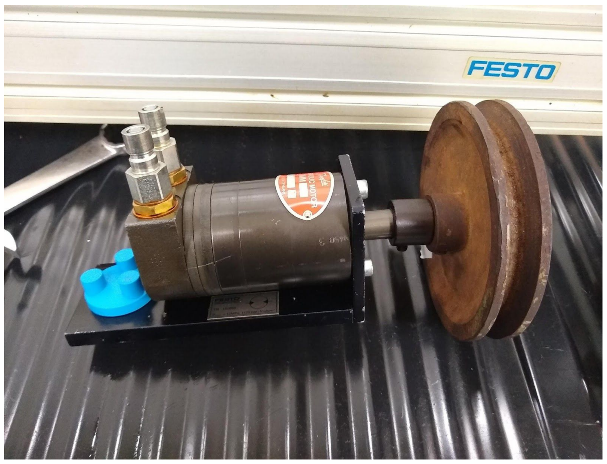


Figure 21: Brake System Sketch



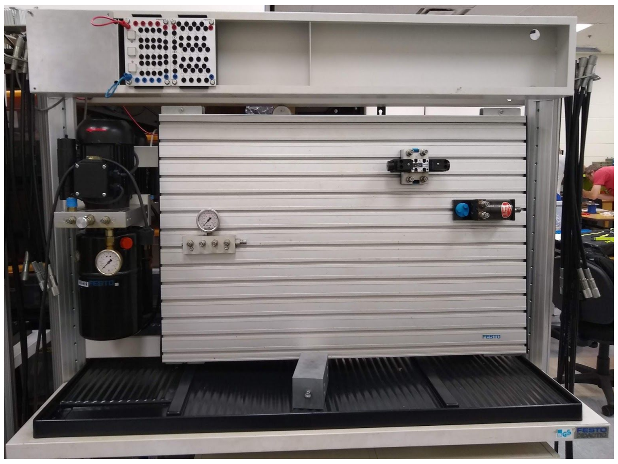


Figure 22: Festo Hydraulic Motor w/ Flywheel

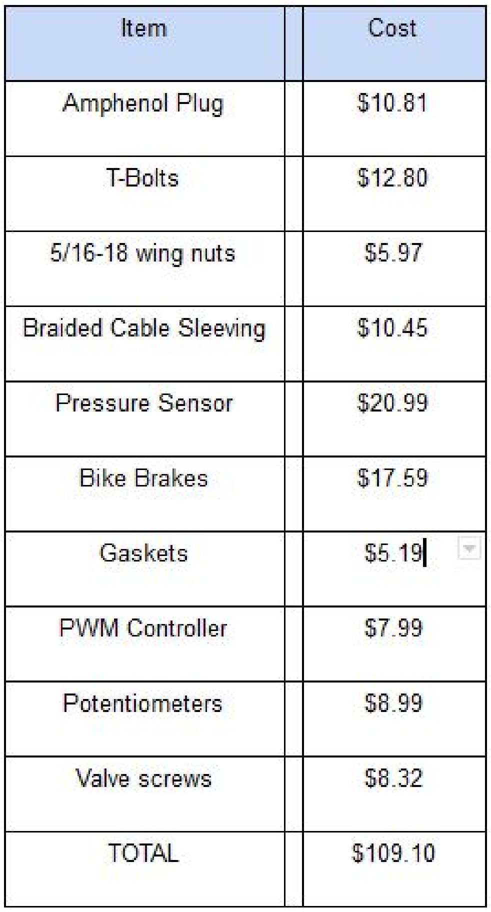
Figure 23: Festo Hydraulic Trainer

BILL OF MATERIALS/ COST ESTIMATES

In order to build this project with the lowest cost possible, a Bill of Materials (BOM) was created to be shared with Edison Community College for funding. Edison supplied us with ‘unlimited funds within reason’ to build our project, and we were directed to keep costs as low as possible. Our initial BOM was made with the intent to document every component we would need for our project, including components we already possessed, minus integral components such as the existing Festo Hydraulic Trainers and proportional valve. This included materials for 3D printing, wire and routing for electrical components, electronics for valve control, etc.

ITEM DESCRIPTION	COST
Arduino	$20.00
Motor Drag System	$150.00
Wiring	$30.00
3D Printer Filament: 3kg	$70.25
IR Emitter and Receiver Diodes	$15.00
Breadboards	$15.00
Soldering Wire	$28.00
JST Connectors	$40.00
Acrylic (For Guards)	$70.00
Aluminum Stock	$30.00
Braided Cable Sleeving	$20.00
Pressure Transducer	$20.00
Amphenol Plug and Cable	$100.00
Sheet Metal	$30.00
Assorted Assemblers	$60.00
Hose Fittings	$50.00
Buck Converter(s)	$40.00
N Channel Logic Level Mosfet	$20.00
USB Data Sync Cable	$10.00
Misc. Electronics	$100.00
Misc. Labor	$100.00
PRELIMINARY TOTAL	$1,018.25
Extra Allowance (Pre. Total + 40%)	$407.30
TOTAL	$1,425.55

Upon completion of the initial BOM we reviewed the components as a team and decided what our first purchases would be. We knew that several of these components would be dependent upon the dimensions and performance of the others, and as such we anticipate ordering more components in the future. As it stands now, our costs up to this point are as follows:



Item	Cost
Amphenol Plug	$10.81
T-Bolts	$12.80
5/16-18 wing nuts	$5.97
Braided Cable Sleeving	$10.45
Pressure Sensor	$20.99
Bike Brakes	$17.59
Gaskets	$5.19
PWM Controller	$7.99
Potentiometers	$8.99
Valve screws	$8.32
TOTAL	$109.10

We did proceed to purchase more items for the project personally, but after COVID-19 halted our progress we proceeded to return unnecessary/unused components, and donated the rest of the components that were stuck on campus with the trainer. We estimate that our costs totaled around $200, but we do not know for certain at this point as we cannot physically tally all of the components used on the trainer.

HYDRAULIC VALVE BLOCK/ MOUNTING METHOD

Upon deciding to combine the valve mounting solution and valve block, we created a hybrid mounting block. The four center holes in the main view of Figure 24 below will route hydraulic fluid to pipe nipples screwed into the sides of the block. The four small holes clustered around the routing channels will be tapped so that the ISO 4401 - specified screws that secure the valve to the block can be used to mount the two. The outermost holes will be dedicated to allowing t-screws to be fed through, which will secure in the slots of the hydraulic trainer and attach the block to the Festo Hydraulic Trainer board by tightening the t-screws in the trainer slots with wingnuts used on the face of the block.

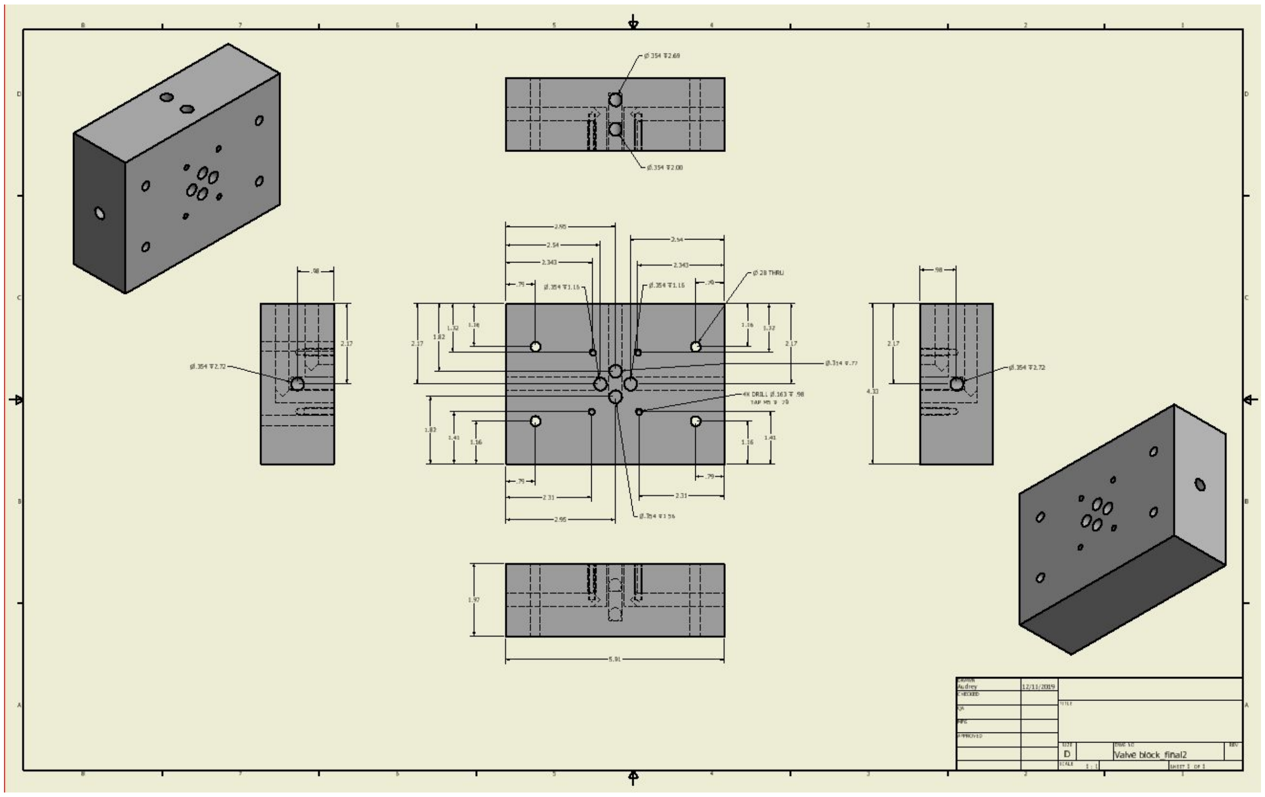


Figure 24: Valve Mounting Block Blueprint

COVID-19 COMPLICATIONS

Unfortunately, due to complications with COVID-19, our group was unable to complete our final project. When Governor DeWine announced the Stay-At-Home Order, the Festo Trainer along with the proportional valve and hydraulic motor were all locked on campus. Neither us nor professors/Dave were able to be on campus, and thus our progress was brought to a halt. (14) Not having access to the trainer meant that we could not measure and gather the data we needed to both physically and electronically complete the project. We also could not progress any further in our coding portions. The following is a summary of what we still had to complete for our project, had COVID-19 not interrupted our project.

ENCODER

We had left our project with an encoder mount already built. This mount was relatively well aligned, but would cause the mounting screws to snap when buttressed to the flywheel on the hydraulic motor. The alignment can be attributed to tensioning springs along the mounting bolt lengths, which allow for fine adjustment and limited movement for absorption of radial load produced at high rotation speed. While the coupling is currently well-aligned, it is highly likely that students will run the system without proper alignment. If the coupling is no longer viable, or the client does not want to adjust the alignment, the coupling can be replaced by a section of quarter inch tubing. Edison State already owns large amounts of this tubing. The tubing will not be subject to enough torque to damage the tubing and will provide flexible coupling of the encoder and motor. Though there is a chance for some false data points, the calculation of the RPMs from the encoder pulses and timings use a rolling average that minimizes the impact of data outliers.

While the mounting solution was finalized and somewhat tested before the pandemic shutdown, there was no chance to test the system as a whole. RPMs could be accurately calculated when the encoder was coupled with the motor. However, there was no chance to test the PID program. Ideally, the PID program could have been tested and fine-tuned to provide both the client and the team with valuable data. Running the system in real-time would have allowed for testing of gain values for the PID program, as well as, discovering limitations to the system. Without being able to test PID controls, the HMI was not able to be fully tested.

VALVE CONTROL

The valve control electronics were very close to being completed. We had been able to get the valve functioning and flipping between ‘forward’ and ‘reverse’ mode, but it had some inconsistencies. The week we were forced to leave our project, we were planning on replacing the operational amplifier with a full H-bridge to see if that would fix our inconsistencies in controlling the valve.

Upon finalizing the valve control voltage electronics and Arduino code, we were going to run the valve and gather real-world data about what valve control voltage corresponded to a certain RPM on the motor flywheel. By using this data, we would have been able to adjust the system to use the PID controls to self-correct and maintain a specified RPM. A rough draft of the control box was sketched using AutoCAD (See Figure 25), but was incomplete because the final valve control circuit, using the H-bridge module, was not tested and confirmed.

