Independent Group Lab: Water Heater with PID Control

Michael Ancel, Tabitha D'Amato, Grant Ewing
MinGyu Kim, Leah Manuel, Noah Schwab

In this experiment, a system of water was heated using an immersion heater subject to control by a PID controller. System identification was completed to gather physical constants pertaining to the system. Controller design was then completed to gather the gains for the PID controller relative to the expected transfer function and the desired success criteria. The controller was tested using a simulated plant before being connected to hardware for experimentation. The hardware was then calibrated and corrected to fix the multitude of issues pertaining to the use of water with the hardware. Finally, the experiment was run. The experiment showed many differences from the simulated results; however, it successfully achieved the success criteria, and worked correctly under multiple disturbances.

I. Nomenclature

E	=	water energy
Q	=	total heat into water
W	=	total work
m	=	water mass
c	=	water specific heat capacity
T	=	current water temperature
ΔT	=	total temperature change
Q_{in}	=	heat input
Q_{out}	=	heat output
Q_{dis}	=	total heat dissipation
Q_{peb}	=	heat change from pebbles
h	=	heat transfer coefficient for energy dissipation
T_{amb}	=	ambient air temperature
T_0	=	initial water temperature
K_{ss}	=	steady state gain
τ	=	time constant

II. Introduction

Temperature control in water is an underrecognized, yet integral, piece of daily life. When a sink or a shower is turned on, temperature control is used to ensure the system outputs water that is a comfortable temperature for humans. Every heated pool follows some sort of temperature control to ensure that the water remains the same comfortable temperature regardless of the surrounding climate. Fish tanks and growth ponds are temperature controlled to ensure the survival of the fish within. The ability to control and modify the temperature of water is useful for the daily life of many living creatures.

This report details the design of a control system for a dynamic heating system for a bucket of water. The objective of the system was to design a controller which heats a volume of water to a desired steady state temperature following some temperature disturbance to the system. To achieve this, an immersion heater controlled by an Arduino Uno was placed in a bucket of water. The disturbance was created by placing frozen pebbles and ice cubes in the water to lower

the temperature. Extensive testing and research were completed to create a model of the system which was used to gather the PID controller gains. Using a thermistor to measure the temperature of the water, the immersion heater was given on/off commands by the PID controller derived in Simulink. All calculations were completed in MATLAB.

Very minimal background knowledge of this system was known prior to the development of this system. Each group member understood PID controllers because of the control lab course, and a few of the members of the group remembered some knowledge of heating from an aerothermodynamics course. From this prior knowledge, one initial idea was using the lumped capacitance method to model the pebble disturbance, where the geometry, average convection coefficient, and thermal conductivity were factors. Previous course material also included computing change of temperature through a wall, so accounting for the temperature gradient through the bucket was also considered. Ultimately, however, a simpler dynamics model was made relating the heat from the immersion heater, heat dissipating into the air, and a constant heat absorbed by the pebbles and ice cubes.

III. System Dynamics Modeling

System Modeling

The system being controlled involves the heating of a volume of water; hence, the thermal dynamics of the system were analyzed with the heat equation using first order terms for modeling this system. The complete derivation is shown in the Appendix. The model to be used comes from the change in energy of the system at steady state and includes contribution from Q_{in} (the heat added from the immersion heater), Q_{dis}(heat lost to dissipate), and Q_{peb} (the heat taken by the disturbance, which would be frozen pebbles and ice cubes).

The Q_{dis} term is the product of the average convection coefficient, h, and the difference in temperature between the ambient temperature (T_{amb}) and the measured temperature $T(t)$. The resulting Laplace transform equation of the measured temperature can be regarded as one transfer function or plant multiplied by various inputs and constants, as simplified in Equation 1.

$$T(s) = \frac{\frac{1}{h}}{\frac{mc}{h}s + 1}\left(Q_{in}(s) - Q_{peb}(s) + hT_{amb}(s) + T_0(s)\right) = \\ = \frac{K_{ss}}{\tau s + 1}(G_1 Q_{in}(s) + G_2 Q_{peb}(s) + G_3 T_{amb}(s) + G_4 T_0(s) \quad (1)$$

where $K_{ss} = \frac{1}{h}$, $\tau = \frac{mc}{h}$, $Q_{in}(s)$ is the control voltage input and $Q_{peb}(s)$, $T_{amb}(s)$, and $T_0(s)$ are constant blocks.

System Identification

Although the dynamics equation is found at the equilibrium state, some variables are still unknown. It might be possible to use equations in a theoretical way, but imprecise information, such as the unknown material of the bucket, does not help to find the exact characteristic values. Therefore, the variables are estimated with some assumptions through the system modeling experiment. When the heater is off and pebble is absent, $Q_{in}(s) = 0$ and $Q_{peb} = 0$, thus creating Equation 2.

$$\left(s + \frac{h}{mc}\right)T(s) = \frac{1}{mc}\left(h\frac{T_{amb}}{s} + T_0\right) \quad (2)$$

Taking the inverse Laplace transform of equation 2, the temperature response T(t) is shown in equation 3.

$$T(t) = T_{amb} + \left(\frac{T_0}{mc} - T_{amb}\right)e^{-\left(\frac{h}{mc}\right)t}. \quad (3)$$

When the heater is on and pebble is absent, $Q_{peb} = 0$ but it is assumed that the heater generates a constant heat energy, giving Equation 4.

$$\left(s + \frac{h}{mc}\right)T(s) = \frac{1}{mc}\left(\frac{Q_{ss}}{s} + h\frac{T_{amb}}{s} + T_0\right) \quad (4)$$

