Final Design Report

Digital Control of an Environmental Testing Chamber

Group 03

Reilly Pickard, Silas Needler, Nizar Zahran Salim Al Aufi

Dalhousie University Faculty of Mechanical Engineering

MECH 4015/4025: Senior Design Project

April 11th, 2023

Table of Contents

List of Figures

List of Tables

1. Introduction and Background

1.1 Project Description

Dr. Farid Taheri is a professor in the department of mechanical engineering at Dalhousie University. His research involves the characterization and development of advanced industrial materials, and many of his experimental studies involve the use of his environmental testing chamber. When this project originated in the summer of 2022, the controller for his environmental chamber was mechanically based and the desired temperature and humidity were set by the turning of a dial. To add a time scale to the inputs, a cam and trail arm system was employed. Therefore, anytime a different temperature cycle was required, the user had to cut a custom cam. Moreover, there was no temperature feedback to the user, and this required the use of separate thermocouples to ensure the desired temperatures were met. Additionally, there was no digital alert system in place for when the chamber exceeded safe temperature ranges. As a result, Dr. Taheri requested that a digital controller be implemented; that is, a full system upgrade to enable the user to modify and track the temperature and humidity of the chamber by digitally inputting setpoint values of temperature, humidity, and their respective time scales. Dr. Taheri also expressed the desire for the ability to download a file containing the temperature and humidity data. He also requested an alert to be sent if the system failed or exceeded the safe range. The group worked closely with the Dr. Taheri and his graduate student to integrate the digital controller hardware with a graphical user interface that provides a user-friendly experience with desired levels of control customization. The scope of the project saw the machine's current heating and cooling systems were to be maintained and controlled. That is, a heating stage is employed for desired temperatures above the ambient and up to 350 ºF, mechanical refrigeration for the first cooling stage, and cryogenic cooling using liquid nitrogen to achieve temperatures between -40 ºF and -100 ºF.

This report will detail the entire design, installation, testing, and analysis process for this project. Once the key project stakeholders are mapped and the requirements are clearly defined, a functional decomposition of the problem that breaks down the relevant subsystems is described. With these subsystems identified and explained, multiple design concepts for each subsystem are presented. Three potential final designs that met the project requirements were then developed by combining subsystem concepts. Building off the ideas laid out in these designs, a prototyping process was conducted and will be outlined. Based on what was learned in this initial prototyping in combination with prior research, engineering justification was used to rate each subsystem on their usability, functionality, and feasibility. The alternative final designs were then ranked by summing the scores of their respective subsystem concepts. Once an initial final design is detailed, the report will then describe the key iterations made in the Winter of 2023 and provide justification for each. From these iterations, a final product will be presented and the thorough testing process that ensured it met each requirement is detailed. Finally, an engineering economic analysis is presented, and several recommendations are listed for the future directions of this project.

1.2 Project Stakeholders

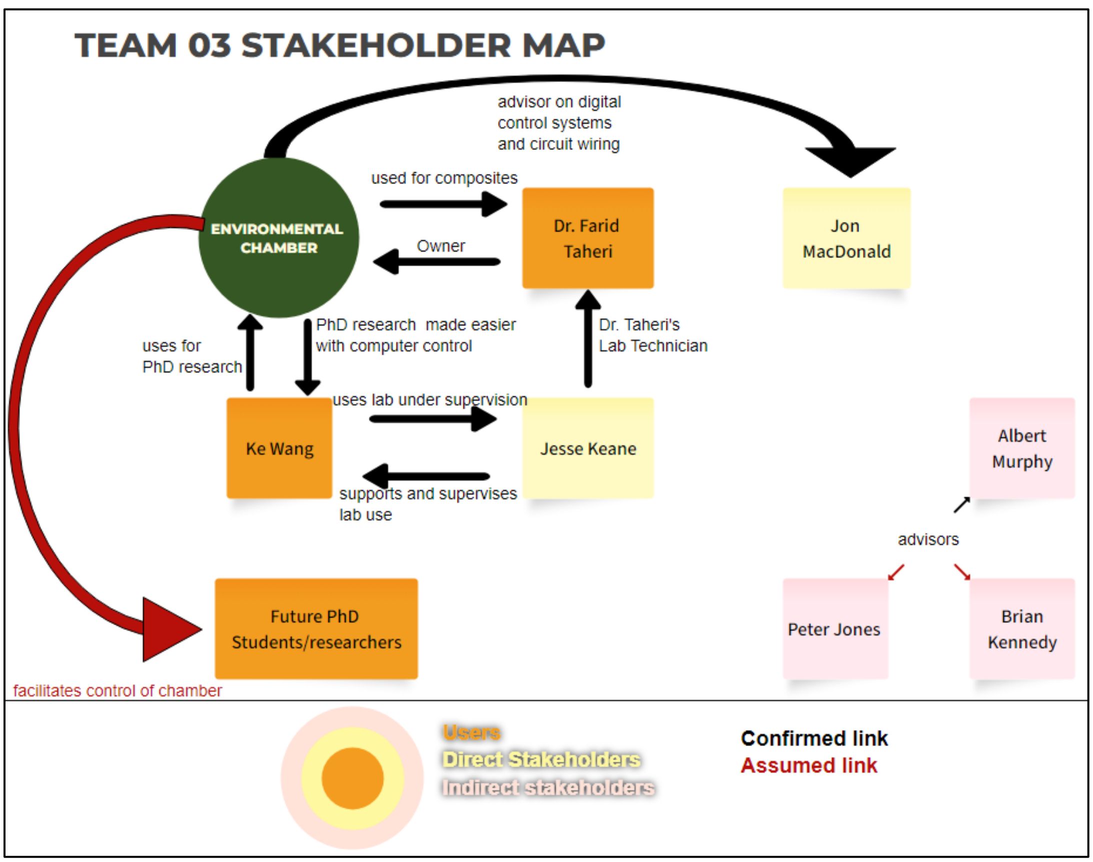


Figure 1: Stakeholder Map

Figure 1 shows the project's stakeholders and their relationships to the machine and to each other. A full list of stakeholders can be found in Appendix A Stakeholder List (clickable).

The group has identified the only current user as Ke Wang, Dr. Taheri's PhD student. This does not however limit him as the only stakeholder. Dr. Taheri is the chamber owner and he determined that upgrading to digital control is more effective than acquiring a new chamber with integrated digital control. He also has a desire to eventually use the chamber for composites curing, a process that he conducts in a different lab. Therefore, it is assumed that future graduate students of Dr. Taheri are also stakeholders. Further, Jesse Keane is the lab technician that oversees activities within the lab and is the contact in charge of safety. Outside of the lab, Jon MacDonald advised the group on the safety and performance of all electrical components in this project.

1.3 Project Requirements

Table 1 details the complete project requirements, originally devised in September 2022. The present status of these requirements is also listed along with any modifications that were made throughout the process. These requirements were verified with third party instruments by means of a thorough testing procedure. The completed testing process is detailed in Section 6.

Table 1: Project requirement breakdown

Category	No.	Requirement	Origin	Owner	Rationale	Priority	Status	Modification	Verification
1. Functionality	**1.1**	The digital controller shall be able to keep the temperature of the environmental chamber between a minimum temperature of -100 ºF and a maximum temperature of 350 ºF.	Client	RP	This represents the temperature range that the client wishes to treat the material within.	High	100%	Updated verification method.	Third party oven thermometer. Third party RTD for -100
	1.2	The controller shall be able to keep the environmental chamber humidity between 20 and 80 %.	Client	RP/NZA	This represents the humidity range that the client wishes to treat the material within.	High	100%	Modified from 10-98% to 20-80% due to equipment limitations. Humidity was deemed not as crucial as temperature due to prior chamber usage.	Third party humidity sensor
	1.3	The temperature of the environmental chamber should increase or decrease to the required setpoint at a rate of 2 ºF/min.	Team	RP	Implemented by team for design of controller algorithm.	Low	100%	None at present.	Third party oven thermometer and timer.

Category	No.	Requirement	Origin	Owner	Rationale	Priority	Status	Modification	Verification
	1.4	The temperature of the environmental chamber shall stay within ± 4 ºF of the setpoint.	Team	RP	Implemented by team for design of controller algorithm.	High	100%	Adjusted wording.	Third party oven thermometer.
	1.5	The temperature and humidity sensors must have 6 ft of lead length within the chamber for repositioning.	Team	RP	This gives the user the ability to monitor temperature from various points on the material and/or the environment.	Low	100%	Reduced to 6ft after measurement of chamber.	Tape Measure
2. Usability	**2.1**	The user shall have the ability to set the temperature between -100 ºF to 350 ºF.	Client	RP	Gives the user the basic ability to control environment temperature.	High	100%	Reworded and added verification method.	Third party oven thermometer.
	2.2	The user shall have the ability to manually set the relative humidity between 20 to 80%.	Client	RP/NZA	Gives the user the basic ability to control environment humidity.	High	100%	Reworded and added verification method. Modified from 10-98 to 20-80 due to equipment limitations. Humidity was deemed not as crucial as temperature.	Third party humidity sensor.

Category	No.	Requirement	Origin	Owner	Rationale	Priority	Status	Modification	Verification
	2.3	The user shall have the ability to select the time where a step change in properties occurs, within the above ranges.	Client	RP	Gives the user the ability to program custom temperature cycles.	High	100%	Reworded and added verification method.	Timer and Thermometer
3. Data Feedback	**3.1**	The system should show a real time graph of current material temperature versus time.	Team	RP	Real time feedback to show chamber performance.	Low	100%	Reworded and added verification method.	Timer and Thermometer
	3.2	The system should show a real time graph of current chamber air temperature versus time.	Team	RP	Real time feedback to show chamber performance.	Low	100%	Reworded and added verification method.	Timer and Thermometer
	3.3	The system should show a real time graph of chamber air humidity versus time.	Team	RP	Real time feedback to show chamber performance.	Low	100%	Reworded and added verification method.	Third party humidity sensor.
	3.4	The user shall be able to view a file containing material temperature versus time data.	Client	RP	Allows user to review the exact environmental conditions in which the treated material was exposed to.	High	100%	Reworded and added verification method.	Timer and third-party thermometer.
	3.5	The user shall be able to view a file containing chamber air humidity versus time data.	Client	RP	Allows user to review the exact environmental conditions in which the treated material was exposed to.	High	100%	Reworded and added verification method.	Timer and third-party humidity sensor.

Category	No.	Requirement	Origin	Owner	Rationale	Priority	Status	Modification	Verification
	3.6	The user shall be able to view a file containing chamber air temperature versus time data.	Client	RP	Allows user to review the exact environmental conditions in which the treated material was exposed to.	High	100%	Reworded and added verification method.	Timer and third-party thermometer.
4. Safety	**4.1**	The power supply to the system shall power off within 10 s if the temperature of the chamber exceeds 350 ºF.	Team	RP	A safety feature that prevents overheating of the system.	High	100%	Reworded and added verification method.	Timer and third-party thermometer. View LED turns off
	4.2	The power supply to the refrigeration system shall power off within 10 if the temperature of the chamber goes below -100 ºF.	Team	RP	A safety feature that prevents overcooling of the system.	High	100%	Reworded and added verification method.	Timer and third-party thermometer. View LED turns off
	4.3	Should the system exceed the operating range of -100 ºF to 350ºF, the controller shall send an alert to the user within 10s.	Team	RP	Alert is sent the client so that they can investigate potential issue and if necessary, manually power off the system.	Low	100%	Reworded and added verification method..	Timer and third-party thermometer. View Message Received
	4.4	The user shall have an option on the controller that powers off the system within 10s.	Team	RP	Allows user to end cycle should there be a safety issue.	High	100%	Reworded and added verification method.	Timer and third-party thermometer. View LED turns off

1.4 Cost Outlook

The client set a budget of $1,000.00 for the entire project, $100.00 of this being allocated to the prototyping process. The client specified that this $100.00 be spent to make the prototype functional rather than conceptual so that he can interact with and inspect it. The client expected that exceeding this 10% allocation was not necessary to design a functional prototype that enabled the team to learn all the specified content. The prototyping process will be described in detail in Section 5.1. An economic analysis is presented in Section 7 to discuss the cost savings and the return on investment.

2. Functional Decomposition

The functional decomposition of a digitally controlled environmental testing chamber can be seen in Figure 2.

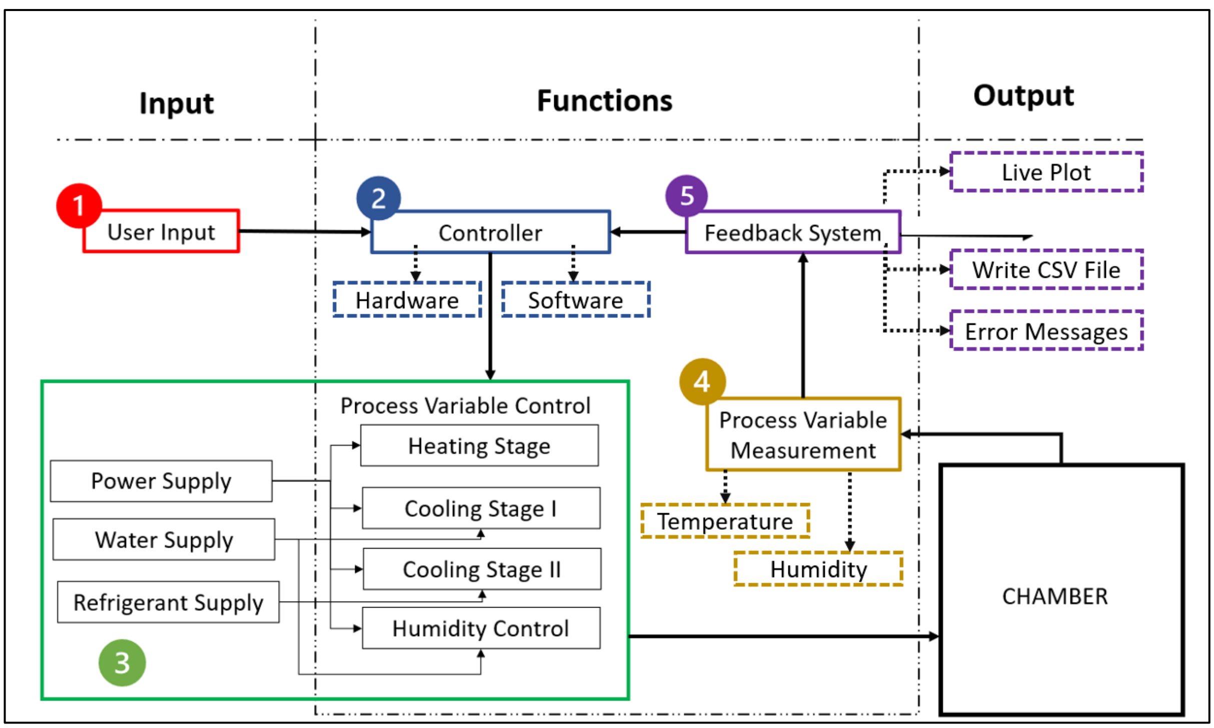


Figure 2: Functional decomposition

Five subfunctions were identified. They are as follows:

1. **User Input:** While this occurs on the input side of the functional decomposition, there was still a significant amount of design decisions to be made to give the operator the

ability to enter information to the system. This subsystem was required to possess the ability to accept both numeric and string inputs from the user, store them as data, and send the information to the controller.

2. **Controller:** A controller is a combination of compatible hardware and software that receives information from both the user and the feedback system. The software was to be programmed in a manner that compared the desired process variable value with the measured output. Depending on the result of this comparison, the controller was to possess the ability to supply power to the physical devices that input or remove energy from the chamber.

3. **Process Variable Control:** This subfunction had to receive information from the controller and control the temperature and humidity of the chamber via the entry or removal of energy. Since the function is different for heating, cooling, and humidity control, this subsystem was broken up further. The subsystems within the process variable control umbrella are as follows:

    3.1 Heating Stage: This function was to be designed to raise the temperature of the chamber should the desired setpoint be above the measured temperature and below 350 ºF.

    3.2 Cooling Stage 1: This function was to be designed to lower the temperature of the chamber should the desired setpoint be below the measured temperature and above -40 ºF.

    3.3 Cooling Stage 2: This function was to be designed to lower the temperature of the chamber should the desired temperature be between -40ºF and -100 ºF.

    3.4 Humidity Control: This function was to be designed to possess the ability to modify the humidity of the chamber based on the desired humidity setpoint.

4. **Process Variable Measurement:** To modify the controller voltage output, process variables needed to be measured and compared to the desired setpoint. The design of this subfunction would therefore need to implement measurement devices that could

accurately measure the temperature and humidity inside the chamber and send this information back to the controller.

5. **Feedback System:** The feedback system can be broken down into two main components.

    5.1 Live Results: A stated desire of the client was the ability to monitor the process variables in real time while also being able to export the temperature vs time data after a cycle is completed. This would indicate the accuracy of the control system as well as expose any errors in the process.

    5.2 CSV Export: It was expressed that the user shall be given the ability to retroactively view the recorded temperature and humidity data to give insights into the conditions under which the material was tested. This can be done by creating an exportable CSV file.

    5.3 Error Messages: A functional requirement of the design was that it should alert the client if the temperature fell below –100 °F or exceeded 350 °F. Therefore, an alert system was to be programmed into the controller.

3. Subsystem Concept Generation

3.1 User Input

One of the main benefits of implementing a digital control system is the ability for the user to easily control the machine output with their custom input. The main function of the user input subsystem is to accept and store process variables from the user and send them to the controller software subsystem. Two main components of the user input are the level of input customization and the design of the interface. Ideally, the two were to be combined in a manner that the chamber operator could easily input desired setpoints as a function of time. Below are three design alternatives that achieve this functionality.

i) **Python TKinter**.

This python add-on package allows for the creation of a graphical user interface (GUI) that can be programmed in a manner that allows a user to input multiple values [1]. With this design, a user can also choose between input types. The first of the input types was a wave cycle in which the user could enter the minimum and maximum values of the process variables, the time between peaks, and the stop time. The second input type allowed the user to enter a process variable and the time at which they would like that variable to remain constant. They could then choose to add another value-time couple by selecting the “add another” check box [1]. The user could repeat for as many values they wish. These variable values and time profiles can be stored, and serial written to the microcontroller wherein the decision algorithm can be conducted. Figure 3 shows a proposed interface built with Python TKinter.

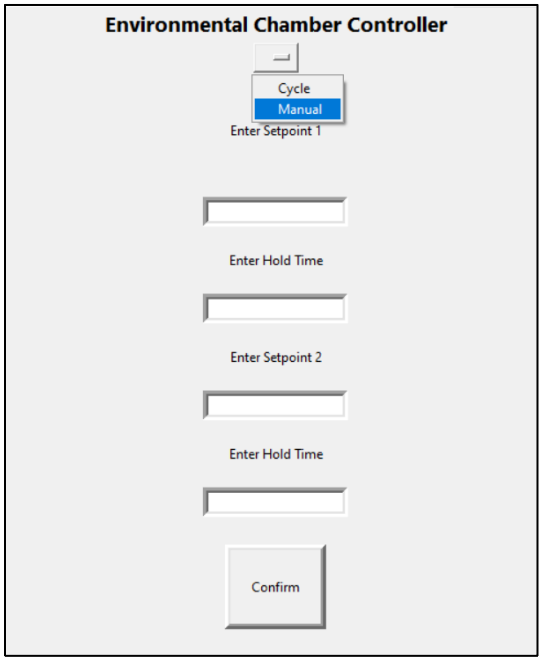


Figure 3: Example Python TKinter GUI

ii) Raspberry Pi (RPI) Touchscreen

This design alternative is akin to the Python TKinter interface module but a bit more complex. A Raspberry Pi GUI only works with the Raspbian operating system and requires the programming of a Raspberry Pi microprocessor [2]. However, this design pays off in user experience. A Raspberry Pi can be controlled from a touchscreen or phone app, which would eliminate the need for a computer connected to the controller [2]. While this required a sophisticated level of programming ability, this design provided a high level of custom control in a user-friendly fashion.

iii) SD Card Input

By installing an SD card entry slot to the microcontroller, the user would have the ability to manually write a setpoint vs time profile in a spreadsheet. This data file could then be saved into an SD card inserted to the user's personal computer, before being inserted to the slot on the microcontroller. The microcontroller would then receive and unpack the information from the storage device and continue with its decision algorithm. SD cards are commonly used for 3D printer controllers, wherein the analog instructions to the controller can come in the form of complex geometries [3].

3.2 Controller Design

3.2.1 Hardware

The central operating function of this project was the controller itself. An effective controller shall have the ability to receive multiple input signals from the user and then compare them with measured values. The microcontroller hardware should consist of enough input and output ports to supply signals in the form of electrical power to the multiple machines being used to control the temperature and humidity. The following microcontrollers accomplish these tasks:

i. **Arduino Nano:** An Arduino Nano is a microcontroller with a variety of serviceable features. It contains several power supply terminals, a time base, and can process both digital and analog inputs and outputs [4]. Moreover, an Arduino Nano contains a USB connection port that allows the programmer to connect the controller to their PC and program control algorithms within the Arduino IDE software that can switch the power supply terminals on and off based on the received inputs. The main benefit of the Arduino in this scope was that it could be easily attached to a breadboard, allowing for easy prototyping and design configuration. The Arduino IDE operates using the C language [4] which was known to the design team based on prior coursework. Therefore, using an Arduino for this design would allow the team to use the pre-existing expertise, and helps future students who aim to improve the design.

ii. **Raspberry Pi:** A Raspberry Pi controller is a microprocessor based miniaturized computer that requires its own operating system [5]. It has all the same functions as the Arduino in terms of power supply outputs and reception of analog inputs, but its multifunctional nature makes it more suitable for more detailed control algorithms [5]. The Raspberry Pi is also open-source and allows for additional operating programs to be written. Moreover, the Raspberry Pi is much more suitable in situations where Bluetooth or Internet of Things (IOT) connection is required. The Raspberry Pi always allows for the implementation of a physical controller based off a touchscreen. However, the learning curve to program the control algorithm to a Raspberry Pi is much steeper, meaning the design implementation would be less feasible and future students and users may run into issues should they not be familiar with the system.

3.2.2 Control Algorithm

As far as programming the controller itself, two iterative systems were originally considered, the first being ON/OFF control and the second being proportional-integral-derivative (PID) control. ON/OFF is a simple algorithm that switches machine components on and off in a binary fashion based on the setpoint and measurement. Figure 4 shows an example graph of how ON/OFF control works. This is trivial to program, but this type of control leads to decrease in performance caused by excessive oscillation near the setpoint, and additionally may cause hysteresis, inaccuracy, and a limited control range [6]. Therefore, the only control technique considered for this project was a PID.

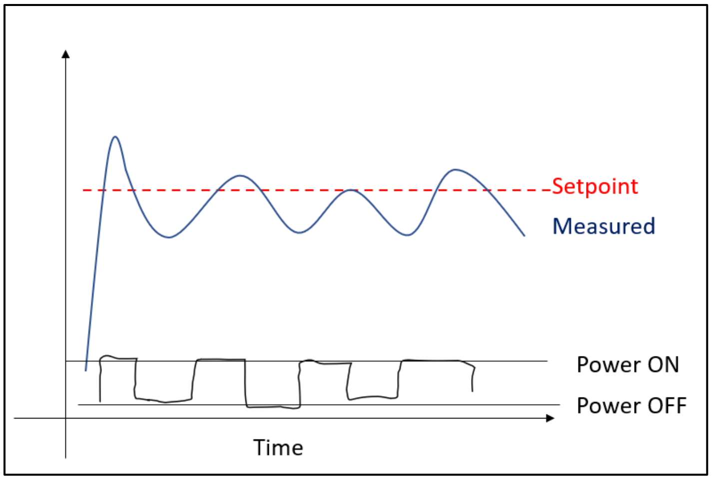


Figure 4: On/Off control w.r.t. Setpoint

A PID control algorithm is a more complex version of ON/OFF control. Rather than simply outputting ON and OFF, a PID controller performs mathematical operations on the deviation error between the setpoint and measured value [7]. There are three main parts. The first is the proportional computation, which multiplies the error by a predefined constant, K_P [7]The second is an integrating function, which integrates the error with respect to time and multiplies it by the integrating constant, K_I [7].The third is a derivative operation which differentiates the error with respect to time and multiplies it by the derivative constant, K_D [7]. The results of these three computations are then summed and a voltage output is sent to the temperature controlling device. This allows for less oscillation and overshoot of the setpoint as when the measured value is close to the desired value, the voltage output is decreased. Therefore, tuning the controller constants would allow for any required rise time and fall times to setpoint values to be met. Thus, even though for the controller algorithm there was only one design was considered, there was still a substantial level of customization in the design as different tuning will lead to different system response. Figure 5 shows a block diagram created in Simulink illustrating the functionality of a closed loop PID controller.

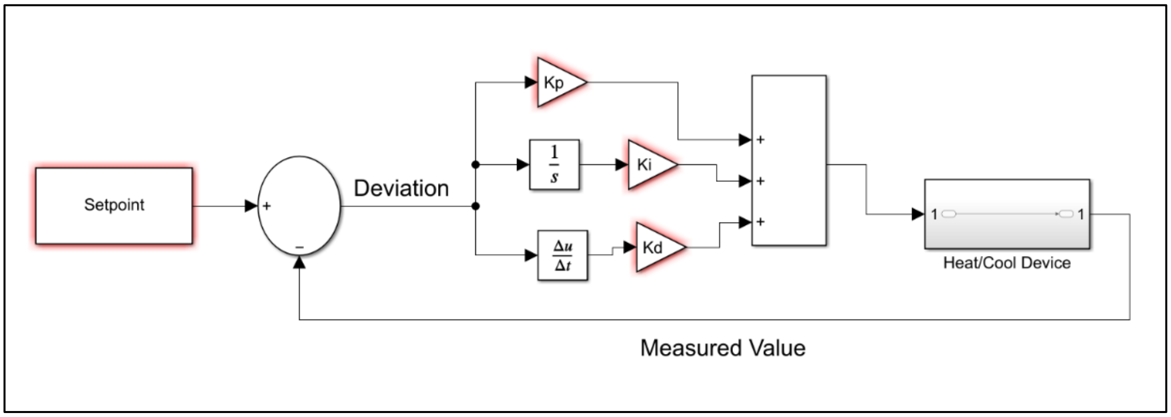


Figure 5: Closed loop PID controller schematic

3.3 Process Variable Control

3.3.1 Heating Stage

The function of the heating stage is to receive a signal from the controller decision algorithm and control the temperature of the chamber should the set point be above the ambient temperature. Below are two designs that achieve this function.

i. **Relay Controlled Electric Heater:** This design used a solid-state relay (SSR) to control the power supply to a heating element. The solid-state relay responds to the signal supplied by the controller and either turns the power to element on if the measured temperature is below the desired, or off if the desired temperature is below the measured value [7]. This provided a low noise solution that is 100 % efficient in terms of energy conversion, as the power supplied is equal to the voltage-current product and is emitted from the element as a heat rate. It would also allow for easy connection to the digital controller. However, electric heating requires high electricity consumption and requires the relay interface to be in between the controller and the element. Figure 6 shows the working principle of the proposed design [7].