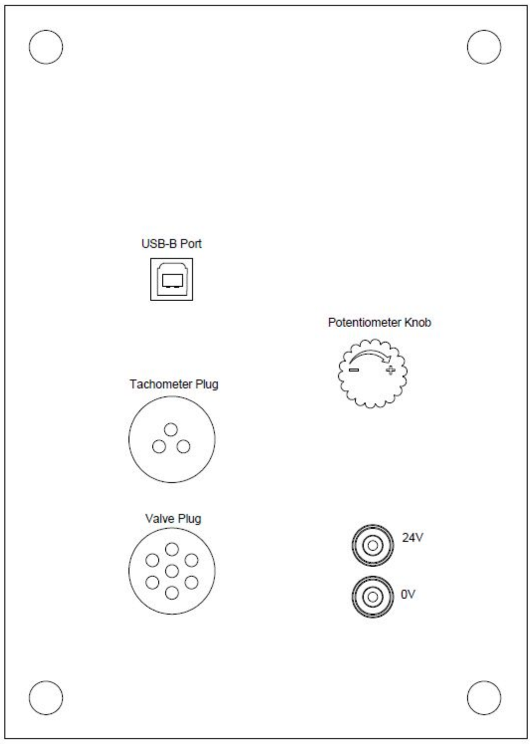


Figure 25: Proposed Control Boxe interface

ARDUINO

At the point we left our project, our Arduino was mostly complete, but still needed some fine-tuning to get it up and functioning. We needed to use data gathered from the trainer to fine-tune the PID controls and enable our code to export relevant data (e.g. the RPMs, PID feedback, etc.) to the GUI program. While the code is written, lack of testing results in uncertainty of the operational integrity.

GUI

The GUI program had progressed relatively far into development when the project was stopped, but is still largely unpolished. While a basic UI was available, complete with input and real-time graphing of the feedback of the system, much of the coding was still rough around the edges and untested. For example, inputs were coded and available for user interaction, but the program did not check that input was positive, numerical, and trimmed of unnecessary zeroes. However, most of the features that the client had requested are present in a pre-finished state.

CUSTOMER CONSULTATION

After completing our project, we had intended to consult with Dave again to get his final opinions on what we had done. We would have used this feedback to refine our final product, especially with concern to the overall GUI design and any other portion of the project with significant interactivity (e.g the braking system, PID parameters, etc.). We would have worked with Dave over the final weeks to produce a product that he would enjoy using in his lessons. We also would have worked closely with Dave to produce a useful troubleshooting manual and lesson plans for use at Edison.

TROUBLESHOOTING MANUAL/ LESSON PLANS

The last portion of our project would have been to create a complete and thorough troubleshooting manual, along with lesson plans to accompany the trainer. The troubleshooting manual would have contained detailed descriptions of each component of the trainer, accompanied by diagrams, the function of each component, and how to set up and tear down the entire project. It would have also contained suggestions for solutions to common problems with the system and other suggestions for general troubleshooting.

The lesson plans would have been co-written with Dave, and would have been formed around his input. They would have contained a complete proportional valve lesson, in-class worksheets, and homework to accompany proportional valve calculations.

CONCLUSION

We conclude that our project contains very strong elements that make it an excellent senior design project. While there were several significant issues that merited redesign, and even revision of prototypes, there were no significant cost changes, which was a major concern of the project. The project was in a good place for completion pre-COVID complications. We look forward to handing this project back to Dave Barth, who will determine how he will assign or finish the remaining portions.

This course has proven significant to our liberal educations and has emphasized the importance of lifelong continued education to all our group members. We have come to learn that our education is something we will take with us and use throughout our lives, especially in this career field. In researching and applying both past and learned skills for the project, we have come to appreciate the skills, knowledge, and documentation provided by others in the field. They have helped us research, learn, and complete a system design for a complicated project. We hold a respect for their past experience and knowledge and understand that differing perspectives can be brought together to create a more robust and complete, finished product. We are excited to take the knowledge we gained, as well as, a love for learning with us in the future.

Further testing is highly recommended for all systems involved and should be conducted after the system as a whole is completed. Our project requires significant, real-world data in order to fine-tune the PID controls and PC program, which definitely need to be gathered as work on the project is continued. In addition, we also advise placing a specific focus on system safety by properly inspecting our project and making safe design choices that we had not been able to make. Notable examples of this would be shielding the moving portions of the motor and braking system, properly insulating and routing wiring, and securely placing key electrical components in the appropriate location in the electrical control box, isolated from all dangerous mechanical and liquid components of our system. We look forward to the progress and completion of our project after the point where we were forced to leave it off.

REFERENCES

1. Festo. *Equipment set TP 501 – Basic level: Basic training in hydraulics*. 2019. https://www.festo-didactic.com/us-en/learning-systems/equipment-sets/hydraulics/training-packages/equipment-set-tp-501-basic-level-basic-training-in-hydraulics.htm?fbid=dXMuZW4uNTc5LjE3LjE4LjU1Ni43NTk2. 21 September 2019.

2. *Proportional Directional Control Valves Series D1FP*. *Proportional Directional Control Valves Series D1FP*, Parker Hannifin Corporation, 2019. https://www.parker.com/Literature/Hydraulic%20Valve%20Division/hydraulicvalve/Catalog%20sections%20for%20websphere/Proportional%20Directional%20Control/Catalog%20-%20Static%20Files/D1FP.pdf

3. Parker. "Parker Engineering Your Success Motion Control." 2019. *Parker Hannifin Corp.* https://www.parker.com/Literature/Hydraulic%20Controls%20Europe/Manuals%20UK/D_FP_20%205715-658%20UK.pdf. 27 September 2019.

4. Bracey, James. *Disc brakes: everything you need to know - Cycling Weekly*. n.d. https://www.cyclingweekly.com/news/product-news/everything-you-need-to-know-about-disc-brakes-202130. 10 October 2019.

5. Unibrow. *Controlling a 5V proportional Valve with PWM*. n.d. https://forum.arduino.cc/index.php?topic=489936.0. 10 October 2019.

6. Varunvp. *Help required for controlling proportional solenoid valve!* n.d. https://forum.arduino.cc/index.php?topic=467353.0. 10 October 2019.

7. Kollmorgen. "Motion Control Solutions | Kollmorgen | Industrial Servomotors Servo Drives AC DC Motors." n.d. https://www.kollmorgen.com/sites/default/files/public_downloads/S400%20Servo%20Drive%20Quick%20Start%20Guide%20EN.pdf. 17 October 2019.

8. Amphenol. "Farnell | Electronic Component Distribution." n.d. *C 16-1_C 16-3 englisch.pdf - 9569.* http://www.farnell.com/datasheets/9569.pdf. 14 October 2019.

9. *Arduino Uno Rev3*, https://store.arduino.cc/usa/arduino-uno-rev3

10. "Language Reference." *Arduino Reference*, 2019, https://www.arduino.cc/reference/en/.

11. Instructables, Tanay. "Measure RPM - Optical Tachometer." *Instructables*, Instructables, 13 Oct. 2017, https://www.instructables.com/id/Measure-RPM-DIY-Portable-Digital-Tachometer.

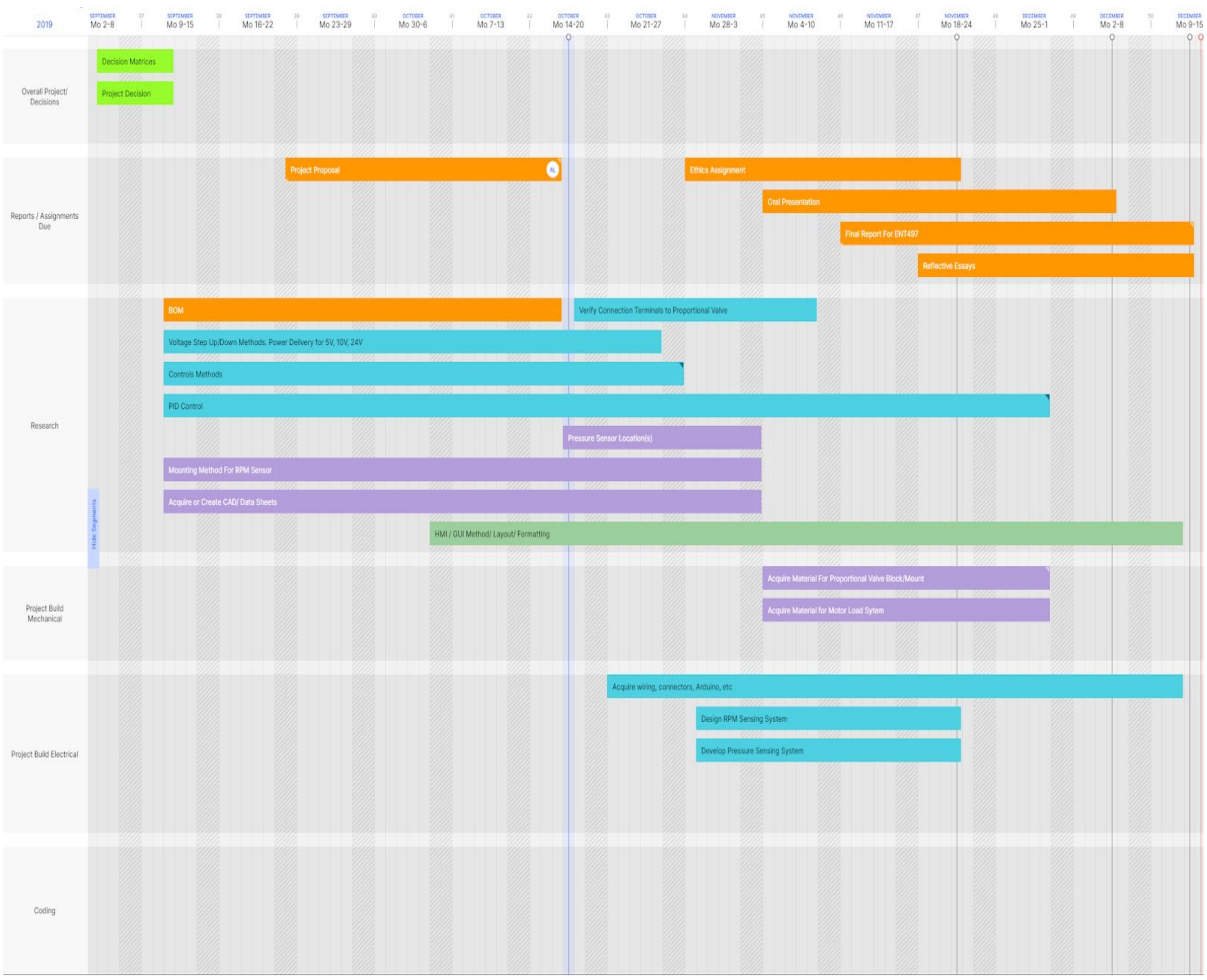
12. Roger, Siefried. "PID Control." ENT 418 Electromechanical Control Systems. 5 Nov. 2019, Middletown, Ohio.

13. Beauregard, Brett. "Arduino PID Library." *Arduino Playground - PIDLibrary*, Arduino, 2017, playground.arduino.cc/Code/PIDLibrary.