The temperature response T(t) is now given in equation 5.

$$T(t) = T_{amb} + \frac{Q_{ss}}{h} + \left(\frac{T_0}{mc} - T_{amb} - \frac{Q_{ss}}{h}\right) e^{-\left(\frac{h}{mc}\right)t} \quad (5)$$

$$\lim_{t \to \infty} T(t) = T_{amb} + \frac{Q_{ss}}{h} \quad (6)$$

There are two ways to find the heat transfer coefficient h value. One way to solve for h is by using Equation 3; without the heater and iced pebbles, the temperature change can be observed as it converges from the initial temperature to the ambient temperature. An experiment was done where the heater was turned on for 120 seconds and then was turned off to let the temperature drop to a lower temperature. Since the heat of the water requires a long time to dissipate, a temperature of 26.4826 °C was chosen to stop at so that a clear linear slope could be seen from the initial temperature (where the temperature starts to decline after the heater is turned off). The initial temperature, current temperature, and time duration can be calculated from Figure 1 and Table I, and mass of water and specific heat for water are already known. By plugging those values into the equation, the coefficient h value can be found.

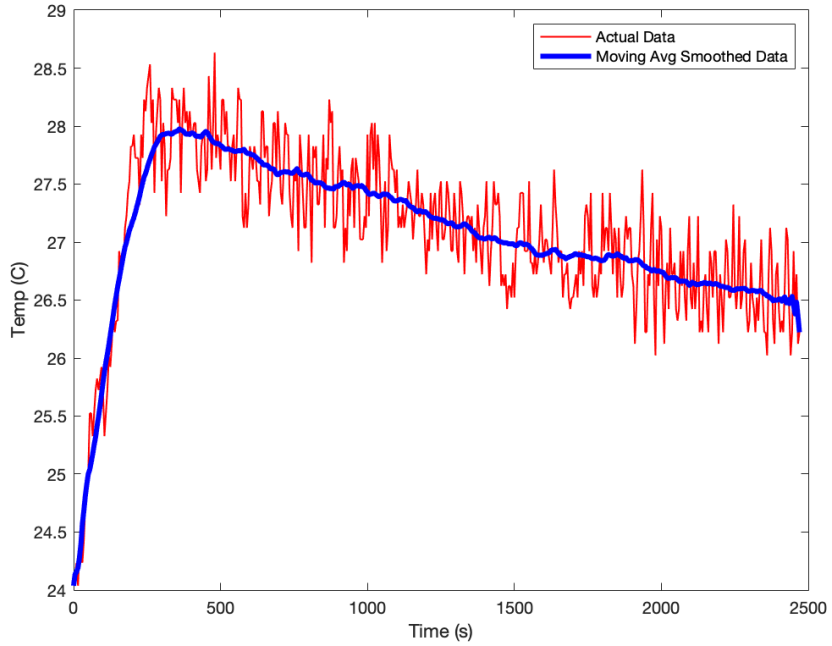

Figure 1. Temperature over time plot after the heater is turned off at 120 seconds

Table I. Measurements from the thermistor and Arduino Uno for the heat transfer coefficient calculation (Derived from Figure 1)

T_{amb} (°C)	T_0 (°C)	T_{target} (°C)	Time duration (s)
28.8333	27.9436	26.4826	2120

Table II. Heat energy values of the heater and disturbance.

Q_{ss} (max heat) (J)	Q_{peb} (J)
1100	5000

The other way is to find the h value using Equation 5. As time increases to infinity with a constant heat source provided, the temperature should converge to a value, and the entire third term in Eq. 5 will become more negligible (decreasing to zero) and be reduced to Equation 6. The maximum constant heat produced by the heater is 1100 J according to the device specification, as shown in Table II, so the maximum heat value was tested to find the exact heat transfer coefficient h value using Equation 6. The heater only had two settings of being off and on when needed. The estimate of the coefficient h is listed in Table III. The data used is recorded in Table III. A comparison of the computed h values is shown in Table III; since the h value calculated using Figure 1 was larger than the value described in the device specification document, the h value calculated using Equation 6 was used for the control design simulation.

The method using equation 5 used the maximum heat produced from the immersion heater, which is equal to the product of the heater efficiency and the power provided by the heat. While initially, a value of 40% was chosen for the efficiency, further comparison with other heaters revealed the efficiency would be close to 100% [1]. In the method using equation 3 and the response plot, error may have arisen from the voltage to temperature conversion in the Arduino, which may have not been very accurate for smaller temperature changes. Also, the residual heat from the heater's metal plate may have prevented accurate temperature readings.

Table III. Estimate of coefficient h, heat transfer coefficient.

	Derived using Figure 1	Derived using Equation 5
h (W/K/m^2)	102.9290	18.1295
Q_{ss} (J)	5060.7	1100

With the heat transfer coefficient value derived by Equation 5, the steady state gain and time constant values were calculated and recorded in Table IV.

Table IV. Steady state gain and time constant for the transfer function.

K_{ss} (J/K/m^2)	τ (sec)
0.0612	5330

Regarding Q_{peb}, pebbles placed in a freezer were used to create the disturbance that the immersion heater would compensate for. The heat flux from the pebble disturbance is not modeled; assumptions made for the pebbles are that they extract a constant amount of heat, Q, and that the heat gradient within the pebbles is negligible. Because actual dynamics for the cold disturbance was unknown, the value Q_{peb}, or the heat taken from the disturbance, was assumed to be a constant 5000 J, as shown in Table II. The constant value allowed for simplification and was calculated by first assuming that 15 grams of ice per second would be used as a constant disturbance (the disturbance of ice was changed to pebbles to maintain the same volume of water in the experiment). It was assumed that the ice would be at the freezing temperature of 0 °C and no colder, and that the specific latent heat of ice is 336 J/g. The quantity Q_{peb} is the product of the specific latent heat of ice and the mass of ice per second.