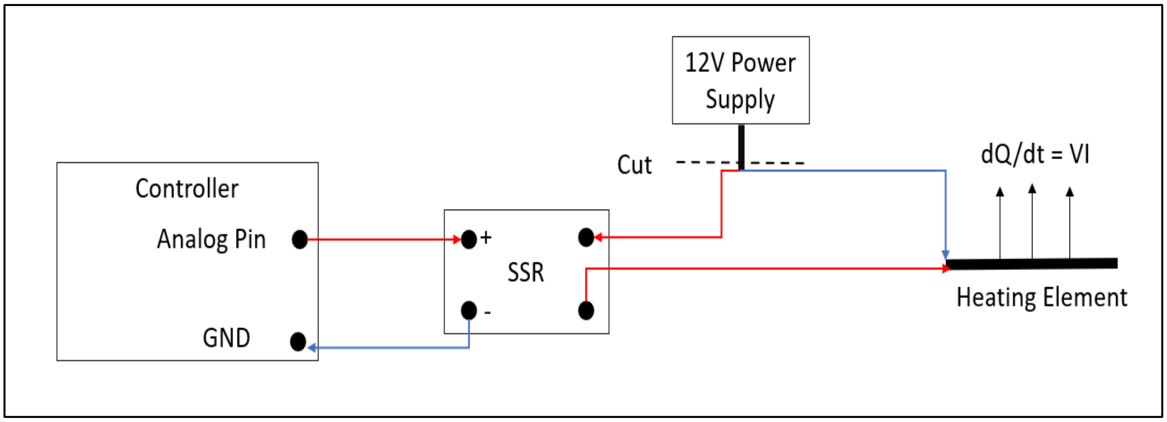


Figure 6: Relay controlled electric haeting schematic

ii. **Heat Pump:** While the system already employed a heating element solution, the team sought to investigate a secondary design. The second alternative to increase the temperature of the chamber was to use a heat pump. The working principle of a heat pump is that it absorbs air from a source, in this case the chamber exterior and runs it through a mechanical cycle that expels excess heat into a sink, in this case the chamber [11]. The heat pump cycle is not 100 % efficient in terms of converting supplied energy into heat and requires an external refrigerant supply [11]. Heat pumps can also be loud and require maintenance. However, heat pumps require less energy input and are often cheaper than electric heating [11]. The working principle of a heat pump is shown in Figure 7.

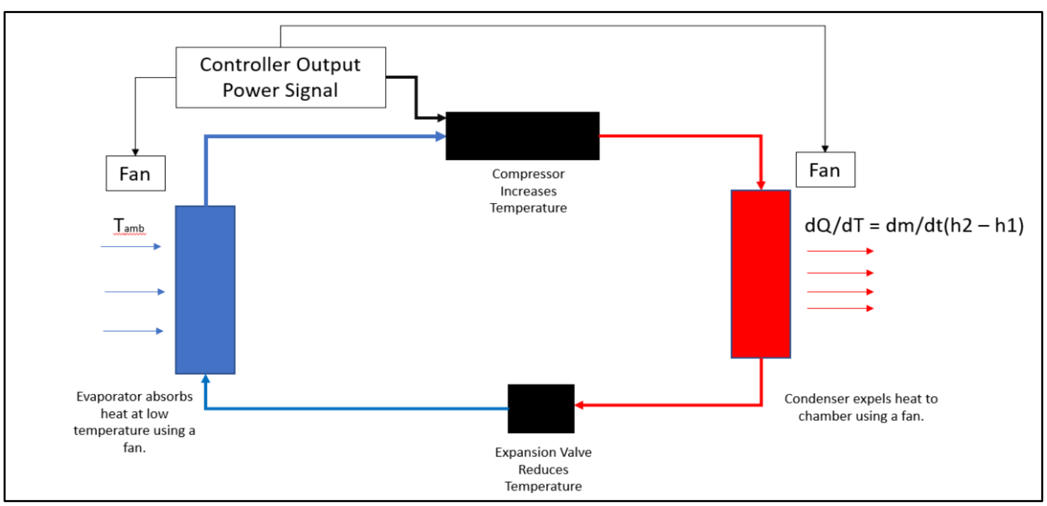


Figure 7: Heat pump heating schematic

3.3.2 Primary Cooling Stage

The mechanical control system for the environmental chamber used a mechanical refrigeration cycle for the primary cooling stage. This system is the reverse as the heat pump cycle. Now the source is the chamber, and heat is pulled from the chamber into the evaporator which is in this case located inside the chamber as opposed to on the exterior. The vapor is then compressed to a high temperature and heat is expelled through a condenser to the ambient air. The vapor is then expanded through a valve back to low temperature, and the process repeats [11].

Therefore, like the heat pump, this function must receive a signal from the controller and modify the compressor and fan power based on the information supplied by the controller. Figure 8 shows the process [11].

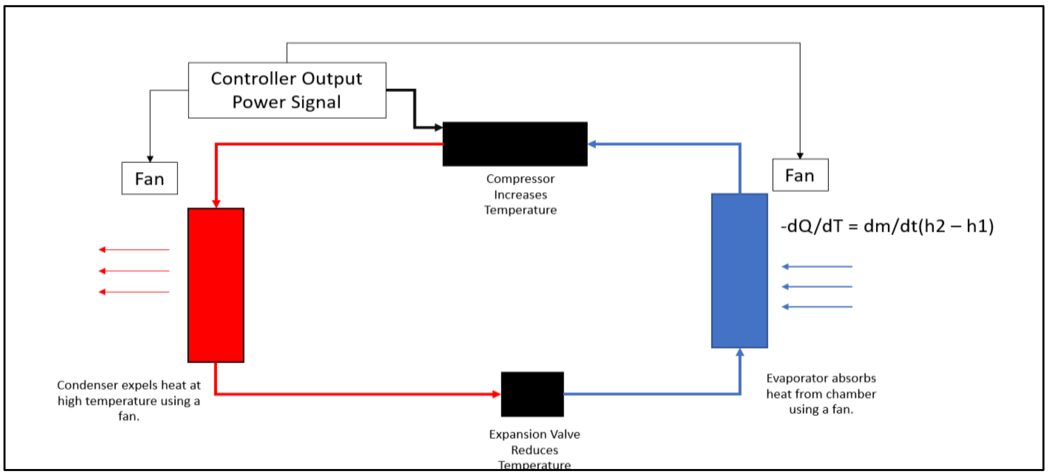


Figure 8: Heat pump refrigeration schematic

Since the compressor refrigeration signal was already installed to the current manual system, the design would only need to implement an electrical connection to the compressor and fans that responds to the control signal. This could be done in the same manner as the electric heater, wherein a solid-state relay is interfaced between the controller and the power supply.

3.3.3 Secondary Cooling Stage

The secondary cooling stage is used when the desired temperature is below -40ºF. A common manner of accomplishing this is via cryogenic cooling, a process in which a liquid refrigerant is introduced to the environment of interest to reduce the temperature of the surrounding environment. The client specified that they currently uses liquid nitrogen as the refrigerant. An electric solenoid valve can be used to control the flow of liquid nitrogen**.** This valve is placed at the end of the liquid nitrogen hose connected to the supply. Note that the mechanically controlled environmental chamber used a liquid nitrogen hose and spray for secondary cooling, so the only design requirement for this subsystem was the digital control of the nitrogen flow valve**.** Based on the output signal received from the controller, the valve opens or closes. As was the case with the electric heater, a solid-state relay is placed between the controller and the power supply connection to the valve. Figure 9 shows a diagram indicating the working principle of the electric solenoid valve to inject liquid nitrogen into the system.

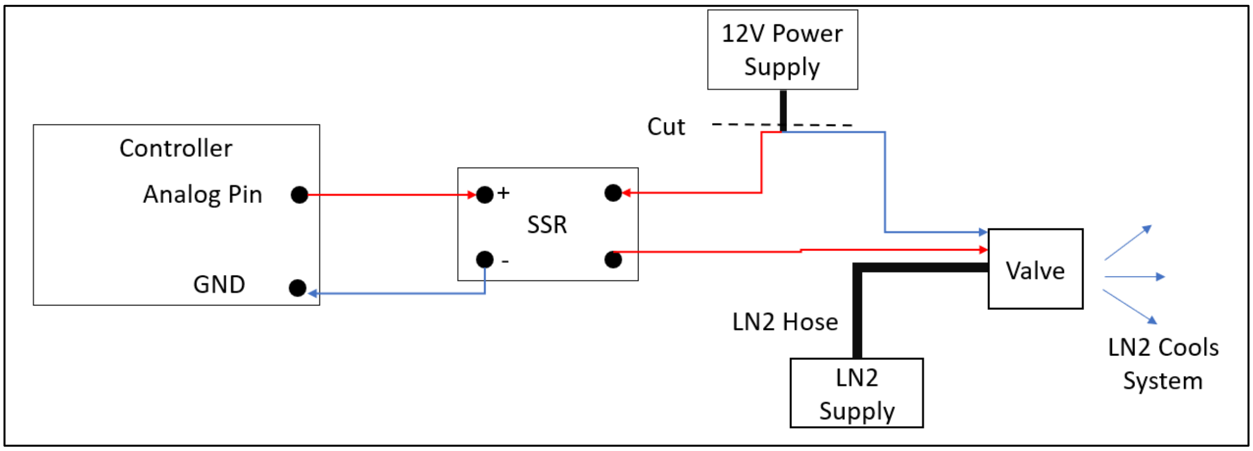


Figure 9: LN2 cooling schematic

3.3.4 Humidity Control

Given that an electric solenoid valve was being used to control the flow of liquid nitrogen into the system, it was ideal to employ a similar technique to inject water vapor in efforts to control the humidity. An atomization humidifier accomplishes this via a series of solenoid valves that keep the flow at the idealized pressure that effectively atomizes the water vapor into the system [10]. Just like the solenoid valve liquid nitrogen control, the atomization humidifier can be connected to the controller's analog input via a solid-state relay.

3.4 Process Variable Measurement

As the working principle of the controller is to run an infinite comparison loop on the setpoint and process variable value, effective measurement of temperature and humidity was a key component of the design solution. Due to the low temperatures in which the chamber achieves (minimum of -100ºF), the options for sensors that work for both temperature and humidity are limited. Therefore, one functional solution was presented at this stage for temperature measurement and one for humidity measurements. Below are the two concepts that can read a process value from the environment and send it back to a controller.

i) **Thermocouple with ADA595 Amplifier Chip:** This design used a thermocouple to read the voltage of the environment and amplified it using an ADA595 chip produced by Omega in a manner that it can be sent back to the controller to be processed. The thermocouple was to be of Type K so it could measure temperatures in the range of -400ºF to 2300 ºF [12]. Figure 10 shows how an amplifying chip can be connected to an Arduino.

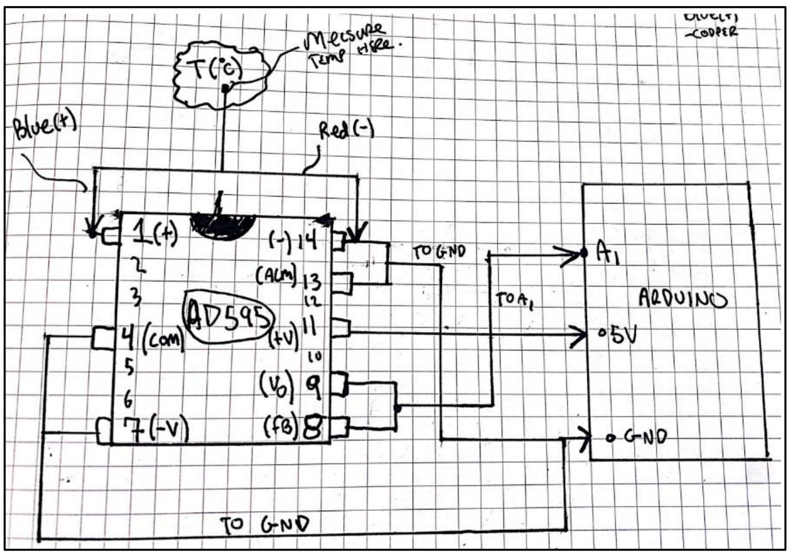


Figure 10:Arduino-Thermocouple interface diagram

ii) **Capacitive Humidity Sensor:** The humidity within the chamber can be monitored with a capacitive humidity sensor. The device consists of a dielectric material placed between two electrodes. When the moisture increases, the absorption of water vapor results in a higher dielectric capacitance [13]. This humidity sensing device could then be connected to an analog input on the microcontroller that reads the value and uses it to control the moisture level within the chamber.

3.5 Feedback System

3.5.1 Data Monitoring and Export

i) **Python Matplotlib:** The Python environment can receive printed serial messages from a controller over the serial port. Once processed, the Python Matplotlib add-on can be used to analyze the data and plot it in real time. Moreover, the user can download the data directly from the Python graphical interface if a download button was configured. Figure 11 shows an example real time Matplotlib plotting of ambient air temperature read from a thermocouple connected to an Arduino.

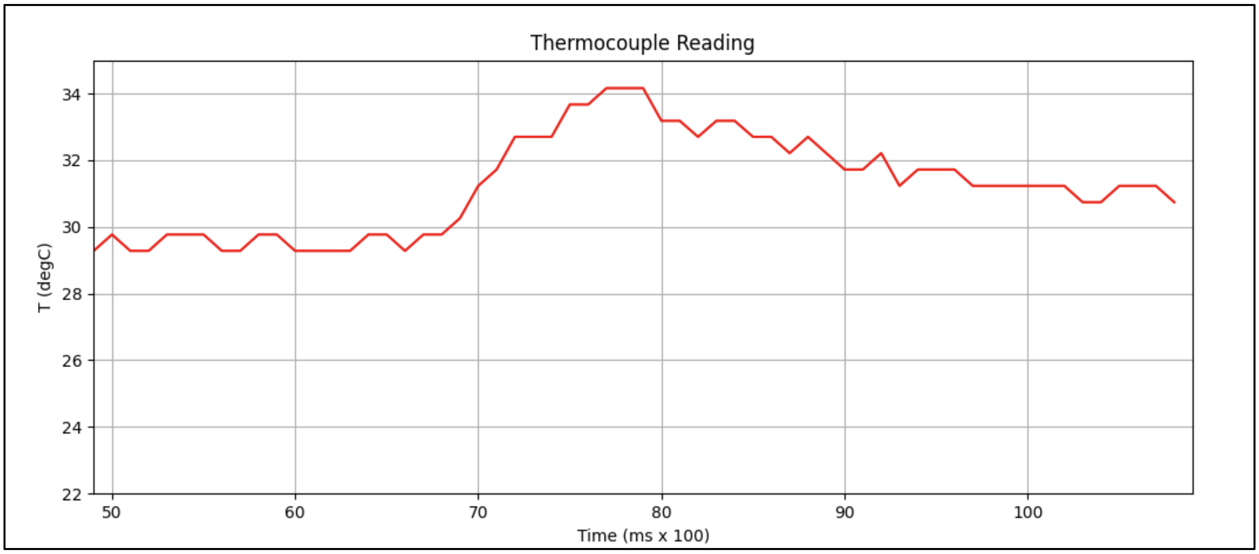


Figure 11: Example Matplotlib output

ii) **SD Card Output:** Like the SD card input, the controller can write the received values to a file that is subsequently uploaded to the connected SD card. Once the user removes the card, they would be able to review the data on their personal computer.

iii) **LabView Data Acquisition System:** LabView allows for the creation of custom virtual interfaces that can display temperature and humidity readings received as analog inputs using built in temperature measurement modules. The use of this system was considered as the software is a core component of the engineering measurements course, taken by all students in this project. Therefore, a custom interface could be implemented with relative ease and could also be maintained by future students.

3.5.2 Error Messages

i) **Python Error Messaging:** There is an available Python module called 'smptplib' that allows emails to be programmed and sent over the simple mail transfer protocol (SMTP) server. Therefore, if the computer was connected to internet, a condition could be programmed into the data feedback system that sent an email when the temperature was out of bounds.

ii) **Raspberry Pi with Built-In Wi-Fi Connectivity:** The Raspberry Pi has a Wi-Fi add-on module which can connect directly to Dalhousie enterprise Wi-Fi. Additionally, the operating system on a Raspberry Pi allows it to connect to an email or SMS server and send messages directly to a user address. This was an all-in-one solution that came at the cost of higher upfront costs and difficulty of programming. If the Raspberry Pi is connected to Wi-Fi, it can also send a CSV of the temperature and humidity data to the operator, which would also satisfy the export functionality.

iii) **Arduino with Built-In Wi-Fi Connectivity:** An Arduino controller can also connect to Dalhousie enterprise Wi-Fi; however, it required the help of a third-party webhook service to send emails/SMS messages. When there is an alert to send, the Arduino can trigger this webhook to send a message to the user. This solution was low cost, easy to implement, and highly customizable as new email addresses/phone numbers could be added to the webhook website directly. The primary downside was the need for a third-party company which may not be reliable in the long term.

iv) **Sound Alerts** Controllers may be fitted with a speaker to indicate when errors occur. By varying the rhythm and pitch of the sounds emitted, the speaker could specify which alert is being triggered. One option was to use a piezo-electric speaker driven directly by the controller using a pulse width modulation signal. The downside to this design was that it may be annoying, and it is also not accessible to hearing impaired individuals.

4. Proposed Designs

4.1 Proposed Design I

To create full-scale conceptual designs, the subsystems discussed in the previous section were combined. This was not done arbitrarily, as certain subsystem designs tended to integrate better with others. For example, if a design employed a Python interface, it was beneficial to also use Python for the feedback system. Each of the proposed designs are formally evaluated based on their subsystems in the following section. Table 2 illustrates the subsystem design combination for the first proposed design.

Table 2: Proposed design I

Subsystem	Design
User Input	Python TKinter Interface
Controller	Arduino PID
Heating Stage	Electric Heating via Solid State Relay
Cooling Stage I	Vapor Refrigeration Cycle
Cooling Stage II	Electric Solenoid Valve for Liquid Nitrogen
Humidity Control	Electric Solenoid Valve for Vapor Humidifier
Process Variable Measurement	ADA595 TC for Temperature & DHT Humidity Sensor
Feedback System	Python MatplotLib W/ Email Alerts

The choice of user interface was Python TKinter. With this design, the user can enter the desired setpoint profile and the Python program then uses a serial write function to send the desired temperature, humidity, and time at each value to the Arduino microcontroller. The Arduino IDE software would then be able to receive the input values and runs this PID control loop using the analog reading of the temperature and humidity sensors. By comparing the current process value to the desired setpoint, the Arduino IDE can communicate to the microcontroller the executed command signal. The temperature sensor proposed was a K-Type thermocouple connected to an Omega ADA595 amplifying chip, while the humidity measurement proposal was a DHT22 capacitive humidity sensor. These measured values can be sent from the Arduino back to the Python environment. The Python interface can plot these values in real time using the Matplotlib extension. Moreover, the Python script can be programmed to send an email alert to the user should the temperature be below -100°F or above 350°F. The design is illustrated in Figure 12.

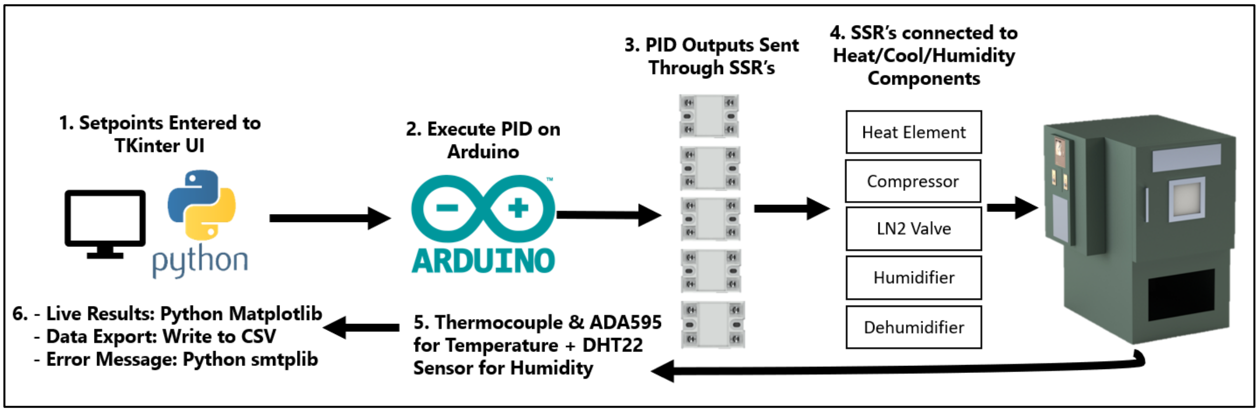


Figure 12: Proposed design I

The main benefit of this design was that it was simple to implement. The team had experience with Arduino control from prior courses, and the teams lead developer had programmed with Python before. Another benefit was that once the sensed values are sent to Python from the Arduino IDE, the Python environment allows for interactive, real time data display with Matplotlib. Moreover, the Python TKinter interface allows the user to download a csv file of the temperature vs. time profile. Finally, Python can send emails to prespecified addresses if it is connected to Wi-Fi. Therefore, a loop can be run within the Python script that sends an error alert to the client should the received values be outside of the safe bounds.

The main downfall of this design was that it required the connection to a host PC. Therefore, the client would need to either buy a PC to install next to the controller or connect his laptop each time he wishes to control the machine. Another potential concern is that future maintenance of the system will require a functional understanding of the Python language and how it communicates with the Arduino, a concept that may not be fully understood by every future student.

4.2 Proposed Design II

Table 3 illustrates the subsystem design combination for a second proposed design.

Table 3: Proposed design II

Subsystem	Design
User Input	SD Card
Controller	Arduino with PID
Heating Stage	Electric Heating via Solid State Relay
Cooling Stage I	Vapor Refrigeration Cycle
Cooling Stage II	Liquid Nitrogen Solenoid Valve
Humidity Control	Vapor Humidifier with Solenoid Valve
Process Variable Measurement	ADA595 TC for Temperature & DHT Humidity Sensor
Feedback System	SD Card Removal for Export. Arduino Email & Siren Alerts for Warnings.

The chosen user input for the proposal was an SD card loaded with an Excel spreadsheet. Values on an SD card can be uploaded to the controller through an installed SD card interface. The chosen controller was an Arduino fitted with built-in Wi-Fi connectivity. This design used the same temperature and humidity devices as the first: a heating element for heating, mechanical refrigeration and nitrogen spray for cooling, and a vapor humidifying system for humidity control. The same measurement devices were also used. To send an email/SMS alert, the controller could directly connect to Wi-Fi and trigger a third-party webhook service. This service would then send an SMS message and email to the desired user. Additionally, this design featured a piezo-electric speaker which could be activated and driven by the controller. The controller could be programmed to play various patterns and pitches to indicate a particular error message. This design is illustrated in Figure 13.

A benefit of this design was its minimalistic nature, as the SD card input allowed the user to use the familiar Excel program to create an input file. Creating a spreadsheet profile also permitted a high degree of customizability in process parameters. The drawbacks of this design included the fact that an Arduino cannot connect to an email server directly and relies upon a third-party webhook company. Moreover, having no user interface required the team to display the measured values on a non-user-friendly Arduino print screen. Further, the use of the SD card required that the client purchase a supply of SD cards that can store large datasets. The design is illustrated in Figure 13.

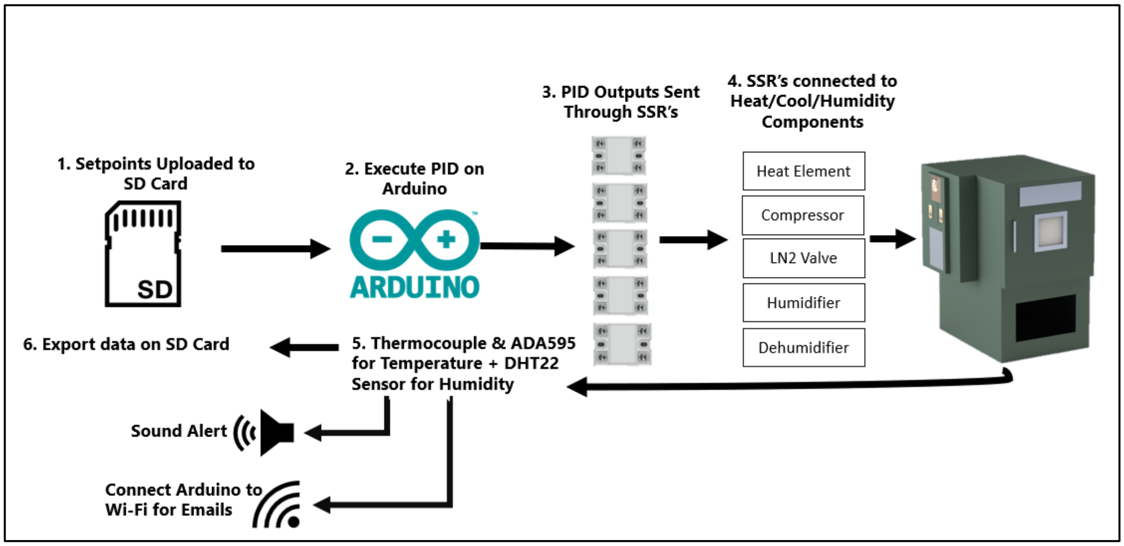


Figure 13: Proposed design II

4.3 Proposed Design III

Table 4 illustrates the subsystem design combination for the third proposed design.

Table 4: Proposed design III

Subsystem	Design
User Input	Raspberry Pi Touchscreen
Controller	Raspberry Pi PID
Heating Stage	Electric Heating via Solid State Relay
Cooling Stage I	Vapor Refrigeration Cycle
Cooling Stage II	Liquid Nitrogen Spray Controlled by Solenoid Valve
Humidity Control	Vapor Humidifier w/ Solenoid Valve

Process Variable Measurement	ADA595 TC for Temperature & DHT Humidity Sensor
Feedback System	Raspberry Pi Touchscreen W/ SMS Alerts and an Emailed CSV file.

The choice of user interface was a Raspberry Pi touchscreen. A Raspberry Pi can be programmed to show selectable options for setpoints and rise times. The choice of the controller hardware and software for this design was also the Raspberry Pi, as it would integrate best with the interface. The choice of temperature and humidity sensors was the ADA595 thermocouple for temperature and the DHT humidity sensor. This design used the machine's existing systems for electric element heating, mechanical refrigeration, liquid nitrogen cooling, and vapor humidifying. The feedback system for this design featured the serial output printed by the Raspberry Pi being sent back to the Raspberry interface wherein a Wi-Fi connection is made. The script could then programmed in a manner that sends email alerts if the temperature is below -100 ºF or above 350 ºF. Moreover, a Raspberry Pi can be programmed to process the values read from the thermocouple and write them into a CSV file. This CSV file was to be emailed to the user so that they could view the temperature and humidity versus time profiles after the curing cycle was complete. This proposed design is shown in Figure 14.

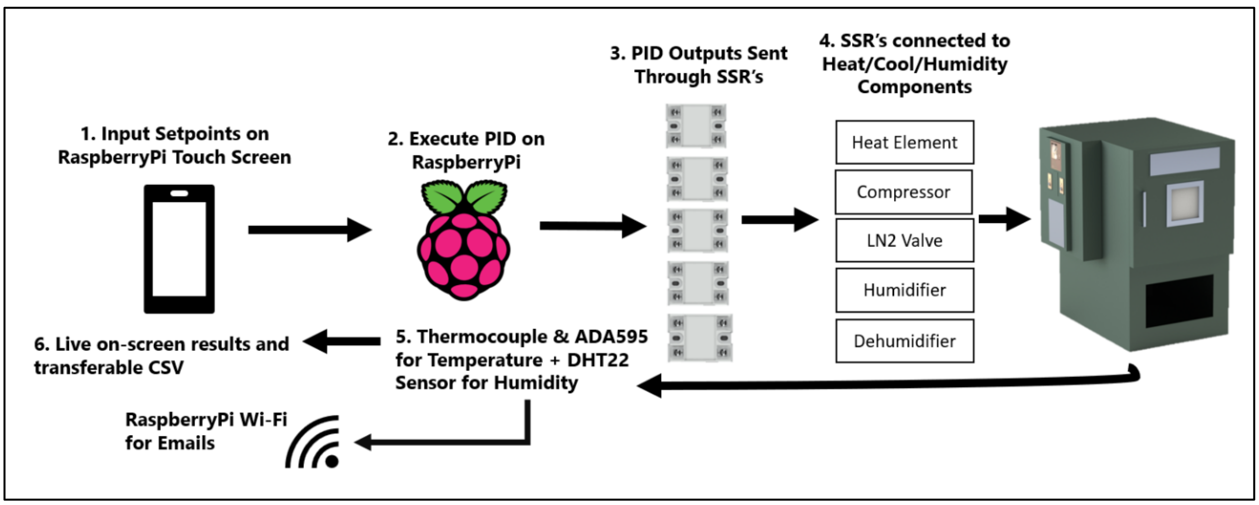


Figure 14: Proposed design III

The main benefit of this design was that a Raspberry Pi touchscreen can be installed directly to the face of the chamber which thereby eliminated the need for a computer connection.

This ability to interact directly with the system is what is seen in most commercially available products. However, this design has downsides, specifically the steep learning curve imposed by Raspbian operating systems. In addition to the extra time required to implement this more ambitious design, a Raspberry Pi is more expensive than an Arduino.