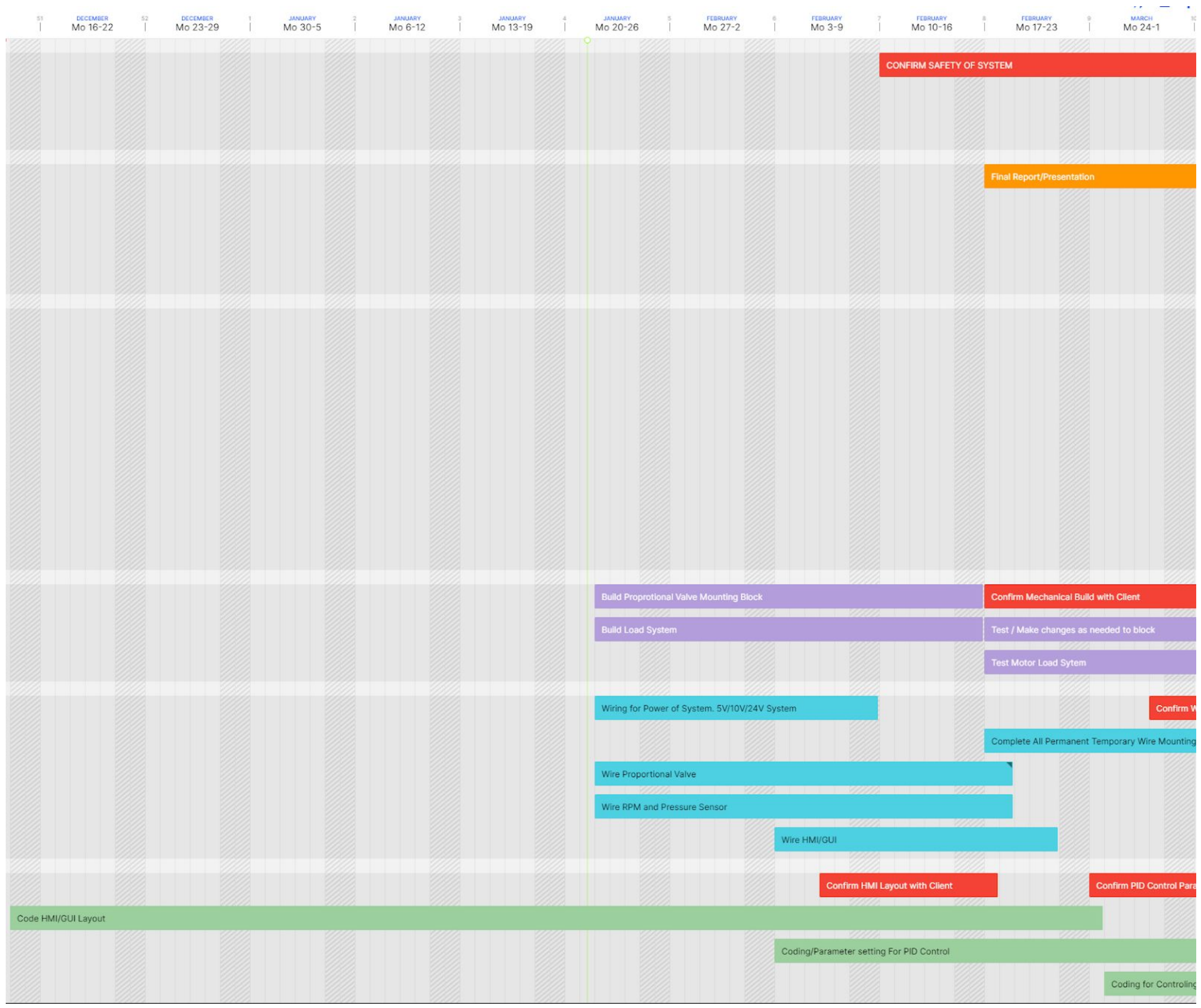
14. Acton, Amy. "Amended Director's Stay At Home Order." *Ohio Government, 2020,* https://coronavirus.ohio.gov/static/publicorders/Directors-Stay-At-Home-Order-Amended-04-02-20.pd

APPENDICES

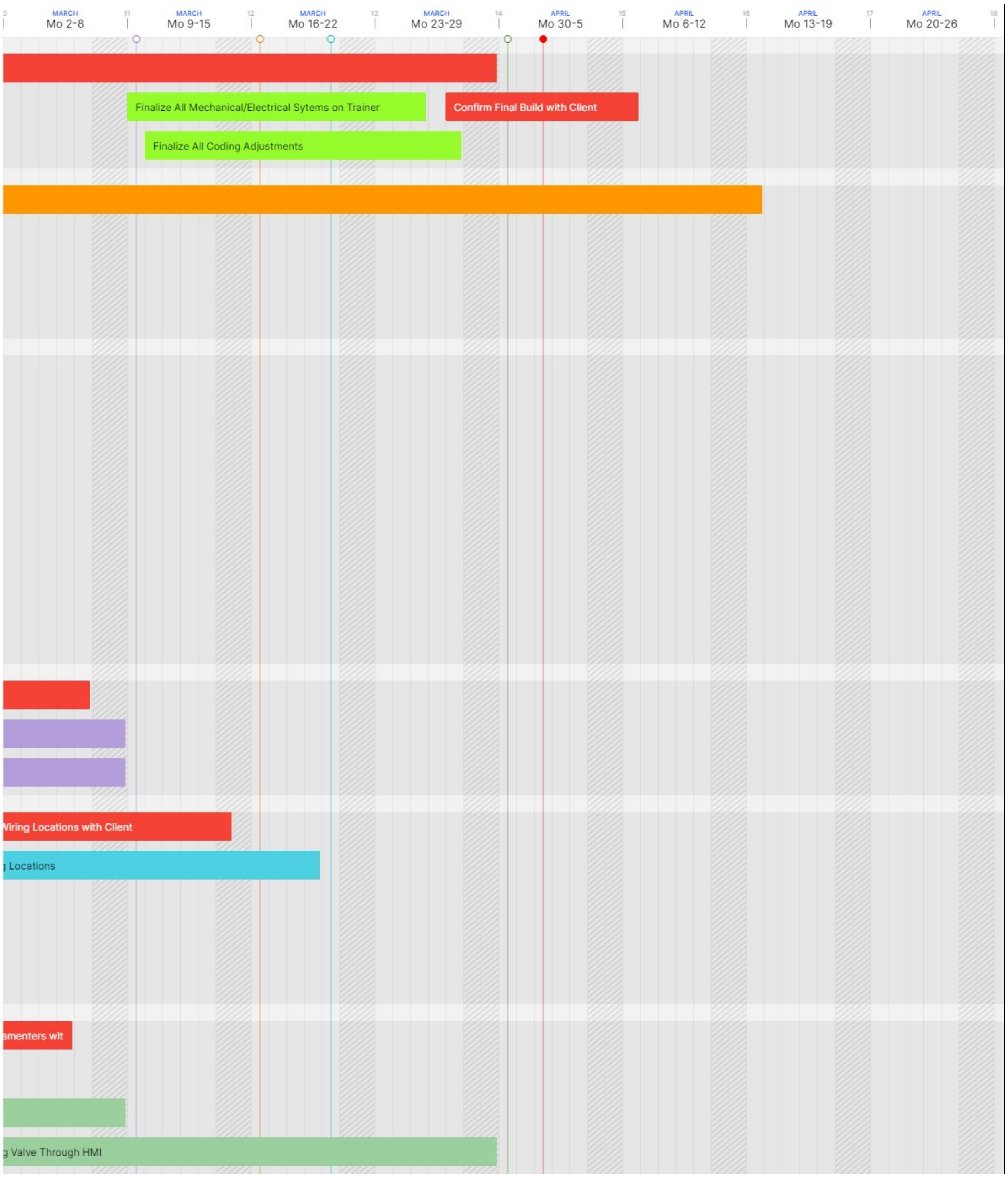
Appendix A: Gantt Chart and Group Schedule



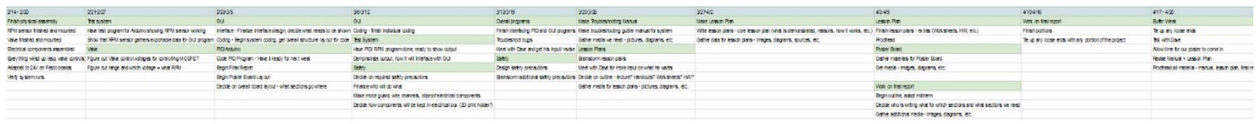
Gantt Chart: September - November



Gantt Chart: December - February



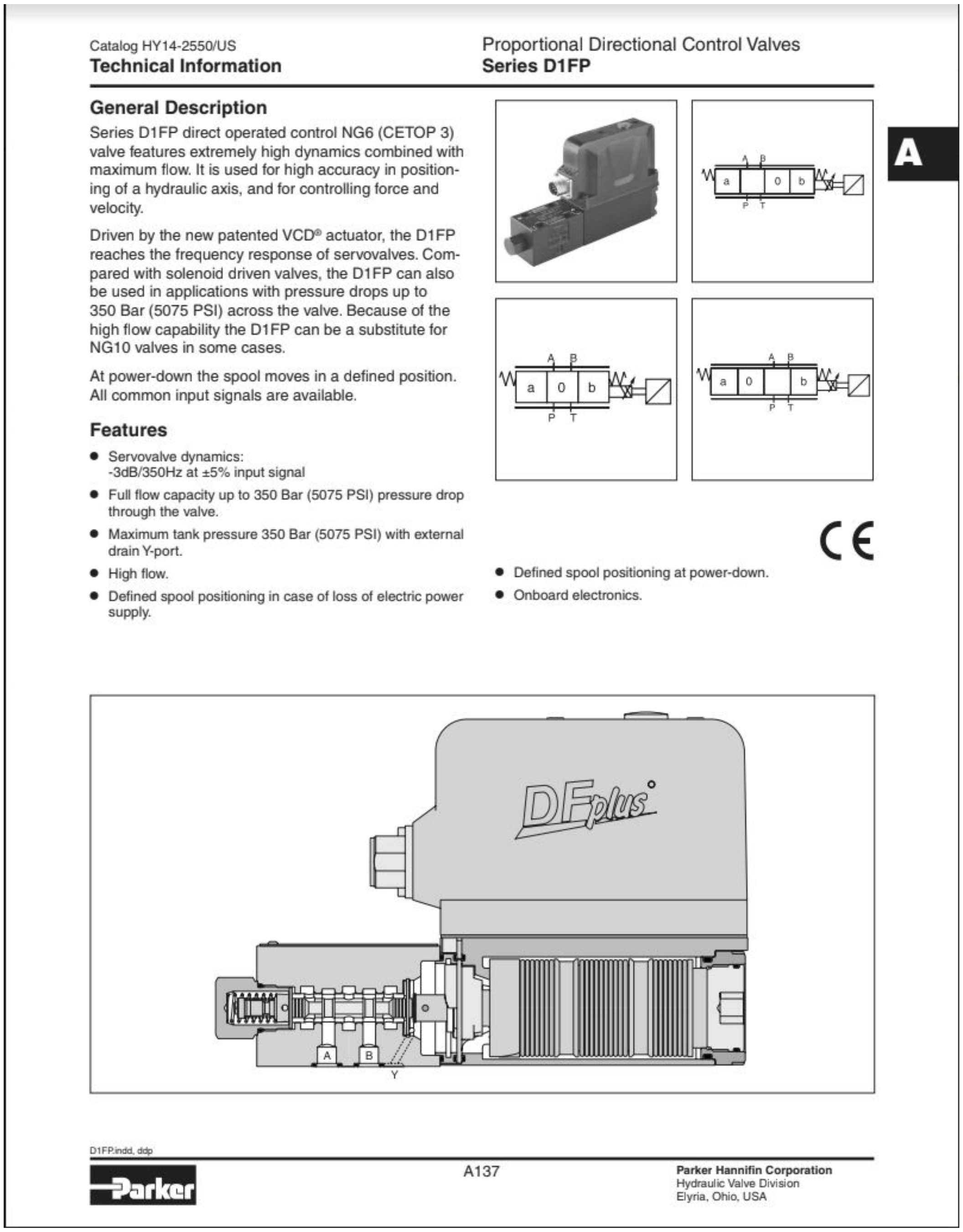
Gantt Chart: March - April



2/14 - 2/20	2/21-2/27	2/28-3/5	3/6-3/12	3/12-3/19	3/20-3/26	3/27-4/2	4/3-4/9	4/10-4/16	4/17 - 4/23
Finish physical assembly	Test system	GUI	GUI	Overall programs	Make Troubleshooting Manual	Make Lesson Plan	Lesson Plan	Work on final report	Buffer Week
RPM sensor finished and mounted	Have test program for Arduino showing RPM sensor working	Interface - Finalize interface design, decide what needs to be shown	Coding - finish individual coding	Finish interfacing PID and GUI programs	Make troubleshooting guide/ manual for system	Write lesson plans - core lesson plan (what is demonstrated, reasons, how it works, etc.)	Finish lesson plans - extras (Worksheets, HW, etc.)	Finish portions	Tie up any loose ends
Valve finished and mounted	Show that RPM sensor gathers exportable data for GUI program	Coding - Begin system coding, get overall structure/ layout for code	Test System	Troubleshoot bugs	Gather media we need - pictures, diagrams, etc	Gather data for lesson plans - images, diagrams, sources, etc.	Proofread	Tie up any loose ends with any portion of the project	Talk with Dave
Electrical components assembled	Valve	PID/Arduino	Have PID/ RPM program done, ready to show output	Meet with Dave and get his input/ revise	Lesson Plans		Poster Board		Allow time for our poster to come in
Everything wired up (esp. valve controls)	Figure out Valve control voltages for controlling MOSFET	Code PID Program - Have it ready for next week	Demonstrate output, how it will interface with GUI	Safety	Brainstorm lesson plans		Gather materials for Poster Board		Revise Manual + Lesson Plan
Adapted to 24V on Festo boards	Figure out range and which voltage = what RPM	Begin Final Report	Safety	Design safety precautions	Meet with Dave for more input on what he wants		Get media - images, diagrams, etc.		Proofread all material - manual, lesson plan, final re
Verify system runs		Begin Poster Board Layout	Decide on required safety precautions	Brainstorm additional safety precautions	Decide on outline - lecture? Handouts? Worksheets? HW?				
		Decide on overall board layout - what sections go where	Finalize who will do what		Gather media for lesson plans - pictures, diagrams, etc.		Work on final report		
			Make motor guard, wire channels, dipproof electrical components				Begin outline, adapt midterm		
			Decide how components will be kept in electrical box (3D print holder?)				Decide who is writing what for which sections and what sections we need		
							Gather additional media - images, diagrams, etc.		