IV. Controller Design

The criteria for success were selected and modified based on the system parameters. A percent overshoot of less than 10% was selected, and a time to reach steady state of less than 10 minutes was selected. The system must also remain within 5˚C once it reaches steady state unless some outside disturbance affected the temperature of the system. No limits were placed on rise time and control effort to allow for a controller that focused on minimal error, rather than cost.

To best meet these requirements and to play to the group's prior knowledge, a PID controller was selected. In order to decide the PID gains which met these conditions best, the Matlab Control System Designer was used in conjunction with the transfer function for a PID controller, shown below in equation 7 and the transfer function for the plant, stated above in equation 1. These gains are listed below in Table V.

$$G(s) = \frac{K_i + K_p s + K_d s^2}{s} \tag{7}$$

Table V. Theoretical PID gains.

Gain	Value
K_i	0.01
K_p	50
K_d	6250

A Simulink diagram of the PID controller used can also be seen below in Figure 2.

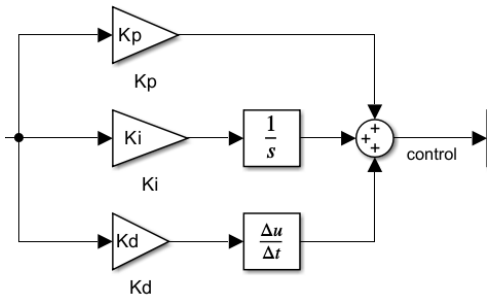

Figure 2. PID Controller

Derivation of these values required a lot of tuning. It was decided that the best solution to meet the steady state requirement was to create an overdamped system. As such, the K_d gain was very high and K_p and K_i gains very small to reduce overshoot. Additionally, the settling time was required to be very quick in proportion to the rise time of the system, and as such, K_i was very small.

V. Simulation

The system was then simulated using Simulink. The previously derived values of PID gains were added to the PID controller, as well as the plant constants derived during system identification. An additional relay block was added to ensure that the simulated control effort does not exceed the maximum heat energy allowed by the immersion heater, or at 1100 J. A step input command was used to communicate the difference from the desired steady state value to the system; the value of this command was a 25˚C difference from the initial temperature. The Simulink block diagram used for simulation is shown in Figure 3 below.

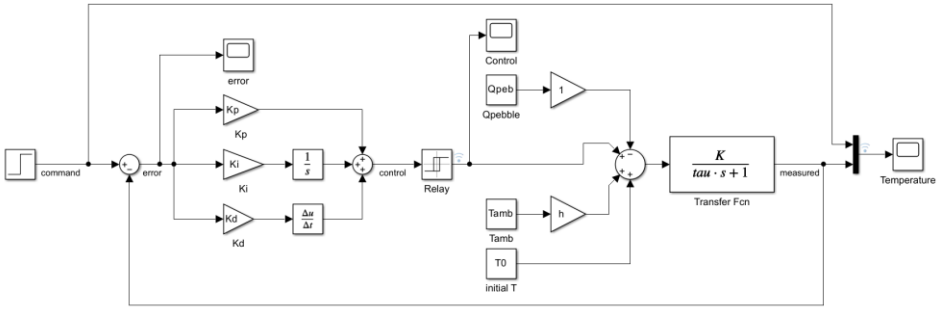

Figure 3. Control Modeling and Design Simulink Model

The controller required no tuning, as it met the success criteria with the simulated gains; however, any tuning to the system would follow the same pattern as expressed in the controller design section. The response of the simulated system to the controller is shown below in Figure 4.

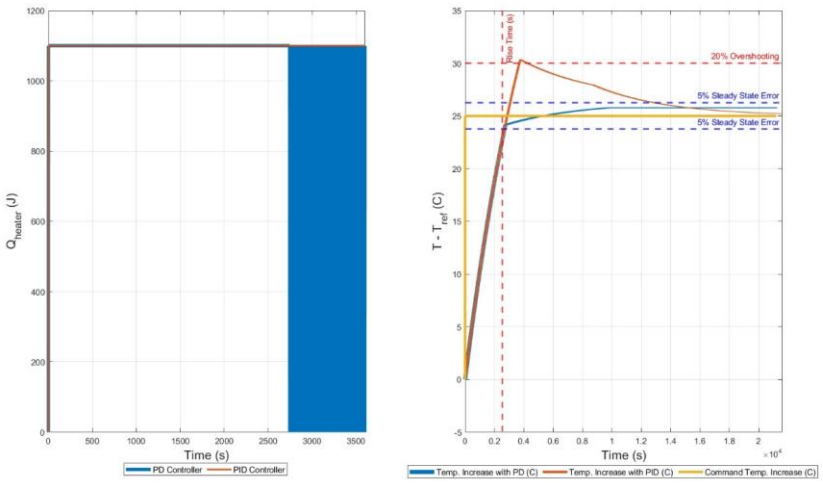

Figure 4. Temperature Response of the System with PD and PID controllers.

VI. Hardware Implementation

System Overview

A photo taken during the experiment can be seen below in Figure 6. The metal bucket with an immersion heater inside and thermistors taped to the sides can be seen near the bottom of the photo. Resting on the chair is the Arduino and circuit connecting the thermistors, allowing measurement of temperature described in detail in future sections. Finally, the black outlet box is the relay that connects the immersion heater and Arduino, which allows controlling of the immersion heater through the Simulink system.