5. Detailed Design and Iterations

5.1 Prototyping

Once three alternative design concepts were created as a combination of several potential subfunction possibilities, initial prototyping was conducted. The initial prototyping stage was used to learn more about each of the subfunctions listed above. Simple prototypes were formulated at this stage so that lessons learned in this process could help guide the evaluation of concepts. As electronics and programming were seen as the largest design challenges, the first prototype was used to improve knowledge of interface design, microcontroller electronics, and output feedback. To do so, a very simple Arduino LED brightness controller that received inputs from a user interface was produced. An Arduino breadboard was wired such that the output pins emitted the desired voltage if a condition in the programmed algorithm was met. Once the output was connected to an LED, the brightness needed to be measured it in a manner that it could be used as feedback for the control algorithm. Through this process, the shortcomings of the prototype could be evaluated, and new and improved features could be implemented into the final design.

The control the LED brightness, a Python TKinter interface that allowed the user to input a setpoint was developed. Learning how to enable this user input and properly send it to the microcontroller was a very big step in the design process as these fundamentals were to be implemented in the final design. To control the LED brightness, a simple ON/OFF Arduino algorithm was written. If the setpoint was above the current brightness, the LED output pin was powered on, and if the setpoint was below, it was powered off. The measured brightness came from a photoresistor, which was connected to the 5V pin along with the analog input pin on the Arduino to determine whether the LED needed to be brightened or dimmed. The constructed prototype can be seen in Figure 15.

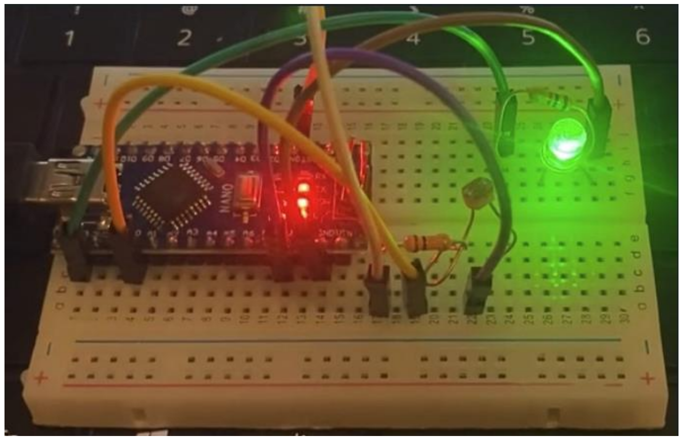


Figure 15: LED brightness controller

After the completion of this first prototype, it was still unknown as to how to connect the microcontroller output pin with an existing piece of electromechanical machinery by interfacing an SSR between the output pin and a temperature control element. Moreover, it needed to be learned how to make these connections safely and effectively without damaging any equipment. To accomplish this, an Arduino controlled toaster oven was developed. The user interface was the same as the LED brightness controller, just for the fact that the user entered a real temperature value in degrees Fahrenheit rather than an arbitrary brightness value. Figure 16 shows the interface.

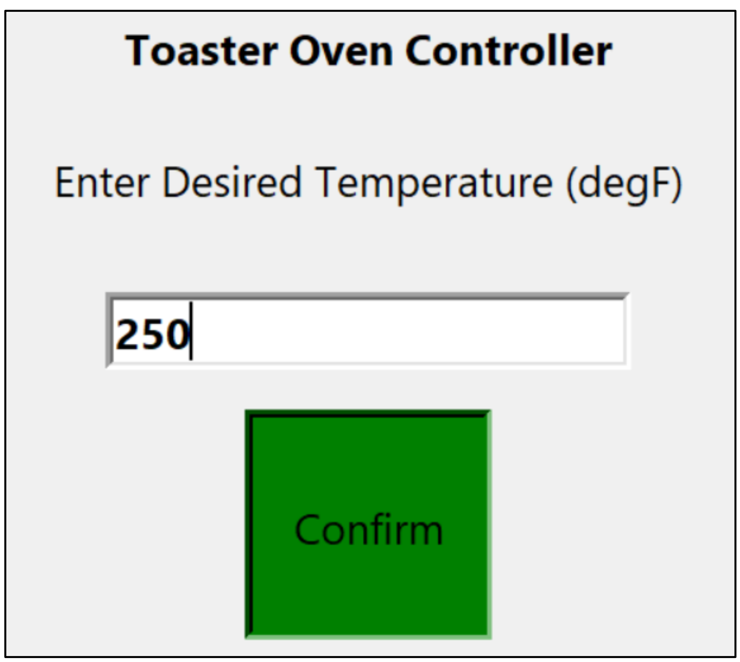


Figure 16: Python interface for toaster oven

The program to read and store the Python inputted values within the Arduino environment was already setup from the LED brightness controller. To measure temperature a Type T thermocouple was placed inside the oven. The two end wires of the thermocouple were then passed through the side of the oven via two drilled holes and are inserted into the Arduino board. One key bit of information that was learned in selecting the thermocouple for this prototype is that in insulating jacket must be placed around the thermocouple wires to withstand the high temperature within the oven. Further, the team used this prototype to learn how to properly read a thermocouple voltage difference into the Arduino using an AD595 amplifying chip. Once the chip output voltage was connected to the Arduino input pin, it needed to be converted to temperature to be compared to the entered setpoint. This was found using Equation 1, noting that the 5.0 represents the voltage range for an analog to digital conversion, the 100 converts mV to V, and 1024 is the max value an analog output can return [14].

$$temperature\ ({}^{0}F) = 1.8 * \frac{(\ 5.0\ *\ analogRead(input_{pin}) *\ 100.0)}{1024.0} + 32 \qquad (1)$$

To supply heat to the toaster oven, an SSR was installed and connected to the heating element, the power supply, the output pin, and the ground. A safe position for the SSR to rest just inside the right side of the toaster oven casing was identified and the machine shop technicians fastened it on. Help was then sought from electrical technician Jon MacDonald to properly interface the SSR between the 120 VAC cord and the heating element. This represented a big design step as the safe positioning and connection of an SSR would be an integral part of the final full-scale installation procedure. Figure 17 shows the installed and connected SSR.

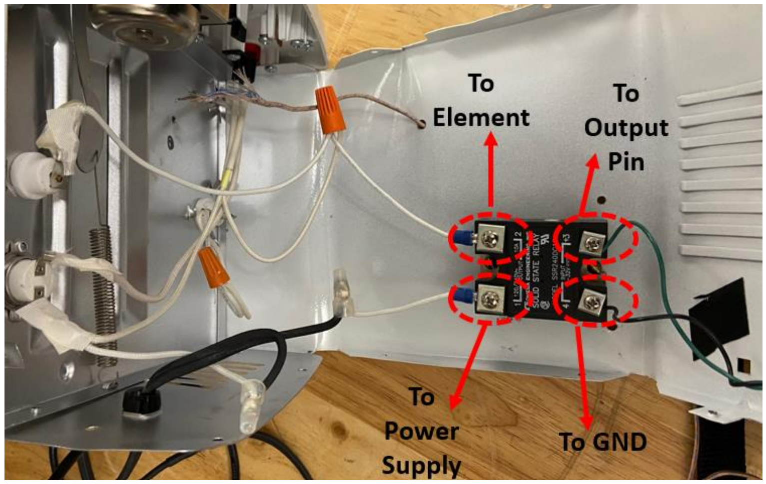


Figure 17: SSR connection to toaster oven

With both the setpoint value and the measured value stored, and the heating element was connected to the Arduino, a feedback loop was programmed to control the output based off the difference between the desired and current values. The algorithm was originally meant to be the simple ON/OFF design used in the LED brightness controller. However, this stage of the prototyping process led to the discovery that Arduino has a built-in PID library.. The function requires the desired setpoint and the measured temperature as inputs in addition to the gain constants, which were arbitrarily chosen as no tuning would be conducted on the toaster oven. Thus, the serial read setpoint and the temperature value read above were called to the PID function, which outputted a voltage. This voltage was then assigned to the output pin (pin 13 in this prototype). The complete wiring diagram for the prototype is shown in Figure 18.

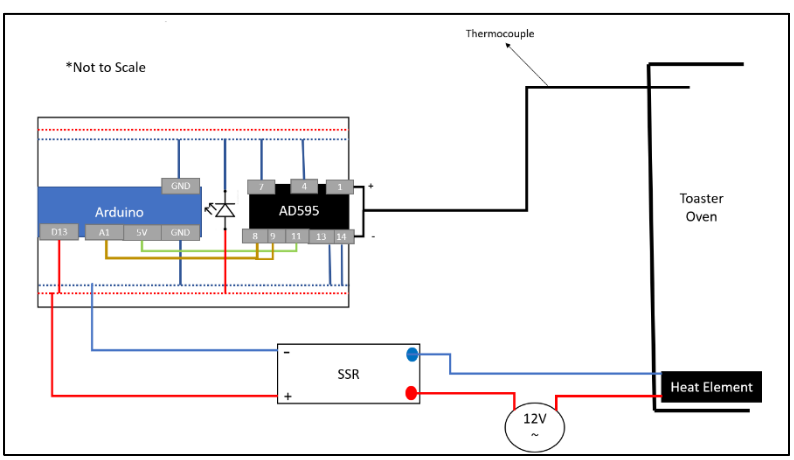


Figure 18: Toaster oven controller wiring diagram

Once the prototype was able to successfully heat up the toaster oven, the prototype functionality was extended by experimenting with the data exportation process. To do so, the thermocouple reading were stored in the Arduino program and written back to Python through the serial port. These values received by Python were plotted in real time using Matplotlib. Figure 19 shows a screen-capture of the Python interface displaying the current temperature of the toaster oven. The user can use this feature to monitor whether the temperature is close to or at the desired value.

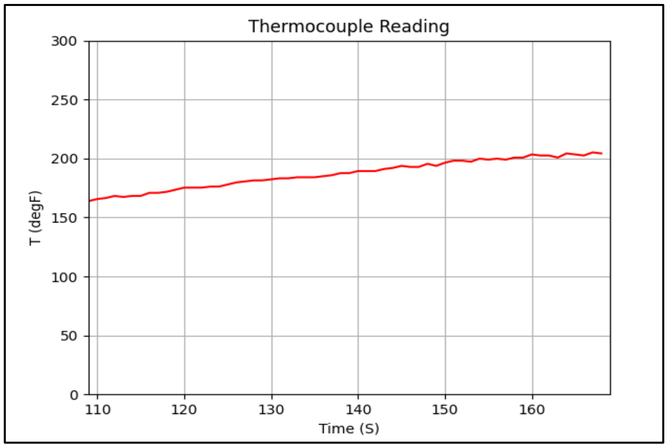


Figure 19: Live toaster oven temperature graph

A function was then written into the Python script that automatically wrote the temperature measurements to a CSV file on the host computer. The user can access such a file to create a plot of the temperature vs. time to see the true values at which the material was treated and assess whether the system was performing optimally. Figure 20 shows the plotted results found by entering a setpoint temperature of 250 ºF into the TKinter interface.

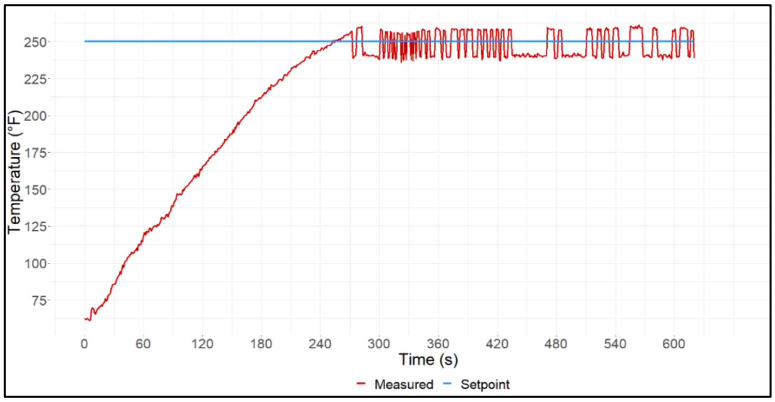


Figure 20: Post-Processing of toaster oven data

From the exported data, it is seen that the temperature response is quite fast, and the toaster oven heats up to the desired setpoint in only 6 minutes. While this ramping action exceeds the requirement of 2 degrees per minute, the toaster oven is much smaller than the environmental chamber, this ramp would need be optimized once the design was integrated at full scale. A full source code for both the Python interface and the Arduino control algorithm for this prototype is available in Appendix B.

5.2 Initial Design

5.2.1 Initial Design Selection Process

In addition to insights gained in the prototyping process, research findings and engineering reasoning skills were used to evaluate the proposed concepts. To assign a score to each alternative, the individual rating of the subsystems that make up the full design were summed. The subsystem concepts were rated in three different categories: usability, functionality, and feasibility. Usability was chosen as a category because the goal of the project is to modernize a pre-existing system, and it was desirable to ensure subsystems were user friendly, accessible, and reliable. Functionality was chosen to ensure the system operated optimally and the potential for faults and systems failures were minimized. Finally, a feasible design was sought after so that the outlined requirements could be met in the allotted time. Even in the case of an extraordinary design, failing to implement due to a lack of feasibility would be a failure on the team's end. For each of these three criteria, each subsystem was rated from 1 to 3. A rating of 1 signified a poor score, while 3 corresponded to a good score. The results of the subsystem design evaluation can be seen in Table 5. Note that any described subsystems that are already installed to the pre-existing chamber were not rated – this includes the heating element, the compressor-based refrigeration cycle, the liquid nitrogen cooling valve, and the humidifier.

Table 5: Subsystem design evaluation

Subsystem	Alternative	Usability	Functionality	Feasibility	SUM
User Interface	Python	2	2	3	7
User Interface	Raspberry Pi Touchscreen	3	3	1	7
User Interface	SD Card Input	2	2	2	6
Controller Hardware	Arduino	2	2	3	7
Controller Hardware	Raspberry Pi	3	3	1	7
Control Algorithm	ON/OFF	2	1	2	5
Control Algorithm	PID	2	3	2	7
Data Export	Python Matplotlib and CSV Download	2	3	3	8
Data Export	SD Card Output	2	2	2	6
Data Export	Raspberry Pi Email CSV	2	2	2	6
Alert System	Arduino WIFI Module	2	2	1	5
Alert System	Python Email	2	2	3	7
Alert System	Raspberry Pi Wi-Fi Module	2	2	2	6
Alert System	Arduino Siren	1	1	3	5

The following justification is given for non-average scores of 1 or 3.

Usability:

- **Raspberry Pi Touchscreen Interface (3):** This modern design allows for direct installation to the chamber, allowing the user to interact directly with the system. As most mobile phones use touchscreens, the user would be very familiar with the operation. The need for the user to purchase or acquire a host computer would also be eliminated, allowing for a more compact and minimal system.
- **Raspberry Pi Hardware (3): A** Raspberry Pi runs its own operating system (Raspbian OS) which means that it functions as a minicomputer. This would allow for it to be connected to both Wi-Fi and Bluetooth.
- **Arduino Siren Alert System (1):** This design option could be very annoying if it constantly beeped. Moreover, a siren is not accessible or usable for hearing impaired individuals.

Functionality:

- **Raspberry Pi Touchscreen Interface (3):** Preliminary research showed that Raspberry Pi interfaces allow for a large degree of customizability and this interface is very functional in terms of receiving multiple inputs and sending them off to the controller.
- **Raspberry Pi Hardware (3):** This hardware is compatible with several different circuit boards and its output pins can be connected to electromechanical elements. Raspberry Pi based devices are known to be very functional and capable of performing complex tasks.
- **ON/OFF Control Algorithm (1):** This is a very old-school method of control engineering that does not allow for fine tuning of rise time and oscillation bandwidth. Moreover, constantly turning the machine on and off can lead to chattering, hysteresis, and unnecessary power consumption. It was not considered for any design.
- **PID Control Algorithm (3):** This is a more advanced method of control engineering and allows for fine tuning of rise time and oscillation bandwidth. The functionality of the PID algorithm was evident to the group in the development of the toaster oven prototype.
- **Python Data Export (3):** The prototyping process showed the functionality of using a Python environment to receive data from the Arduino. It was shown that the data can be plotted in real time with Matplotlib and written to a downloadable CSV.
- **Arduino Siren Alert System (1):** This design requires someone to be in the room to hear the siren. This is not a highly functional design as if there is no one present and an error occurs, the system could fail.

Feasibility:

- **Python User Interface (3):** The prototype led to a significant amount of knowledge about Python programming as a Python TKinter interface was implemented to successfully control the toaster oven. It was therefore concluded that this is very feasible to implement at full-scale.
- **Raspberry-Pi Touchscreen Interface (1):** No team members had ever run or programmed within the Raspbian operating system. It was therefore deemed that this would present an unreasonable time hurdle to successfully implement this ambitious design. It was also not feasible to spend half of the budget on a Raspberry Pi.

- **Arduino Controller Hardware (3):** The development of the prototype gave the team a high level of comfort in wiring an Arduino board. It was therefore deemed that this was a feasible hardware selection that could be easily integrated to the full-scale system.
- **Raspberry Pi Controller Hardware (1):** As opposed to the Arduino, no team members had ever wired a Raspberry Pi board or completed any mechatronic projects using this hardware. It was therefore deemed unnecessary and unfeasible to use a Raspberry Pi microprocessor when the familiarity with the Arduino was so high.
- **Python Data Export (3):** It was shown in the prototyping process that this is a very feasible design. Writing a CSV file with a python data frame is also trivial to implement. The design being used for the toaster oven could be used directly on the full-scale system.
- **Arduino Wi-Fi Alerts (1):** No group members had ever installed the Wi-Fi module to the Arduino and research showed the process may be more complicated than originally thought. Moreover, this would require a third-party webhook service and a connection to Dalhousie enterprise Wi-Fi which further reduces the feasibility.
- **Python Email Alert Systems (3):** While this was not incorporated in to the prototype, the team's comfort level programming in Python provided great confidence that this was a feasible design that could be easily implemented.
- **Arduino Siren Alert Systems (3):** This was a trivial design that only required a speaker be connected to an Arduino output pin, making it very feasible selection that could be easily implemented.

We then rated each design by summing the scores of their subsystems. The outputted results of this calculation are summarized in Table 6.

Table 6: Design evaluation results

Design	Score
Design I: Arduino PID w/ Python Inputs & Outputs	37
Design II: Arduino PID w/ SD Card Inputs & Outputs, Sound Alerts	31
Design III: Raspberry Pi PID w/ Touchscreen Input Display, Emailed Outputs	33

Due to its feasibility - in large part thanks to the prototyping process - the first design which featured an Arduino controller with a Python interface edged out the less feasible Raspberry Pi concept. Engineering judgement and prototyping reflection led to the belief that the Arduino controller is a functional and feasible option. It was also found to be a far more cost-efficient option than the Raspberry Pi.

5.2.2 Initial Design - Manufacturing Plan

With a final design selected, a manufacturing plan was generated to guide the fabrication, installation, and integration of the full-scale design. This manufacturing plan was presented to the machine shop and electrical technicians on January 25th, 2023. First, the plan for fabrication and connection of electrical circuitry connections was devised. With this generated, a manufacturing plan was organized to oversee the fabrication and installation of an electronics case was designed. The original manufacturing plan for both the connection and casing portions is listed below:

1. **Electrical Circuitry Connections**:

   To integrate the designed electronic circuit to the full-scale system, a circuit board needed to be designed. This board would feature the Arduino Nano controller, the thermocouple amplifier and connector, and the screw terminals for the SSR's. Electrical consultant Jon MacDonald recommended that components first be soldered to a perforated prototype board (perfboard) before moving to a more permanent printed circuit board (PCB) solution. The perfboard would serve as a semipermanent solution for the controller, as solid male to female wire connections could be made and all components would be helped in one place. The main benefit about using a perfboard first was that edits and alterations to the wiring setup could be made throughout the testing and installation procedure. Once the perfboard was prepared, the next step in the original electrical installation plan was to decipher the inner electrical connections of the chamber, just as Jon did when we purchased the prototype toaster oven, and it was desired to know what each of the inner wires signified.

2. **PCB Board Housing/Case**:

The initial manufacturing plan included a 3D printed case to house the circuitry. A detailed drawing for case is shown in Appendix C. The case design involved the circuit board being inserted in a press fit manner. Tolerances are fit for a 3D printer, and the final print can be post processed to ensure a secure fit (via sanding or hot gluing). The main benefit of the case was that the plastic is nonconductive, and this provides electrical shielding for the circuitry. The case design also featured a press-fit 3D printed cap. The design was also parametrised to allow for dimensional changes to fit a breadboard, perfboard or PCB.

5.3 Modified Design

5.3.1 Deviations from Initial Design

As discussed above, the plan in the initial final design was to house the Arduino inside a 3D printed casing and connect it via a port hole to the SSR's mounted to the inside of the chamber. However, the manufacturing plan was devised before the team was granted key access to the chamber's laboratory. Therefore, iterations based on new findings were made and a modified final design was devised. The first iteration drew from an inspection of the chamber, revealing an opening where the electrical components are held. The original chamber door was likely removed during the original attempt to employ a digital controller for this system ten years ago. As this opening to the electrical circuitry presented a safety hazard due to the live AC current flowing when the machine is operating, a door now had to be designed to seal off this opening. While this initially presented an increase in scope, it led to the idea of mounting the Arduino and relays directly to this door. Since it would be the door for the electrical components, the connection wires would be easy to access. Moreover, this would eliminate the cost and labor required to 3D print an Arduino case. This also protected the circuitry from the external environment, as it was now housed inside the machine itself.

To design the door, the opening was first measured. It was found to 330.1 mm wide and 432.95 mm high. Moreover, on the sides and top of the opening, the spacing between the rear brackets was measured to determine the added overlap required and to position the through holes. The opening is shown in Figure 21 with annotations showing the rear bracket types

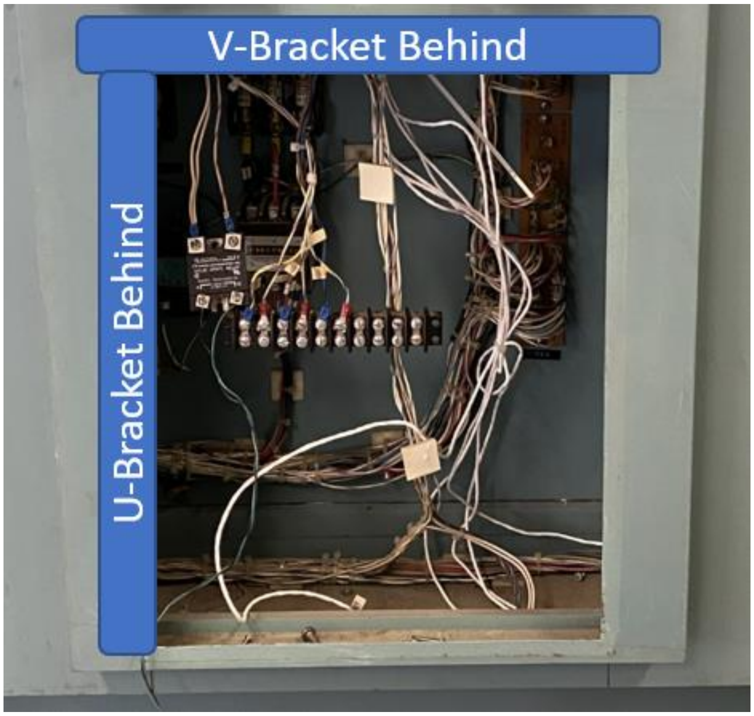


Figure 21: Chamber opening near electrical equipment

The measurements of the opening and the resultant calculations for the door size and hole positions is summarized in Table 7. Note the door is simply just an aluminum plate that was fastened over top of the opening with four self-tapping screws.

Table 7: Aluminum door plate dimensions

Parameter	Dimension (mm)
Height of Opening	432.95
Width of Opening	330.1
Distance between side edge and U-bracket (both sides)	6.25
Distance between top edge and V-bracket (both ends)	7.03
Width of door plate	337
Height of door plate	440
Lateral distance to through holes	3.5 mm
Vertical distance to through holes	4 mm

The detailed engineering drawing for this aluminum door plate is found in Appendix C. Figure 22 shows the fabricated plate fastened to the opening. Figure 23 shows the SSR's, and prototype board mounted to its back.

Figure 22: Aluminum plate door for electrical compartment

Figure 23: Electrical components fastened to rear of aluminum door

In addition to reconfiguring the mounting design, detailed inspection of the chamber led to changes in the wiring diagram. The initial design plan was to connect the SSR's directly to the temperature control components. However, it was found that their positions were too hard to access without disassembling the machine. Therefore, the wires from the original mechanical controller had to be traced to see which ones led to which component. Then, knowing from the client that the mechanical controller still worked, the end wires from the mechanical controller were removed and subsequently attached to the new SSR's. In short, instead of connecting the controller to the components directly, the wires from the mechanical controller were removed and connected to the digital controller. The main benefit of this design is that it allowed for the freedom to not consider heat dissipation when mounting our SSR's, as they were now connected in series with the original relays that would absorb most of the current. The new SSR's are still necessary to help to electrically isolate the microcontroller and prevent damage. Figure 24 shows how the current mechanical machine is wired, while Figure 25 shows how it was used to setup the new digital controller by following the original wires and connecting to them.

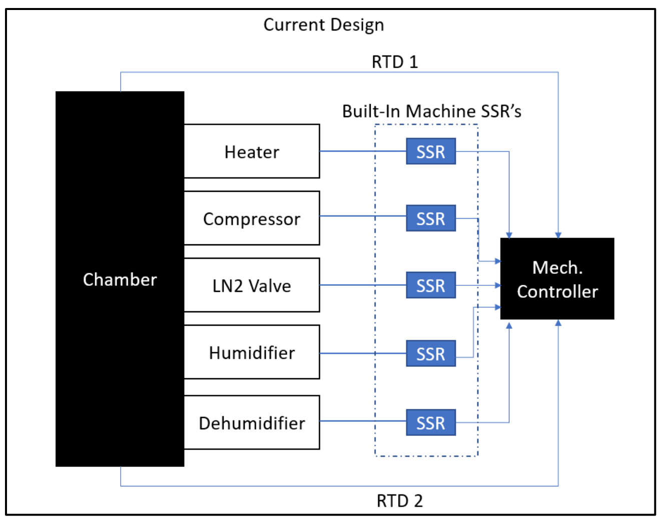


Figure 24: Original mechanical controller wiring

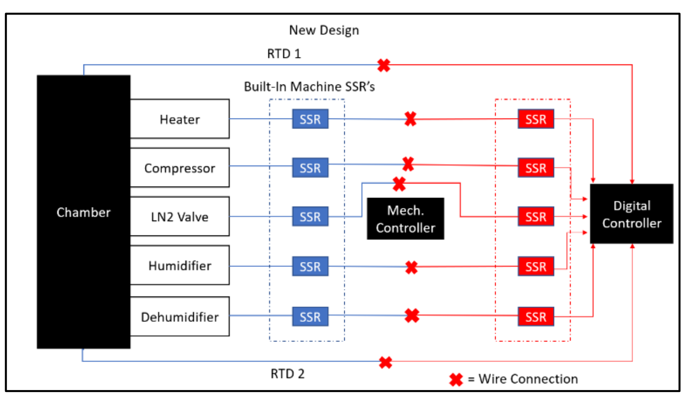


Figure 25: Updated wiring design for digital controller

This method of tracing wires also allowed for the current temperature measuring devices inside the chamber to be found. As they were already installed into the side of the chamber, it was deemed that they would be more reliable and convenient to connect to these temperature signals. Moreover, the sensors were already fastened and secured through the side of the chamber. However, the devices were RTD's, which required a few design changes. The RTD's had three wires, two black and one red. The nominal resistance between the red and black wires at room temperature was 200Ω, meaning they were likely PT200 RTD's. To read the temperature coming from the RTD, MAX31865 amplifiers by Adafruit were ordered and implemented. By consulting the MAX31865 user manual [15] and having Jon MacDonald solder the necessary panels, the amplifiers were safely connected, and the temperature was successfully read from the RTD's. Note that in Adafruit tutorial the source code, the initialized nominal resistance was changed from 100 Ω to 200 Ω. Figure 26 shows a circuit diagram of how the RTD was connected. This change is reflected in the final wiring diagram, found in appendix C.