Gantt Chart: Excel Version

Appendix B: Proportional Directional Valve Specifications



Catalog HY14-2550/US
Technical Information

Proportional Directional Control Valves
Series D1FP

A

General Description

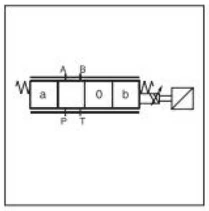
Series D1FP direct operated control NG6 (CETOP 3) valve features extremely high dynamics combined with maximum flow. It is used for high accuracy in positioning of a hydraulic axis, and for controlling force and velocity.

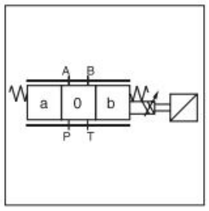
Driven by the new patented VCD® actuator, the D1FP reaches the frequency response of servovalves. Compared with solenoid driven valves, the D1FP can also be used in applications with pressure drops up to 350 Bar (5075 PSI) across the valve. Because of the high flow capability the D1FP can be a substitute for NG10 valves in some cases.

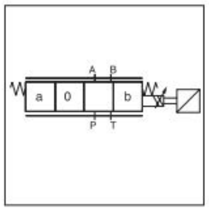
At power-down the spool moves in a defined position. All common input signals are available.

Features

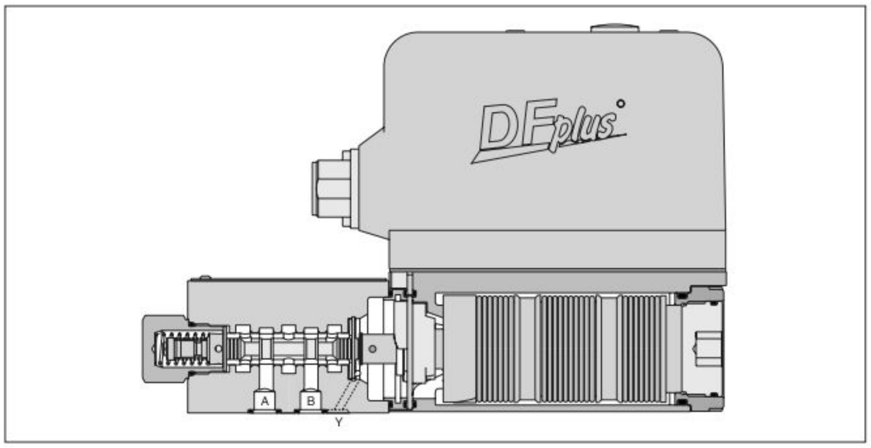
- Servovalve dynamics:
  -3dB/350Hz at ±5% input signal
- Full flow capacity up to 350 Bar (5075 PSI) pressure drop through the valve.
- Maximum tank pressure 350 Bar (5075 PSI) with external drain Y-port.
- High flow.
- Defined spool positioning in case of loss of electric power supply.
- Defined spool positioning at power-down.
- Onboard electronics.







CE



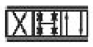
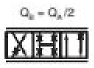
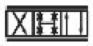
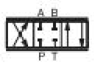
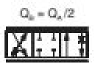
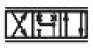
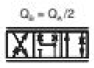
D1FP.indd, ddp

Parker

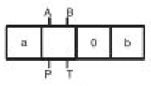
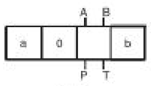
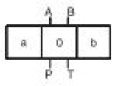
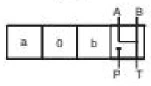
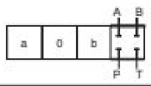
A137

Parker Hannifin Corporation
Hydraulic Valve Division
Elyria, Ohio, USA

D	1	F	P			9				0	
Directional Control Valve	Size DIN NG6 CETOP 3 NFPA D03	Proportional Control	VCD	Spool Type	Spool Position on Power Down [1]	Y-Port Plugged [4]	Seal	Input Signal	Options	Spool/ Sleeve Design	Design Series NOTE: Not required when ordering.

Code	Spool	Flow LPM (GPM) at Δp 35 Bar (508 PSI) per metering edge
	Zerolap	
E50M E50H E50G E50F E50C E50B		40 (10.6) 25 (6.6) 16 (4.2) 12 (3.2) 6 (1.6) 3 (0.8)
B60M B60H B60G B60F B60C		40 (10.6) / 20 (5.3) 25 (6.6) / 12.5 (3.3) 16 (4.2) / 8 (2.1) 12 (3.2) / 6 (1.6) 6 (1.6) / 3 (0.8)
	Underlap approximately -0.5%	
E55M E55H E55G E55F E55C E55B		40 (10.6) 25 (6.6) 16 (4.2) 12 (3.2) 6 (1.6) 3 (0.8)
	Overlap 25%	
E01M E01H E01G E01F E01C E01B		40 (10.6) 25 (6.6) 16 (4.2) 12 (3.2) 6 (1.6) 3 (0.8)
B31M B31H B31G B31F B31C		40 (10.6) / 20 (5.3) 25 (6.6) / 12.5 (3.3) 16 (4.2) / 8 (2.1) 12 (3.2) / 6 (1.6) 6 (1.6) / 3 (0.8)
E02M E02H E02G E02F E02C E02B		40 (10.6) 25 (6.6) 16 (4.2) 12 (3.2) 6 (1.6) 3 (0.8)
B32M B32H B32G B32F B32C		40 (10.6) / 20 (5.3) 25 (6.6) / 12.5 (3.3) 16 (4.2) / 8 (2.1) 12 (3.2) / 6 (1.6) 6 (1.6) / 3 (0.8)

Please order plugs separately. See Accessories.

Code	Spool Position on Power Down
A [2]	
B [2]	
C [3]	
H [6]	
J [6]	

Code	Description
N	Nitrile
V	Fluorocarbon
H	For HFC Fluid

Code	Description
0	6 + PE acc. EN175201-804
5	11 + PE acc. EN175201-804
7	6 + PE + Enable

Code	Signal	Flow Direction [5]
B	+/- 10V	0...+10V -> P-A
E	+/- 20mA	0...+20mA -> P-A
S	4...20mA	12...20mA -> P-A

[1] On power down the spool moves in a defined position. This cannot be guaranteed in case of single flow path on the control edge A→ T resp. B→ T with pressure drops above 120 Bar (1740 PSI) or contamination in the hydraulic fluid.

[2] Approximately 10% opening, only available with zerolap spools and underlap spools.

[3] Only available with overlap spools.

[4] Needs to be removed at tank pressure >35 Bar (507.5 PSI).

[5] Flow direction P→ A with Pin D > Pin E.

[6] Not for flow code M.

Bolt Kit:		
BK209	(4)	10-24x1.25
BK375	(4)	M5x30
Weight:	5.0 kg (11.0 lbs.)	

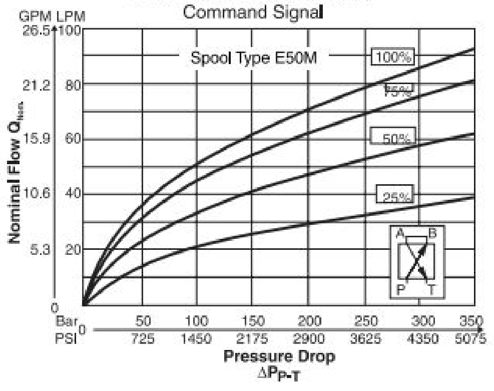
General		
Design		Direct operated proportional DC valve
Actuation		VCD® actuator
Size		NG6 / CETOP 3 / NFPA D03
Mounting Interface		DIN 24340 / ISO 4401 / CETOP RP121 / NFPA
Mounting Position		Unrestricted
Ambient Temperature	**[°C]**	-20...+50; (-4°F...+122°F)
MTTF$_D$ Value	**[years]**	75
Vibration Resistance	**[g]**	10 Sinus 5...2000 Hz acc. IEC 68-2-6 30 Random noise 20...2000 Hz acc. IEC 68-2-36 15 Shock acc. IEC 68-2-27
Hydraulic		
Maximum Operating Pressure		Ports P, A, B 350 Bar (5075 PSI) Port T max. 35 Bar (508 PSI), port Y max. 35 Bar (508 PSI) [1]
Fluid		Hydraulic oil as per DIN 51524...51535, other on request
Fluid Temperature	**[°C]**	-20...+60; (-4°F...+140°F)
Viscosity		
Permitted	**[cSt] / [mm²/s]**	20...380 (93...1761 SSU)
Recommended	**[cSt] / [mm²/s]**	30...80 (139...371 SSU)
Filtration		ISO 4406 (1999) 18/16/13 (acc. NAS 1638: 7)
Nominal Flow at Δp=35 Bar (508 PSI) per Control Edge [2]		3 LPM (0.08 GPM) / 6 LPM (1.6 GPM) / 12 LPM (3.2 GPM) / 25 LPM (6.6 GPM) / 40 LPM (10.6 GPM)
Flow Maximum		90 LPM (23.8 GPM) at Δp=350 Bar (5075 PSI) over two control edges
Leakage at 100 Bar (1450 PSI)	**[ml/min]**	<400 (zerolapped spool); <50 (overlapped spool)
Static / Dynamic		
Step Response at 100% Step [3]	**[ms]**	<3.5
Frequency Response (±5% signal) [3]	**[Hz]**	350 (amplitude ratio -3dB), 350 (phase lag -90°)
Hysteresis	**[%]**	<0.05
Sensitivity	**[%]**	<0.03
Temperature Drift	**[%/K]**	<0.025
Electrical		
Duty Ratio	**[%]**	100 ED; CAUTION: Coil temperature up to 150°C (302°F) possible
Protection Class		IP65 in accordance with EN 60529 (plugged and mounted)
Supply Voltage/Ripple	**[V]**	DC 22 ... 30, ripple <5% eff., surge free
Current Consumption Maximum	**[A]**	3.5
Pre-Fusing	**[A]**	4.0 medium lag
Input Signal		
Voltage	**[V]**	10...0...-10, ripple <0.01% eff., surge free, 0...+10V P->A
Impedance	**[kOhm]**	100
Current	**[mA]**	20...0...-20, ripple <0.01% eff., surge free, 0...+20mA P->A
Impedance	**[Ohm]**	250
Current	**[mA]**	4...12...20, ripple <0.01% eff., surge free, 12...20mA P->A <3.6 mA = disable, >3.8 mA = according to NAMUR NE43
Impedance	**[Ohm]**	250
Differential Input Maximum		
Code 0	**[V]**	30 for terminal D and E against PE (terminal G)
Code 5 / 7	**[V]**	30 for terminal 4 and 5 against PE (terminal ⏚)
Enable Signal (Only Code 5 / 7)	**[V]**	5...30, Ri = 9 kOhm
Diagnostic Signal	**[V]**	+10...0...-10 / +Ub, rated max. 5mA
EMC		EN61000-6-2 / EN61000-6-4
Electrical Connection	**Code 0**	6 + PE acc. EN 175201-804
	Code 5	11 + PE acc. EN 175201-804
	Code 7	6 + PE + Enable
Wiring Miniimum		
Code 0	**[mm²]**	7x1.0 (AWG 18) overall braid shield
Code 5	**[mm²]**	12x1.0 (AWG 20) overall braid shield
Code 7	**[mm²]**	12x1.0 (AWG 18) overall braid shield
Wiring Length Maximum	**[m]**	50 (164 ft.)