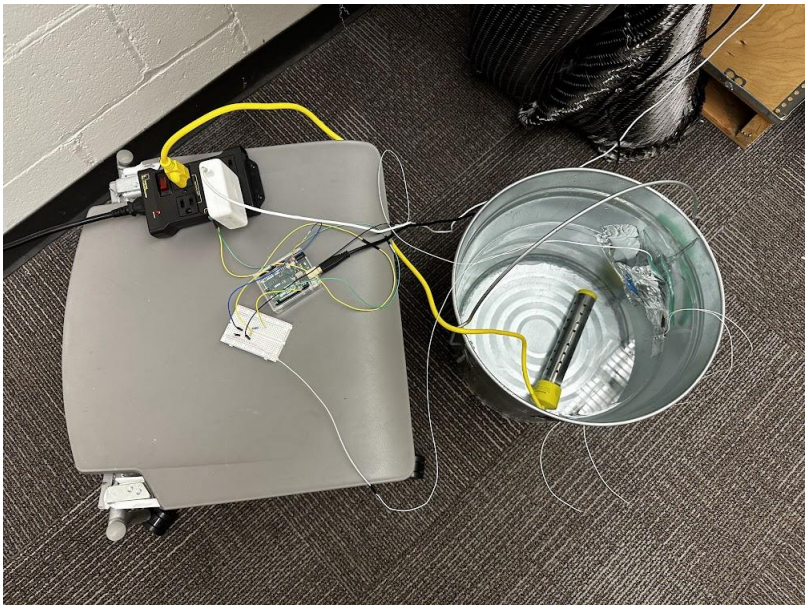

Figure 6. Experimental Set-up

A table of the required components and their specifications are listed below.

Table VI. System components and specifications.

Component Name	Specifications
Metal Bucket	10 gallon (5.5 gallons of water used)
Thermistor	NTC 100K Ohm
Arduino Uno	N/A
Immersion Heater	1500 W
Wooden Spoon	N/A
Frozen Pebbles/Ice	N/A
DC Relay	3-60 V DC

The purpose of the system is to control the temperature of a bucket of water using an immersion heater. The temperature would be lowered by submerging several cold rocks or ice cubes, and the immersion heater would then attempt to bring the bucket temperature up to a set value. A 10-gallon metal bucket was filled with a prescribed volume of water and the immersion heater was placed at the bottom of the bucket. The immersion heater was powered by a DC relay. To measure the temperature, thermistors were taped to the exterior of the bucket. Temperature readings were recorded by an Arduino Uno microcontroller and control design and analysis was conducted in MATLAB and Simulink. The Arduino was then connected to the relay to control the immersion heater.

Because the thermistors were located in the top water surface interior of the bucket and the immersion heater was in the center, large temperature gradients in the water would result in inaccurate temperature readings by the thermistor. The heater impact was very localized. To remedy this issue, the water was continually stirred manually to thoroughly mix it and minimize potential temperature gradients.

For the controller, a standard PID control architecture was developed. However, the immersion heater can only turn on and off and does not have a tunable heat output. Because of this, the PID control signal was modified to produce a binary output of 1 or 0 based on a set threshold to be utilized by the immersion heater.

Thermistor Calibration

To integrate temperature readings of the system with the controller, an NTC 3950 thermistor was used in conjunction with an Arduino Uno. Thermistors are elements that vary resistance with changes in temperature, which can be measured and converted into temperature using the process described below. A diagram of our circuit as modeled with an online software is shown in Figure 7 and the process is to convert the Arduino signal to a temperature value is described below.

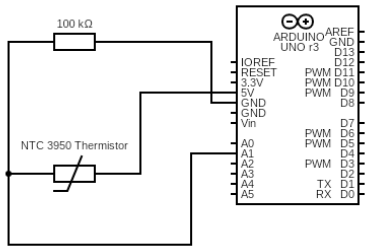

Figure 7. Thermistor and Arduino Circuit Diagram

The thermistor used is a 100k ohm thermistor, meaning the resistance of the thermistor is 100k ohms at 25°C [2]. The variable resistance as a function of output voltage can be found using a voltage divider law based on the circuit built in Figure 6 and is represented in Equation 8.

$$V_{out} = 5 * \frac{100{,}000}{100{,}000 + R_{th}} \tag{8}$$

Solving for the resistance of the thermistor yields the following.

$$R_{th} = \frac{5 * 100{,}000}{V_{out}} - 100{,}000 \tag{9}$$

The Arduino Uno has a 10-bit analog to digital converter (ADC), meaning the range of digital values is between 0 and 1023. The conversion described in Equation 10 is necessary to solve for the output voltage used to calculate the resistance of the thermistor.

$$V_{out} = \frac{5 * ADC_{val}}{1023} \tag{10}$$

Finally, the variable resistance value can be used to find the temperature of the system. Manufacturer data for an NTC 3950 is available in the form of a table relating resistance to temperature, which was initially integrated with Simulink and used in a table lookup. However, upon implementation many issues were found with the table lookup due to instantaneous errors. It was decided to use a curve fit to the manufacturer table data, which was done numerically with MATLAB. This curve fit is represented in equation for in Equation 11 and compared to the manufacturer data graphically in Figure 8. It is important to note that within the 290-330K range of temperatures that the water and system would fall within, the curve fit is quite a good match.

$$T = 592.6433 - 25.1409 * \ln(R) \tag{11}$$

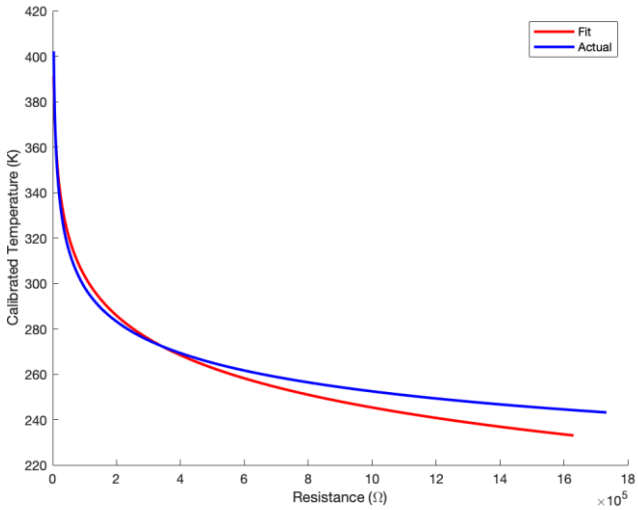

Figure 8. Thermistor Temperature Calibration

The above equations were integrated into a Simulink model, shown in Figure 9.

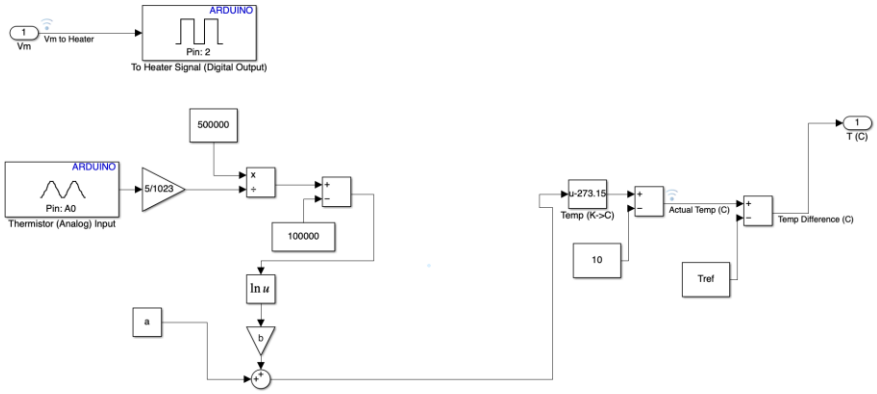

Figure 9. Thermistor Temperature Reading in Simulink

The input to the system is the Arduino Uno ADC value, which is converted to a voltage using the gain described in Equation 10. Next, the voltage is converted to a resistance value through mathematical blocks based on the voltage divider formula given in Equation 9. After this, the resistance is converted to Kelvin using the approximation function described above, where a is 592.6433 and b is – 25.1409. Finally, this Kelvin temperature is converted to Celsius,

where it is ready to be used by the control system. The constant block 10 and the T_{ref} blocks were solutions to problems encountered during experimental testing and are described in detail in the following section.

VII. Experimental Implementation and Evaluation

Implementing the actual experimental system in the real world proved to be difficult. Firstly, there were continual systemic problems with the hardware selection. Upon receiving the requested water holding metal bucket, the label indicated that it should be used for dry storage only, not filling with liquids. This became obvious when the bucket leaked under the pressure and weight of 8 gallons of water for the experiment. The stresses of the water made gaps in the metal of the bucket open and clearly demonstrated that the seams were not watertight. Since there was no time to request another bucket, the team moved ahead with the system. Untreated, the bucket leaked significant amounts of water, and this loss was amounting to an unacceptable level to justify our assumption that water mass was constant.

An initial treatment plan was devised to use metal HVAC tape and cover any exterior gaps that the water was shooting out of. This treatment plan had unfortunately suboptimal results. With a short amount of time, water won out and was able to still leak through the sealing tape at a significant rate. Even with the next addition of gorilla tape, this exterior tape sealing method proved unsuccessful. A final sealing method was pioneered and involved lining the interior of the bucket gaps with thick bands of hot glue. This sealing method proved more successful throughout the multi-week experiment duration, and it limited the water leakage amount to an acceptable, yet still noticeable amount. This sealing method only needed to do minor touchups after multiple bucket cycles of being filled and unfilled with water applied some deformation to the bucket structure and therefore loosened the glue adhesive. A noticeable qualitative behavior of the system appeared during system identification and found that at high water temperatures above 40 C, the rate of water leak decreased. This can likely be attributed to the increase in glue malleability at higher temperatures, thus making it a more optimal material for gap filling.

A second issue presented itself in the thermistors used for temperature data acquisition. The thermistors were originally intended to be placed at various depths in the bucket interior sides with direct exposure to the water. This was planned to keep the thermistors far enough away from the heater element to avoid some localized heating effects. After establishing and calibrating a link between the Arduino, thermistors, and Simulink data acquisition system, the water temperature readings were found to be entirely erroneous with a +/- 30˚C oscillating noise continuously. This made collecting temperatures impossible. Upon doing diagnostics, it was found that the thermistors did not possess the ability to read temperature when submerged in water and that it impacted their resistance value, and subsequent voltage reading into the data acquisition system.

A solution was found to place the thermistor device in a Ziploc plastic bag and then partially submerge the bag along a bucket side so that the thermistor would not have direct contact with the water yet still be fully surrounded by the water pressing against the bag to closely read the temperature. This proved useful in eliminating the oscillation and achieving readable temperature data. After the thermistor calibration process described earlier was implemented, the measured temperature data still had about a positive 10˚C absolute error, found upon comparison with an independent digital thermometer temperature of the water. After brief investigation, it was found that this error stayed constant in magnitude with variation in temperature, so a block was added to the Simulink hardware interface to subtract 10˚C from all readings to correct for this inaccuracy. The final Simulink model is shown in Figure 10.

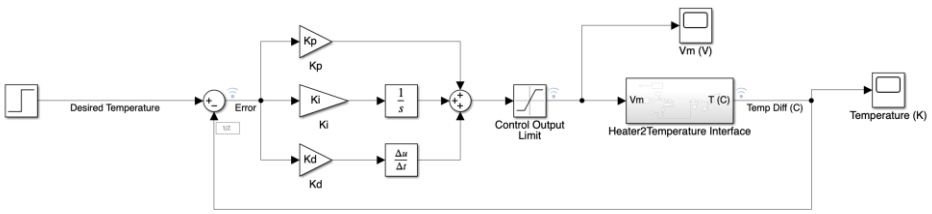

Figure 10. Controller Implementation with Hardware Simulink Block Diagram.

It is important to note that the Heater2Temperature Interface subsystem is exactly made up of the Figure 9 block diagram. This Simulink block diagram should look slimmer than the Figure 3 block diagram. Additionally, a saturation block was added to ensure that the simulated control effort did not exceed the maximum voltage allowed by the relay, or 60 V. The disturbance model and the plant transfer function were quite buggy in the actual implementation. The

controller was not behaving optimally with those components present. For example, the model predicted the heater would take longer than expected to heat up and the computed h value grossly overestimated the heating time. Many of these errors could have propagated from the initial system identification since many of the measuring methods were imprecise due to non-scientific hardware and long measurement time horizons.

The team made the informed decision to simplify the closed loop controller around only the measured margin of error and the correction response since the complicated physics relations were not reflecting the actual real-life behavior of the system. There are many variables of the actual system like convective heat transfer (water and air), water leakage, uneven heating, and non-uniform thermal disturbances that were outside of our control and would have required more sophisticated materials and hardware to quantify and account for. The simplified block diagram result did a better job of attaining a desirable system control which was the end goal.

Following the solution, the experiment was run. After placing frozen pebbles into the water, the controller was turned on, and the water was stirred continuously by hand. The controller was allowed time to bring the system to the desired steady state temperature before adding ice to the water to generate a temperature disturbance once again for the controller to fix. It is important to note that as the pebbles and ice cubes were added and the water was stirred, it took around 100 seconds for the cold disturbance to absorb the heat from the water, which differed from the simulation, causing some error from the simulated results to the experimental results. The experimental parameters are given in Table VII, and the graphical results of the experiment are given in Figures 11 and 12, where Figure 11 shows the control effort and response of the system and Figure 12 shows the design parameters.

Table VII. Experimental Parameters

Maximum Deliverable Heater Voltage (V)	T_{ref}	$\Delta T_{commanded}$	$T_{final_{desired}}$	$T_{final_{actual}}$
$60\ V$	27.5 ℃	Δ25 ℃	52.5 ℃	52.2 ℃

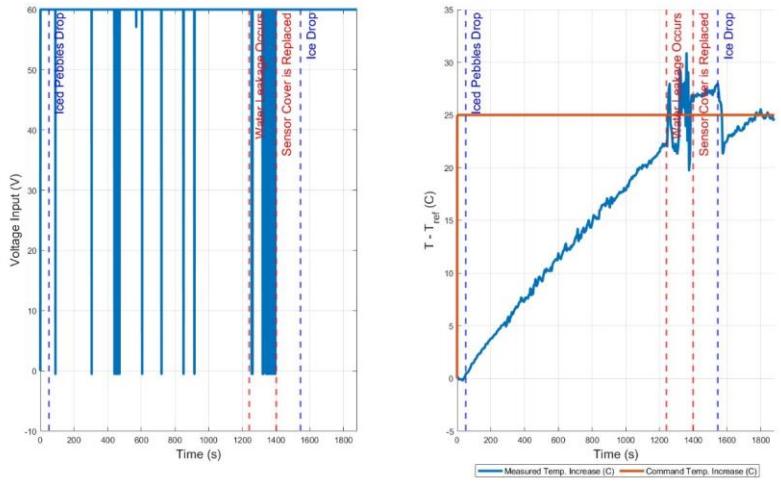

Figure 11. Response data from the actual physical experiment with the PID controller.

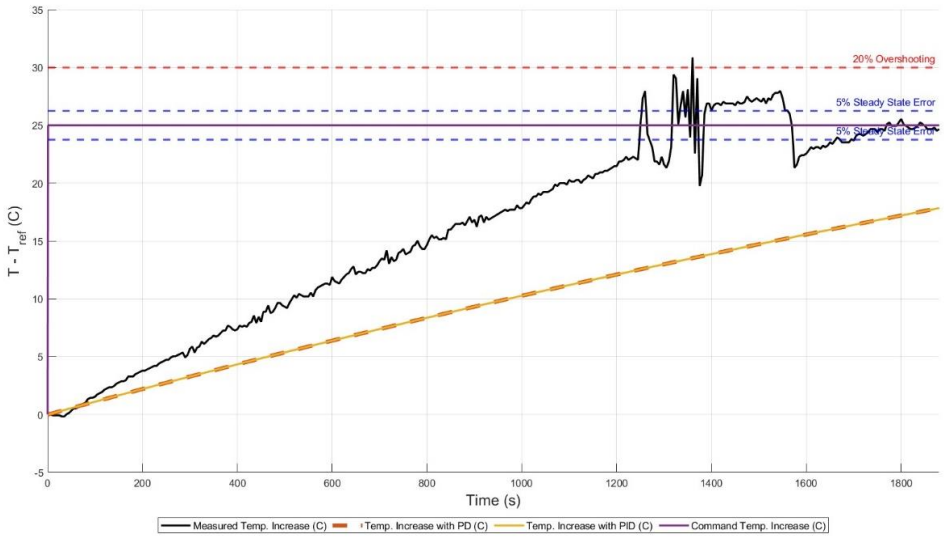

Figure 12. System temperature response as shown with the design parameters.

Another real system problem was discovered with the Simulink controller. The controller did not behave as desired when inputting a commanded temperature into the Simulink controller interface. The Simulink controller acted erroneously when trying to store the initial water temperature to use as a reference and then used that reference to find the magnitude to raise the temperature of the system. This was very problematic, and many steps were taken to troubleshoot. These might have been compounded by the previously discussed issues with the physics model. In the end, the only viable solution found was to change the type of command given to the controller and instead input a desired rise in temperature $\Delta T_{commanded}$ that the system should achieve. This involved using an independent digital thermometer to compute a T_{ref} temperature, which is the initial temperature of the water added to the bucket before heating is applied. This created a parameter, $T - T_{ref}$, which is a measure of the actual temperature of the water over time, T, in terms of the initial water temperature. This then led to the computation of error for the closed loop controller which is shown in Equation 12.

$$Error = \Delta T_{commanded} - \left(T - T_{ref}\right) \tag{12}$$

The result of plotting these metrics when conducting the experiment is an easy-to-read graphic that progressively shows that as error converges to 0, the $T\text{-}T_{ref}$ parameter increases to the value of $\Delta T_{commanded}$. The actual temperature was also plotted as a fail-safe and to help assess experiment safety as the water temperature rises.

It should be noted that in the range 1200 to 1400 seconds in Figure 12, an unplanned temperature oscillation occurred. Quickly upon investigation, this was found to be caused by water entering the thermistor Ziploc plastic bag and disrupting the reading. The bag was quickly changed, and the experiment was continued without lasting disruption. This behavior matched the earlier finding that water touching the thermistor results in erroneous readings. Upon correction, the temperature accuracy was restored to within 1 degree Celsius. At just before 1600 seconds, the temperature change reached its commanded value and the error was reduced to 0. This is when the cooling disturbance of ice cubes were added. This disturbance is meant to physically implement the from the model and test the controller response. For both the initial pebble case and latter ice cube disturbance cases, added it took slightly under 100 seconds for the full impact of the cold disturbance to become measured while stirring the water, as shown graphically most evident for the ice cube case in Figure 12. This time also includes the thermistor water temperature reading time. The ice cubes melted almost upon impact with the system when added. The speed at which these thermal disturbances propagated through the system were much faster than expected from the simulation.

Since the mass of these were much less than the water, the volume change impacts are negligible and were neglected. Clearly the measured temperature drops significantly at this disturbance based on the graph, but the system

corrects and restores the water to the commanded temperature increase in 2000 seconds, which is an order of magnitude faster than the model expected. This is an incredible improvement over predicted performance and most likely points to problems in the underlying model.

After the conclusion of the experiment, multiple physical sources of error were pointed out. Firstly, the simulated plant was likely not a perfect representation of the physical model. As stated earlier, finding the constants h and Q_{max} proved difficult, and there was likely some error in gathering these constants for the simulated system. As such, Another likely source within the vane of the multitude of hardware mishaps is that we didn't have a method of measuring the exact voltage or power delivered to the relay and heater in real time to see if this was consistent with the model. It seems rational to question how much power was delivered and how much energy the heater dissipated into the system when the heating result was so far above the margin expected. Creating a way to measure these actual electrical parameters and the subsequent efficiency would be a next step for future experiments. [SNT8] The thermistor disturbance induced by the water leak actually provided some useful feedback on the controller. During this time period of 1200-1400 seconds, the heater turned off and on abruptly several times. This provides evidence that upon large overshoots, it corrects and turns off. However, if you closely analyze the plot, it doesn't turn off this voltage input until after the overshoot is over a 27 degrees Celsius change in temperature, not the 25 degrees Celsius change as commanded. This implies that there would have been some overshoot at the final steady state since the heater slowly approaching the goal state of a 25-degree Celsius increase would most likely not turn off until after hitting a 27-degree Celsius increase. While this has problematic implications, the physical performance is still remarkable in terms of only overshooting by 2 degrees Celsius when considering that the system performed an order of magnitude faster than the model expected. More error would have definitely been possible.

VIII. Conclusion

The conducted experiment involved the modeling, control design, and control implementation of a water-based temperature control system. The modeling of the system used a thermodynamics approach which proved accurate simulation in Simulink but failed to adequately describe the real-world system during implementation. Many possible causes for this have been discussed, including a multitude of hardware difficulties, convective heat transfer, or unaccounted for heat losses in the system. With the adjusted and simpler control system, the temperature was able to be controlled within specifications of 20% maximum overshoot, a 30-minute settling time, and a steady state tolerance of +/- 5˚C. The proportional, derivative, and integral gains found were the following: 50, 6250, 0.01. With this PID controller, there was only a 0.3 deg difference between the commanded final temperature and the actual final temperature after ice was added and the immersion heater reacted.

The experiment was successful in that temperature measurement from thermistors was viable and this feedback was implemented into the closed loop controller. Heater voltage input control was done nominal through the controller, albeit having a slightly higher margin for turning off than expected. Heating the water to reach the commanded input occurred at an order of magnitude faster than expected and the disturbance model acted more responsive than the model indicated yet system performance didn't have adverse transient behavior. Steady state error had mixed results as there is evidence of overshoot but not outside of the tolerance. Future repetition of this experiment with less error prone hardware is recommended. Higher quality scientific hardware also would likely have improved the data acquisition, especially during the system identification process which would have led to a more accurate thermodynamic physics model of the system that could have been utilized for the control implementation.

References

[1] 'Flanged Heaters,' WATTCO, Available: https://www.wattco.com/product_category/flanged-heaters.

[2] “NTC thermistor R-T Table Sheet”, KEENOVO, Available: https://forum.duet3d.com/assets/uploads/files/1541668164665-ntc-thermistor-r-t-table.pdf.

[3] “Thermistor interfacing with Arduino Uno: Arduino,” ElectronicWings, Available: https://www.electronicwings.com/arduino/thermistor-interfacing-with-arduino-uno.

[4] Kane, P., “Thermistors/Temperature measurement with NTC Thermistors,” Jameco Electronics, Available: https://www.jameco.com/Jameco/workshop/TechTip/temperature-measurement-ntc-thermistors.html.

Appendix

At equilibrium, the change in energy of the system is equal to 0, as shown in Equation 1. Since no work is applied to the system $\sum W = 0$, the equation can be simplified to Equation 2.

$$\frac{dE}{dt} = \sum Q + \sum W = 0 \tag{1}$$

$$\frac{dE}{dt} = \sum Q \tag{2}$$

The energy change in the heat transfer system is proportional to the temperature change; at steady state, the equation can be written as Equation 3. In this equation, m is the mass of water and c is the specific heat of water. Additionally, the energy change is equal to the net heat change, or the difference between the amount of heat going in and out through water in the bucket, shown in the following equation.

$$\frac{dE}{dt} = m * c * \Delta T = m * c * \frac{dT}{dt} \tag{3}$$

$$\frac{dE}{dt} = Q_{in} - Q_{out} \tag{4}$$

Substituting the derivative of energy with Equation 3, the system dynamics can be modeled with Equation 5, where Q_{in} is the heat produced by the immersion heater, Q_{dis} is the heat lost to dissipation, and Q_{peb} is heat lost to a disturbance from frozen pebbles.

$$\dot{T}(t) = \frac{dT(t)}{dt} = \frac{1}{mc}\big(Q_{in}(t) - Q_{out}(t)\big) = \frac{1}{mc}\Big(Q_{in}(t) - \big(Q_{dis}(t) + Q_{peb}(t)\big)\Big) \tag{5}$$

The dissipation from the water to the air can be modeled as the average convection coefficient, h, from water to air, multiplied by the difference between the measured temperature and the ambiance temperature. This formulation assumes that the heat lost through conduction from the water through the walls of the bucket is much less than the heat lost to convection between the water and air.

$$\frac{dT(t)}{dt} = \frac{1}{mc}\big(Q_{in}(t) - Q_{peb}(t) - h\Delta T\big) = \frac{1}{mc}\Big(Q_{in}(t) - Q_{peb}(t) - h\big(T(t) - T_{amb}(t)\big)\Big) \tag{6}$$

Further characterization of the system can be more easily performed in the s-domain after taking the Laplace transform. The Laplace transform equation of Equation 6 can be derived as shown in the equations below. The notation in Equation 8 implies that $Q_{peb}(s) = \frac{Q_{peb}}{s}$ and $T_{amb}(s) = \frac{T_{amb}}{s}$. Furthermore, T(t=0) is the same term as $T_0(s)$, which is a constant. It is the resulting initial condition of the Laplace transform of the derivative of T(t), so it does not follow the same form as the other two constants.

$$sT(s) - T(t=0) = \frac{1}{mc}\left(Q_{in}(s) - \frac{Q_{peb}}{s} - h\left(T(s) - \frac{T_{amb}}{s}\right)\right) \tag{7}$$

$$sT(s) + \frac{h}{mc}T(s) = \frac{1}{mc}Q_{in}(s) - \frac{1}{mc}Q_{peb}(s) + \frac{h}{mc}T_{amb}(s) + T(t=0) \tag{8}$$

$$\begin{aligned} T(s) &= \frac{1}{mcs + h}Q_{in}(s) - \frac{1}{mcs + h}Q_{peb}(s) + \frac{h}{mcs + h}T_{amb}(s) + \frac{1}{mcs + h}T_0(s) \\ &= G_1Q_{in}(s) + G_2Q_{peb}(s) + G_3T_{amb}(s) + G_4T_0(s) \end{aligned} \tag{9}$$

The Laplace transform equation of the measured temperature can be regarded as one transfer function or plant multiplied by various inputs and constants, as simplified in Equation 10.

$$\begin{aligned} T(s) &= \frac{\frac{1}{h}}{\frac{mc}{h}s + 1}\big(Q_{in}(s) - Q_{peb}(s) + hT_{amb}(s) + mcT_0(s)\big) = \\ &= \frac{K_{ss}}{\tau s + 1}(G_1Q_{in}(s) + G_2Q_{peb}(s) + G_3T_{amb}(s) + G_4T_0(s) \end{aligned} \tag{10}$$

Commented [MLJ1]: Even though mc doesn't work in the simulink diagram, should we still include it in the equation since it's write? "mc" would also be added to T0 in eq. 10 if so.
Also wanting to write T_amb/s and Q_peb/s in the equation.

Commented [MK2R1]: Umm I remember I changed it to 1.. I just modified it