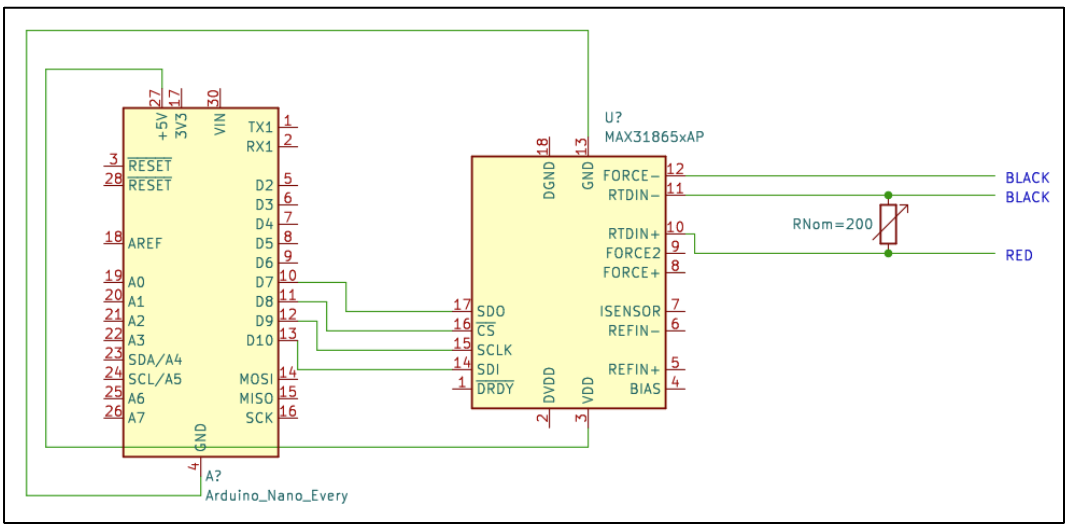


Figure 26: Wiring diagram to RTD's with MAX31865 amplifier

Another iteration made in the final design stage was the implementation of a Kalman filter for temperature readings. It was discovered in the prototyping process that the signal generated by the thermocouple was very sensitive to noise. While the RTD's being implemented fastened securely through the side of the chamber reduced the level of noise, it still became an issue when disturbances were applied. While this acceptable for determining an average temperature, a noisy signal is not optimal for use in feedback control. As output commands are derived from the difference between the measurement and setpoint, continual fluctuations in the measurement can cause instability in the control output. Moreover, when the temperature is near the setpoint, the measurement noise will lead to the machine components rapidly turning on and off. This can place unnecessary stress on the equipment, and a fluttering control output to the components can lead to inefficient power use and the system performance can become less than optimal [16]. Therefore, a Kalman filter was added to the design to help not only stabilize the measurement but send a more accurate reading to the controller. The algorithm is summarized below [17].

i. Define initial state x to room temperature, set measurement noise covariance R to 0.1 (as specified by manufacturer), set noise covariance Q to 0.00001, and initialize the covariance and Kalman gains, P and K, to 0.

ii. Each time a temperature reading, z, is generated from the input signal, update the Kalman gain, the state, and the covariance. This is done as:

$$K_t = \frac{P_{t-1}}{P_{t-1} + R}$$

$$x_t = x_{t-1} + K * (z - x_{t-1})$$

$$P_t = (1 - K) * P_{t-1} + Q$$

For the next temperature computation, K_t, x_t, and P_t are used as the priors. To show the effectiveness of the Kalman filter, a simple experiment was conducted. At room temperature (72 $^{\circ}$F), measurements were recorded on the RTD with disturbance being applied to the sensor by flicking it a few times over the course of a minute. The temperature derived solely from the analog input signal is plotted against the Kalman filter state estimate, x_t, in Figure 27. As expected, the Kalman filter estimate is far more consistent and tends to the true temperature. The temperature reading derived from the analog input however, spikes at each sampling time, and large fluctuations are found when the external disturbances were applied to the sensor.

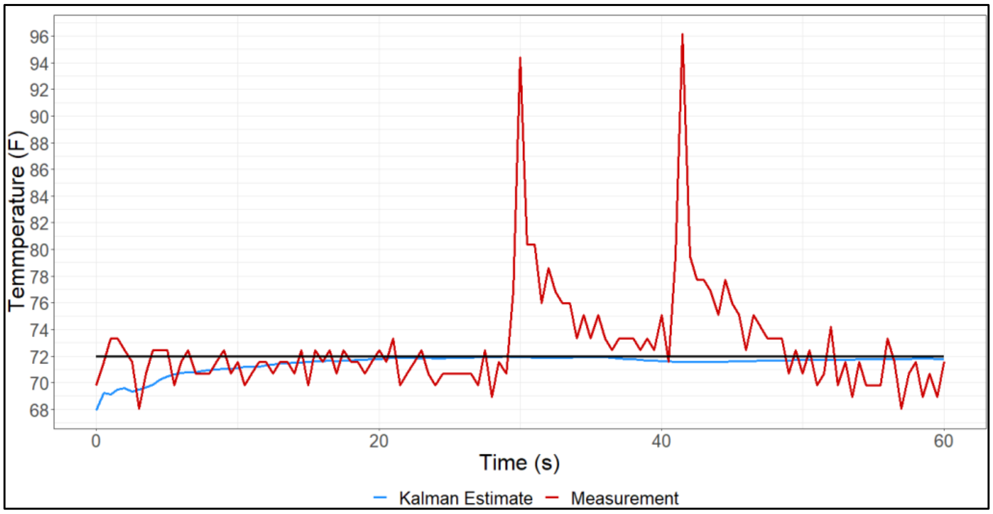


Figure 27: Kalman filter implementation

5.3.2 Interface and Controller Programming

As much of the success of this project was dependent on successful programming, this section has been added to detail the design and logic flow of the software source code. The full source code is available in the Appendix B. There are four main parts to the software:

1. **User Interface:** The user interface must accept multiple temperature, humidity, and time values and send them to the controller. To do so, the GUI designed in Python TKinter that was programmed for the toaster oven prototype was upgraded to allow not only for humidity and time to be sent, but for multiple of these temperature-humidity-time sequences to be written to the controller. To do so, a table was created on the GUI with temperature, humidity, and time columns. If the send button was pressed, the values in each row were appended to a dataset. This dataset was then written to the Arduino controller through the serial port. Note that a delimiter, ",", was placed in between each value as this would be how the Arduino would separate the values. Also note a '\n' was placed before the dataset to let the Arduino know that there was new setpoint data. A flowchart for this code is found in Figure 28.

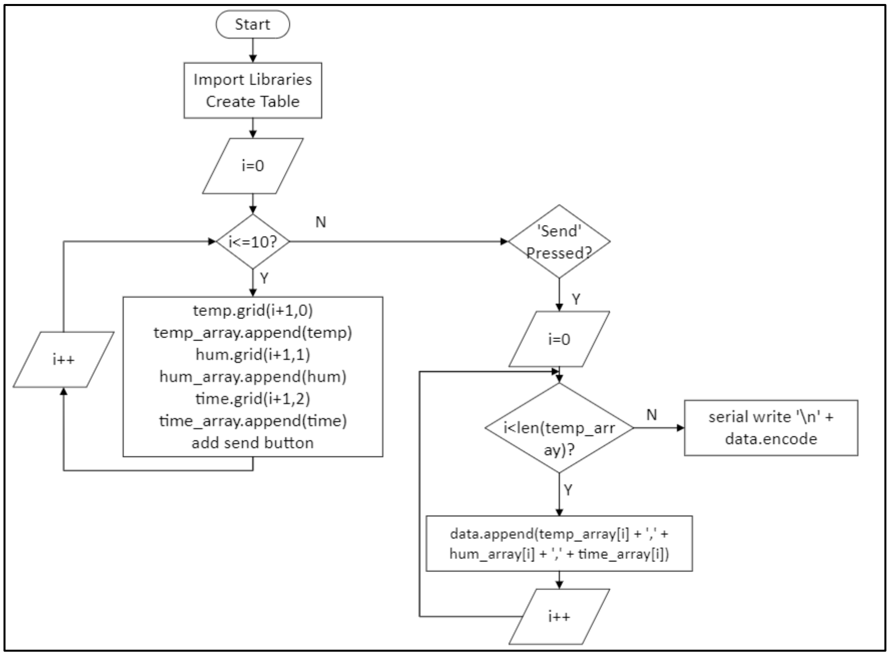


Figure 28: Code flowchart for Python GUI

2. **Setpoint Reception:** Receiving a matrix of values over the serial port is more complex than receiving a scalar or string. Therefore, for the task of receiving the setpoints sent by serial communication from Python to Arduino, a separate function was designed. It is housed within the void loop statement of the main Arduino code, and its first task is to check whether there has been an input on the serial port. If so, it begins an iterative process of receiving the Python data by making use of the delimiters "," and "\n". It iterates this loop until the incremental variable is equal to the length of temperature readings sent. A flowchart for this setpoint reception function is shown in Figure 29.

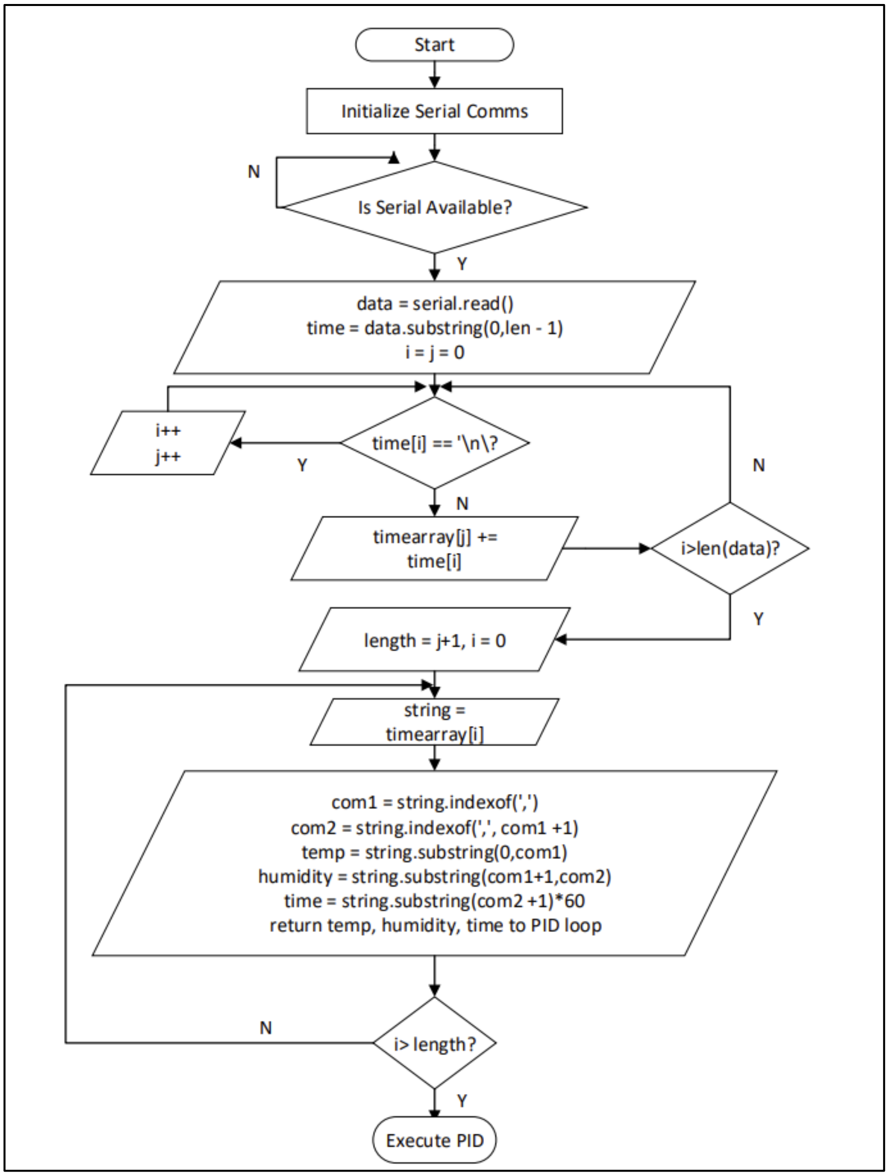


Figure 29: Code flowchart for receiving inputs on the Arduino

3. **PID Controller:** With the setpoints now received and handled by the Arduino, the controller algorithm could be executed. The algorithm continually compares the desired hold time against the internal on-board microprocessor timer, before executing the PID function using the setpoint and sensor values. The PID function is installed from an external library. The flowchart for this code is shown in Figure 30.

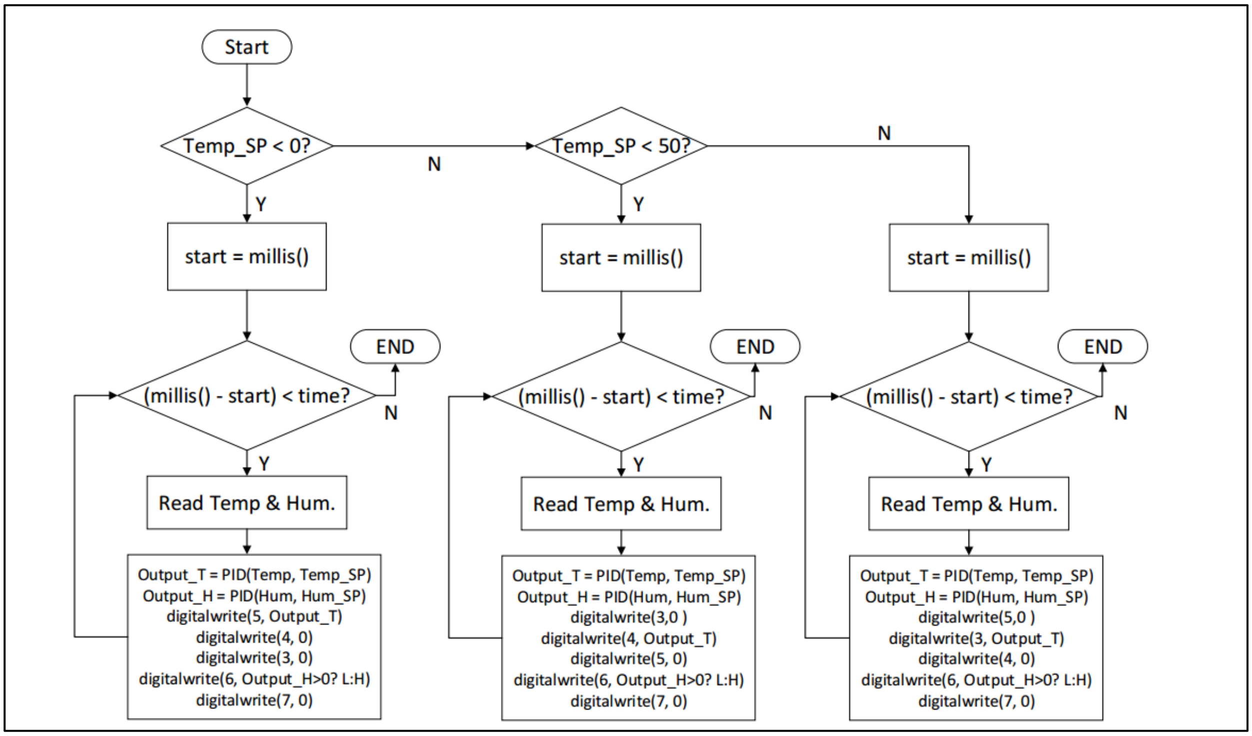


Figure 30: Flowchart for PID control algorithm

4. **Data Retrieval:** The final step was to retrieve the sensor measurements using Python. Python would handle these measurements to display live results, populate a CSV file, and send error messages if necessary. The flowchart for this code is found in Figure 31.

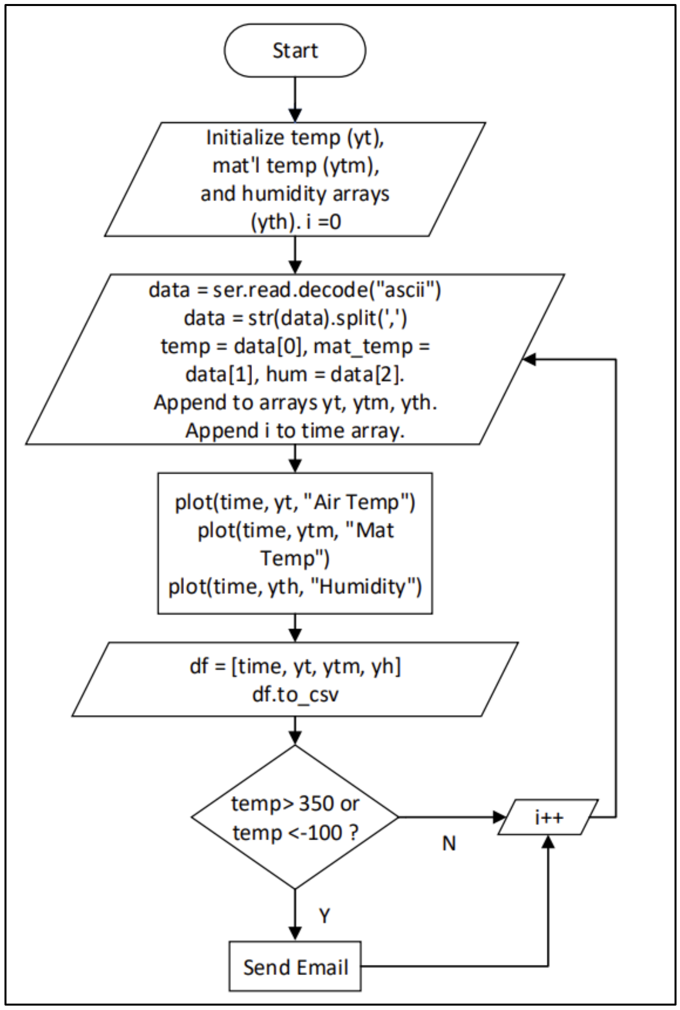


Figure 31: Code flowchart for live plotting, CSV writing, and error e-mails

5.3.3 System Tuning

To get the system to perform adequately and meet the rise time and oscillation requirements, a tuning process was conducted to determine the optimal gains for the PID controller. Rather then conducting tuning tests on the machine, simulation techniques were used as this would significantly reduce the time and labor required. This would also limit the risk of damaging components in the event a combination of PID gains result in an unstable controller. Once the optimal controller was integrated at full scale, an iterative process could be conducted to realize the desired performance of the physical system. The simulation was conducted by designing a MATLAB Simulink diagram shown in Figure 32. A MATLAB script for this simulation is also available in Appendix B.

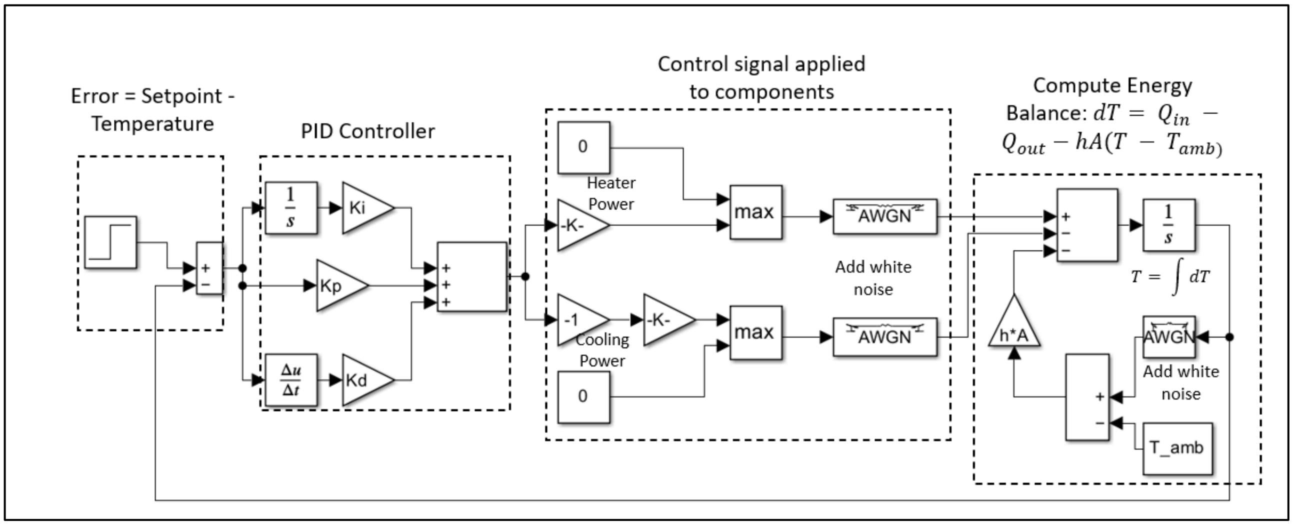


Figure 32: Environmental chamber Simulink block diagram

As shown above, the simulation computes the current chamber temperature by applying an energy balance between the heater input, Q_{in}, the cooling input Q_{out}, and the heat loss $hA(T - T_{\infty})$. To obtain an accurate model, the heater and cooling powers were obtained from the chamber manual and found to be 600W (2047 BTU/hr). This was halved to 300W (1024 BTU/hr) as it is likely the power output has decreased in the 50 years since the chamber was built. Additionally, the chamber surface area and heat transfer coefficient would need to be found to compute the energy balance. Measurements of the chamber interior found that each side had length of 3ft. Therefore , the total surface area for heat loss is the sum of all 6 area faces, or A $= 6 \cdot (3ft)^2 = 54\ ft^2$. The heat transfer coefficient for this system was much harder to find, as the degradation of material in conjunction with the lack of chamber insulation in certain portions made it difficult to find an accurate estimate. To find an estimated heat transfer coefficient and complete the system model, a simple experiment was conducted. The physical system was brought from 85 ^{o}F to 300 ^{o}F and the response was recorded. The gains used on the physical system were $K_P = 10, K_I = 0.01,$ and $K_D = 0.1.$ Simulations were then conducted using these same gains and iterating the heat transfer coefficient. Eventually – when the heat transfer coefficient was set to 350 $\frac{BTU/hr}{ft^2F}$ (2000 $\frac{W}{m^2K}$) ,the two responses matched, and the model was deemed to have an acceptable level of accuracy. While this is a high coefficient of heat loss for a testing chamber, it is indicative of the systems poor insulation caused by the several manually insulated port holes, as well as the degradation in material. The physical response for this experiment is shown in Figure 33, while the simulated result is found in Figure 34.

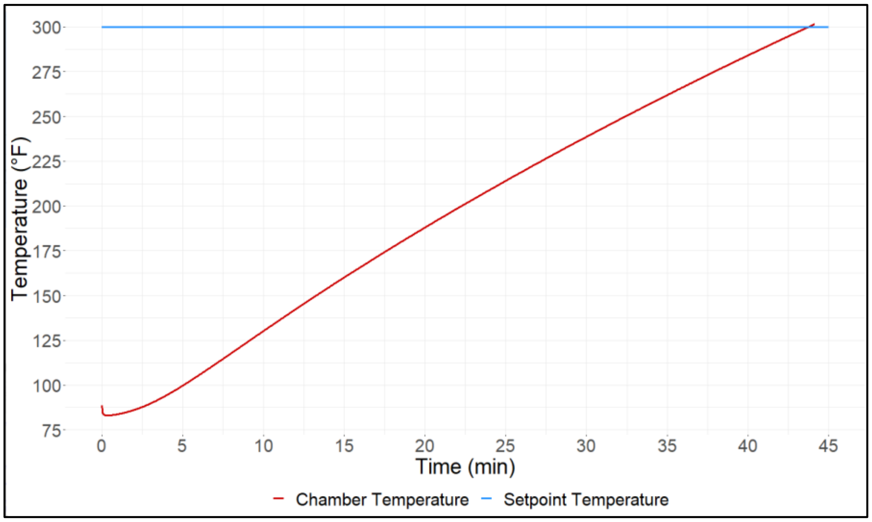


Figure 33: Physical system brought to 300ºF for model validation

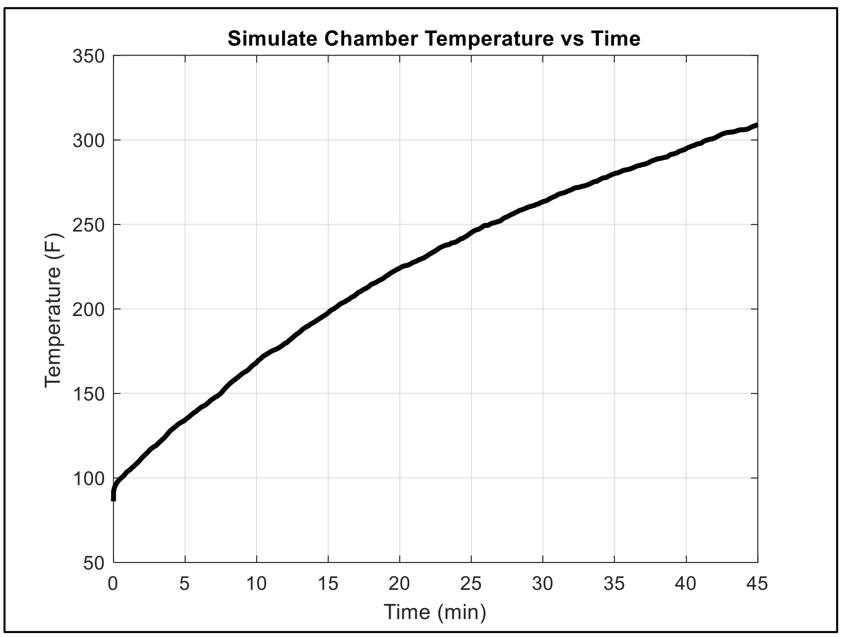


Figure 34: Simulated result of 300ºF step input

With the model now being completed, an iterative tuning process was conducted to find the optimal PID constants that achieved the required 2 ºF rise time with a maximum of 4 ºF overshoot. The experiment above provided a good starting point for the tuning process. With $K_P = 10, K_I = 0.01,$ and $K_D = 0.1$, the temperature rose from 85ºF to 300ºF in 44 minutes, a rate of roughly 5 ºF/min. To achieve the 2 ºF/min rise time, the proportional gain was decreased to 3 and an 8 hour simulation wa conducted with an initial temperature of 85 ºF and a desired temperature of 150ºF. As shown from Figure 35, this resulted in a slightly slower then desired rise rate of (134.5 ºF – 85.43ºF)/(29.15 min) = 1.68 ºF/min.

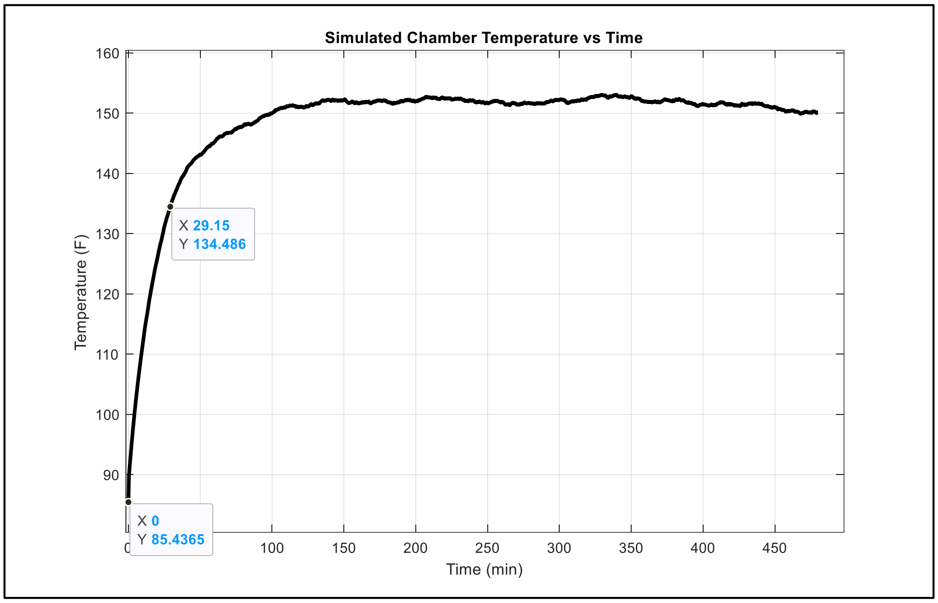


Figure 35: Simulation of system responding with 1.68ºF/min rise rate

To speed the response, K_I was increased to 0.013, as this would not only improve rise time but would contribute to improving the steady state accuracy (source). This resulted in the desired performance as from Figure 36, the rise time is shown to be (135.1 ºF – 85.43ºF)/(24.55 min) = 2.02 ºF/min. Steady state accuracy has also been achieved with all temperatures in the 8 hour simulation are within 4 ºF of the setpoint. These gains could now be implemented to the physical system and further tuning could be conducted to validate the system performs as required.

