

7-2020

Design of Control System with Feedback Loop for a Pulsatile Pump

Ian Scott Sanders
University of Arkansas, Fayetteville

Follow this and additional works at: <https://scholarworks.uark.edu/etd>



Part of the [Controls and Control Theory Commons](#), and the [VLSI and Circuits, Embedded and Hardware Systems Commons](#)

Citation

Sanders, I. S. (2020). Design of Control System with Feedback Loop for a Pulsatile Pump. *Graduate Theses and Dissertations* Retrieved from <https://scholarworks.uark.edu/etd/3763>

This Thesis is brought to you for free and open access by ScholarWorks@UARK. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of ScholarWorks@UARK. For more information, please contact uarepos@uark.edu.

Design of Control System with Feedback Loop for a Pulsatile Pump

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

Ian Sanders
University of Arkansas
Bachelor of Science in Electrical Engineering, 2018

July 2020
University of Arkansas

The thesis is approved for recommendation to the Graduate Council.

Robert Saunders, M. S.
Thesis Director

Morten Olgaard Jensen, Ph.D.
Committee Member

Roy A. McCann, Ph.D.
Committee Member

Alexander Nelson, Ph.D.
Committee Member

Abstract

This paper describes the design and implementation of a closed-loop proportional, integral, differential (PID) control system for a custom in-house pulsatile pump apparatus for the University of Arkansas Biomedical Department. The control system is designed to control a MOONS' PL34HD0L8500 hybrid stepper motor using a dual H-bridge motor driver network with four pulse-width modulated (PWM) inputs to drive a pulsatile pump apparatus at motor stepping frequencies up to 2kHz. The speed of the motor is controlled from a pressure profile transmitted from an external source over RS-232 communication that specifies the motor speed, number of datapoints, and an array of pressure data. Data will be measured from the pump using pressure, flow, and temperature sensors that will output analog data and be read to the control board using analog-to-digital converters (ADCs). A PID controller will be used to match the speed of the motor to the control data by calculating the error between the sensor outputs and the desired profile.

The circuit board is separated into two sections for the control board and motor circuit to isolate the 68V and motor circuitry from the rest of the control board circuitry. The control system circuitry was tested, and while the control board systems were found to be functional, the motor circuit was found inefficient due to the high L/R time constant of the motor, resulting in greatly reduced speed and torque. A new chopper driver design was proposed to solve this issue and simulations conducted through MATLAB Simulink to prove the feasibility of the design.

Acknowledgements

I would like to thank my advisor Robert Saunders for his knowledge and guidance during my time here at the University of Arkansas as well as for giving me the opportunity to pursue my master's degree. I would also like to thank Dr. Morten Jensen for allowing me to work on this project and presenting me with an interesting problem that I have enjoyed working through. A special thanks to Dr. Roy McCann and Dr. Alexander Nelson for their teaching and support through my academic career.

I would also like to thank my project partner Sam Stephens, and my friends Cecily Redman and Nicholas Blair for assisting and supporting me during this process. Lastly, I would like to thank my father, Mark Sanders, for all his knowledge and support he has presented me during my graduate and undergraduate career and for helping me sort out my thoughts and ideas throughout the course of this project.

Table of Contents

Chapter 1: Introduction	1
Chapter 2: Background and Theory of Operation.....	2
2.1 Stepper Motors	3
2.1.1 Types of Stepper Motors	3
2.1.2 Motor Characteristics	4
2.1.3 Functionality.....	6
2.2 H-Bridges and Gate Drivers	7
2.3 Control Systems	11
Chapter 3: Design	16
3.1 Circuit Design	16
3.1.1 Motor Circuit	17
3.1.2 Microcontroller	20
3.1.3 Pressure Sensors	21
3.1.4 Temperature Sensors	23
3.1.5 Flow Sensors.....	24
3.1.6 RS-232 Chip	25
3.1.7 Power Circuit.....	26
3.2 PCB Design.....	27
3.3 System Code.....	29

Chapter 4: Test Results	32
4.1 Control Board Testing.....	32
4.2 Motor Circuit Modifications and Results.....	35
4.3 Chopper Driver Design and Simulations	43
Chapter 5: Conclusion.....	55
5.1 Future Works.....	56
References.....	57
Appendix.....	58
Appendix A	58
Appendix B	60

List of Figures

Figure 1: (a) Permanent magnet, (b) variable reluctance, (c) hybrid stepper motors. [3]	3
Figure 2: MOONS' PL34HD0L8500 Hybrid Stepper Motor. [1]	5
Figure 3: (a) 8 lead connection options, (b) 8 lead wire diagram. [1]	6
Figure 4: H-Bridge Schematic.	8
Figure 5: System Block Diagram.....	14
Figure 6: Schematic Connector Page.....	16
Figure 7: Schematic Top Page.	17
Figure 8: H-Bridge Schematic.	18
Figure 9: Gate Driver Schematic.	19
Figure 10: MSP432E401Y Schematic.	21
Figure 11: Pressure Sensor Schematic.....	23
Figure 12: Temperature Sensor Schematic.	24
Figure 13: Flow Sensor Schematic.	25
Figure 14: RS-232 Communication Circuit.....	26
Figure 15: Power Supply Circuit.	27
Figure 16: Full PCB Design with labels.	28
Figure 17: Code flow diagram.	31
Figure 18: UART transmit and receive.....	34
Figure 19: UART transmit and receive zoom-in.	34
Figure 20: RS-232 transmit vs. CMOS transmit signals.....	35
Figure 21: Original gate driver schematic.	36
Figure 22: Gate Driver Breakout PCB Design.	37

Figure 23: Modified PCB board.	38
Figure 24: PWM input configuration to H-Bridge.	39
Figure 25: Stepper motor outputs at 10Hz stepping rate.	40
Figure 26: Stepper motor outputs at 100Hz stepping rate.	41
Figure 27: Stepper motor outputs at 150Hz stepping rate.	42
Figure 28: H-Bridge with overcurrent detector schematic.	44
Figure 29: Updated PCB design with chopper driver.	45
Figure 30: Pulsatile Pump Simulink Model.	46
Figure 31: PWM Generation Subsystem.	47
Figure 32: PWM outputs at 500 steps/sec.	47
Figure 33: Chopper driver to PWM model.	48
Figure 34: Simulink Pressure Outputs.	50
Figure 35: Simulink Flow Outputs.	50
Figure 36: Simulink Step Angle and Piston Position.	51
Figure 37: Simulink Phase Current.	51
Figure 38: Simulink Phase Voltage.	52
Figure 39: Error vs. PID Calculated Error.	53
Figure 40: Frequency Input with PID adjustment.	53
Figure 41: Pressure Results with PID controller.	54
Figure 42: Original PCB Design, Enlarged.	58
Figure 43: Modified PCB Design, Enlarged.	59

List of Tables

Table 1: Bipolar, Full Step Configuration. [1].....	7
Table 2: Effects of Independent P, I, and D Tuning. [8]	13
Table 3: SR flip-flop logic.....	49
Table 4: Board Materials List.....	60

Chapter 1: Introduction

As cardiovascular diseases continue to be a leading cause of death around the world, the importance of the development of methods and technologies to improve cardiac function has been ever growing. Issues regarding ethical considerations and feasibility of cardiovascular testing have been prominent in experimentation of these methods and technologies until the development of benchtop in-vitro testing setups like the pulsatile pump. The pulsatile pump has been a huge breakthrough in medical studies as it allows for detailed investigations into venous/vascular systems and for testing of medical devices without needing to endanger patients [12]. The ability of these systems to mimic arterial flow and pressure, is essential for quality assurance and calibration of all clinical techniques of blood flow and pressure measurements [15]. Some commercial pulsatile pumps, such as the ViVitro Labs Superpump, have been developed to allow for simulation of the heart at varying heart rates to produce accurate pressure and flow outputs [13].

This work describes the design and implementation of a closed-loop proportional, integral, differential (PID) control system for a custom in-house pulsatile pump apparatus for the University of Arkansas Biomedical Department. The proposed pulsatile pump is a mechanical and hydraulic system with a translational piston moved by a hybrid stepper motor to replicate the flow and pressure functionality of a heart. The objective is to design a PID control system to observe the pressure, flow, and temperature of a fluid apparatus and implement closed-loop feedback to adjust the pump rate. The feedback loop will be initialized from a pressure profile sent to the control board through RS-232 communication from an external source. The profile includes the motor frequency rate, number of data points, and the pressure profile data. The motor frequency variable will set the initial step-rate of the stepper motor and the profile data

then compared to the measured data output of the pump to calculate the error variable for the PID controller. The project is comprised of a custom-made fluid apparatus designed in-house by Sam Stephens and a control board that will control and implement the following hardware:

- 6 Pressure Sensors (MLT0670)
- 6 Flow Sensors (FL1027)
- 6 Temperature Sensors (LM34DZ)
- A 68V Connex Electronics SMPS300R with 6A (Nominal), 7.2A(Peak)
- PL34HD0L8500 Bipolar, Hybrid Stepper Motor
- A 15V 1A DC Power Supply
- External RS-232 source connection through a DB9 header

The proposed pulsatile pump control system differs from commercial products from its adaptability of multiple sensor inputs and its low cost. The system is adaptable in its ability to read and control eighteen analog sensor inputs for pressure, flow, and temperature from the pump system. Each of these sensors then assist in the calculation of the control error to modulate the motor's stepping frequency to meet the desired pressure output. The final product of this setup will also allow for either flow or pressure profiles to be implemented as the reference input to allow for the system to analyze pulsatile flow and pulsatile pressure outputs.

Chapter 2: Background and Theory of Operation

2.1 Stepper Motors

2.1.1 Types of Stepper Motors

Stepper motors are widely used in commercial and industrial applications due to their ability to provide accurate position and speed control in open-loop systems. Some of these applications include CNC machines, 3D printers, and robotic systems [2]. The level of precision and speed control of a stepper motor is dependent on the internal design of the stator windings and magnetic rotor which fall within three classifications: variable reluctance, permanent magnet, and hybrid stepper motors, see Fig. 1.

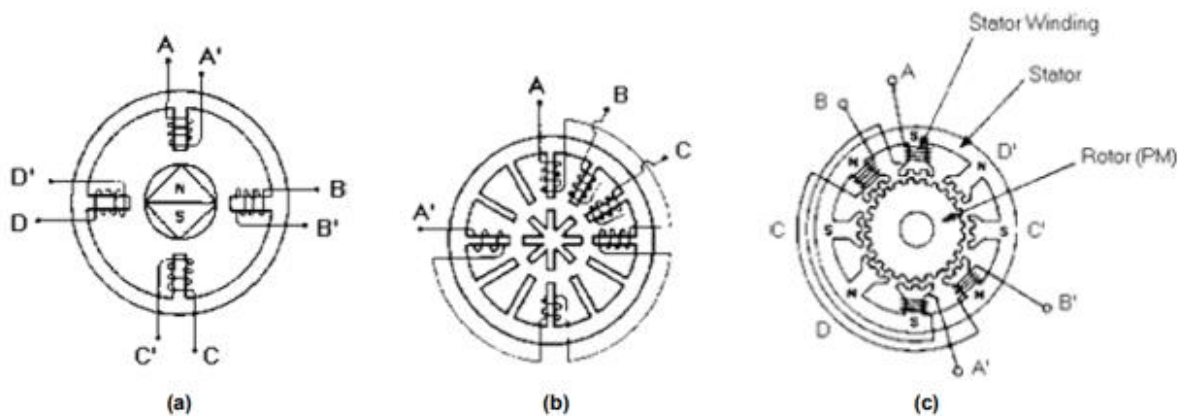


Figure 1: (a) Permanent magnet, (b) variable reluctance, (c) hybrid stepper motors. [3]

The variable reluctance design is more complicated than the hybrid and permanent magnet designs. This is due to the rotor being made of a ferromagnetic material, normally iron, with protruding teeth that are attracted to the stator poles when they are generating a magnetic field, see Fig. 1b. This design results in a higher resolution, but with the main drawback of providing a low dynamic torque [3].

A permanent magnet design incorporates a permanent magnetic rotor with alternating north and south poles that interact with the magnetic field of the stator windings as they are energized, see Fig. 1a. The interaction between the permanent magnet and the stator results in a residual torque, or *détente* torque, to be left even when no current is present, requiring a small torque to be applied to move the motor from its equilibrium position. The permanent magnet design provides a high torque so is very popular in small stepper motor designs but is normally limited to a low-resolution rate [3].

The hybrid stepper motor design is a combination of both the variable reluctance and permanent magnet designs. Using a permanent magnetic rotor with two slotted, multi-tooth rotor end-caps that have their teeth displaced by one-half tooth-pitch as well as a toothed stator that will alternate its stator charge, causing the rotors teeth to align with the charged stator teeth, see Fig. 1c [4]. Each of these rotor endcaps function as one pole of the permanent magnet (one north and one south) so when a stator is energized, the corresponding north and south teeth align with their opposingly charged stator teeth. The result of this design combination is a stepper motor with both high torque and high resolution. Since a high torque is necessary to translate the pumps piston and a high resolution needed to simulate pressure and flow levels at varying stepping rates, the hybrid stepper is the best among these three motor designs for this application.

2.1.2 Motor Characteristics

For this application, the system will implement the PL34HD0L8500 Hybrid Stepper Motor from MOONS' Industries, see Fig. 2, providing a 1.8° step angle and a high torque to support the pulsatile pump's function.

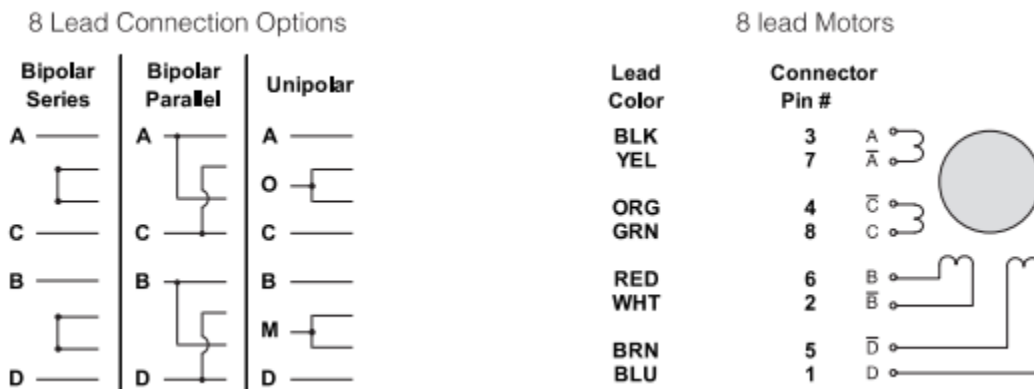


Figure 2: MOONS' PL34HD0L8500 Hybrid Stepper Motor. [1]

Having a low step angle allows for more precise movement of the system for a full 360° rotation. The resolution for this step angle is determined to be 200 steps from Equation (1).

$$\text{resolution} = \frac{360^\circ}{\text{step angle}} = \frac{360^\circ}{1.8^\circ} = 200 \frac{\text{steps}}{\text{revolution}} \quad (1)$$

The PL34HD0L8500 has 2-phases with positive and negative coil windings at opposite stator poles and is configured as a bipolar motor. The bipolar connection is chosen over the unipolar connection to generate a higher overall torque output from the motor. In the bipolar setup, the winding is fully energized and produces more holding torque from the motor, while the unipolar connection will only partially energize the winding and produce less torque as a result. The wiring diagram for the 8 lead motor and 8 lead connection options can be seen in Fig. 3 to wire the stepper motor to its corresponding specifications.



(a)

(b)

Figure 3: (a) 8 lead connection options, (b) 8 lead wire diagram. [1]

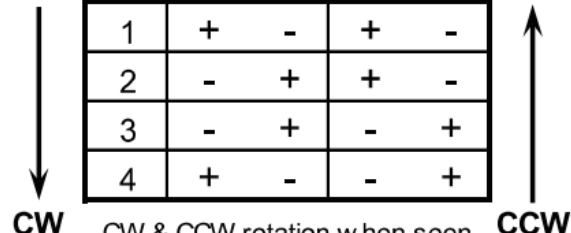
Since the motor is configured as a bipolar, series stepper motor, the connection option seen in the first column of Fig. 3a is implemented using the wiring diagram in Fig. 3b. Resulting in the \bar{A} and \bar{C} leads to be tied together and the \bar{B} and \bar{D} leads to be tied together, creating the phase 1 winding of leads A and C, corresponding to A+ and A-, and the phase 2 winding of leads B and D, corresponding to B+ and B-.

2.1.3 Functionality

A hybrid stepper motor is controlled by generating electrical pulses to allow current to flow through the motor's windings and energize the stator of the motor's poles, moving the magnetic rotor(s) north and south poles to the energized position [1]. The direction in which the current is flowing through the windings during these electrical pulses determines the stator poles direction, and thus the motor's direction. As current is flowing into the A+ phase and out the A- phase of a motor winding, the A+ phase will be positive while the A- phase will be negative. If the current is flowing in the opposite configuration (A- to A+) then the A+ phase will be negative and the A- phase will be positive. The change in the current flow through the stator windings between these states causes the magnetic rotor teeth to move and align to the opposing energized pole. For a bipolar stepper motor to run through a full rotation, both phase windings need to work in tandem to alternate between their logic states. Table 1 shows the phase configurations for the motor to be able to rotate in full steps in both clockwise (CW) and counterclockwise (CCW) directions for a bipolar stepper motor.

Table 1: Bipolar, Full Step Configuration. [1]

STEP	Phase 1		Phase 2	
	A	C	B	D
1	+	-	+	-
2	-	+	+	-
3	-	+	-	+
4	+	-	-	+



 CW & CCW rotation w hen seen
 from flange side of the motor.

As each phase winding is energized in these four step configurations, the motor will rotate in either the CW (forward) or CCW (reverse) direction dependent on the stepping order shown in Table 1 by the arrows. For these stepper motor configurations to work, a switching circuit and driver are needed to redirect the current flow and energize the motor windings in this manner. For this reason, a dual, parallel H-Bridge network and half-bridge gate drivers are used to power the hybrid stepper motor.

2.2 H-Bridges and Gate Drivers

An H-bridge circuit is a network of at least four transistors that function as power switches to alternate the flow of current from a power rail to a load connected between the power switching nodes [5]. A picture of the designed H-Bridge circuit can be found in Fig. 4.

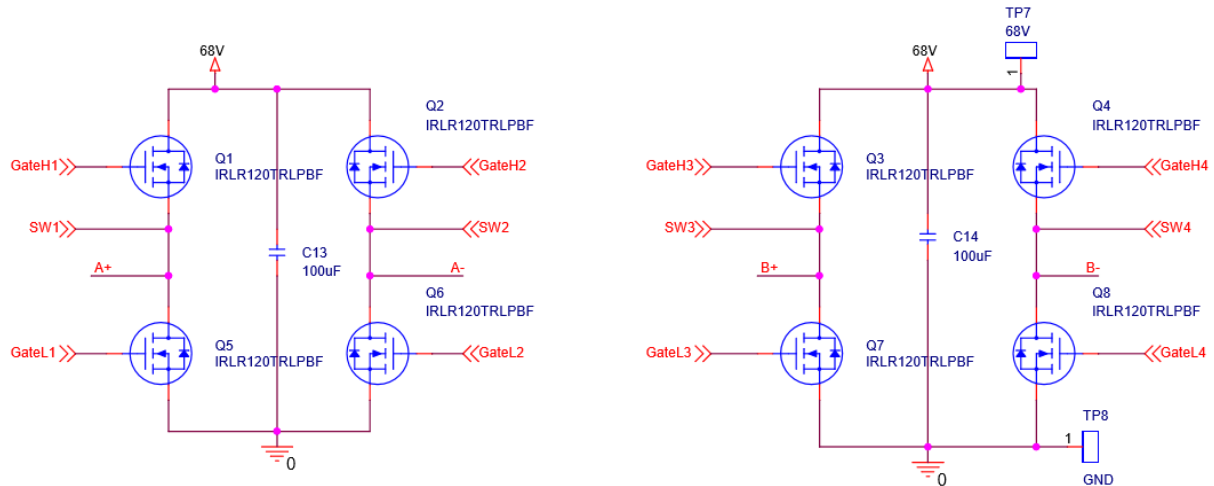


Figure 4: H-Bridge Schematic.

The proposed H-Bridge circuit uses four IRLR120TRLPBF N-Channel 100V 7.7A MOSFETs to function as the high-side and low-side power switches between the motor supply and ground to the hybrid stepper motor phase windings. These MOSFETs were chosen for their high current and voltage capability to function with the SMPS300R switch-mode power supply that has the capability of supplying 68-76V and 6A. The IRLR120 also supports a gate-source voltage (V_{gs}) of 5V to 10V. The switching of the MOSFET gate is controlled using a gate driver IC that buffers a digital signal, a pulse-width modulated (PWM) signal, generated by a microcontroller.

The gate driver is a very important component for running a stepper motor using an H-Bridge. One of these reasons is due to the gate-source voltage that can be seen on the high-side MOSFETs when the gate is charged. When the high-side MOSFET is turned on, current is able to flow through the drain and source to the motor winding, thus resulting in the source voltage of the MOSFET being equivalent to the bus voltage ($V_s = V_{bus} = 68V$), increasing the gate-source threshold voltage necessary for the FET to stay ON resulting in:

$$V_{gs(th)} = V_g + V_s = 8V + 68V = 76V \quad (2)$$

The gate driver chosen is an ISL78434 half-bridge gate driver that functions with independent PWM inputs to its LI, low input, and HI, high input, pins that activate the high-side and low-side driver outputs. To reach the $V_{gs(th)}$ of 76V when the MOSFET is activated, a bootstrap capacitor is incorporated that is connected between the MOSFET's source and the bootstrap supply voltage pin. When the drivers high-side gate output is off, the bootstrap capacitor is charged to the bus voltage level, 68V, plus the gate drivers supply voltage, 15V, resulting in a charged capacitor voltage of 83V to the high-side gate output when active. The internal make-up of the ISL78434 incorporates an integrated bootstrap switch that charges the bootstrap capacitor whenever the LI pin is logic high and the HI pin is logic low. So when the HI pin is logic high, the high-side driver output (HO) is activated and the boot capacitor starts to discharge until the inputs are switched again (LI = high, HI = low). The component datasheet specifies how the bootstrap capacitor value is calculated in relation to the gate driver and MOSFET's voltage and current characteristics. Equations 3, 4, and 5 show these equations along with their variable definitions:

$$\Delta V_{BOOT} = V_{DD} - V_{DH} - V_{GS_{min}} \quad (3)$$

$$Q_{TOTAL} = Q_{G_{max}} + (I_{GSLKG} + I_{HBQ} + I_{HBSLEAK}) * t_{ON} \quad (4)$$

$$C_{BOOT} = \frac{Q_{TOTAL}}{\Delta V_{BOOT}} \quad (5)$$

Where:

- V_{DD} = Gate driver supply voltage
- V_{DH} = Gate driver high current forward voltage
- $V_{GS_{min}}$ = Minimum FET V_{GS} threshold voltage
- $Q_{G_{max}}$ = Maximum FET gate charge

- $I_{GS_{LKG}}$ = Maximum FET gate-source leakage current
- I_{HBQ} = Gate driver boot quiescent current
- $I_{HBSLEAK}$ = Gate driver boot leakage current
- t_{ON} = Time interval FET gate will be charged

The largest issue that can occur within a motor driver circuit is shoot-through, where the high-side and low-side MOSFETs of a half-bridge are simultaneously turned-on and result in a short-circuit of the motor supply. Shoot-through can cause damage to the H-Bridge circuitry and motor, resulting in degradation in performance/lifetime or destruction of the motor and H-Bridge circuitry. To prevent this, the ISL78434 incorporates shoot-through protection using Adaptive Dead Time Control (ADTC) of its internal circuitry to prevent the driver from turning both the high-side and low-side FETs on at the same time. The ADTC works by making sure there is a dead-time interval where both high-side (HO) and low-side (LO) outputs are OFF, that occurs during the transition between output stages of the gate driver (i.e. HO fall to LO rise). This is done by the driver sensing when the HO or LO pins drop below their specified V_{ADTC} threshold voltages, which triggers an ADTC time delay before the opposing HO or LO are activated. These voltage thresholds are monitored using the individual source and sink pins located at each driver output of the gate driver IC that support up to 3A sourcing and 4A sinking of current.