[1] For applications with pT>35 Bar (508 PSI) the Y-port has to be connected and the plug in the Y-port has to be removed.

[2] Flow rate for different Δp per control edge: $Q_x = Q_{Nom.} \cdot \sqrt{\frac{\Delta p_x}{\Delta p_{Nom.}}}$

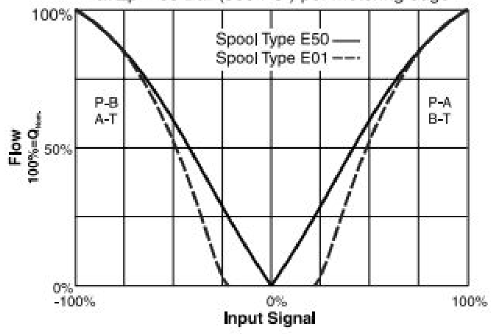
[3] Measured with load 100 Bar (1450 PSI) pressure drop/two control edges.

Functional Limit
at 25%, 50%, 75% and 100%
Command Signal

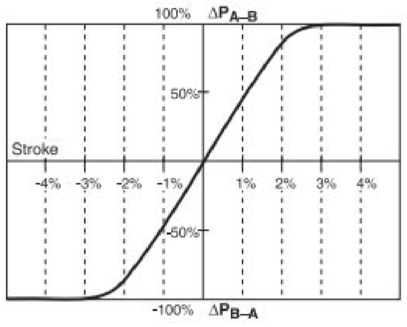


Spool Type **E01/E50**

Flow Curves
at Δp = 35 Bar (508 PSI) per metering edge

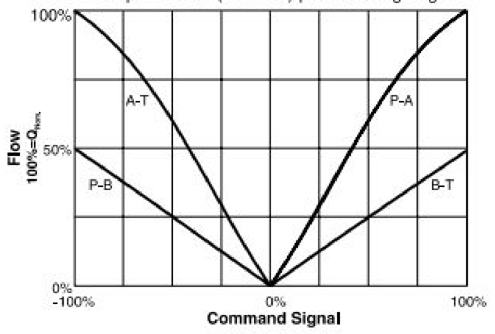


Pressure Gain

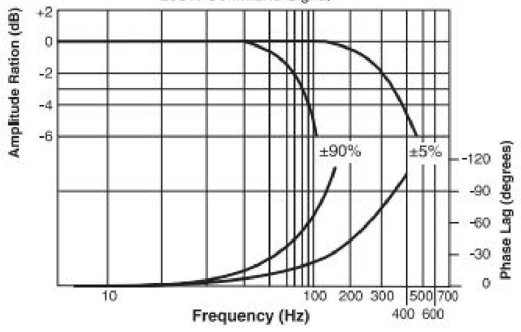


Spool Type **B60**

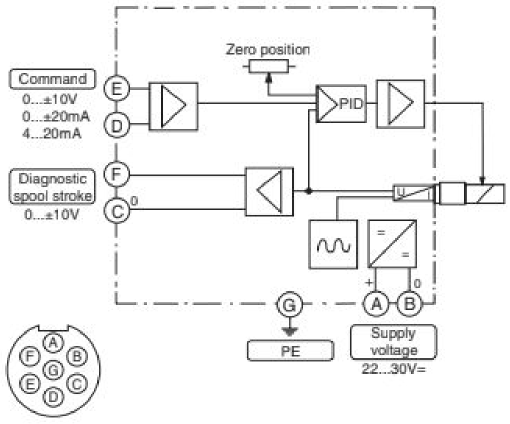
Flow Curves
at Δp = 35 Bar (508 PSI) per metering edge



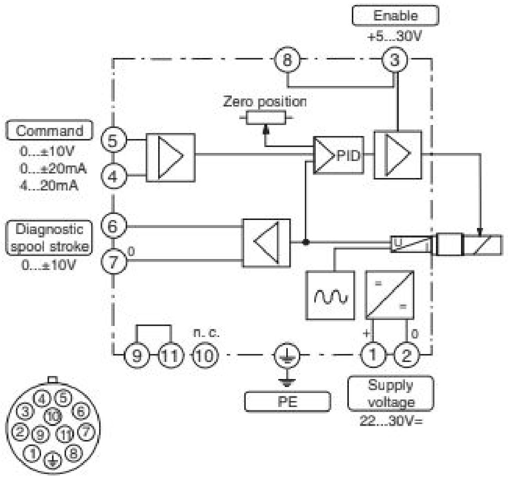
Frequency Response
±5% Command Signal
±90% Command Signal



Code 0
6 + PE acc. to EN 175201-804

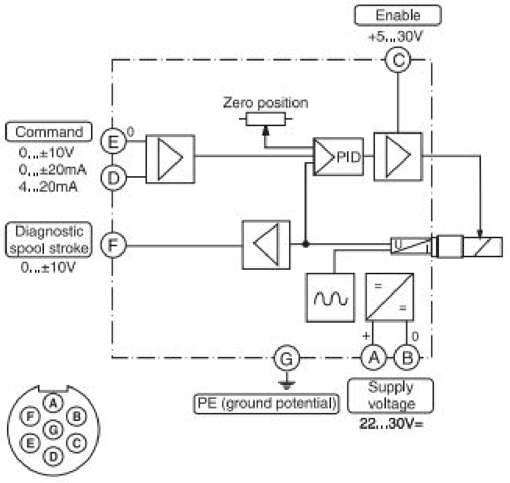


Code 5
11 + PE acc. to EN 175201-804

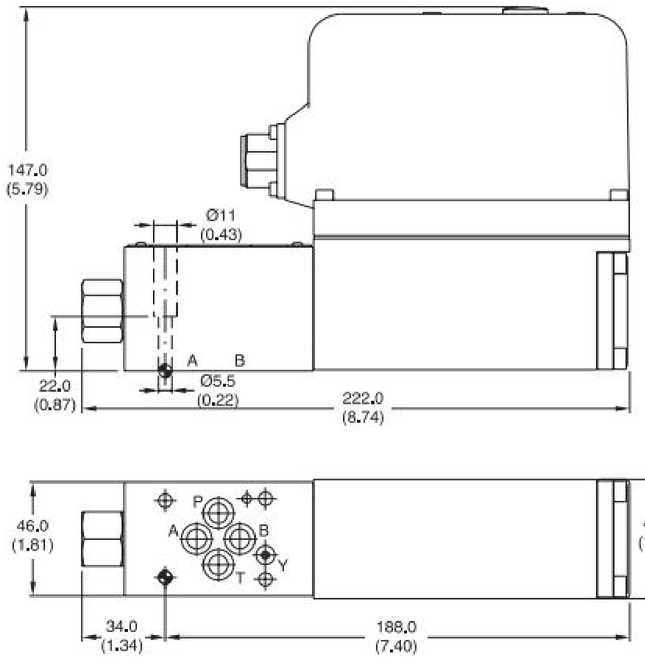


Note: When replacing another valve, verify Pin C is 0 V and not wired as an enable.

Code 7
6 + PE + Enable acc. to EN 175201-804

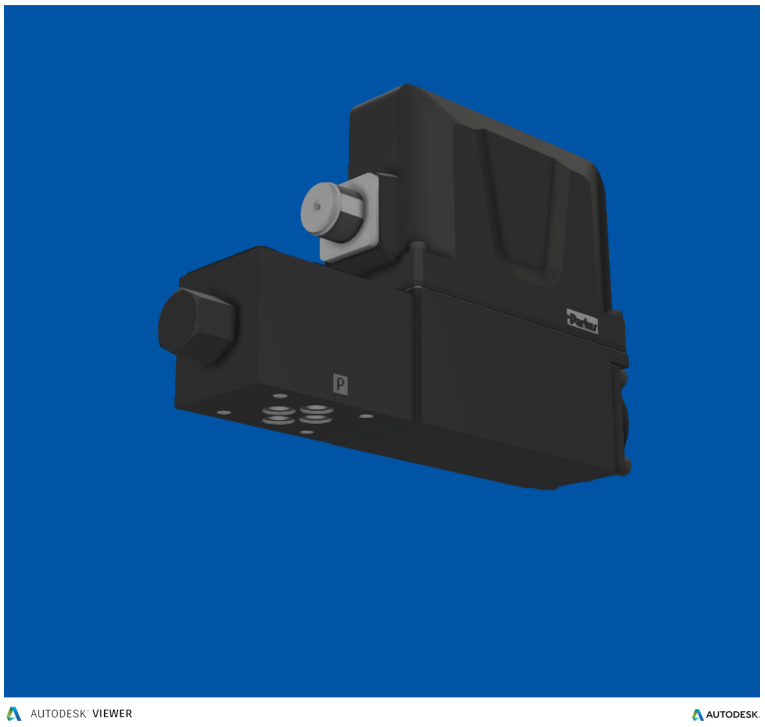


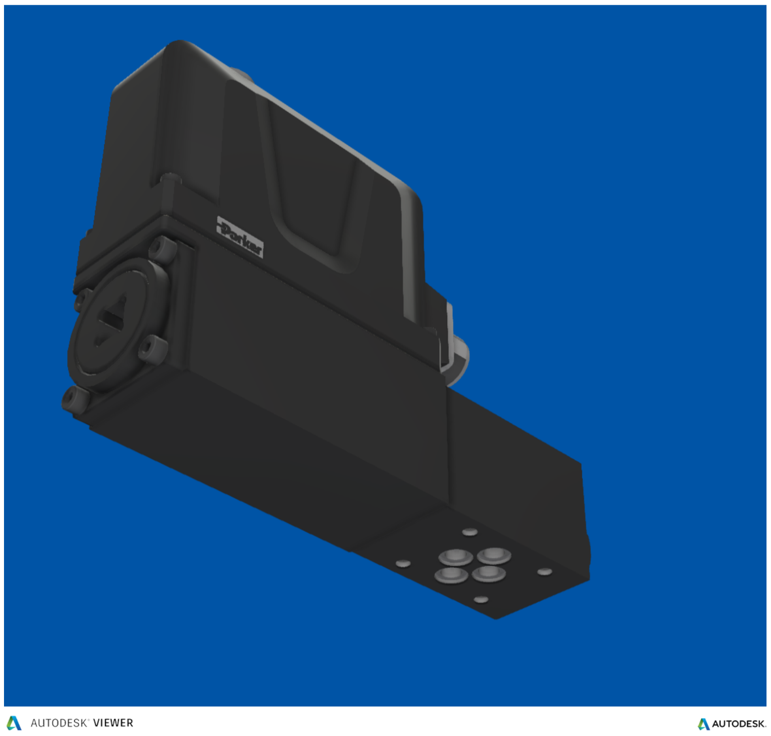
Inch equivalents for millimeter dimensions are shown in (**)



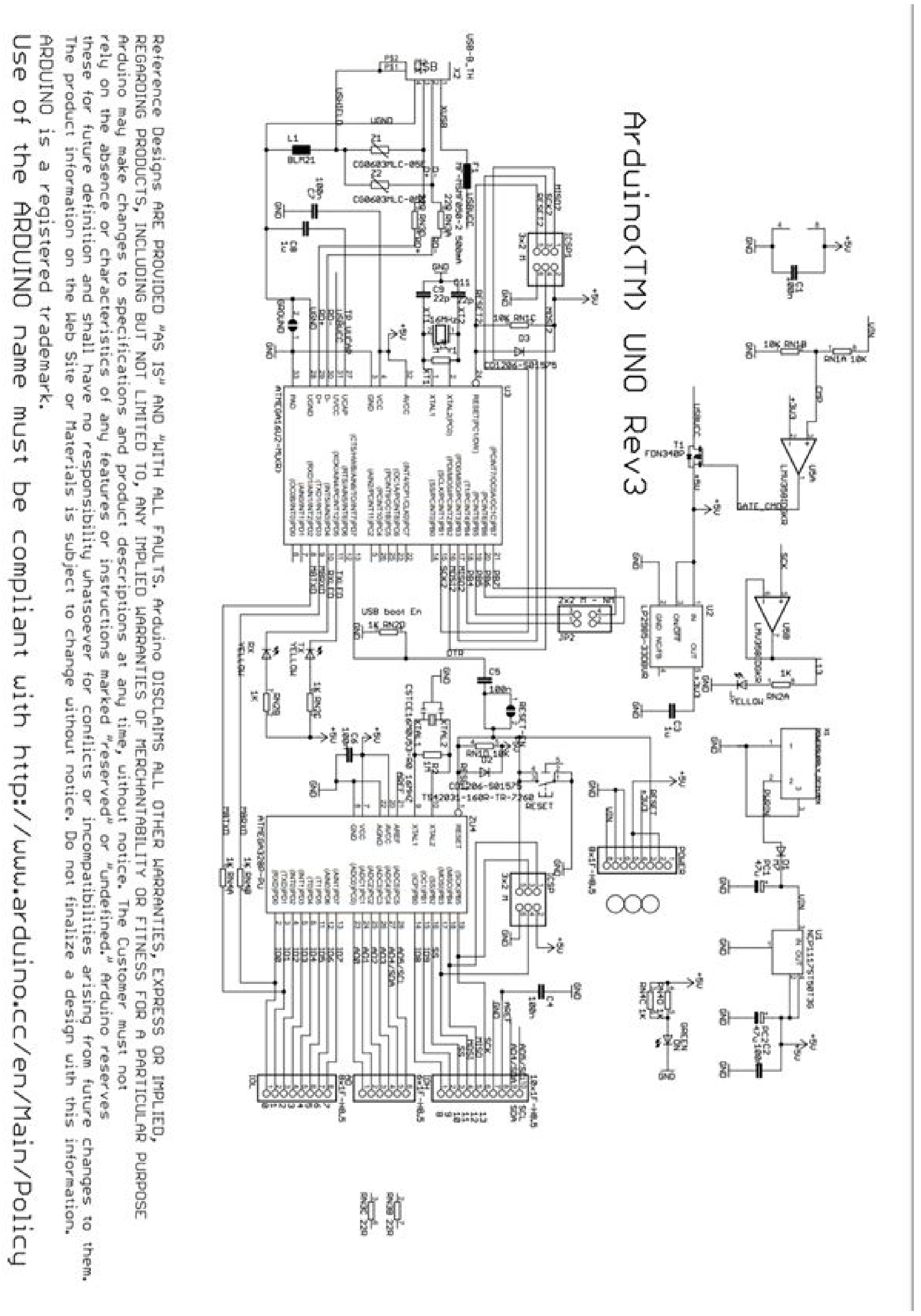
Surface Finish	Kit			Seal ⬭ Kit
$\sqrt{R_{max}6.3}$ ⏥ 0.01/100	BK375 BK209	4x M5x30 DIN 912 12.9 4x 10-24x1.25	7.6 Nm (5.6 lb.-ft.) ±15 %	Nitrile: SK-D1FP Fluorocarbon: SK-D1FP-V for HFC Fluid: SK-D1FP-H

Appendix C: 3D View of Proportional Valve





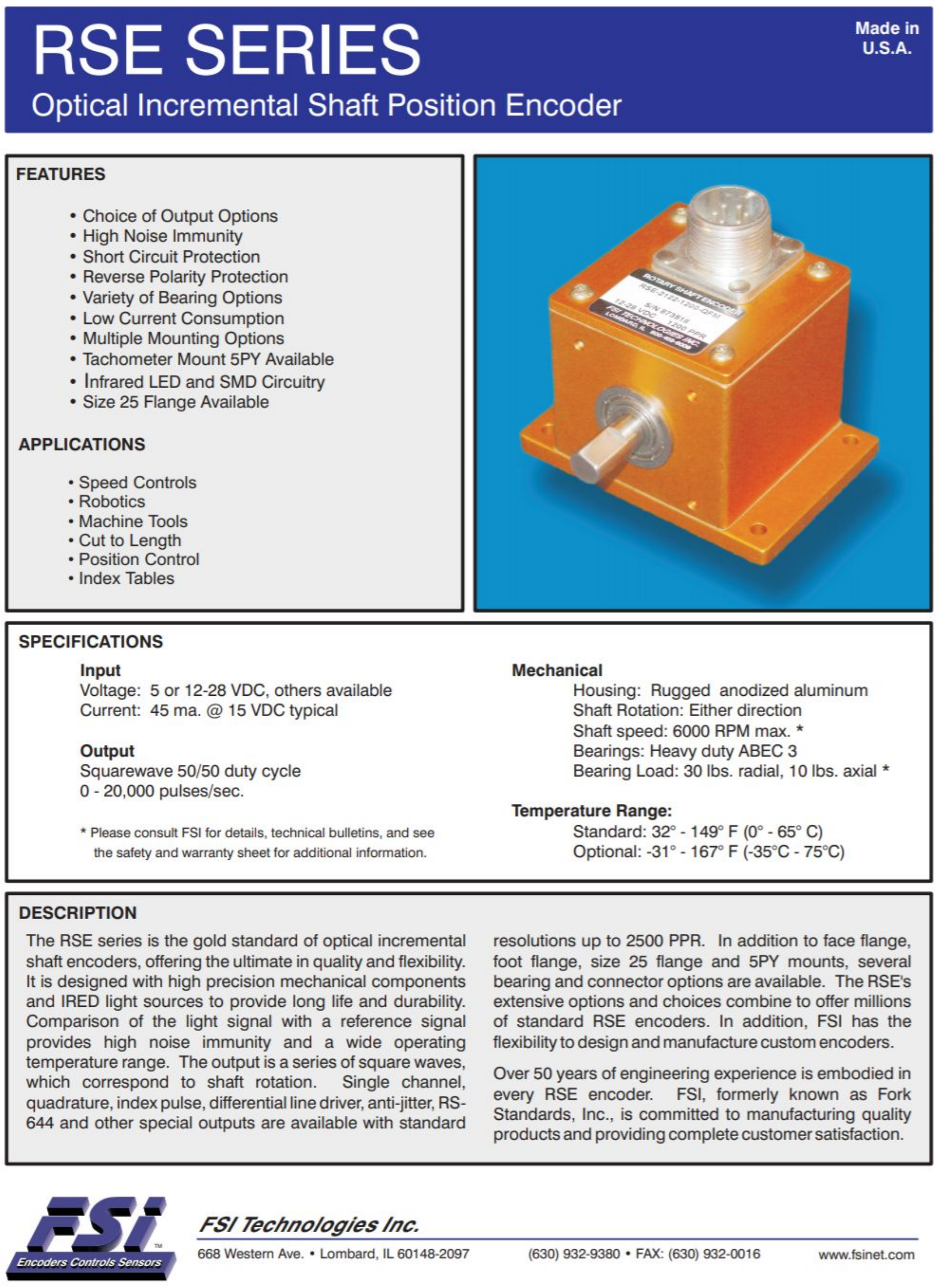
Appendix D: Arduino Uno R3 Schematic



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
ARDUINO is a registered trademark.

Use of the ARDUINO name must be compliant with http://www.arduino.cc/en/Main/Policy

Appendix E: Encoder Data Sheets



RSE SERIES

Made in U.S.A.

Optical Incremental Shaft Position Encoder

FEATURES

- Choice of Output Options
- High Noise Immunity
- Short Circuit Protection
- Reverse Polarity Protection
- Variety of Bearing Options
- Low Current Consumption
- Multiple Mounting Options
- Tachometer Mount 5PY Available
- Infrared LED and SMD Circuitry
- Size 25 Flange Available

APPLICATIONS

- Speed Controls
- Robotics
- Machine Tools
- Cut to Length
- Position Control
- Index Tables

SPECIFICATIONS

Input
Voltage: 5 or 12-28 VDC, others available
Current: 45 ma. @ 15 VDC typical

Output
Squarewave 50/50 duty cycle
0 - 20,000 pulses/sec.

* Please consult FSI for details, technical bulletins, and see the safety and warranty sheet for additional information.

Mechanical
Housing: Rugged anodized aluminum
Shaft Rotation: Either direction
Shaft speed: 6000 RPM max. *
Bearings: Heavy duty ABEC 3
Bearing Load: 30 lbs. radial, 10 lbs. axial *

Temperature Range:
Standard: 32° - 149° F (0° - 65° C)
Optional: -31° - 167° F (-35°C - 75°C)

DESCRIPTION

The RSE series is the gold standard of optical incremental shaft encoders, offering the ultimate in quality and flexibility. It is designed with high precision mechanical components and IRED light sources to provide long life and durability. Comparison of the light signal with a reference signal provides high noise immunity and a wide operating temperature range. The output is a series of square waves, which correspond to shaft rotation. Single channel, quadrature, index pulse, differential line driver, anti-jitter, RS-644 and other special outputs are available with standard resolutions up to 2500 PPR. In addition to face flange, foot flange, size 25 flange and 5PY mounts, several bearing and connector options are available. The RSE's extensive options and choices combine to offer millions of standard RSE encoders. In addition, FSI has the flexibility to design and manufacture custom encoders.

Over 50 years of engineering experience is embodied in every RSE encoder. FSI, formerly known as Fork Standards, Inc., is committed to manufacturing quality products and providing complete customer satisfaction.

FSI™ Encoders Controls Sensors

FSI Technologies Inc.
668 Western Ave. • Lombard, IL 60148-2097   (630) 932-9380 • FAX: (630) 932-0016   www.fsinet.com

STANDARD LINE COUNTS (PPR)

1	2	3	4	5	6	8	10	12	15	16	18	20	22	25	28	30	32
36	38	39	40	48	50	60	64	66	72	80	92	94	96	98	99	100	102
120	125	127	130	144	150	156	170	180	186	200	216	222	228	240	250	254	256
260	300	314	336	360	375	400	500	508	512	600	625	720	745	750	762	800	900
927	1000	1024	1200	1250	1270	1400	1500	1800	2000	2048	2250	2400	2500				

RSE SERIES MECHANICAL

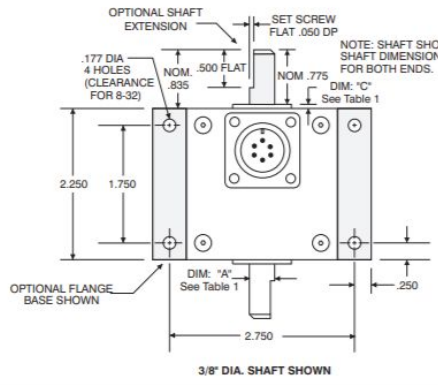
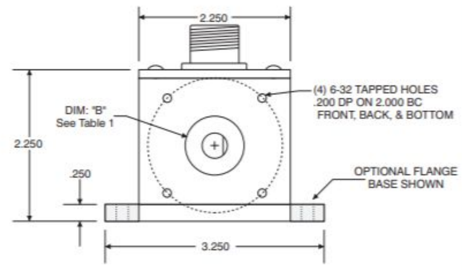


TABLE 1

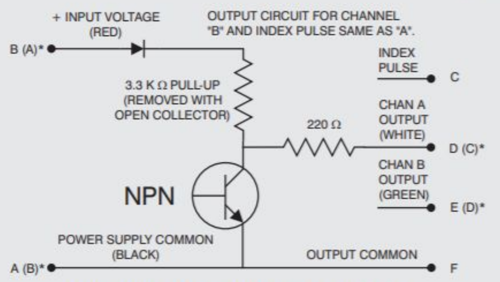
SHAFT & BEARING DIM.		
DIM.	1/4"	3/8"
A	.2495	.3745
B	0.690	0.969
C	0.042	0.062


2.250
(4) 6-32 TAPPED HOLES .200 DP ON 2.000 BC FRONT, BACK, & BOTTOM
DIM: "B" See Table 1
2.250
.250
OPTIONAL FLANGE BASE SHOWN
3.250

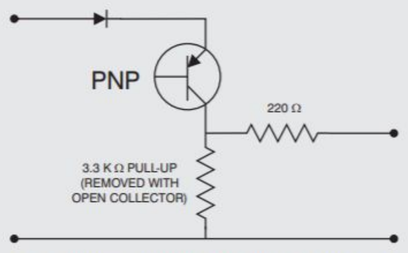
Note: Example encoder with C-1 connector is shown

WIRING INFORMATION

NOTE: Standard pin out shown - Additional output types are available.


+ INPUT VOLTAGE (RED)
OUTPUT CIRCUIT FOR CHANNEL "B" AND INDEX PULSE SAME AS "A".
B (A)*
INDEX PULSE
C
3.3 K Ω PULL-UP (REMOVED WITH OPEN COLLECTOR)
220 Ω
CHAN A OUTPUT (WHITE)
D (C)*
CHAN B OUTPUT (GREEN)
E (D)*
NPN
POWER SUPPLY COMMON (BLACK)
OUTPUT COMMON
A (B)*
F

*** NOTES: (1) (X) INDICATES PIN OUT FOR OPTIONS 8 & 9.**
(2) CHANNEL "B" QUAD & INDEX PULSE ARE OPTIONS - SEE ORDERING GUIDE.



NOTE: "PNP" INPUT AND OUTPUT DESIGNATION SAME AS "NPN"

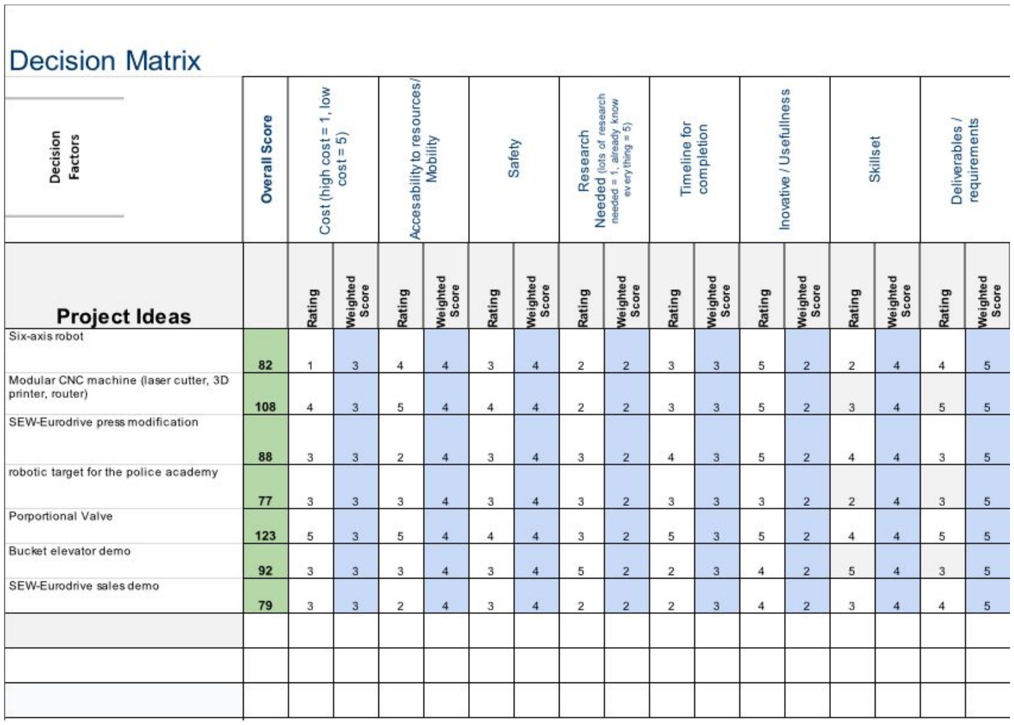
OVER 700 DISTRIBUTORS WITHIN NORTH AMERICA TO SERVE YOU

FSI™ Encoders Controls Sensors

FSI Technologies Inc.

668 Western Ave. • Lombard, IL 60148-2097 (630) 932-9380 • FAX: (630) 932-0016 www.fsinet.com

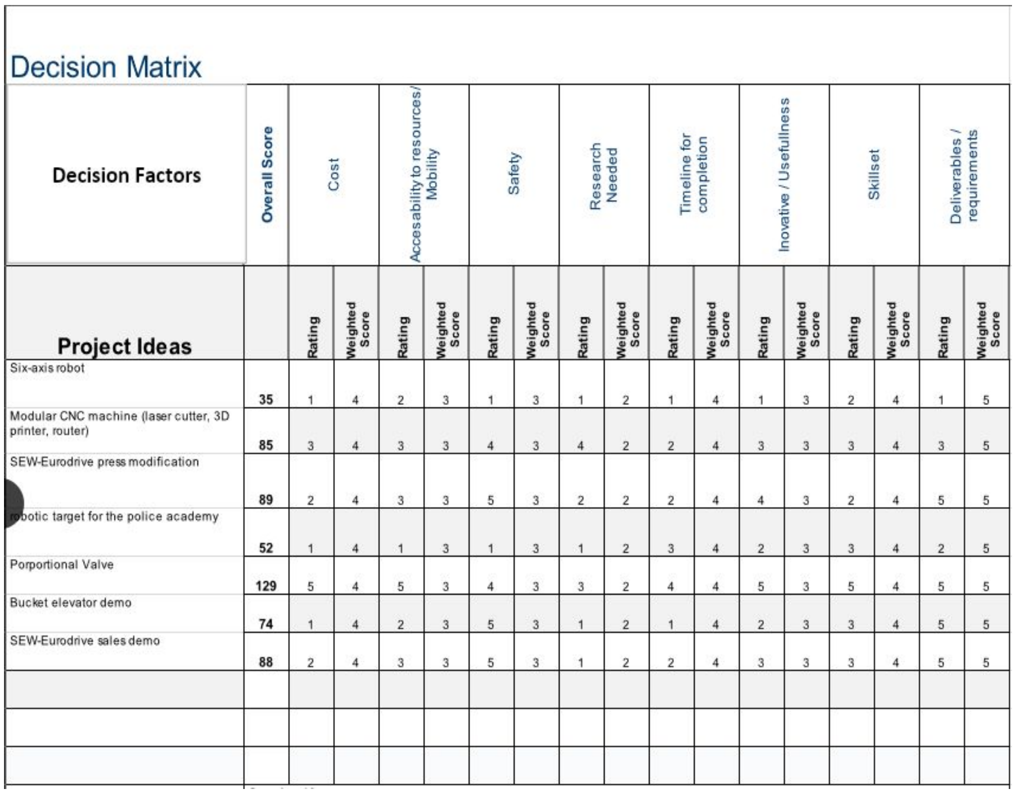
Appendix F: Individual Group Member Decision Matrices



Decision Matrix

Decision Factors	Overall Score	Cost (high cost = 1, low cost = 5)		Accesability to resources/ Mobility		Safety		Research Needed (lots of research needed = 1, already know ev ery thing = 5)		Timeline for completion		Inovative / Usefullness		Skillset		Deliverables / requirements	
Project Ideas		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Six-axis robot	82	1	3	4	4	3	4	2	2	3	3	5	2	2	4	4	5
Modular CNC machine (laser cutter, 3D printer, router)	108	4	3	5	4	4	4	2	2	3	3	5	2	3	4	5	5
SEW-Eurodrive press modification	88	3	3	2	4	3	4	3	2	4	3	5	2	4	4	3	5
robotic target for the police academy	77	3	3	3	4	3	4	3	2	3	3	3	2	2	4	3	5
Porportional Valve	123	5	3	5	4	4	4	3	2	5	3	5	2	4	4	5	5
Bucket elevator demo	92	3	3	3	4	3	4	5	2	2	3	4	2	5	4	3	5
SEW-Eurodrive sales demo	79	3	3	2	4	3	4	2	2	2	3	4	2	3	4	4	5

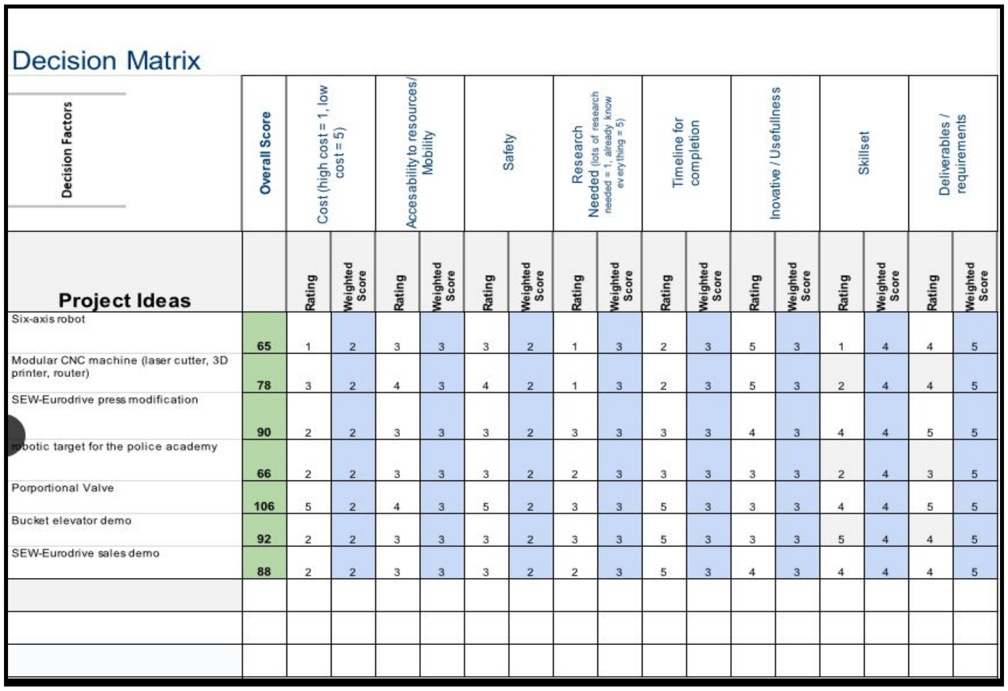
Audrey Fields



Decision Matrix

Decision Factors	Overall Score	Cost		Accesability to resources/ Mobility		Safety		Research Needed		Timeline for completion		Inovative / Usefullness		Skillset		Deliverables / requirements	
Project Ideas		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Six-axis robot	**35**	1	4	2	3	1	3	1	2	1	4	1	3	2	4	1	5
Modular CNC machine (laser cutter, 3D printer, router)	**85**	3	4	3	3	4	3	4	2	2	4	3	3	3	4	3	5
SEW-Eurodrive press modification	**89**	2	4	3	3	5	3	2	2	2	4	4	3	2	4	5	5
robotic target for the police academy	**52**	1	4	1	3	1	3	1	2	3	4	2	3	3	4	2	5
Porportional Valve	**129**	5	4	5	3	4	3	3	2	4	4	5	3	5	4	5	5
Bucket elevator demo	**74**	1	4	2	3	5	3	1	2	1	4	2	3	3	4	5	5
SEW-Eurodrive sales demo	**88**	2	4	3	3	5	3	1	2	2	4	3	3	3	4	5	5

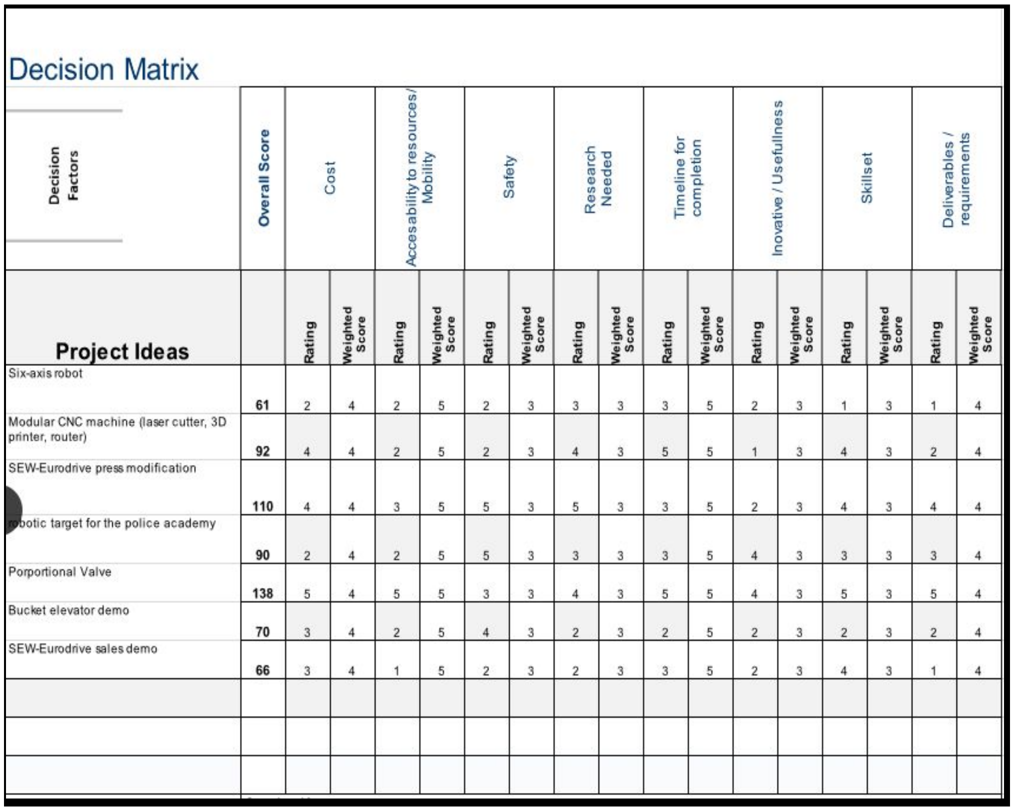
David Poeppelman



Decision Matrix

Decision Factors	Overall Score	Cost (high cost = 1, low cost = 5)		Accesability to resources/ Mobility		Safety		Research Needed (lots of research needed = 1, already know everything = 5)		Timeline for completion		Inovative / Usefullness		Skillset		Deliverables / requirements	
Project Ideas		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Six-axis robot	**65**	1	2	3	3	3	2	1	3	2	3	5	3	1	4	4	5
Modular CNC machine (laser cutter, 3D printer, router)	**78**	3	2	4	3	4	2	1	3	2	3	5	3	2	4	4	5
SEW-Eurodrive press modification	**90**	2	2	3	3	3	2	3	3	3	3	4	3	4	4	5	5
robotic target for the police academy	**66**	2	2	3	3	3	2	2	3	3	3	3	3	2	4	3	5
Porportional Valve	**106**	5	2	4	3	5	2	3	3	5	3	3	3	4	4	5	5
Bucket elevator demo	**92**	2	2	3	3	3	2	3	3	5	3	3	3	5	4	4	5
SEW-Eurodrive sales demo	**88**	2	2	3	3	3	2	2	3	5	3	4	3	4	4	4	5

James Clements



Decision Matrix

Decision Factors	Overall Score	Cost		Accesability to resources/ Mobility		Safety		Research Needed		Timeline for completion		Inovative / Usefullness		Skillset		Deliverables / requirements	
Project Ideas		**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**	**Rating**	**Weighted Score**
Six-axis robot	**61**	2	4	2	5	2	3	3	3	3	5	2	3	1	3	1	4
Modular CNC machine (laser cutter, 3D printer, router)	**92**	4	4	2	5	2	3	4	3	5	5	1	3	4	3	2	4
SEW-Eurodrive press modification	**110**	4	4	3	5	5	3	5	3	3	5	2	3	4	3	4	4
obotic target for the police academy	**90**	2	4	2	5	5	3	3	3	3	5	4	3	3	3	3	4
Porportional Valve	**138**	5	4	5	5	3	3	4	3	5	5	4	3	5	3	5	4
Bucket elevator demo	**70**	3	4	2	5	4	3	2	3	2	5	2	3	2	3	2	4
SEW-Eurodrive sales demo	**66**	3	4	1	5	2	3	2	3	3	5	2	3	4	3	1	4

Luke House

Appendix G: Additional Figures

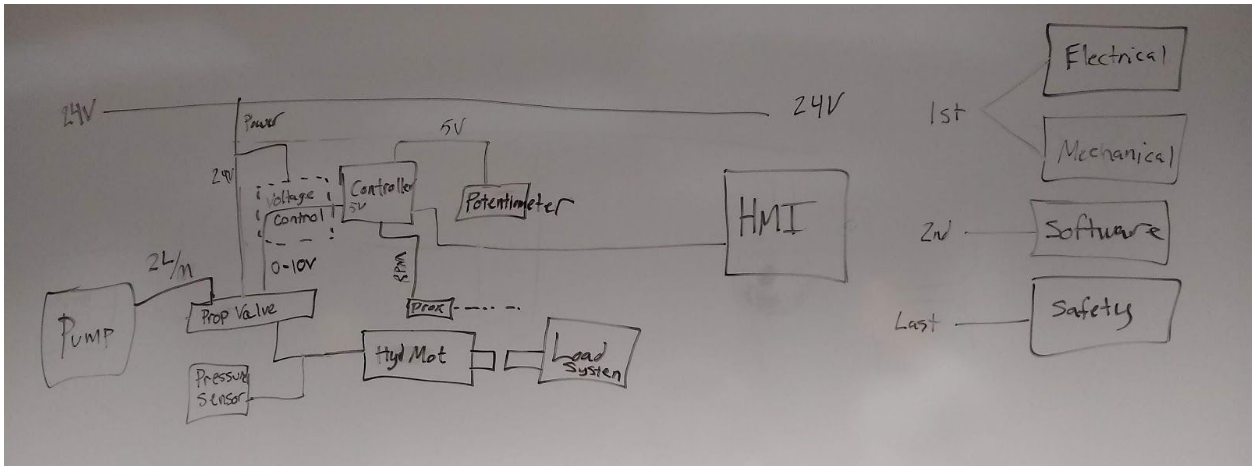


Figure 26: Original System Build Sketch

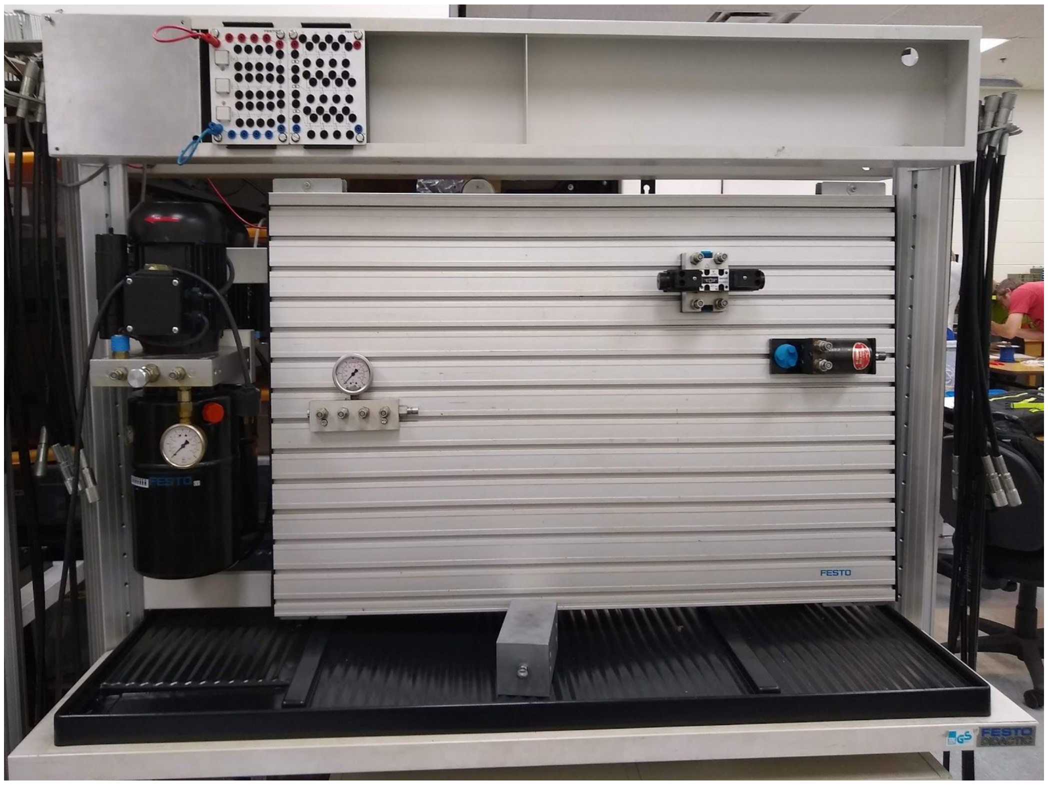


Figure 27: Festo Hydraulic Trainer

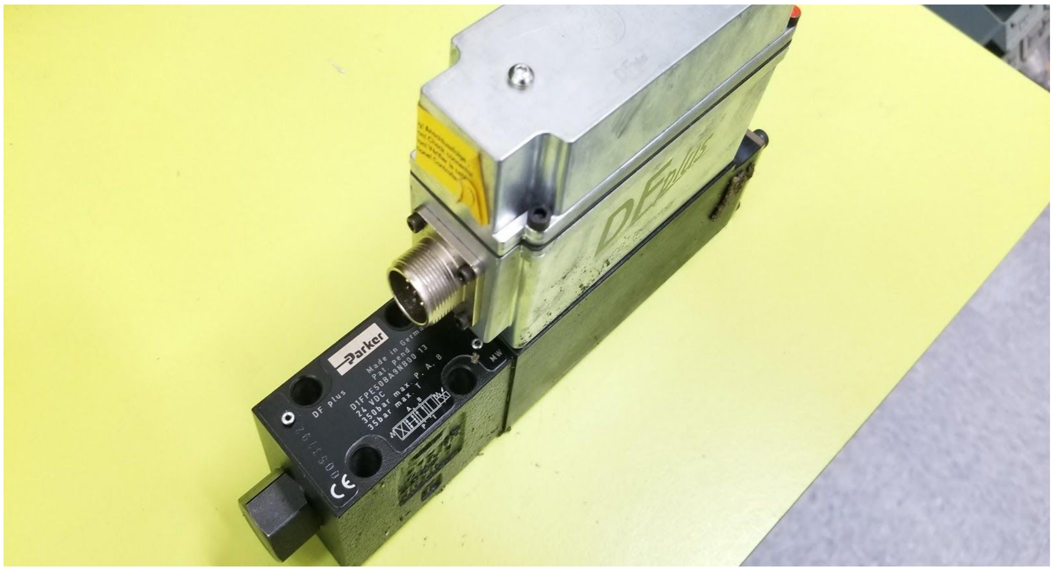


Figure 28: Parker Proportional Valve

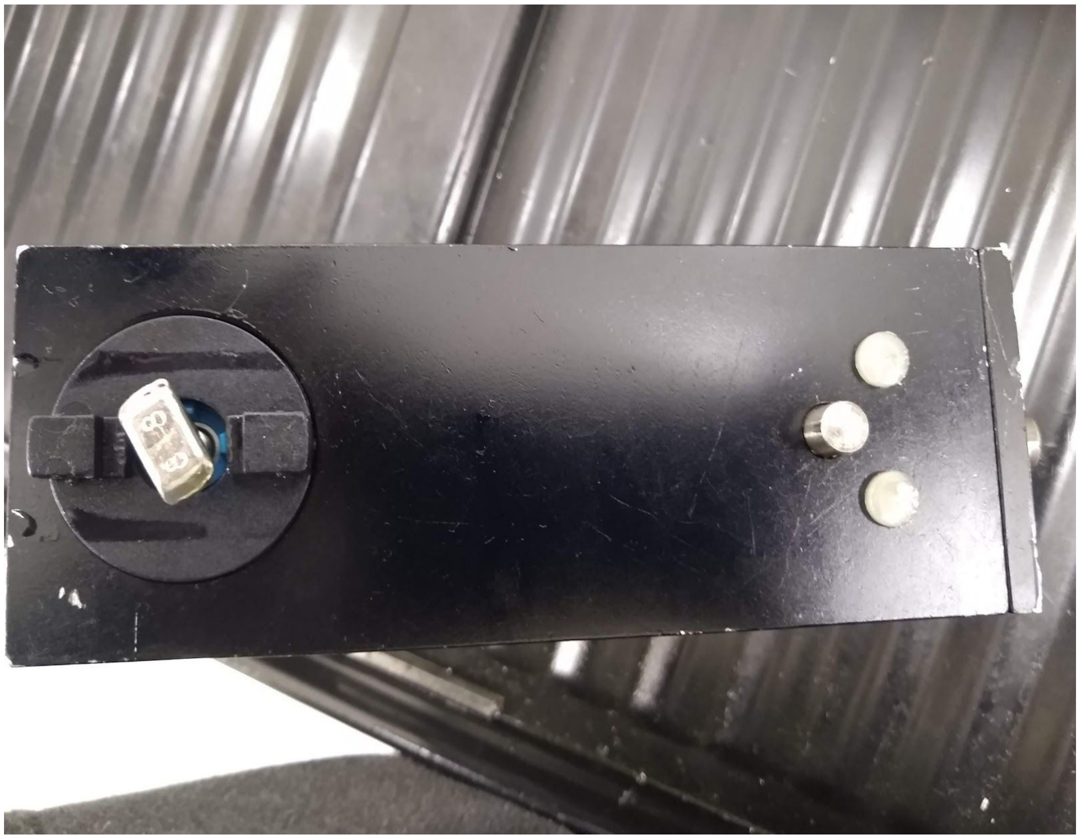


Figure 29: Mounting System for Festo Hydraulic Trainer