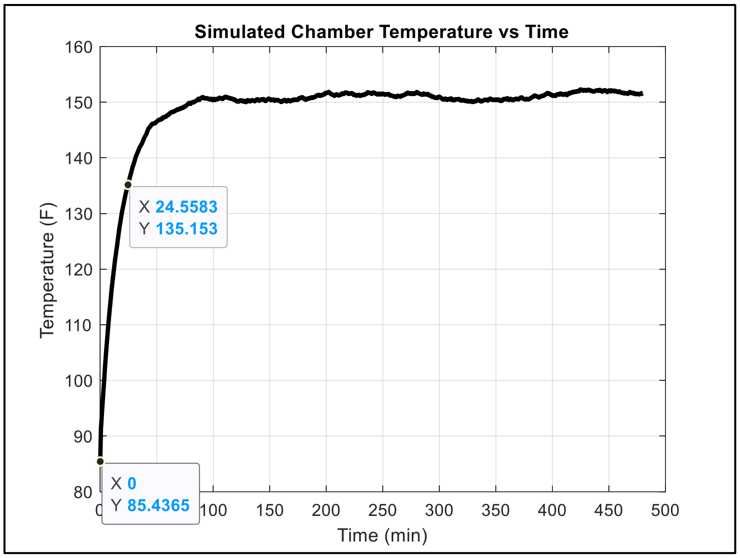


Figure 36: Simulation of system responding with required 2° F/min rise rate

5.4 Final Product

With the necessary iterations made from the initial design, the interface and controller being programmed, and the system being tuned, a final product had been produced and installed. This section provides a summary of the final design selection for each subsystem and reiterates the engineering justification as to why each subsystem design was chosen. The chosen subsystem designs are as follows:

1. **User Input:** As detailed in Section 5.3.2, the user input interface was created using Python TKinter. It allows the user to input a range of temperatures, humidities, and times to hold for a given trial. This user interface design was chosen for three main reasons. First, the teams head developer is very comfortable programming in Python. Second, a Python application can be packaged to a .exe file that allows it to be downloaded and opened on any host computer. Finally, a TKinter interface was used in the initial prototype and the integration process to full-scale was minimized. The user interface is shown in Figure 37.

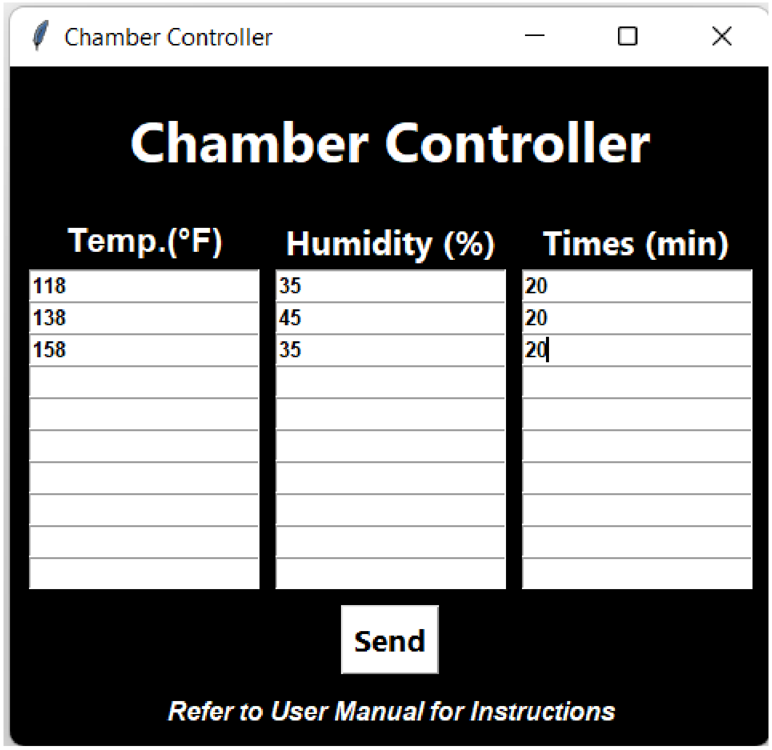


Figure 37: Final design for GUI

2. **Controller Software:** The control algorithm used in this final design is a PID. The team never waivered from this solution over the course of the design process as no other control method allowed for the tuning of the performance specifications of 2^{o}F/min rise time and max 4^{o}F oscillation was discovered. PID control theory was described in Section 3.2.2 and the C algorithm used to execute the PID for this system is detailed in Section 5.3.2.
3. **Controller Hardware:** An Arduino Nano is the microcontroller in the final design. While switching to a RaspberryPi microprocessor was considered, the lower cost of the Arduino in conjunction to the prior knowledge of C Arduino programming possessed by the team led to the selection of this final hardware selection. Additionally, an Arduino Nano was used for all prototyping and therefore the hardware integration to the full-scale system was less complex then if a RaspberryPi was used.
4. **Temperature and Humidity Control:** As described in Section 5.3.1, seeing the chamber in person allowed the team to trace the wires leading into the existing mechanical controller and simply attach them to the new digital controller. This is a

much sounder and safer design then the original design of finding and interacting with the machine components directly. New SSR's were placed between the digital controller and the machine components for heat dissipation reasons. This also helps protect the Arduino.

5. **Temperature and Humidity Measurement:** Much like the reasoning given for temperature and humidity control, their respective measurement systems were designed by tracing the wires off the old mechanical controller. Both the wet and dry bulb RTD's could be read on the Arduino nano, and the RTD's being pre fastened to the inside of the chamber provided less measurement noise.
6. **Data Feedback and Error Handling:** This design solution did not change drastically from the prototype. The live temperature readings are sent through the serial port from the Arduino to Python and are displayed on a live plot. These results are also written to a CSV file for post processing. The only addition is that the script is now programmed to send an email to the user using the simple mail transfer protocol (SMTP) server if the temperature falls outside of the -100°F to 350°F range.

The final design showing the interaction of these six subsystems is detailed graphically in Figure 38.

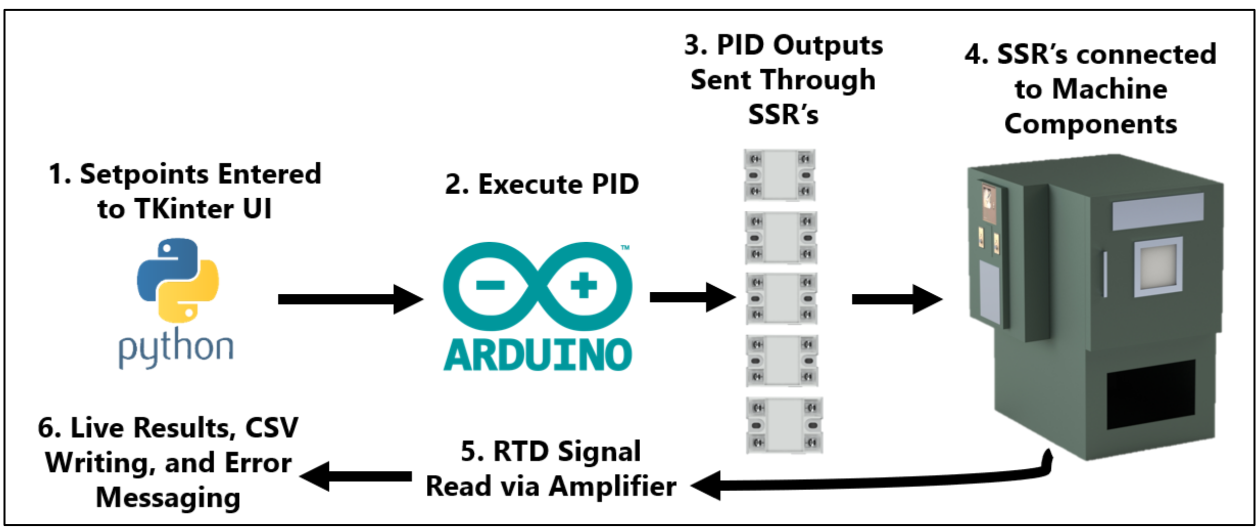


Figure 38: Interaction of 6 subsystems in final design

6. Verification and Testing

With the new digital controller having been installed and integrated at full-scale to control the environmental chamber, a testing procedure was conducted to ensure that the system met all the outlined requirements (Table 1). Note that third party instruments were used to validate the measurements. A *MasterChef* oven thermometer was used for temperature, while an *AccuTime* timer was used for time. The first test was to verify requirement 1.1, which stated that the digital controller shall be able to keep the temperature of the environmental chamber between -100 ºF and 350 ºF. To verify this requirement, the oven was first brought to 350 ºF. It is noted that the proportional gain was increased to 15 for this test to avoid the excessive time required to bring the system to 350 ºF with a 2ºF/min rise time.

Figure 39 shows live temperature readings on the interface trending toward 350 ºF at 54:41. Note the blue LED is connected to the same output as the heating element, so it is on as the system heats up. The door to was then opened and the timer was placed next to the thermometer to validate the temperature measurement. This is shown in Figure 40, noting that the opening of the door is likely the cause of a slight dip in temperature.

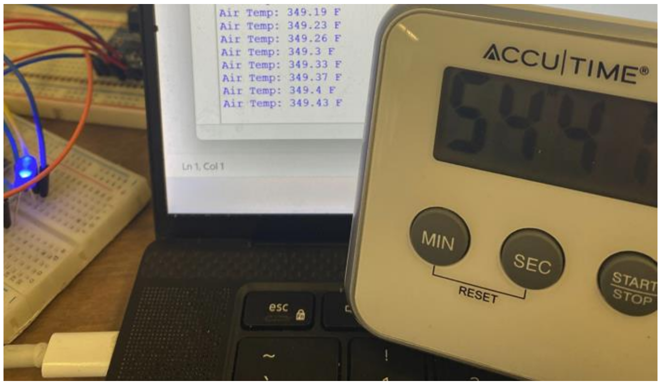


Figure 39: Test 1 – Live temperature trending toward 350F

Figure 40: Test 1 - Third Party thermometer reading just below 350F

As the opening of the door caused the temperature to stay below 350 ºF, the system was still heating up, as verified by LEDs on the side panel shown in Figure 41.

Figure 41: Test 1 - System LEDs still lit up with system below 350F

After the door was closed again, the system continued to heat up and passed 350ºF. When it passed 350ºF, the side panel LEDs turned off as shown below in Figure 42.

Figure 42: Test 1- System LED's turning off as temperature eclipses 350F

From Figure 43, the Arduino LED also turned off once the system hit 350ºF and the system began to cool.

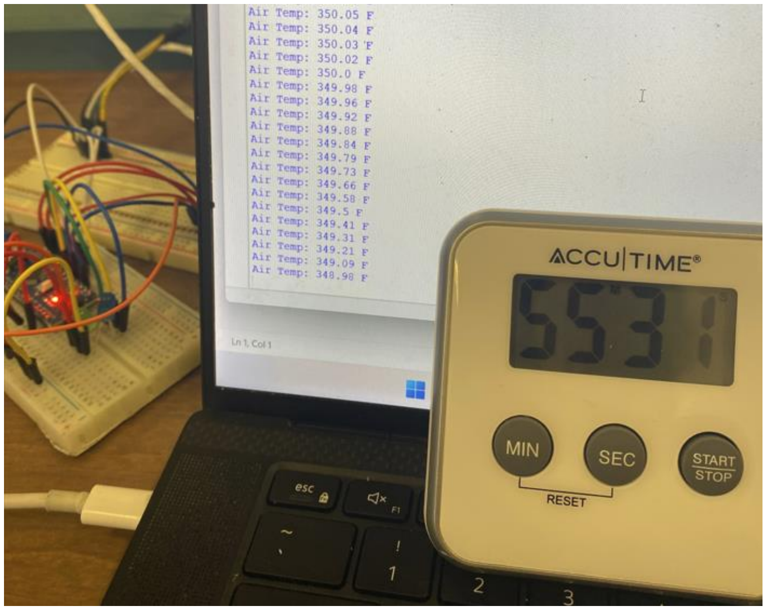


Figure 43: Test 1 - Live temperature readings pass 350F

When the system eclipsed 350ºF, an email alert was sent to the user, as shown in Figure 44. Therefore, in addition to requirement 1.1, requirement 4.1 was also met as the system stopped heating once the system hit 350ºF. Moreover, requirement 4.3 was met as an email was sent indicating that the machine exceeded the maximum temperature range. The data was written to a CSV file and Figure 45 shows a temperature vs. time graph for this first test.

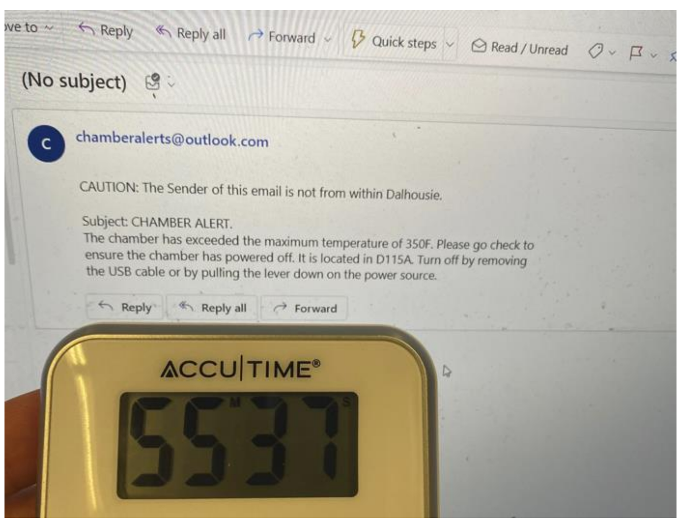


Figure 44: Error e-mail being received in required time as system exceeded 350F

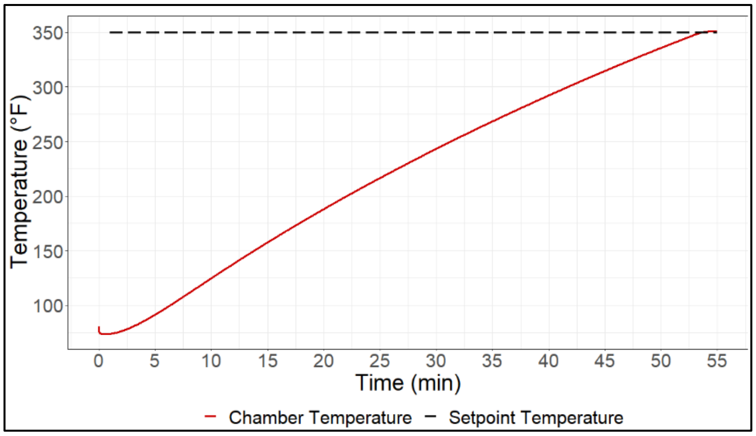


Figure 45: Test 1- Post experiment temperature log for 350F step input

To complete the verification of Requirement 1.1, the system was brought to -100F. The live reading and third party RTD temperature measurement is shown in Figure 46, the error message is shown in Figure 47, and the post experiment temperature log is shown in Figure 48. The proportional gain was once again 15 for this test.

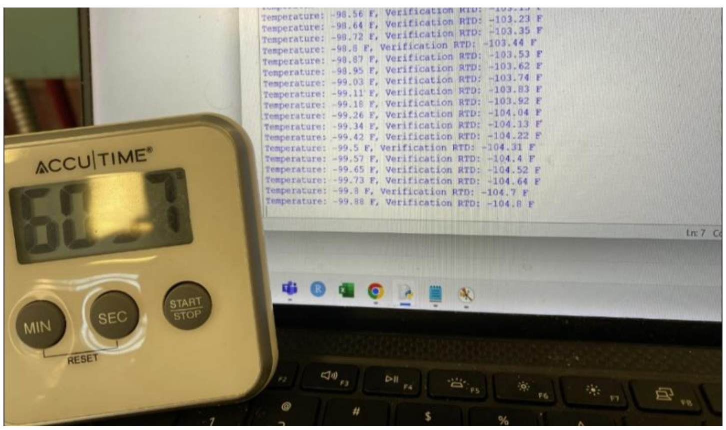


Figure 46:Test 2 - System Being Brought to -100°F

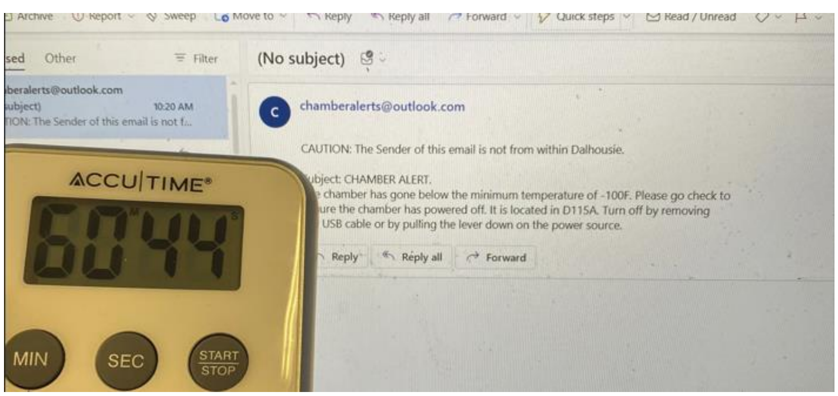


Figure 47: Test 1 – e-mail alert as system drops below -100°F

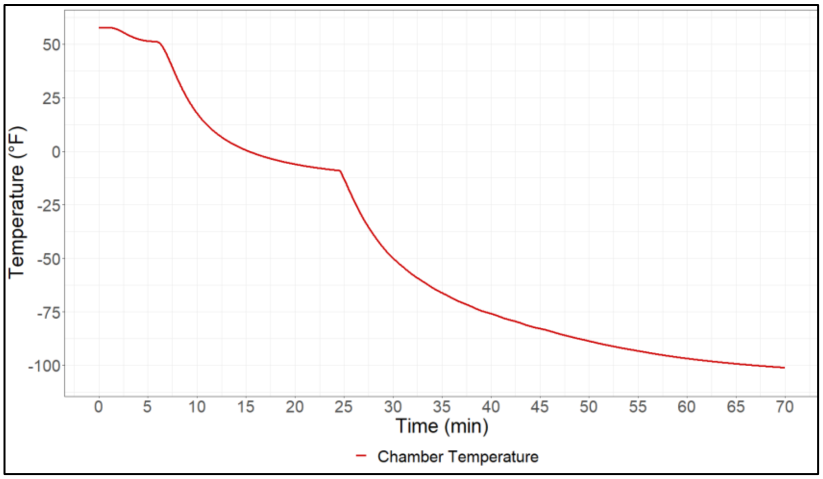


Figure 48: Test 2 – Post experiment temperature log for -100F step input

With the completion of these first two tests, requirement 2.1 was also met, which stated that the user shall have the ability to enter setpoints between -100°F and 350°F that result in the desired system response.

Moving on, requirement 2.3 stated that the user shall have the ability to input a time on the user interface that designates when they would like the system to change to a different setpoint. Three different setpoints and hold times were entered and it was found that the system could effectively meet each one. Figure 49 shows the setpoint and time inputs, while Figure 50 shows the post experiment temperature log. The temperature log verifies requirement 1.4, which states the temperature shall stay within $\pm$ 4 ºF of the setpoint at steady state. The experiment is validated with measurements using the oven thermometer and an *AccuTime* timer at each setpoint. These results are shown in Figure 51 through 53. Note that the higher proportional gain of 15 was also used for this test to avoid excessive rise and settling time. The 2 ºF/min rise time was validated using the tuned in a separate experiment to be detailed later in this section.

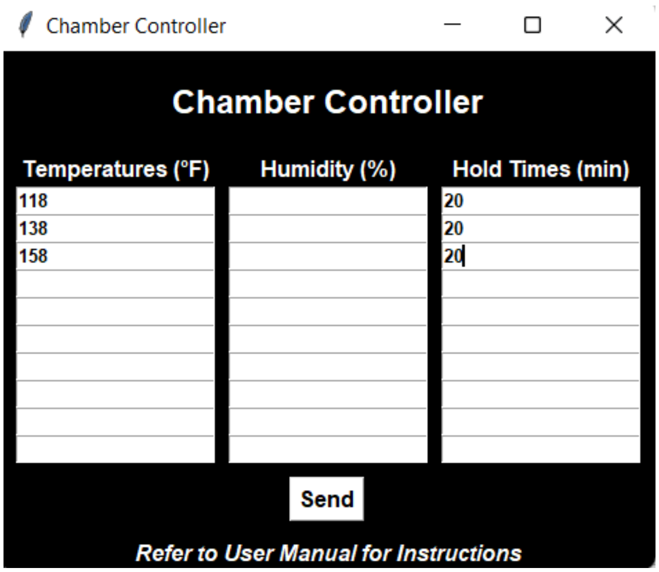


Figure 49: Test 3 - Multiple setpoints inputted to interface

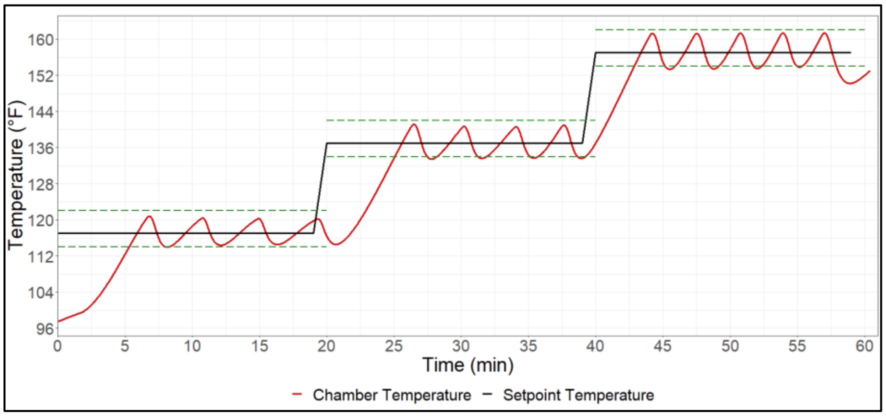


Figure 50: Test 3 - Logged temperature response to multiple setpoints

Figure 51: Test 3 - Verification of 118°F reading at 12:48

Figure 53: Test 3 - Verification of 138°F reading at 32:11

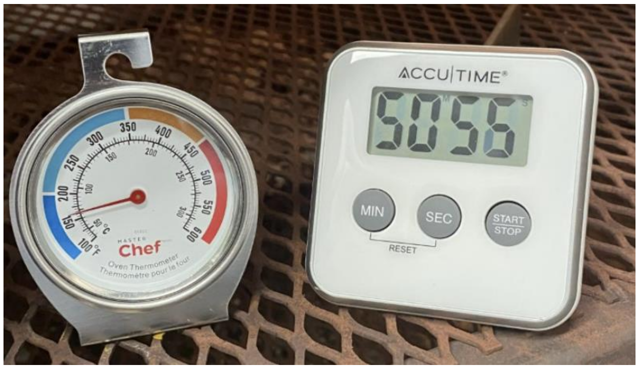


Figure 52: Test 3 - Verification of 158°F reading at 50:56

To ensure the system was also capable of meeting multiple setpoints using both heating and cooling, a similar test was conducted, except for the fact the second setpoint was a lower temperature than the first. This would help show that both the heating and cooling elements were performing adequately. This would also show the systems repeatability. The interface with the setpoints is shown in Figure 54, the temperature log is shown in Figure 55, and the verification photos are shown in Figures 56 through 58.

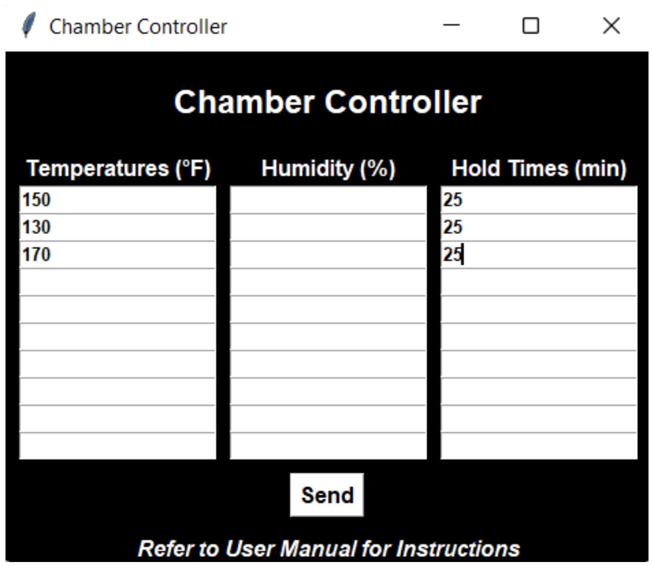


Figure 54: Test 4: Multiple setpoints entered to interface - heating + cooling

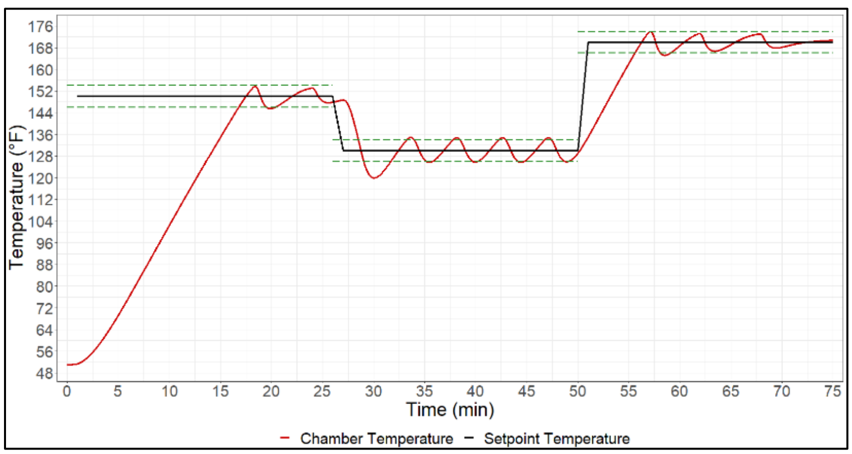


Figure 55: Test 4: Temperature log of system meeting both heat and cool setpoints

Figure 56: Test 4 - Verification of 150F reading at 23:28

Figure 57: Test 4 - Verification of 130F reading at 32:11

Figure 58: Test 4 - Verification of 170F reading at 61:27

With these tests now completed, the proportional gain was decreased back to the value determined in the simulation and the system was heated up at 2°F/min to verify this requirement. With the initial temperature at about 90°F, a setpoint of 120°F was inputted, and it is shown from the temperature log in Figure 59 that the system temperature increased by the required 30°F in a linear fashion in roughly 15 minutes, satisfying the requirement that the system shall have the ability to move to a setpoint at 2°F/min. The verification of this measurement is shown in Figure 60.