Along with generating the necessary signal levels to charge the MOSFET gates, the gate driver also monitors the internal supply voltage and bootstrap voltage levels for undervoltage conditions that will disable the internal circuitry of the driver. The VDD under-voltage lockout (UVLO) is activated when the gate driver supply falls below a minimum rising threshold voltage of $V_{DDR_{UVLO}(\min)} = 6.8V$ that will disable the sourcing outputs of both driver outputs (HO and LO) as well as the sinking output of the low-side driver. The bootstrap UVLO is activated when

the high-side bootstrap supply voltage, pin HB, referenced to the high-side driver reference voltage, pin HS, drops below the minimum rising threshold of $V_{\text{HBRUVLO}(\text{min})} = 6\text{V}$, disabling the high-side driver sourcing output but allowing the low-side driver output to respond to driver inputs [6].

2.3 Control Systems

A control system is a configuration of system components that work together to provide a desired system response using a controller [7]. Control systems can be applied in many different methods from being mechanical or software based, manual or automatic, and functioning in closed-loop or open-loop configurations. The difference between open-loop and closed-loop systems is significant. A closed-loop system measures the actual output and then compares it to the desired output, calculating the error between the desired and measured, and then modifying the input until the actual output matches the desired output. Open-loop systems don't function as well as closed-loop systems as there is no error variable or feedback, so the controller controls the input directly and changes it over time in hopes that the output resembles the desired input.

The earliest application of an automatic feedback controller for industrial processes was the flyball governor developed in 1769 by James Watt's. The flyball governor was a fully mechanical device that controlled the speed of a steam engine by controlling a valve which regulated the amount of steam entering the engine [7]. The introduction of feedback controllers like the flyball governor, help to control the outputs of systems and allow for better accuracy and stability in response to their input criteria. Outside of mechanical systems, feedback controllers and networks can also be developed in software using an embedded system, such is the case for this project.

Along with the multiple configurations for the control system itself, the controller can also be implemented in different ways depending on what is being controlled. Some of these possible controller make-ups are the Proportional (P), Proportional Integral (PI), Proportional Derivative (PD), and Proportional Integral Derivative (PID) controllers. These controller configurations are based around the three controller gains: proportional gain (K_p), integral gain (K_i), and derivative gain (K_d) relative to their effect on the feedback error. This relationship is best seen from the PID controller algorithm expressed in the time-domain:

$$u(t) = K_p[e(t) + T_d \frac{de(t)}{dt} + \frac{1}{T_i} \int_0^t e(\tau) d\tau] \quad (6)$$

Where $u(t)$ is the control variable, $e(t)$ is the system error, T_d the derivative time constant, and T_i the integral time constant. Equation (6) can also be re-written as:

$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{d}{dt} e(t) \quad (7)$$

$$\text{Where } K_d = \frac{K_p}{T_d} \text{ and } K_i = \frac{K_p}{T_i}$$

The PID controller can also be expressed in a discrete form by manipulating equation (7), with the controller gains expressed in hat notation to signify the shift between the discrete time and continuous time:

$$u(t) = \widehat{K}_p e(k) + \widehat{K}_i \sum_{j=1}^{n-1} e(j) + \widehat{K}_d e_c(k) \quad (8)$$

For equation (8), the control variable is the summation of the proportional ($K_p e(k)$), integral ($K_i \sum_{j=1}^{n-1} e(j)$), and derivative ($K_d e_c(k)$) variables. The functionality of these control variables to the error signal can be summarized as: the proportional term calculating the current error, the integral term calculating the sum of all error, and derivative term calculating the rate of change of the error. The effects of each of the controller terms for a closed-loop system can be best summarized in Table 2 [8].

Table 2: Effects of Independent P, I, and D Tuning. [8]

Closed-Loop Response	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Increasing K_p	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_i	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_d	Small Decrease	Decrease	Decrease	Minor Change	Improve

Based on these effects, the expected output functionality for each controller configuration can be analyzed to a certain degree. The PI controller can be seen to result in a lower rise time and a decrease in the steady-state error, but ultimately result in an increase in overshoot and settling time on the output. This result matches the effects on the error described previously. The proportional gain will amplify error to increase the overshoot and decrease the rise time, and the integral gain will sum all the calculated error to result in an increase to overall settling time of the output. The PD controller can be seen to improve the overall functionality of the basic P controller, but this is not necessarily true since the derivate term is very unpredictable due to its effect on the error. Since the derivative term determines the rate of change in the error over time, a decrease in overshoot and instability on the output can be expected. With these functions in mind, the PID controller can be expected to have an increase in overshoot and steady-state error, with an increase in settling time and reduction in stability. The PID controller will function overall as a more efficient solution to the stepper motor control compared to the other

configurations when tuned correctly, since each of the controller terms will assist one another in reaching the desired output.

The function of the control system for the pulsatile pump is to modify the stepping speed of the hybrid stepper motor through a closed-loop PID controller by calculated the error between the desired pressure output and the measured pressure output. A block diagram of the system can be seen in Fig. 5.

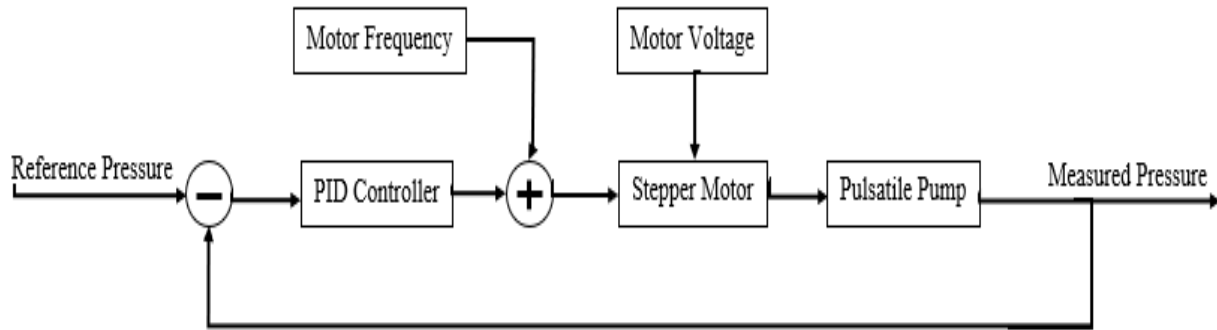


Figure 5: System Block Diagram.

The desired pressure data is sent to the microcontroller using RS-232 communication to a Universal Asynchronous Receiver Transmitter (UART) port. The profile input will contain a message structure detailing the motor frequency, the number of data points, and the profile data. This information is used to set the motor stepping rate and the reference pressure for the control system. The measured pressure is fed back into the system through an Analog-to-Digital Converter (ADC) to calculate the controller error using the microcontroller, see equation (9).

$$Error = Reference\ Pressure - Measured\ Pressure \quad (9)$$

The error is then passed through the PID controller, where the proportional, integral, and derivative variables are derived from equation (8) to give us the results shown in equations (10), (11), and (12). The Integral and Previous_Error values are initialized to zero prior to motor start.

$$\textit{Proportional} = K_p * \textit{Error} \quad (10)$$

$$\textit{Integral} = (\textit{Integral} + \textit{Error}) * K_i \quad (11)$$

$$\textit{Derivative} = (\textit{Error} - \textit{Previous_Error}) * K_d \quad (12)$$

Through simulations of the pulsatile pump apparatus in MATLAB Simulink, discussed in section 4.3, the PID gain values were determined to be: $K_p = 25$, $K_i = 10$, and $K_d = 4$ to be able to result in a stable output to reach the desired pressure profile.

Chapter 3: Design

3.1 Circuit Design

From discussions with Dr. Jensen and Sam Stephens in the biomedical department, the system needs to support a 72V and 5A supply for a PL34HD0L8500 2-phase hybrid stepper motor, have the ability to take in 18 independent analog inputs for pressure, flow, and temperature sensors, and be able to communicate through RS-232 communication to receive a data profile. The schematics for the system were designed using the OrCAD Capture CIS v17.2.0 Software supplied by the University of Arkansas Electrical Engineering Department. A top page and connection page for the schematic can be found in Fig. 6 and Fig. 7 to show all the systems inputs and outputs along with their connections within the circuit.

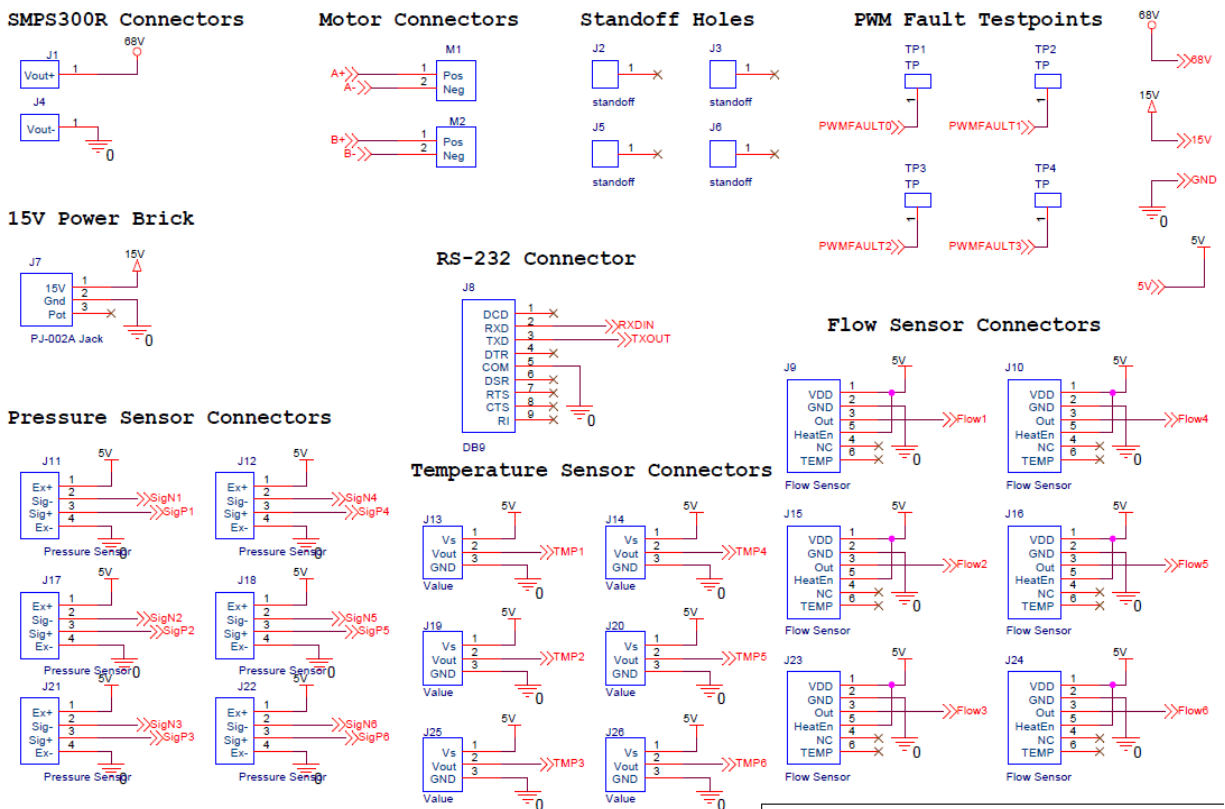


Figure 6: Schematic Connector Page.

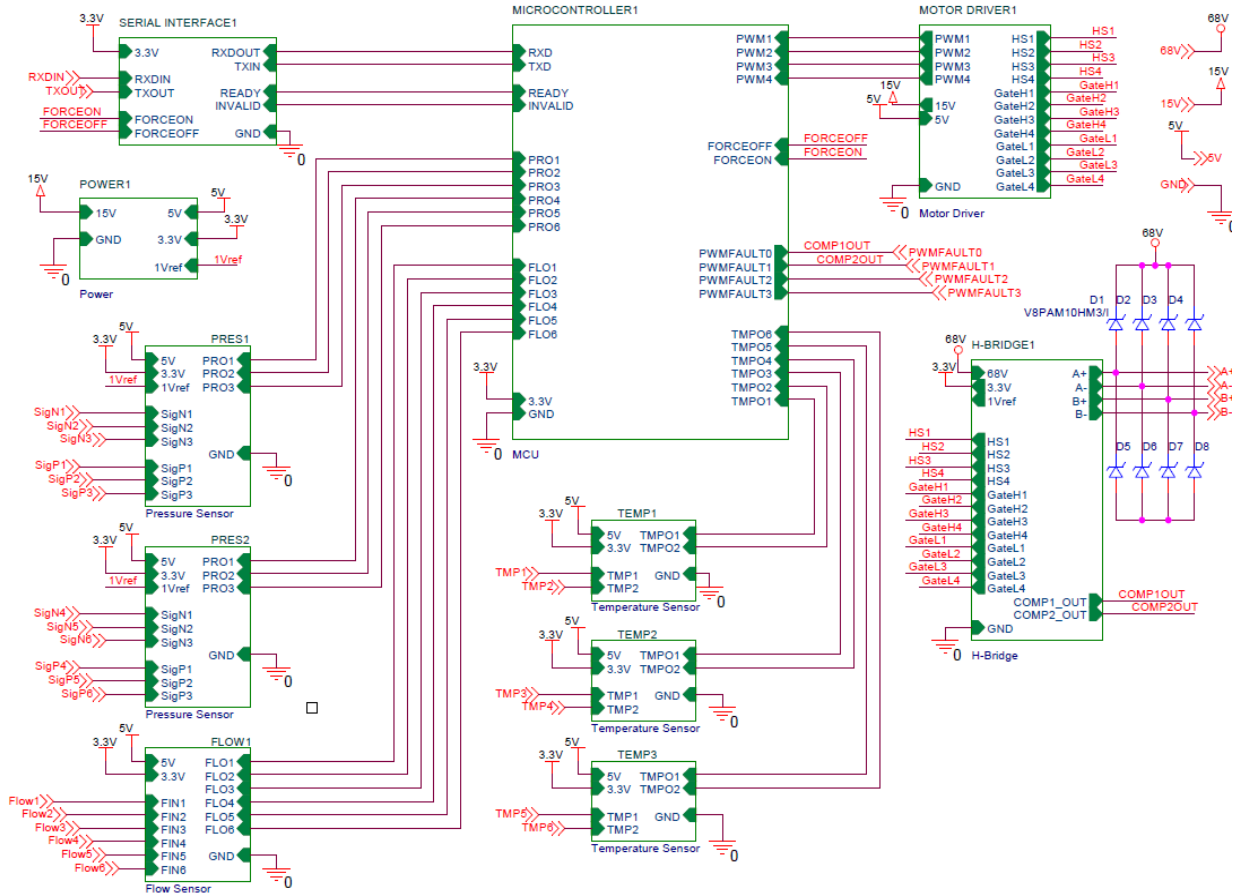


Figure 7: Schematic Top Page.

3.1.1 Motor Circuit

The power supply to meet the motor voltage requirements is a Connex Electronics SMPS300R configured for a mains voltage of 120V and output voltage of 48V, determined from the product datasheet, to supply the system with 68-76V and 6A nominally. Based on the motor functionality described in section 2.1, it was determined that a dual H-bridge network is needed to change the flow of current through the motor windings to match the switching states shown in Table 1. As described in section 2.2, each H-bridge was designed with IRLR120TRL PBF MOSFET transistors rated at 100V and 7.7A and a 100uF DC Bus capacitor attached between the motor supply and ground. The H-Bridge schematic can be seen in Fig. 8.

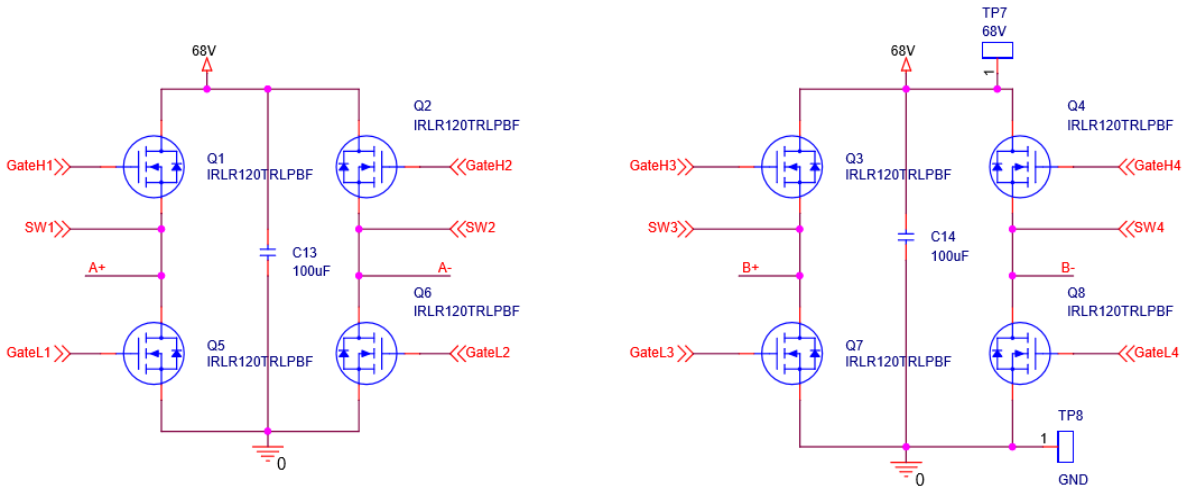


Figure 8: H-Bridge Schematic.

A configuration of four half-bridge gate drivers, ISL78434, with four PWM outputs is was determined to drive the MOSFET gates of the dual H-Bridge network. The ISL78434 is driven by a supply voltage of 15V to function within its supply range of 8 to 18V along with the Enable pin supplied with 5V to enable the driver for PWM inputs. The gate driver is configured for two independent PWM inputs, HI and LI pins, to drive each high-side and low-side driver output, HO and LO pins, with independent sourcing and sinking pins for each output. As can be seen in Fig. 9, each gate driver pair functions off two PWM inputs that are opposing configurations between the HI and LI pins on each driver. This is done to prevent any possible shoot-through on the H-bridge. For example, when PWM1 is active high and PWM2 is active low, both the Q1 and Q6 MOSFETs are turned ON while MOSFETs Q2 and Q5 remain off.

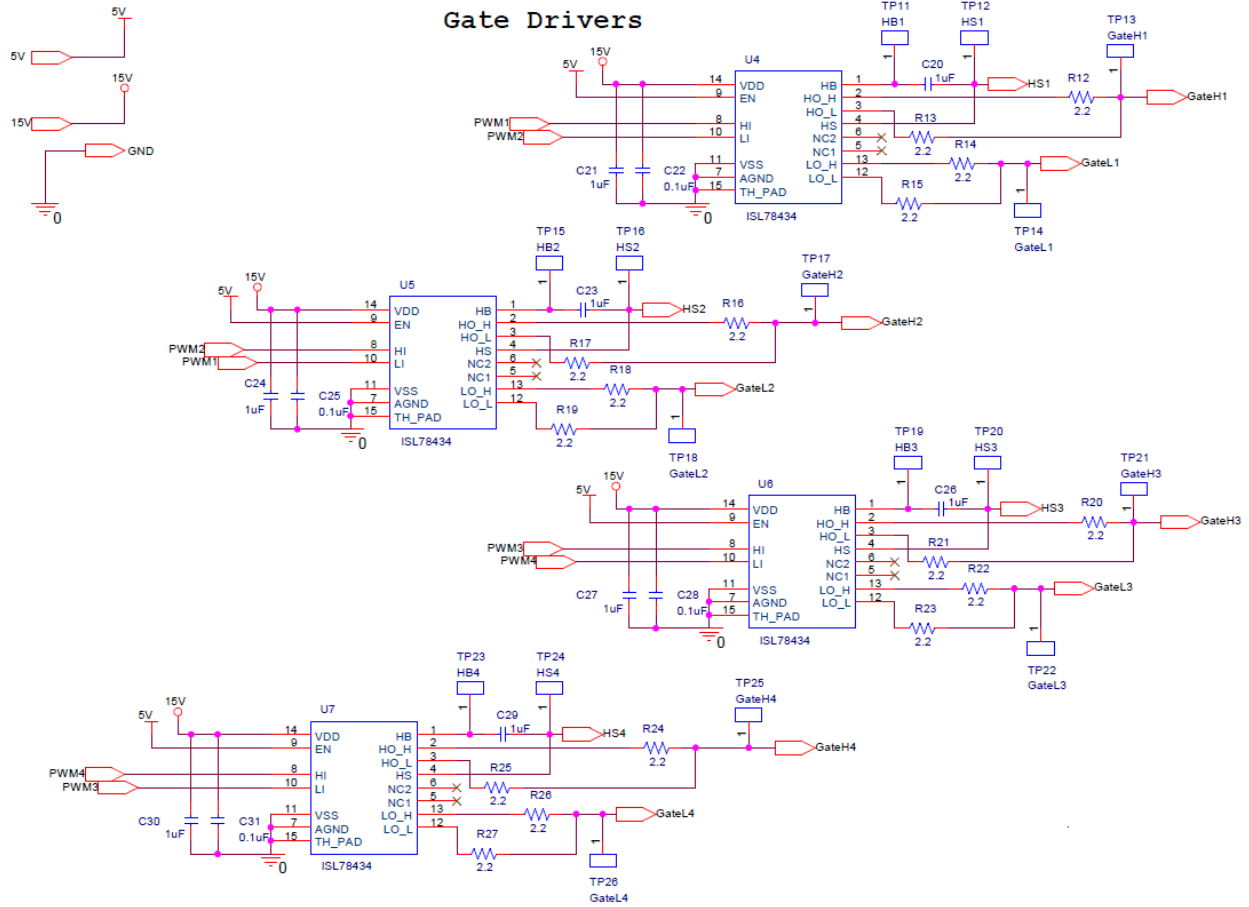


Figure 9: Gate Driver Schematic.

