Spring 2023

Level and Temperature Controlled Vessel

Cameron Macesich
cwm40@uakron.edu

Senior Design Report:
Level and Temperature Controlled Vessel

Prepared for

Dr. Paul Schreuders

By
Cameron Macesich
April 10, 2023

Executive Summary

In the reactor experiment of CHEE 360 Chemical Engineering Lab, the student is asked to operate a Continuously Stirred Tank Reactor (CSTR) system and a system of CSTR's in series. These systems require both volume and temperature to be controlled closely to safely produce consistent and predictable results. Currently the students use manual control, which is prone to slow responses, operator errors, and lack repeatability. To overcome these challenges, an automated control system was designed, built, and evaluated.

A pair of control systems consisting of two linked Arduino microcontrollers, a pair of pumps, and sensors to measure the temperature and liquid level in a development-scale reaction vessel. These components form two feedback control loops to maintain user designated setpoints, minimizing the need for human operators to manually adjust flowrates. The Arduinos apply a Proportional-Integral (PI) control equation. The associated tuning constants with each control system were determined by performing both open loop step tests and closed loop tuning. The level controller had operating parameters of $K_P = 30$ and $K_I = 0.2$, and the temperature control system used $K_P = 14.26$ and $K_I = 0.67$. These tuning parameters produced closed loop stable steady states but had aggressive responses to setpoint changes with <15% overshoot. These parameters are most appropriate for lab systems where students may make changes to setpoints during the experiment and want to minimize time spent in a transitory state. The code written for the Arduino Mega 2560 R3 level controller and that for the Arduino Uno R3 temperature controller are shown in Appendices A and B, respectively. A circuit diagram of the system is shown in Appendix D.

For implementation, in the experimental reactor system, it is recommended that the system undergo an evaluation at the intended reaction conditions to assure the temperature

control system has sufficient capability to remove heat from the process and prevent runaway reactions. It is also recommended that a better method for outlet flow manipulation be developed, as it impacts reactor performance but lacks quantitative control. The 12-volt process pumps were limited to a span of 4.2—10 volts using the available interfaces. Additional control could be achieved by using a different motor controller with less voltage drop. Alternatively, superior pumps with gearboxes could be used to increase torque at low rotation speeds, increasing the pumps' ability to operate at lower flow rates.

The project was a demonstration of skills in process equipment engineering in designing and building the system, computer science in writing the code for the control system, process controls and analysis in tuning the system, and incorporated process safety elements. As an aspiring automation and controls engineer, the project was a means to get firsthand experience working with electronics, instrumentation, and controlled systems. The project provides an opportunity for future students to expand upon my work to explore reactions in CSTR's and serves as a demonstration of how programming and other technology skills, which grow in importance in industry each year, can be further incorporated into the chemical engineering curriculum at the University of Akron.

Introduction

In industrial applications, chemical manufacturers use automated systems to control valves, pumps, and other process equipment. The use of automated systems increases product quality and productivity by replacing process components that require human control with computer-controlled systems and instruments. These computer controls are faster and more precise than their human counterparts, reducing both the operating labor and time spent away from process setpoints. The reactor activities performed in CHEE 360, Chemical Engineering Lab, all required human control, and could be improved by the implementation of an automated system to control the vessel level.

The purpose of the performed activities was to create an automated vessel akin to that used in the continuously stirred tank reactor section of the Chemical Engineering Lab reactor lab section with integrated monitoring and control of the vessel liquid level and temperature.

Background

Continuously Stirred Tank Reactor

Continuously stirred tank reactors (CSTR's), commonly used in biomanufacturing due to their superior controllability and lower cost when compared to their tubular counterparts [1], are steady state process units that operate on the assumption of constant liquid volume and with perfect mixing [2, p. 38]. Reaction rate ($-r_A$) in the vessel depends on the rate constant, k. The relationship between rate constant and temperature is described by Equation 1. CSTR's are also chosen for their simplicity in design; a CSTR is a mixed tank with an inlet and an outlet which allows measurements to be taken of the bulk liquid at any point in the vessel and any adjustments

to the bulk material (such as temperature and pH) can be made directly on the bulk by means of heat exchangers on or in the vessel and chemical additions directly to the bulk.

$$k(T) = A * exp\ (-E_a/RT) \quad (1)$$

Controlling the vessel level and temperature are important for CSTR operation as they impact the fundamental relationships that impact the reaction occurring in the vessel. The design equation (Equation 2) shows that the fractional molar conversion of reagent to product (X) is dependent upon the molar flow rate of reagent into the vessel (F_{A0} [mol A/minute]), the reactor's volume (V [m^3]), and reaction rate ($-r_A$ [mol A/(minute*m^3)]) [2, p. 38].

$$V = \frac{F_{A0}X}{(-r_A)_{Exit}} \quad (2)$$

A CSTR is designed under the assumption of constant volume, so the volume of liquid in the reactor must be controlled for the reactor to have an effluent concentration profile matching the intended design. Volume control is also a safety concern; high level excursions can result in overflows in open vessels or rapid pressure increase in closed vessels from compression of air in the top of the reactor as liquid level rises. High pressure events can trigger relief devices, which may result in unplanned downtime for cleaning and maintenance, and a subsequent loss in plant productivity [3]. In a CSTR, the only removal of heat occurs from conduction through the walls of the vessel, through a heat exchanger (if present), and by the flow of process liquid out of the reactor. For an exothermic reaction, which generates heat, further measures must be taken to remove heat from the system to prevent runaway reactions and ensure product concentration adheres to design criteria. Use of a temperature control loop with a jacket or heat exchanger in the vessel can mitigate this risk by removing heat from the system, slowing down the rate of reaction.

Control of CSTR's

As noted earlier, in CHEE 360, Chemical Engineering Lab, the level in the CSTR was manually controlled by the adjustment of the feed and outlet pump setpoints by the students operating the reactor. To determine if the system was at steady state, tape was placed on the vessel at the desired liquid level and the student watched to see if the level deviated from the mark over time. Manual processes such as this create variability in experiments and decrease experiment repeatability. Use of an automated feedback control system to continuously monitor and control the liquid level would increase the speed and accuracy of input pumping rate to level setpoint and reduce deviations from setpoint.

Computer-Based Controllers

Computer controlled processes can utilize various control systems to read input from sensors and instruments, run the user program, and set outputs to devices out in the field like valves and pumps.

The objective for the level control system is to hold the level in the vessel steady at its setpoint and automatically perform changes to vessel level when a setpoint change is made by the operator. The objective for the temperature control system is to keep the temperature in the vessel at a specified setpoint and resist deviation from setpoint due to generated heat to ensure the reaction proceeds at the expected rate of reaction.

The most common industrial control solution is the programmable logic controller (PLC). A PLC is a hardened computer designed for industrial use that can quickly react to inputs and set outputs. They are coded in ladder logic, a graphical programming language that is based on the hardwired control ladder diagrams that they have replaced. Ladder logic uses a combination of

basic logical and mathematical functions to operate ([4, p. 475]). Using a computer over a hardwired control system allows the user to make changes to the control system without having to reconfigure or rewire the system. While useful in industrial applications, PLCs often require special licensed software to code and have high upfront cost for the unit itself and the field devices which it controls.

Microcontroller-Based Controls

A microcontroller unit (MCU) is similar in concept to a PLC in that it is designed to receive input and output from field devices and run a user program, but differs in the programming language of operation, the robustness of components, and most importantly, the cost of the device and its sensors. MCU's are often used by students and hobbyists due to their reduced financial barriers to entry and ease of use. The Arduino Mega 2560 (Mega) is one such MCU, and is distinguished by its low cost, the relatively large size of the board, high number of input and output pins relative to similar systems, and its C++ based programming language. The software used to program and load the program onto the board, the Arduino Integrated Development Environment (IDE) [5], is free to use and there are various free libraries that implement more complex programs like Proportional, Integral, Derivative (PID) control and communication with digital sensors.

Process Sensors

A level sensor is required for level control of the vessel. An ultrasonic sensor can measure the distance between itself and a target by measuring the time it takes for an emitted soundwave to travel to the target and then back to a receiver on the instrument. The time taken for the reflected wave to be detected is divided by two times the speed of sound to determine the

distance to the object. The advantage of an ultrasonic sensor over a laser distance sensor is that the ultrasonic sensor does not depend on an object's opacity, making it ideal for measuring the distance to fluids, which would allow the majority of the light waves to pass through.

Process Actuators

DC Motor Driven Pumps

To add fluid to the vessel for level control, a pump is required. Pumps can be controlled by two types of electric motors, alternating current (AC) and direct current (DC). For smaller loads, DC motors present the advantage of being able to draw power from batteries and the speed and direction is easily controlled by changing the voltage across its terminals. A MCU like the Mega cannot deliver true continuous analog output voltages to its pins, but it can use Pulse Width Modulation (PWM) to simulate an analog output voltage by very quickly triggering its digital pins on and off to form a square wave [6]. The average voltage of the resultant wave is between 0 and 5V, the maximum output voltage of the Mega. Equation 3 describes the output voltage of an Arduino PWM pin. The PWM output (n) is the numerator of the fractional time that the output is energized. The ratio of n:255 is known as the Duty Cycle. When n=255, the output is always on. Figure 1 shows an example of a PWM output, where the output voltage of 2.5 volts is the result of the output being energized 50% of the time.

$$V_{PWM} = 5V * \frac{t_{on}}{Period} = 5V * \frac{n}{255} \tag{3}$$

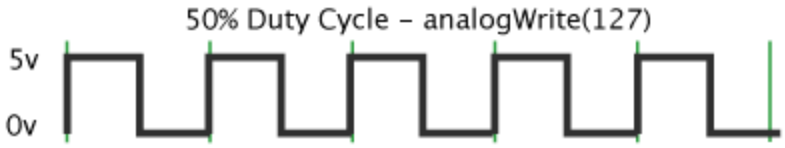


Figure 1: Example of how a digital (on/off) output can use PWM to simulate an analog output [6].

Peristaltic pumps operate by squeezing a tubing element with rotating rollers to push liquid forward through the tube [7, p. 208]. They are often used in bioprocessing because the liquid flow rate through the pump is directly proportional to the rotational speed of the rollers (Figure 2), leading to high precision in volumetric flow rate for dosing applications, and for their ease of cleaning. The fluid is within the tube at all times and does not need to go through a pump head which may be harder to clean.

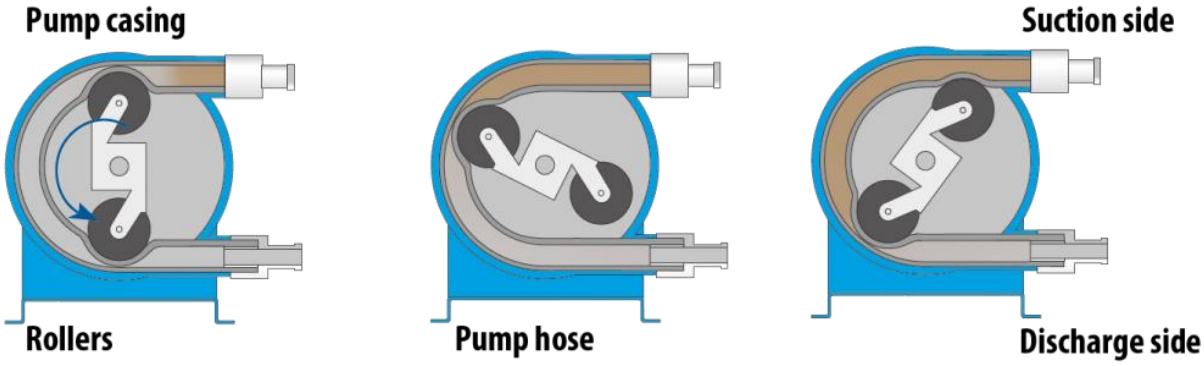


Figure 2: Operating principle of a peristaltic pump. Liquid is pulled by the rollers from the suction side to the discharge side [8].

Diaphragm pumps operate by expanding and contracting a diaphragm of a flexible material in the wall of a chamber with one-way check valves on the inlet and outlet [7, p. 206]. As the diaphragm expands, the liquid in the chamber is pushed out of the outlet. When it contracts, the volume of the chamber increases, and fluid is sucked into the chamber for the inlet. Diaphragm pumps are preferred for higher flowrate applications than peristaltic pumps, which

are limited to smaller flowrates, but both are positive displacement pumps with near constant volumetric flow rate per cycle [7, p. 207].

Electromechanical Relays

Electromechanical relays are magnetic switches that use electromagnetism to switch contacts. They provide the ability to use a small control current to energize or de-energize a higher current system [9]. Relays may be normally open (NO), meaning that when the control side is de-energized, the process side is de-energized, or normally closed (NC), where the process side is energized when the control side is de-energized. In lighter applications, a control relay will suffice, but when the process side load is large, like that of a three phase 480V pump in a factory, special hardened relays called contactors are used that can withstand the higher process current draw. For the smaller 12V-2A DC system used in this project, a standard control relay will suffice. Relays can be used to increase process safety by giving the operator or controller the ability to toggle the power to process equipment. For emergency stop functionality, the control relay is always hardwired into the emergency stop circuit to reduce reliance on the controller in case the system needs to be immediately halted by the operator [9].