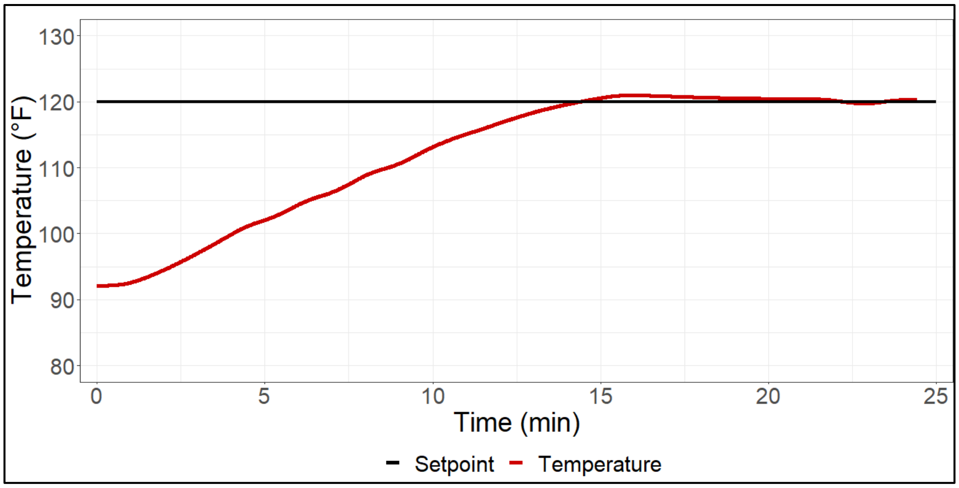


Figure 59: Test 5 - Temperature log showing 2°F/min rise

Figure 60: Test 5: Verification of 120F measurement at 15:00

The humidification aspect of the chamber was then tested. Requirement 1.2 states the system shall have the ability to bring the chamber relative humidity to a maximum of 80 % and a minimum of 20 %. Requirement 2.2 states that the user shall be able to input a desired humidity on the user interface. Therefore, we could validate both requirements simultaneously by first entering 80 % in the user interface and measuring the humidity with a third-party sensor. We then repeated for 20% relative humidity. For these measurements, we once again validated with third party measurements by purchasing a *BestAir* thermometer that also outputs relative humidity. It is noted that inspection of the chamber led to the discovery that the wet bulb RTD was not being fed a recirculating supply of water and therefore the wet bulb measurements were always the same as the dry bulb, resulting in a false humidity reading. To conduct the tests, the wet bulb RTD was manually wetted with a wet cloth. However, there was no recirculation and therefore this means there is a large source of error associated with the humidity measurement.

Moreover, the verification thermometer for relative humidity was found to be very sensitive to disturbance. Therefore, when the door was opened to take a picture for validation of the measurement, the humidity reading tended quickly towards the ambient humidity of 30%. Still, these tests show that both the humidifier and dehumidifier do work as the machine humidity tended towards the setpoints. Moreover, the relative humidity being correct near ambient shows the wiring of the wet bulb RTD is correct. The only thing holding back the full validation of these requirements is the lack of recirculating water supply to the wet bulb thermometer, which is outside of the scope of this project. Figures 61 and 62 show the results of bringing the humidity past 80%. Note the temperature measurement on the thermometer matches the dry bulb temperature on the output screen, further showing the only discrepancy is caused by the wet-bulb sensor. As mentioned previously, the humidity tended quickly to 30% as soon as the door was open.

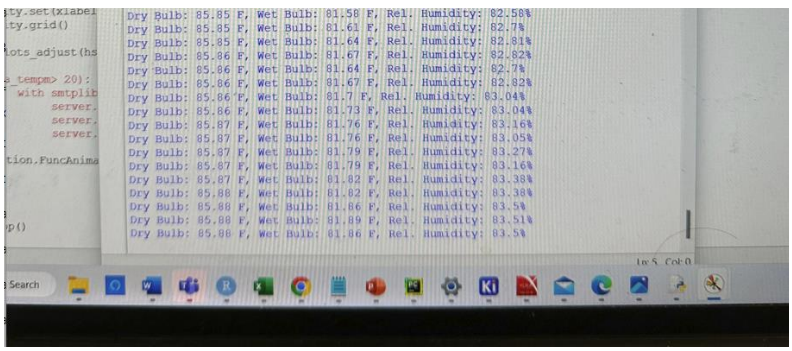


Figure 61: Test 6 - Live humidity headings of over 80%

Figure 62: Test 6 – Third-Party humidity measurement of only 70%

Figures 63 and 64 show the system being brought below 20% humidity.

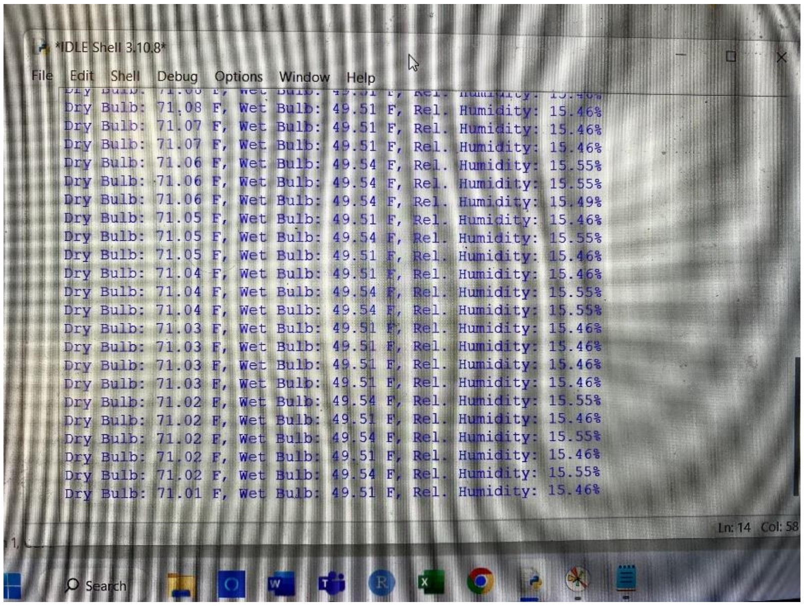


Figure 63: Test 6 – Live humidity readings below 20%

Figure 64: Test 6 – Verification of humidity readings below 20%

With the humidity element now added, all data feedback requirements (3.1 through 3.6) were verified. Requirements 3.1 through 3.3 indicated that the live air temperature, material temperature, and humidity shall be displayed on the interface. Requirements 3.4 through 3.6 stated that these readings shall be written to a csv file that the user can access for data analysis.

To do so, the machine was turned on and left at its current state of 100 ^{o}F and 30% humidity from a prior experiment. Figure 65 shows the verification of the requirement as the live results of the air and material temperatures along with humidity are being outputted. Note the external sensor to connect to the material is a separate PT100 RTD borrowed from Peter Jones. Figure 66 shows the validation of the measurements using the humidity sensor for humidity and the air temperature. The oven thermometer is aligned with the RTD and the sample piece of material to validate its temperature measurement.

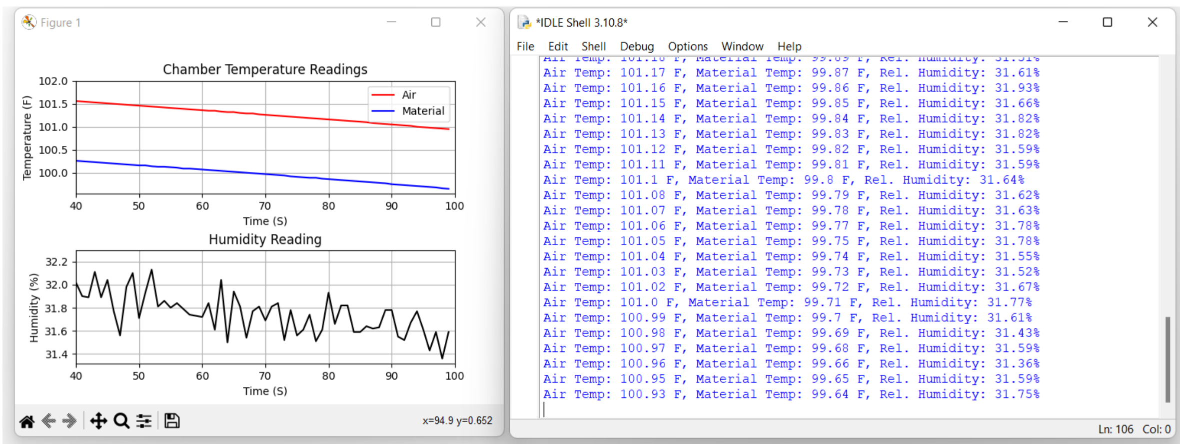


Figure 65: Test 7 - Live air temperature, material temperature, and humidity readings

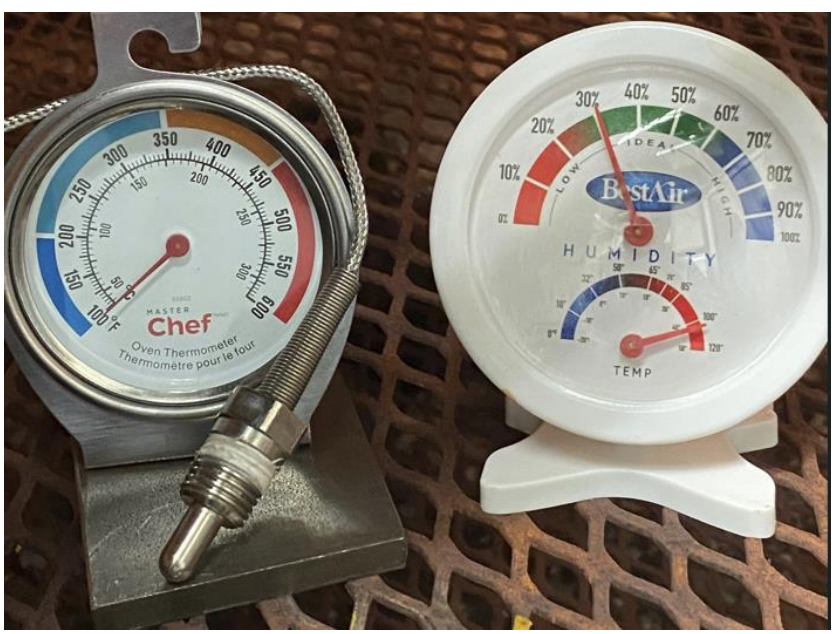


Figure 66: Test 7 - Validation of air, material, and humidity readings

Once the machine was disconnected, the temperature and humidity readings were made available in a CSV file, as shown in Figures 67 and 68. Note that this method of writing the readings to a CSV are how the temperature logs of Figures 46, 49, 52, and 57 were created.

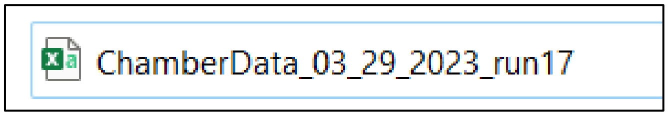


Figure 67: Test 7- Downloaded CSV file verification

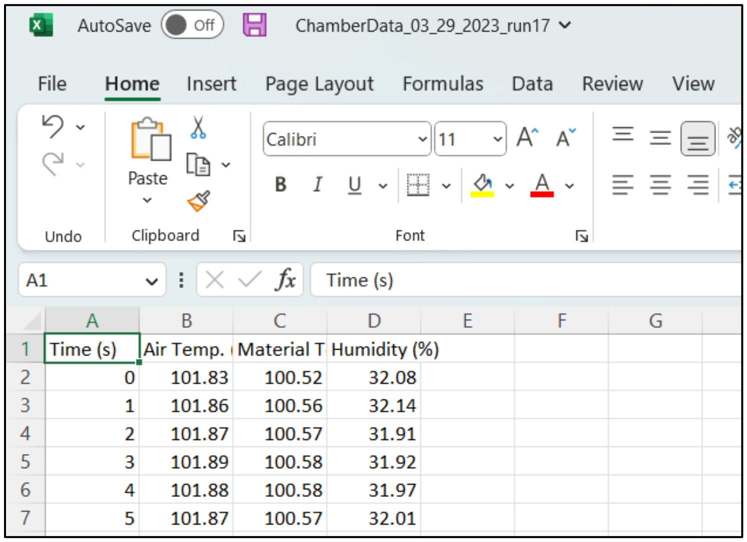


Figure 68: Test 7- Downloaded CSV file showing log of air temperature, material temperature, and humidity

Still left to verify was requirement 4.4, which stated that the user shall have the ability to manually stop the chamber by terminating the program. To test this, a heating command was applied, and the LED turned blue. After a few seconds we saw that if the user disconnected the USB, the machine powered off, and the requirement was satisfied as the user can manually stop the program. The results are shown in Figures 69 and 70.

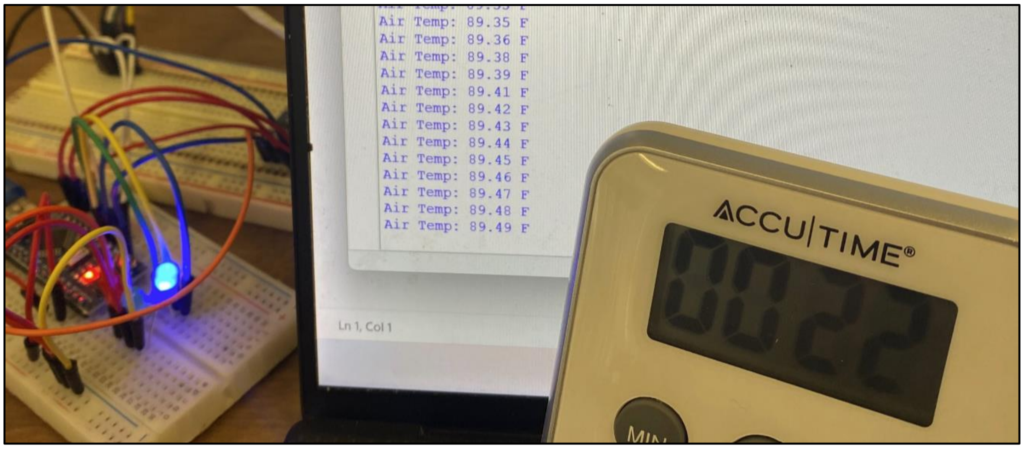


Figure 69: Test 8- Heating LED still on with USB plugged in

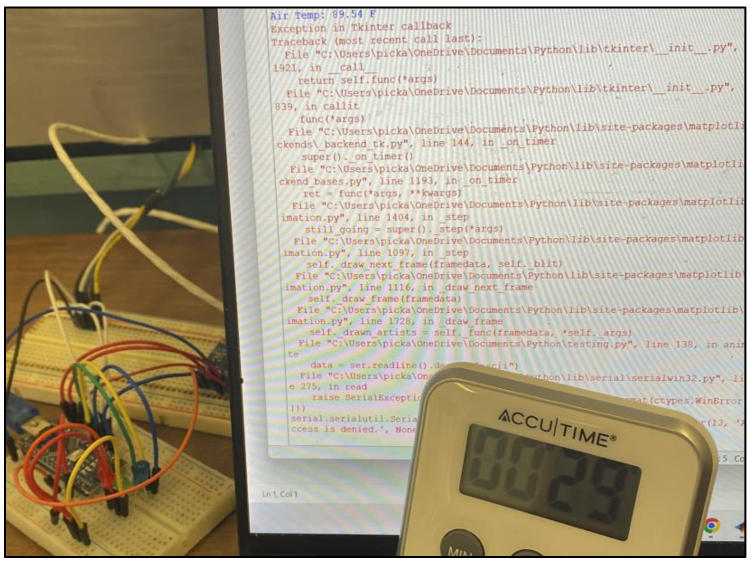


Figure 70: Test 8 - LED off and system terminated once USB is disconnected

Requirement 1.5 was then the only milestone left to meet. It stated that the material temperature sensor must have 6ft of length leftover upon entry to the chamber to allow it to move about the system. Note that this requirement was modified once the team was granted access to the laboratory. The inner measurements of the chamber working area were 3'x3'x3'. Therefore, the maximum lead length required inside the chamber to place the sensor on any piece of the material would be $\sqrt{3^2 + 3^2}$ = 4.24ft. Adding on the extra foot to get the sensor from the circuit through the port hole and into the chamber, the requirement was modified to 6ft. We validated this requirement by using a tape measure to record the length of the RTD used to measure the material temperature. As shown in Figures 71 and 72 it measured 81in (6'9''), which validated the requirement.

Figure 71: Test 9 – Tape-Measure measurement of material RTD lead length

Figure 72: Test 9 – Verification that RTD has more than 6ft of lead length.

Table 8 shows a summary of the testing results.

Table 8: Summary of Testing Results

Test	Description	Requirements Met
1	Bring System to 350 $^{\circ}$F	1.1, 2.1, 4.1, 4.3
2	Bring System to -100 $^{\circ}$F	1.1, 2.1, 4.2, 4.3
3	System meeting various setpoints – heat only	1.4, 2.3
4	System meeting various setpoints – heat and cool	1.4, 2.3
5	Ramp system at 2 $^{\circ}$F/min	1.4
6	Bring system to 10 and 98% RH	1.2, 2.2 *Partial Validation
7	Live results and CSV file writing	3.1, 3.2, 3.3, 3.4, 3.5, 3.6
8	Manually Stop the Program	4.4
9	RTD Lead Length Measurement	1.5

7. Engineering Economic Analysis

To complete this project, the client provided the team with a budget of $1000. Throughout the project, all purchases were tracked. The main expenses were the purchase of a single SSR and toaster oven for prototyping, and a large electronics order to integrating the circuitry at full scale. Table 9 shows all items purchased during this project, along with their respective usages and costs. Further, a pie-chart is shown in Figure 73 to compare the areas of the teams' expenses against the $1000 budget. A DigiKey quote for the full electronics order made on March 9th, 2023, is shown in Figure 74. This represents all the project expenses past the prototyping stage. It was originally planned that the third-party instruments needed for testing would need to be purchased, but the team wound up finding all these instruments either at home or at Dalhousie.

Table 9: Full breakdown of all project expenses

Item	Usage	Unit Price	Quantity	Total
Toaster Oven	Prototype	$10.00	1	$10.00
Thermocouple Amplifier	Prototpe Sensor	$20.00	1	$20.00
Solid State Relay	Prototype & Full Scale Electrical Connections	$26.91	5	$134.55
RTD Amplifier	Full Scale Temperature Readings	$21.97	3	$65.91
Header Strips	Full Scale Circuit Board	$1.83	10	$18.25
Screw Terminals	Full Scale Circuit Board	$0.87	10	$8.70
Perfboard	Full Scale Circuit Board	$19.56	1	$19.56
USB Panel Mount	Full Scale Computer Connector	$7.75	1	$7.75
			HST	$42.71
			Total	$327.43

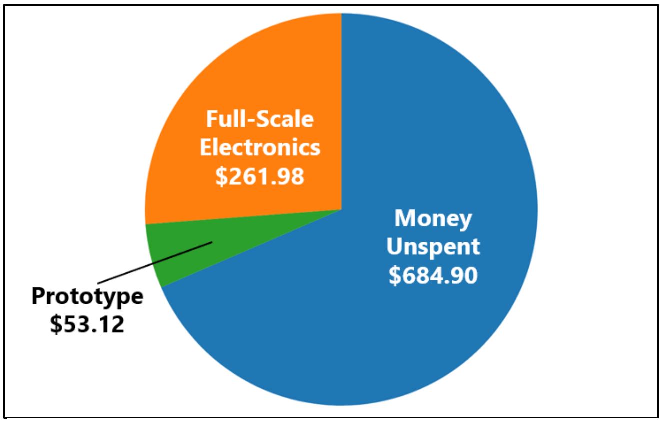
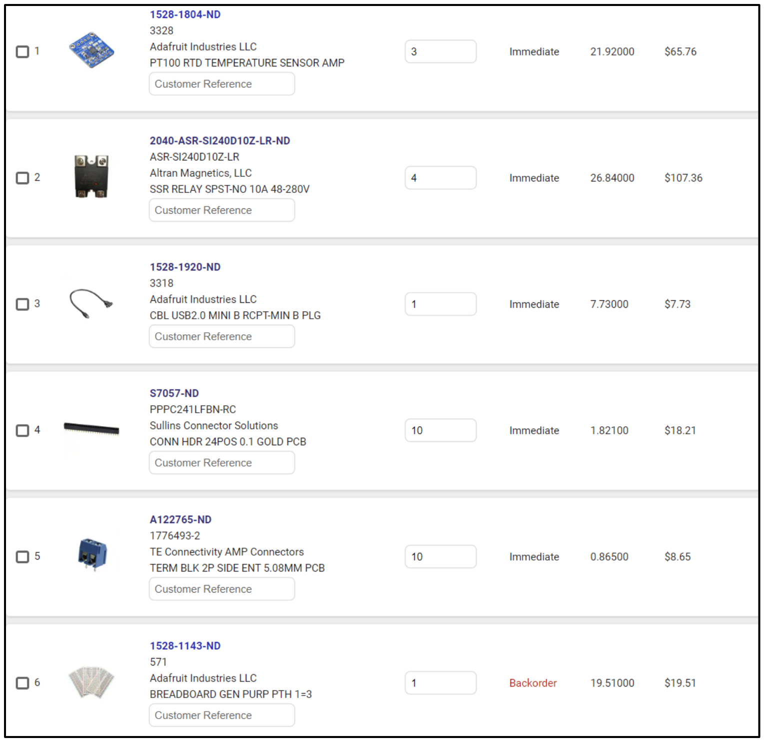


Figure 73: Budget breakdown



1	1528-1804-ND 3328 Adafruit Industries LLC PT100 RTD TEMPERATURE SENSOR AMP Customer Reference	3	Immediate	21.92000	$65.76
2	2040-ASR-SI240D10Z-LR-ND ASR-SI240D10Z-LR Altran Magnetics, LLC SSR RELAY SPST-NO 10A 48-280V Customer Reference	4	Immediate	26.84000	$107.36
3	1528-1920-ND 3318 Adafruit Industries LLC CBL USB2.0 MINI B RCPT-MIN B PLG Customer Reference	1	Immediate	7.73000	$7.73
4	S7057-ND PPPC241LFBN-RC Sullins Connector Solutions CONN HDR 24POS 0.1 GOLD PCB Customer Reference	10	Immediate	1.82100	$18.21
5	A122765-ND 1776493-2 TE Connectivity AMP Connectors TERM BLK 2P SIDE ENT 5.08MM PCB Customer Reference	10	Immediate	0.86500	$8.65
6	1528-1143-ND 571 Adafruit Industries LLC BREADBOARD GEN PURP PTH 1=3 Customer Reference	1	Backorder	19.51000	$19.51

Figure 74: DigiKey quote for full electronics order

Only spending $327.43 represented an immediate cost saving for the client, as there is no longer a need to invest in an off-the-shelf controller or a brand-new machine. In addition to these initial savings, an engineering economic analysis could be completed to determine the payback period of the initial investment and determine an estimated return on investment. To do so, a few assumptions were necessary. Currently, each experiment takes about an hour and a half of labor. This time includes cutting the cam, installing the cam, attaching/detaching thermocouples, and retrieving the data from the thermocouple. Assuming a graduate student makes roughly $10 per hour, this is $15 of labor cost per experiment. If 40 separate experiments requiring new cams for time scales are conducted a year, the operation of the current mechanical chamber costs the client $600 per year in labor. With the new digital machine, the labor costs for each use are reduced as there is no cam cutting and manual thermocouple sensing required. If a conservative estimate is made and it is assumed that 30 minutes of labor will now be spent on each experiment, the operational cost of the chamber will reduce to $200 per year. Therefore, the implementation of a digital system will be saving the client $400 per year. Under these assumptions, the payback period on the initial $1000 investment is about 43 weeks. This is calculated by dividing the $327.43 investment by the weekly savings ($400/52). An additional assumption is made that every 5 years the client will need to reinvest another $200. This money will go towards upgrades to the controller, the relays, and other electronic equipment, along with maintenance and labor costs.

With these yearly savings and outflows, a cashflow diagram covering the next 10 years can be constructed. With said cashflows, Equation 2 is used to find the net present value (NPV) [18]:

$$NPV = \frac{C}{(1+r)^n} \qquad (2)$$

.

C represents the cashflow in year n, while r is the discount rate. Setting the discount rate to the reported December 2022 Nova Scotia inflation rate of 7%, the NPV of yearly cashflows are summarized below.

Table 10: Engineering economic analysis of net present value and return on investment

Year	Discount Rate	Inflow	Outflow	Inflow PV	Outflow PV
1	7.00%	$400.00	$327.43	$373.83	$306.01
2	7.00%	$400.00	$0.00	$349.38	$0.00
3	7.00%	$400.00	$0.00	$326.52	$0.00
4	7.00%	$400.00	$0.00	$305.16	$0.00
5	7.00%	$400.00	$200.00	$285.19	$142.60
6	7.00%	$400.00	$0.00	$266.54	$0.00
7	7.00%	$400.00	$0.00	$249.10	$0.00
8	7.00%	$400.00	$0.00	$232.80	$0.00
9	7.00%	$400.00	$0.00	$217.57	$0.00
10	7.00%	$400.00	$200.00	$203.34	$101.67
			Cumulative PV	$2,809.43	$550.28
			Net PV	$2,259.16	
			ROI	410.55%	

The total NPV of the digital controller implementation is found by subtracting the total NPV of the outflowing cash from the total NPV of the inflow [18]. We find an NPV of $2259.18, which is a 410.55% return on investment (ROI) from the total outflow of $550.28. In addition to labor cost savings, the digital controller may also increase the lab's output by enabling more tests to be conducted in the same amount of time as the mechanical machine. This could potentially result in increased sponsorship money or other benefits. To conclude this section, it is estimated that the client made a wise financial decision by using the Capstone program to design and implement a new digital controller for his environmental chamber.

8. Future Considerations

With the design now integrated at full scale, consideration is now given to how the system may be improved going forward. While the original scope of converting the system from mechanical to digital control has been met, there are still several enhancements that the client may wish to consider in the future. First, the addition of a pre-set setpoint memory feature could be of interest. As it currently stands, process setpoints and times are manually entered to the UI. If the same tests are conducted repeatably, it would be beneficial to have these setpoints saved to the software memory allowing the user to repeat the run with the press of a single button. Additionally, the addition of a sinusoidal input option should be investigated. If the desired test is of a cyclic nature and requires the continuous oscillation between two setpoints, entering the same two setpoints and times to the UI could become tedious and time consuming. Therefore, a feature that allows the user to customize a sinusoidal profile of desired amplitude and period could be desirable. It may also be beneficial for the user to be able to customize the transient steady state characteristics of the temperature response directly from the user interface. This would only involve tweaking the controller program to have the PID constants be $K_p,\ K_I,$ and K_D be variable inputs rather then constants hard coded into the software.

In addition to software enhancements, the installation and testing process has left the team with several recommendations regarding the overall performance of the existing system. First, it is recommended that the compressor and liquid nitrogen cooling systems be inspected. While the system was cooled to the necessary temperature of -100^0F, the cooling at low temperatures was very slow, and we could find no external confirmation that the liquid nitrogen system was working adequately. Moreover, the compressor takes a long time to start up and judging by the noises it emits, it seems to stall a few times before it begins to run. A thorough inspection of these components is recommended, specifically to verify whether they have adequate refrigerant supply. Another hinderance on performance discovered throughout the testing process was inadequate insulation. In addition to the two port holes on both sides of the chamber that need to manually be insulated with foam, standing next to the door when the machine is operation leads one to believe the also energy escaping through the seals. This lack of insulation results in higher energy requirements to bring the machine to extreme temperatures, and it also makes the system more prone to external noise and disturbances. Finally, as was mentioned in the testing and validation section, there seems to be no water supply going to the wet bulb RTD. For the accurate control of the chamber humidity, this should be addressed immediately, as manually wetting the sensor is not a sustainable solution.

9. Hand-Off Plan

With the final design now completed and installed, the team has prepared a hand-off plan for the client. The client will be able to use it as reference for any troubles they may come across as they integrate it into their daily operation. The hand-off is broken down into three main categories, each of which are summarized as follows.

1. **The Product:** The product itself will need to be passed off to the client so they can begin to use it. In this case, the product mainly consists of the software required to control the chamber. The team has packaged this software into an executable file that the user can run on their computer to bring up the controller interface and begin controlling the chamber. The client has been provided this file via email and need only to download it to their computer to run the software.
2. **User Manual:** To effectively use the controller interface, the client has also been provided a user manual (attached in Appendix A). The user manual first details the complete downloading and installing instructions of the software. This includes how to open and run the executable on any operating system. It then illustrates the basics of how to enter desired temperature and time setpoints into the interface. Additionally, it shows how to retrieve the temperature vs. time data from the local computer after the test is complete. The user manual also lists best practices for ensuring safety such as keeping the closed during operation, unplugging the Arduino should an undesirable result occur, and highlighting the upper and lower bounds on temperature setpoints.
3. **Project Files:** The team will also be providing the client with all the project files that have been accumulated over this process. To do so, each member of the team will be uploading their personal “Mech4015” folder to a OneDrive folder, and this cumulative folder will be shared with the client. It is expected that any file used or produced during both terms can be found in this folder. These files include, but aren’t limited to PDF

logbooks, reports, timesheets, meeting minutes, engineering drawings, SolidWorks files, figures, excel sheets for calculations and budget, purchasing quotes, emails, and of course the scripts used for all the code produced in this project.