As mentioned in section 2.2, for the high-side MOSFET to stay on, the high-side driver needs to reach a V_{gs} threshold voltage of 69V on the MOSFET gate. To meet this threshold, the bootstrap capacitor boosts the high-side gate driver output to an equivalent voltage of 83V which slowly discharges over time when HO is activated. To make sure a sufficient bootstrap capacitor is selected, equations (3), (4), and (5) are used to determine the on-time of the capacitor. To calculate the on-time, the ΔV_{BOOT} and Q_{Total} are first calculated from equations (3) and (5), and then used to calculate t_{ON} from equation (4). The results of calculation of ΔV_{BOOT} can be seen in equation (13).

$$\Delta V_{BOOT} = 15V - 1V - 1V = 13V \quad (13)$$

With $C_{BOOT} = 1\mu F$, the total charge is determined from equation (5) and calculated in equation (14).

$$Q_{Total} = C_{BOOT} * \Delta V_{BOOT} = (1\mu F) * (13V) = 13\mu C \quad (14)$$

With the total charge of $13\mu C$, the on-time of the capacitor is determined from equation (4) and calculated in equation (15).

$$t_{ON} = \frac{Q_{TOTAL} - Q_{Gmax}}{I_{GSLKG} + I_{HBQ} + I_{HBSLEAK}} = \frac{13\mu C - 12nC}{100nA + 320\mu A + 80\mu A} = 0.03247 \text{ sec} \quad (15)$$

As can be seen from these calculations, with $C_{BOOT} = 1\mu F$ the slowest the MOSFET gate can be switched to allow for the driver to stay on is 31Hz. This is well within the functional region of the system as we are expected to be running at frequency rates up to 2kHz on the motor. The sourcing and sinking resistors of the gate driver outputs were selected to be 2.2 ohms, to allow for fast switching of the MOSFET gates.

3.1.2 Microcontroller

Following the motor driver design, the microcontroller for the circuit is then determined based on the system peripheral requirements. The MSP432E401Y microcontroller was chosen since it can support all the needed peripherals for the 18 analog sensors, 4 PWM outputs, and the UART module for the RS-232 communication. The MSP432E401Y uses a 3.3V supply voltage and can be programmed through a Joint Test Action Group (JTAG) interface. Ceramic 22uF decoupling capacitors were placed at all supply pins of the microcontroller to filter out noise and prevent any possibility of overvoltage characteristics. The microcontroller is programmed using Texas Instruments (TI) Code Composer Studio v9.1 software with a TI XDS110 Debug Tool to analyze the active registers and code functionality connecting through the JTAG port. Since the microcontroller functions at 3.3V, the input signals from the analog sensors, as well as the RS-232 signal, must be buffered to function within a 3.3V voltage range to prevent damage to the

microcontroller or its pins. The schematic design for the microcontroller can be seen in Fig. 10.

The JTAG connector conforms to the compact-TI 20 pin pinout and circuit design for receiving and transmitting information between Code Composer Studio and the MSP432.

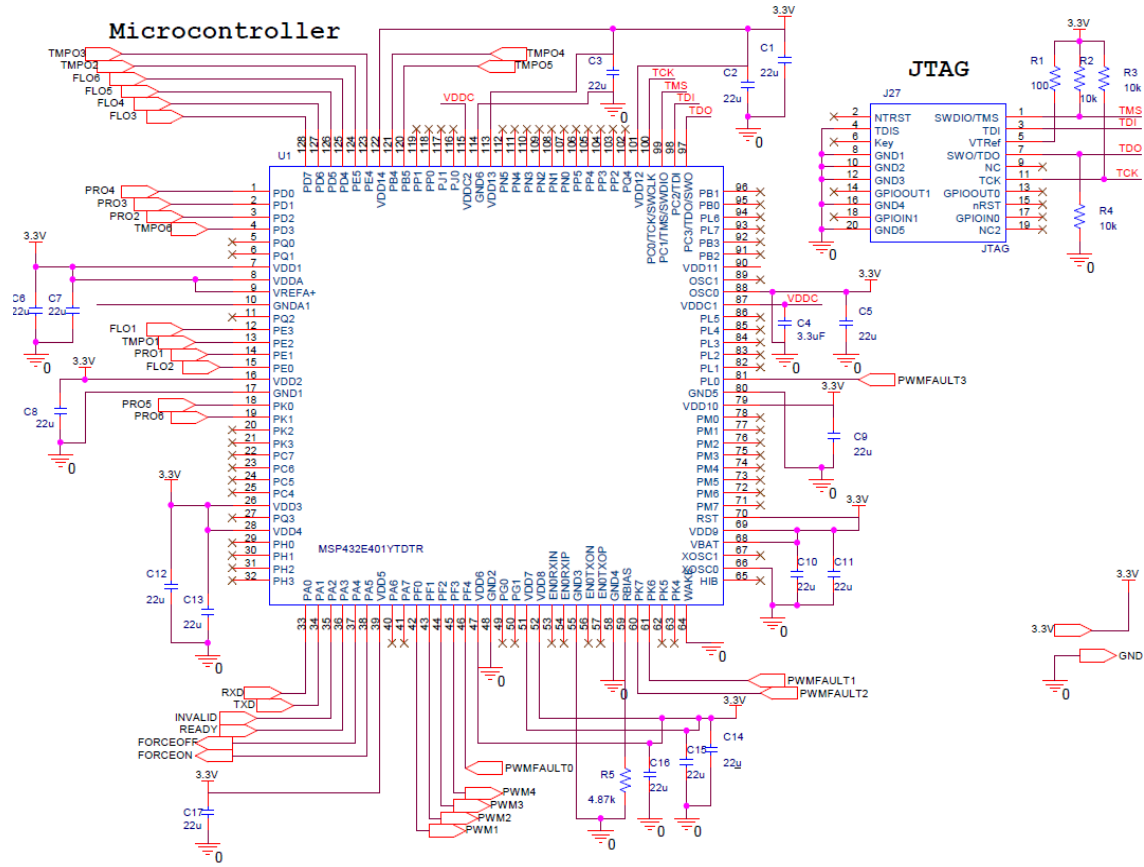


Figure 10: MSP432E401Y Schematic.

3.1.3 Pressure Sensors

The pressure sensor selected is an AD Instruments MLT0670 pressure transducer that is powered by a 5V supply voltage with an internal Wheatstone bridge configuration. So as fluid pressure increases in the transducer, the variable resistors of the Wheatstone bridge increase and decrease opposingly to result in a differential voltage output on the positive and negative terminals of the MLT0670 connector. Pressure tests were conducted on the sensor using a fluid apparatus setup by the Biomedical Department at the Engineering Research Center (ENRC) of

the University of Arkansas where water volume was increased in a vertical pipe with the pressure transducer attached between the bottom of the pipe and a reservoir. The voltage output of the pressure sensor was analyzed regarding pressure values in mmHg, and was determined that the output would range between 3.5mV to 7.5mV for slightly below average (100/70 mmHg) and above average (140/90 mmHg) blood pressure levels for both systolic and diastolic blood pressures. Due to the differential output voltage characteristic of the MLT0670 and low voltage differentials, the INA826 instrumentation amplifier was selected for its high gain capability ranging from 1 to 1000 V/V that is calculated based on the gain resistor selection, allowing for easy manipulation of gain values if necessary. Since the pressures differential voltage has a range of 3.5 to 7.5mV, a large gain factor is needed to amplify the signal. The gain value was determined using equation (16).

$$V_{out} = (V_{in+} - V_{in-}) * Gain + V_{Ref} \quad (16)$$

$$\text{Where } Gain = \left(1 + \frac{49.4k \text{ ohms}}{R_G}\right)$$

The amplifier was designed with a $R_G = 249 \text{ ohms}$, $(V_{in+} - V_{in-}) = 7.5mV$, and $V_{Ref} = 1.024V$ to result in a $Gain \approx 200 \text{ V/V}$ and a $V_{out} \approx 2.52V$, allowing the ADC to function within the voltage range of 1.7V to 2.5V to meet all necessary pressure levels both above and below the expected range. The reference voltage connection was added to the amplifier to offset the amplifiers output by 1.024V and help reduce the necessary gain factor. The INA826 was configured for single-supply operation at 3.3V and ground to limit the amplifier from outputting outside of the microcontroller's voltage range. The reference pin voltage is supplied by the power circuit using the MCP1501T-10E chip to generate a stable 1.024V output signal. The schematic design for the pressure sensor can be seen in Fig. 11 below.

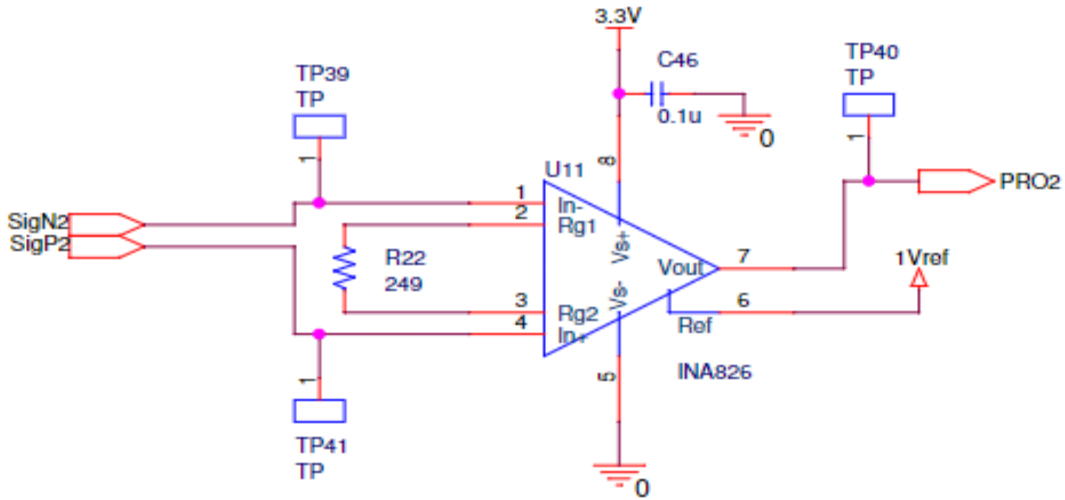


Figure 11: Pressure Sensor Schematic.

3.1.4 Temperature Sensors

The temperature sensor circuit was designed based on Texas Instruments LM34DZ temperature sensor for its small form factor, accuracy to within $\pm 1^\circ\text{F}$, and supply range functioning from 4V to 30V. The resolution of the LM34DZ is $10\text{mV}/^\circ\text{F}$. Since the human body's temperature normally keeps within the range of 90° to 110° , the output from the temperature sensor is within 0.9V to 1.1V. This voltage range was better extended for readability within the ADCs range by using a noninverting op-amp with a Gain = 2 to extend the temperature sensors voltage range to within 1.8V to 2.2V. The LMV358IDGKR dual op amp circuit was selected to function with a single supply of 3.3V and to save space. The gain divider network was calculated using the non-inverting op amp equation (17), where a $R_1 = R_2 = 1\text{k}\Omega$ to result in a Gain = 2.

$$V_{out} = (V_{in+} - V_{in-}) * Gain \quad (17)$$

$$\text{Where } Gain = 1 + \frac{R_2}{R_1}$$

A diode clamp and series resistor were added to protect the input of the op-amp. The temperature sensor circuit can be seen in Fig. 12 below.

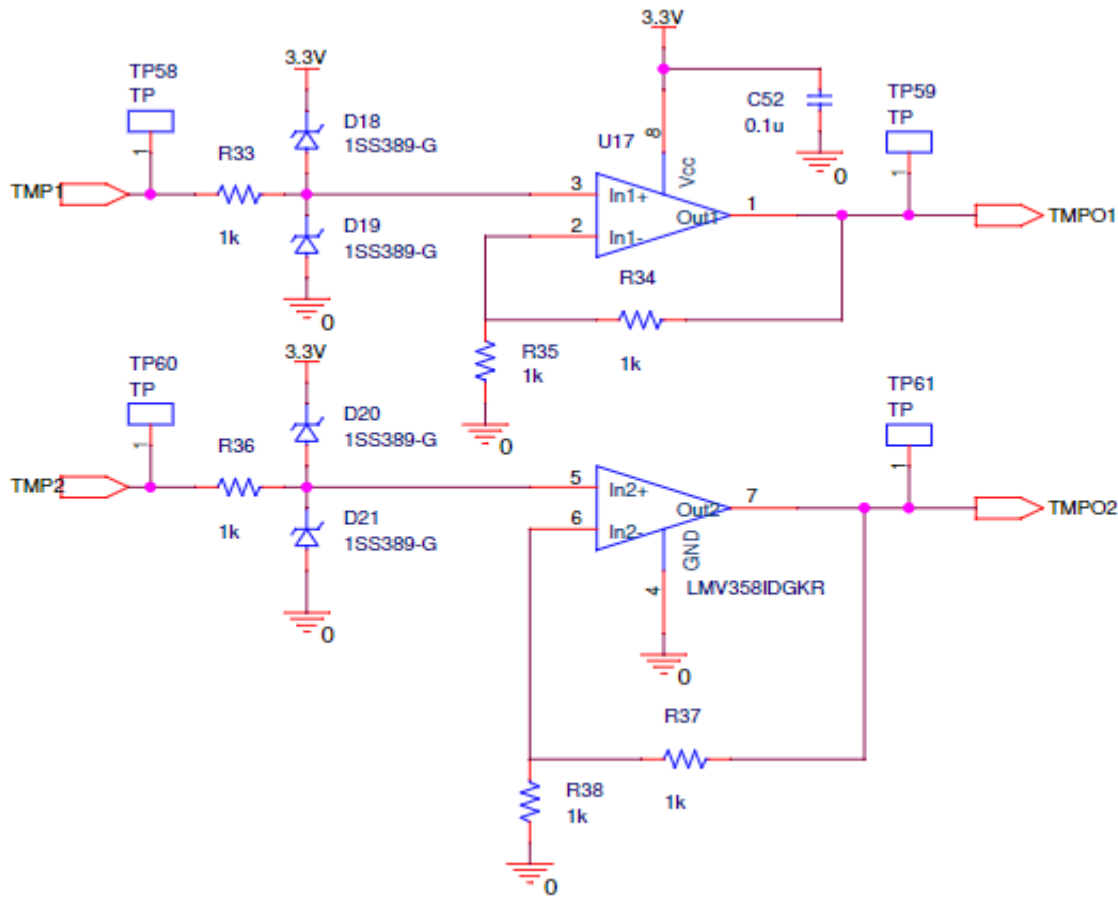


Figure 12: Temperature Sensor Schematic.

3.1.5 Flow Sensors

The flow sensor circuit was designed based on the Renesas Electronics FS1027 Liquid Flow Sensor. The FS1027 is powered by a 5V supply and outputs the flow level between 0 to 10 liters/min as a voltage output ranging from 0 to 4.5V with a sample rate of < 5ms. The flow outputs max voltage was reduced to a 3.3V level using a resistive divider network and a protective diode clamp. The resistive divider was comprised of a 1kΩ series and 2.74kΩ parallel resistor. The divided output is then sent through an OPA4313IPWR quad op-amp circuit that

buffers the output and implements impedance matching to the ADC input. The flow sensor schematic can be seen in Fig. 13.

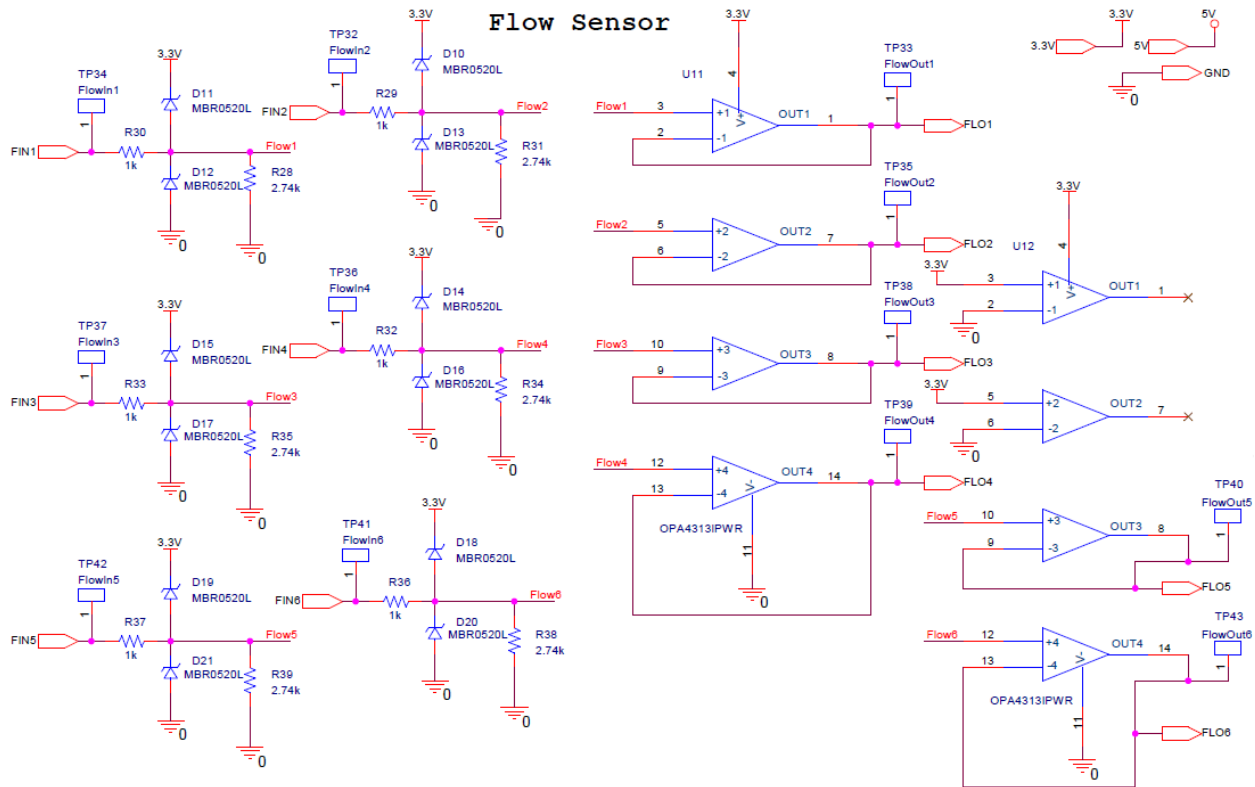


Figure 13: Flow Sensor Schematic.

3.1.6 RS-232 Chip

The RS-232 communication to the MSP432 is conducted through the UART receive port, pin 33, of the microcontroller which requires a 3.3V CMOS logic signal sent in 8-bit packet formats. Due to the voltage levels of the RS-232 communication protocol reaching voltages of $\pm 25V$, an RS-232 line driver/receiver chip is required to convert the RS-232 signal to a 3.3V CMOS TTL logic value. The MAX3227IDBR chip with a single supply voltage of 3.3V was chosen for this task. Following the design documentation in the datasheet, 0.1uF capacitors were placed between the charge-pump voltage-doubler terminals, each charge-pump output, and the supply pin. The FORCEON, FORCEOFF, READY, and INVALID connections of the

MAX3227 were also connected to pins 38 through 35 on the microcontroller respectively as general-purpose input/output (GPIO) ports. The UART receive port, pin 33, and transmit port, pin 34, of the microcontroller were connected to the DOUT and RIN pins of the MAX3227. A DB-9 connector is used to connect the control board to the external communication source, where pins 2 and 3 of the connector correspond to the receive and transmit ports of the MAX3227 chip. The RS-232 Communication Circuit can be seen in Fig. 14.

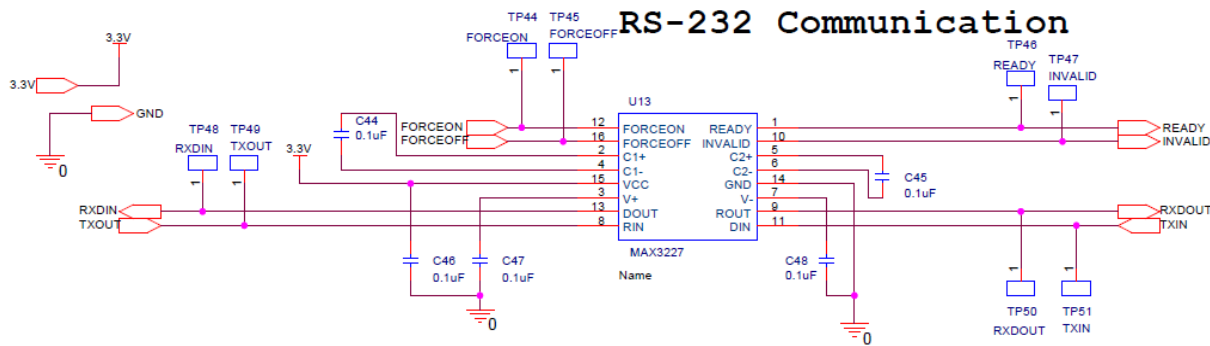


Figure 14: RS-232 Communication Circuit.

3.1.7 Power Circuit

A DC power supply of 15V and 1A is connected to the control board through a 2x5.5mm barrel power jack. The LM2678 switching regulator will convert the 15V supply to a 5V output. A switching regulator was chosen over other step-down circuit options for its high efficiency and low power loss for systems with large input and output voltage differences [9]. The LM2678 was selected for its power capability to function within the expected current output range of ~150mA. The components for the LM2678 buck circuit were chosen based on the recommendations provided in the component datasheet.

The 5V output of the LM2678 will supply power to the enable pin of the gate drivers and all of the pressure, temperature, and flow sensors, as well as serve as the step-down voltage input for the REG103GA-3.3 linear voltage regulator. Since the expected current draw from the

microcontroller and sensor amplifiers is low (<50mA), the REG103GA-3.3 with its low voltage dropout and low complexity was chosen to supply 3.3V to the control board. The final component of the control board power supply is the MCP1501T-10 voltage reference that outputs a 1.024V reference voltage to the pressure sensor circuits INA826 instrumentation amplifier reference pin. Each of these circuits were designed from the recommended configurations in their component datasheets, see Fig. 15.

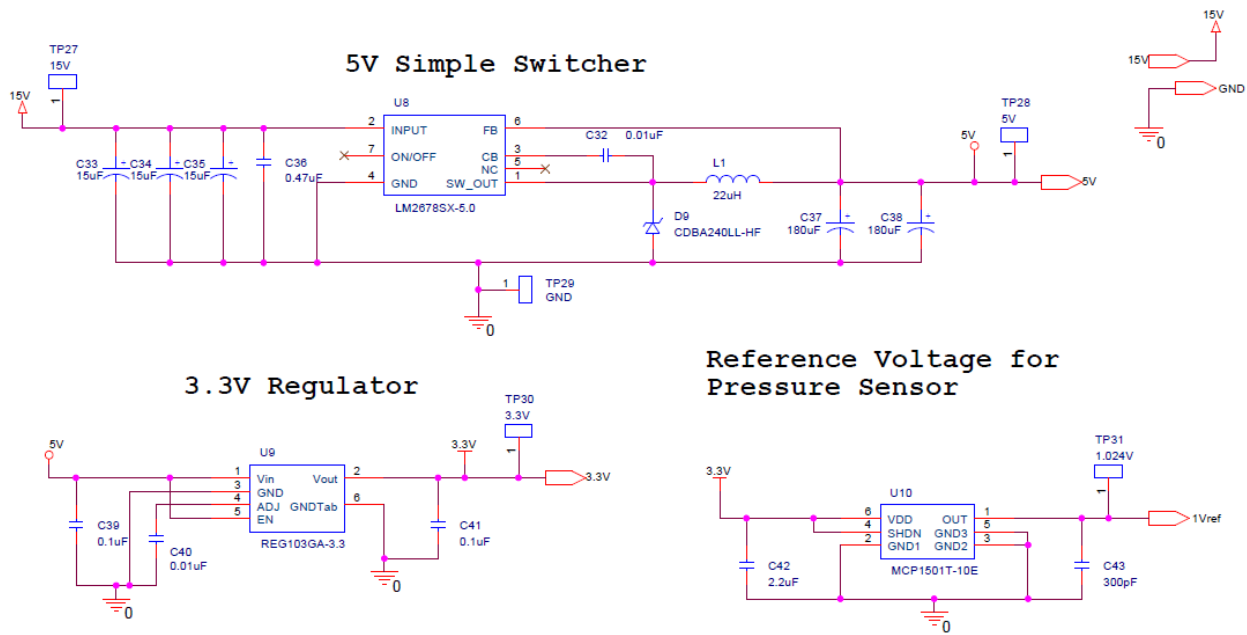


Figure 15: Power Supply Circuit.

3.2 PCB Design

For the PCB Design process, a 4 in. by 6 in. circuit board was constructed in the Allegro PCB Editor v17.2.0 software to accommodate all of the necessary connectors for the 18 varying pump sensors, motor connectors, power supplies, and DB9 RS-232 connectors. The board is separated into 2 sections for the control board and motor driver circuit with a connection between each of their ground planes. These circuits are isolated from one another to help isolate any noise that will be present due to the switching noise of the motor control circuit. The

majority of the component footprints were custom-made using the Allegro Padstack Editor v17.2.0 software along with the PCB Editor's component wizard to make sure all the measurements of padstacks, pin spacing, and component dimensions were matching to the components specifications. The full PCB design is shown in Fig. 16 with labels and outlines marking the different components sections of the circuit board. See Appendix A for an enlarged version of the PCB design.

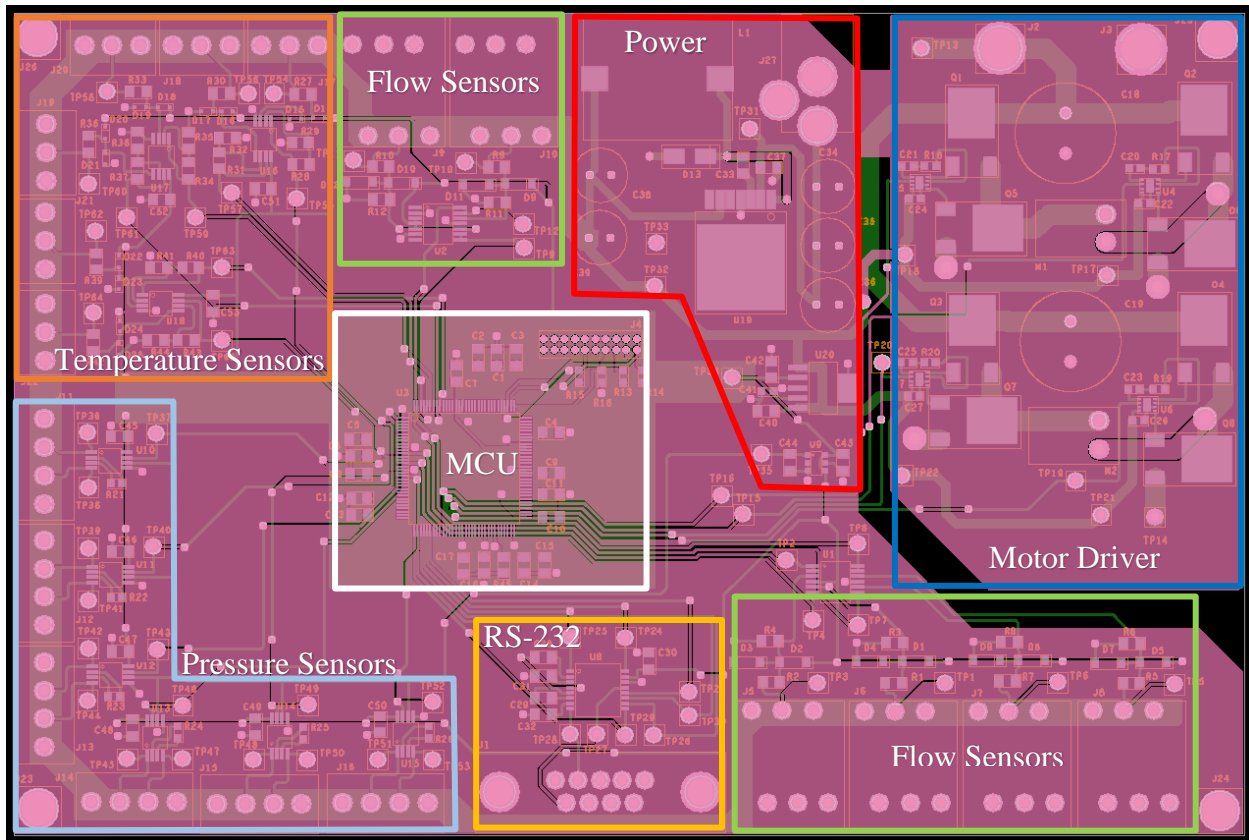


Figure 16: Full PCB Design with labels.

As a ground plane was used to connect all the ground traces for the circuit, bottom traces were used as little as possible to allow for the grounds to have large connections to the power supply ground. This was done by keeping the connections from the microcontroller and sensor circuits as tight as possible, limiting any long connections on the bottom of the board and using

multiple vias to avoid any “islands” becoming present on the bottom copper. An “island” on a PCB design is defined as sections of copper that are part of a large plane which are cutoff due to other trace connections cutting through the copper plane, disconnecting them from their specified net and leaving them as islands on the larger copper. Complex planes for the 3.3V connections to the microcontroller and 5V connections to all the pump sensors were used to provide enough thermal relief and to provide more stable connections. The 3.3V plane can be seen in the center of the board surrounding the microcontroller and 5V plane surrounding the edges of the board to connect to all the sensor terminals. The majority of the copper trace connections within the circuit were 12 mils. With the current of the motor circuit expected to reach 6A on the H-Bridge and outputs, a trace width of 100 mils was chosen to support currents up to 8A on the line. The trace widths for the 5V connections were also increased to 40 mils to be able to support any current draw levels to the control board.

3.3 System Code

The system code was created following the register tables and information provided by the MSP432E401Y datasheet and reference manual for configuring and using the PWMs/GPIOs, ADCs, UART, and Timers of the microcontroller. Timer0 is one of the eight timers available on the MSP432E401Y that allow for the system to increment to a specified count value relative to the microcontroller frequency to function as a time interval (i.e. 16MHz clock with a count of 1.6 million results in a 0.1 second timer interval). The code flow diagram for the system function can be seen in Fig. 17. Once all the required ports and registers for the microcontroller are initialized, the system first functions by waiting for the pressure data profile to be sent via RS-232 communication prior to starting up any functionality for the motor. Upon receiving the profile, the pressure data is loaded into a data array to be used in the PID calculations and the frequency

data loaded into the Timer0 count register. The Timer0 is then enabled, and the timer increments until its count value is reached, generating an interrupt. The system will remain in an infinite while loop until the Timer0 interrupt is enabled, then entering the interrupt sub-routine where the motor is stepped clockwise using switch cases on each Timer0 interrupt. The initial stepping of the motor will run through a “ramp-up” phase where it starts at a 1Hz stepping rate and increases up to the set stepping rate.

When the motor is “ramping-up”, the ADCs and PID controller are disabled using a `step_flag` variable that, while false, will let the motor “ramp-up” to its desired frequency and then set the `step_flag` to enable the ADC reads and PID controller. After the “ramp-up”, the system then waits for the `ADC_flag` to be set following a successful ADC read of the sensor data to enable the PID error calculations and adjust the motors step rate. The ADC read is set to occur when the Timer0 interrupt is triggered, allowing for the ADC to read the sensors during each step interval and then store the data into a data array before setting the `ADC_flag`.

Whether or not an ADC read occurs, the motor will still be running and so the position of the pulsatile pumps shaft needs to be monitored to determine when a direction change needs to occur in the motor states. When the pulsatile pump is approaching the pump cap, the direction is reversed for the motor by following the Table 1 states in the reverse flow from the current clockwise or counterclockwise orientation. Lastly, after the shaft position is checked for a direction change occurrence, the newly calculated, or recurring, motor step rate is reloaded into the Timer0 count register to change the Timer0’s interrupt interval and then the Timer0 interrupt is cleared, and the process is reset to await another interrupt occurrence.

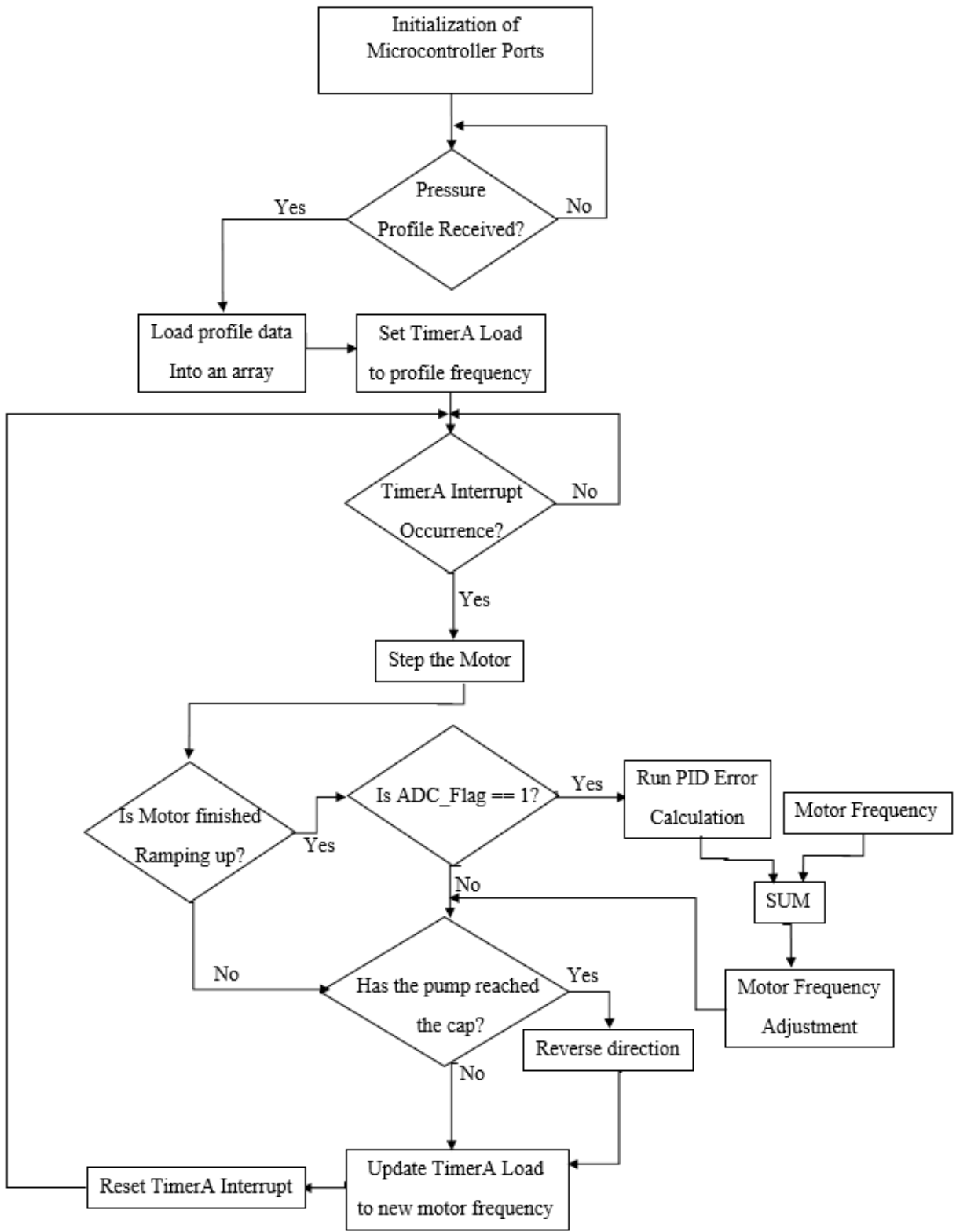


Figure 17: Code flow diagram.

Chapter 4: Test Results

4.1 Control Board Testing

The fabrication and testing of the control board was done using a Fluke Digital Multimeter, a Keysight InfiniiVision MSIX2024A Oscilloscope, and DP832 Programmable DC Power Supply. Extensive resistivity tests were done while soldering each circuit section of the board to check for misconnections or short circuits prior to running voltage tests.

Two errors were discovered in the 20pin JTAG footprint when testing the microcontroller circuit due to the pinout diagram of the XDS110 Debug Probe being misread. The first error in the footprint was in the row pitch being set to 50 mils rather than 100 mils, causing the XDS110 probe to not connect directly. This error was resolved by using a breadboard with wires and an additional debugger probe with a 20pin female connector with correct pitching. The second error on the JTAG was the pinout connections to the debugger being inverted on the row connections. This was resolved by soldering the through-hole JTAG connector to the bottom side of the board and the debugger probe connected on the underside of the board to have the correct pinouts between the debugger and JTAG connector.

Following the microcontroller testing, the pressure, temperature, and flow sensor circuits were soldered and tested individually using a multimeter and the ADC channels on the microcontroller. The temperature and flow sensor circuits were tested by attaching the LM34DZ temperature sensor to the necessary pins on the terminal blocks and watching that the output changed as the temperature changed. The temperature sensor was used for the flow circuit testing rather than an Analog Signal Generator due to the output impedance of the generator and the flow circuits resistive divider resulting in a low current input to the OPA4313IPWR op-amp that was unable to be read. Since the MLT0670 pressure sensor outputs through a Wheatstone bridge

circuit, the pressure circuit was tested using two DC power supplies connected to the positive and negative input terminals to the INA826 instrumentation amplifier. These DC power supplies were set to varying voltage levels that would sum to 5V (i.e. 2.5V and 2.5V, 2.6V and 2.4V, 2.7V and 2.3V, etc.) and the output signal measured through both the multimeter and ADC inputs of the microcontroller.

The RS-232 communication circuit was then tested using a makeshift pressure profile in Code Composer Studio and sending each set of pressure data in 8-bit packets. The UART communication receive was then tested by looping the transmission output pin of the DB9 header back to the receiver pin and analyzing the outputs using the Keysight oscilloscopes serial communication analysis. The oscilloscope was set to RS-232 communication in the Serial options with a baud rate of 9600, 8-bit word length, one stop bit, and one parity bit for the signal configurations. The scope waveforms for the UART receive input and transmit output can be seen in Fig. 18 and 19, and the converted RS-232 transmit output compared to the CMOS transmit output in Fig. 20.

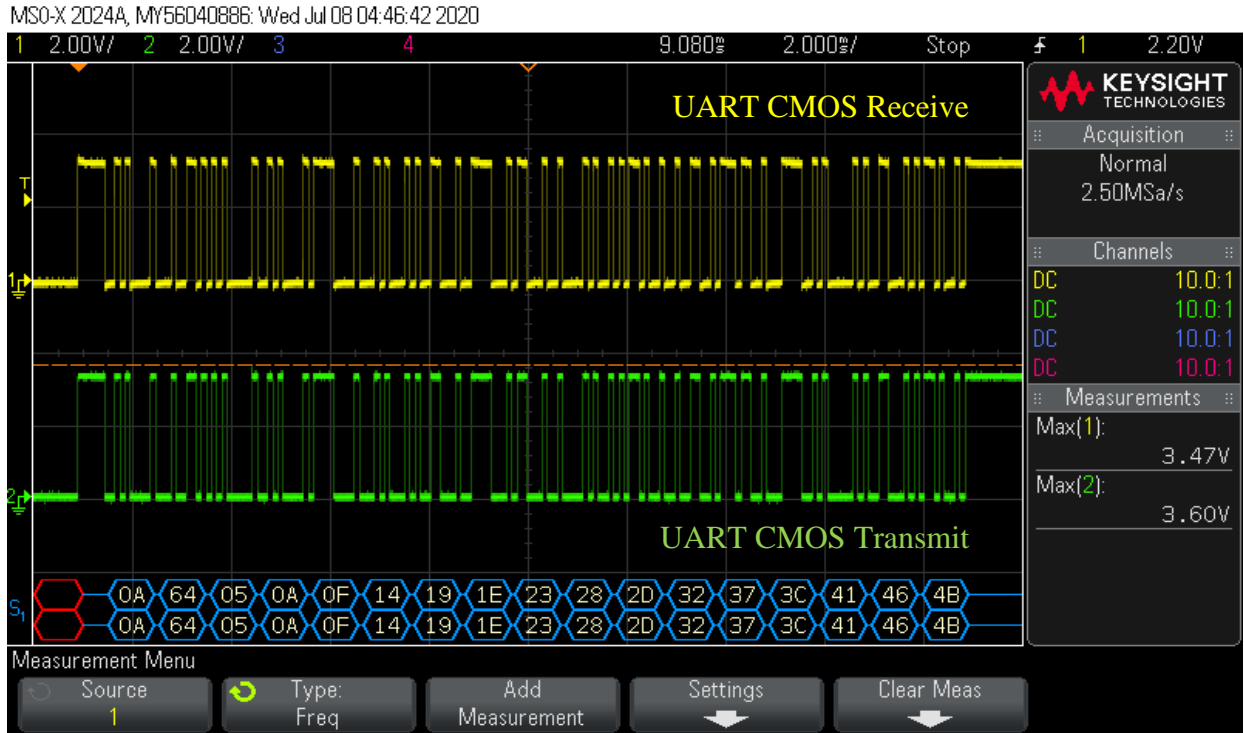


Figure 18: UART transmit and receive.

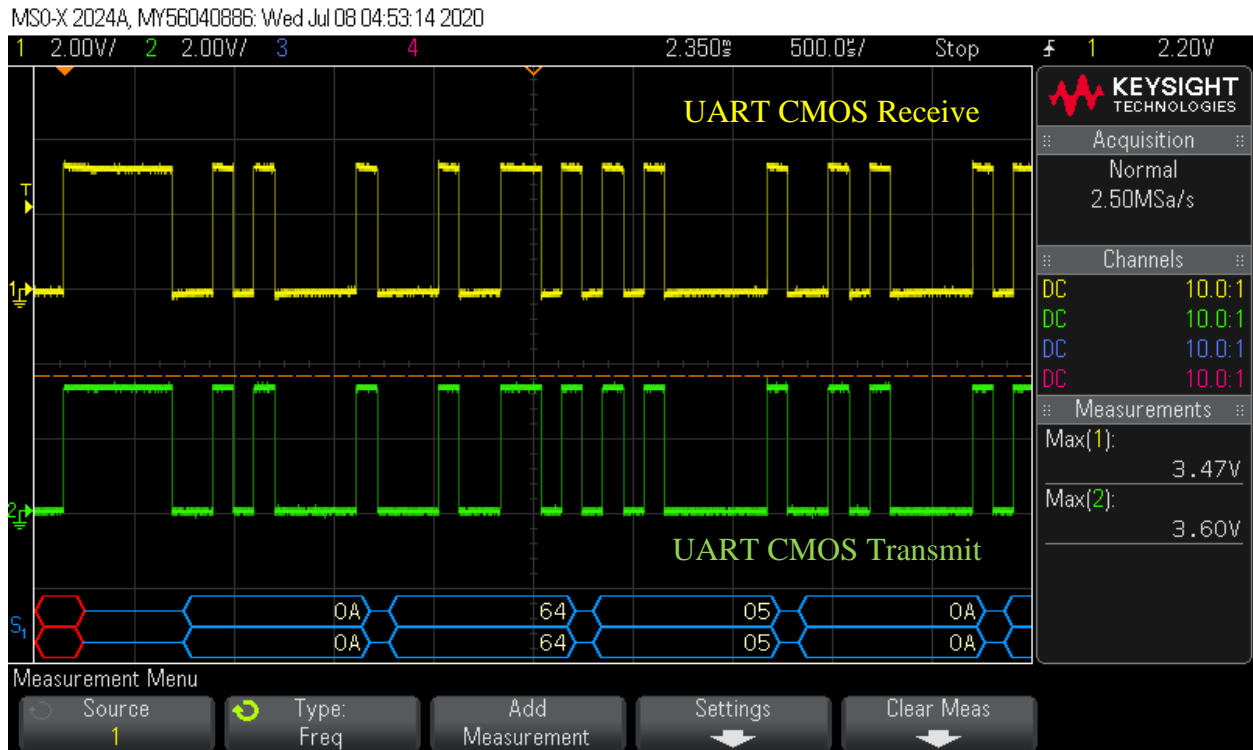


Figure 19: UART transmit and receive zoom-in.

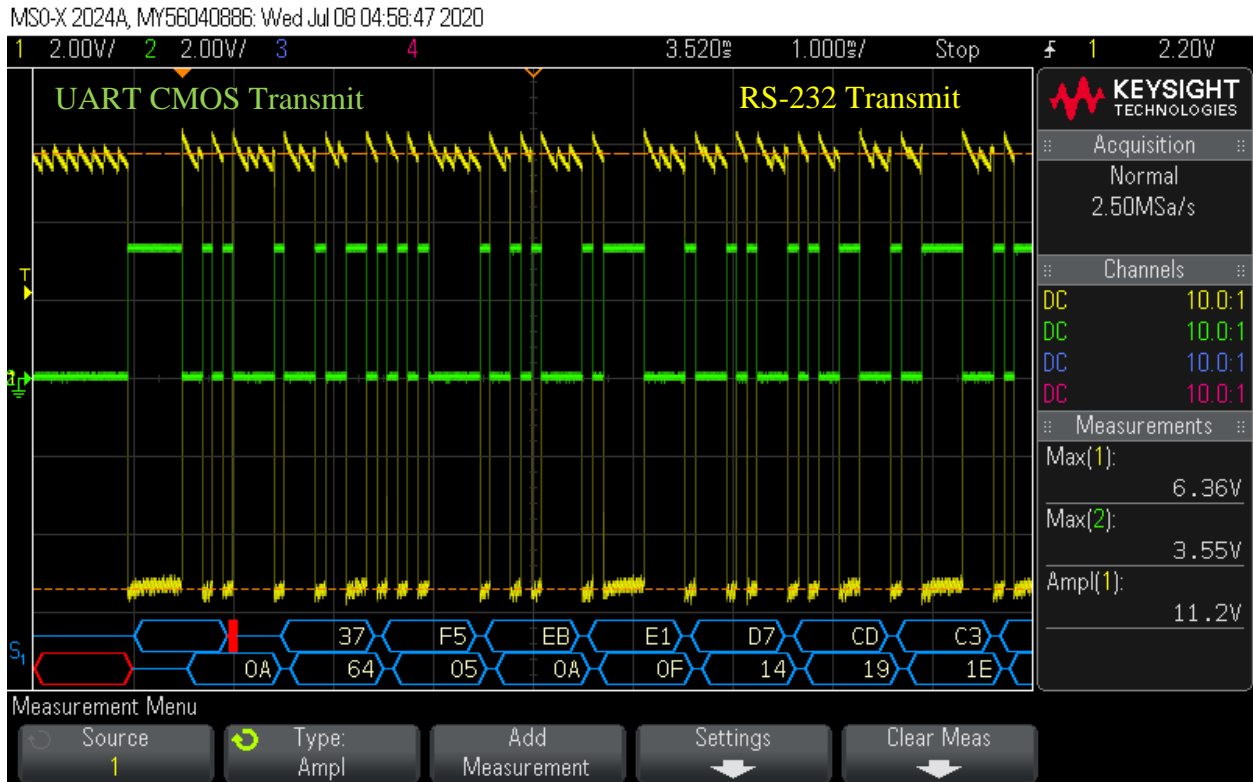


Figure 20: RS-232 transmit vs. CMOS transmit signals.

The MAX3227 RS-232 chip converts the received serial signal from an inverted bipolar 5V profile to the 3.3V CMOS logic while still retaining the profile information in its 8-bit packet format. The data output from the UART seen in Fig. 19 shows that the data is sent out least-significant bit (LSB) first and with an idle high polarity. The UART data can be read from this signal by an active low start bit signaling the start of the packet, the eight data bits, and an active high stop bit. Finally, following the testing of all the control board circuitry, the motor driver circuit was assembled and tested with design issues being discovered through the testing process.

4.2 Motor Circuit Modifications and Results

The original motor driver circuit, see Fig. 21, was designed with the TPS51604 half-bridge gate driver controlling the IRLR120 power MOSFETs. The TPS51604 was tested but was found to be functioning below expectations. This is due to the gate driver only being capable of

boosting the gate voltage to 35V, which is well below the rated 68-72V supplied to the motor. Thus, the gate driver circuit was redesigned to that shown in Fig. 9 with the replacement of the TPS51604 with the ISL78434 half-bridge gate drivers.

A separate miniature PCB board was constructed with this new gate driver and its circuitry to sit on top of the already constructed PCB using through-hole wires and board-to-board connections, see Fig. 22,

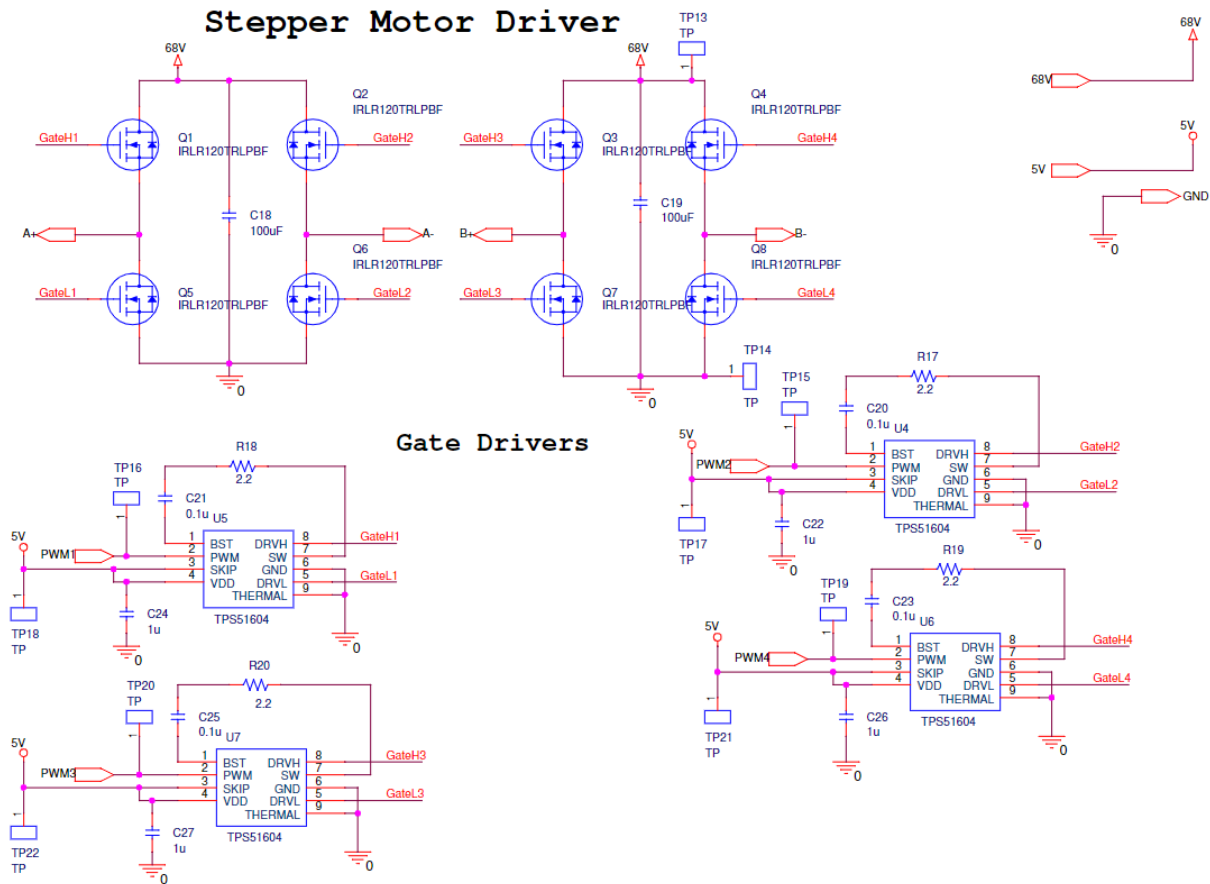


Figure 21: Original gate driver schematic.

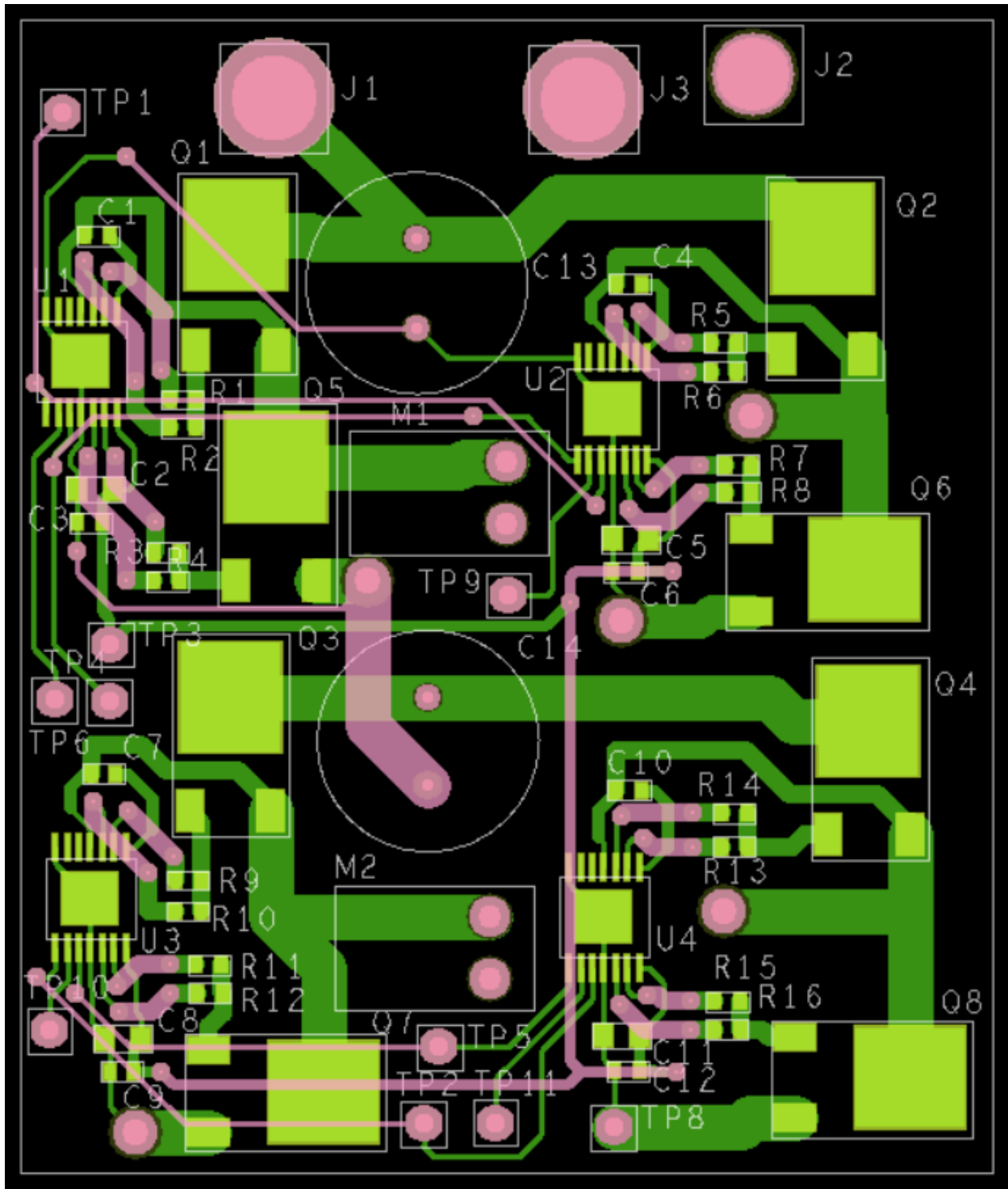


Figure 22: Gate Driver Breakout PCB Design.

The new breakout board was milled using the University of Arkansas milling machine so precise care was taken when sizing the padstacks of the component connections. Small test mills

were run on the board with the various drill holes, solder-mount pads, and vias to make certain there were suitable trace connections being made on the milled PCB. The milled, soldered, and mounted breakout board to the control board can be seen in Fig. 23.

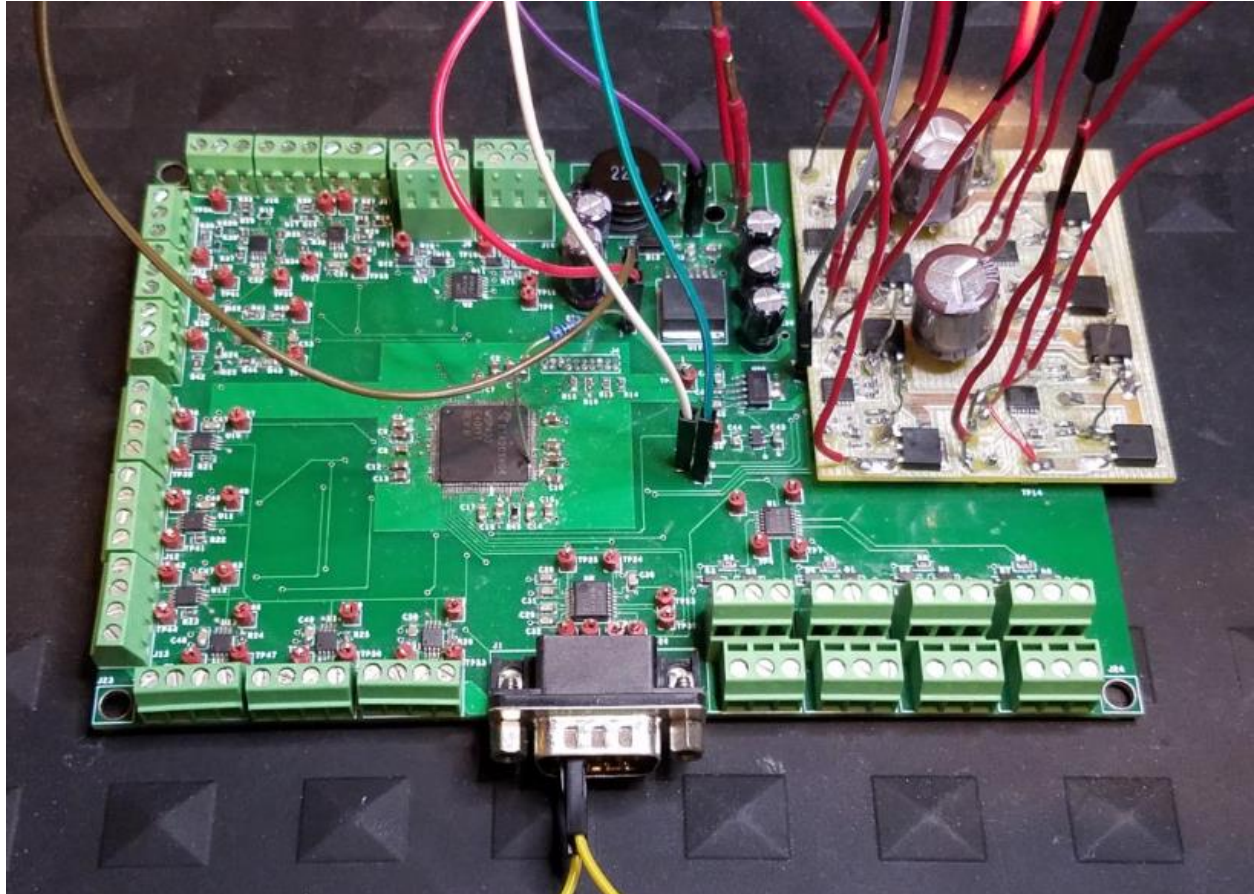


Figure 23: Modified PCB board.

The connections between the control board and breakout board that could not be done through direct solder connections were connected using female-to-female wire plugs between the test point connections of the boards. Extensive testing was done on the modified system at varying stepping rates to determine if the system was functioning correctly. The pulse-width modulated (PWM) signals supplied to the independent driver inputs were created using the MSP432's GPIO pins. The GPIOs were configured to match the stepping sequence in Table 1 by

setting a timer interrupt that triggered at the set motor frequency to increment through the motor states by setting the corresponding GPIO bit. Each half-bridge of the H-bridge was supplied with two PWM signals, one to the high-side and one to the low-side MOSFET, that were inverted on the opposite half-bridge, see example drawing in Fig. 24.

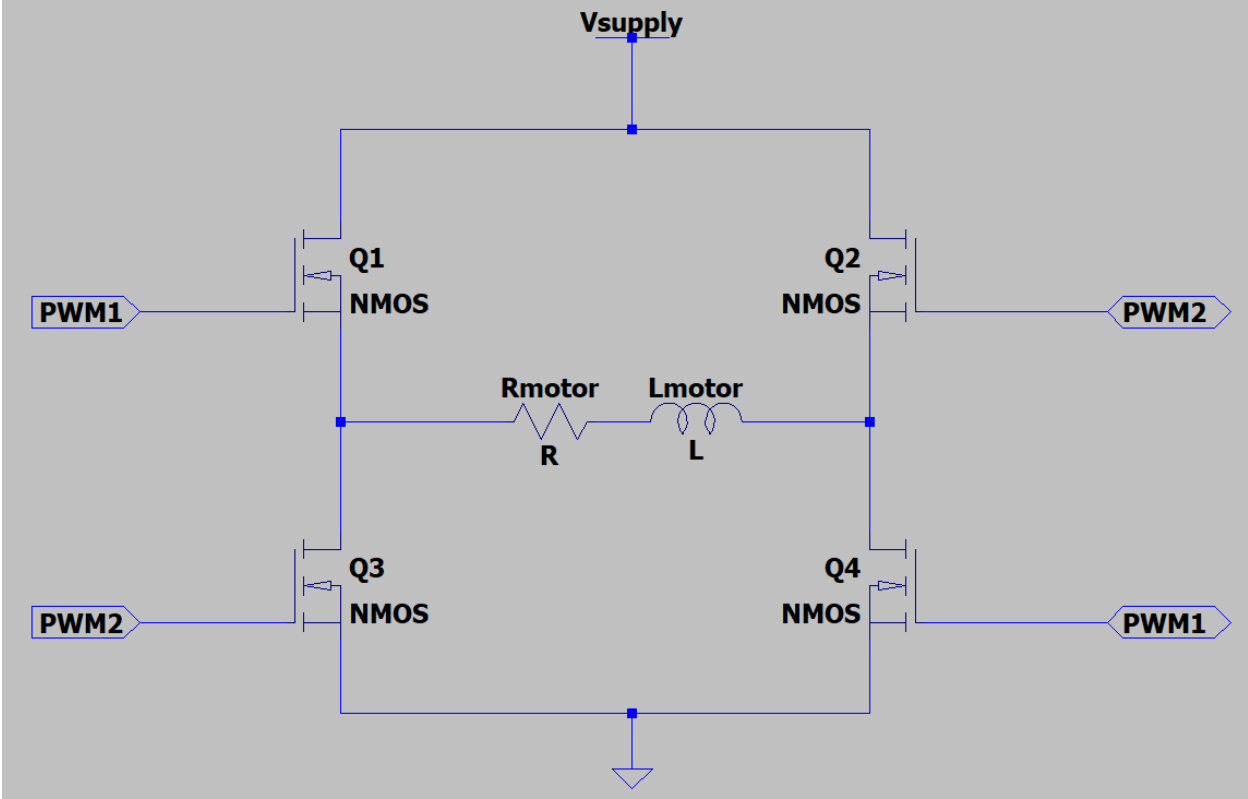


Figure 24: PWM input configuration to H-Bridge.

The signals were connected in this manner to the HI and LI of the ISL78434 gate drivers, corresponding to the high-side gate output and low-side gate outputs respectively, so that no shoot-through was possible when running the motor. During each test of the motor circuit, an initial start-up phase was initiated to step the motor from 1Hz up to the desired frequency interval to both allow the motor to warm-up and to prevent any possible issues from starting the motor at a high frequency rate at the start.

A supply voltage of 20V and 3A current was supplied to the motor circuit, and tests were conducted on the system at varying frequencies. The tests were to be conducted in a frequency range of 1Hz to 2kHz to allow the motor to run up to 10 rotations-per-second as the maximum speed and to analyze the motor circuits output characteristics. Unfortunately, through testing the motor at a 10Hz stepping rate, it was found that although 20V was supplied to the motor, only a 1.5V phase voltage was being supplied to the windings and the full 3A of current was being drawn, see Fig. 25.

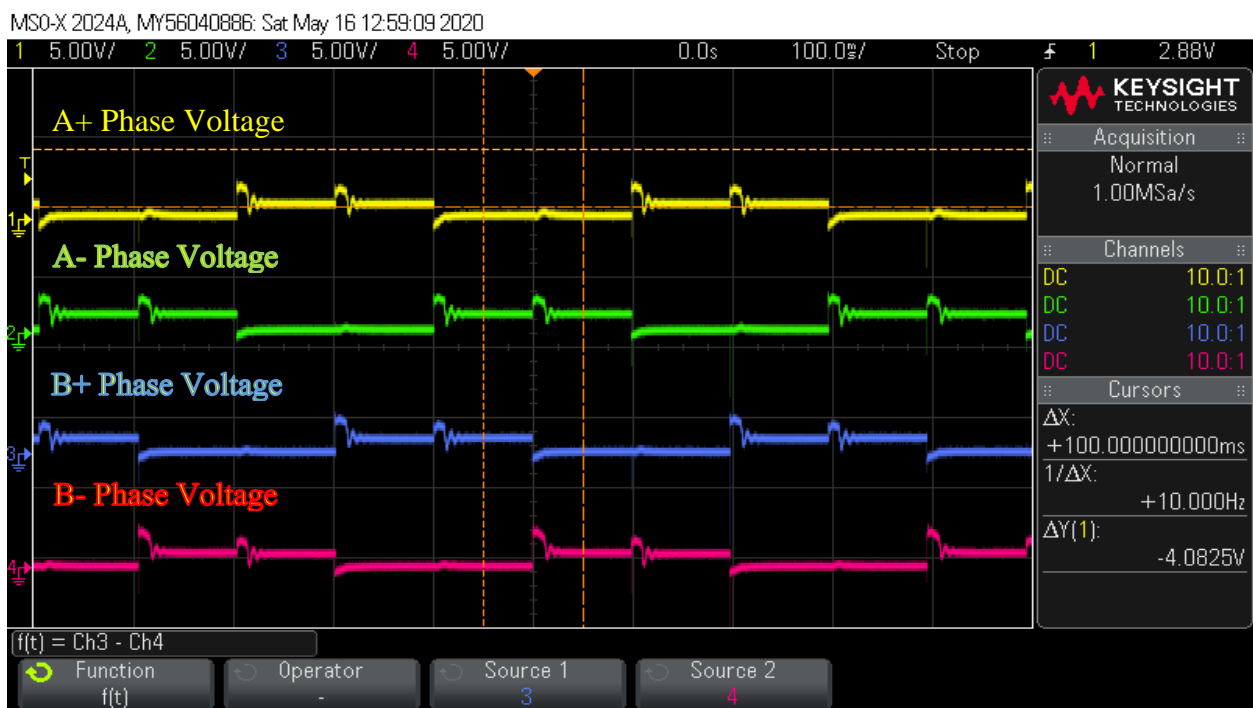


Figure 25: Stepper motor outputs at 10Hz stepping rate.

The PL34HD0L8500 has a winding resistance of 0.5Ω and winding inductance of 3.6mH when connected in series. So in the low frequency range the impedance of the motors inductive windings is equal to 0.5Ω , and with 3A of current being drawn from the supply, through Ohms Law a voltage of 1.5V will be supplied to the motor winding. With this identified, the motor voltage was decreased from 20V down to 3V to prevent damage to the motor from generating

too much power/heat. The next test conducted was at a stepping rate of 100Hz, where the power supply was found to supply 3.4V at 1.5A of current draw and at 150Hz, 3.5V was supplied with 0.57A of current draw. The phase voltages for the 100Hz and 150Hz tests with a power supply of 3V and 3A can be seen in Fig. 26 and 27 respectively.

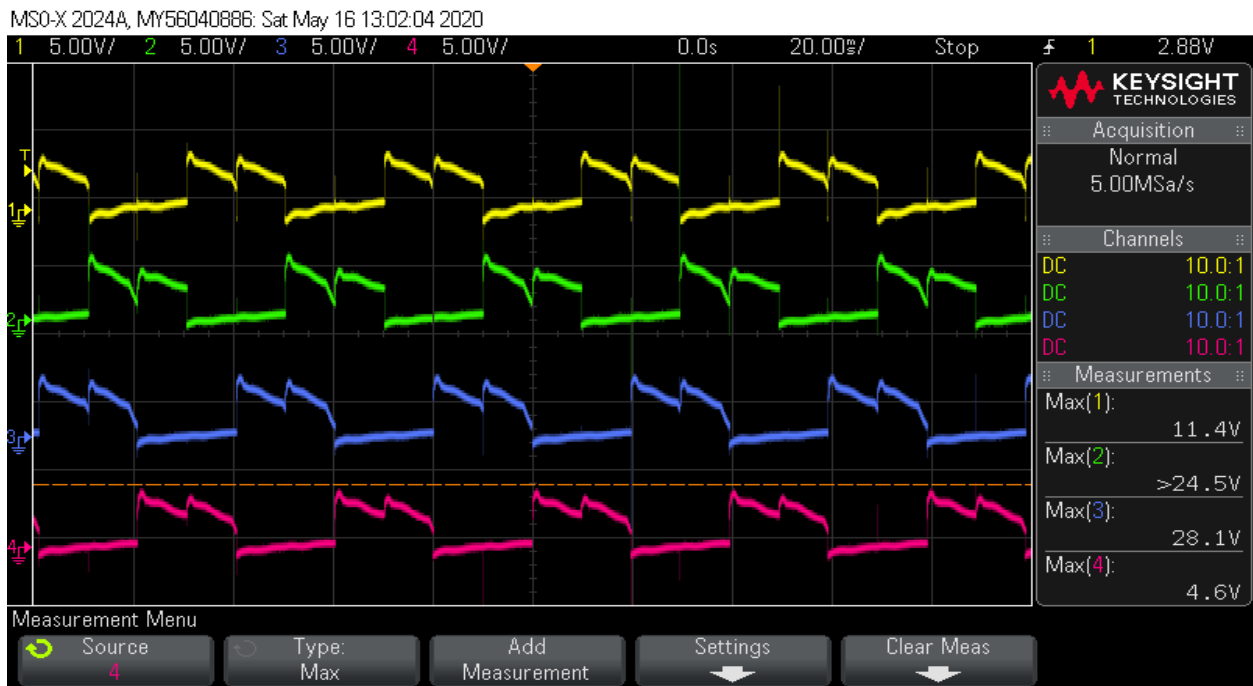


Figure 26: Stepper motor outputs at 100Hz stepping rate.

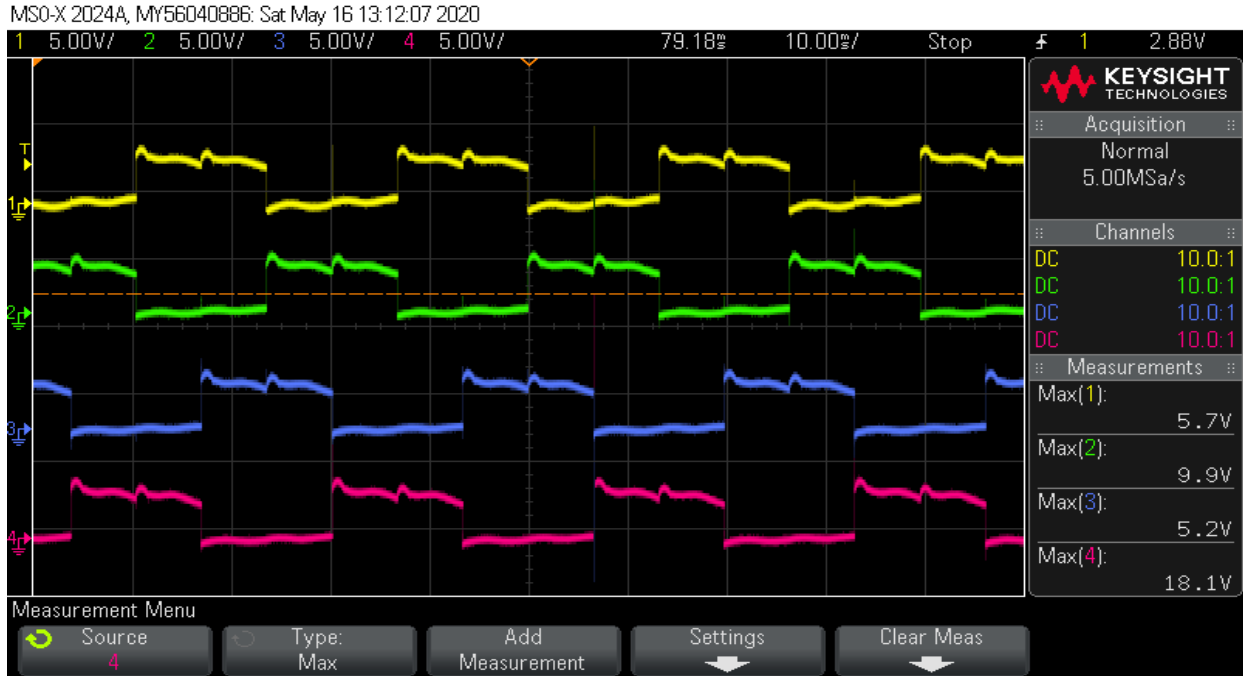


Figure 27: Stepper motor outputs at 150Hz stepping rate.

From analyzing the phase voltages of the 100Hz and 150Hz tests compared to those of the 10Hz tests, it can be seen that as the frequency is increased, the initial peak interval occurring at each step on the 10Hz waveforms is shortening, and the more this interval shortens, the lower the current draw from the motor. This interval is the charging interval of the stepper motor windings determined by the L/R time constant calculated with the winding resistance and inductance.

$$\text{Motor Time Constant } (\tau) = \frac{L_{\text{winding}}}{R_{\text{winding}}} = \frac{3.6 \text{ mH}}{0.5 \Omega} = 7.2 \text{ ms} \quad (18)$$

For the windings to reach full charge, an interval of $5 * \tau$ is required at the rated voltage and current levels which results in an interval of 36 milliseconds to fully energize each winding. With this, when the motor is stepped faster than 27Hz, the motor cannot fully energize its windings and therefore will not generate its full torque, resulting in the circuit being unable to meet the torque and speed requirements. Modifications can be made to the L/R of the motor to transform it into an L/nR configuration by adding additional series resistance to the winding.

This would allow a higher supply voltage to be applied, but would result in significant power and heat issues, so is not an efficient solution. [11] Therefore, the conversion of the motor driver circuit to a constant current driver needs to be done to be able to function at high stepping rates and higher voltage supplies.

4.3 Chopper Driver Design and Simulations

A chopper driver is a very efficient solution to implement a constant current driver to a stepper motor. The chopper driver functions by supplying a higher voltage level to the stepper motor than its rated voltage to bring the current up to its rated level very quickly. This relationship is best seen in the voltage equation for an inductor, see equation (19).

$$V_L = L \frac{di_L}{dt} \quad (19)$$

If equation (20) is reordered to solve for the rate of change in current.

$$\frac{di_L}{dt} = \frac{V_L}{L} \quad (20)$$

Then it can be seen that as the voltage of the inductor is increased, the rate of change of the current also increases, resulting in the rise time of the current through the winding to increase and energize the winding faster. Allowing for the L/R charge time to be reduced without having to change the time constant itself.

When the current of the chopper driver reaches its rated current, the driver “chops” the voltage, switches the MOSFETs off, allowing the current in the motor to slowly discharge from its rated level until it drops below a threshold where the MOSFETs are then turned back on. This chopping of the voltage keeps the current of the motor at its rated level, allowing for a high torque output and for the system to function as a constant current motor driver. The current motor driver design can be converted into a chopper driver by implementing an overcurrent

detector between the H-Bridge and ground. The overcurrent detector is designed with a low-impedance and high wattage sense resistor between the low-side MOSFET sources and ground, and a voltage comparator with a reference voltage and resistive divider to set the compared threshold voltage. The new design for the H-bridge with overcurrent detector can be seen in Fig. 28 with the updated PCB design in Fig. 29. [11] See Appendix A for an enlarged version of the updated PCB design.

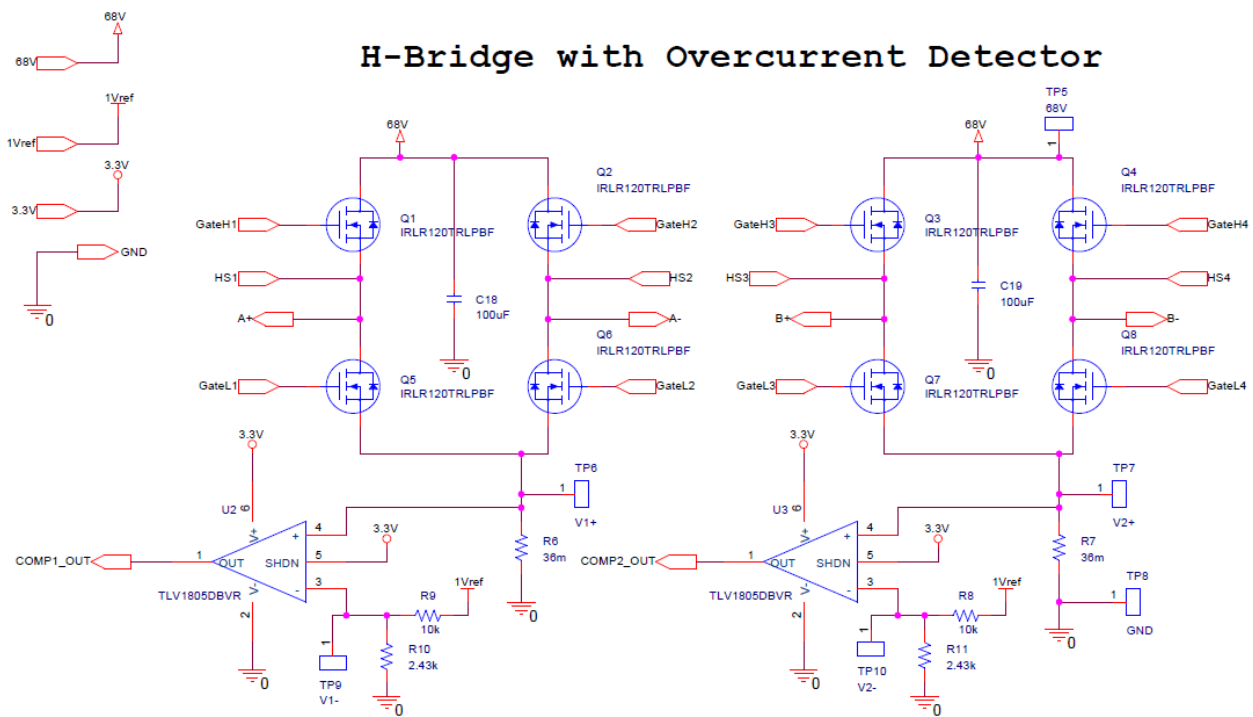


Figure 28: H-Bridge with overcurrent detector schematic.

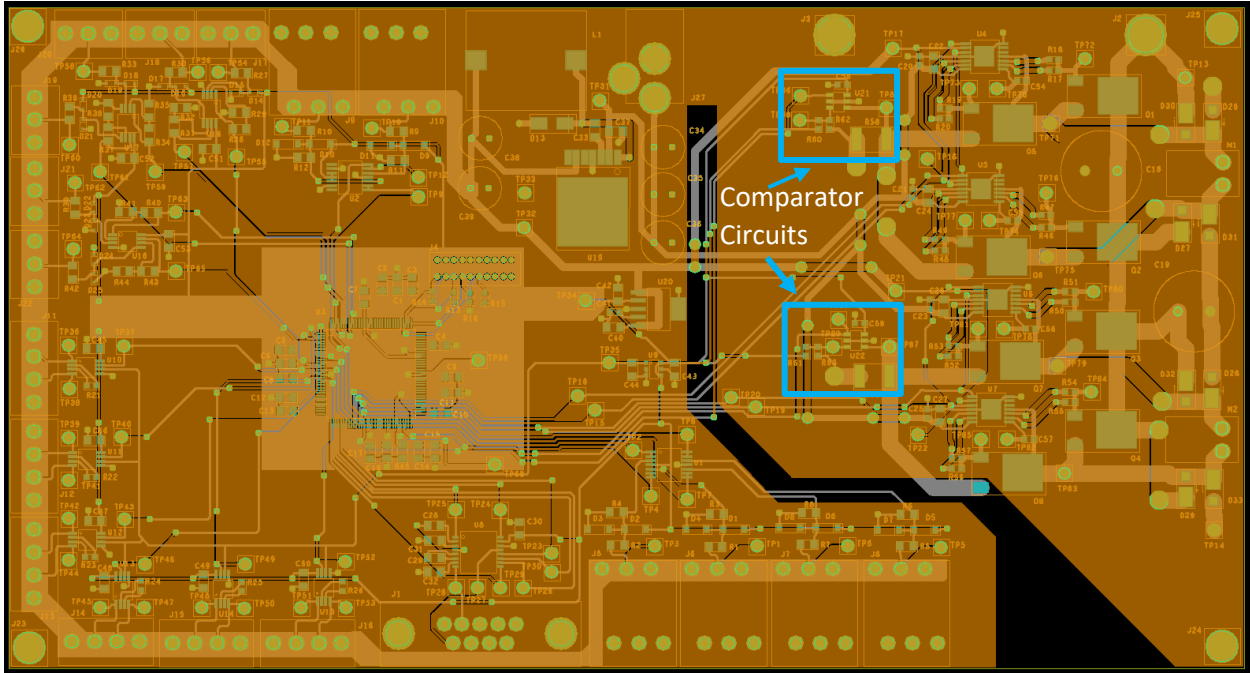


Figure 29: Updated PCB design with chopper driver.

The PCB boards size was increased by approximately 1 inch in length when analyzing the circuit board and breakout board setup, resulting in a board dimension of 7.5 inches by 4 inches. The updated PCB with chopper driver follows the same configuration setup as the original PCB shown back in Fig. 16, but with the orientation of the H-Bridges and gate drivers modified as shown in Fig. 29 to support the chopper driver connections to the comparator.

A TLV1805 general-purpose comparator was chosen to function with a 3.3V voltage supply and for its low input offset voltage of $\pm 0.5\text{mV}$. A $36\text{m}\Omega$ 2W sense resistor was selected to support a current of 6A and hold a residual voltage of 0.216V to the positive input of the comparator if the 6A current was reached. The negative rail of the comparator was supplied with the reference source voltage of 1.024V through a resistive divider made up of a $10\text{k}\Omega$ resistor in parallel with a $2.43\text{k}\Omega$ resistor to result in a voltage of 0.2001V on the negative input. Since the 0.216V at the positive input corresponds to 6A, the threshold current for the H-bridge will be:

$$I_{th} = \frac{V_-}{V_+} I_{rated} = \frac{0.2001 V}{0.216 V} (6 A) = 5.558 A \quad (21)$$

So when the current is charged above the threshold current of 5.558A, the comparator will output high (3.3V) to the MSP432's PWM fault pins that will be configured through an interrupt to chop the voltage when the current threshold is met. The fully compiled materials list for the updated PCB Design can be found in Appendix B.

A simulation of the entire pump system was also constructed in MATLAB Simulink with the addition of the chopper driver to the H-bridge circuit. The constructed Simulink model can be seen in Fig. 30 that uses Simscape Electronics, Rotational, Translational, and Hydraulic systems to be able to construct and simulate all sections of the pump apparatus.

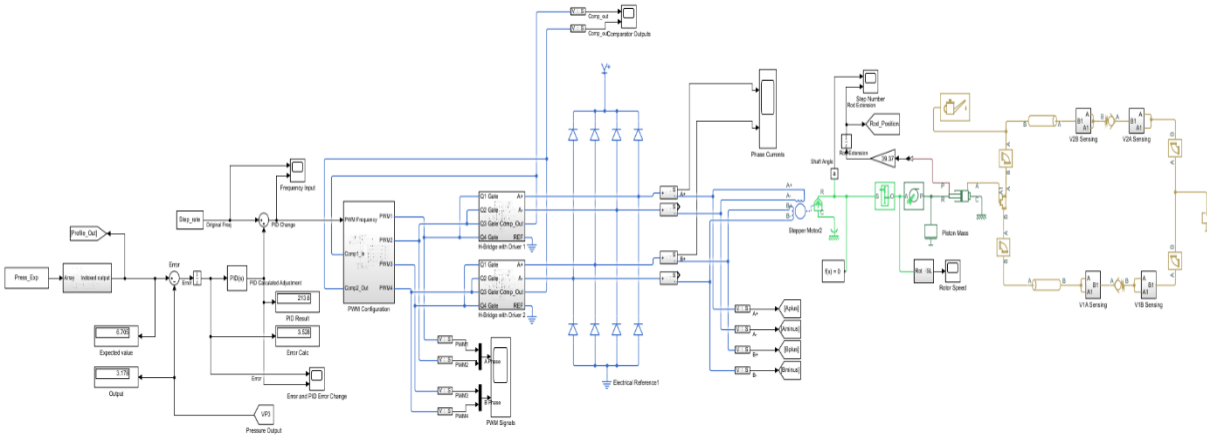


Figure 30: Pulsatile Pump Simulink Model.

The system was constructed following the same design rules as stated earlier with a PID controller that calculates the error between a desired pressure profile and the sensed pressure output to change the motors stepping rate. To be able to generate PWM signals with a modifiable frequency input, a PWM signal generator was constructed to create all four PWM profiles for the stepper motor with implemented direction change, see Fig. 31. The PWM outputs at a 500 steps/sec stepping rate can be seen in Fig. 32.

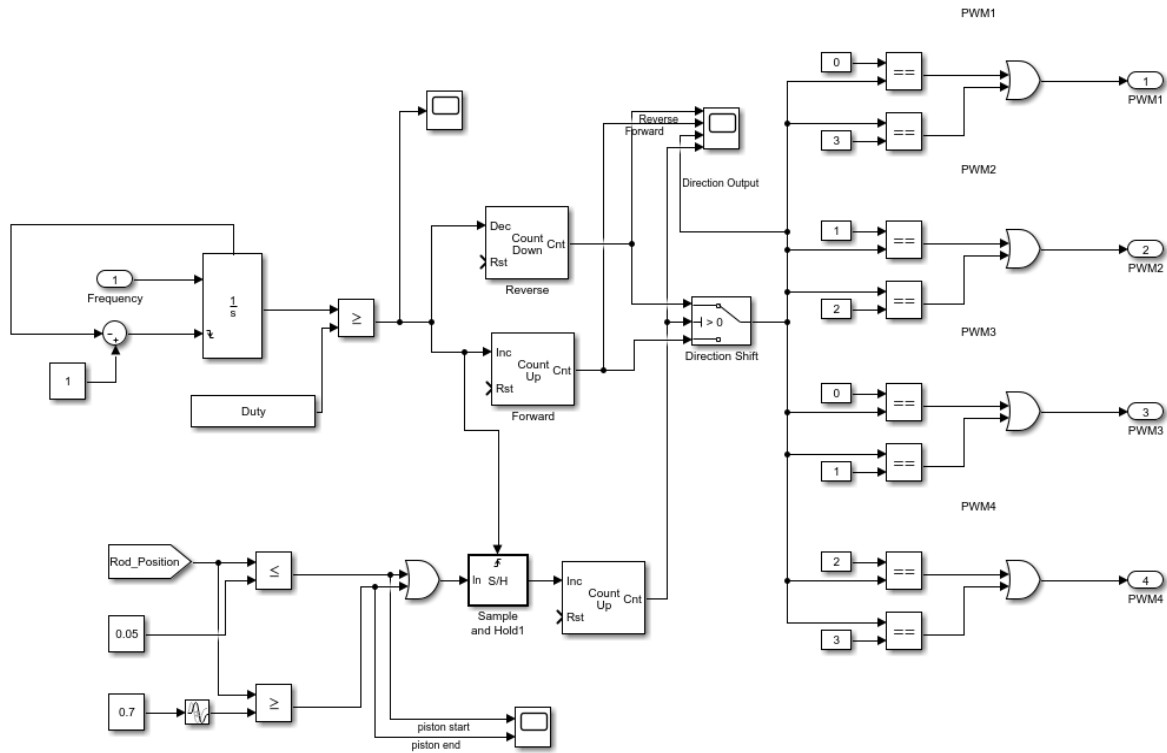


Figure 31: PWM Generation Subsystem.

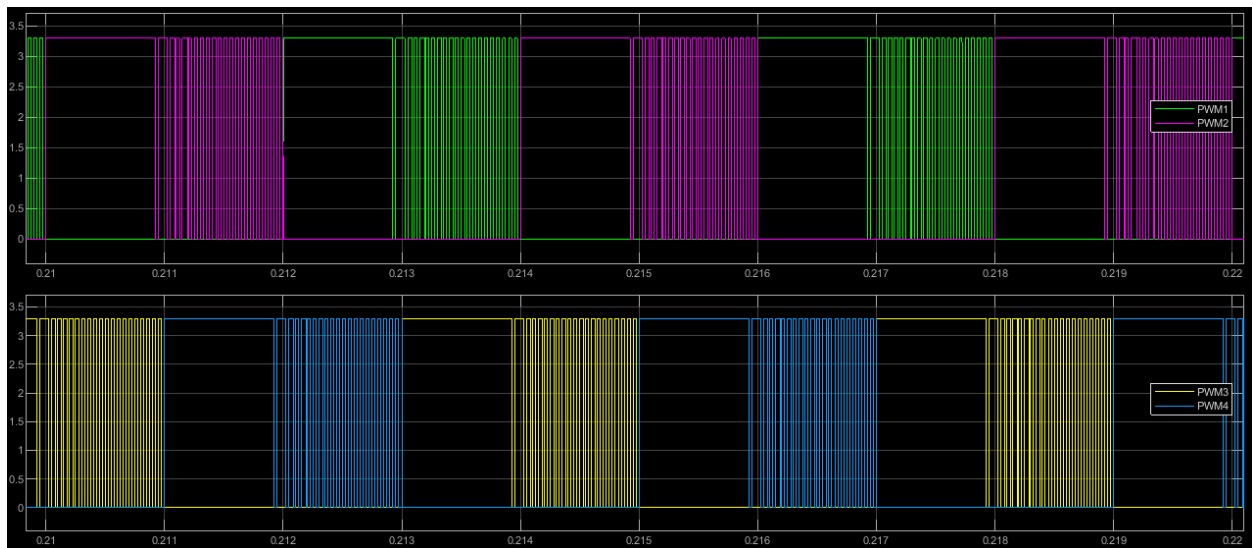


Figure 32: PWM outputs at 500 steps/sec.

The H-bridge model was designed according to Fig. 28 with the overcurrent detector to generate a constant current of 6A to the stepper motor. Since implementing the chopper driver

with output to a microcontrollers fault pins proved to be difficult, the comparators output was connected through a SR Flip-Flop block where the comparator was connected to the set (S) terminal and a 20kHz 3.3V clock connected to the reset (R) terminal. The flip-flops inverted output, \bar{Q} , was then connected to a two-input AND gate along with the H-bridge PWM signals, see Fig. 33. Following the logic of an SR Flip-Flop shown in Table 3, whenever the comparator outputs high, the flip-flop supplies a chopped signal to its H-bridge's PWM inputs, that feed into the gate drivers of the H-bridge to switch the MOSFET gates on and off.

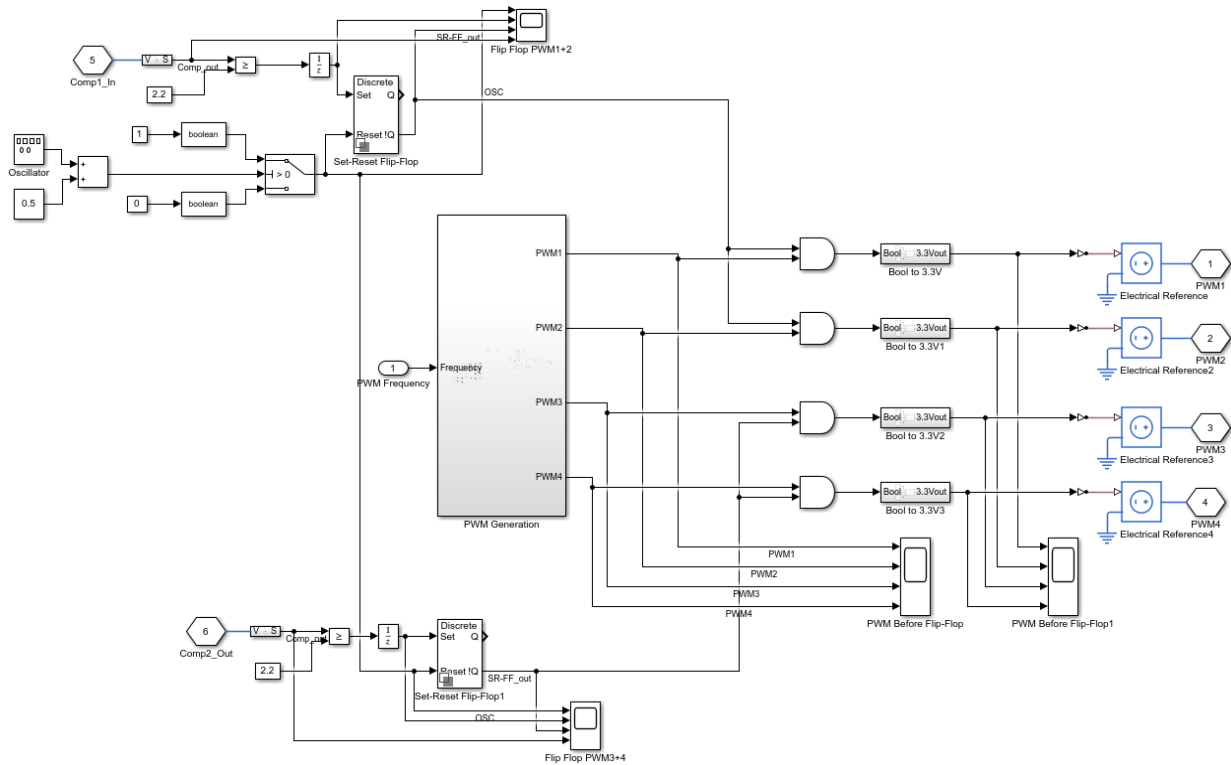


Figure 33: Chopper driver to PWM model.

Table 3: SR flip-flop logic.

S	R	Q	\bar{Q}
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

The hydraulic, rotational, and translational systems were designed and configured with the help of Sam Stephens to simulate the designed in-house mechanical pump apparatus. A gear box block with a ratio of 1:3.288 and a wheel and axle block with a wheel radius of 0.203 inches were connected between the stepper motor rotor and hydraulic cylinder piston to convert the motors rotational stepping into a translational piston movement. The hydraulic cylinder was configured to have a piston area of $\sim 7 \text{ in}^2$ and piston stroke of 0.75 inches to connect through a T-Junction block to the pump apparatus. The pump apparatus was designed with hydraulic tubing with a diameter of 3 inches that connect to two check valves with cracking pressures of 2 psi on opposite sides of the apparatus to simulate the heart's inlet and outlet valves.

The system was tested and configured to travel the full piston length of 0.75 inches in 0.4 seconds. From initial tests on the system, this requirement was reached with the PWMs generated at a frequency of 250Hz, corresponding to a step rate of 500 steps/sec, see the following figures for simulation outputs.

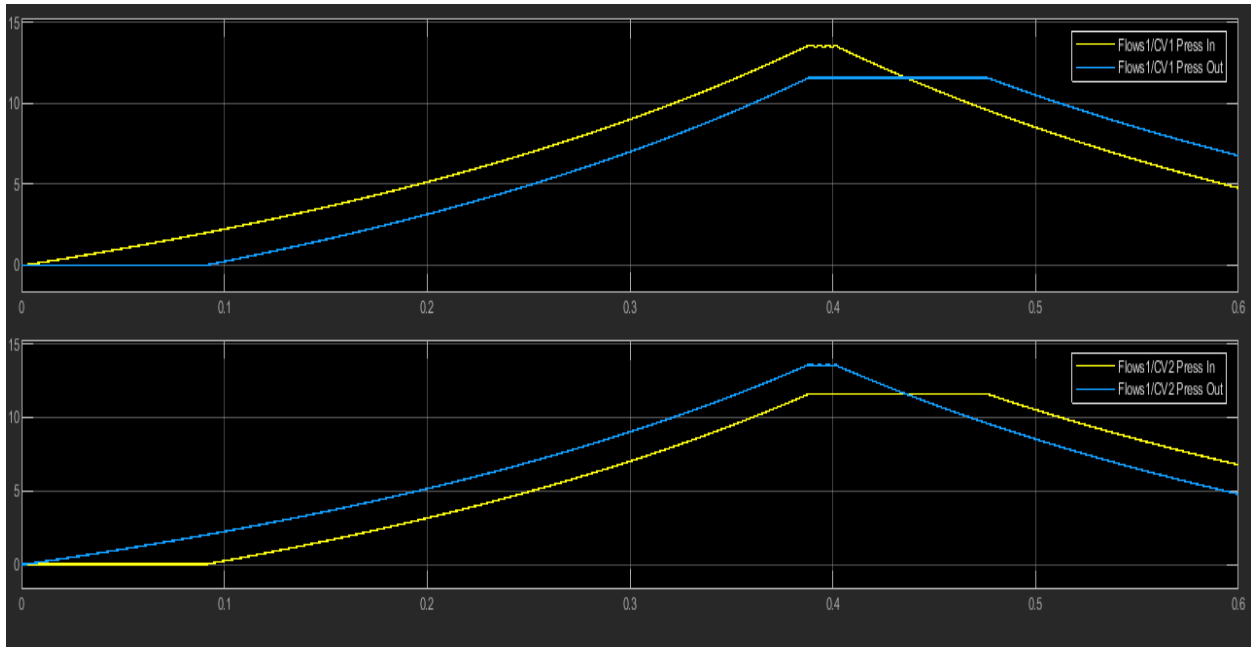


Figure 34: Simulink Pressure Outputs.

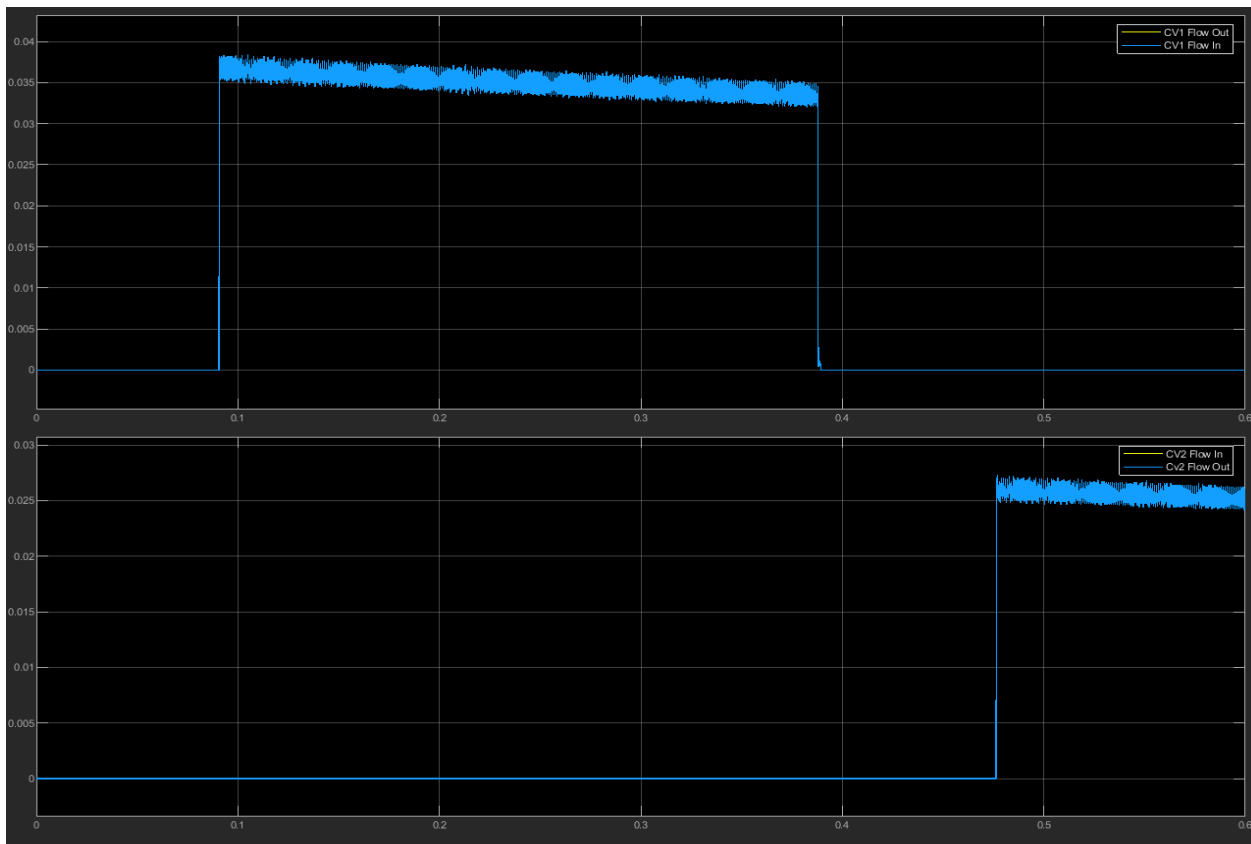


Figure 35: Simulink Flow Outputs.

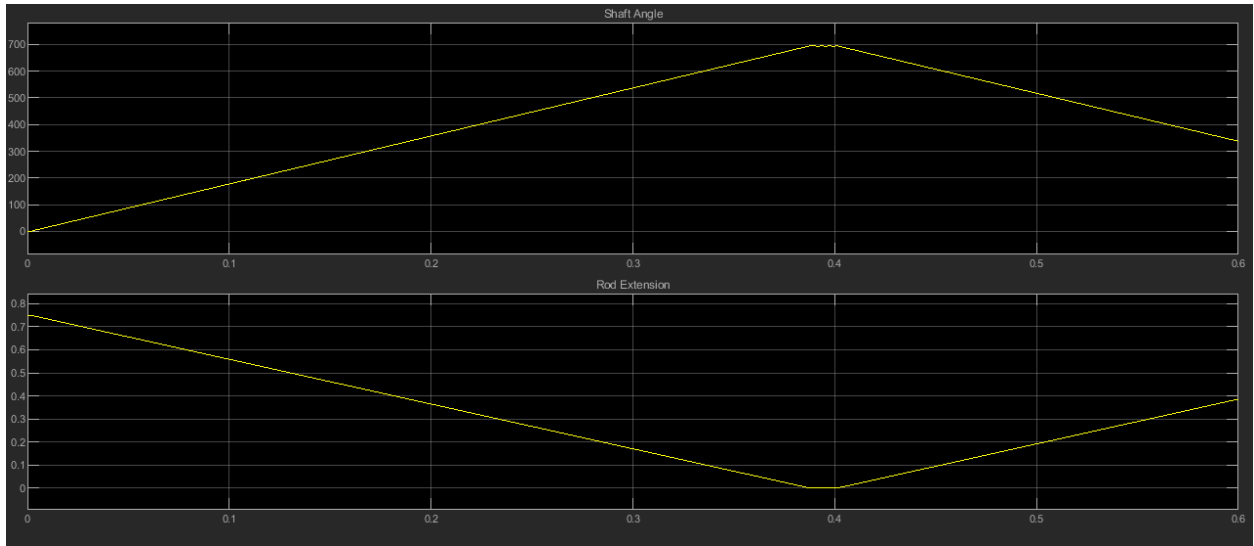


Figure 36: Simulink Step Angle and Piston Position.

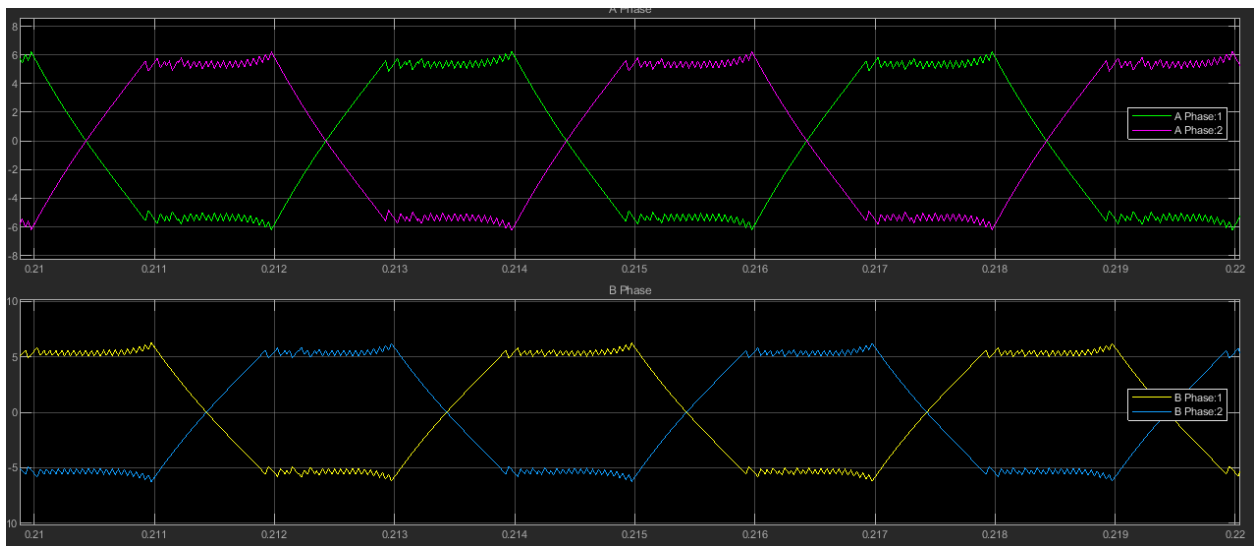


Figure 37: Simulink Phase Current.

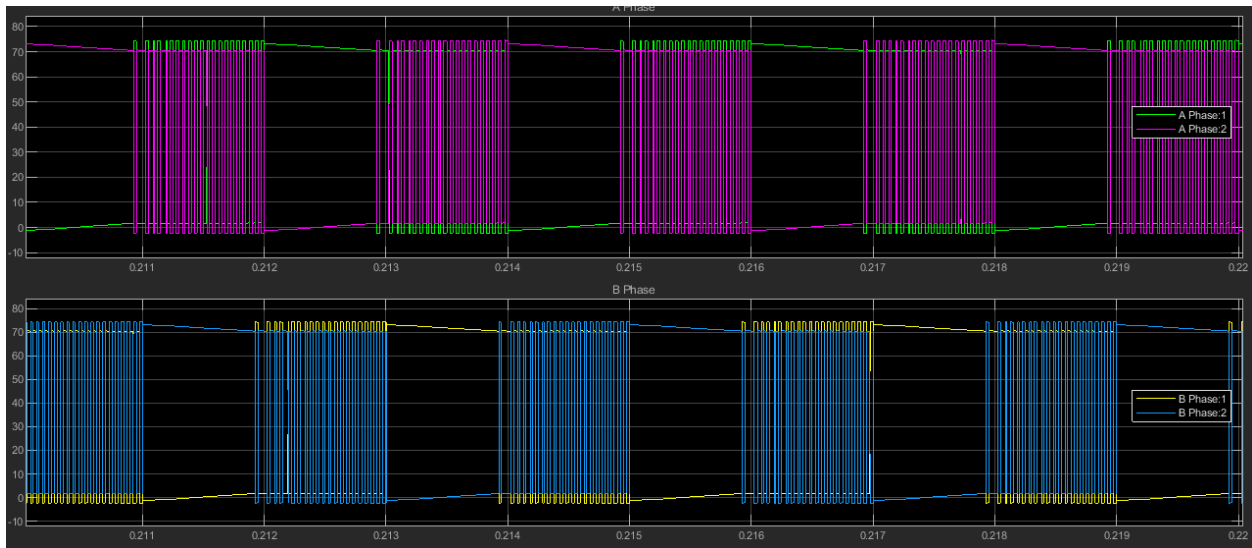


Figure 38: Simulink Phase Voltage.

This functionality at 500 steps/sec to reach a full piston translation in 0.4 seconds corresponds to a steady heart rate of 75 beats-per-minute (bpm). The chopper driver implementation can be analyzed in Fig. 32, 37, and 38, where the chopper driver is turning the PWM signal on and off, resulting in the chopping of the phase voltage to keep the phase current as close to the rated level as possible. The simulation can run up to 2000 steps/sec to reach a full piston translation in 0.186 sec, corresponding to a heart rate of 160 bpm.

Once the functionality of the simulation was able to meet the expected standards, the PID control system was then implemented and tested at varying frequencies to determine its functionality with a make-shift pressure profile. The results of the PID controllers pressure output, change in frequency, and error calculations at 1600 steps/sec can be seen in Fig. 39, 40 and 41 with controller gains of $K_p = 25$, $K_i = 10$, and $K_d = 4$.

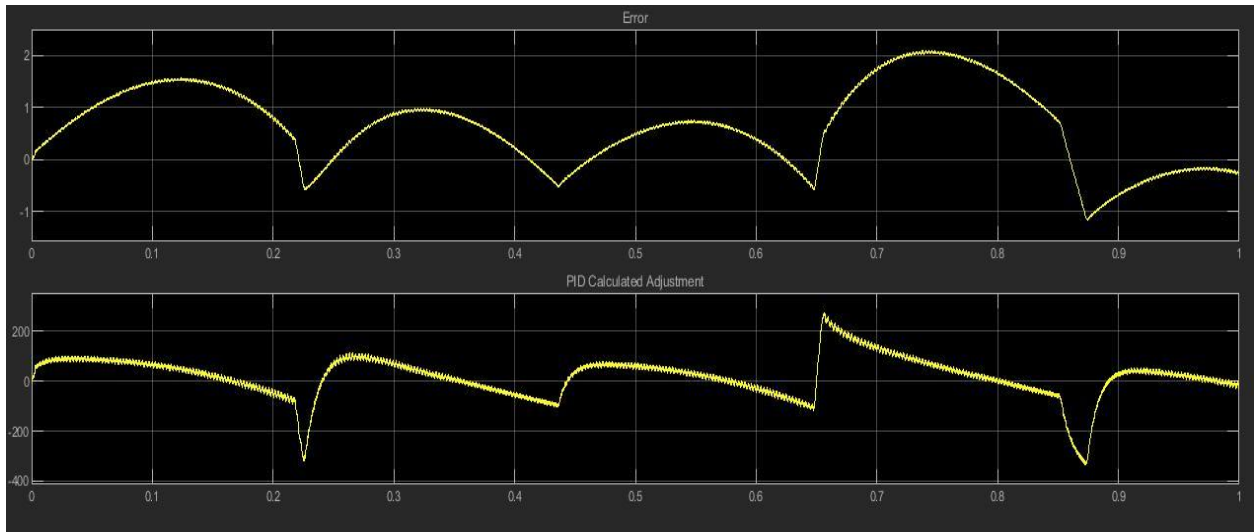


Figure 39: Error vs. PID Calculated Error.

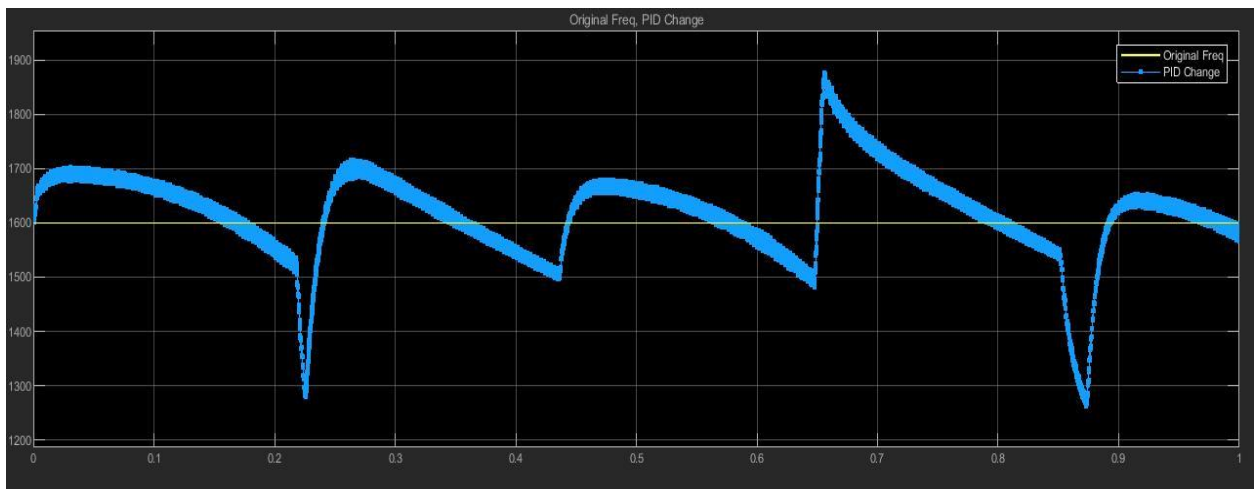


Figure 40: Frequency Input with PID adjustment.

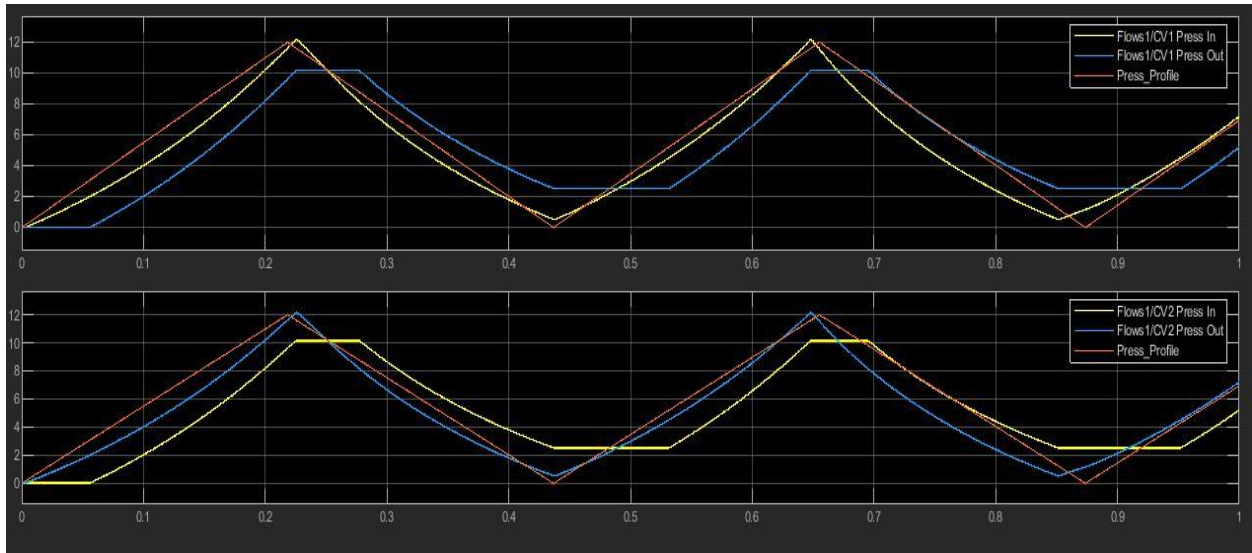


Figure 41: Pressure Results with PID controller.

Analyzing these figures, the controller starts correcting for the system error very quickly by increasing the starting frequency by $\sim 100\text{Hz}$. This results in the pressure output to nearly match with the desired output prior to the direction change, as can be seen in both the error calculations in Fig. 39, and the pressure results in Fig. 41. The error starts to stabilize until the simulation hits ~ 0.65 seconds, where the PID error spikes drastically and increases the frequency by approximately 300Hz . This is unexpected as the pressure output is ahead of the desired output, resulting in a negative error differential that should be reducing the frequency. Following this anomaly, the PID controller is quickly correcting for this issue until the next direction change occurs in the pump where the system is seen to stabilize again. From these results, it is concluded that the PID controller with selected gains is functional for this system to result in a stabilizing pressure output over a short period.

Chapter 5: Conclusion

The paper presented outlines the design, theory, and testing of a control system for a pulsatile pump. The control system was designed to take inputs from the pump in the form of pressure, temperature, and flow sensor data to be received through ADCs and used in the error calculations of a PID controller to vary the speed of the stepper motor. An additional pressure profile from an external source is set to be received through UART over RS-232 communication to specify the motor speed, number of data points, and an array of pressure data to be used in the PID error calculations. Upon profile receipt, the control board then sends out four PWM signals set at the specified motor speed to step the motor in a full-stepping configuration.

The basic functionality and theory behind stepper motors, motor drivers, control systems, and the system code flow were explained to give initial background information on the devices and concepts being used. The control board is separated into two main areas: the control board and the motor circuit. The control board takes care of the input signals (temperature, flow, pressure, and RS-232) and holds the microcontroller and power supply levels (3.3V, 5V, 1.024V) for system components. The motor circuit houses the gate drivers, H-bridges, overcurrent detectors, and the motor power supply connections.

Although the control circuitry for the RS-232 communication and sensor inputs were found to function and meet the designs expectations, the design of the motor driver circuitry proved to be inefficient to meet the project's torque and speed requirements. This issue was due to the L/R time constant of the motor not taken into consideration when designing the motor driver, preventing the motor from running at speeds higher than 30Hz to generate the rated torque and current. As a result, a new H-Bridge circuit was designed to incorporate a constant current chopper driver to improve the stepper motor torque output and efficiency.

The proposed chopper driver was then simulated in MATLAB Simulink with the rest of the motor circuit and pump apparatus to analyze the designs functionality. Through testing, the pump was able to travel the full piston length of 0.75 inches in the desired 0.4 second period when set to a step rate of 500 steps/sec. Being able reach this piston travel in 0.4 seconds corresponds to a heartbeat every 0.8 seconds to give a steady heart rate of 75 bpm. The pressure and flow outputs in Fig. 34 and 35 show that the apparatus is functioning as expected with the flow rate through the first check valve present once the cracking pressure is reached. As the pressure at the pump inlet decreases when the motor is reversed, check valve 2 is then open, allowing for fluid flow and pressure release in the reservoir. Simulations of the proposed chopper driver design are shown to be functional at fixed step rates of 2000 steps/sec to allow for heart rate simulations of up to ~160 bpm. With the simulation model proven to meet the designs standards, the PID controller was implemented with controller gains of $K_p = 25$, $K_i = 10$, and $K_d = 4$ to result in stable simulation outputs that meet the desired pressure profile levels.

5.1 Future Works

The future works for the project will require that a level switcher be selected and implemented into the board design to determine when the pumps piston has reached the cap to avoid any possible damage to the motor or pump. Additionally, the updated design, with the selected level switcher, will need to be fabricated and fully tested to analyze the functionality of the chopper driver with the stepper motor. Tests will need to be conducted with the communication between the control board and external source with varying profile inputs to test the functionality with the stepper motor. Lastly, the control board and motor will need to be tested with the pump apparatus to determine the accuracy of the PID controller and if modifications need to be made to its PID constants from those specified.

References

- [1] MOONS' Industries, "Hybrid Stepper Motor Catalog." 2019.
- [2] K. M. Le, H. Van Hoang and J. W. Jeon, "An Advanced Closed-Loop Control to Improve the Performance of Hybrid Stepper Motors," in *IEEE Transactions on Power Electronics*, vol. 32, no. 9, pp. 7244-7255, Sept. 2017, doi: 10.1109/TPEL.2016.2623341.
- [3] Faulhaber, "Application Note 001: Stepper motor basics."
- [4] M. K. Jenkins, D. Howe and T. S. Birch, "An improved design procedure for hybrid stepper motors," in *IEEE Transactions on Magnetics*, vol. 26, no. 5, pp. 2535-2537, Sept. 1990, doi: 10.1109/20.104789.
- [5] P. Beard. "Application Note Regarding H-Bridge Design and Operation." Nov. 14, 2014.
- [6] Renesas, "ISL78424, ISL78434, ISL78444 Datasheet." Sept. 10, 2018.
- [7] R. C. Dorf and R. H. Bishop, "Introduction to Control Systems," *Modern Control Systems*, 12th ed., Pearson, 2011, pp. 1-34.
- [8] Kiam Heong Ang, G. Chong and Yun Li, "PID control system analysis, design, and technology," in *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559-576, July 2005, doi: 10.1109/TCST.2005.847331.
- [9] R. Nowakowski and R. Taylor, "Linear vs. switching regulators in industrial applications with a 24V bus" *Analog Applications Journal*, Texas Instruments Incorporated, 2013.
- [10] STMicroelectronics, "AN468 Application Note: Stepper-Motor Performance Constant-Current Chopper Drive-ups." Dec. 2003.
- [11] M. Gomez, "App Note - High-Torque/High-Power Bipolar Stepper Motor Driver Using 8-bit PIC® Microcontroller," Microchip Technology Inc., 2017.
- [12] Mechoor RR, Schmidt T, Kung E. A Real-Time Programmable Pulsatile Flow Pump for In Vitro Cardiovascular Experimentation. *J Biomech Eng*. 2016;138(11):10.1115/1.4034561. doi:10.1115/1.4034561
- [13] ViVitro Labs Inc., "Superpump Brochure," 2018.
- [14] L. Liu, F. Wang, W. He, T. Li, W. Zhao and J. Ji, "Optimal control of permanent-magnet motor for pulsatile axial blood pump applications," 2011 International Conference on Electrical Machines and Systems, Beijing, 2011, pp. 1-5, doi: 10.1109/ICEMS.2011.6073935.
- [15] Law, Y.F., Cobbold, R.S.C., Johnston, K.W. *et al.* Computer-controlled pulsatile pump system for physiological flow simulation. *Med. Biol. Eng. Comput.* **25**, 590–595 (1987). <https://doi.org/10.1007/BF02441756>

Appendix

Appendix A

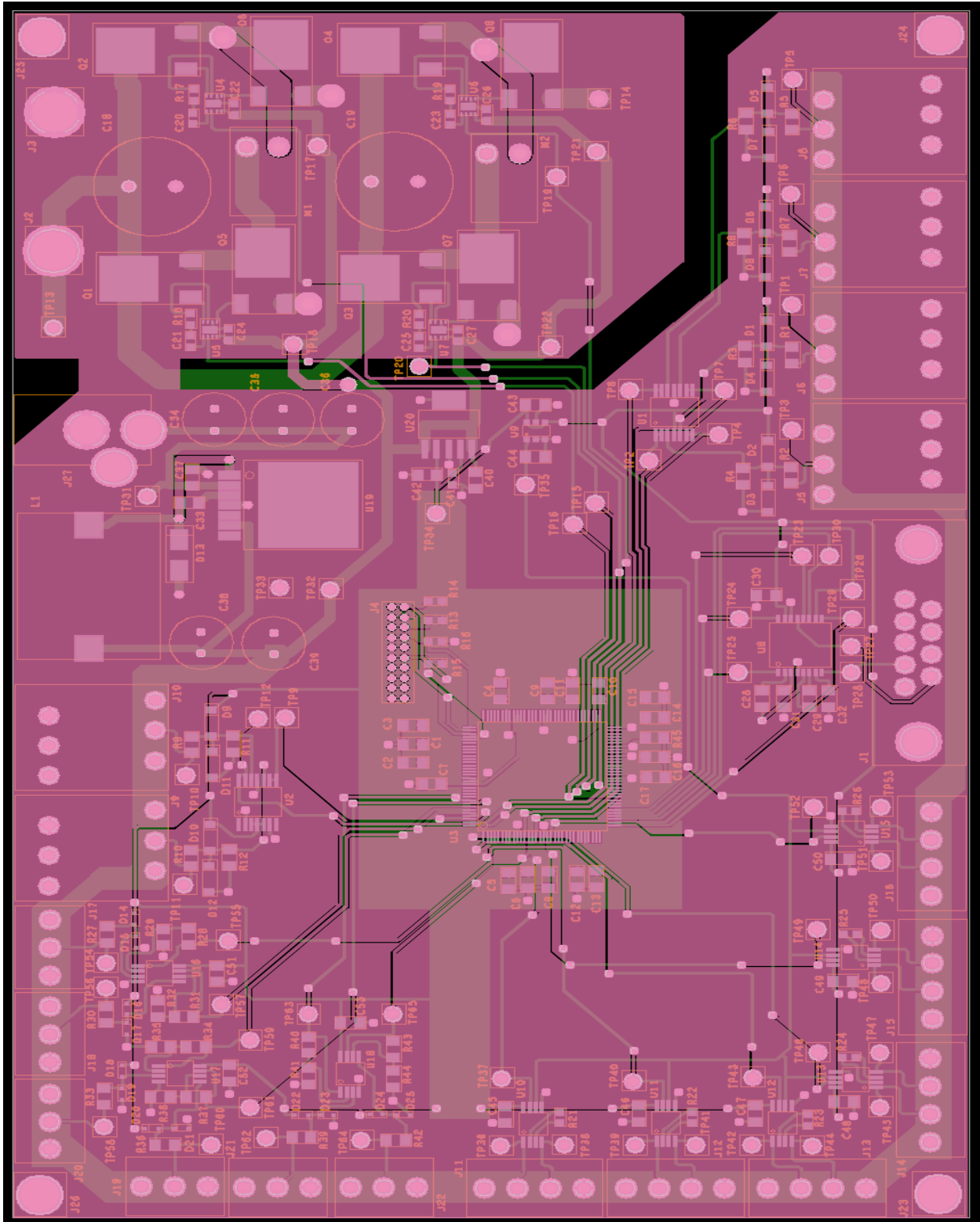


Figure 42: Original PCB Design, Enlarged.

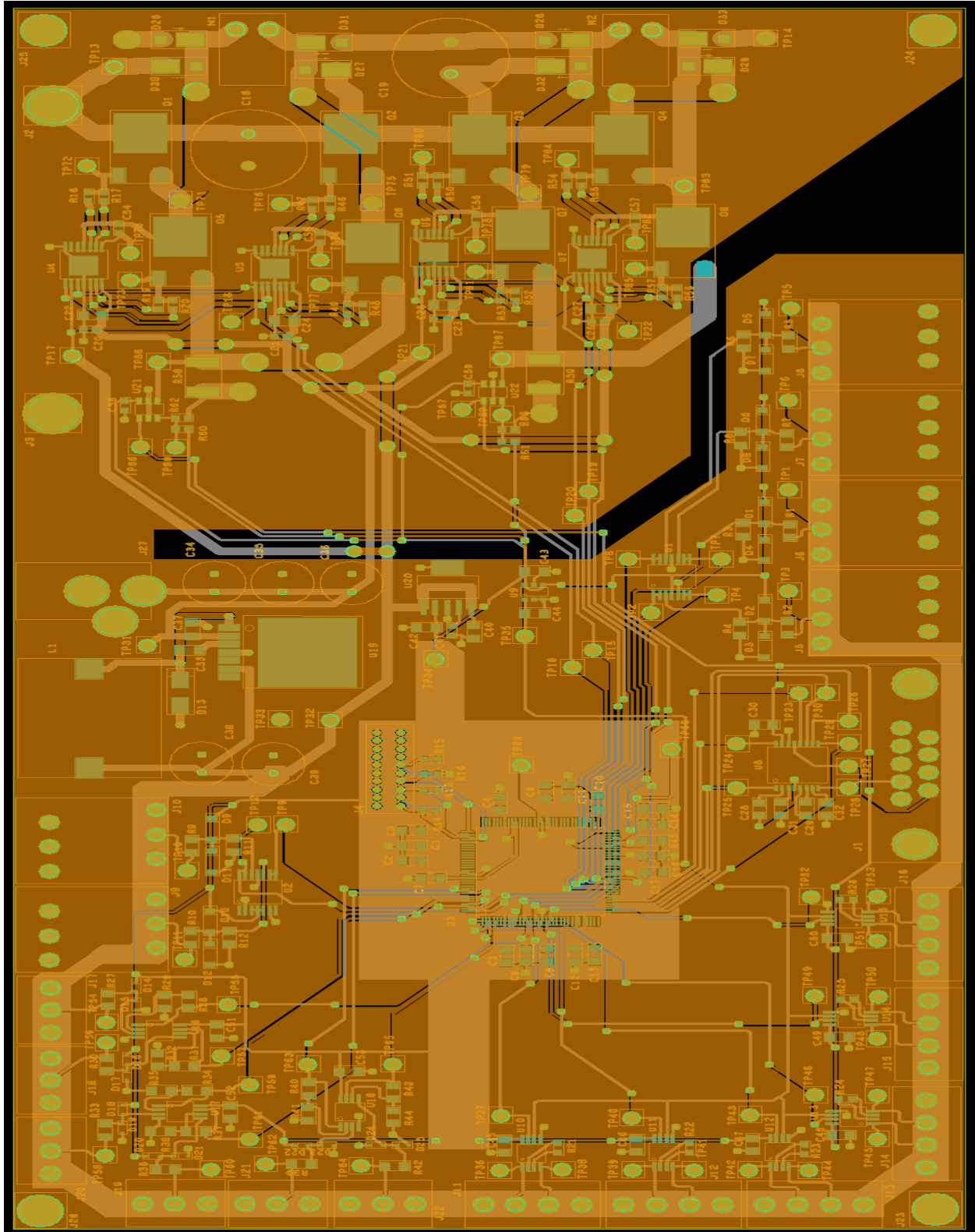


Figure 43: Modified PCB Design, Enlarged.

Appendix B

Table 4: Board Materials List.

Part Number	Manufacturer Part Number	Customer Reference
PS0S0DSX0-ND	PS0S0DSX0	AC_POWER_SOCKET
CP-002A-ND	PJ-002A	15V_POWER_JACK
277-15427-ND	3240155	68V_FASTON_CLIPS
AE10968-ND	A-DS 09 A/KG-T2S	DB9_JACK
277-5744-ND	1751264	PRES_TERMINAL
732-2748-ND	691214110003	TEMP_TERMINAL
277-1357-ND	1708039	FLOW_TERMINAL
493-15377-ND	UKL1H150KEDANA	POWER_15UF_CAP_RADIAL
478-11391-1-ND	08053C103JAT2A	POWER_0.01UF_CAP_5V_0805
ASPI-1306T-220M-TCT-ND	ASPI-1306T-220M-T	POWER_22UH_IND
641-1698-1-ND	CDBA240LL-HF	POWER_5V_SCHOTTKY
493-15709-ND	UPM1C181MED	POWER_180UF_CAP_RADIAL
36-5000-ND	5000	TESTPOINT
478-7946-1-ND	08053C104JAT2A	POWER_0.1UF_CAP_0805
1276-1188-1-ND	CL21B225KPFNNE	POWER_2.2UF_CAP_0805
478-11879-1-ND	08053C301JAT2A	POWER_300PF_CAP_0805
LM2678SX-5.0/NOPBCT-ND	LM2678SX-5.0/NOPB	POWER_5V_LIN_REGULATOR
REG103GA-3.3-ND	REG103GA-3.3	POWER_3.3V_REGULATOR
478-7901-1-ND	06033C103JAT2A	POWER_0.01UF_CAP_3.3V
MBR0520LCT-ND	MBR0520L	FLOW_SCHOTTKY
296-35691-1-ND	OPA4313PWR	FLOW_AMPLIFIER
296-47509-1-ND	MSP432E401YTPDTR	MCU_MSP432
RR08P10.0KDCT-ND	RR0816P-103-D	MCU_10KOHM_RES
RR08P100DCT-ND	RR0816P-101-D	MCU_100OHM_RES
445-7679-1-ND	C2012X5R0J226K125 AB	MCU_22UF_CAP
IRLR120TRLPBFCT-ND	IRLR120TRLPBF	MOTOR_MOSFET
296-19849-1-ND	MAX3227IDBR	LABVIEW_RS232_CHIP
478-11627-1-ND	0805ZC104JAT2A	LABVIEW_0.1UF_CAP
296-29663-1-ND	INA826AIDGKR	PRESS_AMPLIFIER
478-7946-1-ND	08053C104JAT2A	PRESS_0.1UF_CAP
296-13455-1-ND	LMV358IDGKR	TEMP_AMPLIFIER
RNCP0805FTD1K00CT-ND	RNCP0805FTD1K00	TEMP_1KOHM_RES
641-1784-1-ND	1SS389-G	TEMP_SCHOTTKY
478-7946-1-ND	08053C104JAT2A	TEMP_0.1UF_CAP

LM34DZ/NOPB-ND	LM34DZ/NOPB	TEMP_SENSOR
565-1736-ND	EKZE101ELL101MK16S	MOTOR_100UF_CAP
277-1779-ND	1988956	MOTOR_TERMINAL
RNCP0805FTD1K00CT-ND	RNCP0805FTD1K00	FLOW_1KOHM_RES
RMCF0603FT249RCT-ND	RMCF0603FT249R	PRESS_249OHM_RES
1276-2966-1-ND	CL21B474KBFNFNE	POWER_0.47UF_CAP_0805
P21067CT-ND	ERJ-PB6D2741V	FLOW_2.74KOHM_RES
399-3129-1-ND	C0805C335K8PACTU	MCU_3.3UF_CAP_0805
P4.87KDACT-ND	ERA-6AEB4871V	MCU_RES_4.87KOHM_0805_RBIAS
-ISL78434AVEZ-T7ACT-ND	ISL78434AVEZ-T7A	MOTOR_GATE_DRIVER
RHM2.2AYCT-ND	KTR03EZPF2R20	DRIVER_RESISTOR_2.2OHM
587-6009-1-ND	HMK107B710KAHT	DRIVER_CAP_0.1UF_BOOST
399-5089-1-ND	C0603C104K5RACTU	DRIVER_CAP_0.1UF_VSUPP
399-8004-1-ND	C0805C105K3RACTU	DRIVER_CAP_1UF_VSUPP
A124968CT-ND	2238026-2	MOTOR_FASTON_CLIP_FEMALE
A27906CT-ND	2-520102-2	MOTOR_FASTON_BLADE_MALE
94785-01-ND	94785	BUTT_CONN_18AWG
296-53434-1-ND	TLV1805DBVR	MOTOR_CHOPPER_COMPARATOR
A109683CT-ND	RLP73M3AR036FTDF	MOTOR_CHOPPER_RSENSE_36Mohm
RHM10KADCT-ND	ESR03EZPF1002	MOTOR_CHOPPER_RES_10KOHM_0603
RNCF0603DTE2K43CT-ND	RNCF0603DTE2K43	MOTOR_CHOPPER_RES_2.43KOHM_0603
V8PAM10HM3/IGICT-ND	V8PAM10HM3/I	MOTOR_SCHOTTKY_DIODE
SAM10253-ND	TMS-110-02-G-D	MCU_JTAG_20PIN_50MILx100MIL_PITCH
579-MCP1501T-10E/CHY	MCP1501T-10E/CHY	POWER_1.024VREF_REGULATOR