Fall 2018

Developing controls software for hydraulic motor test stand

Alex Shaw
Iowa State University

Developing controls software for hydraulic motor test stand

By

Alex J. Shaw

A technical paper submitted to the graduate faculty

In partial fulfillment of the requirement for the degree of

MASTERS OF SCIENCE

Major: Industrial and Agricultural Technology

Program of Study Committee:

Matthew Darr, Major Professor

Steven Hoff

Robert McNaull

Iowa State University

Ames, IA

2018

Contents

Abstract

Two hydraulic motors were tested to gather data for the development of a prognostics system to detect the early stages of bearing failures. A controls system is required to operate the test motors at the desired speed, pressure, and displacement. Along with controlling the test motors, an electric drive motor, the pump stack, and cooling fans needed to be controlled and monitored.

Introduction

Agricultural machines occasionally experience failures of the hydrostatic wheel motors. Other off road machines that rely on hydrostatic drive systems experience similar failures. In the event of a failure, typically the inner most bearing on the motor output shaft is the failing component. The operator is not aware of the problem until the failure is catastrophic. Debris from the failed bearing quickly distributes throughout the entire hydraulic system. This causes extensive and costly repairs as well as significant downtime for the machine. The goal of this research project is to create a prognostic system to detect early stages of bearing failure and warn the operator before catastrophic failure occurs. The work covered in this document is specific to the test stand used to test these research hypotheses.

To provide system data for the development of the prognostics tool, a test stand was designed and built to gather detailed data for hydraulic wheel motors under variable load conditions. For each run cycle, two test sample motors will be loaded at high drive pressure and will be connected on either end of a single shaft. By pressuring the same ports on the opposing motors, the system will generate high torque at zero rotational speed. A third motor will be used to drive the two opposing test motors at the desired rotational speed. This regenerative system requires relatively low input power since the flow rates to the test motors are essentially zero and the power to drive the test motors at speed will be just high enough to overcome system inefficiencies. Also, in order to accelerate bearing failure a radial load will be placed on both opposing motors to target the failure of the inner bearing within a 300 hour window. Data will be collected using a range of temperature sensors, acoustics, and accelerometers.

There are two significant electronic systems on this test stand. The first is for extensive data collection and the second is for controlling the test stand itself. My task for this project was to develop and implement the test stand control. This will include controlling displacement on the three hydraulic motors and the hydraulic pump. This control system will be closed loop, so speed sensors, temperature sensors, and pressure sensors will also be used to set the speed of the motors. As a part of the control system, there will be multiple levels of protection built in to ensure that data is always collected during operation and that the test stand will never enter a state where it could damage itself.

Objectives

Control the hydraulic motor stand in a way that would mimic the actual vehicle's hydraulic drive motors. This stand should be easily controlled by the operator and keep all controllable variables within a safe threshold.

Materials and methods

The main hardware that was used for the controls system was a National Instruments Compact RIO 9114. This was chosen due to the availability of the controller. In conjunction with the cRIO a number of cards were used. Two NI 9472(8 channel 0-24v sourcing digital output), NI 9361 (8 channel counter input), NI 9212 (8 channel thermocouple input), and an NI 9205 (32 channel +-10v Analog input) were utilized. As far as sensors that were used, omega K type thermocouples, 100 psi pressure transducers, and 10,000 psi pressure transducers were provided to get feedback on the system. To provide feedback on the motors a Danfoss 149055 speed and temperature sensor was chosen. The factory sensor that comes on these motors configuration only included speed. In order to match the vehicle's hardware a Danfoss S690 hydraulic tandem pump was chosen to provide the hydraulic power to the stand. In a vehicle this pump's displacement was controlled via a cable that went to the cab. In our environment we wanted this to be controlled electronically so the manual control was exchanged out for a set of AT304672 electric pump displacement control valves. To detect bearing deterioration a pair of trident DM4500 wear debris monitors were chosen. The wear monitor can detect ferrous particles from 40 microns to 700 microns and nonferrous particles from 150 microns to 700 microns. Due to a temperature limitation of these debris monitors the oils max temp needs to be under 185F. The case drain from the hydraulic motors can be higher than the upper limit of the debris monitors, so two oil coolers were implemented in between the motors case drain and the sensors. To cool the oil down further a 17 kW hydraulic oil cooler was placed between all of the returning oil and the reservoir. To drive the entire stand a 50hp electric motor was chosen due to its availability after being retired from another test stand. Controlling the 50hp electric motor is a Yaskawa A1000 variable frequency drive. This VFD was already installed at the testing facility and just required minimal setup to get it to work with our motor. In conjunction with the VFD setup there were also three emergency stops and two pressure switches installed on the system. Emergency stops were placed on either side of the stand as well on the computer station. The pressure switches were installed on the low and high side of test motor one and set to trip at 7500 psi. In the event of an emergency, the tripping of one of these emergency stops or the pressure switches would kill power to the electric motor, and in turn depressurize the hydraulic system.

Hydraulic schematic

The system that is being controlled consists of two hydraulic test motors mounted on the same shaft (Figure 2). Pressure is supplied via the rear of a tandem hydraulic pump being powered by an electric motor. Pressure is being supplied on the same port of both motors causing them to deadlock from the opposing motor. The system only needs to supply enough flow to overcome the inefficiencies in the motor and thus require very little power to generate high pressures. The second part of the stand is to supply speed to the test motors. A drive hydraulic motor is mounted in parallel with the two test motors and coupled via a belt. The front of the tandem pump supplies flow to the drive motor. The drive motor overcomes one of the test motors and allows the motors to start spinning at a desired speed and a desired pressure. Figure 1 shows the basis of the test stand with the drive motor on the left top and two opposing test motors on the right top. On the bottom left of the stand is the electric motor suppling power to the tandem pump stack in the bottom center of the figure.

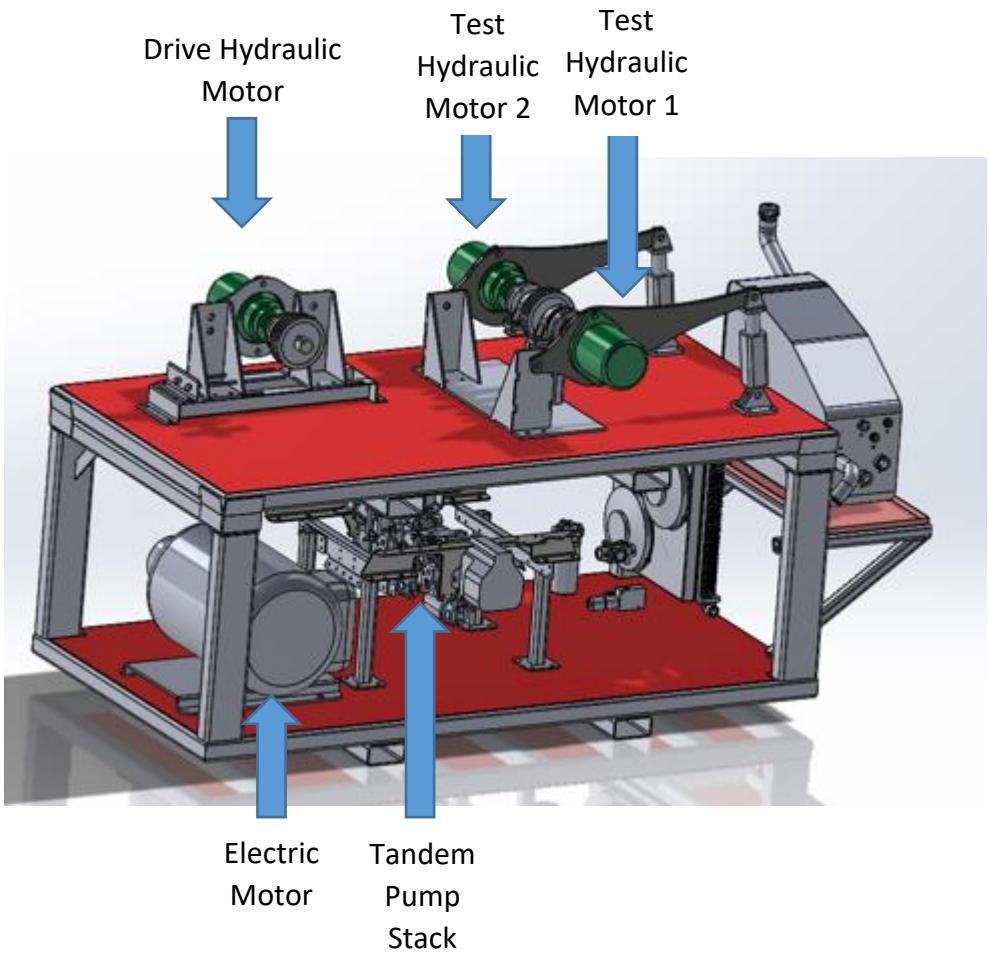


Figure 1 – Test stand

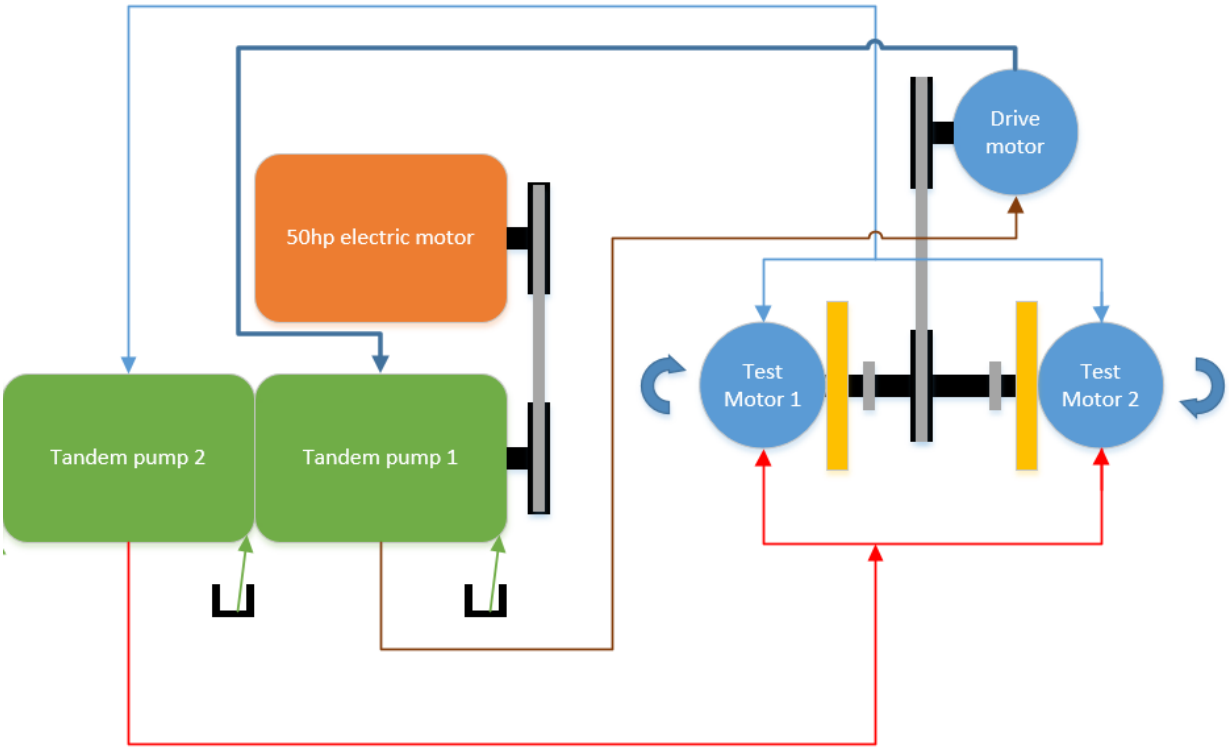


Figure 2 – Simplified hydraulic schematic

Software development

FPGA

The FPGA side of the cRIO is well tailored to very fast processing. With this in mind it was chosen to tackle the job of calculating speed and also generate PWM signals for the fans, tandem pump stack, and test motors. In order to calculate the speed of the motors the cRIO needs to use the input of the 86 tooth tone ring embedded in the test motors as well as time. The FPGA doesn't have easy access to a clock to calculate time; instead it has "ticks". The FPGA that I was working with was running at 40 MHz which equates to 40,000,000 ticks per second. The method I chose to calculate speed was to take a starting tick time, count 86 pulses, and then take a second tick time. The difference of these two tick values was divided by 40,000,000. This yields seconds per revolution. To convert this to revolutions per minute, the inverse of the seconds per revolution was calculated and then multiplied by 60 seconds per minute. This value was then passed back to the real time side of the CRIO to be further used with controls and displaying it to the user.

In order to control the fans, tandem pump displacement control, and motor displacement control pulse width modulation (PWM) was used. PWM allows a user to yield an analog like signal from a purely digital output (figure 3). In order to yield an analog like signal, a frequency is picked and a duty cycle is selected. The change of the signal's duty cycle results in a higher or lower "analog" output. Due to the high rate of change and required accuracy the FPGA was used to generate the PWM signal. As previously discussed the FPGA does not have an easily accessible clock and

instead relies on ticks. Due to the limited size of the FPGA the duty cycle to ticks calculation is done on the real time side and then passed to the FPGA. The PWM generation loop reads in low ticks and high ticks. It then sets the value to LOW and waits until the low tics value has elapsed. After the low tick's time has been satisfied it sets the output to HIGH and waits until the high side ticks value has elapsed. This same process is carried out for test motor 1, test motor 2, auxiliary motor, tandem pump 1, tandem pump 2, tank fan, test motor 1 case drain fan, and test motor 2 case drain fan.

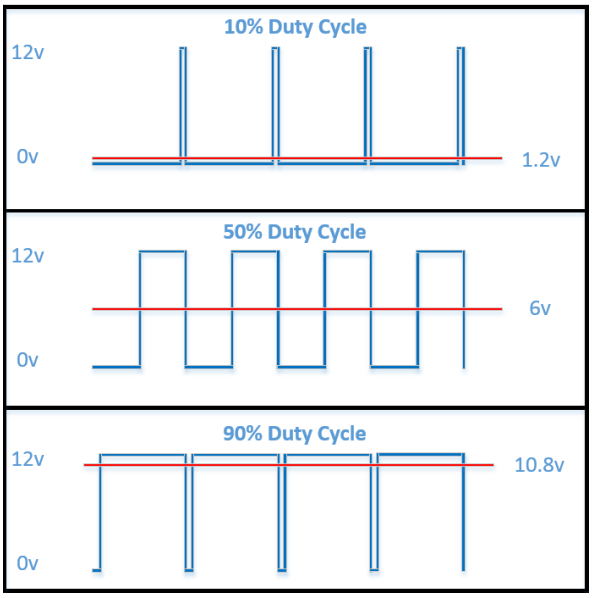


Figure 3 – PWM example

Real time

The real time software is comprised of 4 core loops that run in parallel (figure 4). The first of these loops is titled main. In this loop all of the sensors are read in, the state machine is housed here, as well as the duty cycle set points for PWM generation. When a duty cycle is set the number of ticks on and off are calculated for the given frequency and then passed along to the FPGA to carry out the PWM generation. As an example the tandem pump runs at a frequency of 200 Hz. If the desired duty cycle was 20% high ticks would be (40,000,000 ticks per second / 200 Hz) * (20/100) =**40,000 ticks** and low ticks would be (40,000,000 ticks per second / 200 Hz) * (1-(20/100)) = **160,000 ticks.** These two values then get sent to the FPGA where it carries out the desired signal.

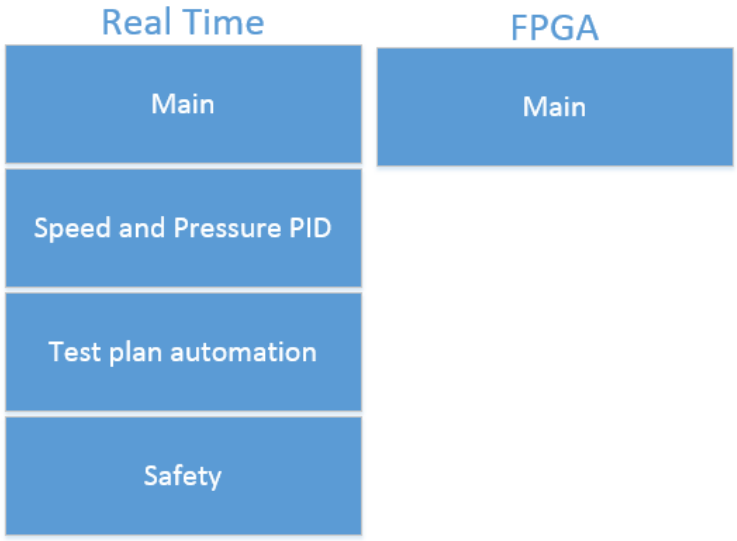


Figure 4 – Block Diagram of Loops

PID loop

To accurately control the speed of the dyno as well as the pressure, two Proportional-Integral-Derivative (PID) controls were used. The National Instrument webpage says "Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control."(NI). The PID controller takes in a set point that the user would like the system to reach as well as a measured variable of the current value of the system. For the speed control PID the auxiliary motor RPM was the measured variable. The error between the set point and the measured variable is calculated and then passed into the proportional, integral, and derivative portions of the control. The proportional control is the ratio of error to its proportional gain. This portion can control the rate of response of the system, but will also increase overshoot past the set point. The integral portion relates to error over time. As the error persists over time the I effort increases driving it closer to the set point faster. The integral term can also increase the rate of response of the system. The derivative deals with the rate of change of the error. An increase in the derivative gain can remove oscillation around the set point in the system.

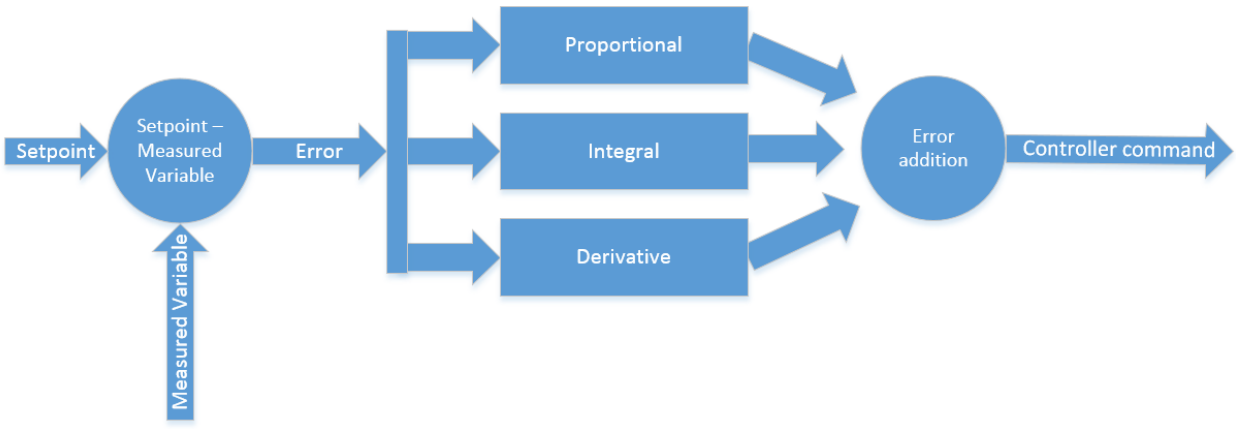


Figure 5 – PID Controller

State machine

At the heart of the controls software lies a state machine. This state machine has 11 states. Starting up the software for the first time the default state is off. The operator enters in a desired speed and pressure, then selects the start button. This action moves the state machine to the "start electric motor" state. A signal is sent to the variable frequency drive to start the electric motor to drive the tandem pump stack. After 15 seconds the motor has had enough time to fully spool up and now the state machine enters the "bring up to speed" state. The goal of the "bring up to speed" state is to slowly bring the test motors to the desired testing speed. In this state all three coolers are activated and the speed PID controller is enabled.

Once the motors are within 5 RPM of the set point the state machine moves on to "bringing up to pressure". This state will set the displacement of the motors to the desired value and then enable the pressure PID controller. Once the actual pressure is within 20 psi of the desired pressure, the state machine moves onto the "running" state. The "running" state keeps the speed and pressure PID controllers enabled and allows the system to run continuously. Once the test is finished and the operator hits the stop button, the state machine progresses to "destroke pressure". Tandem pump 2, test motor 1 displacement, and test motor 2 displacement is set to 0 and the state machine goes to "destroke pump 1". Pump 1 is what drives the auxiliary motor to the desired speed. In this state the pumps displacement will get dropped down slowly until the auxiliary motor stops.

Once the motors have stopped spinning then the "stop electric motor" state can be entered. Very similar to the "start electric motor" state, the enabling signal is pulled to low which results in disabling the VFD. This ramps down the electric motor supplying power to the tandem pumps. After 15 seconds the motor has fully stopped and can move on to "cool down rest". Due to the temperature sensitivity of the particle counting sensors that the system uses, heat creep can be an issue (figure 7). In order to tackle this issue the fans are kept on for a number of minutes after the system is off. This allows the coolers to pull off the excess heat in the oil and prevent damage to the debris monitors. After the desired time then the system returns to the "off" state and is ready to start a new test run.

At any time in any state if the E-stop is pressed everything turns off. This includes the tandem pumps, fans, test motors, auxiliary motor, and the electric motor. The only way to exit this state is to reset the system and then it returns to the default state of "off". Lastly there is a "test" state that is used strictly for debugging. This state will allow you to interact with any part of the system that you want. This can allow the user to quickly troubleshoot if there is something that they can't identify with the system running normally.

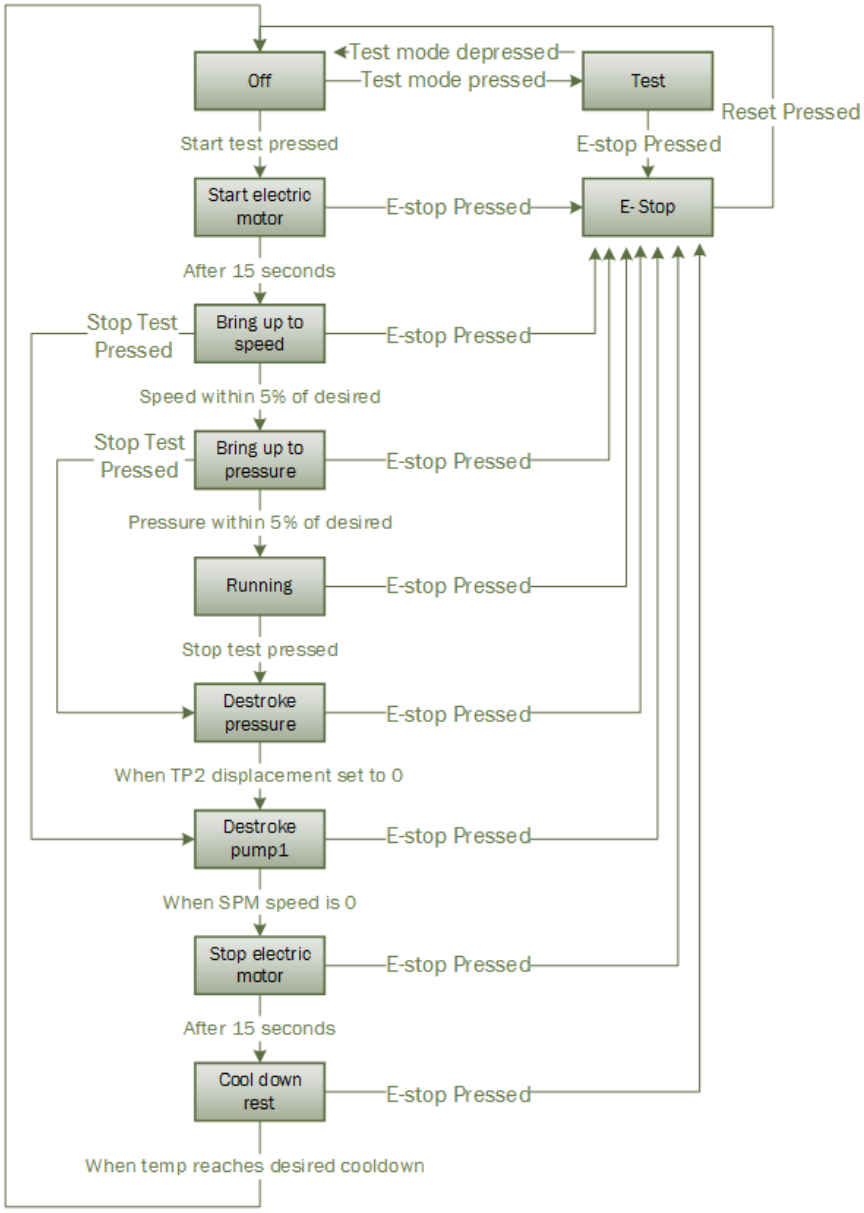


Figure 6 – State Machine

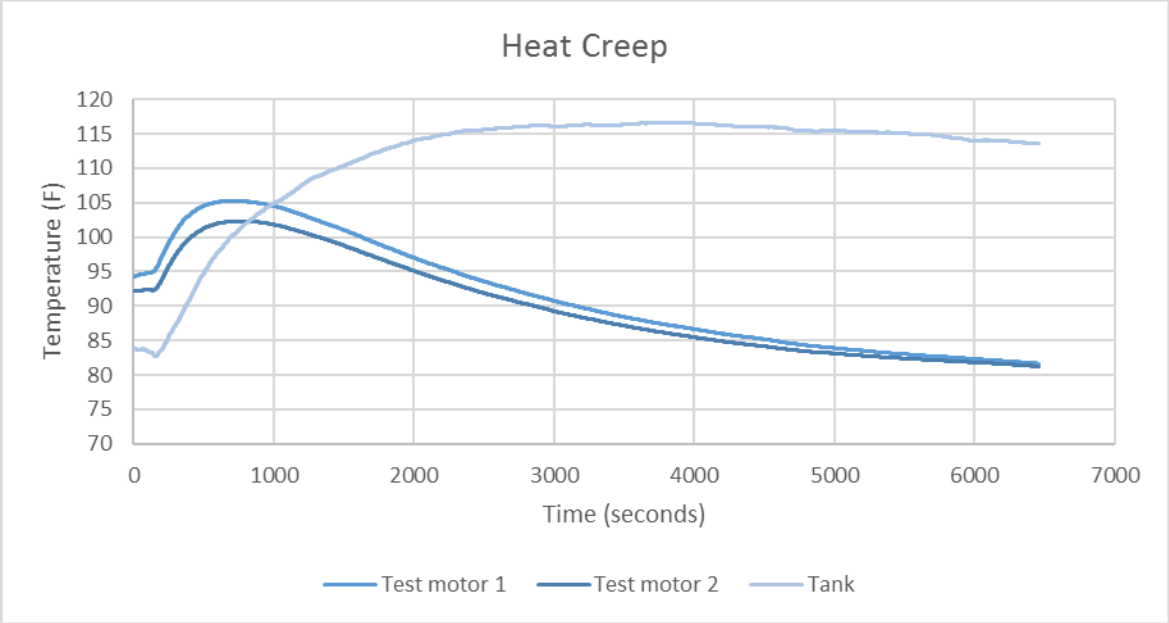


Figure 7 – Heat creep

Safety loop

The safety loop is arguably the most crucial part of the entire program. This loop runs in parallel with the main loop and monitors all critical signals. The safety modes that are checked are pressure, speed, and temperature. While the hydraulic lines are rated for 6000 psi constant and will withstand peaks of up to 8000psi, the software is programmed to trigger the E-stop at 6000 psi. There is also a backup hard wired pressure switch set at 7500psi on the test motors. If this is tripped it will trigger the E-stop to the VFD and shutdown the electric motor. Next on the safety spectrum is speed safety. The system is constantly checking the difference between the auxiliary motor and the two test motors. If their speeds differ by more than 10% it is an indicator that either something has failed in the drivetrain or one of the speed sensors is inoperable. With either option, the E-stop is triggered, shutting down the system. The last main concern in the realm of safety is temperature. If the temperature of test motor 1 case drain or test motor 2 case drain goes over 150 degrees Fahrenheit the E-stop is triggered to protect the debris monitors.

Test plan automation

After a few tests have been run on the current software it became apparent that there needed to be more automation. This system reads in a comma delimited text file that has the desired speed, pressure, displacement, and time step. This system also signals the data acquisition system when to start recording. This will allow for a variety of different pressure and speed steps in a short amount of time while also reducing the opportunities for a user to accidently type in the wrong number. Due to this being programmed on the real time side of the cRIO it becomes difficult for a user to select a test plan csv to run off the stand computer. In order to avoid the unnecessary complexity of using a network stream, I instead mapped a file transfer protocol (FTP)

folder on the cRIO to the test stand computer. The user can upload their test plan to the cRIO using this mapped drive. Once uploaded this file is read into the LabVIEW user interface running on the real time side. This file gets read into a two dimensional array. When the user is ready to start the test procedure they can select “Auto Start”. From there the system will wait the desired amount of milliseconds, and then update the speed, pressure, and displacement set points. After giving the system time for all of the settings to settle, it then sends a command to the data acquisition system to start recording. Once the specified time has elapsed for that setting, it then moves on to the next setting just like before. When the final entry is completed, a signal is sent to initiate the dyno to shut down.

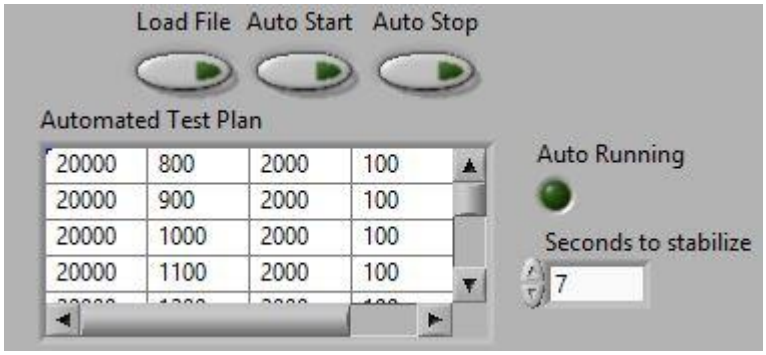
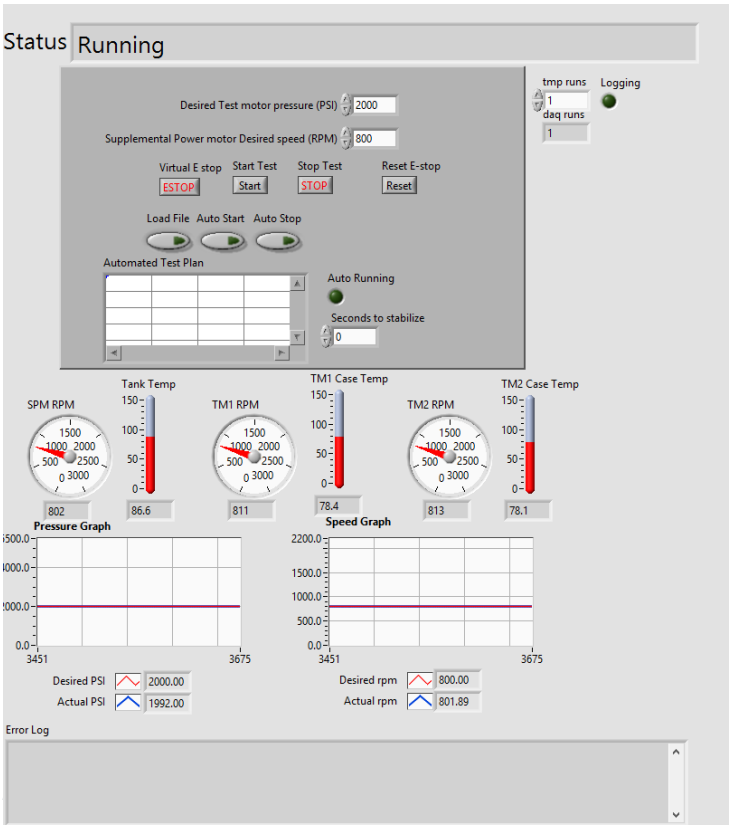


Figure 8 – Auto Run GUI

Graphical User Interface

Getting the dyno running is only half the battle. A simple user friendly graphical interface is needed to interact with the dyno. This part was redesigned multiple times to optimize its ease of use. This final interface includes a number of user friendly features. Displayed at the top is the status of the state machine in the software. There are two user input boxes where the desired pressure and speed can be selected. To get the system running a user only has to enter the desired pressure and speed and click the start test button. While the system is running the user can update the desired pressure and speed and the system will immediately change to those settings. If the user would like to run an automated test plan they simply need to upload a comma delimited file to the cRIO, select “Load File”, and then “Auto Start”. While this is running the auto running light with be illuminated and the system will step through the settings in the test plan. When the test plan finished it will automatically stop the dyno. Select system information is displayed below the user controls. The speed in revolutions per minute is displayed for all three motors. The temperatures of the tank, test motor one case drain, and test motor two case drain are displayed in Fahrenheit. The two graphs show the desired pressure and speed in relation to their process variables. This will allow from a quick glance to see if the system is operating the way the user expects it. Finally, the bottom of the user interface houses an error log. This displays any errors that have recently triggered an E-stop from the software.



PID Speed results

In order to quantify how well the speed PID is performing data first needed to be collected on a machine to see the variability of speed when the operator selects a set speed they want to run. The base line data was collected on an agricultural vehicle and a speed run of 10 MPH was selected. This vehicle was equipped with 72 inch diameter tires and the planetary gear box with a combined RPM of the tire to MPH ratio of 124.6:1. Using this ratio the RPM of the motor was back calculated from the speed in MPH. It should be noted that the precision of this measurement is .12465 RPM due to the precision of the speed output from the vehicle. This allows for a base line to compare the stand with. The data off the vehicle yielded an average speed of 10.0 MPH with an average of 1246.5 RPM at the wheel motor (figure 12). Standard deviation of the test motor was 6.7 RPM. The same test was duplicated on the test stand. With the 124.6:1 RPM to miles per hour conversion in mind, the test stand was set to 1247 RPM. This will yield a machine speed equivalent of 10 miles per hour. The stand's average RPM was 1247.3 and the standard deviation was 1.6 RPM. The graph below shows the vehicle's wheel motor RPM compared to the dyno's RPM. The dyno successfully controls the speed of the motor within the actual operational speed range of the vehicle.

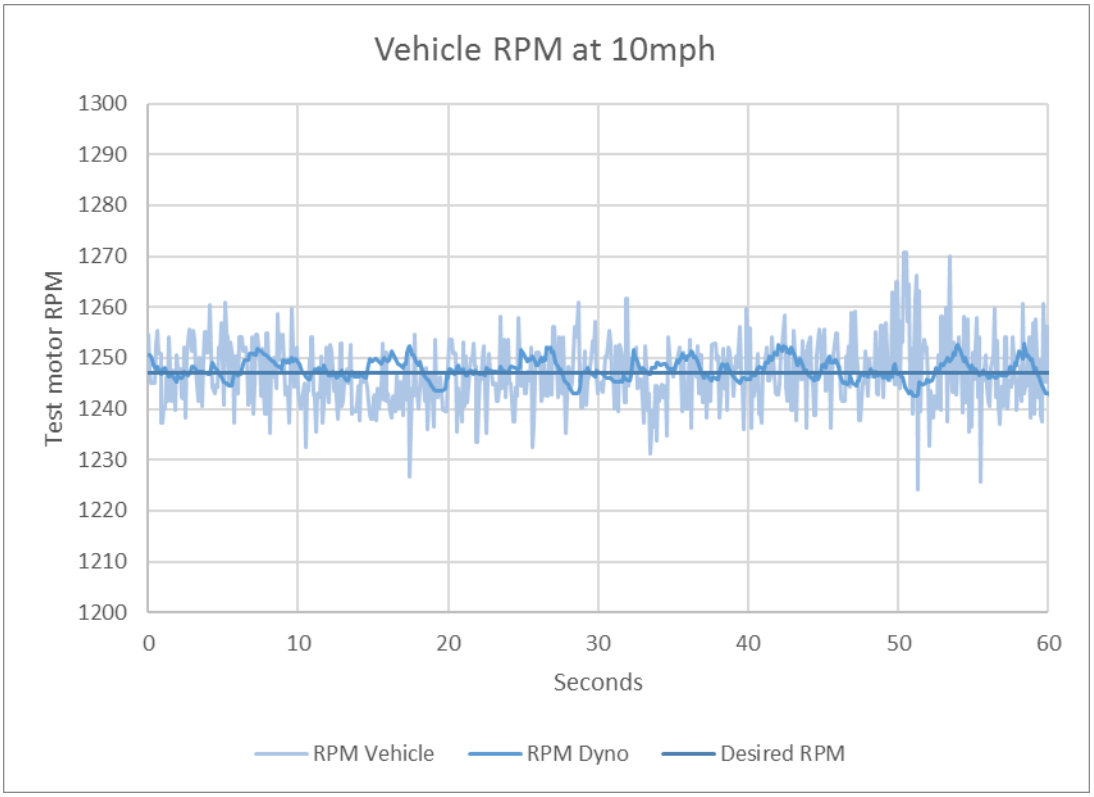


Figure 10 – Vehicle vs dyno speed

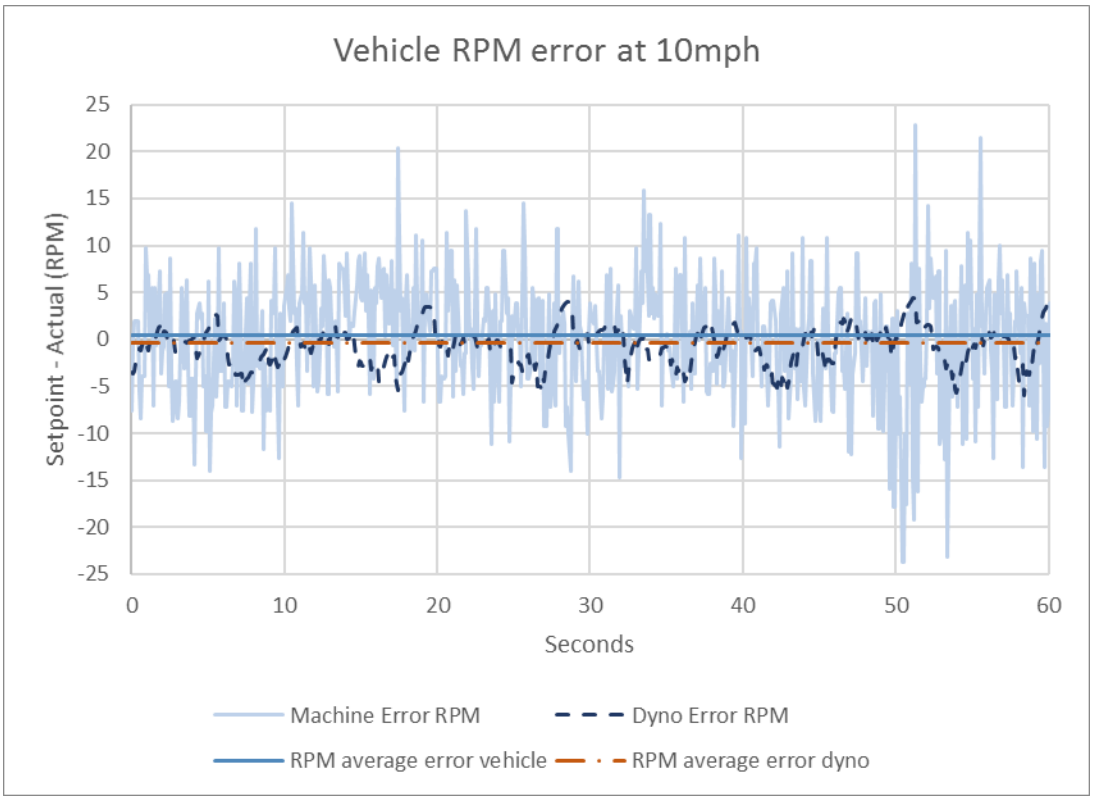


Figure 11 – Speed RPM error

RPM statistics table	
Vehicle mean	1246.5
Vehicle stdev	6.7
Vehicle Coefficient of Variation	0.005
Dyno mean	1247.3
Dyno stdev	1.6
Dyno Coefficient of Variation	0.001

Figure 12- RPM statistics table

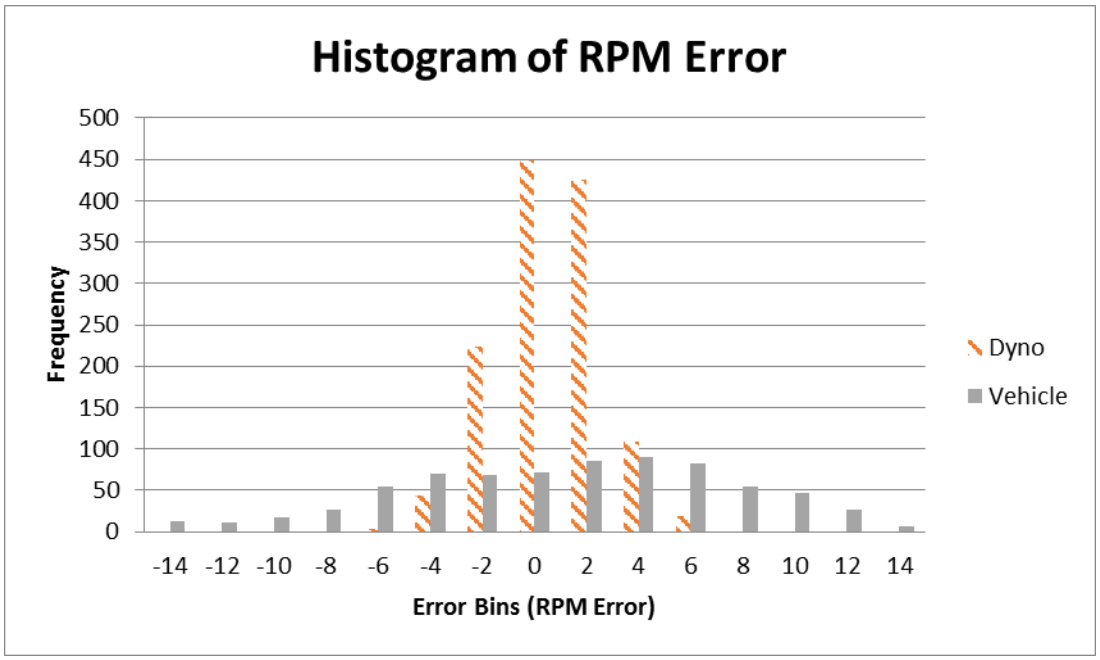


Figure 13 – Histogram of RPM error dyno vs vehicle

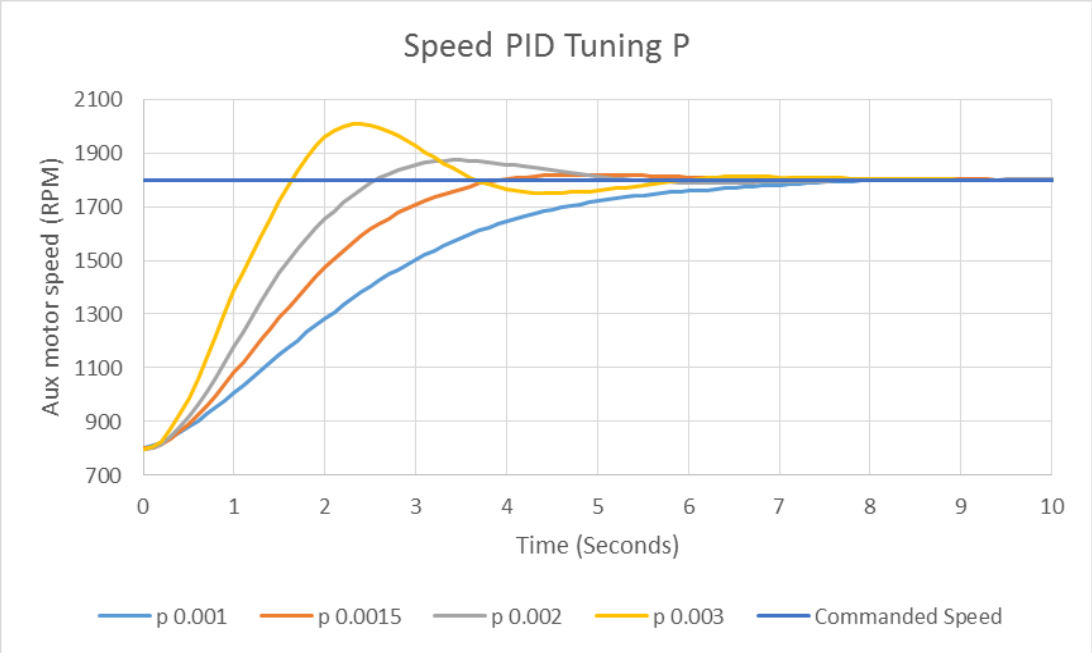


Figure 14 – Speed Tuning P gain

Dialing in the PID values for the speed control was an iterative process. The P gain needed to be high enough to minimize transition time yet not too high to cause over run and chasing of the set point. Figure 14 shows a few of the preliminary tested PID values that were used. The value of 0.0015 was chosen due to its limited over run, fast settling time, and desirable response time. The final controller uses P: 0.0015, I: 0.0010, D: 0.0005.

The speed PID's accuracy incorporates how well it holds a speed as well as how fast and accurately it can move between speeds. Figure 15 represents a test conducted on the stand that started at 800 RPM and every 30 seconds step up 200 RPM until 1800 RPM was reached. Then it stepped down 200 RPM at a time until the speed was back to 800 RPM. This test shows the ability of the speed PID controller to change speeds quickly while minimizing overrun. On average in this test the system can adjust and stabilize to a new speed setting in 4.4 seconds. Figure 16 represents the error from the desired set point subtracted by the actual RPM of the test motor. This data came from the speed step data and filtered out the speed transitioning periods. This graph shows that while the speed error range is +/– 5 RPM, the average error is only 0.3 RPM. This means that on average the test motor is 0.3 RPM over the commanded speed. This could be used in the future to define an offset to more accurately control the motor.

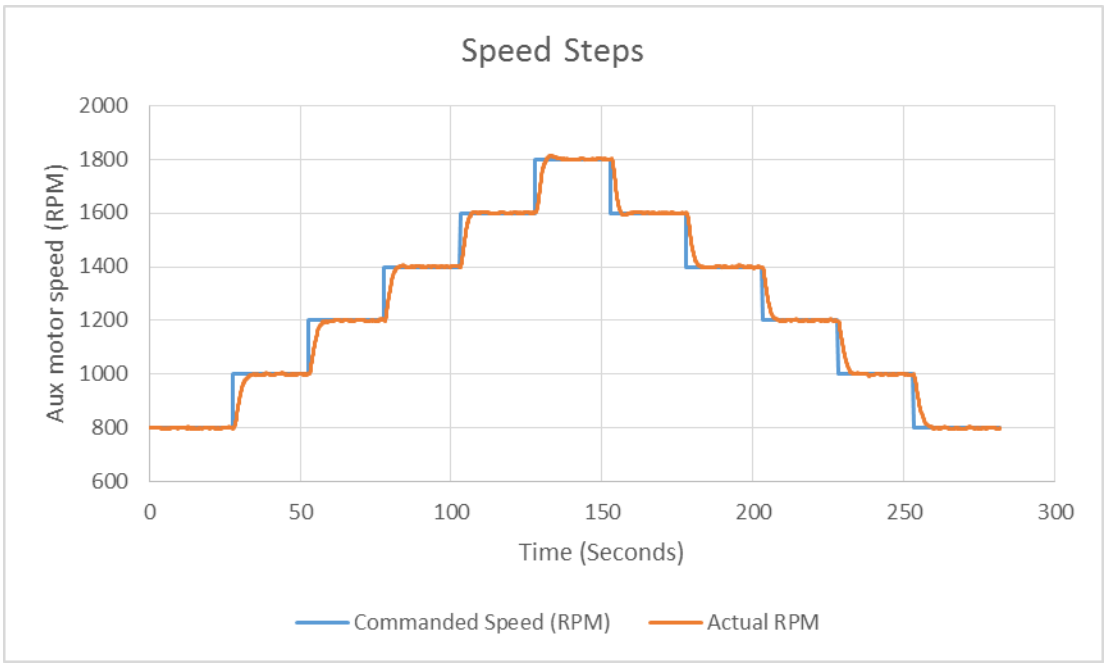
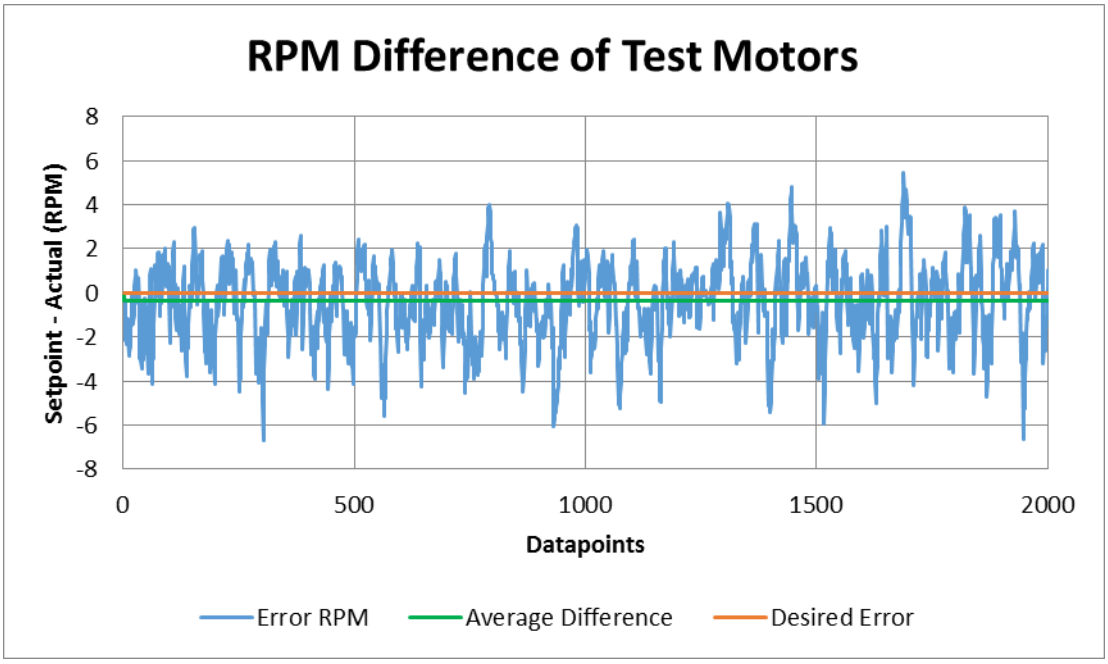
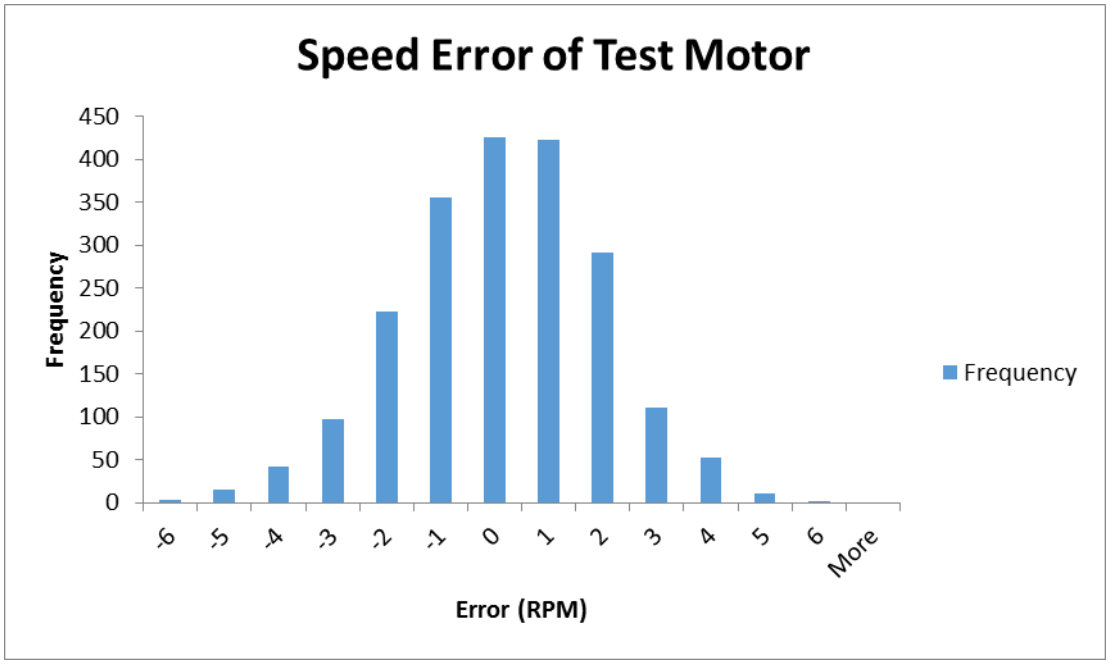


Figure 15 – Speed step response





PID *Figure 17 – Histogram of speed error of test motor*

Pressure results

Using the same dataset of the test vehicle running at 10 MPH, system pressure was recorded to compare to the test stand. While this data does provide some insight into operational pressures of the vehicle and the dyno it is not a truly equitable comparison. The vehicle runs entirely on hydraulics, therefore other systems could also alter the system pressure. The vehicle's goal is also to control the speed and not the pressure. On the dyno the pressure and speed are both being controlled, so it is a reasonable assumption that the dyno will have a substantially lower pressure variability than the vehicle. As figure 18 shows, the vehicle had an average system pressure of 1103.2 psi and a standard deviation of 53.2 psi. The dyno had an average system pressure of 1102.0 psi and a standard deviation of 16.1 psi.

Pressure statistics table	
Vehicle mean	1103.2
Vehicle stdev	53.2
Vehicle Coefficient of Variation	0.05
Dyno mean	1102.0
Dyno stdev	16.1
Dyno Coefficient of Variation	0.01

Figure 18 - Pressure statistics table

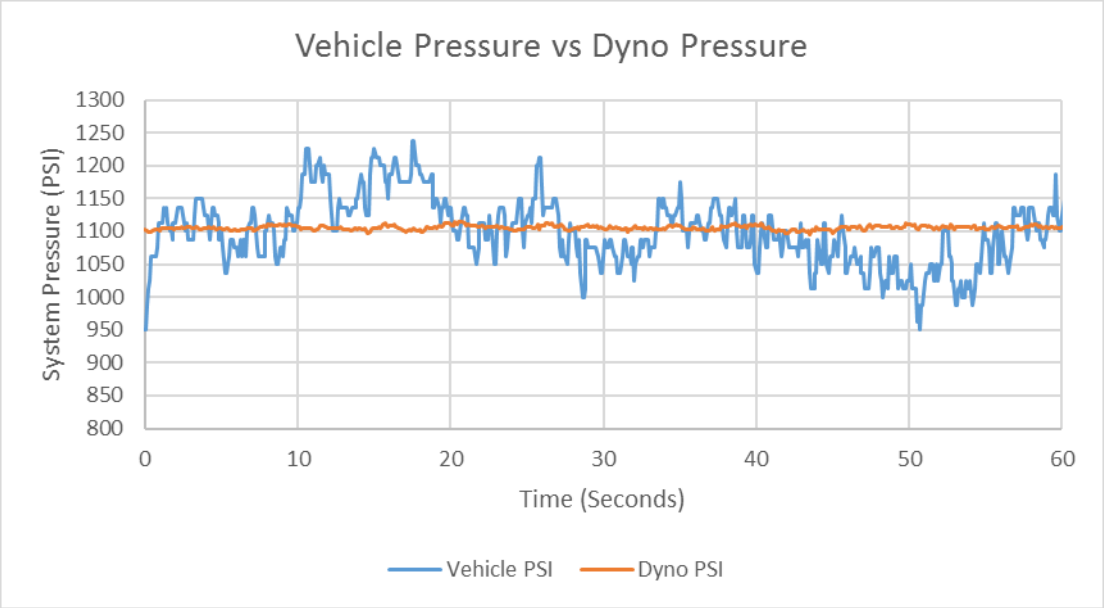


Figure 19 – Vehicle vs dyno pressure

Dialing in the PID gains for pressure was more involved than the speed PID. After exploring a range of P, I and D gains, I selected an I gain of 0.02 and a D gain of 0.01. I then went back to the P gain and tried the values that I had the best results with. The P gain of 0.008 was finally chosen due to its faster settling time and limited oscillation compared to the others. The final gains for this control were P: 0.008, I: 0.02, D: 0.01.

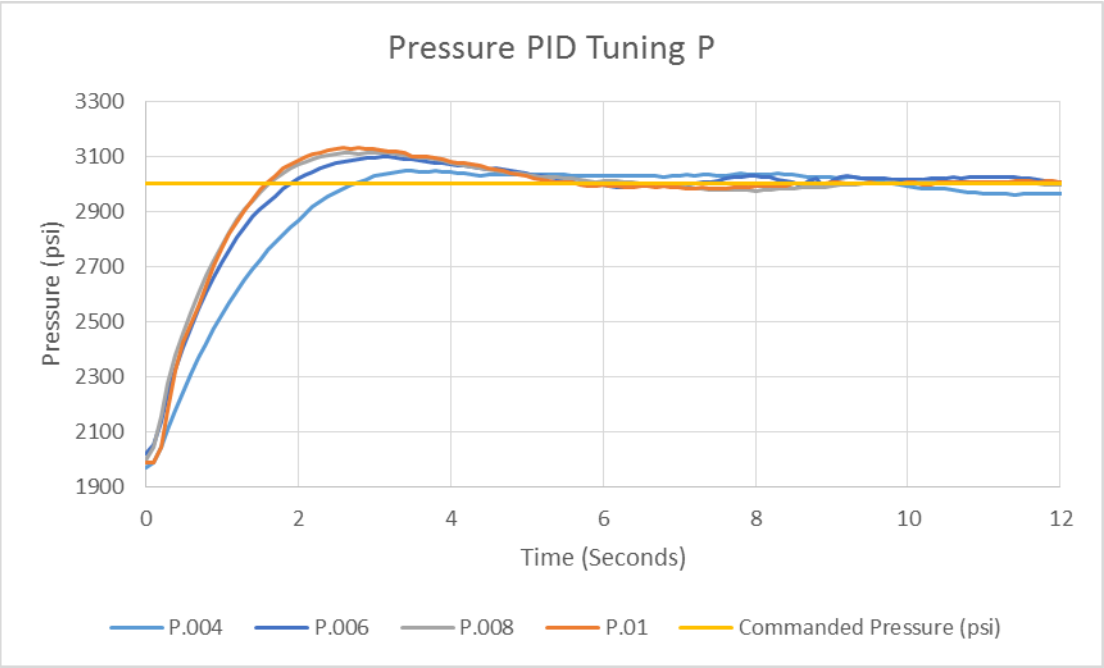


Figure 20 – Pressure PID P gain tuning

To show the dyno's pressure response a test was run to vary pressure in steps of 500 psi from 1000 psi to 3500 psi and then back down (figure 21). Due to pressure being less of a changing variable in tests on the dyno, 60 seconds per pressure step was chosen to show the holding ability of the pressure PID controller. In this test it took an average of 2.0 seconds for the system to stabilize between pressure steps. Figure 22 takes a look at the set point subtracted by the actual pressure. On average the system stays +/-25 psi around the set point. However two outlier peaks are present in this dataset with an amplitude of 80 psi. The PID corrected these two outlier peaks within 4.7 seconds.

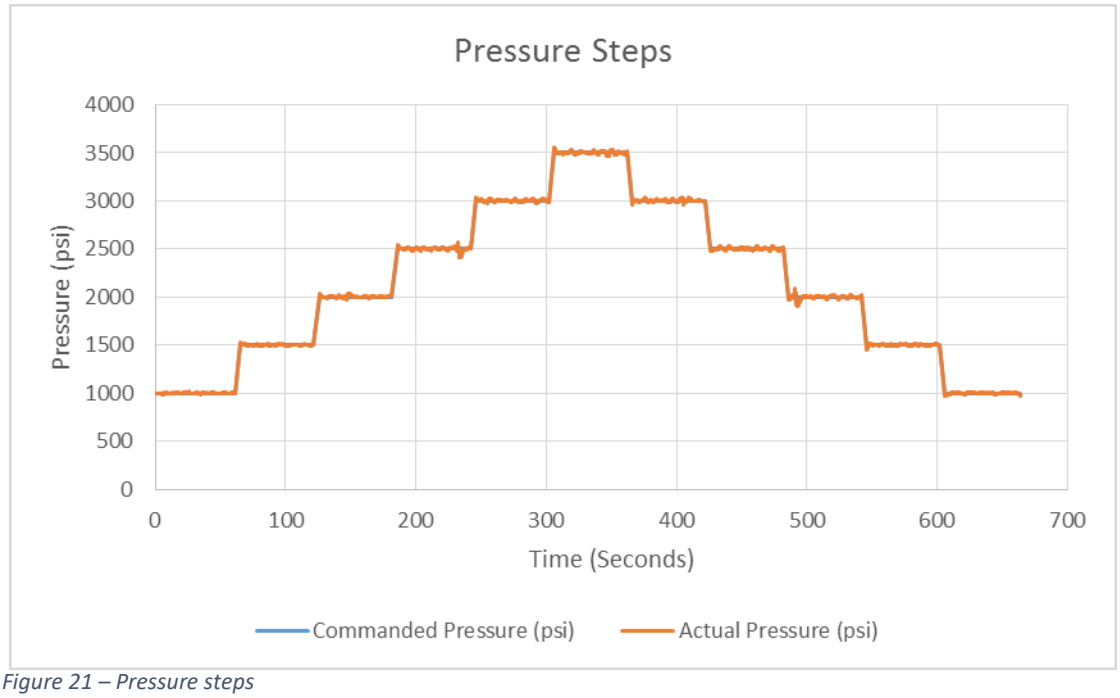


Figure 21 – Pressure steps

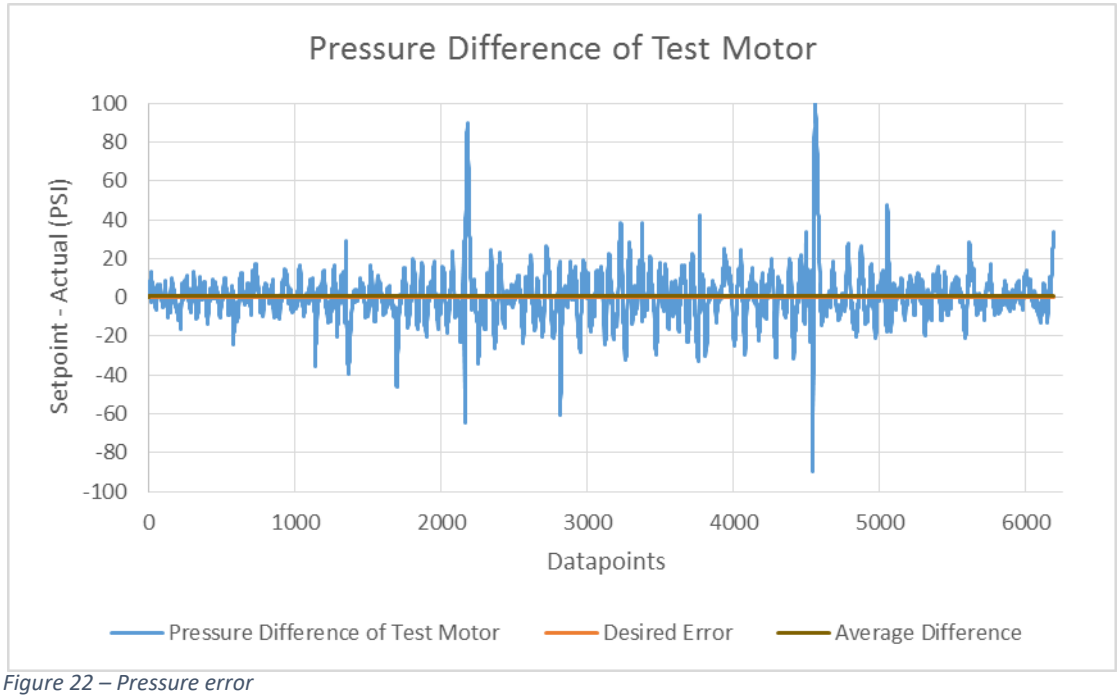


Figure 22 – Pressure error

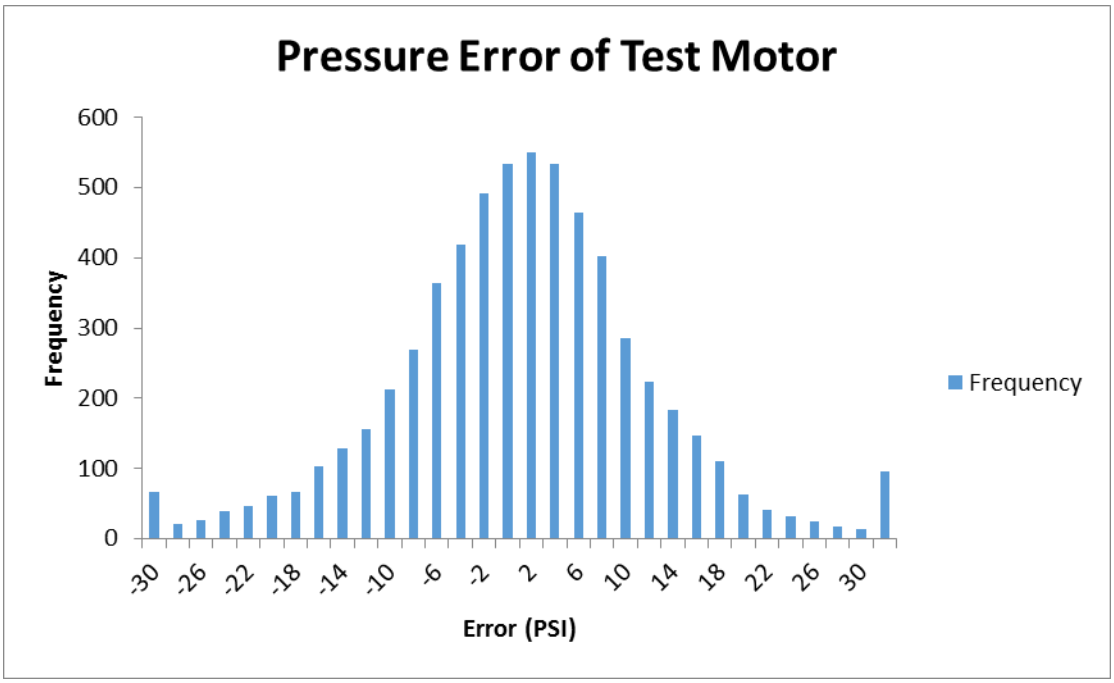


Figure 21- Histogram of pressure error of test motor

Conclusion

While this test stand is still being refined some great headway has been made. At present the controls software can maintain +/- 25 psi of commanded pressure and +/- 5 RPM of commanded speed. Comparing these results back to the vehicle that is being emulated, we get a lower coefficient of variation on both speed and pressure. This shows that the test stand is capable of controlling the hydraulic motors more accurately than we see in the field. The test stand not only controls the motors accurately, but also has an easy to use graphical user interface. This reduces the training time for new operators, and allows for easy operation.

Moving forward, the controls software could be improved by revamping the “test plan automation” loop in the real time side of the cRIO. Currently the system can control the speed, pressure, and displacement in steps and the data acquisition system can record the motors outputs from those stepped settings. Future development will require a profile of smoothly changing settings over a period of time that the data acquisition can record. This will more accurately represent what the motors are actually experiencing on a machine in the field.

The test stand will be used to gather data to create a prognostic system for hydraulic systems. This can detect early stages of bearing failure and warn the operator before catastrophic failure occurs, saving the operator costly repairs and downtime.

Work cited

"PID Theory Explained." PID Theory Explained - National Instruments, www.ni.com/white-paper/3782/en/.