10. Conclusion

This report detailed the complete design process for the design of a digital environmental chamber controller. Once the requirements, stakeholders, and budget for this project were outlined, a functional decomposition was completed to identify the necessary subsystems to take the users inputted information and turn it in to the required output of digital process value control. The identified subsystems were the user input interface, the controller hardware and software, the electromechanical devices to modify temperature and humidity, sensors to measure these values, and the feedback system to relay these readings back to the controller and emit warnings should the measurements fall outside the required range. Multiple design solutions for each subsystem were proposed and detailed, and several of the design ideas were used to formulate an LED brightness controller and a toaster oven prototype. With the help of what was learned in prototyping, the three alternative designs were rated on usability, functionality, and feasibility. The first alternative design was chosen, and an installation plan was formulated. The report then detailed the key iterations made on this initial design in the winter of 2023, and described how a final product was developed and installed to the full-scale chamber. The testing procedure was then outlined, and it was concluded that all requirements were met, with one caveat being that the humidity range requirement was only partially validated. A review of the project expenses is then detailed in the form of an engineering economic analysis, and this showed that the 10-year-ROI for this project is 410% and the payback period is 10 months. Several future considerations were highlighted including the addition of more customized inputs, and the recommendation that all machine components be inspected do to aging, improper insulation, material degradation, and insufficient water and refrigerant supply. Finally, a plan was outlined to hand off this completed project to the client, and this included a transfer for all project files, a downloadable copy of the software, and a user manual.

References

[1] J. E. Grayson, Python and Tkinter Programming, Greenwich: Manning Publications Company, 2000.

[2] R. J. Smythe, Arduino and Raspberry Pi. In Advanced Arduino Techniques in Science, Berkeley: Springer, 2021.

[3] D. E. Bolanakis, Microcontroller Prototypes with Arduino and a 3D Printer: Learn, Program, Manufacture., : John Wiley & Sons. , 2021.

[4] S. F. Barrett, "Arduino microcontroller processing for everyone!," *Synthesis Lectures on Digital Circuits and Systems,* vol. 8, no. 4, pp. 1-513, 2013.

[5] R. Tang, "Arduino Vs Raspberry Pi - Which One is Better for Robotics Projects?," *Product Design & Development,* 2017.

[6] Ulpiani, G., Borgognoni, M., Romagnoli, A., & Di Perna, C. (2016). Comparing the performance of on/off, PID and fuzzy controllers applied to the heating system of an energy-efficient building. *Energy and Buildings,* 116, 1–17.

[7] https://doi.org/10.1016/j.enbuild.2015.12.027
C. Williams, "Feedback and Temperature Control," [Online]. Available: https://courses.cs.washington.edu/courses/cse466/01au/Lecture/PIDoverview.html. [Accessed 25 October 2022].

[8] K. Ulrich and S. Eppinger, Product Design and Development, McGraw-Hill, 2012.

[9] K. Astrom, Control System Design Chapter 6: PID Control, CalTECH, 2002.

[10] R. Singh, B. Singh. and A. Gehlot, Arduino and Scilab Based Projects., Bentham Science Publishers. , 2019.

[11] M. Moran, H. Shapiro, D. Boettner and M. Bailey, Fundamentals of Engineering Thermodynamics, Wiley & Sons, 2011.

[12] T. A. Howell, "Thermocouple Reference Tables," Oak Ridge National Laboratory, Report ORNL/TM-12907, May 1995. [Online]. Available: https://doi.org/10.2172/117422. [Accessed: April 03, 2023].

[13] N. Hu, Y. H. Park, and J. Lee, "Development of a miniature humidity sensor for monitoring trace moisture in dry air," IEEE Sensors Journal, vol. 14, no. 7, pp. 2272-2279, Jul. 2014, doi: 10.1109/JSEN.2014.2310152.

[14] Allocca, P. (2011). Beginning Arduino Programming (Chapter 5). Apress.

[15] Adafruit Industries. (n.d.). Adafruit MAX31865 RTD PT100 amplifier. [Online]. Available: https://learn.adafruit.com/adafruit-max31865-rtd-pt100-amplifier/arduino-code. [Accessed: Apr. 02, 2023].

[16] Bergman, T. L., Lavine, A. S., Incropera, F. P., & Dewitt, D. P. (2011). Fundamentals of Heat and Mass Transfer. John Wiley & Sons

[17] Seto, M (2023). 4-State Estimation. Dalhousie University - Department of Mechanical Engineering. Halifax NS, Canada.

[18] T. E. Copeland, J. F. Weston, and K. Shastri, Financial Theory and Corporate Policy, 4th ed. Boston, MA: Pearson, 2005, ch. 5, sec. 5.1.

[19] "Analysis of Nova Scotia's Consumer Price Index for December & Annual 2022," Nova Scotia Finance and Treasury Board, Jan. 17, 2023. [Online]. Available: https://novascotia.ca/finance/statistics/topic_news.asp?id=18508&fto=21u&rdval=2023-01#:~:text=gov.ns.ca%20%2D%20Nova,Canada%20%2D%20Government%20of%20Nova%20Scotia&text=Nova%20Scotia's%20All%2DItems%20Consumer,at%209.3%25%20in%20June%20202 . [Accessed: Apr. 02, 2023].

Appendix A Stakeholder List

Click here to return to map

Stakeholder	Type	Relevance	Stakeholder	Type	Relevance
Dr. Fared Taheri	Client	Main client who brought project forward. Is very interested in project result as his research will incorporate the use of the chamber.	**Peter Jones**	Department Engineer	Responsible for performance of technicians and is therefore impacted by outcome of project.
Ke Wang	Client	Graduate student of the main client who wishes to use the environmental chamber. Very interested in project result as it impacts his research.			
Jesse Keane	Technician	Technician responsible for laboratory in which chamber is held. Affected by result as he will need to review any changes made in this laboratory.			
Jon Macdonald	Technician	Electronics technician for mechanical students. Interested in result as he is providing electrical equipment and components. Also offering electronics expertise.			
Brian Kennedy	Technician	Machine technician for mechanical students. Will be of use should a prototype chamber be constructed.			
Albert Murphy	Technician	Head of Mechanical Engineering machine shop and is therefore directly connected to the prototype construction for this project.			

Appendix B:Source Code

Toaster Oven Prototype: Python Code

```
## Toaster Oven Controller Interface
## Author: Reilly Pickard
## Last Update: January 29th, 2023
## Description: interface to control
## the temperature of a toaster oven
# by sending input temp through serial port
# to an Arduino connected to oven

import serial
import time
import tkinter
from tkinter import *
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import pandas as pd
import numpy as np
import csv

df1 = pd.DataFrame([[ 1]])

def set_button1_state():
        global b,z
        b = entry.get()
        ser.write(b.encode())
        print("")
        z=1
        print(z)

ser = serial.Serial('com4', 9600)

tkTop = tkinter.Tk()
tkTop.geometry('800x800')
tkTop.title("Team03")
label3 = tkinter.Label(text = 'Environmental Chamber Controller'
                      ,font=("Segoe UI", 20,'bold')).pack()


label = tkinter.Label(text = "Enter Desired Temperature (degF)", font=("Segoe
UI", 20,))

entry = tkinter.Entry(tkTop,

    fg = "black",

    bd = 5
)


button1state = tkinter.Button(tkTop,
    text="Confirm",
```

```
                              font=("Segoe UI", 20,),
    command=set_button1_state,
    height = 8,
    fg = "black",
    width = 16,
    bd = 5,
    activebackground='green'
)

label.pack(pady = 40)
entry.pack(pady = 10)

button1state.pack(side='top', ipadx=10, padx=10, pady=15)

fig=plt.figure()

ax = fig.add_subplot(1,1,1)
ax.set_xlim([20,40])
ax.set_ylim([0,6])
ax.plot([i for i in range(10)],[i for i in range(10)])
global z
z=0
data_string=""
xs=[i for i in range(10)]
ys = []
data=0

def animate(i):
    if(z==1):
        data_string=ser.readline()
        data=float(ser.readline())
        ys.append(data)
        df = pd.DataFrame(ys)
        print(data)
        ax.clear()
        ax.set_xlim([float(len(ys)-60),float(len(ys))])
        ax.set_ylim([0,300])
        ax.plot([i for i in range(len(ys))],ys, 'r')
        ax.set(xlabel='Time (S)',ylabel='T (degF)')
        ax.set_title('Thermocouple Reading')
        ax.grid()
        df.to_csv('file.2csv')



    else:
        print("")




anim = animation.FuncAnimation(fig, animate, interval=10)

plt.show()

tkTop.mainloop()
```

Toaster Oven Prototype: Arduino Code

```
#include <PID_v1.h>
const int thermocouple_input = A2;
double Setpoint, Input, Output;
double Kp=0.1, Ki=10, Kd=0.01;
float S1;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup() {
  Serial.begin(9600);
  myPID.SetMode(AUTOMATIC);
  myPID.SetTunings(Kp, Ki, Kd);
}

void loop() {
float temperatureC;
temperatureC = ( 5.0*analogRead(thermocouple_input)*100.0) /
1024.0;
float temperature;
temperature = 1.8*temperatureC +32.0;
while(Serial.available()){

   S1 = Serial.parseFloat();
}

Setpoint = S1;
Input = temperature;
myPID.Compute();
analogWrite(13, Output);
Serial.print(temperature);
Serial.print("\n");
delay(500);

}
```

Final Python Code:

```
Environmental Chamber GUI
# Author: Reilly Pickard (contact: reilly.pickard@dal.ca)
# Last Update: April 5th, 2023
# Description: This GUI was created to control the
# environmental testing chamber located in D115A at
# Dalhousie University, Halifax, NS
# This was done to meet the requirements of the Team 3's 2023
# Mechanical Engineering Capstone Project.
# It has two main parts: A setpoint input table that accepts multiple
```

```
# temperature, humidity, and time-scale values. It also plots the live
results
# for air and material temperature, along with the humidity readings.

### Imports
import serial
import time
import tkinter as tk
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import pandas as pd
import csv
import smtplib
import ssl
import serial.tools.list_ports

### Initialize e-mail messages
port = 587
smtp_server = "smtp-mail.outlook.com"
sender = "chamberalerts@outlook.com"
recipient = "reilly.pickard@dal.ca"  ### Change this to user email
sender_password = "Pass2208*" ## Please do not share
message1 = """
Subject: CHAMBER ALERT.
The chamber has exceeded the maximum temperature of 350F. Please go check to
ensure the chamber has powered off. It is located in D115A. Turn off by
removing
the USB cable or by pulling the lever down on the power source. """
message2 = """
Subject: CHAMBER ALERT.
The chamber has gone below the minimum temperature of -100F. Please go check
to
ensure the chamber has powered off. It is located in D115A. Turn off by
removing
the USB cable or by pulling the lever down on the power source. """
SSL_context = ssl.create_default_context()


#### Get a list of available ports
available_ports = list(serial.tools.list_ports.comports())
##### Intialize Serial Communication
if not available_ports:
    print("No ports available. Please connect your device.")
else:
    # Iterate through available ports and look for Arduino
    for port in available_ports:
        if "Arduino" in port.description:
            # If Arduino is found, connect to it
            ser = serial.Serial(port.device, 115200)
            print("Connected to", port.description, "on", port.device)
            break
    else:
        print("Could not find an Arduino.")

#### Define a function to send data to Arduino
def send_to_arduino():
    temperatures = [temperature_entries[i].get() for i in
```

```
range(len(temperature_entries))]
    humidities = [humidity_entries[i].get() for i in
range(len(humidity_entries))]
    hold_times = [hold_time_entries[i].get() for i in
range(len(hold_time_entries))]
    temperatures = [float(temperature) for temperature in temperatures if
temperature]
    humidities = [float(humidity) for humidity in humidities if humidity]
    hold_times = [float(hold_time) for hold_time in hold_times if hold_time]
    data = []
    for i in range(len(temperatures)): ## send appended inputs with delimeter
','
        data.append(str(temperatures[i]) + ',' + str(humidities[i]) + ',' +
str(hold_times[i]))
    ser.write('\n'.join(data).encode())

####Create the GUI
root = tk.Tk()
root.title("Chamber Controller")
root.geometry('475x375')
root.configure(bg='black')

##table headers
header_label = tk.Label(root, text="Chamber Controller", font=("Arial", 18,
"bold"), bg="black", fg="white", pady=20, padx = 30)
header_label.grid(row=0, column=0, columnspan=3, sticky="nsew", padx  = 20)

temperature_label = tk.Label(root, text="Temperatures (°F)", bg = "black", fg
= "white", font = ("Arial", 12, "bold"))
temperature_label.grid(row=1, column=0, padx=(12, 5))

humidity_label = tk.Label(root, text="Humidity (%)", bg = "black", fg =
"white", font = ("Arial", 12, "bold"))
humidity_label.grid(row=1, column=1, padx = (5,5))

hold_time_label = tk.Label(root, text="Hold Times (min)", bg = "black", fg =
"white", font = ("Arial", 12, "bold"))
hold_time_label.grid(row=1, column=2, padx=(5, 12))

##initialize storage arrays
temperature_entries = []
humidity_entries = []
hold_time_entries = []

### Create a Table and append entries
for i in range(10):## change from 10 to however amount of entries you want
    temperature_entry = tk.Entry(root, font = ("Arial", 10, "bold"))
    temperature_entry.grid(row=i+2, column=0, padx=(12, 5))
    temperature_entries.append(temperature_entry)

    humidity_entry = tk.Entry(root, font = ("Arial", 10, "bold"))
    humidity_entry.grid(row=i+2, column=1, padx = (5,5))
    humidity_entries.append(humidity_entry)

    hold_time_entry = tk.Entry(root, font = ("Arial", 10, "bold"))
    hold_time_entry.grid(row=i+2, column=2, padx=(5, 12))
    hold_time_entries.append(hold_time_entry)
```

```
## Send button and message about user manual
send_button = tk.Button(root, text="Send", command=send_to_arduino, bg =
"white", fg = "black", font = ("Arial", 12, "bold"))
send_button.grid(row=12, column=0, columnspan=3, pady = (10,10))

header_label2 = tk.Label(root, text="Refer to User Manual for Instructions",
font=("Arial", 12, "bold", "italic"), bg="black", fg="white")
header_label2.grid(row=13, column=0, columnspan=3, sticky="nsew")


##### Live Plot #######
fig = plt.figure()

# Add temperature subplot
ax_temp = fig.add_subplot(1,1,1)
ax_temp.set_xlim([20, 40])
ax_temp.set_ylim([0, 400])

ax_temp.set_title('Temperature Reading')
ax_temp.grid()

ys_temp = [] ## Init arrays
ys_tempm = []
data_temp = 0

#Add humidity subplot
ax_humidity = fig.add_subplot(2,1,2)
ax_humidity.set_xlim([20, 40])
ax_humidity.set_ylim([0, 100])
ax_humidity.set(xlabel='Time (S)', ylabel='Relative Humidity (%)')
ax_humidity.set_title('Humidity Reading')
ax_humidity.grid()
ys_humidity = []
data_humidity = 0

### Animate the plot
def animate(i):
    global data_temp, data_humidity
    data = ser.readline().decode("ascii") ## Receive the data
    data = str(data)
    data = data.split(',')  # split the temperature and humidity values using
a comma separator
    data_temp = round(float(data[0]),2)
    data_tempm = round(float(data[1]),2)
    data_humidity = (float(data[2]))
    ys_temp.append(data_temp)
    ys_tempm.append(data_tempm)
    ys_humidity.append(data_humidity)
    df = pd.DataFrame({'Air Temp. (F)': ys_temp, 'Material Temp.(F)':
ys_tempm, 'Humidity (%)': ys_humidity})
    print(f"Temperature: {data_temp} F, Verification RTD: {data_tempm} F,
Rel. Humidity: {data_humidity}%")
    ## temperature plot
    ax_temp.clear()
    ax_temp.set_xlim([len(ys_temp)-60, len(ys_temp)]) ## this limit is set
for showing 60 data points in a graph
```

```
    ax_temp.plot([i for i in range(len(ys_temp))], ys_temp, 'r')  ## plots
the temperature graph
    ax_temp.plot([i for i in range(len(ys_tempm))], ys_tempm, 'b', label =
"Material")
    ax_temp.set_title('Chamber Temperature Readings')
    ax_temp.grid()
    ax_temp.legend()
    ax_temp.set(xlabel='Time (S)', ylabel='Temperature (F)')
    filename = f'ChamberData{current_date}.csv' ## append the current date to
the file name
    df.to_csv(filename)
    ## Humidity plot
    ax_humidity.clear()
    ax_humidity.set_xlim([len(ys_humidity)-60, len(ys_humidity)]) ## this
limit is set for showing 60 data points in a graph
    ax_humidity.plot([i for i in range(len(ys_humidity))], ys_humidity,
'black')  ## plots the humidity graph
    ax_humidity.set_title('Humidity Reading')
    ax_humidity.set(xlabel='Time (S)', ylabel='Humidity (%)')
    ax_humidity.grid()

    plt.subplots_adjust(hspace=0.5)  # Add space between subplots

    ### Send email messages
    if(data_tempm> 400):
            with smtplib.SMTP(smtp_server, port) as server:
                 server.starttls(context=SSL_context)
                 server.login(sender, sender_password)
                 server.sendmail(sender, recipient,  message1)
    if(data_tempm< -150):
            with smtplib.SMTP(smtp_server, port) as server:
                 server.starttls(context=SSL_context)
                 server.login(sender, sender_password)
                 server.sendmail(sender, recipient,  message2)

anim = animation.FuncAnimation(fig, animate, interval=10)


plt.show()

root.mainloop()
```

Final Arduino Code:

```
/* Environmental Chamber Controller - PID Execution
Author: Reilly Pickard (contact: reilly.pickard@dal.ca)
Last Update: April 5th, 2023
Description: This code accepts the list of inputted temperatures,
humidities, and time scales from a Python GUI to execute PID control
for an environmental chamber. Feedback is generated by RTDs amplified by Adafruit
MAX31865
are used to read temperature off  Pt200 RTDs
*/
```

```
// Includes
#include <PID_v1.h>
#include <Adafruit_MAX31865.h>
#include <Adafruit_SPIDevice.h>

// initialize wet and dry RTD's
#define RREF 430
Adafruit_MAX31865 dry = Adafruit_MAX31865(10, 11, 12, 13);
Adafruit_MAX31865 wet = Adafruit_MAX31865(9, 11, 12,13);
// Initialize global variables
double temp_setpoint, temp_input, temp_output;
double hum_setpoint, hum_input, hum_output;
//PID Gains. Tune these for desired response.
double Kp=10, Ki=0.017, Kd=0.11;
PID tempPID(&temp_input, &temp_output, &temp_setpoint, Kp, Ki, Kd, DIRECT);
PID humPID(&hum_input, &hum_output, &hum_setpoint, Kp, Ki, Kd, DIRECT);

// Kalman filter variables
double x = 65; // initial state
double P = 1; // initial covariance
double Q = 0.00001; // process noise covariance
double R = 0.1; // measurement noise covariance
double K = 0; // Kalman gain

// Set up Code
void setup() {
Serial.begin(115200);
// Initialize the PIDs
tempPID.SetMode(AUTOMATIC);
tempPID.SetSampleTime(100);
humPID.SetMode(AUTOMATIC);
humPID.SetSampleTime(100);
// begin RTDS
dry.begin(MAX31865_3WIRE);
wet.begin(MAX31865_3WIRE);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
}

/// Function for reading RTD's
float read_temperature() {
// Read raw temperature value
float T = wet.temperature(200, RREF);
float tw = dry.temperature(200, RREF) ;
```

```
// Apply Kalman filter
K = P / (P + R);
x = x + K * (T - x);
P = (1 - K) * P + Q;

// Convert to Fahrenheit and return filtered temperature value
float temperatureF = 1.8 * x + 32.0;
float twF = 1.8*tw + 32.0;
// Compute humidity
float Ed =  6.112 * exp((17.67 * T)/(T + 243.5));
float Ew = 6.112 * exp((17.67 * tw)/(tw + 243.5));
float RH = 100*(Ew - 0.6687*(1+0.0015*tw)*(T-tw))/Ed;

//Print values back to Python to plot
Serial.print(String(temperatureF));
Serial.print(",");
Serial.print(String(temperatureF)); // Change to material if desired
Serial.print(",");
Serial.println(String(RH));
delay(500);
// Apply initial fail_safe
if(temperatureF > 350.0){
digitalWrite(3,LOW);
} //
return temperatureF;
}
//Loop
void loop() {
  if (Serial.available() > 0) { // Check for new inputs

    String line = Serial.readStringUntil('\n'); // read the inputted string
    // Use Delimeter and Subsets to find temp, humidity, and times from input
    String setpoints_hold_times = line.substring(0, line.length() - 1);
    String setpoints_hold_times_array[100];
    int i = 0;
    int j = 0;

    for (i = 0; i < setpoints_hold_times.length(); i++) {
      if (setpoints_hold_times[i] == '\n') {
        j++;
        i++;
      }
      setpoints_hold_times_array[j] += setpoints_hold_times[i];
    }
    int length = j + 1;
```

```
// Use Delimeter and Subsets to find temp, humidity, and times from input
    for (i = 0; i < length; i++) {
      // Update setpoints
      String setpoint_hold_time_humidity = setpoints_hold_times_array[i];
      int comma1 = setpoint_hold_time_humidity.indexOf(',');
      int comma2 = setpoint_hold_time_humidity.indexOf(',', comma1 + 1);
      temp_setpoint = setpoint_hold_time_humidity.substring(0,
comma1).toDouble();
      hum_setpoint = setpoint_hold_time_humidity.substring(comma1 + 1,
comma2).toDouble();
      float hold_time = setpoint_hold_time_humidity.substring(comma2 +
1).toInt();
      hold_time = hold_time*60;
      // Now read temperature
      temp_input = read_temperature();
      //Serial.println(String(hold_time*1000.0));


      if (temp_setpoint < 0) {
        // Low temp Control cryogenic cooling valve
        unsigned long start_time = millis(); // start timer
        while ((millis() - start_time) < hold_time * 1000) {
          temp_input = read_temperature();
          //hum_input = read_humidity(); Wet bulb not working
          tempPID.Compute();
          humPID.Compute();
          digitalWrite(5,LOW);
          digitalWrite(4,temp_output>0 ? LOW:HIGH);
          digitalWrite(3, LOW);
          //digitalWrite(6, hum_output > 0 ? LOW : HIGH); // wet bulb not working
          //digitalWrite(5, hum_output < 0 ? LOW : HIGH); // wet bulb not working

        }
      } else if (temp_setpoint < temp_input) {

        // Control refrigeration compressor
        unsigned long start_time = millis();
        while ((millis() - start_time) < hold_time * 1000) {

          temp_input = read_temperature();
          hum_input = read_humidity();// Wet bulb not working
          tempPID.Compute();
          humPID.Compute();
          digitalWrite(5, LOW);
          digitalWrite(4,temp_output>0 ? LOW:HIGH);
```

```
      if(temp_input < temp_setpoint){ // if we overshoo, need to heat up
      digitalWrite(3, HIGH);
      }else{
       digitalWrite(3, LOW);
      }
      //digitalWrite(6, hum_output > 0 ? LOW : HIGH);
      //digitalWrite(5, hum_output < 0 ? LOW : HIGH);

    }
  }else{ // Heating required
    unsigned long start_time = millis();
    unsigned long end_time = start_time;
    while (millis() - start_time < hold_time * 1000) {
      temp_input = read_temperature();
            if(temp_input > 350.0){ // FAILSAFE
    digitalWrite(3,LOW);
      }else{

      //hum_input = read_humidity();
      tempPID.Compute();

      humPID.Compute();
      digitalWrite(5, LOW);
      if(temp_input >(temp_setpoint+3)){ // Turn on compressor for overshoot
        digitalWrite(4, HIGH);
      }else{
      digitalWrite(4, LOW);
      }
      analogWrite(3, temp_output  ); // Send PID signal to heater
     // digitalWrite(6, hum_output > 0 ? LOW : HIGH);
      //digitalWrite(5, hum_output < 0 ? LOW : HIGH);
      delay(500);
      end_time = millis();
      }

    }

  }
}
float temp = read_temperature();
// Turn stuff off if idle
  digitalWrite(5, LOW);//
  digitalWrite(4, LOW);//
  digitalWrite(3, LOW);//
```

```
  }
  // Turn stuff off if idle
  float temp = read_temperature();
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
  digitalWrite(3, LOW);
  }
```

MATLAB Simulation Code:

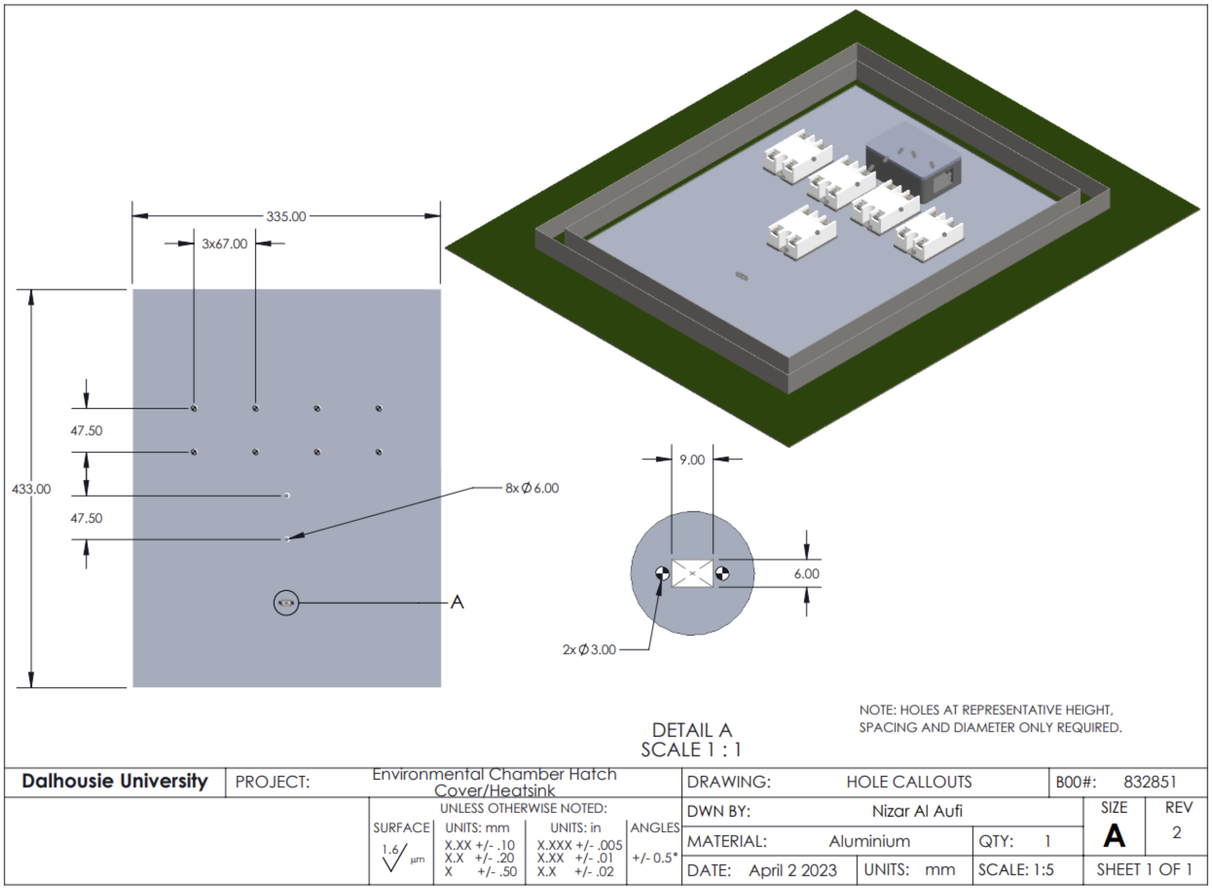
```
% Define the environmental chamber parameters
chamber_volume = 27; % ft^3
chamber_height = 3; % ft
chamber_width = 3; % ft
chamber_length = chamber_volume / (chamber_height * chamber_width); % ft
chamber_surface_area = 2 * (chamber_height * chamber_length + chamber_height *
chamber_width + chamber_length * chamber_width); % ft^2


% Set the initial conditions of the chamber
initial_temperature = 85; % Fahrenheit
temperature_noise_stddev = 0.06;
heater_disturbance_stddev = 0.06;

% Define the inputs to the chamber
heater_power = 2000; % BTU/hour
cooler_power = 330; % BTU/hour

% Define the temperature and humidity setpoints
temperature_setpoint = 400; % Fahrenheit

% Define the controller gains
Kp = 0.8;
Ki = 0.0047;
Kd = 0.8;

% Define the controller sample time
sample_time = 0.5; % second

% Define the simulation time
sim_time = 2700; % seconds

% Initialize the simulation
time = 0;
temperature = initial_temperature;

temperature_error = temperature_setpoint - temperature;

temperature_integral = 0;

temperature_derivative = 0;

temperature_change_history = []; % Initialize the temperature change history
```

```
heater = [];
cooler = [];
temperature_error_previous = 0;

% Run the simulation
while time <= sim_time
    % Compute the controller output
    temperature_output = Kp * temperature_error + Ki * temperature_integral + Kd *
temperature_derivative;
    % Update the inputs to the chamber
    heater_power_input = max(0, temperature_output) * heater_power + normrnd(0,
heater_disturbance_stddev);
    cooler_power_input = max(0, -temperature_output) * cooler_power;

    % Compute the chamber heat transfer coefficient
    if temperature > temperature_setpoint
        heat_transfer_coefficient = 350.252; % BTU/hour-ft^2-Fahrenheit
    else
        heat_transfer_coefficient =350.252; % BTU/hour-ft^2-Fahrenheit
    end

    % Compute the chamber heat transfer
    heat_transfer = heat_transfer_coefficient * chamber_surface_area * (temperature -
70);

    % Compute the chamber energy balance
    chamber_energy_balance = heater_power_input - cooler_power_input - heat_transfer;

    % Compute the chamber temperature change
    temperature_change = chamber_energy_balance / (chamber_volume * 62.42796*60); %
Fahrenheit/minute

    % Update the chamber temperature
    temperature = temperature + temperature_change * sample_time + normrnd(0,
temperature_noise_stddev);

    % Update the controller error terms
    temperature_error = temperature_setpoint - temperature;

    temperature_integral = temperature_integral + temperature_error * sample_time;

    temperature_derivative = (temperature_error - temperature_error_previous) /
sample_time;

    % Update the time and error variables
    time = time + sample_time;
    temperature_error_previous = temperature_error;
    heater(end+1) = heater_power_input;
    cooler(end+1) = cooler_power;
    temperature_change_history(end+1) = temperature_change;
end
% Plot the temperature vs time
figure;
t = 0:sample_time:sim_time;
```