Control Algorithms

Liquid Level Control

Systems require different control methods depending on the desired response and order of the system. Using the mass balance for a single CSTR with a pumped inlet and gravity drained outlet, the relationship between the deviation from steady-state level of the liquid in the tank ($H^*(s)$), the controlled variable (CV), can be derived in terms of the manipulated variable (MV), the deviation from steady state inlet pump PWM setpoint ($P^*(s)$). The derivation for the system

in the Laplace domain is shown in Appendix C. The transfer function for the gravity drained reactor system in terms of the deviation from steady state values has a first order relationship between the MV and CV, as shown in Equation 4, where K is the gain and τ is the process time constant.

$$H^*(s) = \frac{K}{\tau s + 1} \cdot P^*(s) \tag{4}$$

A first-order response is desired from the system, so by direct synthesis the transfer function for the desired level controller was derived, shown in Equation 5. The desired controller is a Proportional-Integral (PI) controller with no derivative action. A PI controller calculates the MV setpoint using the instantaneous error from CV setpoint (the term K_P) and the amount of accumulated error in CV over time (the term K_I/s), where K_P and K_I are tuning constants. The error is the difference between the measured value of the CV and its setpoint value. Larger tuning constants increase the magnitude of controller response.

$$G_C = K_P + \frac{K_I}{s} \tag{5}$$

Temperature Control

The relationship between the pumped flow of cold water through the heat exchanger, $Q^*(s)$, can be related to the temperature of the water, $T^*(s)$, in the vessel by means of the energy balance. The complete derivation of the transfer function for tank temperature in terms of heat exchanger flow (Equation 6) is shown in Appendix C. τ_1 and τ_2 are placeholder values for process time constants, and K for process gain, which can all be determined experimentally performing a step test and fitting the results with an equation. A step test is performed by manually adjusting the value of the manipulated variable and recording the response of the controlled variable.

$$T^*(s) = \frac{K}{\tau_1\tau_2 s^2 + (\tau_1 + \tau_2)s + 1} \cdot Q^*_{HEX}(s) \tag{6}$$

The ideal controller for the system is of equal order to the system, second order. By the Internal Module Control (IMC) method, which establishes starting controller settings several common process models [4], the best control solution for a second order system is a PID controller shown in Equation 7 and in Equation 8 where the integral term is in respect to the time constant τ_I.

$$G_c = K_P + \frac{K_I}{s} + K_D s \tag{7}$$

$$G_c = K_P + \frac{1}{\tau_I s} + K_D s \tag{8}$$

While accurate in its description of the system, second order models are more difficult to fit and characterize in real processes with measurement noise, which controllers with derivative action are very sensitive to [4, p. 192]. Lower order systems are also more convenient for control system design [4, p. 92].

For complex responses, Skogestad [10] states that the order of a higher order system can be reduced a First-Order-Plus-Time-Delay (FOPTD), seen in Equation 9, by means of the "Half Rule", where one half of the largest neglected time constant is added to the largest time constant, while the other half and remaining time constants are added to the delay.

$$\frac{K}{(\tau_1 s + 1)(\tau_2 s + 1)} \rightarrow \frac{K e^{-\tau_2 s/2}}{(\tau_1 + \tau_2/2)s + 1} \tag{9}$$

When the heat exchanger transfer function is reduced by the "Half Rule" to FOPTD (Equation 10), the IMC method can be used to determine coefficients for a PI controller. Therefore, if a FOPTD function is fit to an open loop response, the controller parameters can be estimated using the IMC relations [11] shown in dependent form in Equation 11.

$$T^*(s) = \frac{Ke^{-\theta s}}{\tau s + 1} * Q^*_{HEX}(s) \tag{10}$$

$$G_c = K_C\left(1 + \frac{1}{\tau_I s}\right), K_C = \frac{1}{K} * \frac{\tau}{\theta + \tau_c}, \tau_I = \tau \tag{11}$$

Materials:

- Laptop Computer with the following software installed:
  - PuTTY SSH and Telnet Client v0.78 [12]
  - Arduino Integrated Development Environment v2.04 [5]
    - PID_v2 Library [13]
    - OneWire Library [14]
    - DallasTemperature Library [15]
  - Microsoft Excel [16]
  - Loop-Pro Trainer [17]
- Adafruit Arduino UNO Rev3
- CSTR Vessel
  - 400 mL beaker with stopcock outlet
  - Custom made by Jack Gillespie
- Ring stands (x2)
- Ring stand clamps (x2)
- Inland Arduino Super Learning Kit [18][1]
  - Inland Arduino Mega 2560 Rev3
  - HC-SR04 Ultrasonic level sensor
  - AA Battery Bank (9V)
  - 5V Relay Control Module SRD-05VDC-SL-C
  - Solderless Breadboard
  - 10 kΩ Resistor (x4)
- Taiss 3pcs Momentary Push Button Switch 22mm 10A 440V 1NO 1NC DPST Red Yellow Green Pushbutton Switches LA38-11BNRYG[2]
- Push-Pull Emergency Stop Button
  - Donated By Scott Smejkal
- uxcell Push Button Switch Control Station Box 22mm 4 Button Aperture White and Black[2]
- Treedix 5 pcs Prototype Shield Board Compatible with Arduino Mega 2560 Solderable Universal BreadBoard PCB Double Sided Tinned Gold Plated Holes[2]
- General Protoboard[2]

[1] Material Purchased from Micro Center(https://www.microcenter.com/)

- L298N H-bridge motor driver
- 12V 2A DC Power Supply
- Gikfun 12V DC Dosing Pump Peristaltic Dosing Head with Connector For Arduino Aquarium Lab Analytic Diy AE1207[2]
- 2mm ID silicon tubing[2]
- Gikfun Mini DC 6V to 12V R385 Water cooled Water Pump Air Diaphragm Pump EK1856[2]
- Gikfun DS18B20 Temperature Sensor Waterproof Digital Thermal Probe Sensor for Arduino (Pack of 5pcs) EK1083[2]
- ¼" ID Clear PVC Tubing[2]
- ¼" OD copper tubing

Methods:

Initial Training

The initial training activities performed were activities from the Inland Arduino Super Learning kit. Activities performed were those predetermined to be relevant to the project objective. The activities are summarized below:

- Activity 1: Hello World [18] - Print to Serial.
- Activity 2: LED Blinking [18] - Use of timers and output pins to make an LED light blink.
- Activity 3: PWM [18] - The use of an analog input device (potentiometer) to manipulate the brightness of an LED using the Arduino AnalogWrite(PIN, INT) command.
- Activity 4: Traffic Light [18] - More complicated wiring of 3 LED's which were activated by the Arduino by a timer.
- Activity 6: Button Controlled LED [18] - Using a digital input (momentary pushbutton) to energize an LED light on a digital output. The wiring for the momentary pushbutton would later be replicated to make the button control panel.

[2] Material purchased from Amazon (amazon.com)

- Activity 26: HC-SR04 Ultrasonic Sensor [18] - Use the HC-SR04 sensor to determine the range of an object by sending a sound wave then measuring the time required to detect the echo. The activity would form the foundation of the level sensor code.

Electronics Prototyping

Once the Arduino introductory activities were completed, the system was mocked up on a solderless breadboard. The initial control program was created, with the primary function to read input from the button panel (shown in Figure 4) and get input from the DS18B20 temperature and HC-SR04 level sensors. The emergency stop button was wired to be normally closed, while the momentary pushbuttons were wired on their normally open terminals. Readings from the sensors were transferred to the Laptop Computer using a USB cable and PuTTY. The data was recorded as .LOG files and transformed into comma separated value (.CSV) files in Microsoft Excel for analysis in Loop-Pro Trainer. Next, the level and heat exchanger pumps were wired into the system by sending a 0-5V Pulse Width Modulated (PWM) signal to the L298N H-bridge motor controller interface, which controlled the pump direction and speed using power from the 12V DC wall power supply (Figure 3).

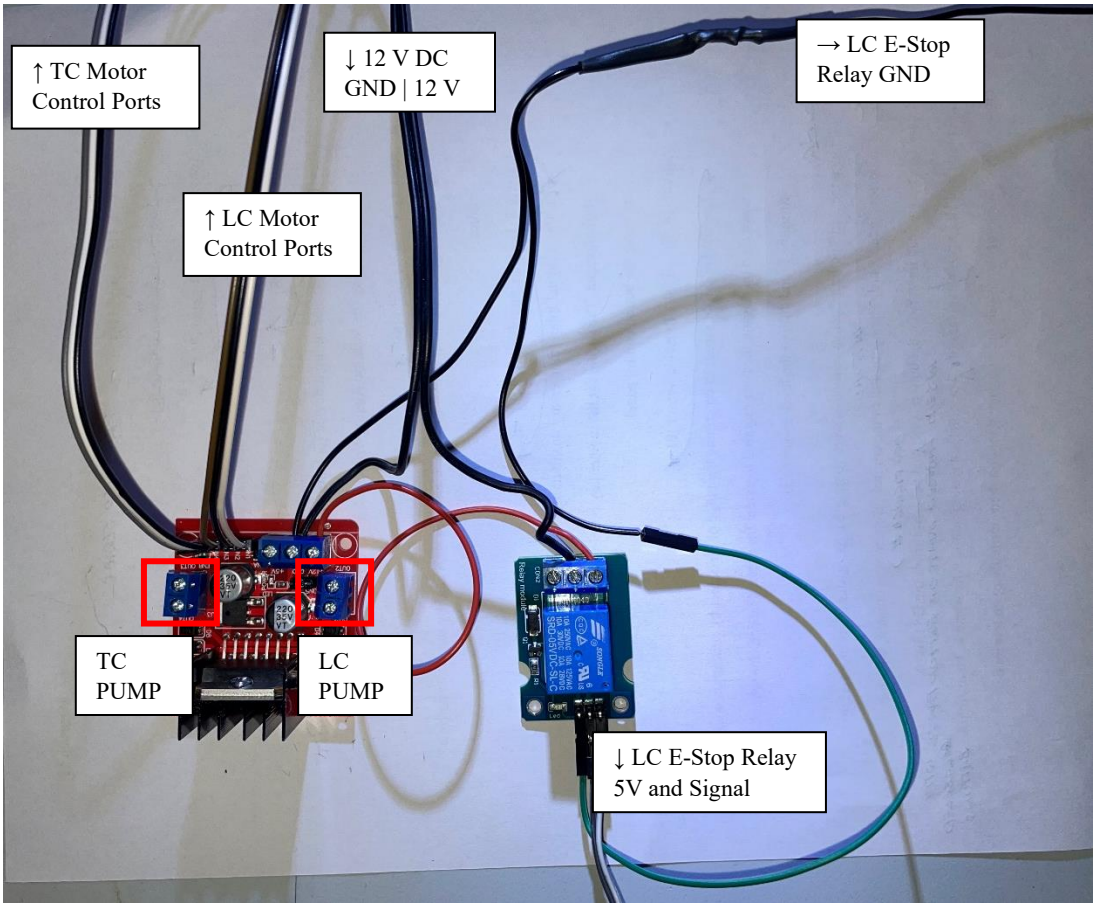


Figure 3: L298N Motor control interface and E-Stop Relay for the control system. Wires are labeled.

The emergency stop button, initially wired directly to an Arduino input pin that digitally controlled pump setpoints, was reconfigured to energize the normally open control pins of a 5V relay control module that regulated the 12V power. The relay provided a hardwired control for the pump emergency stop mechanism. When the emergency stop button is pushed, the relay is de-energized, killing power to the pumps. Once all electrical components were working correctly, shields for the Arduinos were fabricated with ports for all input and output devices (Figure 5 and Figure 6).

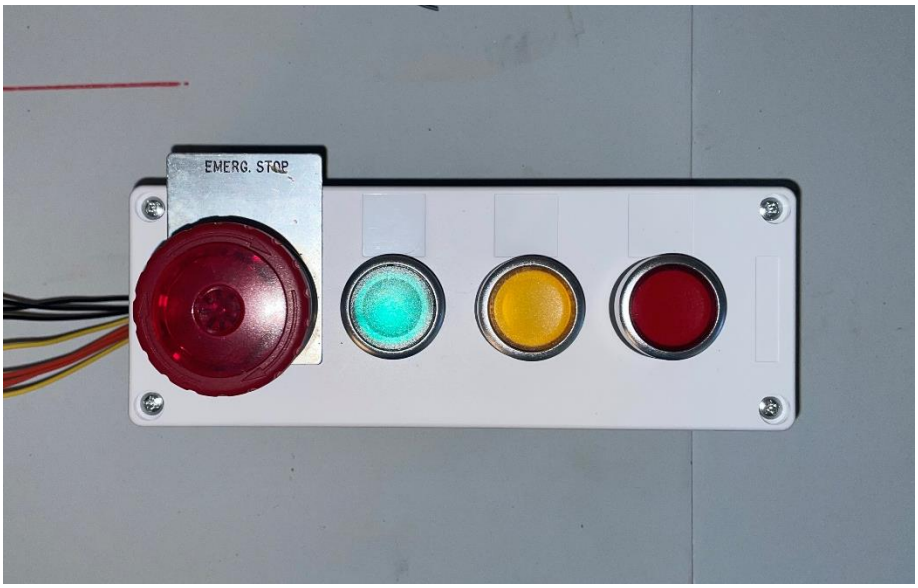


Figure 4: Four button panel fabricated for control system. Left to right: RED – Emergency Stop, GRN – Start, YEL – User-Programmable, RED – Normal Stop.

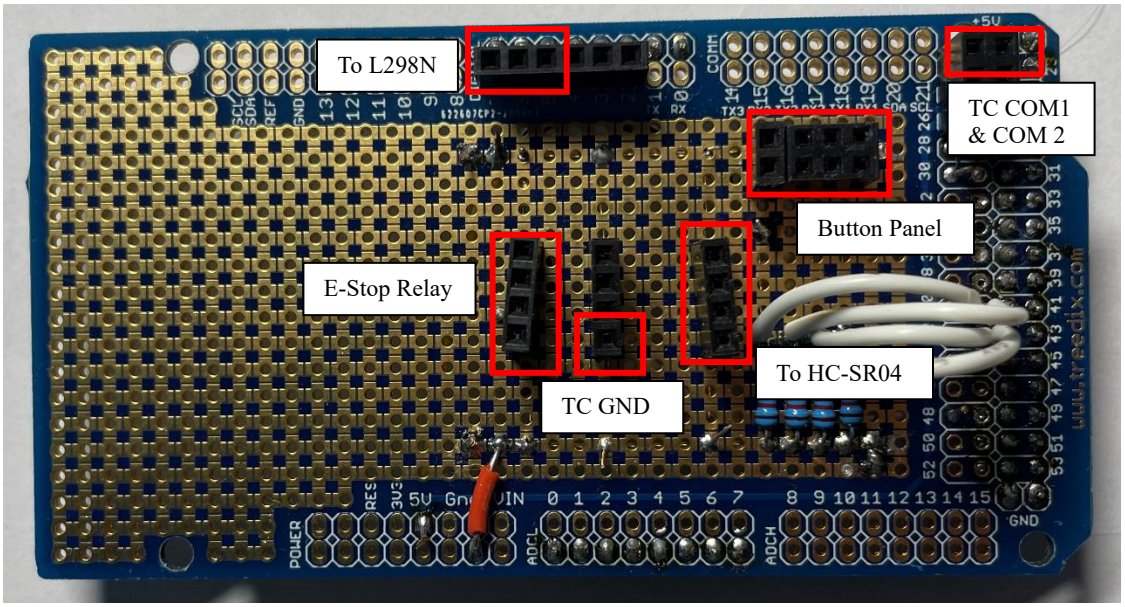


Figure 5: Arduino Mega level controller shield. Ports are labeled.

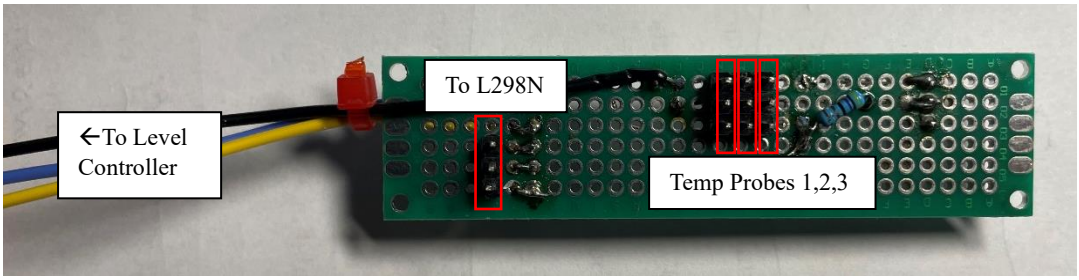


Figure 6: Arduino Uno temperature controller shield. Ports are labeled. Three wires go to level controller for communication: BLK – TC GND, YEL – COM1, BLUE – COM2

Heat Exchanger Fabrication

A heat exchanger was fabricated for the vessel using ¼" OD copper pipe as follows:

1. A small slot was milled in the bottom of a 2" OD guide tube around which the copper pipe could be wrapped to make a helical coil.
2. The copper pipe was given a 90 degree bend using a pipe bender at what would become the bottom of the exchanger.
3. The copper pipe was threaded into the notched end of the guide tube.
4. The copper pipe was placed into the notch in the guide tube.
5. The guide tube was clamped into the chuck of a lathe and slowly rotated while maintaining tension on the copper pipe to coil it around the guide.
6. The copper heat exchanger was then removed from the guide tube and given a 90-degree bend to direct the top exchanger pipe upwards, away from the bottom.

The reaction vessel, exchanger, and sensors can be seen in Figure 7.

Figure 7: Reaction vessel and exchanger with internal temperature probe and HC-SR04 ultrasonic sensor.

Experimental Activities

The level control system comprised of the reaction vessel with an outlet stopcock, included a peristaltic fill pump controlled through the L298N motor control interface an HC-SR04 ultrasonic level sensor, a four-button panel, and the Arduino Mega. The devices interfaced with the Mega via the fabricated shield (Figure 5). See Appendix D for full system wiring. Level control testing was completed without the temperature controller enabled or connected. The heat exchanger was removed from the reaction vessel for level control testing.

Activity 1:

Pump curves were created by applying different PWM duties to the pumps and calculating the volumetric flow rate by recording the time to fill a graduated container. To mitigate the impact of pressure drop due to friction on pump evaluations, the temperature control pump was connected to the heat exchanger and level control pump had the same inlet and outlet line lengths as during normal operation. The outlet was directed to a graduated container. The pump was turned on and the required amount of time to fill to a graduation height on the container was recorded. For the sake of time, the target graduation was lower for smaller flowrates. Figure 9 and Figure 10 show the resulting pump curves.

Activity 2:

Code was written to perform open loop step tests of the level and temperature systems to determine tuning parameters. Initial level tuning activities were performed without the heat exchanger installed in the vessel. As the level pump span was determined to be 13 – 65 mL/min, the reactor outlet stopcock was closed to restrict flow out of the vessel to this range. The pump was turned on to a setpoint of PWM 180. After ten minutes, the pump setpoint was bumped up to PWM 220, and the response was observed and recorded (Figure 11). After failing to reach steady state in the first attempt, the outlet flow rate was reduced and the experiment was repeated with a smaller change in level control motor setpoint of 110 to 115 (Figure 12).

Activity 3:

Due to large time constants and difficulties in an achieving open loop steady state in level response in Activity 2, closed loop setpoint change tests were performed using rules-of-thumb [19] to determine initial tuning parameters ($K_p = 50$, $K_i = 0.3$). A PI test program was written to control liquid level to a setpoint of 7 cm. The reset button was rewired to bump the setpoint

between 7 and 8 cm when pushed. The system was allowed to come to steady state, then the setpoint was changed to 8 cm and allowed to come to steady state, then bumped back down to 7 cm. The responses were exported to the laptop computer and compiled using the PuTTY SSH and Telnet client. In response to large oscillations shown in Figure 13, the tuning parameters were adjusted to $K_p = 30$ and $K_i = 0.2$, and the activity was repeated. Results of the repeated activity are shown in Figure 14.

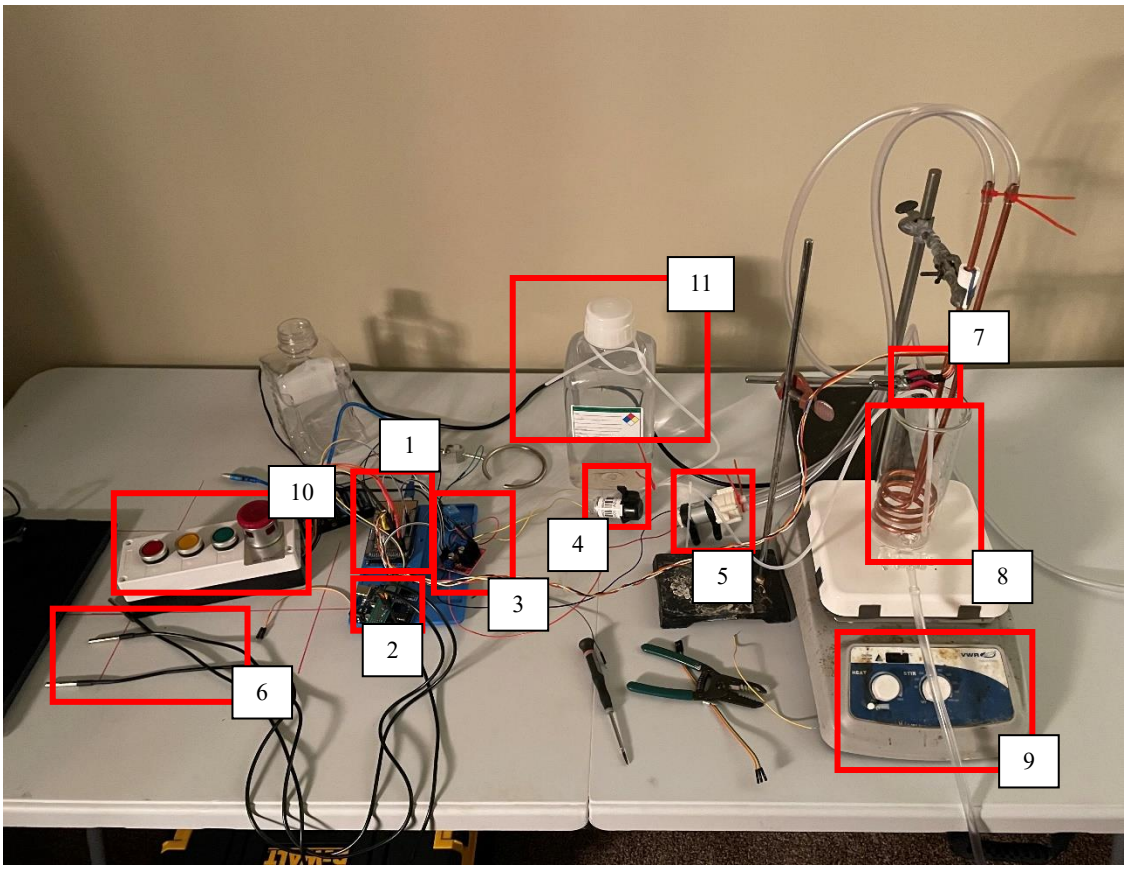


Figure 8: Experimental set-up and equipment for temperature control activities. Equipment:

1.) Mega Level controller 2.) UNO Temperature controller 3.) L298N & E-Stop Relay 4.) Level control peristaltic pump 5.) Temperature control diaphragm pump 6.) Temperature probes (not installed) 7.) HC-SR04 ultrasonic sensor 8.) Reaction vessel with heat exchanger installed and a magnetic stir bar 9.) Hot plate with magnetic impeller 10.) Button Panel 11.) Reactor feed material (water)

Not pictured: Chilled water vessel, reactor effluent collection

The temperature control system is pictured in Figure 8. The vessel was placed on a hot plate with a magnetic impeller. The copper heat exchanger was suspended in the reaction vessel

above the stir bar, and tubing was connected between it and the pump outlet. The tubing on the pump inlet was routed to the bottom of the chilled water vessel. Finally, the outlet of the heat exchanger was routed to discharge at the top of the chilled water vessel. The chilled water vessel comprised of a bucket filled with ice. The water, discharged into the top of the chilled water vessel from the exchanger, flows down through the ice to the bottom where it's drawn into the inlet line and sent back through the exchanger. DS18B20 submersible temperature probes connected to the UNO temperature controller were placed at the bottom of the chilled water vessel where the pump draws water for the exchanger, at the outlet of the reactor, and at the cooling water discharge after the exchanger. See comment in Methods: Activity 7 about temperature probe placement for reading vessel temperature.

Activity 4:

Step tests were performed on the temperature controller to determine initial controller tuning parameters. The level control system was enabled, and liquid level was held at 8 cm. A trial was performed with the hot plate at 50°C. The temperature control pump was set at PWM 110, and temperature was allowed to come to steady state. It was then increased up to PWM 150, and the temperature response was recorded. The temperature control pump setpoint was bumped back down to PWM 110 and response recorded. Figure 17 shows the response of temperature to pump setpoint change. It was noted that the system took more than ten minutes to come to a new steady state when temperature pump setpoint was decreased, so the hot plate setpoint was increased to 70°C so increase heat transfer into the vessel from the hot plate. The system was allowed to come to steady state, and the setpoint of the temperature control pump was increased from PWM 110 to PWM 150 and the response is shown in Figure 18.

Activity 5:

The data from the final step test was imported into Loop-Pro Trainer and fit with a First-Order-Plus-Time-Delay (FOPTD) transfer function (Equation 12). Using the fitted FOPTD response from Loop-Pro Tuner (Figure 19), the IMC method was used to derive a controller equation (Equation 13) from Equation 12. Next, the system was operated in closed loop using the derived controller tuning parameters. To test the system, the volume was held at 10 cm and hot plate set to 70°C. The controller was allowed to come to steady state at 9C and the setpoint was switched down to 6°C and then back to 9°C to analyze the impact of different initial conditions on the system. The data was run through Loop-Pro tuner and the controller was re-optimized to produce a faster response.

Activity 6:

Activity 5 was repeated with only 8 cm of level in the vessel to determine the impact of extra volume on the inability to hit setpoint. The hot plate was set to 70°C and the system was allowed to come to steady state at T = 10°C. The setpoint was bumped down to 8°C and the response was recorded. The higher setpoints compared to Activity 5 was due to inability to hit 6°C setpoint in Activity 5. At roughly t = 50 minutes, the temperature in the vessel and outlet line was checked with a digital instant read thermometer. Noting the large discrepancy, the outlet line was insulated. Further observation of the outlet probe reading was noted, and the run was terminated at t = 67 minutes. The recorded data is displayed in Figure 21, and shows the system was unable to reach its low temperature setpoint of 8°C.

Activity 7:

The temperature probe was removed from the outlet line and suspended centrally in the vessel as shown in Figure 7. Care was taken to assure the probe did not contact the copper pipe

of the cooling coil nor interfere with the ultrasonic level sensor reading. Activity 6 was repeated, and the setpoint was cycled between 10°C and 8°C after attaining steady state at the new setpoint. The results are displayed in Figure 22, showing successful temperature control.

Summary of Activities

The performed activities were designed to evaluate equipment performance, create a model of the system so that controller parameters could be determined analytically, and to evaluate the performance of the controlled system in both maintaining steady state and in response to changes in setpoint.

Data and Results:

Activity 1:

The spans of the pumps were determined by running them at their highest and lowest functional input voltage. The relationship between pump input voltage via controller PWM setpoint and pump flowrate is not linear, as demonstrated by the quadratic relationships in Figure 9 and Figure 10. The nonlinearity increases the overall order of the system and reduces the accuracy of the fitted FOPTD models. In the system, the magnitude of the gain in the response is dependent on pump initial steady state setpoint. Resultingly, fitted models and the controller equations derived from them are only relevant to the tested conditions due to system nonlinearity.

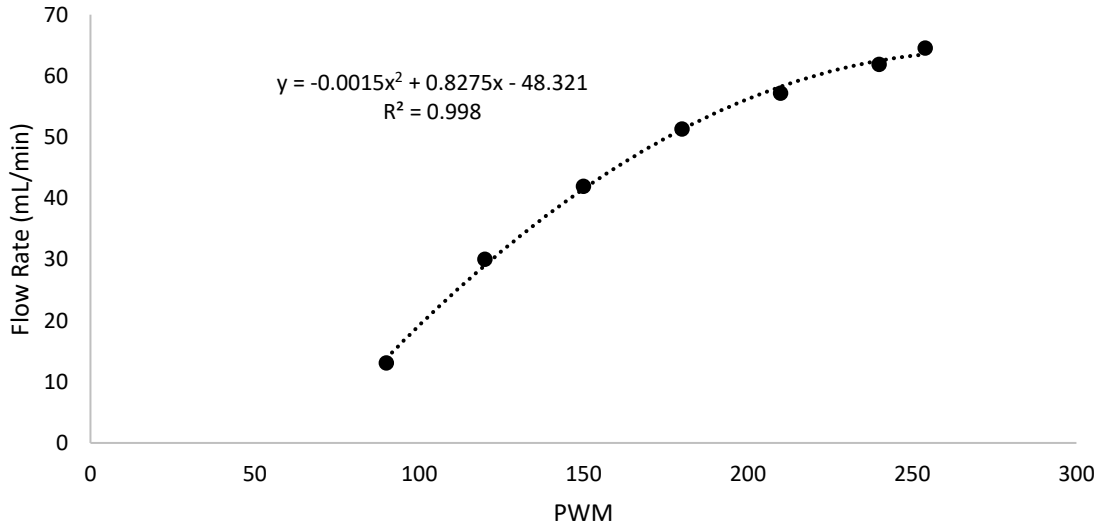


Figure 9: Pump curve generated from the peristaltic level pump relating the flow rate to the Pulse Wave Modulated (PWM) input.

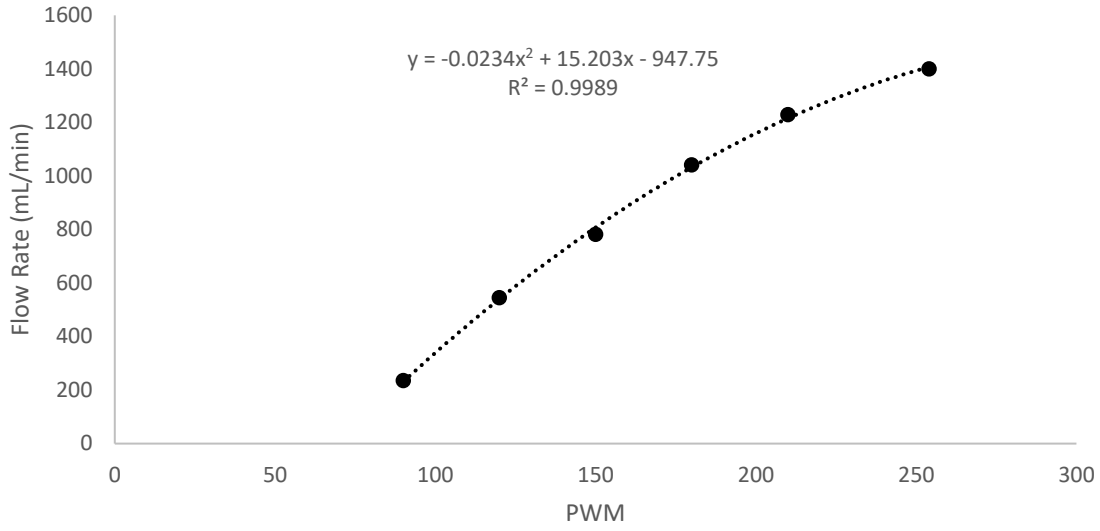


Figure 10: Pump curve generated from the diaphragm temperature control pump relating the flow rate to the Pulse Wave Modulated (PWM) input

Activity 2:

Activity 2 demonstrates that the behavior of liquid level in the tank is more akin to an integrating function, where there is only one steady state condition and deviation from that results in liquid level change. The cause for deviation from the analytically derived model (Equation 4) may be that the magnitude of outlet flow rate relative to liquid level change is not significant due to the large pressure drop of the stopcock. Due to inability to perform a step test, a controller was instead designed using rules-of-thumb for level control [19].

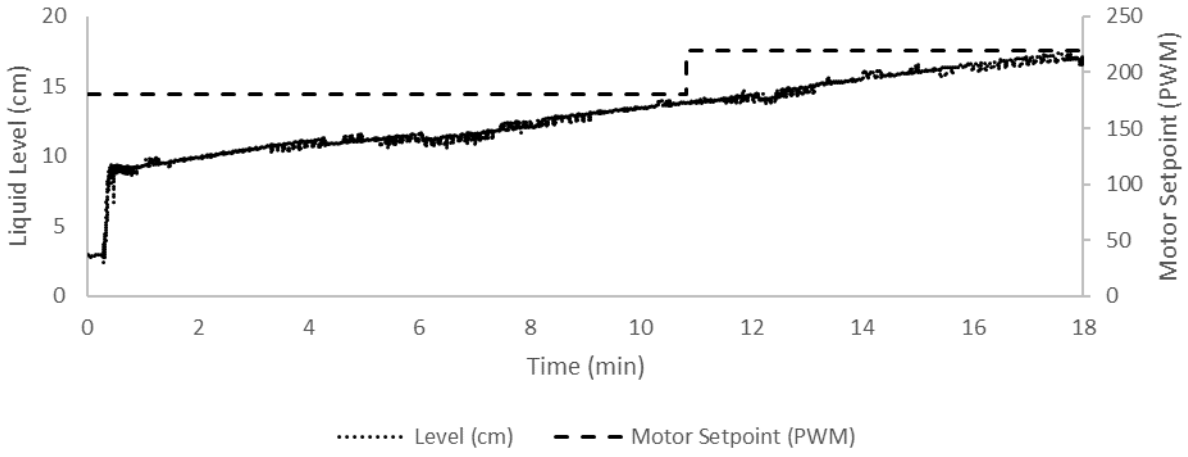


Figure 11: Attempted open loop step test of level controller. The liquid level setpoint was changed before steady state was achieved and the vessel overflowed.

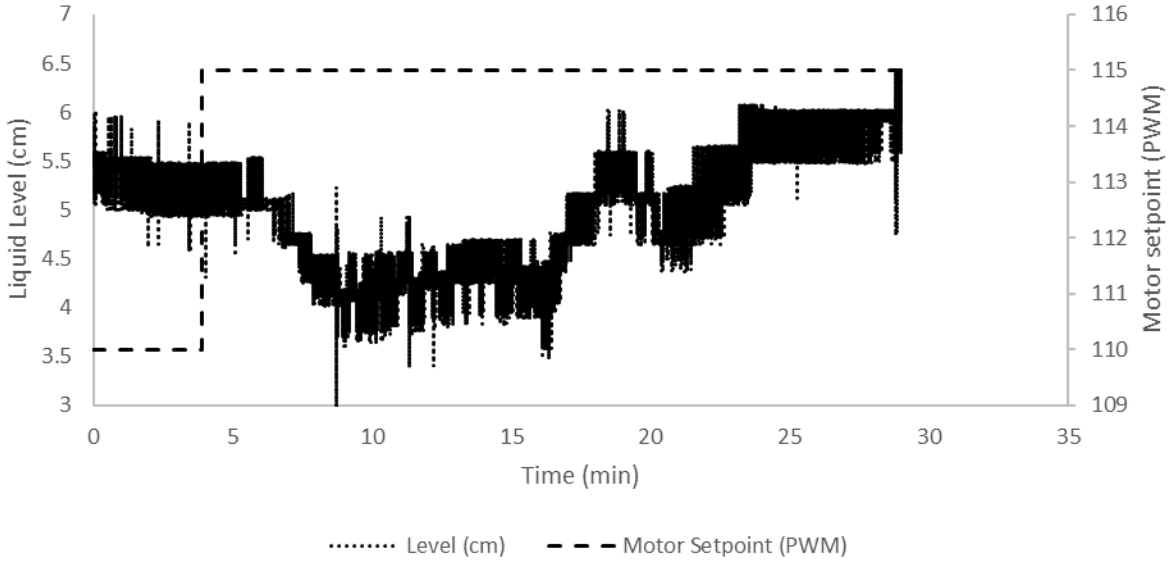


Figure 12: Second attempt to perform an open loop step test of the motor setpoint on the vessel level. The level in the vessel did not respond as expected from the change in setpoint, as it decreased when pump motor setpoint was increased. The cause for the response may have been due to the slow change in head height of the reagent feed container. For further level controller testing, the feed and effluent lines were put into the same vessel to mitigate this phenomenon.

Activity 3:

The performance of the liquid level controller was evaluated by its response to a step change and ability to maintain steady state. The first trial had undampened oscillations that persisted until the outlet valve was opened further. As seen in Figure 13 the system was then able to maintain setpoint and execute a setpoint change, but the high outlet flow rate relative to maximum pump flow rate resulted in a long time to complete the setpoint change (20 minutes) in respect to the systems intended purpose as a lab system with a quick response time. The noise in the measurement data also caused large proportional responses to outlier values, so a low pass filter was implemented in the measurement program. To combat the oscillating behavior, the controller proportional and integral gains were both reduced. The magnitude of setpoint changes to be performed were reduced and the outlet flow rate was adjusted to be more central in the level feed pump span. Figure 14 shows that the changes were success full in reducing oscillation and

decreasing time to complete setpoint changes. The responses were fit with FOPTD transfer functions to provide a means of evaluation in Figure 15 and Figure 16.

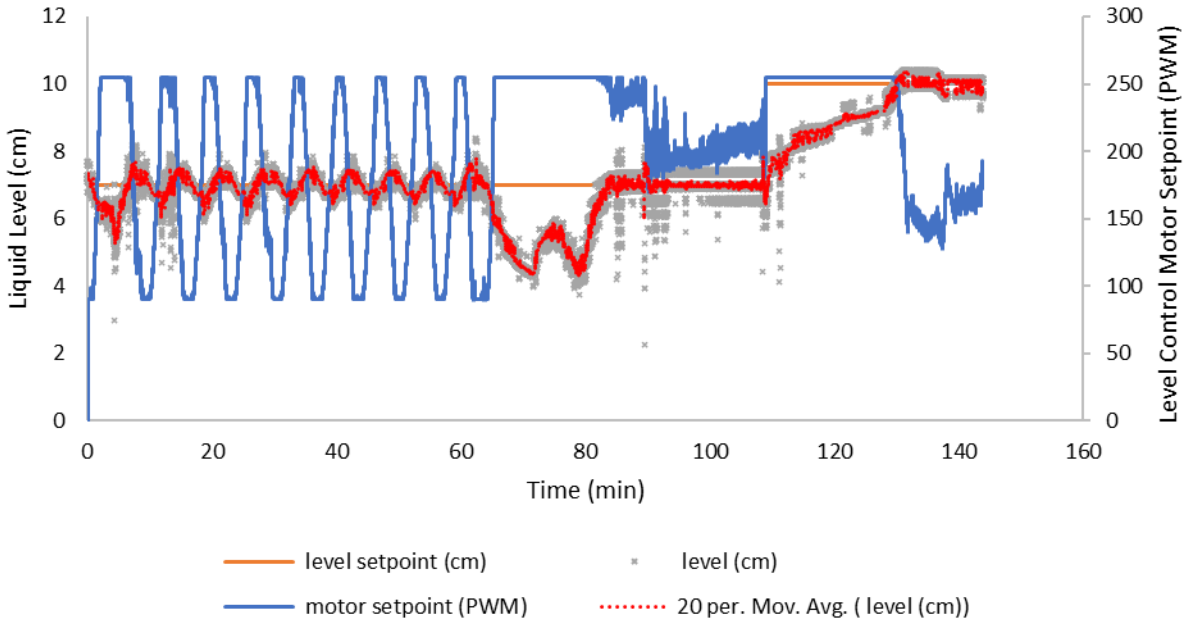


Figure 13: Closed loop system test of the level controller. $K_P = 50$, $K_I = 0.33$ ($\tau_I = 3$) From 0-60 minutes, level control was observed, but there appeared to be undamped oscillation. At 60 minutes, the response to a change to deviation variable (outlet valve opened wider), was observed until new steady state achieved. At t=110 minutes, setpoint was changed from 8 to 10 cm, and the response was observed. A 20 point moving average line was fitted to the line to smooth level readings.

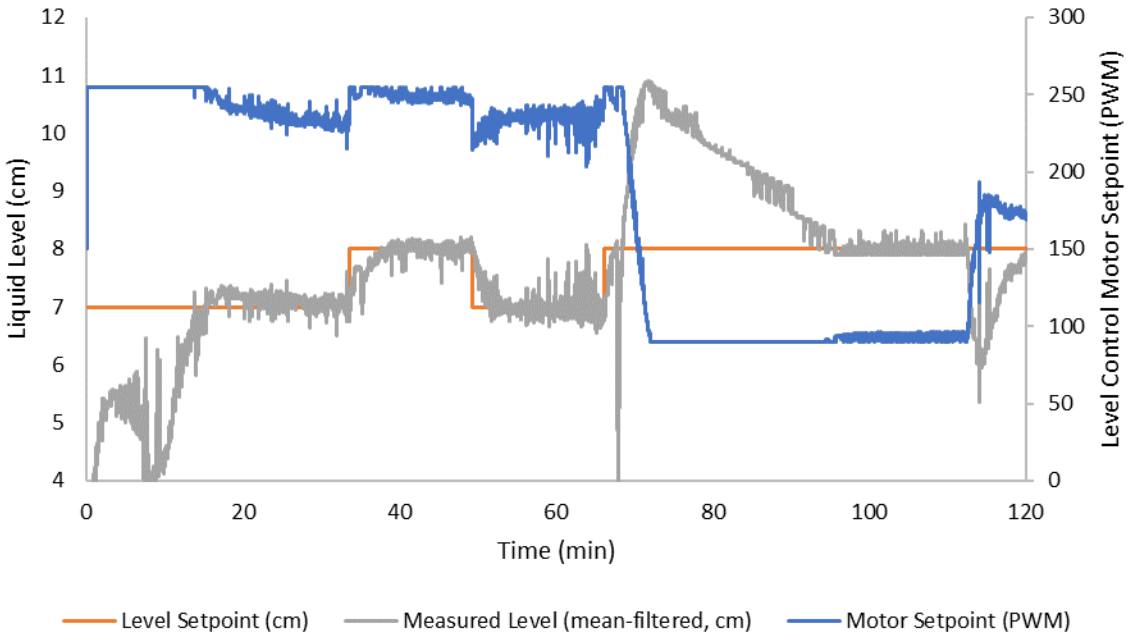


Figure 14: Closed loop system test of the level control system. K_P=30, K_I = 0.2 (τ_I=5). The level data had a 20 point mean filtered applied in the code. The vessel was allowed to fill, then the setpoint was changed from 7 to 8 cm, allowed to come to steady state, and changed back down to 7 cm. After steady state was achieved, response to disturbance was tested by restricting the flow through the outlet from 50 to 20 mL/min. Note: the low level reading at t=67 min was caused by an unintentional disturbance to the level sensor. It was repositioned, and proper readings resumed.

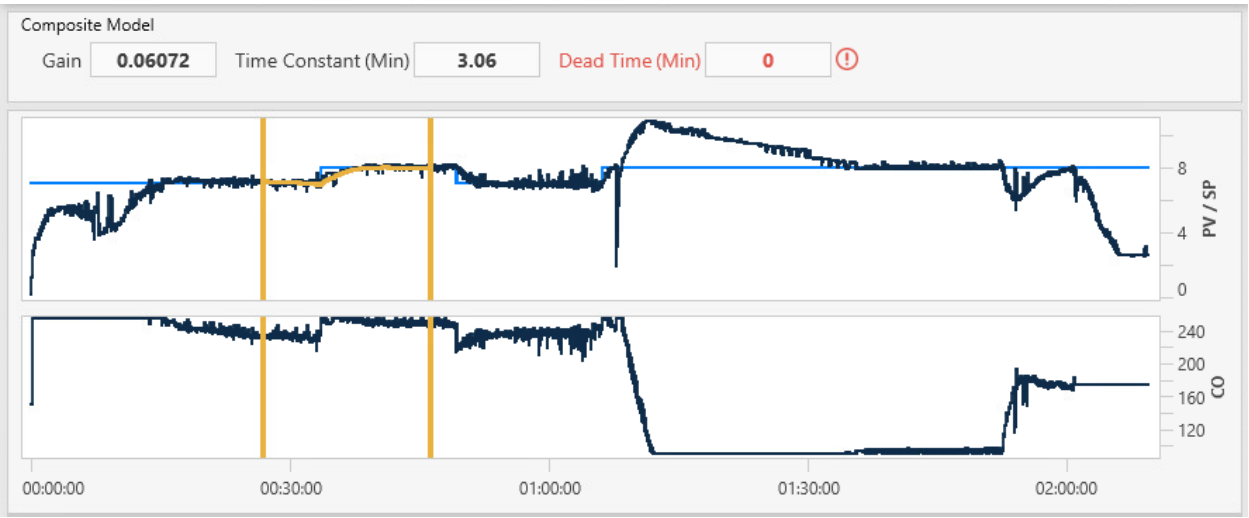


Figure 15:Closed loop transfer function of the level control loop with K_P = 30 and K_I = 0.2 for setpoint increase.

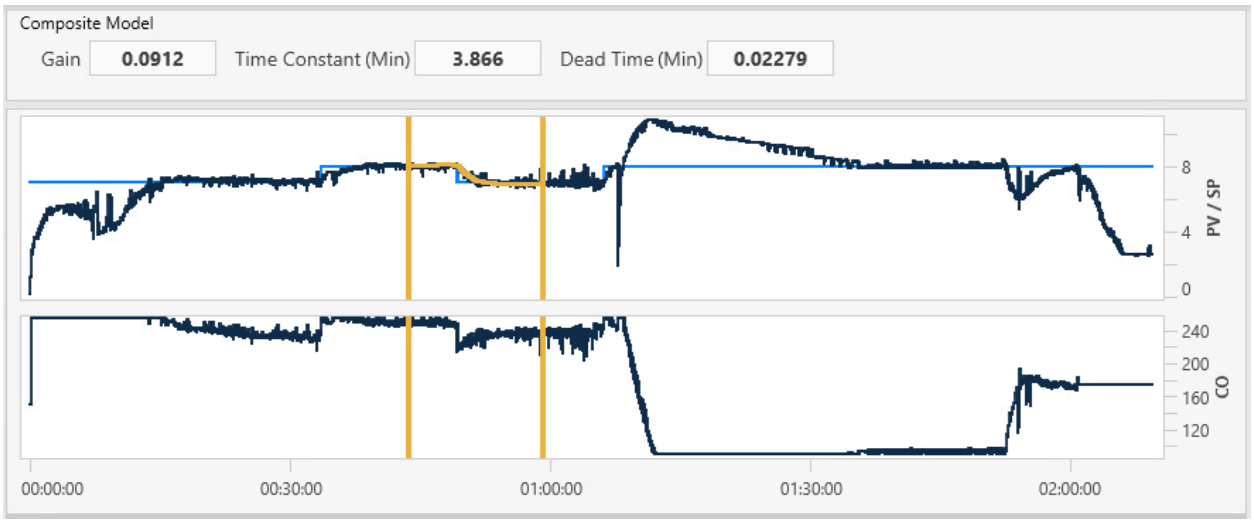


Figure 16:Closed loop transfer function of the level control loop with K_P= 30 and K_I = 0.2 for setpoint decrease.

Activity 4:

The outlet temperature response to decrease in heat exchanger cooling water flow rate, seen in Figure 17, appeared to be first order. The response to an increase in setpoint lacked a smooth first order change, so a second trial was performed. The second trial (Figure 18), performed at a higher hot plate setpoint, exhibited a more ideal response and was fit with a model (Figure 19) for controller design purposes. The hot plate temperature setpoint was increased to show the system could maintain a steady state with more heat being added to the system, simulating a more exothermic reaction. Increasing the themal load demonstrated that the system was able to find a new steady state with less than 1°C deviaiton and provide protection against runaway reactions.

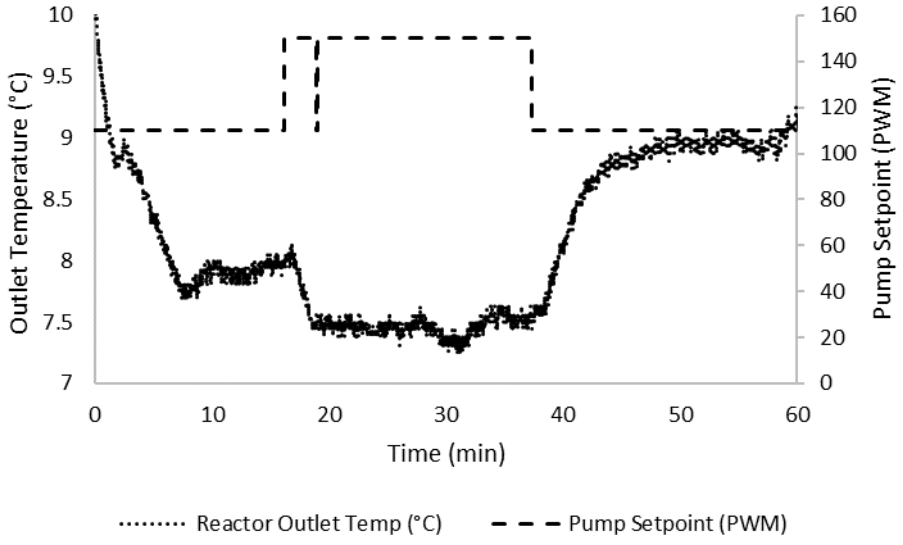


Figure 17: Activity 4 step testing. Hot plate set to 50°C, level at 8 cm.

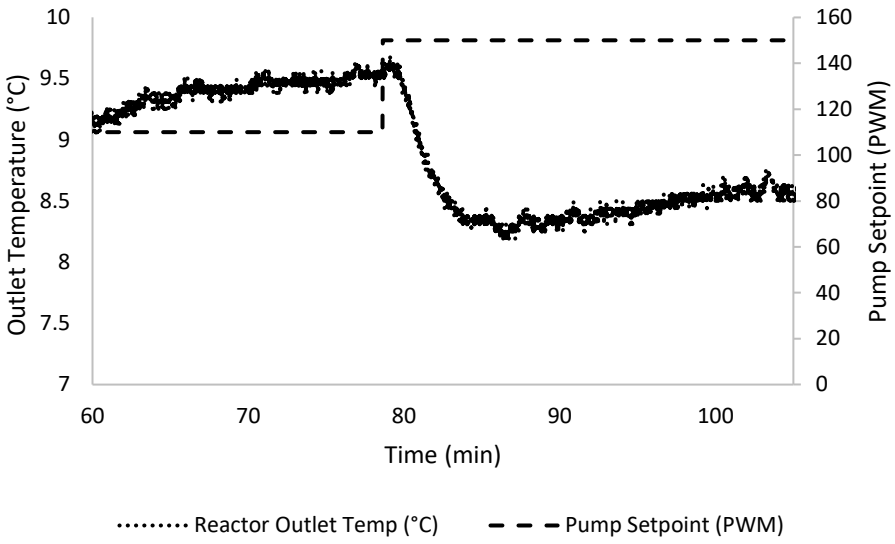


Figure 18: Activity 4, temperature response to step change in pump PWM setpoint. Hot plate at 70°C, level at 8 cm.

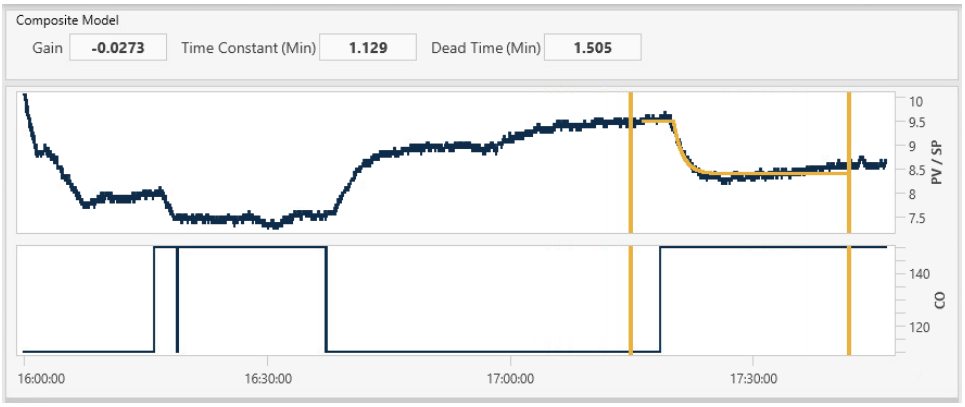


Figure 19: Loop-Pro fit first order response to upward change in temperature pump setpoint seen in Figure 18. Hot plate at 70°C, level at 8 cm.

Equation 12 shows the First-Order-Plus-Time-Delay (FOPTD) transfer function fit to the process data in from Activity 4 shown in Figure 18 and Figure 19. Equation 12 relates the manipulated variable (heat exchanger cooling water pump PWM setpoint) to the temperature of the CSTR outlet for step increase with hot plate at 70°C, level at 8 cm, and allows for the design of a controller using the Internal Model Control (IMC) method.

$$T^*(s) = \frac{-0.0273e^{-1.505s}}{1.129s + 1} * P^*_{PWM}(s) \tag{12}$$

Activity 5:

The temperature controller designed using the results of Activity 4 was able to maintain its high setpoint of 9°C, but was not able to reach the low setpoint of 6°C. The system maintained the high setpoint with ~105 PWM output, but was unable to reduce the temperature to the low setpoint using the rest of the available cooling water pump span. The liquid level in the vessel was 2 centimeters greater than in Activity 4, and it was hypothesized that if the level was returned to the conditions in Activity 4, then the performance of the control system would better reflect the data from Activity 4.

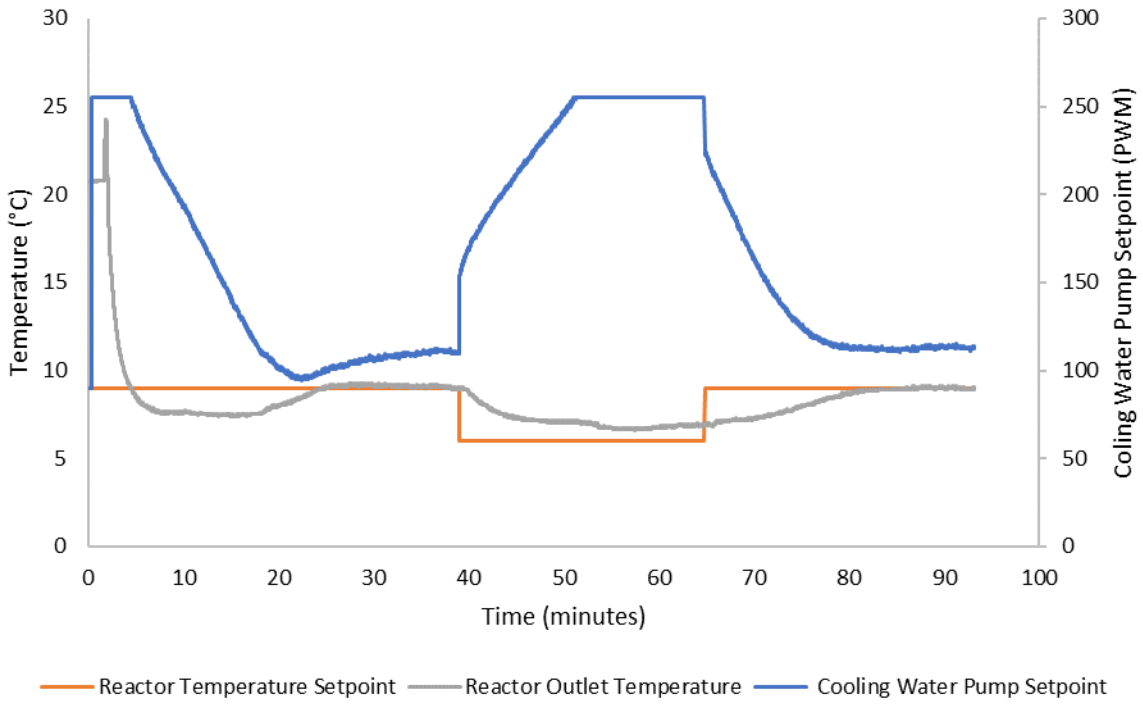


Figure 20: Temperature control during Activity 5. The temperature measured was unable to reach the low setpoint of 6˚C despite 100% heat exchanger cooling water pump motor output for 15 minutes.

Activity 6:

Despite the changes implemented, the system was still unable to reach the lower setpoint and the hypothesis of increased liquid level causing the poor controller performance was proven false.

During the experiment, the temperature in the vessel was checked with a digital thermometer and it was discovered that it was 5°C less than the measured value at the end of the outlet line. Insulating the pipe did not appear to impact the change significantly. A new hypothesis was postulated; if the temperature reading was taken directly from the vessel, then the low setpoint could be achieved in Activity 6.

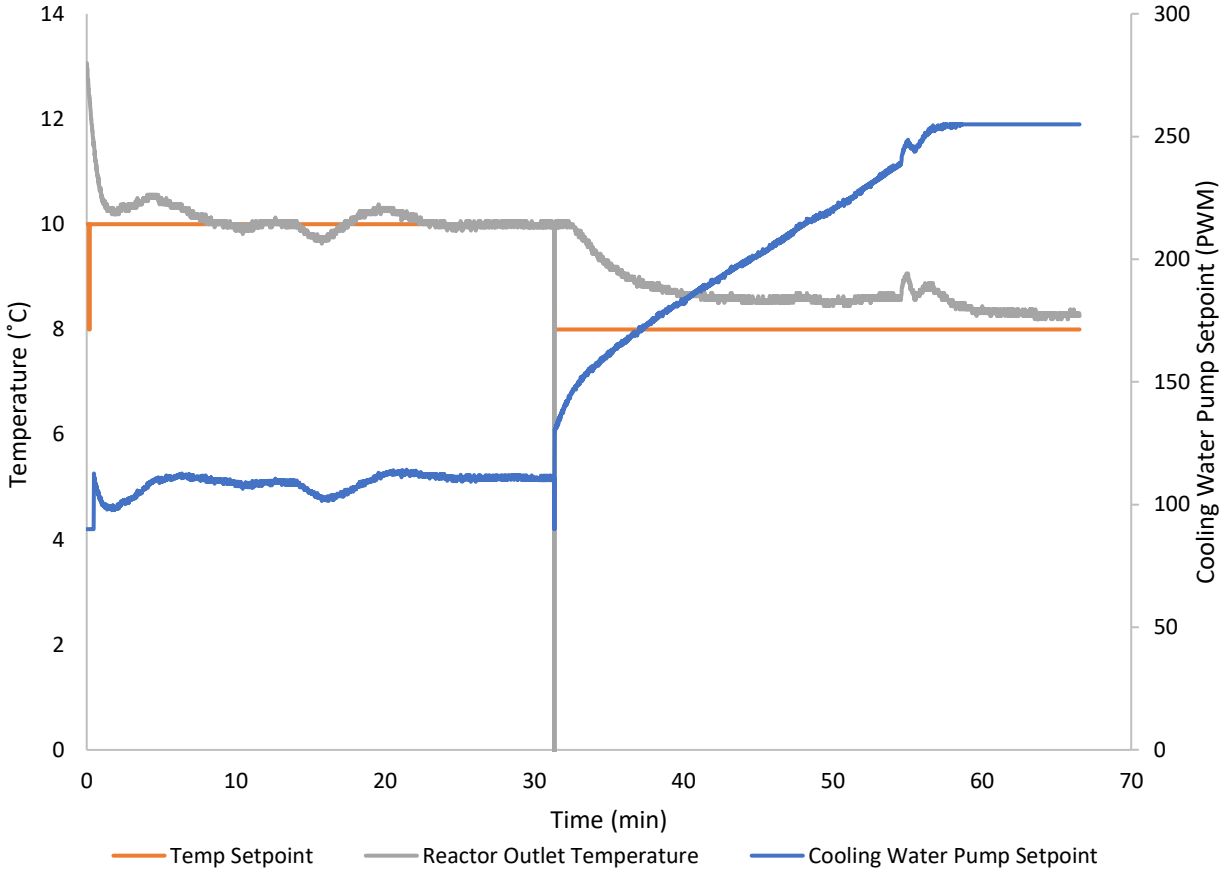


Figure 21: Temperature control during Activity 6. The temperature measured was unable to reach the low setpoint of 8°C despite 100% motor output about 30 minutes after setpoint change. Insulating the outlet line at t = 55 minutes had minimal effect. The experiment was terminated without performing a setpoint increase.

Activity 7:

The temperature control system derived from Activity 4 was able to maintain setpoint and quickly execute setpoint changes after moving the measurement point to the interior of the vessel. The hypothesis postulated in Activity 6 was confirmed. The reactor temperature control

system had a closed loop time constants of ~1.3 minutes and was able to execute setpoint changes with less than 15% overshoot.

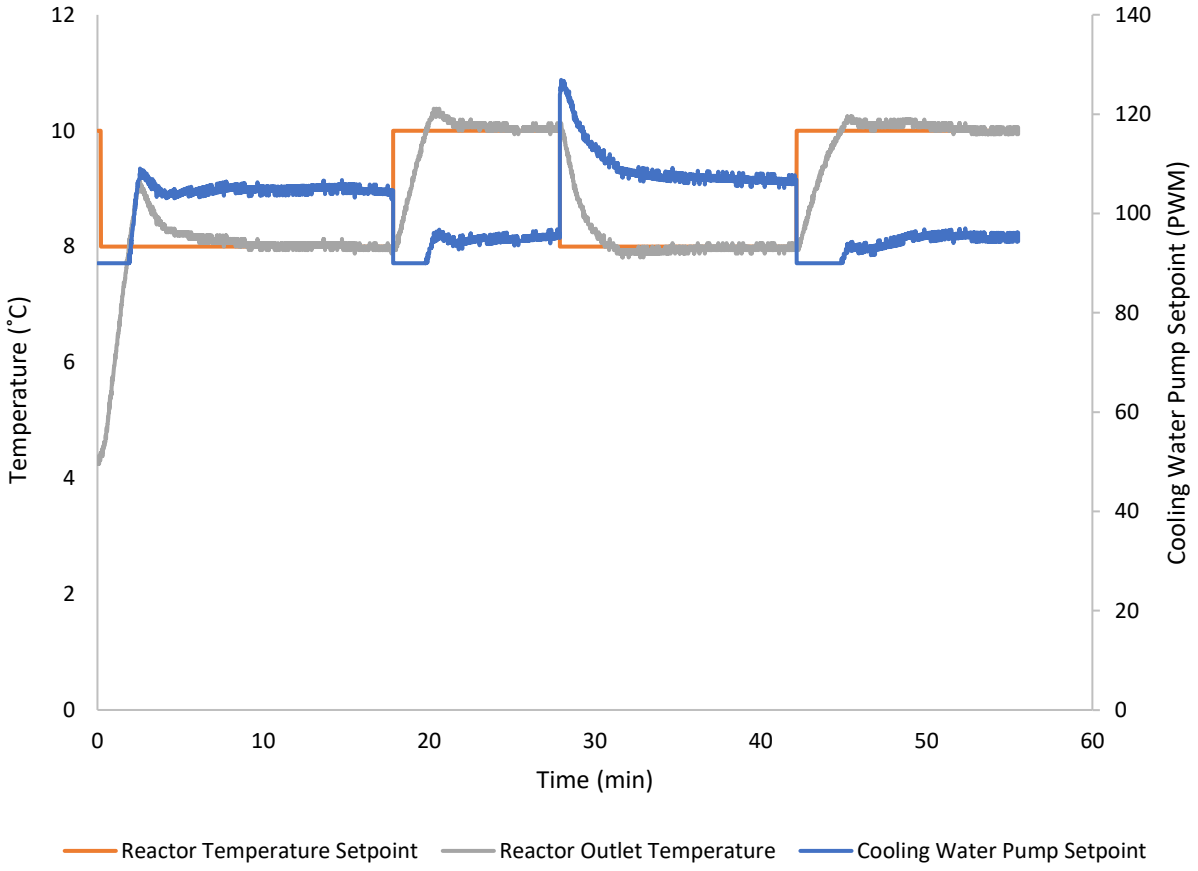


Figure 22: Temperature response to setpoint change in Activity 7, after moving the probe to the vessel interior. It is noted that the system was able to reach setpoint using only a third of the pump setpoint span.

Discussion and Analysis:

Program Development

The code for each control element was written individually and tested, prior to being transferred into the main program. Upon transfer to the main program, elements often needed further debugging due to interactions with other systems. Programs to control the system were continually updated as needed and the attached code reflects the most recent version of each program.

One of the evolutions undergone by the control system was the inclusion of a second Arduino to handle temperature measurement and control to reduce the impact of measurement delay from temperature readings on the level control system. The total program execution time for one reading of the level controller was observed to be about 50 milliseconds (ms). The temperature probes, which operated using the OneWire [14] and DallasTemperature [15] libraries to communicate with the Arduino on a digital pin, took 600 ms to give a temperature back to the control system. To minimize the effects of the larger delay on the level controller, the temperature control functionality was removed from the code for the Arduino Mega and given to an Arduino Uno.

Liquid Level Activities

There were several difficulties in the level step tests and controller tuning (Activity 2). The first issue to be remediated was the significant noise in the reading of vessel level by the HC-SR04 ultrasonic sensor. During data analysis of Activity 3, a 20-value moving average low pass filter was applied to the data to smooth it during the analysis phase. To reduce the controller reaction to outliers during operation, the level control program was altered to include a 20-point moving average data smoothing function (about one second of data readings). After introducing the data smoothing function during measurement in the second trial of Activity 3, the impact of high and low outliers was reduced on the level control system, and the manipulated variable setpoint was more stable. A median filter was considered, but ultimately ruled out due to ease of implementation of the mean filter and the difficulty in creating a computationally efficient median filter.

For level control, steady state conditions could not be achieved in open loop with the equipment available, shown in Figure 11. The level pump span was determined to be 13 – 65

mL/min as seen in Figure 9, so the reactor outlet flow rate was had to be within that span for level control to be achieved. In order to provide both the fastest setpoint increase and decrease response, the reactor outlet was set as close to the middle of the pump span as possible at 40 mL/min. In execution, drain speed varied between 35 – 45 mL/min. The reactor outlet had a significantly higher flow rate ceiling at 400 mL/min drain rate, so closing it enough to get the proper outlet flow rate required multiple trial-and error attempts and was prone to operator error. The use of a globe valve, which has a more linear relationship between flow and valve position [7, p. 200], might remediate this problem by providing finer control of the outlet flow rate. For the available vessel, a stopcock with a smaller diameter hole could provide increased accuracy in achieving lower flow rates.

Reactor Liquid Level Control

There was no data available describing the transition between two steady-states of vessel level due to a change in the feed pump setpoint because of the integrating behavior of the system observed in Activity 1. In the absence of a process model, rules of thumb were used to provide initial tuning setpoints and the system was tested in closed loop mode. The rule of thumb, that systems where the manipulated variable (i.e., the feed flow) causes a very slow reaction in a process variable, the controller gain is usually 1 – 100 and the integral gain is 0.05 – 0.3 [19]. As such, values of $K_P = 50$ and $\tau_I = 3$ ($K_i = 0.33$) were used for the first trial of closed loop testing. The resulting response data is shown in Figure 13. While the vessel level was controlled at setpoint, the heavy oscillation due to integral wind up and high gain indicated that the magnitude of controller response to error had to be reduced. For the second closed loop setpoint test, seen in Figure 14, K_P was reduced to 30 and τ_I increased to 5 ($K_I = 0.2$). The response seen appeared to be first order, with a time constant of 3.06 for a setpoint increase (Figure 15) and 3.86 minutes

for a setpoint decrease (Figure 16). The discrepancy in time constants indicates that the level control open loop equation is non-linear, thus the response depends on the initial steady state of the system. Further differences in filling speed compared to draining speed comes from the imperfect positioning of the drain speed within the center of the fill pump span. If the drain rate was faster than the midpoint of the pump span, then the maximum volumetric flow rate of draining would be faster than the maximum volumetric flow rate of filling the vessel, and vice versa. The resulting system would exhibit nonlinear responses that aren't within defined by the process model.

Reactor Temperature Control

Temperature control open loop step testing proceeded successfully, shown in Activity T-1 (Figure 17). FOPTD responses were fit to the process data in Loop-Pro [17] (Equation 12). The differences in response from a setpoint increase and decrease demonstrate that the process is non-linear. The activity was repeated with the hot plate at a higher setting to simulate more heat from the reaction and increase the rate of heating at low pump setpoints, shown in Figure 18. The data was reformatted to fit the specifications of Loop-Pro Tuner [17] and a controller equation (Equation 13) was fit to have less than 10% overshoot. The response 70°C was used for the controller equation and moving forward due to concern that the lower setpoint would limit the span of achievable temperatures in the vessel due to the relatively fast minimum flow rate of the temperature control pump (235 mL/min) compared to the rate of reagent feed (~30 – 40 mL/min) and to demonstrate the heat exchanger system can remove more heat from the reactor.

$$\mathrm{Gc} = 14.41 + 0.67 * \frac{1}{\mathrm{s}} \tag{13}$$

Activity 5 was intended to demonstrate control over the system using the controller equation (Equation 13), but the results shown in Figure 20 indicated that the system was not responding as predicted to changes in setpoint. Responses were noted to be sluggish relative to the process time constants determined in Activity 4 step testing, and when analyzed in Loop-Pro Trainer, the observation was analytically confirmed. The time constants for the closed loop system were on the order of 10^1 for a setpoint increase to 10^3 minutes for a setpoint decrease, compared to the process time constant of 1.48 minutes. Activity 6 was then designed to determine the root cause of the sluggish response, eliminating discrepancies between the step tests in Activity 4 and the control testing in Activity 5.

Activity 6 was intended to be a repeat of activity 5 where the level was reduced to 8 cm, to align with the data from Activity 4. During Activity 6, it was noted that there was only a small (<1°C) difference between the heat exchanger inlet and outlet temperatures, indicating that limited heat transfer between the vessel and heat exchanger was occuring. A measurement of the outlet line and the vessel was taken, and it was found that the temperature at the outlet of the CSTR line was approximately 5°C hotter than the temperature in the tank. The outlet line was insulated with a towel to attempt to mitigate the effects of heat transfer into the line. It was hypothesized that the difficulty in attaining fast responses in Activities 5 and 6 can be attributed to heating of the reactor effulent en route to the temperature probe. To test the hypothesis that the temperature response will be quicker if the probe was put directly into the vessel, the activity was repeated under those conditions.

In Activity 7, the temperature probe placed inside the reactor vessel rather than on the product outlet line produced significantly faster and more accurate results, leading to better control of the temperature in the vessel overall. The hypothesis proposed after Activity 6 was

proven to be correct. The response of the process had a closed loop time constant of 1.3 minutes and a maximum overshoot of 15% was observed during the run (~0.3°C).

Conclusion:

The objective of the project was to create a chemical reactor system capable of maintaining its level and temperature without operator intervention for use in CSTR experimentation. Building such a control system required knowledge in programming, electronics fabrication, and process control system analysis. Despite initial difficulties in each system, both level and temperature control systems were successfully developed that were able to control the process to setpoint and execute setpoint changes. The operating parameters of $K_P = 30$ and $K_I = 0.2$ for level control and $K_P = 14.41$ and $K_I = 0.67$ for temperature control provided reasonably fast responses with some overshoot in response to a setpoint change. These responses are appropriate for lab functionalities without an inline separations process. Using the control systems, the reactor would be able to safely maintain vessel level, temperature, and record significant operating parameters for later analysis. Thus, the viability of the system as a controlled CSTR system was demonstrated.

Recommendations For Future Work

Reaction Analysis

Prior to operation with reagent, it is recommended that the system be wet-tested with water and a hot plate at the design setpoints to simulate real process conditions. A thorough evaluation of the system, the reaction, and potential deviation variables must be completed to reduce risk coming from operation outside of proven ranges. In high temperature applications, the chilled water in the temperature control system could be replaced with normal process water,

but effective heat transfer is dependent on a large average temperature difference between the vessel volume and heat exchange so the impacts of making this change on the system's ability to remove excess process heat would need to be assessed.

Reaction Completion Sensing

For the system's intended purpose as a reactor system, it is recommended that another analytical element be added to the control system: a product concentration sensor. Direct monitoring of outlet product concentration would allow feedback control by means of manipulating the temperature or the level in the vessel. In a chemical reactor, it is important to attain steady state operation with a constant stream composition as downstream separations operations can be greatly impact by small disturbances, which propagate as the deviated material moves through more process units.

Liquid Level Data Filter

The level controller was severely impacted by noise from the ultrasonic sensor. To alleviate the problem, a 20-point mean filter was implemented to reduce the impact of single outliers on the proportional aspect of the controller. A mean filter is simple to program and has an efficiency of O(1) (number of computations is fixed), but it does not remove the impact of outliers completely; it only spreads them over 20 data points, so they still affect the integral term of the controller equation. A median filter would completely remove the impact of outlier readings but is less efficient as it requires data sorting which has a best average efficiency of $O(n^2)$ (the number of computations is related to the square of the number of data items). The benefits to control would need to be weighed against the increase in processing time per data

read operation and potential impact on process transfer functions from further manipulating the data.

Product Outlet Revision

The span of outlet flow rate (0-400 mL/min) is significantly greater than the pump span, and as a result sensitivity to outlet stopcock positioning on the vessel caused difficulties in creating identical conditions between different runs if the stopcock was bumped or intentionally manipulated. To alleviate this problem, the diameter of the hole in the stopcock can be significantly reduced or a valve with a more linear flow characteristic like a globe valve can be used. The use of an outlet flowmeter could reduce the time in outlet flow adjustments, but the low flow rate may cause financial barriers in finding an appropriate instrument. An alternative solution would be to use a pump with higher flow rate and then operating the entire system at a higher flow rate, but this greatly compromises the maximum product concentration attainable by the system by reducing the maximum space time at which the system can operate.

Pump Choice and Control

The span of the pumps was limited to 48% of their 12V nominal voltages due to high end losses from voltage drop and insufficient torque to rotate at low voltages under PWM 90. The L298N H-bridge motor driver suffers from a voltage drop of 2 volts from the power source to the motor terminals due to the H-bridge functionality. As reversing motor direction is not a necessary functionality in this application, the L298N could be replaced with a less parasitic system and attain a higher high-end voltage. A 14V power supply could also be used to achieve 12V at the motor terminals. Changing the motor driver to a higher quality component would help slightly with the low-end losses, but to have a full 0-255 PWM span, superior pumps with gearboxes

would be required to maintain rotation at low voltages. The current system is functional, but later user application may necessitate a larger span in cooling water flow rate of feed flow later.

References

[1] L.-K. Ju, *Class Lecture, Topic: Enzyme Reactors,* Akron: CHEE 473 Bioreactor Design, The University of Akron, 2023.

[2] H. S. Fogler, Essentials of Chemical Reaction Engineering, Indiana: Pearson Education, Inc., 2016.

[3] D. Crowl and J. Louvar, Chemical Process Safety, 4th Edition ed., Prentice Hall, 2019.

[4] D. Seborg, Process Dynamics and Control, Hoboken,NJ: John Wiley & Sons, Inc., 2016.

[5] Arduino, "Downloads," Arduino, [Online]. Available: https://www.arduino.cc/en/software. [Accessed 16 04 2023].

[6] T. Hirzel, "Basics of PWM (Pulse Width Modulation)," Arduino, 9 March 2023. [Online]. Available: https://docs.arduino.cc/learn/microcontrollers/analog-output. [Accessed 12 March 2023].

[7] W. L. McCabe, Unit Operations of Chemical Engineering, New York, NY: McGraw-Hill, 2005.

[8] Tapflo UK, "Peristaltic Pump Guide," [Online]. Available: https://www.tapflopumps.co.uk/blog/peristaltic-pump-guide/. [Accessed 23 03 2023].

[9] F. Petruzella, Programmable Logic Controllers, McGraw-Hill, 2022.

[10] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Journal of Process Control,* vol. 13, no. 4, pp. 291-309, 2003.

[11] I. L. Chien and P. S. Fruehauf, "Consider IMC Tuning to Improve Controller Performance," *Chemistry Engineering Process,* vol. 86, no. 10, pp. 33-41, 1990.

[12] S. Tatham, "PuTTY: a free SSH and Telnet client," 29 10 2022. [Online]. Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/. [Accessed 37 03 2023].

[13] B. Beauregard, "Arduino-PID-Library," 20 January 2021. [Online]. Available: https://github.com/imax9000/Arduino-PID-Library. [Accessed 20 March 2023].

[14] P. Stoffregen, "OneWire Library," 4 June 2022. [Online]. Available: https://www.pjrc.com/teensy/td_libs_OneWire.html. [Accessed 20 March 2023].

[15] M. Burton, "Dallas Temperature Control Library," Arduino, 15 01 2016. [Online]. Available: https://www.milesburton.com/w/index.php/Dallas_Temperature_Control_Library. [Accessed 18 April 2023].

[16] Microsoft, "Microsoft Excel," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/excel. [Accessed 16 04 2023].

[17] Control Station, "Loop-Pro Trainer," [Online]. Available: https://controlstation.com/education/. [Accessed 17 04 2023].

[18] keyestudio, "052043 Super Learning Kit for Arduino," Micro Electronics, Inc., 26 April 2020. [Online]. Available: https://wiki.keyestudio.com/052043_Super_Learning_Kit_for_Arduino. [Accessed 24 January 2023].

[19] Cross Company, "How to tune a PID loop," 2022. [Online]. Available: https://www.crossco.com/resources/technical/how-to-tune-pid-loops/. [Accessed 26 March 2023].

[20] Arduino, "What is Arduino?," Arduino, 5 February 2018. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed 12 March 2023].

[21] T. Hirzel, "Basics of PWM (Pulse Width Modulation)," Arduino, 09 03 2023. [Online]. Available: https://docs.arduino.cc/learn/microcontrollers/analog-output. [Accessed 23 03 2023].

Appendices:

Appendix A: Level Control PI Code

```
/*

   COPYRIGHT (C) 2023 Cameron Macesich (cwm40)

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify,
merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be included in all copies
or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE
OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


   Honors Project: Level and Temperature Controlled Vessel

   Author.  Cameron Macesich

               cwm40@uakron.edu

   Version. 1.01 07APR23

   Purpose: This program is for an Arduino Mega 2560 for use as a level controller.
The level controller will hold the level to a user defined setpoint and reads user
input from a 4-button panel. The yellow button can perform a user defined function.

*/


#include <PID_v2.h>


//motor

#define motorPin 7  //Motor control pin, PWM, level

#define motorIN1 6  //level Dir N1 on L298N

#define motorIN2 5  //Level Dir N2 on L298N


//level indicator HC-SR04

#define echoPin 34  // Echo Pin

#define trigPin 30  // Trigger Pin
```

```
//bumpSwitch
#define bumpSwitch 42  //input to read yellow button. NORMALLY OPEN
#define start 41       //input read green start button. NORMALLY OPEN
#define stopBtn 43     //input read stop button. NORMALLY OPEN
#define eStop 40       //Read the estop button. NORMALLY CLOSED

#define com1 22  // Tell temp controller state of bumpSwitch
#define com2 23  // Tell temp controller to start/stop

//Ultrasonic Level Sensor HC-SR04
int maximumRange = 200;    // Maximum range needed
int minimumRange = 0;      // Minimum range needed
float duration, distance;  // Duration used to calculate distance
double level = 0;

//Boolean Flags for system operation
bool switchFlag = 0;
long tSwitch = 0;
bool startFlag = 0;
bool stopFlag = 0;

//Setpoints for motor
double motorSP = 110;
int alarmSP = 0; //For when the level is too high send motor this value as setpoint

//USER ENETERED VALUES
double levelSP = 8;          //cm; The PI controller will hold vessel level to this
value
float sensorHeight = 18.37;  //cm; NEEDS TO BE ZEROED by the user. Should read 0.0 on
empty vessel

//Declaration for PI controller object
//Default: Kp=30, Ki=0.2 for level control
PID levelPID(&level, &motorSP, &levelSP, 30, 0.2, 0, DIRECT);
```

```
//Average filter: Array to read the Level into and smooth it by computing the average
value over 20 readings.
double levelRead[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};
int readCtr = 0;      //index of read variable
double sumRead = 0;   //Holds the sum of all values in array
double avgLevel = 0;  //holds the average level


void setup() {
  // put your setup code here, to run once:
  Serial.begin(19200);

  //Level sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  //Level Control Pump
  pinMode(motorPin, OUTPUT);
  pinMode(motorIN1, OUTPUT);
  pinMode(motorIN2, OUTPUT);

  //Button Panel
  pinMode(bumpSwitch, INPUT);
  pinMode(start, INPUT);
  pinMode(stopBtn, INPUT);
  pinMode(eStop, INPUT);

  //communication with temp controller
  pinMode(com1, OUTPUT);
  pinMode(com2, OUTPUT);

  //Send headers for data to computer
  Serial.println();
```

```
  Serial.println("time, motor setpoint, level setpoint, level");


  //Send level pump direction to L298N
  digitalWrite(motorIN1, HIGH);
  digitalWrite(motorIN2, LOW);


  levelPID.SetOutputLimits(90, 255);  //motor goes beep-beep below PWM 90 with 12 V power source
}
void loop() {
  // put your main code here, to run repeatedly:
  //read the buttons to determine running state
    //if any stop button is triggerd OR system is already stopped, reamin true. Seal in circuit broken by start button
    stopFlag = (!digitalRead(eStop) || digitalRead(stopBtn) || stopFlag) && !digitalRead(start);
    //if start buttun pushed and no stop buttons are pushed, start system and stay running. Seal in circuit broken by stop buttons
    startFlag = (digitalRead(start) || startFlag) && !digitalRead(stopBtn) && digitalRead(eStop);
  //Tell the temp controller what the running state is
  if (startFlag) {
    digitalWrite(com2, HIGH);
  } else {
    digitalWrite(com2, LOW);
  }
  //Bump the change the state of the user designated button, only allow for switch every 1 second to avoid double reads
  if (!switchFlag && (digitalRead(bumpSwitch)) && ((millis() - tSwitch) > 1000)) {
    switchFlag = true;
    tSwitch = millis();
  }
  if (switchFlag && (digitalRead(bumpSwitch)) && ((millis() - tSwitch) > 1000)) {
    switchFlag = false;
    tSwitch = millis();
```

```
  }


/* USER DESIGNATED FUNCTION
This will perfrom some user defined action depending on the state of the yellow
button.
Can be used to execute a setpoint change.
*/
  if (switchFlag == true)  //tell temp controller state of user designated variable
and perform user defined function
  {
    digitalWrite(com1, HIGH);
    //ADD STATE HERE
    //example: levelSP =10;
  } else {
    digitalWrite(com1, LOW);
    //ADD STATE HERE
    //example: levelSP =8;
  }


  //Read the current level, compute new average level
  addValAvg(levelRead, sensorHeight - getDistance(), avgLevel, sumRead, readCtr);
  level = avgLevel;


  if (startFlag) {
    if (level > (16))  //high-high Level, send shut off signal to level pump
    {
      levelPID.SetMode(MANUAL);
      analogWrite(motorPin, alarmSP);
    } else  //normal PID response
    {
      levelPID.SetMode(AUTOMATIC);
      levelPID.Compute();
      analogWrite(motorPin, motorSP);
    }
```

```
  } else  //If startflag is false, send shutoff signal to motor
  {
    levelPID.SetMode(MANUAL);
    analogWrite(motorPin, 0);
  }
  //Print the time, motor setpoint, level setpoint and current level to serial for
PuTTY with comma to separate them
  Serial.print(millis());
  Serial.print(", ");
  Serial.print(motorSP);
  Serial.print(", ");
  Serial.print(levelSP);
  Serial.print(", ");
  Serial.println(level);
}
//return the distance of the surface from the sensor as a floating point number
float getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);

  //Calculate the distance (in cm) based on the speed of sound.
  distance = duration / 58.2;

  if (distance >= maximumRange || distance <= minimumRange) {
    /* Return a negative number to indicate "out of range" */
    delay(50);
    return -1;
```

```
  } else {
    //Delay 50 to let echo die down then return distance
    delay(50);
    return distance;
  }
}
//This function reads a value into the array and calculates the average value of the
numbers in the array
void addValAvg(double arr[], double newVal, double &avgVal, double &sumArr, int &ctr)
{
  //subtract the old value from the current total of all numbers in array
  sumArr = sumArr - arr[ctr];
  //change set value at index equal to the new value
  arr[ctr] = newVal;
  //add the new value to the sum of all values in array
  sumArr = sumArr + arr[ctr];
  //compute the average value of the array
  avgVal = sumArr / 20;
  if (ctr < 19)  //advance index variable by one for next read
  {
    ctr++;
  } else  //reset the index variable to 0 once index 19 is read
  {
    ctr = 0;
  }
  return;
}
```

Appendix B: Temperature Control PI Code

```
/*

   COPYRIGHT (C) 2023 Cameron Macesich (cwm40)

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify,
merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be included in all copies
or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE
OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

   Honors Project: Level and Temperature Controlled Vessel

   Author.  Cameron Macesich

               cwm40@uakron.edu

   Version. 1.01 07APR23

   Purpose: This program is for an Arduino UNO Rev3 for use as a temperature
controller. The temperature controller will hold the temperature to a user defined
setpoint and reads communications from the level control system. Can execute a user
defined function when told by level controller.

   THIS PROGRAM RELIES ON Level_Controller_Final running on an Arduino Mega 2560 to
tell it to run

*/

#include <PID_v2.h>


//For temperature probes

#include <OneWire.h>

#include <DallasTemperature.h>

#define tempPin 8  //temperature probe

OneWire oneWire(tempPin);

DallasTemperature sensors(&oneWire);


//motor

#define motorPinT 11  //Motor control pin, PWM, Temp
```

```
#define motorIN1 12   //Temp Dir N3
#define motorIN2 13   //Temp Dir N4

//bumpSwitch from level controller, yellow button
#define bumpSwitch 10  //input Source: Level controller COM1
//Reads the running state from the level controller
#define start 9  //input Source: Level controller COM2

//Temperatures read by the probe
double 4, 5, 6;

//variables for performing user designated function
bool switchFlag = 0;
long tSwitch = 0;

//Bool variables to hold running state
bool startFlag = 0;

//PID, setpoints for manip var and controlled var
double motorSPT = 90;
double tempSP = 10;
double Kp = 14.41;
double Ki=0.67;

//PID, object calculates new pump motor setpoint
PID tempPID(&4, &motorSPT, &tempSP, Kp, Ki, 0, REVERSE);

void setup() {
  Serial.begin(19200);

  //Temp Sensor
  sensors.begin();
```

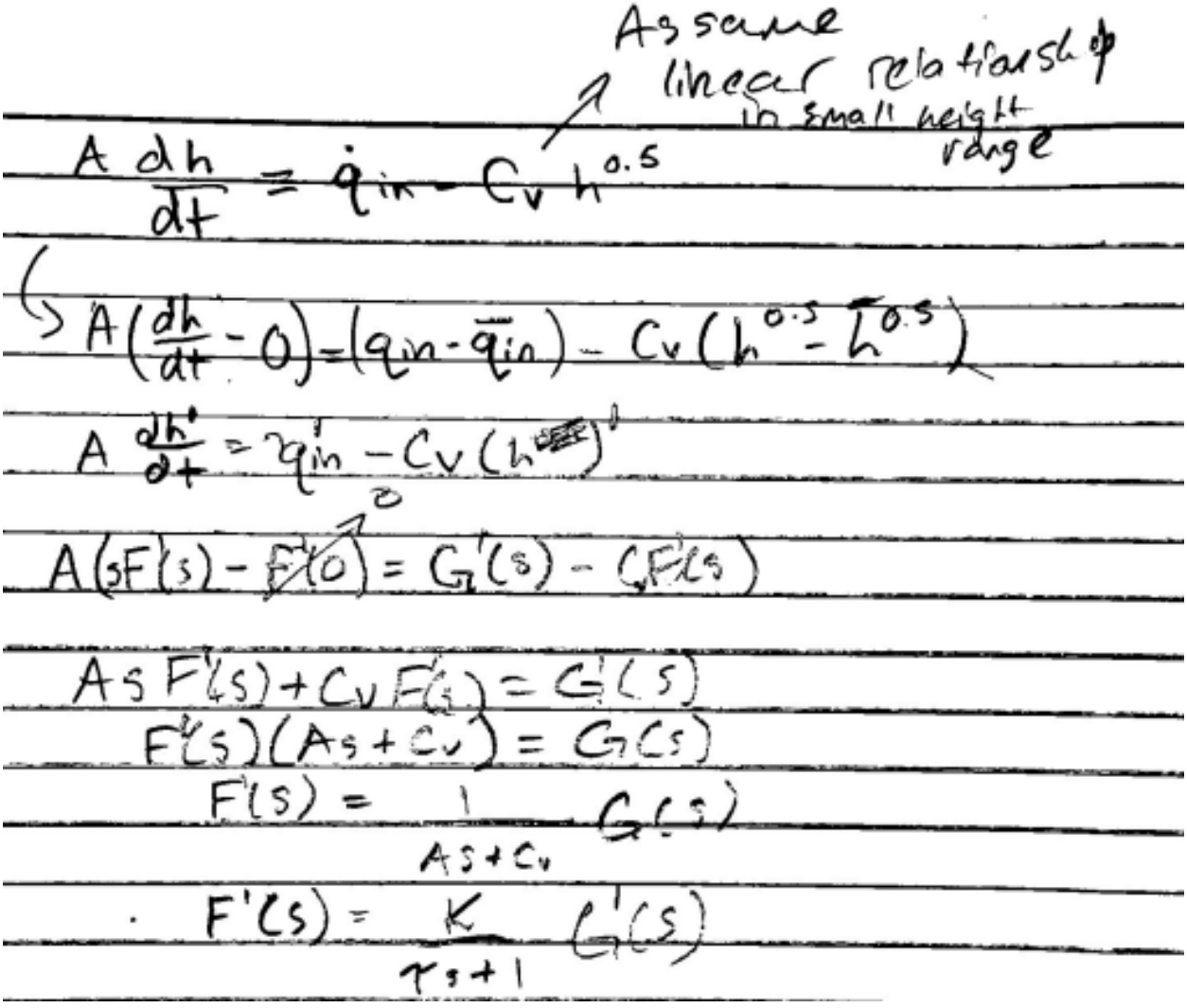
```
  //Temp Control Pump
  pinMode(motorPinT, OUTPUT);
  pinMode(motorIN1, OUTPUT);
  pinMode(motorIN2, OUTPUT);

  //Inputs from level controller
  pinMode(bumpSwitch, INPUT);
  pinMode(start, INPUT);

  /*Note: the emergency stop is read by level controller. Pressing the button still disables the motors by cuttin power to control relay
  regardless of the start state. there will be a delay in reading that the pumps were de-energized, but they will cut out immediately
  */

  //print header for PuTTY
  Serial.println();
  Serial.println("time, motor setpoint, Temp setpoint, Temp out, HEX Out, HEX IN");

  //Set motor State to forward on L298N
  digitalWrite(motorIN1, HIGH);
  digitalWrite(motorIN2, LOW);

  //Set PID limits to working limits of pump
  tempPID.SetOutputLimits(90, 255);  //motor goes beep-beep below 90
}

void loop() {
  //Read input from level controller to determine if the system should be running or stopped
  startFlag = digitalRead(start);

  /*perform user designated function
  */
```

```
  switchFlag = digitalRead(bumpSwitch);
  if (switchFlag == true) {
    //Example: tempSP = 8;
  } else {
    //Example: tempSP = 10;
  }

  //read temperatures of all probes
  sensors.requestTemperatures();
  4 = getTemp(0); //Probe in vessel
  5 = getTemp(1); //Probe at HEX outlet
  6 = getTemp(2); //Probe near HEX inlet

  //Calculate the pump setpoint depending on temperature in tank: CLOSED LOOP MODE
  if (startFlag == HIGH) {
    tempPID.SetMode(AUTOMATIC);
    tempPID.Compute();
    analogWrite(motorPinT, motorSPT);
  } else {
    tempPID.SetMode(MANUAL);
    analogWrite(motorPinT, 0);
  }

  //Print to Serial for PuTTY
  printData();
}

//Cleaner funciton for reading temp sensor. Returns celsius temperature reading from
probe at index probeNum
float getTemp(int probeNum) {
  //returns the temperature detected by probe indexed at probeNum
  return sensors.getTempCByIndex(probeNum);
}
```

```
//Print the time in milliseconds, motor setpoint, temp setpoint, and all 3
temperature probe values to the serial port
void printData() {
  Serial.print(millis());
  Serial.print(", ");
  Serial.print(motorSPT);
  Serial.print(", ");
  Serial.print(tempSP);
  Serial.print(", ");
  Serial.print(4);
  Serial.print(", ");
  Serial.print(5);
  Serial.print(", ");
  Serial.println(6);
}
```

Appendix C: Derivation of Transfer Functions

Gravity Drained Tank Outlet Flow Rate Dependence on Liquid Height



Reactor Temperature Dependence on Cooling Water Flow Rate

Temperature Control — assume steady state level

1) Process Mass balance $\frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} \rightarrow \dot{m}_{in} = \dot{m}_{out} = \dot{m}_p = \rho \dot{q}_p$

2) HEX Mass balance $\frac{dm_{cw}}{dt} = \dot{m}_{cold} - \dot{m}_{hot} \rightarrow \dot{m}_{cold} = \dot{m}_{hot} = \rho \dot{q}_{hex}$

0, no accumulation

3) HEX Energy balance

$$\frac{dH_{hex}}{dt} = H_{in} - H_{out}$$

$$\rho C_p V \cdot \frac{dT_{hot}}{dt} = \dot{m}_{cold} \cdot C_p \cdot T_{cold} - \dot{m}_{hot} C_p T_{hot}(t) + U \cdot A \cdot \Delta T_M$$

$$\Delta T_M = \frac{\Delta T_c - \Delta T_H}{2} = \frac{T_p(t) - T_c - T_p(t) + T_H(t)}{2} = \frac{T_H(t) - T_c}{2}$$

Mean temp difference

$$\rho C_p V \frac{dT_{hot}}{dt} = \rho \dot{q}_{hex}(t) C_p \cdot T_{cold} - \rho \dot{q}_{hex}(t) \cdot C_p \cdot T_H(t) + UA\left(\frac{T_H(t)}{2} - \frac{T_c}{2}\right)$$

$$\frac{dT_{hot}}{dt} = \frac{\dot{q}_{hex}(t) T_{cold}}{V} - \frac{\dot{q}_{hex}(t) \cdot T_H(t)}{V} + \frac{UA}{2C_p \rho V} T_H(t) - \frac{UA\, T_c}{2C_p \rho V}$$

Taylor series to linearize

$$\frac{\partial}{\partial T_H(t)} = \frac{UA}{2C_p \rho V} - \frac{\bar{q}_{hex}(t)}{V}$$

$$\frac{\partial}{\partial \dot{q}_{hex}(t)} = \frac{T_{cold}}{V} - \frac{\bar{T}_H}{V}$$

$-a$

$$s T_H'(s) = \left(\frac{UA}{2C_p \rho V} - \frac{\bar{q}_{hex}}{V}\right) T_H'(s) + \frac{1}{V}(T_{cold} - \bar{T}_H) Q_{hex}'(s)$$

0

$$T_H'(s) = \frac{\frac{1}{V}(T_{cold} - \bar{T}_H)}{s + a} Q_{hex}'(s)$$

$$T_H(s) = \frac{\frac{b}{a}}{\frac{1}{a}s + 1} Q_{HEX}'(s) = \frac{K_a}{\tau_2 s + 1} Q_{hex}'(s)$$

Process EB

$$\frac{dH}{dt} = H_{in} - H_{out}$$

$$\rho C_p V \frac{dT_p(t)}{dt} = \rho C_p T_{in} q_p(t) - \rho C_p T_p(t) \cdot q_p(t) - \frac{UA}{2}\left(T_H(t) - T_c\right)$$

$$\frac{d\,T_p(t)}{dt} = \frac{1}{V} T_{in}\, q_p(t) - q_p(t) T_p(t) \cdot \frac{1}{V} - \frac{UA}{\rho C_p 2V} T_H(t) - \frac{UA}{2\rho C_p V} T_c$$

$$\frac{d}{dT_p(t)} = -\bar{q}_p \cdot \frac{1}{V}$$

$$\frac{d}{dq_p(t)} = \frac{1}{V}\left(T_{in} - \bar{T}_p\right)$$

$$\frac{d}{dT_H(t)} = -\frac{UA}{2\rho C_p V}$$

$$s T_p'(s) = -\frac{1}{V}\bar{q}_p \cdot T_p'(s) + \frac{1}{V}\left(T_{in} - \bar{T}_p\right) q_p'(s) - \frac{UA}{2\rho C_p V} T_H'(s)$$

$$T_p'(s) = \frac{\frac{1}{V}\left(T_{in} - \bar{T}_p\right)}{s + \frac{\bar{q}_p}{V}} Q_p'(s) - \frac{\frac{UA}{2\rho C_p V}}{s + \frac{\bar{q}_p}{V}} T_H'(s)$$

$$T_p'(s) = \frac{\frac{1}{\bar{q}_p}\left(T_{in} - \bar{T}_p\right)}{\frac{V}{\bar{q}_p} s + 1} Q_p'(s) - \frac{\frac{UA}{2\rho C_p \bar{q}_p}}{\frac{V}{\bar{q}_p} s + 1} T_H'(s)$$

$$T_p'(s) = \frac{K_1}{\tau_1 s + 1} Q_p'(s) - \frac{K_2}{\tau_1 s + 1} T_H'(s)$$

$$T_p'(s) = \frac{K_1}{\tau_1 s + 1} Q_p'(s) - \frac{K_2}{\tau_1 s + 1}\left(\frac{K_3}{\tau_2 s + 1} Q_{Hex}'(s)\right)$$

$$T_p'(s) = \frac{K_1}{\tau_1 s + 1} Q_p'(s) - \frac{K_2 \cdot K_3}{\tau_1 \tau_2 s^2 + (\tau_1 + \tau_2) s + 1} Q_{Hex}'(s)$$

Transfer fn. for Temperature relative to flow rates

Appendix D: Wiring Diagrams