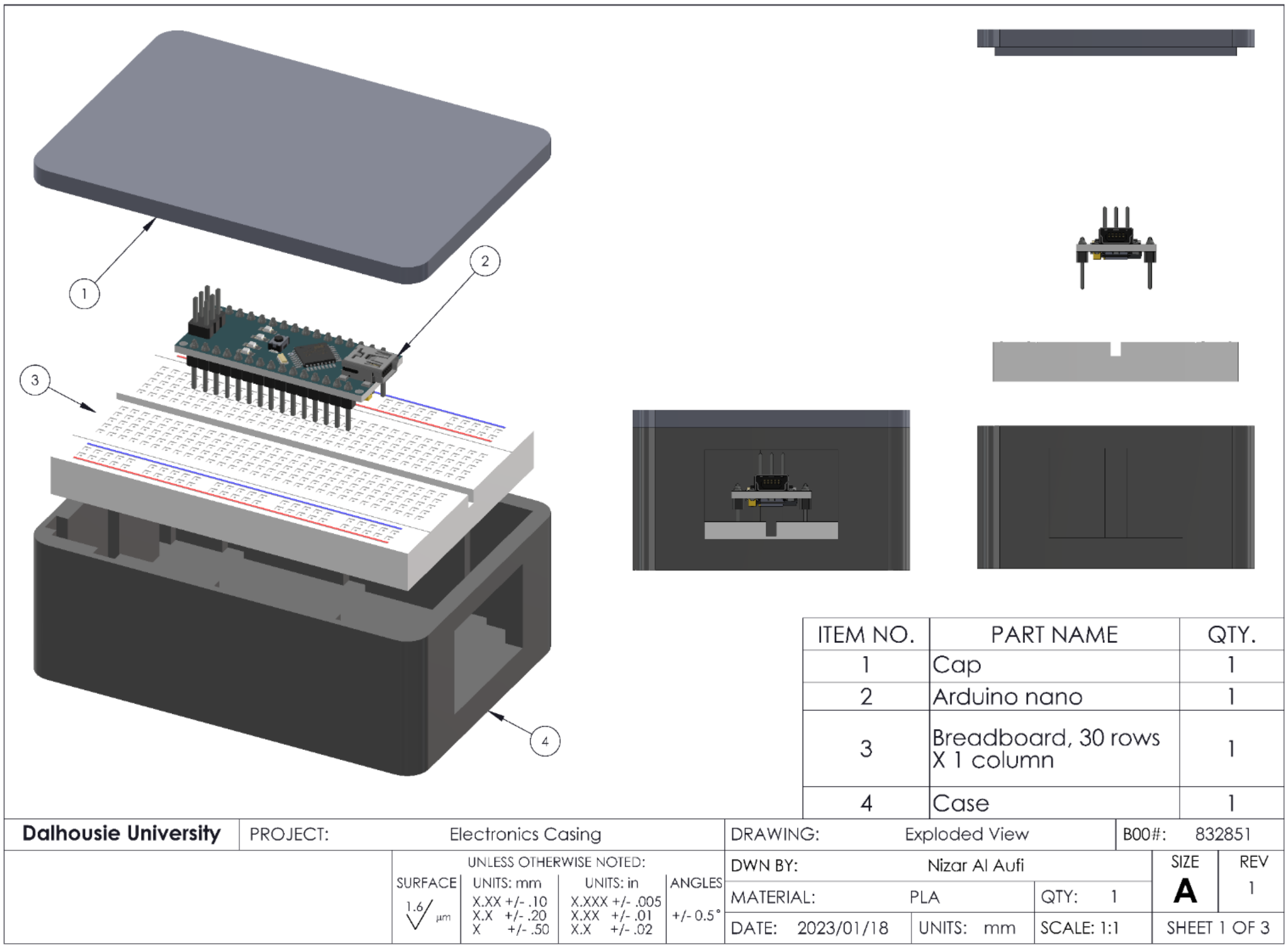
```
plot(t/60, initial_temperature*ones(1,length(0:sample_time:sim_time)) +
cumsum(temperature_change_history)*sample_time, ...
    LineWidth = 2.5, color = 'k');
xlabel('Time (min)');
ylabel('Temperature (F)');
title('Simulate Chamber Temperature vs Time');
grid on
```

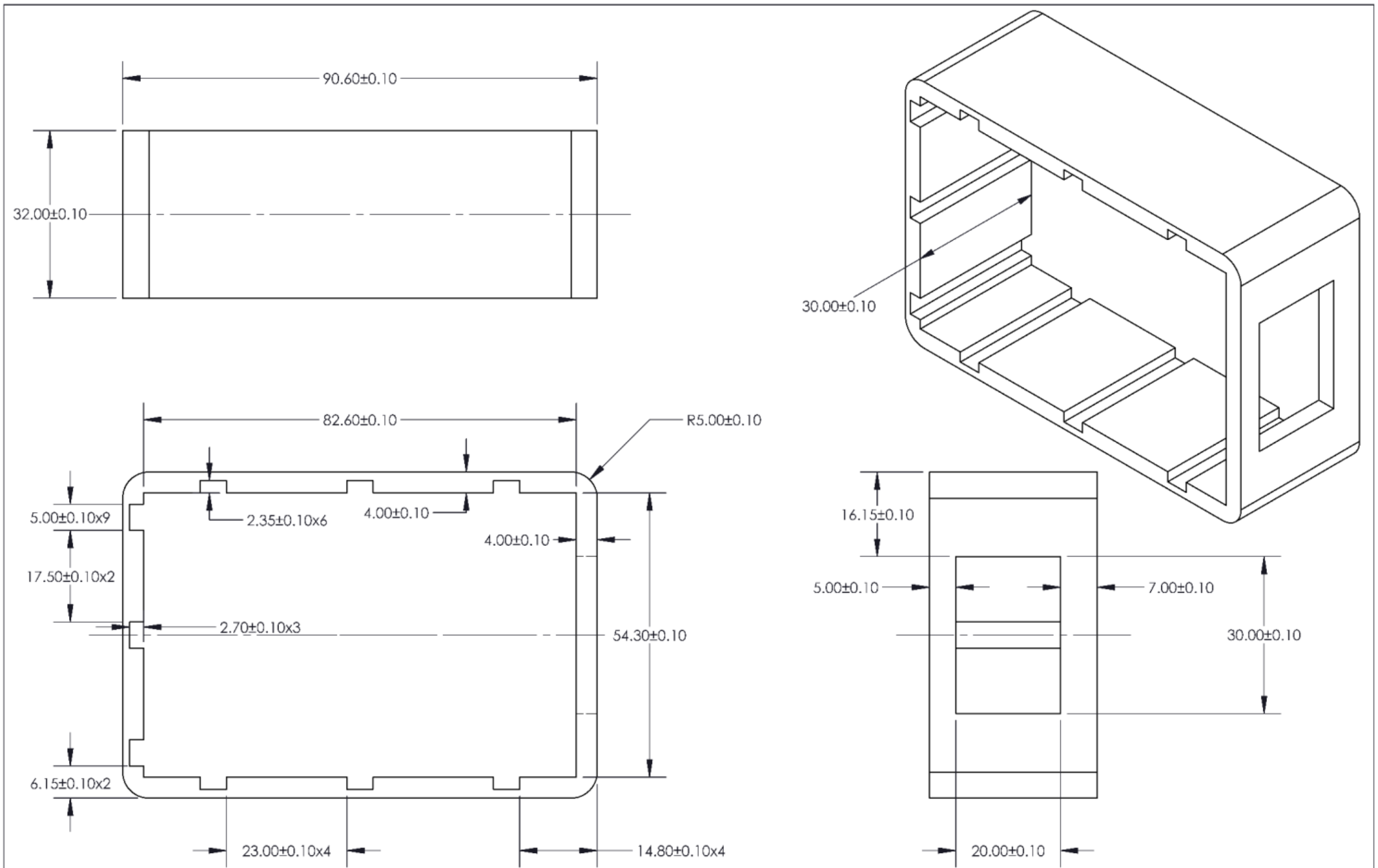
Appendix C: Engineering Drawings

Plate with SSR hole call outs and USB panel mount

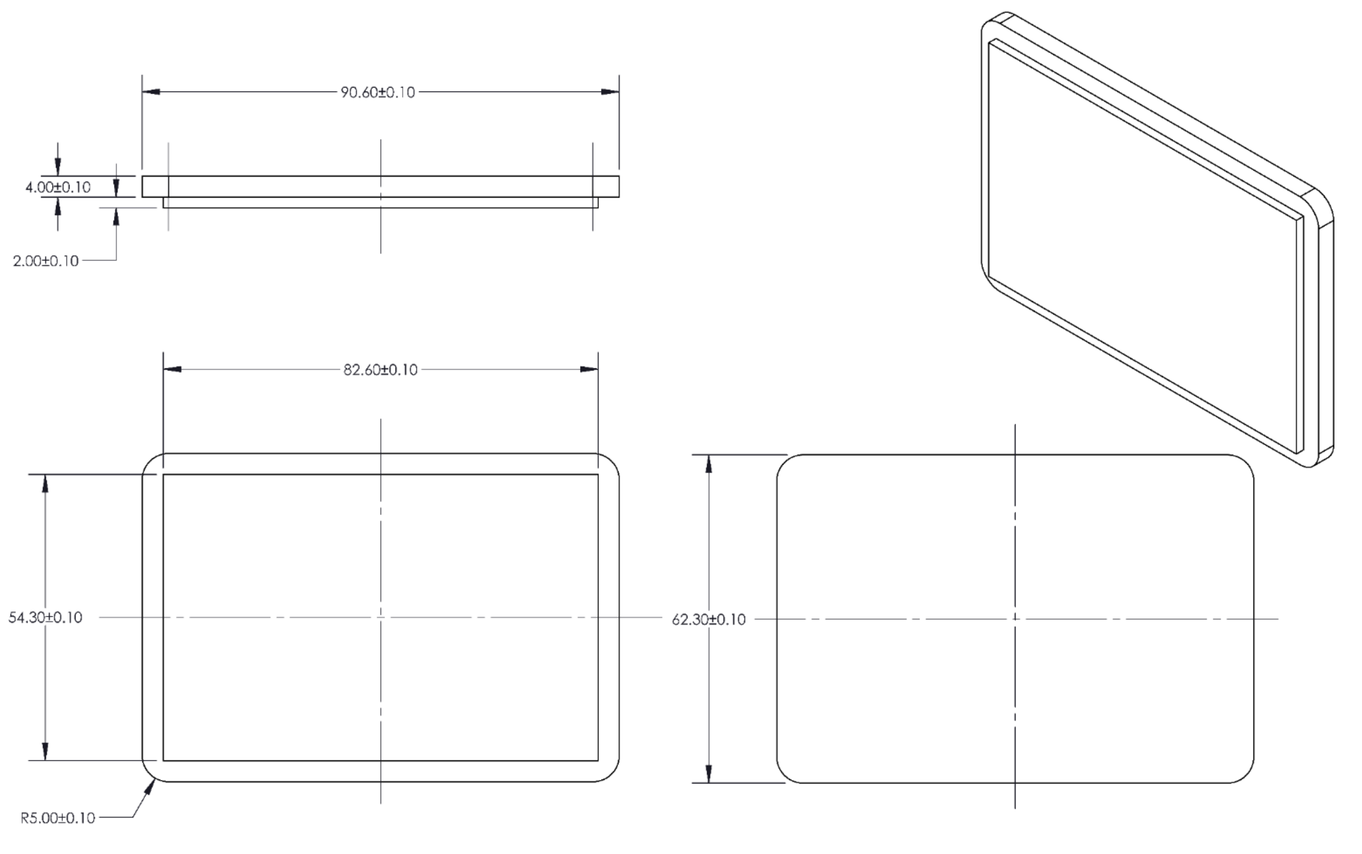


Original Design for Electronics Case (Not Fabricated):



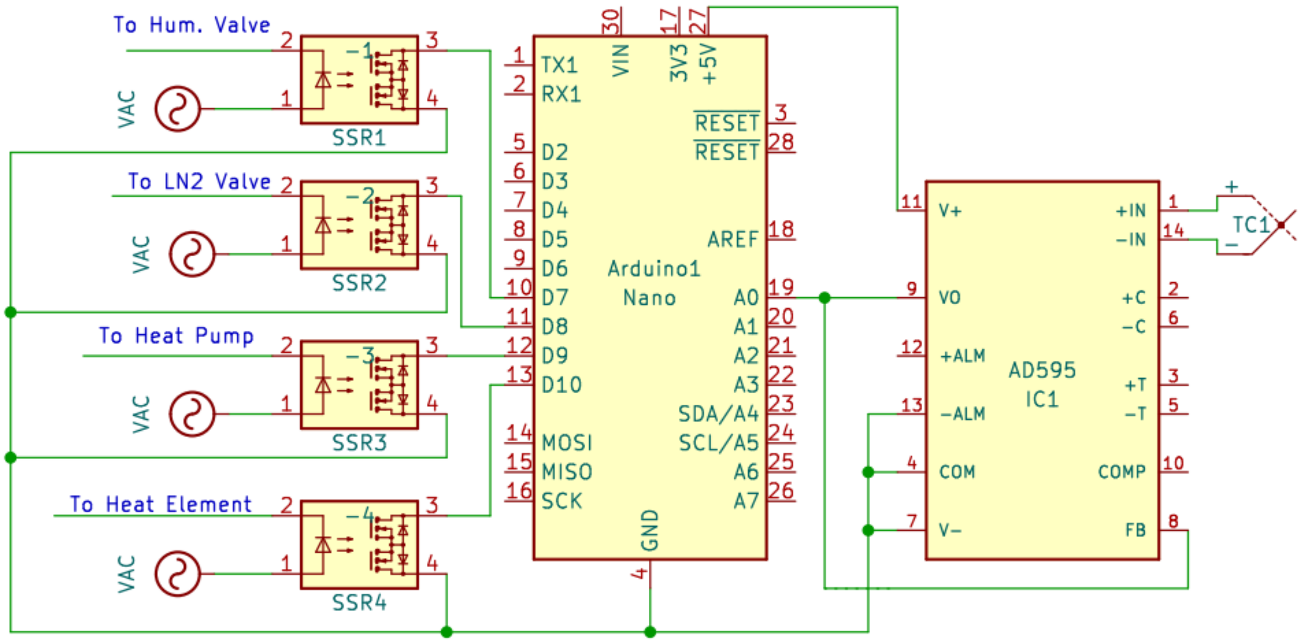


Dalhousie University	PROJECT: Electronics Case		DRAWING: Case Orthographic & Isometric		B00#: 832851	
	UNLESS OTHERWISE NOTED: SURFACE 1.6 µm; UNITS: mm X.XX +/- .10, X.X +/- .20, X +/- .50; UNITS: in X.XXX +/- .005, X.XX +/- .01, X.X +/- .02; ANGLES +/- 0.5°		DWN BY: Nizar Al Aufi		SIZE A	REV 1
			MATERIAL: PLA	QTY: 1		
			DATE: 2023/01/18	UNITS: mm	SCALE: 1:1	SHEET 1 OF 1

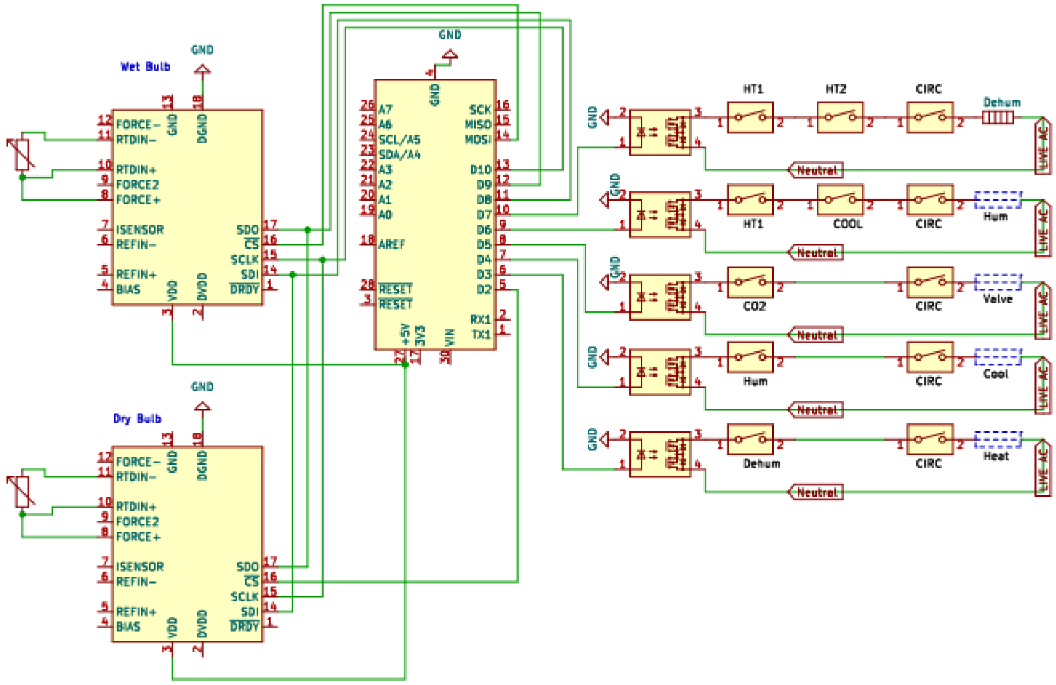


Dalhousie University	PROJECT: Case Cap				DRAWING: Cap Orthographic & Isometric		B00#: 832851
	UNLESS OTHERWISE NOTED:				DWN BY: Nizar Al Aufi		SIZE: A / REV: 1
	SURFACE: 1.6 µm	UNITS: mm X.XX +/- .10 X.X +/- .20 X +/- .50	UNITS: in X.XXX +/- .005 X.XX +/- .01 X.X +/- .02	ANGLES: +/- 0.5°	MATERIAL: PLA	QTY: 1	
					DATE: 2023/01/18	UNITS: mm	SCALE: 1:1 / SHEET 1 OF 1

Original Wiring Diagram



Full FINAL Wiring Diagram:



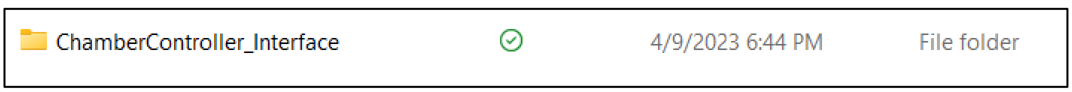
Appendix D: User Manual

Environmental Chamber Controller
User Manual

This document contains a user manual for operating the environmental chamber in room D115A, located on the Sexton Campus at Dalhousie University. The digital controller for the existing chamber was designed, fabricated, and installed by Team 03 as part of the 2023 Mechanical Engineering Capstone program. For any inquiries, please contact reilly.pickard@dal.ca.

1. Downloading and Opening Software

An email has been sent to the user containing a folder named "ChamberController_Interface". Download this file. In your downloads you should see the folder, as shown in the figure below:



Open the file and navigate to the ChamberController_Interface.exe file. It is the only file whose type is "Application". Open the file shown:

2. Connecting to the Arduino

Locate the USB port coming from the door () of the chamber and connect your USB-Mini. Connect the other end to the USB port on your computer.

If you are properly connected, you will see this:

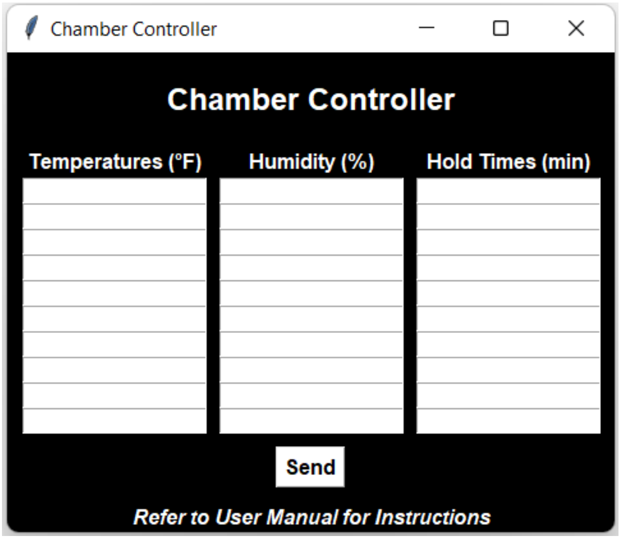
```
Connected to USB-SERIAL CH340 (COM4) on COM4
```

If not, you will receive the error shown:

```
No ports available. Please connect your device.
```

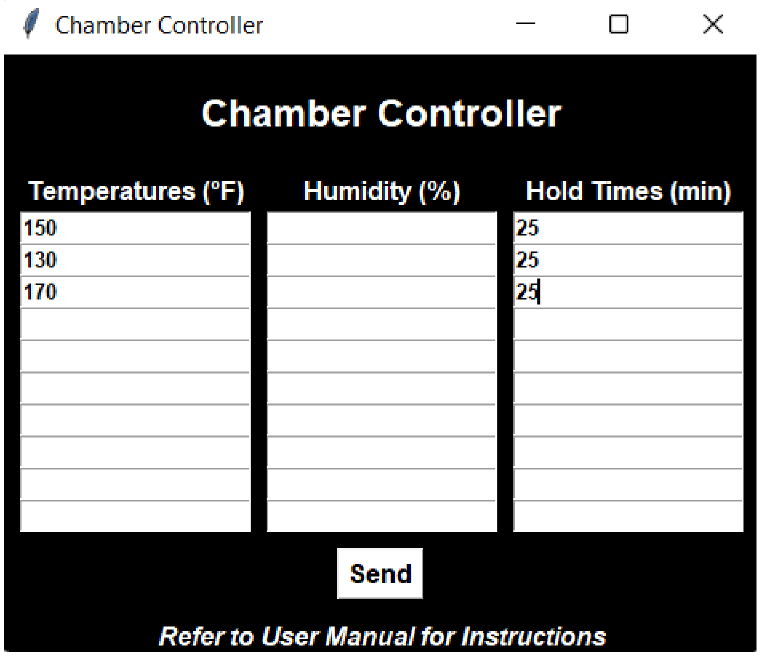
3. Controlling the Chamber

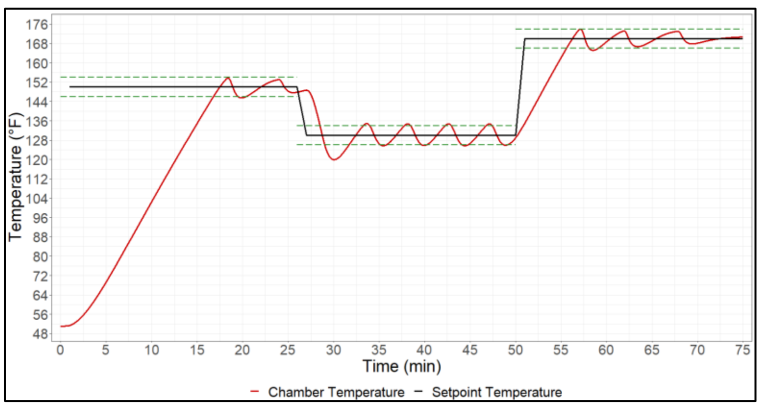
Once the app is open, wait about 10-20 seconds while it loads. You should begin to see temperature readings on the plot and being printed to the command window. To begin controlling the temperature and humidity, navigate to the window shown.



Enter the temperature and humidities you want to achieve, and the time you wish to hold them. Once the hold time is elapsed, the controller will go to the next row and aim to meet the second temperature and humidity and hold these for the second hold time. Once you have filled out the table with your desired setpoints, press the “Send” button to start. To ensure that the chamber responds, make sure the power lever is switched up, and that the circulation, heating, refrigeration, humidification, and dehumidification switches are flipped ON. If you wish to go below 0^{o}F, also ensure the CO2 switch is on.

Shown below is an example of 3 setpoints being entered and met. **Note: the "hold time" counts the rising time. So, if the current temperature is 70 $^{\circ}$F and you wish to hold a temperature of 90 $^{\circ}$F for 10 minutes and the current rise time is 2 $^{\circ}$F/min, you must enter 20 minutes of time to account for the rise.**





Note: Leaving the humidity portion blank just means that humidity will not be controlled during the run.

4. <u>Safety</u>

The user should also be made aware of the potential safety hazards associated with the system. Best practices include, but are not limited to:

i. Do not open the door to the controller housing while the machine is in operation. There is live AC flowing and this is very dangerous.

ii. Do not open the door to the chamber while the treatment is in process. Not only will the extreme temperatures be of danger to the user, but this will cause a large disturbance in temperature reading due to the loss of insulation.

iii. Do not bring liquid around when operating. Liquid may damage not only your computer in the middle of a run but may damage the USB connection to the Arduino.

The user should also note that the design is equipped with the following safety features:

i. If the system temperature exceeds 350 ^{o}F or drops below -100 ^{o}F, 'Off' signals are sent to each of the machine components.

ii. If the temperature exceeds these defined bounds, an e-mail alert is also sent to the user, as shown below:

C

chamberalerts@outlook.com

CAUTION: The Sender of this email is not from within Dalhousie.

Subject: CHAMBER ALERT.
The chamber has exceeded the maximum temperature. Please go check to ensure the chamber has powered off

Reply | Reply all | Forward

iii. The machine may also be powered off by either disconnecting the USB, or closing the power lever shown in Figure 8. Should the machine be out of maximum range and the temperature doesn't seem to be changing, please disconnect the power rather than trying to change setpoints on the interface.

Power Lever: