MEMORANDUM

FURNACE IMPROVEMENT SYSTEMS
CAL POLY

To: Dr. Richard Savage, Department of Biomedical Engineering, Dr. Ben Hawkins, Department of Electrical Engineering, Dr. Hans Mayer, Department of Mechanical Engineering, Cal Poly SLO

From: Kenji Anzai kanzai@calpoly.edu
Ethan Fedor efedor@calpoly.edu
Patrick Zailaa pzailaa@calpoly.edu

Date: December 6, 2018

Subject: Furnace improvement project Critical Design Review document

The Final Design Review (FDR) documents the senior project and marks the end of the project work. Included in the FDR are modified sections of the Scope of Work, Preliminary Design Review, Critical Design Review, and additional sections addressing the manufacture and testing of the system. The Final Design section details the system and its components, the Manufacturing section explains how the system was manufactured, and the Design Verification section outlines how the design was tested to ensure it meets requirements.

Final Design Review

Furnace System Improvements Senior Project
December 6, 2018

Abstract

This Final Design Review (FDR) documents the work done by the group to manufacture the system to deliver improvements to the furnace in the Cal Poly microfabrication laboratory (better known as the cleanroom). The document describes the material covered in the Scope of Work (SOW), including background research, customer needfinding, and the problem statement, material associated with concept development and selection for Preliminary Design Review (PDR), material detailing the design and manufacturing plan of the system for Critical Design Review (CDR), and new material detailing the manufacture of the system for FDR.

Research done for the SOW resulted in a list of engineering and customer specifications. These in turn led to the development of the goal of the project: to make the furnace system more user-friendly to operate by simplifying the tubing layout, introducing digital control, and creating a consolidated user interface.

Following the development of the problem statement, the problem was decomposed into its individual functions, and solutions developed for each function. These solutions were analyzed, and the best were combined into a final design. The solution is a system consisting of a tubing subsystem, a user interface, and a control subsystem to connect the two. A prototype system has been designed to test key functionalities of the control system, and design, manufacturing, and test plans have been created for the subsystems.

The system was intended to allow the user to set up a temperature and gas program, start the run, and not interact further with the system until the end of the run. In the event of an unsafe condition detection, the system would raise an alert (it was assumed that users would remain in the room as a matter of safety during runs). The fluid lines were vastly simplified, and gas programming is performed on a touchscreen user interface connected to a Raspberry Pi board. It was originally intended to be able to perform temperature programming on the same interface, but this functionality has been removed from the project as a matter of safety, as requested by the sponsors. The new interface and all controlling functions run on software written by the team, which is publicly available and free for any subsequent project to modify and use. The openness of this setup allows further improvements to the electronics and software as necessary.

The system was mostly built as designed, although several key functionalities are incomplete. As such, the system was not installed in the cleanroom. Therefore, finishing the remaining components and installing them in the cleanroom are tasks which are left to subsequent senior project teams.

Next steps for subsequent projects include implementing bubbler refill, building a graphical user interface for the gas routing system, implementing communication with the thermocouples, installing the pressure monitoring system, and implementing automatic gas switching. After this, the system may be installed. Upon installation, the original gas programming interface will be retained, and the new system will be installed so that it will be possible to throw a switch to revert to the old interface in the event of system malfunction.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The current furnace assembly in the California Polytechnic State University (Cal Poly) Microfabrication Laboratory was built to house two analog furnaces and has been upgraded over the years with new components to create the partially analog and partially digital system which exists today. Unfortunately, these upgrades often involved the addition of components and piping without the removal of the components they replaced, resulting in many redundant components and tubing often referred to as "spaghetti." This has resulted in a system, which while functional, requires a large degree of operator skill and time to control, and results in frequent operator errors. The goal of this project was to simplify the layout and control of this system by redesigning the tubing system and introducing a consolidated user interface to streamline operation of the furnace.

The customers of this project are the professors and students who use the Cal Poly Microfabrication Laboratory, colloquially referred to as the Cleanroom. Specifically, this project will be addressing those who work with the furnace assembly in the cleanroom. These customers include the director of the Microfabrication Laboratory, Dr. Richard Savage, and Dr. Hans Mayer, a laboratory instructor, and all the students in a microfabrication class (BMED 435) which utilizes the laboratory.

The project team is composed of three senior Mechanical Engineering students at California Polytechnic State University (Cal Poly): Kenji Anzai, Ethan Fedor, and Patrick Zailaa.

2 Background

Background research included research into the cleanroom system, silicon wafer processing, Arduino capabilities, and valves. The information was obtained by a combination of interviews and online research.

2.1 Customer Needs

Customer needs were analyzed and determined by interviews conducted during sponsor meetings in the cleanroom.

2.1.1 Interviews: The Current Cal Poly Cleanroom Furnace System

Cal Poly's cleanroom laboratory provides students to learn about microfabrication through exercises in which they fabricate semiconductor parts such as transistors and diodes. To this end, the cleanroom contains several machines and devices which are part of the process, including fume hoods, etchers, aluminum depositors, and furnaces. These furnaces were the focus of the project.

There are two furnaces in the cleanroom for processing silicon wafers. One furnace is dedicated to oxidizing processes, while the other is dedicated to diffusing processes. The furnace system can be broken down into several elements and subsystems which control its processes and feed the furnaces the required inputs for their processes.

Figure 1 shows a general outline of the furnace system. From this diagram, the major elements can be identified as the gas tanks, chase panel, flow rate and mass flow control panel, the gas flows control panel, the bubbler assembly, the diffusion furnace, and the oxidation furnace. A more thorough (although still incomplete) diagram of the system is visible in Figure 1 below. A more detailed diagram is shown in Appendix E.

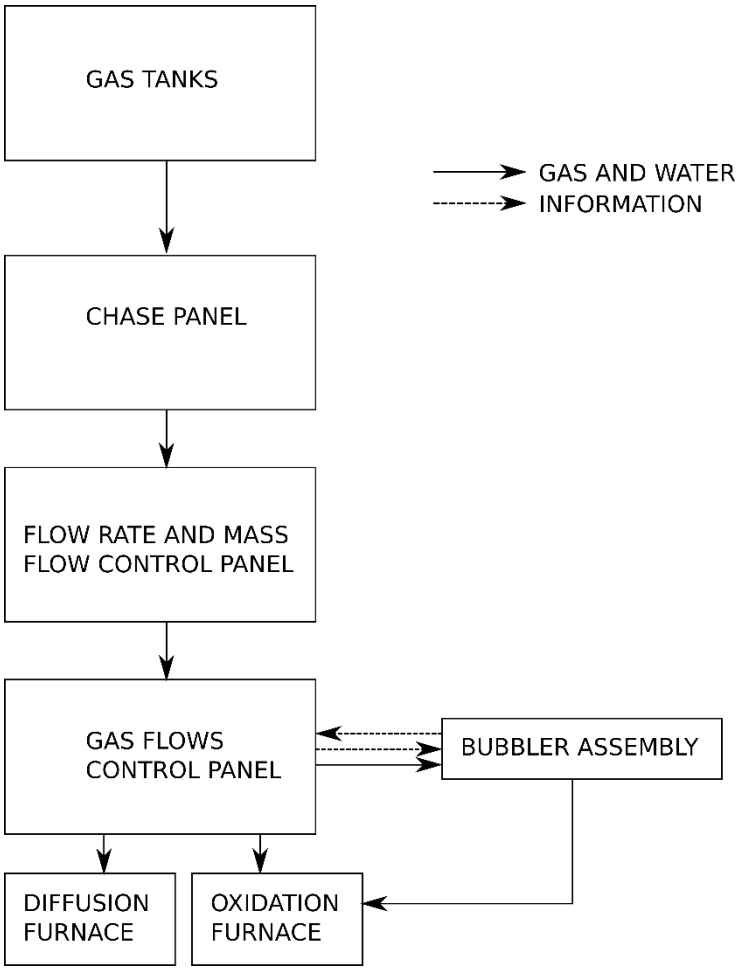


Figure 1. Simplified diagram of mass and information flows in the furnace assembly. A more detailed diagram will be generated as the project progresses and more research is performed on the system.

Also visible on the diagram is the flow of gas and water throughout the furnace system, which constitutes the fluid delivery subsystem. The fluid delivery subsystem ultimately delivers gas to the two furnaces and to the multiple solenoid valves employed across the device. There are four gases (high purity nitrogen, high purity oxygen, low purity nitrogen, and low purity oxygen) and one liquid (deionized water) which is dispensed from this station, and all tanks are equipped with hand-operated valves. From the gas tanks, the fluids flow through pipes to the chase panel, where they pass through another set of hand-operated valves, and then on through the flow rate and mass flow control panel, where the flowrate of the high purity nitrogen and oxygen can be regulated with knobs. The fluids then pass through the gas flows control panel, where gases may be turned on and off for a length of time. If the gas flows control panel is set to provide wet gas to the furnace,

the high purity gases can be redirected to pass through a bubbler assembly before injection into the furnace. Otherwise, the gas is fed directly into the furnace.

There are several control subsystems in the furnace system which are not apparent in the diagram. Firstly, the gas flows control panel requires a control system to command several solenoid valves, which open and close based on the settings on the control panel. Every solenoid valve in the system requires a delivery system by which pressurized gas may be delivered to it as well as some regulator to control the flow of the gas. The bubbler assembly is encapsulated in heating mantles, which are controlled with dials on the gas flows control panel. Finally, the two furnaces each have their own independent control panels.

Every element in the system, including the two furnaces, is a rudimentary component which has been repurposed to fit the apparatus. At some point in time, the pieces were assembled together into a functional assembly, likely in a student project, but there is no documentation of this. The hand-made nature of the assembly has resulted in numerous problems, as identified below.

2.1.2 Customer observations

Dr. Hans Mayer, one of the faculty who operates the cleanroom, identified several problems with the current setup. Firstly, the valves on the chase panel and gas tank are manually operated, so any operator must physically cross the laboratory to turn on the flow of gas. Also, there is no way to see whether the valves are open or closed except by crossing the room to examine them, and no way to determine if there is enough gas to complete a process except by in-person examination at the tank. These three factors unnecessarily complicate the procedures performed with the furnace, increasing the possibility of error. The controls are divided across four locations and are not necessarily intuitive. The temperature setting on the diffusion furnace, for instance, is a combination of three settings which students must read the laboratory manual to know how to operate since it is impossible to determine how it operates by examination. Furthermore, the temperature readout for the diffusion furnace is an analog dial. The oxidation furnace temperature control panel allows for a temperature regimen to be input and followed, but the furnace will not wait for the proper temperature to be reached before proceeding to the next operation, which can result in operations being effectively truncated. The gas control panel allows for a gas to flow for a certain amount of time but does not allow for the input of a gas regiment. Thus, an operator must be present at the changeover to program in the new gas and duration. Additionally, while the bubbler boiled, it unnecessarily fed steam directly into the furnace whether or not wet gas was called for, so even when bubbler valves are bypassed, water vapor being injected into the furnace. This becomes an issue as the water must start boiling about five minutes before it is required. Finally, there is no liquid level detector for the water in the bubbler, so an operator must be present to press a button to inject more water every ten minutes when the bubbler is in use.

Addressing these problems and others discovered along the way was the focus of this project.

2.2 Products: Existing designs, patent search

While there are no publicized systems which meet the requirements of the furnace system, there are a few which may be of assistance. Systems exist, for instance, which can monitor and manage disparate systems around a large space, which may be of use for monitoring this complex system

(European Union Patent No. EP 3091499, 2014). There are also similar wireless systems for applications in high temperature environments, which may be of use if thermocouples are to be mounted inside of the furnace (USA Patent No. US4294682 A, 1981). Ultimately, however, most of the required system will need to be generated in-house as there is no publicly available precedent for the needs of the laboratory. Another patent we found regarding microfabrication subsystems came in the form of a high frequency pressure controller system for nitrogen lines (Japan Patent No. 19970105494, 1997). This Patent used an electronic controller to regulate nitrogen lines with high disturbance. A high frequency sensor was used in tandem with a traditional electronically controlled membrane regulator and a blow off valve to control the pressure in a nitrogen line for the purpose of supplying quartz tubes for semiconductor furnaces. A technical research section is presented in Appendix A.

3 Objectives

The objectives of the project were generally to simplify the furnace system and add control capabilities. This section elaborates further on these points.

3.1 Problem statement

As mentioned in the Customer Observations section on page 9, there are elements of the furnace assembly which are unintuitive and unnecessarily complex. Due to the presence of complex analog equipment, it was impossible to address the identified problems within the given time. Therefore, the scope of this project was limited to the elements which can be replaced (such as valves and pipes) and elements which can be controlled digitally.

Our problem statement is as follows:

Students and professors who use the cleanroom furnace need a simplified gas feed network connected with a comprehensive control interface for the furnace system because the system is composed of many connected and occasionally redundant systems, resulting in complex and unintuitive operating procedures which lead to many process errors.

The project was evaluated according to the extent to which it addresses the needs expressed in this problem statement.

3.2 QFD process, Specifications Table, discussion of specifications

Based on the information gathered regarding the customer's needs, a Quality Function Deployment analysis was performed (full table visible in Appendix K), and a list of engineering parameters developed as displayed in Table 1 below.

Table 1. Engineering specifications generated from customer wants specifications. Risks of H, M, and L correspond to High, Medium, and Low risk, respectively. Compliance symbols I, S, A, and T represent compliance verification by Inspection, Similarity to an existing design, mathematical Analysis, and Testing, respectively.

Spec Number	Parameter Description	Requirement or Target	Tolerance	Risk	Compliance
1	Electricity directly controlling oxygen lines	None	-	M	I
2	Size	36" x 48" x 15"	Max	L	I, S
3	Number of control panels	Less than 2	Max	M	I, S
4	Voltage	Less than 120 V AC	Max	L	I, T
5	Temperature setting program	Yes	-	M	T
6	Gas regimen programming	Yes	-	M	T
7	Unsafe condition detection	Yes	-	H	A, T
8	Temperature resolution	Less than 10 °C	± 10 °C	L	A, T
9	Pressure resolution	Less than 1 kPa	± 1 kPa	L	A, T
10	Cost	Less than $5000	Max	M	A
11	Notifies user of insufficient gas	Yes	-	M	A, T
12	Automated bubbler refill	Yes	-	M	T
13	Number of pneumatic valves	0	Max	M	I

An explanation for each of these specifications follows.

1. Electronic control devices for lines can spark, constituting an explosion hazard for high purity oxygen lines. For reasons of safety, it is imperative that there be no electronic devices directly controlling high purity oxygen lines. This was to be tested by inspection.
2. The new system of tubes and wires must fit in the space underneath the furnaces in the current system, and in the same boxes that house the current assemblies. Since the project involves eliminating redundant components, this should not be problematic. This was to be tested by inspection.
3. Simplicity is paramount for the system. Therefore, it was important to maintain or reduce the number of control panels in the furnace delivery system. This was tested by inspection.
4. Voltage must not exceed 120V AC maximum voltage to minimize the hazard of electric shock. This maximum voltage was chosen because this is what is required to operate the solenoid valves. Compliance was to be tested by measurement.
5. A temperature setting program for setting a temperature regimen, monitoring and controlling the process is requested, and is one of the goals of the control system. This would be an improvement over the existing system, which does not allow for measurement

or automatic control. The existence of the system was to be determined by inspection, and its functionality determined by testing. While not absolutely required, such a program would make the system more user friendly to operate.

6. Gas regimen programming. A program was envisioned that could control an entire run, from start to finish, without having the need for an operator to be present during the run. This includes ramping the temperature up when it needs to be ramped, holding it constant when it needs to remain constant, and regulating the gas flows into the furnace as needed and at the correct times. The presence of such a system was to be determined by inspection, and its functionality determined by testing.
7. Unsafe condition detection. A list of potential safety hazards and failure modes in the furnace operation was determined, and the system can be designed to detect and report such hazards or shut down if the situation is too dangerous to continue. The existence of this system was to be determined by inspection, and its effectiveness determined by testing.
8. Temperature resolution in an installed measurement device was to be determined through measurement and testing.
9. Pressure resolution in an installed pressure measurement device was to be determined through measurement and testing.
10. The project budget was approximately $5000. Expenditures were monitored throughout the course of the project, and the total was to be tallied at the end to determine compliance.
11. A convenience feature requested by users is a system which detects the gas level in the gas tanks and notifies users if insufficient gas is present to complete the process. The presence of this program was to be determined by inspection, and its effectiveness determined by testing.
12. The feasibility of replacing the entire bubbler subsystem with a steam generator was examined and determined to be feasible. The existence of this system was to be determined by inspection, and its effectiveness determined by testing.
13. It was desired to reduce the number of pneumatic valves to simplify the system. If the pneumatic valves could be replaced with solenoid valves, the low-purity nitrogen supply could be removed from the system. The pneumatic valve count was to be determined by inspection.

As part of the QFD method, current competitor systems were analyzed. It was difficult to find many competing systems since most existing solutions to this problem are unpublicized private setups in factories and universities across the country. We did however manage to find literature on a similar project involving a PID temperature control of a diffusion furnace system at the University of Southern Maine (M.G. Guvench, 1997). This system, along with our current furnace system were used as competitors in our QFD table. 1)Appendix D displays how those systems measure up to the identified engineering specifications.

4 Concept Design Development

4.1 Concept development process & results

Concepts were developed by breaking the system up into its component functions and developing concepts for performing each individually. The first part, known as functional decomposition, led to the identification of five component functions:

1. Routing: The system must direct gas to the appropriate location
2. Controlling: The system must allow for control of furnace temperature and gas flows
3. Measuring: The system must measure pressure and temperature values in key locations
4. Displaying: The system must display information to the user
5. Alerting: The system must alert the user to anomalous conditions

Ideation for each function yielded a number of possible solutions for routing, controlling, and displaying, a full list of which is presented in the Pugh Matrices in Appendix C.

4.2 Concept selection process & results

The initial selected solutions were analyzed by comparing their various properties against each other and the current system. This method (carried out using Pugh Matrices, as seen in Appendix C) yielded the component systems of the initial design.

The initial design was to perform its functions with the following components:

1. Solenoid valves to route gas to the appropriate location
2. A PC with microcontroller architecture will be deployed to control the furnace
3. Electronic pressure gauges and thermocouples will measure the pressures and temperatures of important locations
4. Labview will be used as the user interface
5. Popup windows will be used to alert the user of anomalous conditions

4.3 Preliminary analyses, concept models/prototype, proof-of-concept testing

Prototyping of the user interface revealed that a more intuitive control was possible even without deploying any new electronic hardware or functionality. Successful hardware and software implementation would allow for temperature and gas flow regimen programming interfaces, further consolidating and facilitating control of the furnace.

4.4 Selected Concept

The final concept consisted of three subsystems: the gas delivery system, comprised of the pipes which deliver the gas to the furnace, the user interface and controller, comprised of the electronics and wires used to control the gas flows and temperature regiment of the furnace, and the software to operate the control system. The systems were to be implemented as displayed in Figure 2 and Figure 3 below and are the second iteration of the design. All iterations are diagrammed in Appendix F.

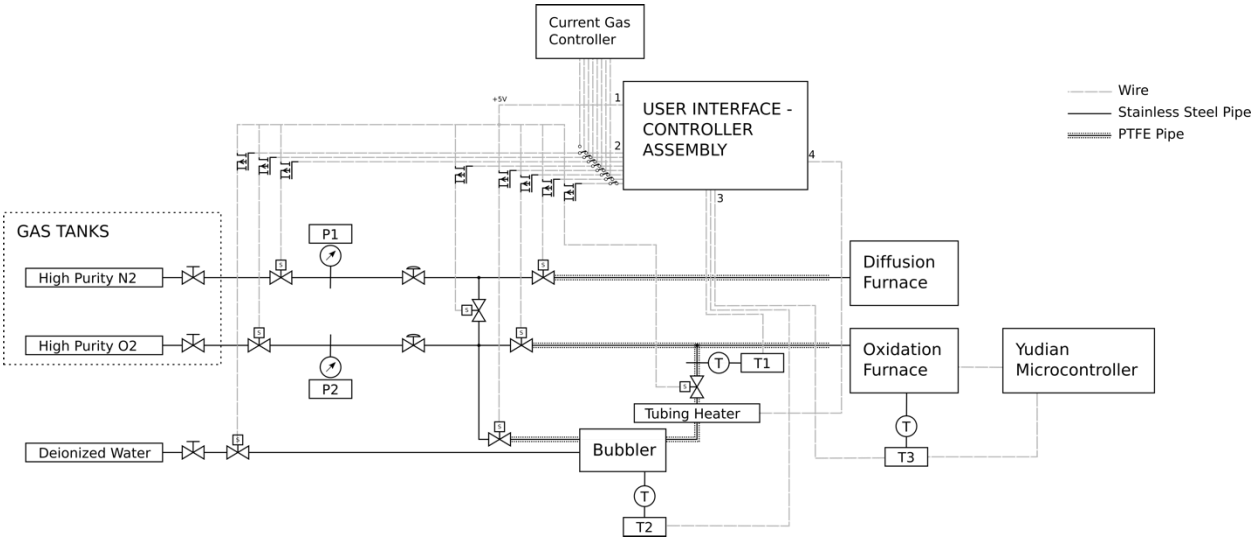


Figure 2. Diagram of proposed system, consisting of a tubing system, user interface-controller assembly, and wiring. The user interface-controller assembly was designed separately, and is displayed in Figure 3 below.

By using oxy-safe solenoid valves, this design eliminated the need for pneumatic valves and the low-purity nitrogen system, eliminating nearly half of the valves in the original system. The redesigned user interface put as many controls and readouts on one panel in as compact and intuitive of an arrangement as possible, preferably on a touchscreen or computer monitor. The controller will be a microcontroller programmed to take an input signal and translate it into an output for each controllable device. Every signal fed into the Raspberry Pi, except for the volumetric flowrate knobs, which were set once and never changed.

Originally, the system was designed to allow users to control furnace temperatures as well as gas flows. After PDR, however, it was requested by our sponsor that the temperature setting interface be kept as it is on the original system, as implementing the change on our interface would require a potentially dangerous dissection of the current furnace interface. Also, the sponsor requested that control of the furnace be transferrable to the current interface in the event of emergencies. Further, it was discovered that the tubing could be simplified further from the layout shown in PDR. For these reasons, the user interface was modified to exclude temperature control. However, gas control was maintained, and it will be possible to make the gas control dependent on the temperature of the furnace, a feature which was requested by the sponsor. The system was also modified to allow the control of the gas flows to be switched back to the current control system. Figure 2 and Figure 3 reflect this change and constitute the system to be implemented.

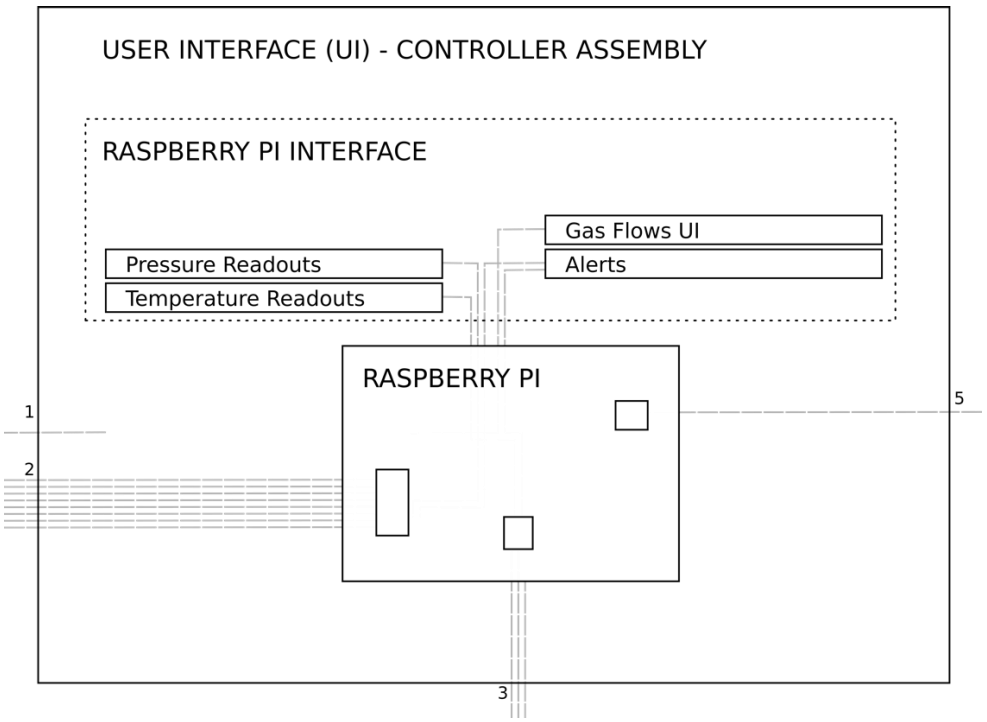


Figure 3. Selected user interface-controller assembly consisting of a digital user interface (a touchscreen monitor) and a Raspberry Pi computer to control the valves.

4.5 Selected concept functionality

There were several design criteria that were put forth that our solution was required to fulfill. The design had to be completely automated, it had to be programmable, it had to fit within the footprint of the current oven, it had to be portable to future design changes, it had to be safe, and it had to be easy to use. Most importantly, the build on the furnace needs to happen quickly and it needs to be implemented correctly the first time.

The design we have selected meets all the above criteria. The proposed pneumatic line and valve scheme was designed to maximize simplicity, eliminating the redundant clutter of the current system. In addition, the extra valve after the bubbler was included to ensure that steam does not find its way through into the oven when not required. With the addition of the Raspberry Pi, the system was capable of monitoring and controlling all valves in the gas lines, and all temperatures in the furnace. The Raspberry Pi board was programmable and had data acquisition capabilities. Because the board would have read information from the sensors, it would be able to throw alarms for high/low pressure in the lines, and high/low temperature in the furnace and would also be able to make note of these warnings in the collected data. The Raspberry Pi was an elegant solution to many of the issues at play because of its versatility, and its software could be easily be patched for new features or debugging. This solution also allowed for a simple touchscreen interface, making it relatively easy to fit the interface within the current footprint. All hardware in the gas and electrical lines were industry standard, making it easy to port additional design updates into the selected design. Lastly, this design could be completed and tested before its implementation into the furnace. All valves and relay voltages could be checked and debugged before they are built

into the oven. This significantly decreased the likelihood of error when the new componentry is added to the furnace.

4.6 Discussion of challenges, unknowns, and risks with current concept

The biggest challenge that faced the group was the development of the electronics and software necessary to create the control system. This system would be preferred to simplify the user experience, but the software requirements were beyond our largely mechanical understanding. All efforts were made to learn about and deploy the electronics, but a contingency plan would be put in place in the event that it was not possible to deploy the system. If it became apparent that the electronic system requirements were beyond the capacity of the group to implement, the following contingency plan was to be implemented:

1. The tubing and valve system would be implemented as defined above.
2. The control panel would be designed as described above without new digital components.
3. The system will be set up in such a way that the electronic components could be simply plugged in once created without the replacement of any mechanical components.
4. The design of the electronics would be delegated to a subsequent electrical or computer engineering senior project, and all research will be made available to the subsequent team.

Several other risks associated with this concept include the risk that coding errors or software malfunctions would create a dangerous situation with the furnace controls. Adequate simulation and controlled testing should minimize this risk, but care should be taken in the early use of this system, and the laboratory should not be left unattended for long periods of time while the system is running.

5 Final Design

The final design of the system is detailed in this section. Design outlines for code and interfaces have been provided. Note that this final design was not completely implemented due to time constraints and represents the goal rather than the system which was actually achieved. Section 6.2 details the system as built.

5.1 Functional description

As discussed in section 4, the final design consists of 3 subsystems: the routing, the electronic hardware, and the software. The routing hardware consists of solenoid valves, pressure gauges, couplings, and tubing to route any desired gas to any desired output. The electronic hardware consists of relays, sensors, computer systems and displays for the user interface. The software consists of cooperative tasks responsible for identifying user input and making the necessary changes to the system by reading from the sensors and giving tasks to the electronic hardware to make the appropriate changes in the valve hardware.

5.1.1 Routing Panel

The routing hardware is the heart of the system. A diagram of the routing can be seen below in Figure 4. The lines go from high purity nitrogen and high purity oxygen tanks through regulators to control the pressure, then through a series of valves and to the chase panel with manual ball valves as a failsafe. From the chase panel the lines go through a series of solenoid valves connected to the controller. Depending on the desired gas routing, the controller can actuate the valves such that the oxygen can go through the bubbler to the oxidation furnace or straight to the oxidation furnace, and the high purity nitrogen line can to both the thermal oxidation furnace and the diffusion furnace. As seen in Figure 4, the lines in the routing panel are stainless steel for structural rigidity while the lines going from the routing panel to the furnace are Teflon to allow for routing flexibility around the frame of the furnace. The routing panel will contain the six solenoid valves and the tubing between them and is connected to all fluid lines to and from all other locations. A drawing of this panel is displayed in Figure 5.

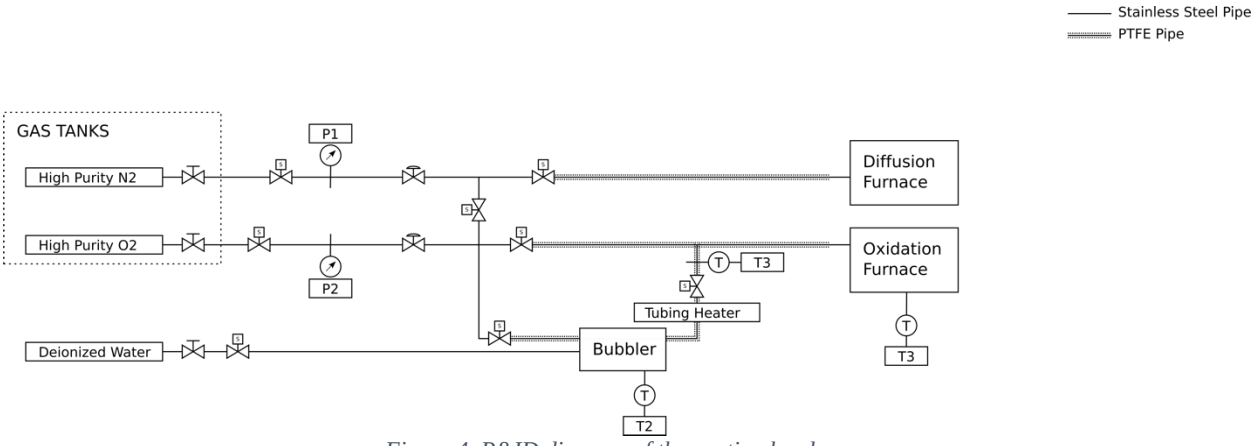


Figure 4. P&ID diagram of the routing hardware.

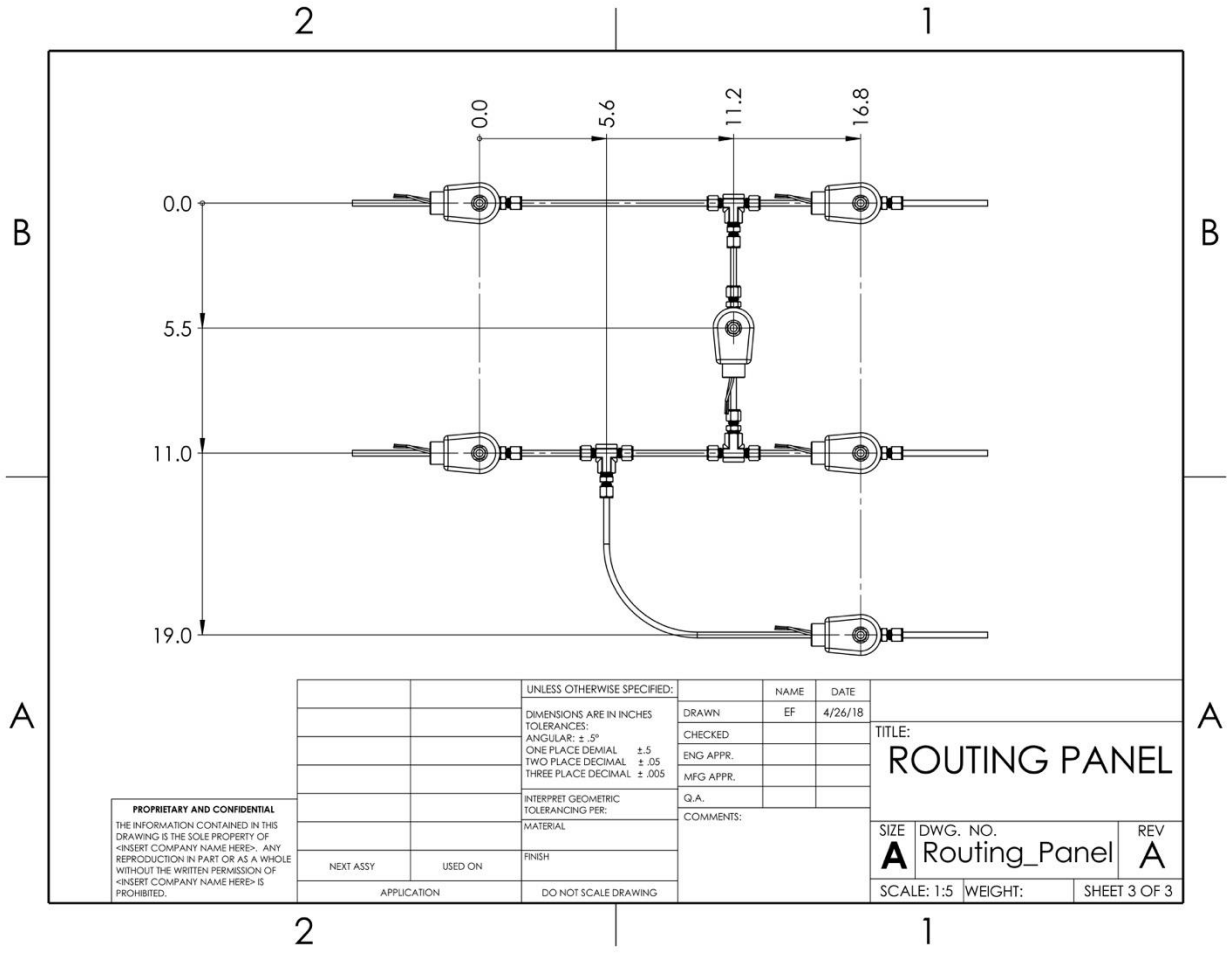


Figure 5. Drawing of the routing panel tubing and valve layout. This will be mounted on Unistrut channels and installed in place of the current tubing system.

5.1.2 Electronic Hardware

The electrical hardware acts as the intermediary between the routing hardware and the software; it exists to provide data to the software, and to allow the software to interface with the mechanical componentry. The primary functions of the electronic hardware are computing, sensing, and actuation. The system is controlled with a Raspberry Pi. The Raspberry Pi is a microcomputer that allows for a wide range of inputs, outputs, computing languages, and displays. The Raspberry Pi is coupled with a touchscreen display, a keyboard, and a mouse to allow for interaction with the computation system. For sensors, the system will have thermocouples in each furnace to tell the computer system the temperature. The system will also have pressure transducers at the regulators in the gas tanks to alert the software when there is a possibility of running out of gas during a run. Lastly, the system will have a fluid height sensor in the bubbler to alert the system when the bubbler needs to be refilled. For the system actuation, the control panel will control a series of solid state relays so that the low voltage logic in the computer can be amplified to power levels that can control solenoid valves and heating elements; the relays will then be wired to the solenoid

valves seen below in Figure 6 and to the heating elements in the bubbler and the heating element on the tube from the bubbler to the furnace. An electronic diagram is displayed in Figure 7. It is important to note that not all of the systems diagrammed here in the original design were implemented. Among those systems not implemented include thermocouple readers, pressure gages, and bubbler refills. This is detailed more in Section 6.2.

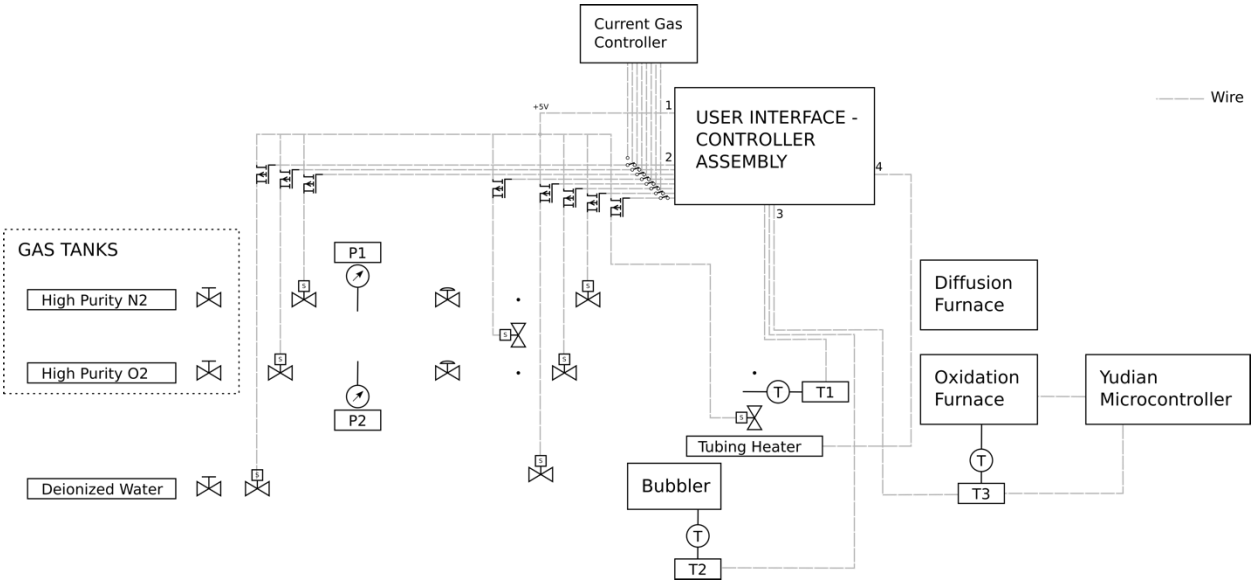


Figure 6. Wiring subsystem diagram. Each valve will be powered by wall voltage, which flows on lines regulated by relays (shown here as MOSFETS), which are in turn controlled by signals from the Raspberry Pi in the user interface-controller assembly. A full electronic diagram is shown in Figure 7.

5.1.3 Software

The software for the controller was written in Python and implemented on the Raspberry Pi board. This combination of software and hardware was chosen because they were easy to use and suitable for simple applications such as toggling valves. A system architecture consisting of seven basic tasks was originally planned to operate the system and would have passed data to each other as shown in the task diagram in Figure 8. Each task would be given a frequency and priority with which to run, and a task manager would put these tasks into a queue to be run. Every clock cycle, the task manager would examine the list and run any task that is ready to run. The actual software as implemented in Appendix O is a simple while loop which iterates until the user tells the system to stop.

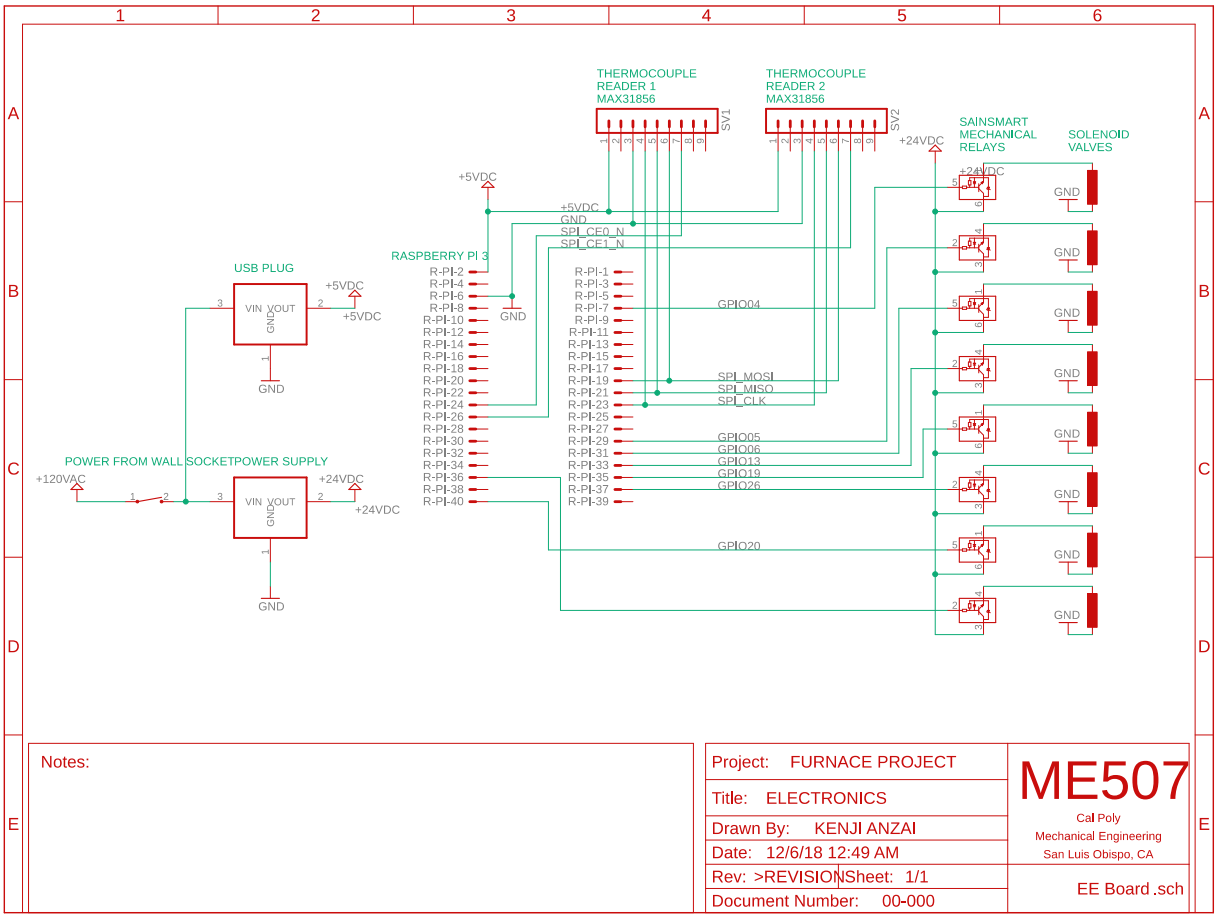


Figure 7. Electronic schematic of the wiring subsystem. Each solenoid valve (represented by the inductors on the right) will be powered by wall voltage, which is controlled by relays, which are in turn controlled by signals emanating from the Raspberry Pi board (Represented by the 40-pin header in the center).

As shown in Figure 8, each task has a period, priority, inputs, and outputs. Each task was assigned a period according to how frequently it was estimated the task would need to run, and a priority according to its importance. For example, the task which monitors the user interface for user inputs runs most frequently of all to ensure no noticeable lag when the user inputs a command, while a safety check runs only once per second since the slow response times of systems and pressures ensures that unsafe conditions will not develop within a faster timespan than a second. Ultimately,

these periods and priorities are estimates, and will likely need to be modified once the program is written and tested.

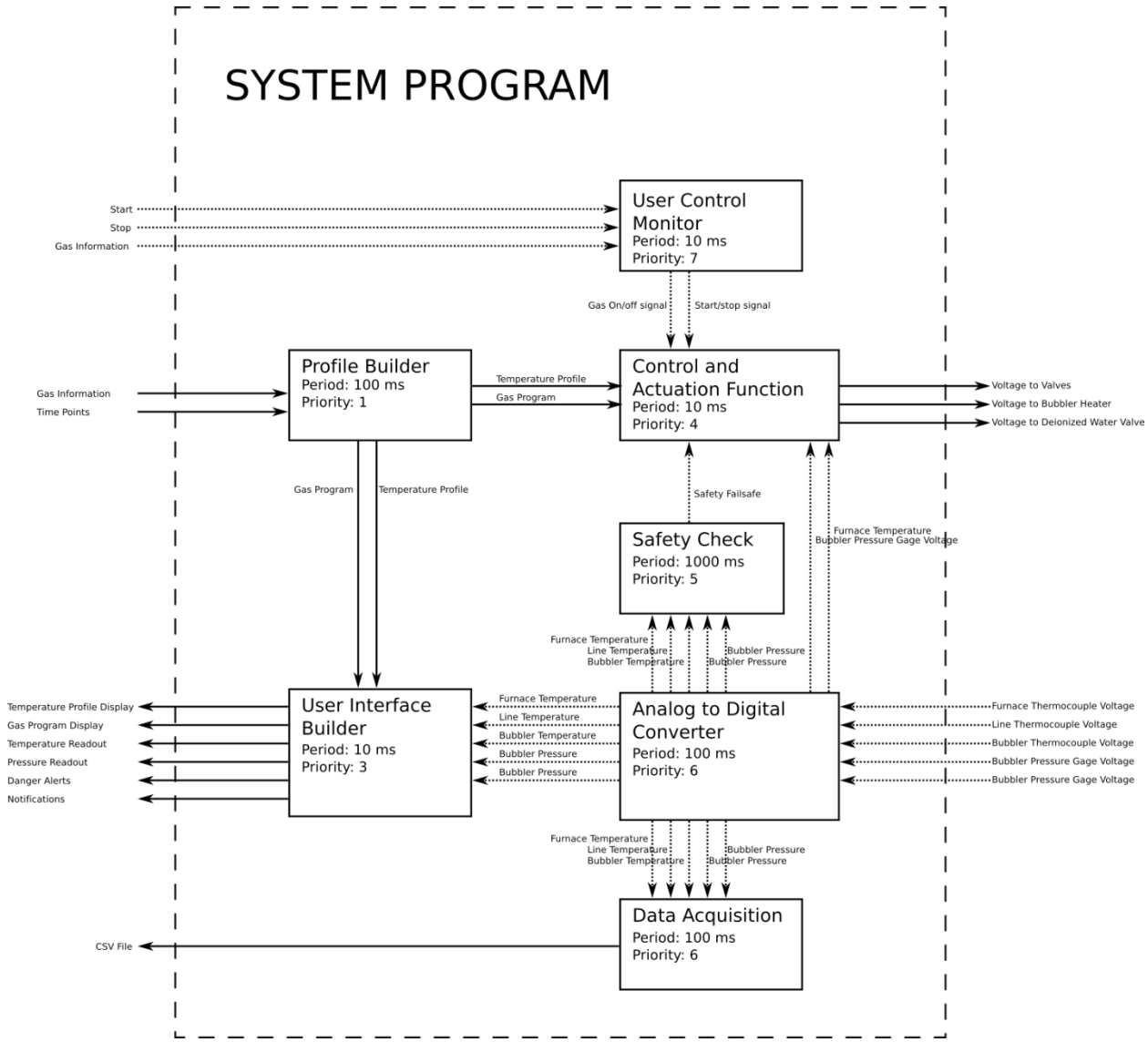


Figure 8. Task diagram of proposed controller software. Each task has been assigned a period and priority according to its perceived needs. Note that this is not a true task diagram as it also includes inputs and outputs of the system, but nonetheless serves to diagram the designed behavior of the system.

5.1.4 Graphical User Interface

The Graphical User Interface (GUI) was planned to be another integral part to our system. It was to be the main communication medium between the user and the system. Through the GUI, the user could have control over the gas flow sequencing to the furnace. The GUI also would have displayed vital information to the user such as unsafe condition alerts, low pressure alerts, and would have allowed the user to monitor the status of the system at any given point by displaying the readings of the different sensors.

Qt designer was used to develop the interface. Qt designer is a program that utilizes elements that are called Qt widgets (such as text boxes, push buttons, menus, etc...), which can be dragged and dropped on the screen to create the GUI. The advantage to using Qt designer is that it obviates the need to write many lines of code to generate a user interface by writing the code for us based on a design created through a program, with the disadvantage that the functionality is buried in large amounts of libraries which constitute Qt designer. Ultimately, this opacity is what led to the GUI failing to meet the requirements in a timely manner and is why development was halted.

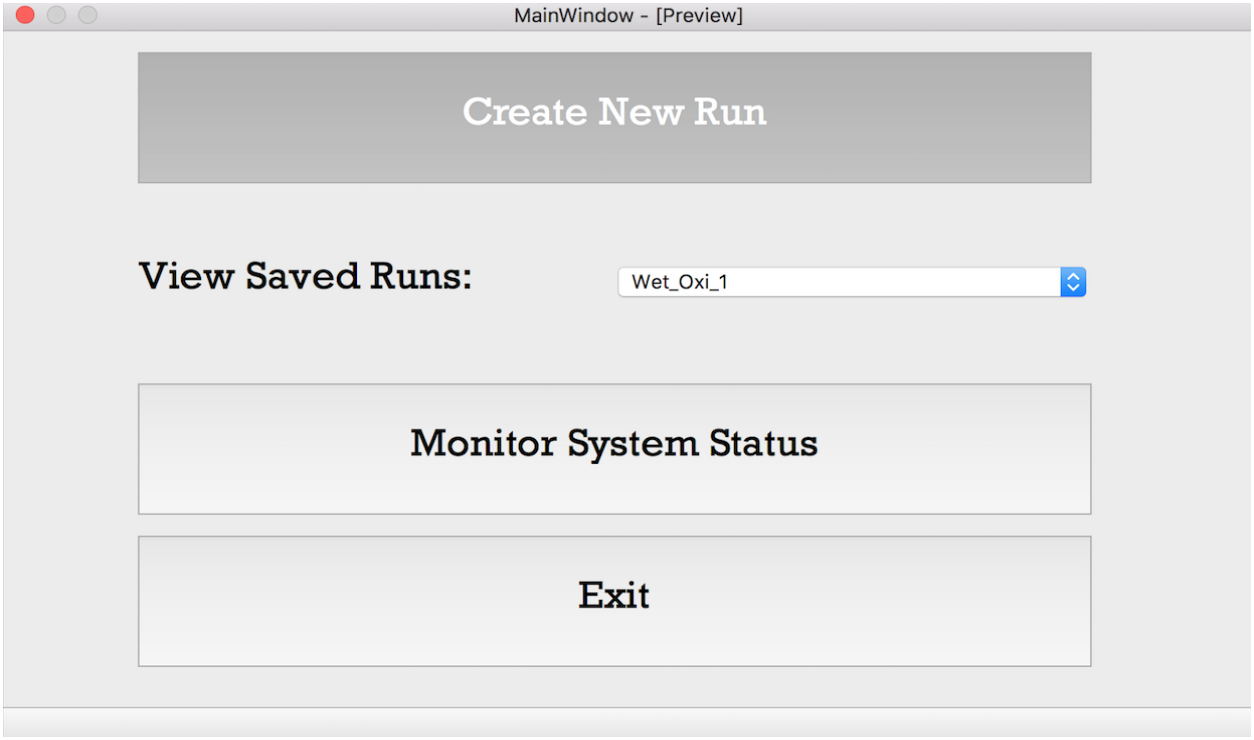


Figure 9. GUI Main Window display developed using Qt designer.

Principles of user interface design were used in developing the GUI with our main focus being clarity, simplicity, and functionality. The Raspberry Pi display selected is relatively small (800 x 480 pixels or 7" x 4.2") so crowding the screen with too many buttons widgets is ill-advised. In addition, students and instructors interfacing with the display wear latex gloves as part of the cleanroom protocol, so it is essential that push buttons and any widgets that can be toggled are sufficiently large such that anyone attempting to click them can do so relatively easily without interfering with any adjacent buttons. The following is a discussion of the GUI envisioned, if not successfully built.

The main functions of the GUI would be to display the relevant information to the user obtained by the various sensors, allow the user control over the valves, and display any unsafe condition alerts. With these functions as well as the Principles of UI design mind, Figure 9 shows a possible main window that fits our requirements. The user can decide to start a new run, select a saved pre-determined run, or check the status of the system. Selecting a new run would take the user to a

window such as the one shown in Figure 10. There, the user can create a temperature profile by inputting data into the table provided. Selecting a gas would prompt a dialog window asking the user to choose a temperature at which the gas will be triggered. The graphics on the right can expand to fill the entire screen if selected. The top graphic is the temperature profile, over which the actual temperature will be plotted real-time. The gas sequencing will be displayed on the same graphic below the time axis. The bottom graphic deals with the system status. It can tell the user which valves are open or closed and display the sensor readings where they appear on the diagram.

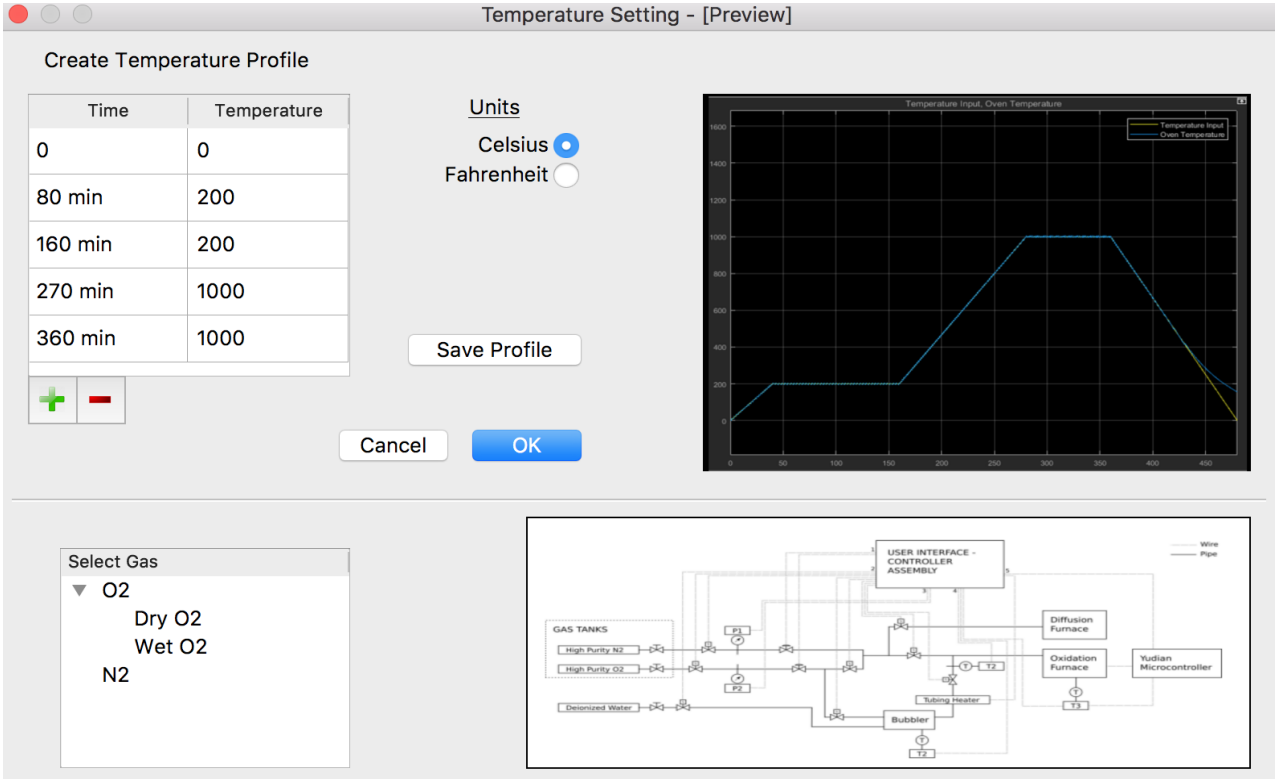


Figure 10. GUI temperature and gas flows setting window using Qt designer.

5.2 System Implementation

All the hardware in our system was be purchased rather than fabricated. This means that the equipment worked independently but had trouble interfacing. Therefore, it was important to test the equipment thoroughly, and will be important to do so again before it is integrated into the current system. One of the requirements for the project was that it be implemented swiftly and efficiently as it cannot malfunction during a lab period. Much of the system is completed, but more work is still needed to build out remaining capabilities and test the system before integration into the furnace. In the event of a malfunction, the system will be installed so that control can be switched back the current system for manual user control.

5.3 Design Requirement Satisfaction

The system as described will, upon installation, eliminate the need for pneumatic valves, and with them the need for low purity nitrogen. Also, the system will remove yards of vestigial tubing and obsolete components. This will greatly simplify the tubing network, allowing users to better understand the anatomy of the furnace and allow for easier troubleshooting, repair, and improvement.

Also, the system as designed will allow the user to set gas flows and temperature profiles in the furnace, then not interact with the interface until the run is completed. This will be accomplished by ensuring that furnace temperature milestones trigger each new gas segment, a condition which is highly programmable. This eliminates the need for the user to be at the furnace to manually switch the gas once the furnace has reached a certain temperature, as is now the case. Also, the water level in the bubbler will be monitored, and automatically refilled when the bubbler runs low, eliminating the need for the user to constantly monitor the bubbler to check the water level. These two improvements free the user from the need to constantly monitor the furnace. For safety reasons, the user should be present in the laboratory so that the furnace is not left unattended during a run, but with this automation, the user can dedicate more attention to other tasks around the laboratory that would have otherwise been spent monitoring the furnace. Such an improvement will go a long way toward streamlining furnace use and has the potential to eliminate many furnace errors which currently occur due to process complexities. While the system as built does not meet these criteria since it was not completed, a completed system still could.

5.4 Safety, Maintenance, and Repair considerations

Safety, maintenance, and hygiene were all of great concern as the system routes oxygen and is in a cleanroom. The major safety concerns are sparking hazard, due to the use of solenoids to control high purity oxygen, and electrocution, as we will be working with 110 VAC. Initial concerns of sparking hazard appeared to be of small concern because specialized lubricant free and electrically isolated solenoid valves can be purchased specifically for use with oxygen gas. It is also important to note that PTFE tape cannot be used in manufacture as the small partial pressure of volatile organics in the tape can combust when exposed to pure oxygen. It was also important to consider the materials used in the system. The sponsors expressed a preference for stainless steel wherever possible, Teflon wherever convenient, and bronze wherever necessary. Special care was also put into avoiding dissimilar metal junctions in the presence of wet oxygen. The other major safety concern was high voltage electricity. This danger was dealt with by making it difficult for a user to come into contact with the high voltage sources unless they actively try. To minimize our own exposure, the small amount of high voltage wiring that we had was done by a trained professional. A Failure Mode Effect Analysis (FMEA) was conducted on the system, as viewable in Appendix J, which lists every conceivable failure and its effect on the performance of the system. Using this tool, it was determined that while the system could fail in any number of different ways, the components could be made sufficiently robust to prevent such failure, and there were no foreseen cases in which the system was inherently destined to fail.

The next important consideration was hygiene as this product is to be used in a clean room. Unfortunately, since this is a low-grade clean room, exact specifications on materials and practices are hard to come by and come down to using good judgement in most cases. The primary hygiene concern was that of the wetted surface of the gas routing. Our sponsors asked that this be made

from stainless steel wherever possible, and PTFE tubing everywhere else. A small exception was made for the valves which are made from brass for budgeting purposes. Apart from the wetted surface, the only other requirements were that the materials used were not to overtly kick off particulate matter. Paper, cardboard, and wood were not allowed, and metals and plastics were used instead.

To support to maintenance and repair, the product was built to be capable of complete disassembly. However, maintenance and repair should be infrequent as there are few moving parts in the system.

5.5 Explanation and justification of material selection and part sizing.

Several materials were employed in the system. The most important material selections revolve around the routing tubes. As the gasses are extremely high purity for silicon applications, the routing must also be clean. Traditionally, stainless steel is the material of choice for high purity applications because it does not contaminate the routing gas. Most plastics tend to off-gas and ruin the integrity of the high-purity gas being routed, however, there are several exceptions, including PFA and PTFE. Additionally, the customers expressed a desire for flexible tubing to make quick disconnects and reconnects slightly easier, and to make the furnace more portable for future usage. This would necessitate flexible tubing, which means plastic over stainless steel. This material selection is an important one as stainless steel must be used in key areas. Steel must be used in the routing panel where rigidity is important for the structural integrity of the apparatus, and near the entrance to the furnace as plastics cannot withstand the temperature of the furnaces. In addition, plastic and steel tubing require different fastening mechanisms; stainless steel tubing requires proper swaging valves while PTFE tubing can be fitted with barbed valves coupled with band clamps.

5.6 Prototype cost analysis

As seen below in Table 2, the final design comes in under budget. Most of the expenditure comes from the valves and other routing hardware. It should be noted that this list only contains major online purchases and does not include small local purchases. Therefore, the overall cost will be slightly higher than shown, but because the project is so comfortably under budget, it shouldn't be an issue. This table is also displayed in Appendix H with part numbers and suppliers.

Table 2. Materials List and Cost

Item	Unit Price	Quantity	Total
SSRs	15.99	2	31.98
Acryllic Panel 1/8"x24"x48"	24.14	1	24.14
TC amplifier	17.50	2	35.00
Raspberry Pi 3	35.00	2	70.00
Touchscreen	79.95	1	79.95
Type-K TC	9.95	1	9.95
GPIO Extension	9.99	1	9.99
M3x0.5x20mm threaded standoff	24.00	1	24.00
Teflon Tubing	181.50	1	181.50
Bread Board	9.95	1	9.95
Power supply	23.40	1	23.40
F-F jumper wire	4.95	1	4.95
m-m jumper wire	4.95	1	4.95
Yor-Lok fitting	8.73	15	130.95
T-Junction	36.27	5	181.35
Barbed Fittings	16.01	1	16.01
Band Clamps	6.02	1	6.02
Strut Channel, 6ft	27.27	1	27.27
M3x0.5x10mm screw	8.70	1	8.70
M3x0.5x5mm screw	11.72	1	11.72
M3x0.5x10mm nut	2.12	1	2.12
Valve	47.00	6	282.00
male disconnect	0.35	50	17.5
female disconnect	0.29	50	14.5
ring terminals	0.29	50	14.5
wire (red)	15.95	1	15.95
switches	53	2	106
Blow off	12.28	1	12.28
		Total	1356.63

5.7 Changes after Critical Design Review

After CDR, the project was significantly rescoped to allow for completion in the remaining build time. As software was not a major strength of the project team, progress was slow on the GUI. It was realized that it would be best to direct resources to areas of expertise, and the goal was changed to finishing the manufacture of the hardware and some basic command line functionality for proof of concept. This way, the final software package could be written after the conclusion by a team with more experience to fulfill the sponsor's software requests in full. Therefore, it is critical that the final hardware package be as complete as possible and portable to any additions that need to be made after the fact. The following is an overview of the project package after rescope.

The valving had the highest priority and both the valve panel and valve logic are to be manufactured as previously planned. The control panel is also high priority and was also to be manufactured as planned but with increased modularity. The control panel has two relay boards in case any unforeseen powered componentry needs to be controlled in the future along with the valves (it was later discovered that one of the boards, the solid-state board, is for AC power rather than DC power. It is still useable, but not for any application in the current project). The control panel also has 2 thermocouple amplifiers with SPI inputs to allow for efficient use of the Raspberry Pi ports. The Raspberry Pi also has ample GPIO pins for fluid level switching for the automatic refill of the steam generator. An input is provided to supply power to the both the Raspberry Pi and the power supply and a double pole-double throw switch will be included to increase safety. With the software package, the most important goal is to have a functioning manual mode to control the valves through for proof of concept.

6 Chapter 6 - Manufacturing

The system needs to properly route the fluid lines from the chase panel to their appropriate final destinations —the furnaces and the steam generator. The tubing network proposed, which includes all the valves, tubes, and fittings as it is drawn in Figure 7, will be placed in the 36" by 48" space underneath the furnace. That said, it is possible to break down the manufacturing plan into three distinct phases: manufacturing operations in the shop, component assembly, and integration into the cleanroom. Most components are off-the-shelf parts, including the wires, electronic components, as well as the valves and fittings. For the gas routing panel, these components include the high purity solenoid valves and Yor-Lok fittings that connect those valves to the pipes through our stainless steel and Teflon tubes. The cleanroom has a surplus of stainless steel and PTFE Teflon tubing supply that we used in our system. The electronic components are the Raspberry Pi and its monitor, an AC-to-DC rectifier (power supply), mechanical relays which connect to the solenoids, and a ground node. The electronics are housed inside a 14x18x3.5 acrylic box that is held together by screws and interlocking teeth. A bill of materials is shown in Table 2.

6.1 Plan

This section details the plans developed to build the system. It is divided into sections detailing how to manufacture components, assemble components, and integrate them into the cleanroom.

6.1.1 Manufacturing Operations

This phase of the process describes all operations needed to build components before assembly. There are separate operations for the electronic housing and the gas routing panel.

For the electronic housing:

1. On an 11x17" sheet of paper, all components were placed in their desired locations in relation to one another, marking the locations of the holes where they would be screwed in, as shown in
2. Hole sizes and distances on individual electronic components were measured with calipers, and the distance between them as shown on the paper measured with rulers.

3. The backside of the box was created using a CAD program, shown in Appendix G. The dimensions obtained from the paper in step 2 were used to locate holes on the back side of the box.
4. The remaining five sides of the box were created in Solidworks, with teeth and screws spaced to distribute the load of clamping the box and supporting the self-weight of the box and the weight of the electronic components. This is important as acrylic is brittle and will crack if overloaded.
5. The CAD files of the sides of the box were exported to Adobe Illustrator as vector files.
6. All components were laser cut into ¼" thick acrylic sheets.

For the gas routing panel:

1. Cut the stainless tubes down to size per the dimensions given in Figure 5.
2. Deburr the tube edges using a sanding belt and file to get a smooth and straight finish.

6.1.2 Assembly

In this stage of the process the tubes were cut down to size and bent into their desired shapes, the board was cut down to size, and the parts were ready for assembly.

For the electronics:

1. Components were attached to their places via screws. Printed circuit boards and the Raspberry Pi were mounted on standoffs.
2. Components were wired together according to the schematic in Appendix G.
3. After all the electronics assembled and wired together the side walls and front face of the box were fitted together using their interlocking teeth. The faces were attached using screws and nuts which fit into the t-slots along the edges.

For the routing panel:

1. Tubes were pressed into the Yor-Lok fittings and twisted to tighten.
2. Yor-Lok fittings were screwed onto T-junctions and valves as appropriate.
3. Barbed fittings were screwed onto the sides opposite of valves from the sides now fitted with stainless steel tubes, and onto the T-junction on the oxygen line which routes gas to the bubbler.
4. One valve was fitted with barbed fittings on each side
5. A PTFE tube was routed between the T-junction from step 3 to the valve from step 4. The PTFE tube was attached by simply pressing the tube onto the barbed fitting.

6.1.3 Integration into the Cleanroom

The manufacture and wiring of the system is largely complete, but the system has yet to be installed in the cleanroom. In order to do so, the following must be done:

1. Wipe the system with a clean rag to remove any debris to comply with cleanroom standards.
2. Remove old and excess tubing and wiring from old system from underneath the furnace to make room for the routing panel.

3. Insert the routing panel on the tray beneath the furnace and mount the electronics box to the brackets on the side of the furnace.
4. Connect the valves to the relays, and the tubes to the corresponding fluid lines in order to close the system loop.

6.2 Procurement and Build Activities

For testing purposes, a Raspberry Pi, a valve, a relay, and LED were acquired. When it was determined that the Raspberry Pi could indeed turn the LED on and off and control a valve, the design was finalized with these components. Simultaneously, the group began working together to explore the creation of a user interface.
User interfaces were designed in QtCreator, and several weeks were dedicated to programming the system to transition between pages and communicate with the program operating the furnace.

Upon creation of the manufacturing plan for CDR, the specified components were ordered, and the team adjourned for summer holidays. Upon reconvening, it was discovered the acrylic panel, power supply, solid state relays, screws, standoffs, Yor-Lok fittings, barbed fittings, T-junctions, and thermocouple readers had arrived, but the required valves were not ordered, and a new order was sent out immediately to rectify this. While waiting for the valves, the stainless steel tubes were cut down to appropriate sizes, and the acrylic board was quickly fitted with holes to mount the electronics. A mounting bracket for the tubing system was created out of Unistrut.

In the meantime, several weeks of working yielded painfully slow progress on the graphical user interface. Due to the sheer number and magnitude of the challenges that would be needed to overcome to deliver the graphical user interface, and the fact that this was a non-essential part of the project contributed to the decision to terminate work on the graphical user interface and redirect efforts instead to the electronic and mechanical portions of the project. The designs for the interface are viewable in Appendix M, but no further coding has been done on this front.

Upon the arrival of the valves, the tubing system was assembled. At this time, the system was wired together and switched on for the first time. As the assembled electronics board was determined to be generally good, but contained some awkward component orientation, unsafe exposed terminals, and was too flimsy, it was determined that a second iteration would be necessary.

Simultaneously, code was written to read gases and temperatures from a .txt file and import this as a table of gases to be run as a regimen assuming thermocouples could be read in the future. It was discovered then that the MAX31856 thermocouple reader chips ordered were designed to port with Arduino chips and had reading libraries written in C++, not Python. An open-source Python library was found on GitHub to read the MAX31856, and it was decided to try to implement these. First attempts were made to load thermocouple-reading libraries onto the Raspberry Pi, but since the Raspberry Pi 2 in question had no internet capabilities (the Raspberry Pi that did was destroyed during a testing mishap over summer), it was impossible to do so.

Over the course of a week, plans for the electronic housing box were drawn up in SolidWorks using the method described above. These were designed to have interlocking teeth and t-shaped slots to allow the sides to fit together and attach using screws and nuts, along with slots for wires

to enter and exit and holes to allow for the circulation of air to cool the power supply. After the whole was designed and determined to be ready for manufacture, they were cut on the laser cutter. The electronics were transferred the board, and the walls were attached together.

At this time, a method was written to provide for manual operation of the furnace which simply paused the program until the user input a gas, then switched valves to the appropriate combination. As this was to determine system functionality only and did not rely on temperature readings, it was suitable for testing the functionality of the ability of the system to route gas, not determine the safety of the system. Running the code on the assembled electronic assembly confirmed that the code was capable of properly controlling the system. However, when valves were added, they could not be reliably opened. Consultation with the instructor and some testing revealed that the problem resided in the fact that the solid-state relay used was for AC power, not DC, causing it to work improperly. Therefore, the valves were all switched to the mechanical relay, and the solid-state relay was relegated to do nothing on the board unless an AC purpose can be found for it.

One final attempt was made to operate thermocouples. A new Raspberry Pi 3 was acquired, and the thermocouple reading libraries were installed. However, the serial port communication could not be made to operate properly for any of the attempts made, so the decision was made to continue without relying on thermocouple readings as time was short.

Shortly before the project exposition, the electronics and valves were wired together, and the system was tested on the valves. The test revealed that the system was indeed capable of taking a user input and controlling valves. As of this writing, the system possesses the hardware capabilities to take temperature readings, but lacks the code required to do so. The system can also support the addition of hardware and software components to operate pressure gages and a bubbler refill system, elements which were originally called for, but which could not be implemented within the required timeframe.

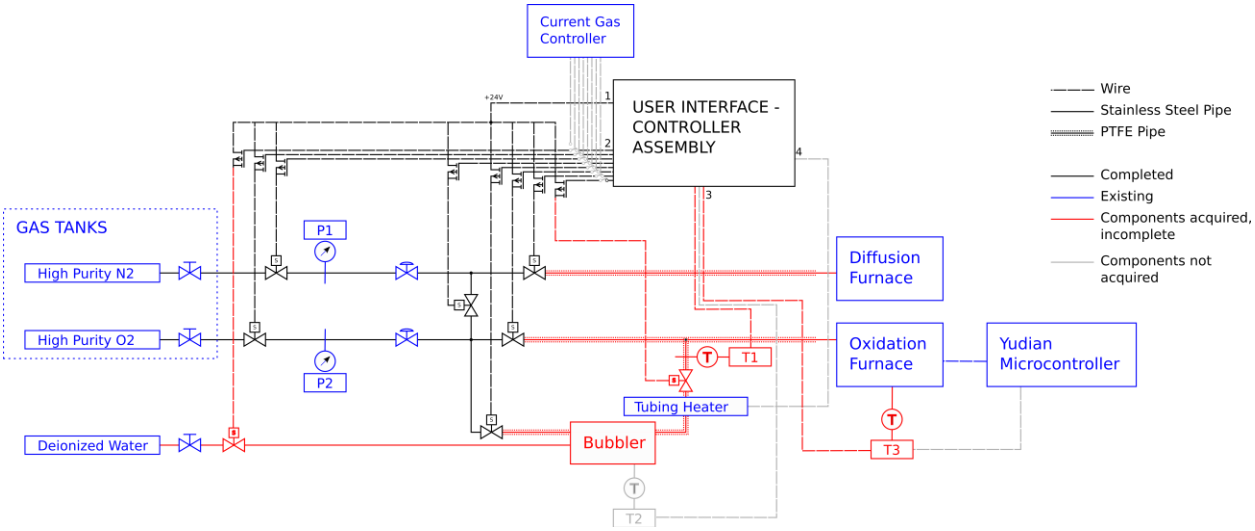


Figure 11. System as built upon termination of project. Black components represent components completed by this project, blue components represent existing components that will not be replaced, red components represent components for which materials are purchased but not implemented, and gray components represent components for which no materials have been acquired.

7 Design Verification

The Design Verification Plan is the method by which the system was tested to ensure it met the engineering requirements set forth in the SOW. This is documented in the Design Verification Table and the subsequent testing descriptions.

7.1 Design Verification Plan (DVP) Table

The testing excerpt from the Design Verification Plan (DVP) is shown in the table below. The entire table is viewable in Appendix G. In order to ensure the system met the criteria specified here, tests as enumerated in the following section were performed on the system. It should be noted that since the thermocouple readers were not made operational and pressure gages were not installed, many of the originally required functions could not be implemented. Therefore, the DVP table has many entries labeled "NA" or "-."

7.2 Tests

1. In order to determine oxygen line safety, all oxygen lines were thoroughly examined for any points where they may be punctured, overheated, sparked, or otherwise damaged. This was conducted when the routing panel was assembled and should be conducted when the whole system is installed in the furnace.
2. The components were assembled, and the subsystems were measured with a tape measure to verify they stayed within set boundaries. The final test of dimensions should be done when the system is installed in the furnace system.
3. The number of control panels on the whole assembly necessary to control the furnace were counted to verify that there were two or fewer.
4. Voltage measurements were taken at the power source of the electronic systems, as well as at all other potentially high voltage points in the system. The system is considered safe when the maximum voltage at any point which may come into direct contact with a user does not exceed 40 Volts.
5. The gas programming system was tested on the control system level by measuring pin outputs. The system was tested again by connecting the relays and valves, entering inputs into the controller and observing valve response. The system was then connected to the tubing board and tested without gas flow. Finally, this test should be conducted a third time with gas flow when the system is installed in the cleanroom.
6. The unsafe condition detection was not performed as the thermocouple reader chips could not be made to communicate with the Raspberry Pi. Unsafe condition detection should be tested first on the system level by simulating an unsafe condition by altering the signals arriving at the controller from the rest of the system. The control system should then be installed on the furnace, and the temperature and pressure raised slightly above safe levels to observe the system response.
7. The thermocouples, again not tested since the reader chips were not in operation soon enough, should be tested upon being rendered functional by measuring bodies at a known temperature and comparing the readouts to the actual values.
8. The pressure gages were not acquired as priorities were directed to other components of the system. When they are installed, they should be tested by measuring a body at a known pressure and comparing the readouts to the actual values.

9. A cost analysis has been performed on the system.
10. The insufficient gas notification can only be tested after the installation of pressure gages. It should be tested by simulating a low pressure and observing the controller response. This should be repeated after installation.
11. Bubbler refill was not implemented but should be tested by simulating the conditions of low water, and measuring the output of the pin which controls the deionized water valve. This should be repeated after adding the valve and relay, and at the assembly level after installation.
12. A count for pneumatic valves was be undertaken at the end of assembly to verify that there were none in the system.

Table 3. Testing excerpt from the Design Verification Plan

Item No	**Specification #**	**Test Description**	**Acceptance Criteria**
1	Oxygen Line Safety	Examination	No risk of line puncture or ignition
2	Size	Measurement	Routing component fits within a box of size 36" x 48" x 15", control panel fits within box of 36” x 26” x 12”
3	Number of control panels	Examination	Two or fewer
4	Voltage	Measurement	No voltages greater than 40V
5	Gas control	Testing	Success
6	Unsafe condition detection	Unsafe Condition Simulation	No Fail
7	Temperature resolution	Measurement	Temperature reading accurate to within 10ºF
8	Pressure resolution	Measurement	Pressure reading accurate to within 1 kPa
9	Cost	Cost Analysis	Total cost less than $5000
10	Notifies user of insufficient gas	Simulate insufficient gas conditions	No Fail
11	Automated bubbler refill	Testing	Success
12	Number of pneumatic valves	Examination	No pneumatic valves

7.3 Results

Since many of the originally desired components were not acquired or made functional within the allotted time span, many of the originally required specifications were not implemented. Therefore, the system failed to meet several originally required specifications. However, the system does meet all the specifications possible under the circumstances.

The results of testing were as follows:

1. The oxygen lines were examined and determined to be safe. One temporary exception was a Teflon tube line which was bent, but this will be remedied by replacing the line and permanently attaching the valves to their final locations. This test should be conducted again when the whole system is installed in the furnace.
2. The assembled components fit easily within the required dimensions. The tubing system would fit easily underneath the furnace, and the control panel would easily occupy a smaller space than the current control panel.
3. There is only one control panel in our system. Therefore, the system passes.
4. The system has one place where 120VAC wall voltage is exposed, but this is sealed within the electronic housing box far from any location into which prying fingers can reach. It is possible, however, to insert thin metal rod-like objects into the housing. Users are specifically advised against this action in the operator's manual (see Appendix M), as it could result in electric shock.
5. The gas programming system was determined to be operational for manual control of gasses, wherein a user specifies a gas to be routed to a furnace, and the system routes that gas to the appropriate furnace. Since thermocouples are as of yet inoperable, automatic gas switching was not implemented. Therefore, this requirement is fulfilled, although not to the extent originally planned.
6. Unsafe condition detection, which requires pressure gages and thermocouples, was not implemented and therefore this design criteria is unfulfilled.
7. Thermocouple amplifiers are inoperable. Therefore, their resolution is untested.
8. Pressure gages were not acquired. Therefore, the system leaves this requirement unfulfilled.
9. Total expenditures amounted to $1356.63, far under the $5000 limit established. Therefore, this requirement is fulfilled
10. The system is unable to detect insufficient gasses since pressure gages were not acquired. Therefore, the system does not meet this requirement
11. Bubbler refill was not implemented. Therefore, the system does not meet this requirement.
12. The system contains no pneumatic valve, therefore meeting the requirement.

It was generally agreed toward the end of the project that the original specifications which were thought feasible at the beginning of the project were too ambitious and too many. Lack of understanding of the electronics and control systems required to build the project contributed to this optimism and precluded any reservations that may have existed at the time. It was only after many months and much learning that the sheer magnitude of the project became clear, and it was by this time too late to rescope the project.

8 Project Management

The design process was in this process. The design process began with problem definition, research, and customer interviews. From this, a knowledge of the underlying principles was developed and an understanding of the type of system that would be needed was determined. Through research and customer interviews, a set of customer needs engineering specifications was developed, which were then fed into a QFD chart to give an understanding of which problems are most important to solve. These specifications were the culmination the Scope of Work (SOW).

After problem definition and SOW came the conceptualization phase. The first segment of conceptualization was ideation, which consisted of brainstorming and filtration to screen useful ideas to be employed in our design. Afterwards, design ideas were correlated to the specifications determined in the SOW. A few rough block diagrams of idealized solutions were created, then the best traits of all of these were combined to compile one final selected design. Some of the systems in place in the conceptual design will be relatively easy to accomplish and some will be more difficult. The systems proposed in the concept design were compartmentalized and ranked on their difficulty, which determined whether or not the subsystem was feasible through decision matrix analysis. For the remaining subsystems, the relative difficulty of the system determined how much work needs to be invested in each. With this information the individual subsystems were designed, and a computational model was created to prototype the subsystems in order to validate the prototype designs. This work was the subject of the Preliminary Design Review (PDR).

After PDR, a Failure Modes Analysis was conducted, and unclarified details from PDR were solidified for the intermediate design review (IDR). A prototype was designed for the critical design review (CDR), and design and manufacturing plans were solidified.

After CDR, parts were ordered, and fabrication and testing continued until the conclusion of the project. At the end of the quarter came a manufacturing and test review which was a summary of the progress in the way of testing and manufacture.

Fall Quarter 2018 was the last quarter of the design process. The manufacture was progressed as far as possible, the operating manual developed, a hardware demonstration performed, and lastly, a final design review (FDR) and senior project exposition given. Especially toward the end of the project, it became clear that in spite of best efforts in the design, not everything worked as planned, and the planning had not anticipated all necessary components. For this reason, not all aspects of the project were completed as planned. A timeline of the project plan is laid out in a Gantt chart included in Appendix J.

9 Conclusion and Recommendations

Background research and customer interviews conducted for the SOW revealed that the cleanroom furnace system is unnecessarily complex and tedious to control. In order to solve this, it was decided to simplify the furnace system in the cleanroom at Cal Poly and to add a more intuitive control system to simplify the operation of the furnace during experiments. A set of engineering

specifications were generated from this problem statement in order to evaluate the completion of the project at a later date.

The concept design development took these goals and developed a solution to address them. A solution consisting of a gas delivery system, a user interface to control the system, and a control system to connect the two was devised. The design and manufacturing plan was completed, and the manufacturing began.

Six months of design and one month building revealed that more electrical engineering and programming knowledge than possessed by the team was required to complete the project as planned. The project was far more software and electronics-oriented than originally anticipated, and as such the group possessed insufficient knowledge to make judgements as to whether requirements were feasible. As a result, results were overpromised and underdelivered.

That being said, the project is incomplete, not a failure. Significant progress was made in areas where mechanical knowledge was required. The valve tubing system is built and only needs to be installed in the system. The electronics housing was designed and built, and most of the hardware is already acquired. A future team consisting of a mechanical engineer to design the bubbler refill and find appropriate pressure transducers, an electrical engineer to design the hardware required to connect these to the Raspberry Pi, and a computer science student to program the user interface should be able to complete the objectives left unfulfilled by this senior project using the remaining funds.

We submit this document for sponsor review. Please inform us if the contents of this document are satisfactory to you so that mistakes and misunderstandings can be eliminated.

References

Arduino. (2018, 1 18). *What is Arduino?* Retrieved from Arduino: https://www.arduino.cc/en/Guide/Introduction

Barrett, S. F. (2012). *Arduino Microcontroller: Processing for Everyone! Second Edition.* Laramie, WY: Morgan & Claypool.

Cvjetkovic, V. M., & Matijevic, M. (2016). Overview of Architectures with Arduino Boards as Building Blocks for Data Acquisition and Control Systems. *International Journal of Online Engineering, 12*(7). Retrieved 1 18, 2018

Deczky, A. E. (1981, October 13). *USA Patent No. US4294682 A.*

Fuchslocher, C. F. (2014). *European Union Patent No. EP 3091499.*

Gale, B. K. (n.d.). Fundamentals of Micromachining. Utah, USA.

HIROSHI, K., SHIGERU, O., & HIKARI, M. (1997). *Japan Patent No. 19970105494.*

M.G. Guvench, R. S. (1997). *Programmable PID Temperature Control of Multi-Tube Multi-Zone Diffusion Furnace.* Portland, Maine.

May, G. S., & Spanos, C. J. (2006). *Fundamentals of Semiconductor Manfucaturing and Process Control.* Hoboken, New Jersey: John Wiley & Sons, Inc. .

National Instruments. (2018). *What is Data Acquisition?* (National Instruments) Retrieved 1 22, 2018, from National Instruments: http://www.ni.com/data-acquisition/what-is/

Samsung. (2015, May 27). *Eight Major Steps to Semiconductor Fabrication, Part 6: The Addition of Electrical Properties*. Retrieved from Samsung Newsroom: https://news.samsung.com/global/eight-major-steps-to-semiconductor-fabrication-part-6-the-addition-of-electrical-properties

Wikipedia. (2018, May 3). *Solenoid Valve*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Solenoid_valve

Wolf, S. (2004). *Microchip Manufacturing.* Sunset Beach, California: Lattice Press.

Appendices

Appendix A. QFD House of Quality
Appendix B. Decision Matrices
Appendix C. Benchmarking test results
Appendix D. Furnace block diagram
Appendix E. Concept Layout Drawings
Appendix F. Drawings Package
Appendix G. Budget and procurement list
Appendix H. Design verification plan and report
Appendix I. Failure Mode Effects Analysis
Appendix J. Design Hazard Checklist
Appendix K. Gantt Chart

Appendix A. Technical Literature Review

9.1 Technical: Technical literature review

Some technical research was done to examine the larger context of the system and potential products to be used in the design of the final product. These are discussed below.

9.1.1 Silicon Wafer Fabrication

Silicon circuit elements are ubiquitous in most modern electronics and are produced by processing silicon wafers through a process to infuse them with the necessary structures to produce the desired properties. The stages in this process (the order of the stages depends on the desired elements) are detailed below.

9.1.1.1 Oxidation Theory

The first part of the microfabrication process is oxidation. There are several oxidation methods including thermal oxidation, electrochemical anodic oxidation, and plasma-enhanced chemical vapor deposition (PECVD). Thermal oxidation at high temperatures is the most widely used method, and is the process adopted in the Microfabrications laboratory at Cal Poly. Thermal oxidation forces an oxidizing agent (oxygen gas or water vapor) to diffuse into the silicon wafer surface at high temperatures (900 - 1200 °C) and chemically react with it to produce a thin smooth layer of SiO_2 (May & Spanos, 2006). The thermal oxidation apparatus consists of a resistance heated furnace, a cylindrical fused quartz tube containing the silicon wafers, and a source of pure oxygen or water vapor, shown in Figure 12.

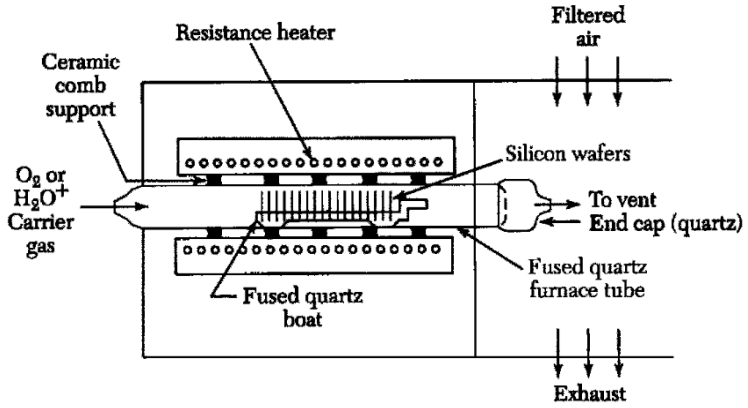


Figure 12. Thermal oxidation furnace apparatus. From (May & Spanos, 2006).

The oxidation could either be wet (oxidized in water vapor), or dry (oxidized in O_2 gas) and in both cases the reactions occur at the Si/SiO_2 interface. Silicon from the wafer is consumed in the reactions, and the amount consumed is 44% of the final-oxide-thickness. For IC applications the SiO_2 film thickness ranges from 3.0-400 nm, with wet oxidation producing thicker oxides than dry oxidation. The Deal-Grove model (or the linear-parabolic model) provides some insight on the mechanisms by which oxide-films grow (works for thicknesses of 30 nm and greater). It predicts the oxide thickness for a given set of oxidation conditions, and can be written in the following form:

$$\left(\frac{x_{ox}^2}{B}\right) + \left(\frac{x_{ox}}{B/A}\right) = t + \tau$$

where t is the time of the oxidation step, B and B/A are the parabolic and linear rate constants, respectively, and are a function of oxidation temperature and whether the oxidation is dry or wet, and τ is a factor accounting for any oxide previously present. In addition to oxidation temperature, and whether wet or dry oxidation is being performed, other factors impacting B and B/A (and consequently the oxide growth rate) include the doping-concentration at the surface of the silicon wafer, the oxidant-pressure during the growth process, and the presence of chlorine in the oxidizing ambient (Wolf, 2004).

9.1.1.2 Impurity Doping and Diffusion Theory

Impurity doping gives the semiconductor its electrical properties. There are two methods for doping: diffusion and ion implantation, and they both can complement each other. Ion implantation is a more recent technique (developed in the 1970s) where dopants are implanted into the wafers by an ion beam. Compared with diffusion methods it has more precise control and doping reproducibility, and the process parameters (depth, concentration) are specified directly in the equipment settings for the implant dose and energy. Consequently, ion implanters have been more widely utilized in industry and are sometimes even necessary to fabricate some of today's integrated circuits. However, one of the downsides of this method is that it is significantly costlier than the diffusion method, meaning that most academic institutions will not have the budget to acquire an ion implanter which could cost up to several million dollars. The cleanroom at Cal Poly, instead, utilizes a diffusion furnace for its doping applications.

Diffusion is either done by deposition of the gas phase of the dopant or by doped-oxide sources. The wafers are placed in a high temperature quartz-tube furnace and a gas mixture that contains the desired dopant is passed through it, very similar to the oxidation furnace setup. The dopants themselves can be introduced through solid sources, liquid sources, or gaseous sources. For diffusion in silicon, boron is used to create p-type junctions, and arsenic and phosphorous are used for n-type. In the diffusion process the doping concentration is maximum at the surface and decreases monotonically the deeper we go inside the wafer, as opposed to ion implantation where the peak distribution occurs inside the semiconductor.

Diffusion occurs in two stages: a pre-deposition step and a drive-in (or redistribution) step. In pre-deposition, an oxide layer containing the dopant is formed on the silicon surface by flowing the dopant containing gas, oxygen, and a carrier gas into the furnace tube containing the wafers. At high temperature and for a short amount of time, impurities enter the top surface of the silicon. The pre-deposition step is kept short to limit the distance that dopants travel inside the silicon wafer. This step determines the *amount* of impurities present inside the silicon but not their final distribution. After the pre-deposition is complete, the dopant-oxide-layer is removed from the silicon so as not to add any additional impurities in the subsequent step. A longer, higher temperature process (drive-in) is then conducted to drive the dopants to their desired final locations. So pre-deposition controls the amount of dopants to be added, and drive-in determines their final locations.

The main driving force for diffusion is the concentration gradient of the impurity, and Fick's Laws of Diffusion formulated in 1855 provided mathematical tools that can be used to determine the diffusion profile at any given time throughout the process (i.e. C(x,t)). Fick's first law states that the flow of impurity particles (or particle flux, F), going from higher to lower concentration, is directly proportional to the concentration gradient. The proportionality constant between the flux and the concentration gradient is also known as the diffusion-coefficient, D, which is exponentially related to temperature$\left(D = D_0 \exp\left(\frac{-E_a}{kT}\right)\right)$. Fick's second law takes the form of a second order

partial differential equation with respect to position, and a first order partial differential equation with respect to time, and therefore requires two boundary conditions and one initial condition to solve. There are two cases to consider in solving these equations, one for pre-deposition and another for redistribution. In both cases the concentration very deep inside the silicon and away from the surface is zero ($C(\infty,t) = 0$), and in both cases the initial concentration inside the silicon wafer, is either known (in the case of pre-dep) or assumed (in the case of drive-in) to be zero ($C(x,0) = 0$). The assumption for the drive-in case is valid because all the impurities are located within a very thin region beneath the surface after pre-deposition compared to their final distribution. This leaves us with the second boundary condition to solve Fick's second law equation; in pre-deposition the dopant-surface-concentration value remains constant ($C(0,t) = C_s$) at its solid-solubility-value (i.e. the maximum amount of solute that can be held per unit volume) and in redistribution the total amount of dopant remains constant throughout. The solution to these equations are as follows:

- For pre-deposition, the solution takes the form $C(x,t) = C_s erfc(\frac{x}{2\sqrt{Dt}})$, where erfc is a mathematical function known as the complementary-error-function, and the term in the denominator, $\sqrt{Dt}$ is referred to as the diffusion length.
- For drive-in the solution takes the form $C(x,t) = \frac{Q_0}{\sqrt{\pi Dt}} \exp\left(\frac{-x^2}{4Dt}\right)$, where Q_0 is the total amount of dopants present (from the pre-deposition phase), and $\frac{Q_0}{\sqrt{\pi Dt}}$ is the surface concentration which decreases over time.

What these solutions show is the importance of controlling the *temperature* and *time* to in doping operations obtain the desired dopant profile.

9.1.2 Data Acquisition Systems and Arduino Technology

Data acquisition (DAQ) is the process of making physical measurements using a computer (National Instruments, 2018). Measurable quantities include voltage, temperature, or sound. DAQ measurement hardware, sensors, and a computer with programmable software are the components in a DAQ system. There are many ways to accomplish this; traditionally, the different elements are connected together with wires transmitting the data to each other, as is the case with the DAQs used in mechanical engineering laboratories at California Polytechnic State University. However, other methods exist, such as a patented system wherein the sensors in an aluminum smelting plant transmit data to the computer using infrared signals since transmitting wires or radio were infeasible (USA Patent No. US4294682 A, 1981). This general flexibility makes them adaptable to many operating systems and configurations, and they can now be constructed easily even by those without computing background with the advent of Arduinos.

Arduino is a microcontroller described as “an open source electronics platform” (Arduino, 2018). Utilizing user-friendly hardware and software, Arduino was developed with easy-to-use hardware and software to allow processing power to be readily available to everyone (Barrett, 2012). Arduino boards come in standard configurations, and can be modified with the addition of shield boards, which allow the Arduino board to control many different systems ranging from 3D printer servo motors to light sensors. Manipulating the information on the individual pins allows for more close user control of the information, but the Arduino language allows for higher-level coding, making for a more user-friendly experience.

Cvjetkovic and Matijevic of the University of Kragujevac have written extensively on the ability of Arduino to function in a DAQ architecture. By examining the various programming modes and

interfaces available to Arduino users, they have developed architectures for the application of Arduino to DAQ purposes. With the correct programming, it is even possible to deploy a large number or Arduino-connected sensors to acquire a wide swath of data distributed over a large area wirelessly (Cvjetkovic & Matijevic, 2016). Such data acquisition capabilities are powerful in areas where disparate wireless measurements are required.

Arduino-based DAQ systems will be a good option for prototyping DAQ systems for our project. While it may not be possible to reach the level of computing sophistication postulated by Cvjetkovic and Matijevic, it should be possible to use basic Arduino-based DAQ systems to simulate a rudimentary DAQ model for a final product.

9.1.3 Solenoids, Mass Flow Meters, and Pressure Transducers

Solenoid valves are finite-way valves that are operated electronically. Most commonly-used solenoid valves are 2 position valves (can only be fully open or fully closed) and can either be made in a normally open configuration or normally closed configuration. When the valve is not in its "normal" position, the valve will consume power, typically 5-15 watts. Valves are commonly brass or cast iron for use with air, mild liquids, and inert gasses. Prices range from 15 to 80 USD. Valves don't usually need anything except for a supply voltage meaning the valve requires a digital controller to operate. Valves suitable for our application (pressurized high purity oxygen) take wall voltage (120VAC), meaning that a controller is not sufficient to control the valve. A secondary high-voltage circuit will need to be designed with a MOSFET or relay intermediary. Solenoid valves work by putting a coil and magnetic shuttle on top of the piston in a plunger type valve. Diagram of a solenoid valve shown below in Figure 13. These will likely be used in abundance as the customer is looking to eliminate the pneumatic actuation in the current system. Special solenoids can be purchased to safely interface with a pure oxygen environment.

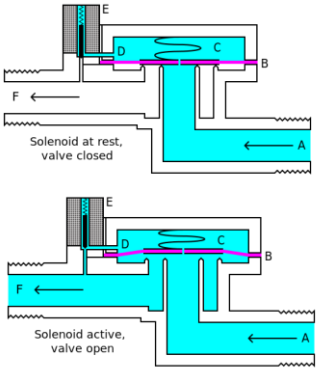


Figure 13. Basic function of a solenoid actuated valve *(Wikipedia, 2018)*. Solenoid controls a small port which then induces fluid feedback to a metallic diaphragm allowing a relatively weak solenoid to control a relatively high pressure fluid line. This allows the solenoid to draw much less power than it would otherwise have to.

Servo valves serve a similar function to solenoid valves but are continuously variable in their degree of openness. Servo valves cost roughly the same amount as solenoid valves but require more sophisticated control mechanisms than solenoid valves. They can be used to regulate flow rate variably rather than in a binary manner. They can be run on low supply voltage like solenoid valves. They do not require current when fully open or fully closed but do require current when they are in an intermediate state. These can be used in conjunction with flow meters in a control loop to regulate mass flow rate of gas flow. Like solenoid valves, they are usually designed for use with inert gasses and it is difficult to source servo valves for pure oxygen.

Electronic flow meters are one of the more expensive components we might need to incorporate. They typically range in price from 400-1500 USD. Flow meters come pre-calibrated and often have internal conversion factors for a range of gasses. Flow meters usually give analog signal from 1-5 volts which will port well with any controller or microprocessor. They can be used in conjunction with servo valves to create control loop to regulate flow rate. Flow meters frequently contain no motors, active elements, or reactive materials aside from anodized aluminum.

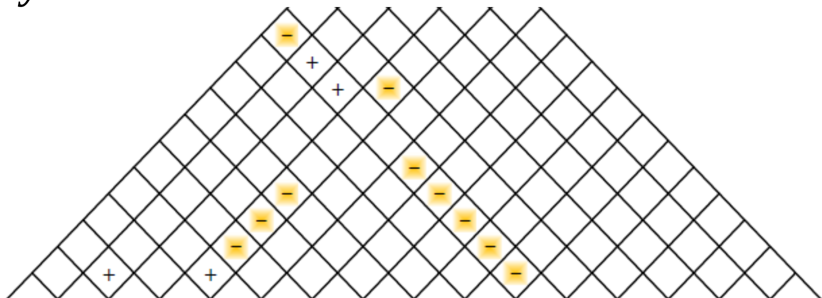
Pressure transducers are electronic sensors that measure gauge pressure or differential pressure. They work by applying the line pressure to a diaphragm with a strain gauge on it. This strain gauge is attached to an internal Wheatstone bridge with internal temperature correction. Pressure transducers need an excitation from a microcontroller or DAQ and requires 2 more wires to measure strain voltage. Transducers can range in their operating pressure range from 0-100 to 0-10,000 PSI and go up in price from low to high voltage. They are inert, usually made from stainless steel, and low voltage so there should be no issues with pressure transducers around oxygen. For our purposes, pressure transducers would most likely be used around the gas tanks themselves meaning they need an operating pressure around 500 PSI. At this pressure, transducers range in price from 70 to 150 USD.

Appendix B. QFD House of Quality

Relationships	
Strong	●
Moderate	○
Weak	▽

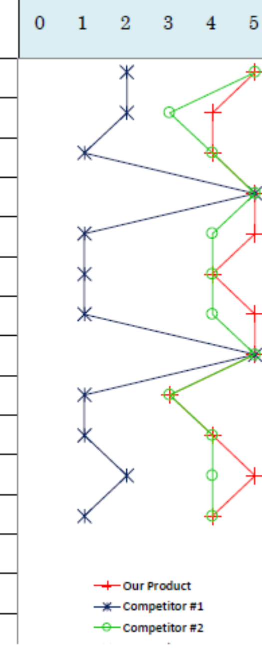
Correlations	
Positive	+
Negative	–
No Correlation	

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



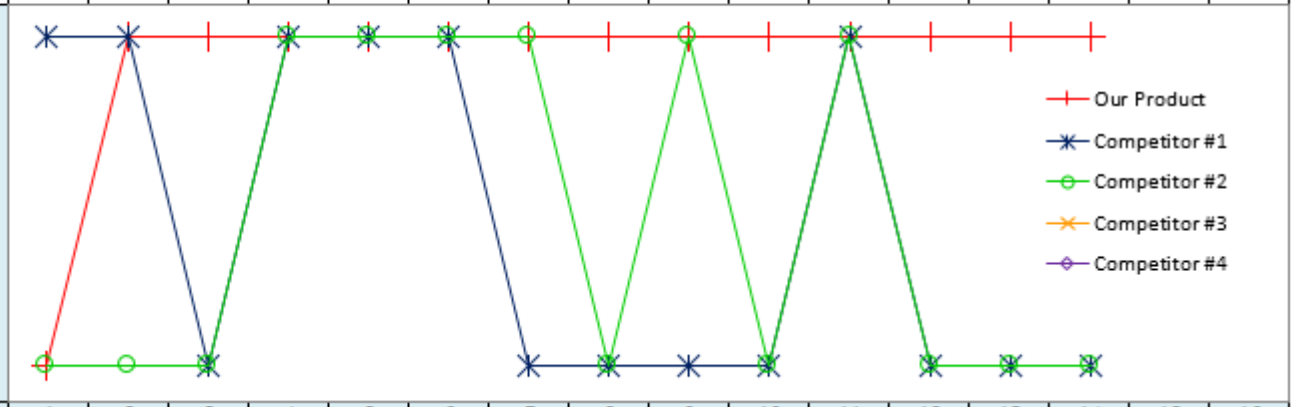
Row #	Weight Chart	Relative Weight	Clean Room Instructors	Students			Maximum Relationship	WHAT: Customer Requirements (explicit & implicit) / HOW: Engineering Specifications	1 No electricity controlling oxyg	2 Dimensions	3 Number of Control Panels	4 Power Consumption	5 Voltage	6 Temperature Setting Program	7 Gas Regimen Programming	8 Unsafe Condition Detection	9 Temperature Resolution	10 Pressure Measurement Error	11 Cost	12 Notifies user of insufficient ga	13 Automated Bubbler refill	14 Number of pneumatic valves	15	16	Our Current Product	Competitor #1: Current Syste	Competitor #2: Southern Maine			Row #
								Direction of Improvement	▼	▼	▼	▼	▼	◇	◇	◇	▼	▼	▼	◇	◇	▼								
1		10%	8	8			9	Programmable						●	●										5	2	5			1
2		6%	5	5			9	Simple		▽	●														4	2	3			2
3		9%	5	10			3	Intuitive			○														4	1	4			3
4		12%	10	10			9	Fits within same space		●															5	5	5			4
5		6%	5	5			3	Centralized Controller			○														5	1	4			5
6		12%	10	10			3	Procedural fail safes								▽				○					4	1	4			6
7		6%	10	0			9	Budget											●						5	1	4			7
8		9%	7	8			3	Automated Support Process							○						○				5	5	5			8
9		3%	6	0			3	Portable		○	○														3	1	3			9
10		12%	10	10			9	Accurate							▽	▽	●	●							4	1	4			10
11		2%	4	0			9	Gets Rid of pneumatics	▽													●			5	2	4			11
12		12%	10	10			9	Safe	●	▽		●	●			●	▽	▽							4	1	4			12
13		0%																												13
14		0%																												14
15		0%																												15

NOW: Current Product Assessment - Customer Requirements



Current Product Assesment - Engineering Specifications

HOW MUCH: Target	Yes/No	x, y, z dimensions	<=2	Max Power	Max Voltage	Yes/No	Yes/No	Yes/No	<=10 °C	<= 1 kPa	<= $5000	Yes/No	Yes/No	0		
Max Relationship	9	9	9	9	9	9	9	9	9	9	9	3	3	9		
Technical Importance Rating	111	137	111	109	109	87.4	127	133	121	121	50	36.4	27.5	20		
Relative Weight	9%	11%	9%	8%	8%	7%	10%	10%	9%	9%	4%	3%	2%	2%		
Weight Chart																
Our Product	0	5	5	5	5	5	5	5	5	5	5	5	5	5		
Competitor #1: *Current System*	5	5	0	5	5	5	0	0	0	0	5	0	0	0		
of Southern Maine PID Tem. Controller	0	0	0	5	5	5	5	0	5	0	5	0	0	0		
Competitor #3: *Product Name*																
Competitor #4: *Product Name*																
Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



Appendix C. Decision Matrices

Table 4. Valve selection decision matrix

	Baseline: Current valve system	Weight	Solenoid valves	Oxy-friendly solenoid valves	Hand-operated gate valves
Criteria					
Routes gas to the appropriate location	0	2	0	0	0
Simple	0	1	1	1	1
User-friendly	0	1	0	0	-2
Safe	0	inf	-2	0	0
Scores			Disqualified	1	-1

Table 5. Controller selection decision matrix

	Baseline: Current system	Weight	Ready-made controller	PC with microcontrollers
Criteria				
Allows temperature regimen programming	0	2	0	0
Allows gas regimen programming	0	2	1	1
Dynamic switching	0	1	0	1
Difficulty of implementation	0	2	-1	-2
Scores			0	-1

Table 6. User interface selection decision matrix

	Baseline: Current control panels	Weight	Consolidated control panel	Labview	Terminal window	Customized GUI
Criteria						
Intuitive	0	1	1	2	-2	2
Simple	0	1	1	2	-1	2
User-friendly	0	2	0	1	-2	2
Ease of installation	0	3	0	-1	-1	-2
Improveability	0	1	0	0	2	1
Safe	0	inf	0	0	0	0
Scores			2	3	-8	3

Appendix D. Benchmarking test results

Below are the benchmarking results extracted from the QFD table. A binary scoring system was used: a maximum score of 5 was given if the specification that was met, and 0 to indicate that the specification has not been met.

Table 7. Benchmarking table

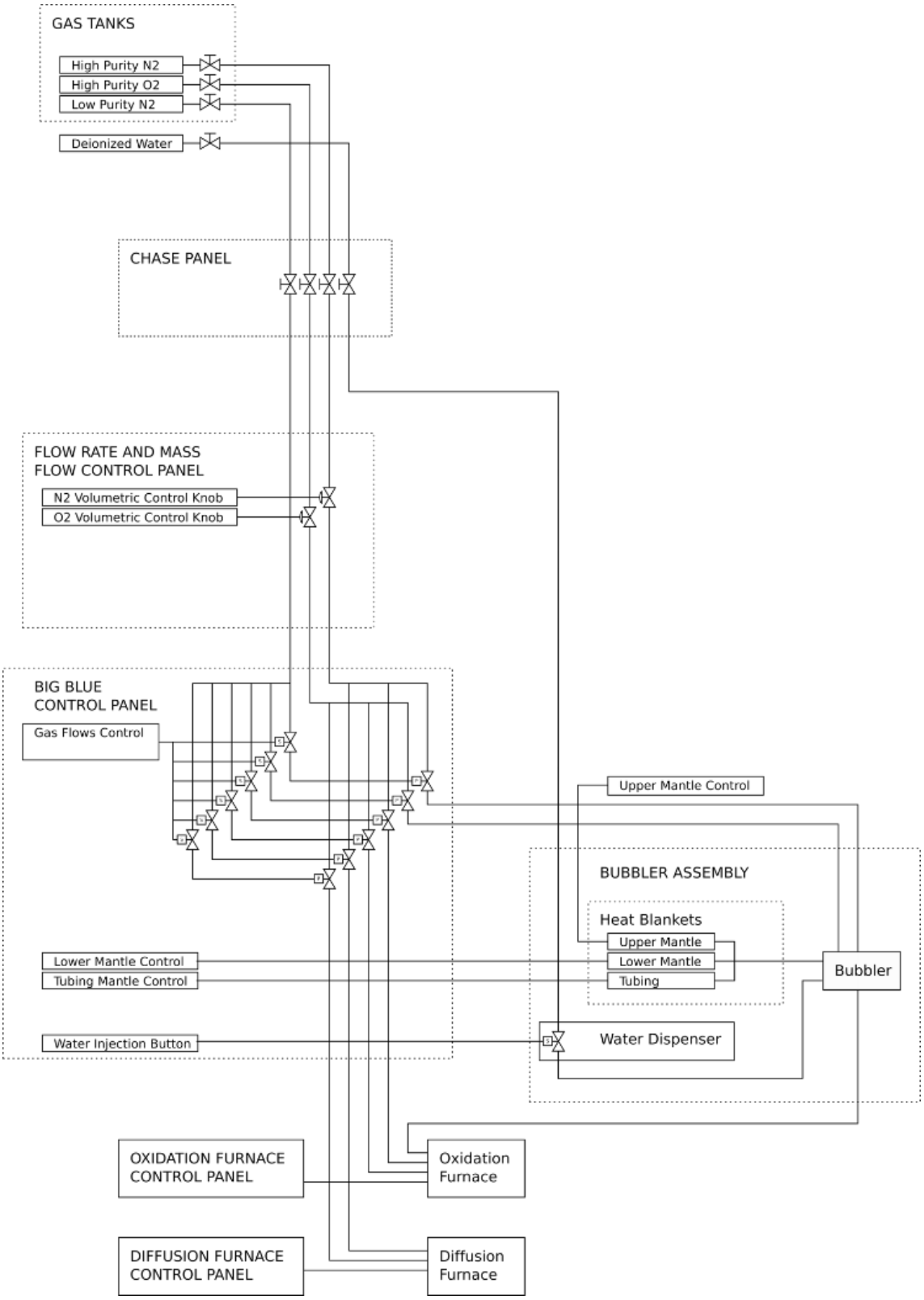
HOW: Engineering Specification	HOW MUCH: Target	Our Product	Competitor #1: *Current System*	Competitor #2: *Univ. of Southern Maine PID Temp. Controller*
No electricity controlling oxygen lines	Yes/No	0	5	0
Dimensions	36" x 48" x 15"	5	5	0
Number of Control Panels	<=2	5	0	0
Voltage	120 V AC	5	5	5
Temperature Setting Program	Yes/No	5	5	5
Gas Regimen Programming	Yes/No	5	0	5
Unsafe Condition Detection	Yes/No	5	0	0
Temperature Resolution	<=10 °C	5	0	5
Pressure Measurement Error	<= 1 kPa	5	0	0
Cost	<= $5000	5	5	5
Notifies user of insufficient gas	Yes/No	5	0	0
Automated Bubbler refill	Yes/No	5	0	0
Number of pneumatic valves	0	5	0	0

Appendix E. Furnace block diagram

The furnace block diagram on the following page describes the layout of the furnace system and its associated gas delivery flow as it currently exists.

High purity oxygen and nitrogen flow from gas tanks through a set of hand operated gate valves on top of the tanks through another set of hand operated gate valves on the chase panel. From there, they pass through a volumetric flowmeter in the flow rate control box, and then to a box where they pass through pneumatic valves. The valves direct the gases in one of three ways. The oxygen and nitrogen can both be routed directly to the oxidation or diffusion furnace to provide dry gas, and the oxygen can be routed through a bubbler to provide wet gas.

Low purity nitrogen is used as compressed air for actuating the pneumatic valves, and are in turn regulated by solenoid valves, which are controlled by electrical lines coming from the gas flows controller (a programmable interface)


GAS TANKS
High Purity N2
High Purity O2
Low Purity N2
Deionized Water
CHASE PANEL
FLOW RATE AND MASS
FLOW CONTROL PANEL
N2 Volumetric Control Knob
O2 Volumetric Control Knob
BIG BLUE
CONTROL PANEL
Gas Flows Control
Upper Mantle Control
BUBBLER ASSEMBLY
Heat Blankets
Upper Mantle
Lower Mantle
Tubing
Bubbler
Lower Mantle Control
Tubing Mantle Control
Water Injection Button
Water Dispenser
OXIDATION FURNACE
CONTROL PANEL
Oxidation
Furnace
DIFFUSION FURNACE
CONTROL PANEL
Diffusion
Furnace

Appendix F. Concept Layout Drawings

For all cases, the concept consists of three subsystems: the tubing, the wires, and the user interface-controller assembly. The wires emerging from the user interface-controller assembly have been grouped into numbered bundles according to their destinations: 1 for wires connecting the volumetric flowrate knobs to the flow control valves (these are set once and never modified, which is why they have not been included in any digital interfaces of the device), 2 for wires to valves, 3 for wires from pressure gages, 4 for wires from thermocouples, and 5 for wires to heaters. The tubing arrangement shown was determined early on by Dr. Savage as the optimal solution, and the wiring to connect the various points to the controller is predetermined by the location and number of points (every communicating element requires a wire). Therefore, it was concluded that the wiring and tubing systems were static, and that no further design was required. The system layout is shown below, and is the same as that shown earlier in the document.

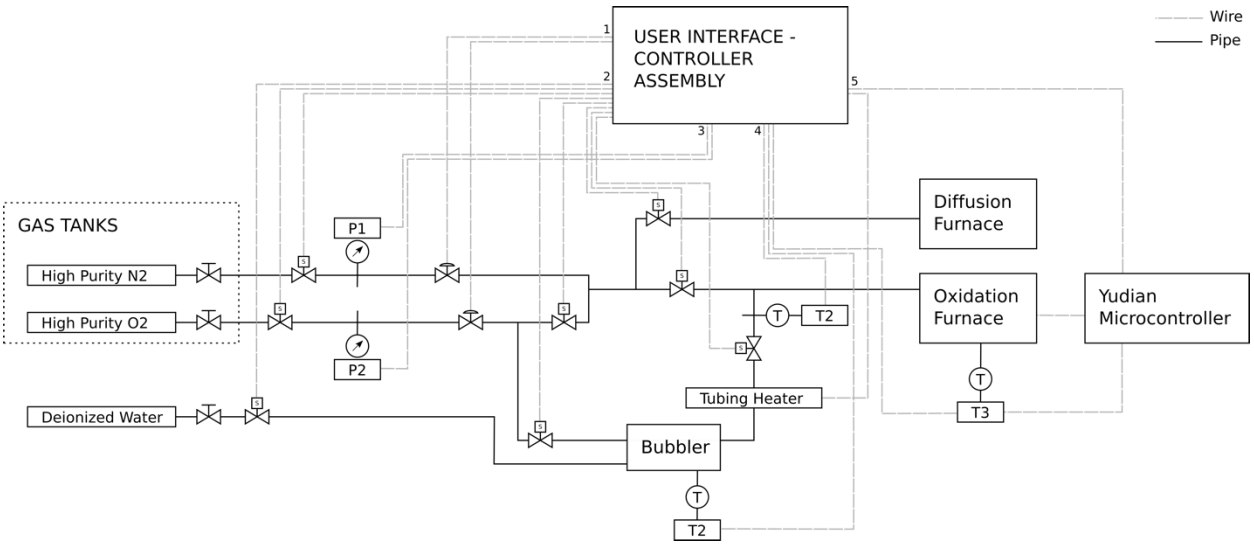


Figure 14. First Iteration system diagram

The user interface-controller assembly, however, was less well understood, and thus was considered separately with several iterative designs. The first and most basic of our designs is the current system controls, but with the controls rearranged to fit onto one control panel in a more intuitive interface. In this arrangement, the signals are routed between the user and the devices through an analog/digital converter, and all analysis and response is done by the operator. In the second design, an off-the-shelf controller is used, allowing the user to control the furnace and the gas flows separately. In this arrangement, the user must still monitor the furnace and adjust the gas regiment as necessary to match the temperature regiment. In the final arrangement, a microcontroller is programmed to act as the controller. This would be the most elegant design, but would also be the most difficult to construct due to the programming involved. The three designs are shown below in the order in which they were described.

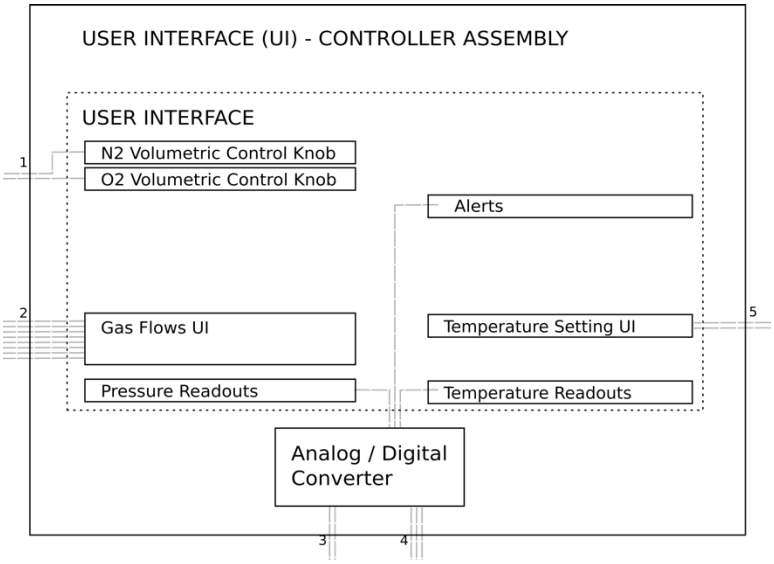


Figure 15. The current user interface, the easiest to implement but least user-friendly to operate.

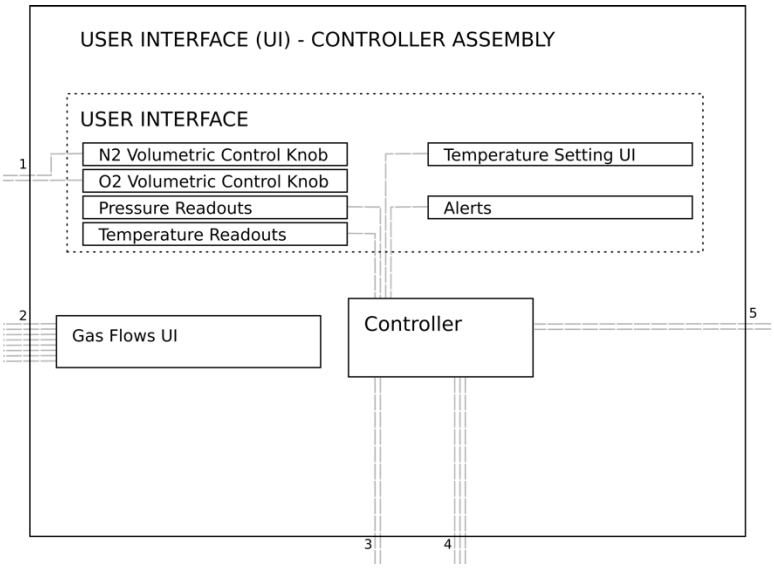


Figure 16. User interface employing an off-the-shelf controller. This allows for limited increased functionality of the furnace control interface.

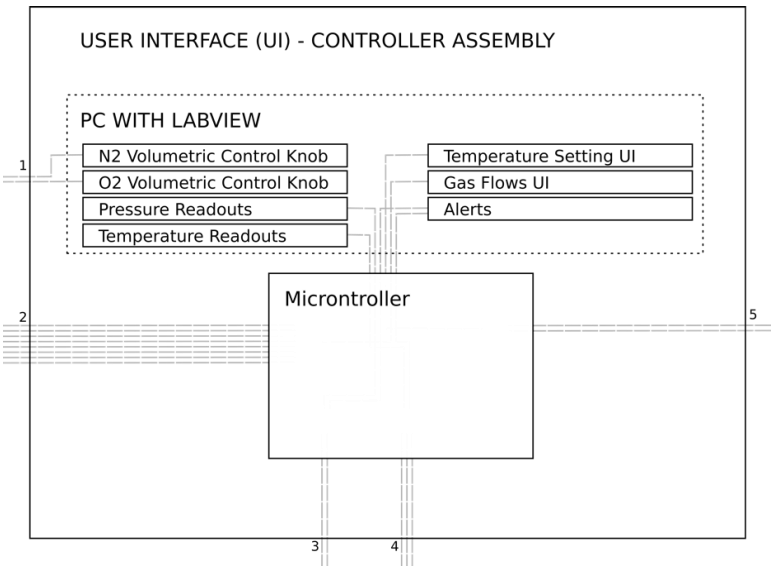


Figure 17. Most sophisticated interface, in which the user can control nearly all aspects of furnace operation. Signals are routed through a microcontroller, which controls the furnace system as a whole. This system requires customized programming, making it the most difficult of the three to implement.

After PDR, it was requested by our sponsor that the temperature setting interface be kept as it is on the original system, as implementing the change on our interface would require a potentially dangerous dissection of the current furnace interface. Also, the sponsor requested that control of the furnace be transferrable to the current interface in the event of emergencies. Further, it was discovered that the tubing could be simplified further from the layout shown in PDR. For these reasons, the user interface was modified to exclude temperature control. However, gas control was maintained. The system was also modified to allow the control of the gas flows to be switched back to the current control system. The figures below reflect this change, and are the system as is planned to be implemented.

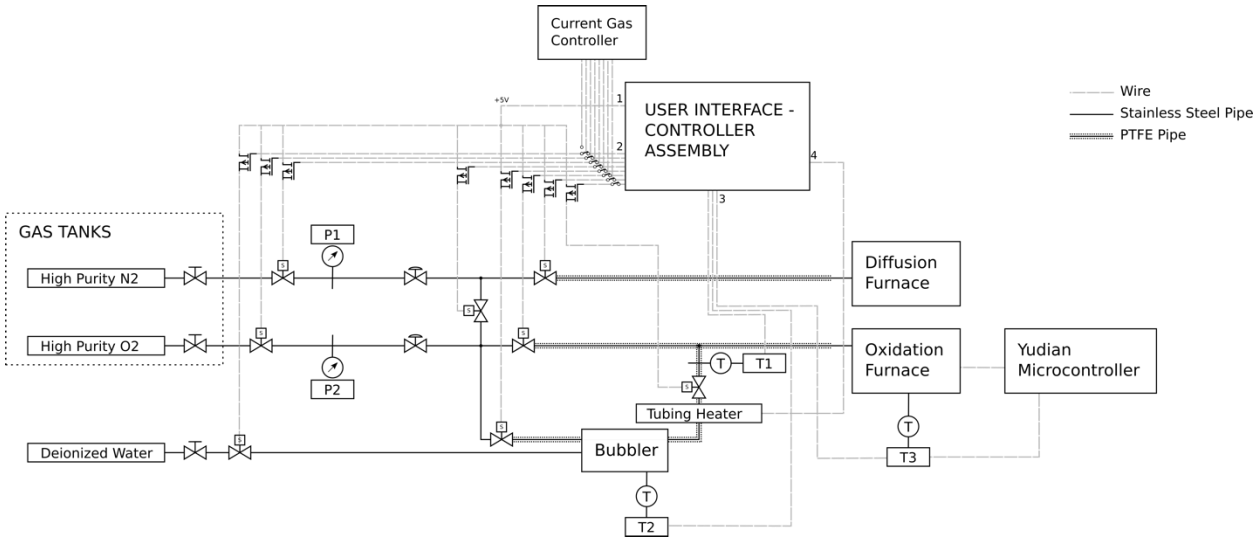


Figure 18. Second iteration system diagram, as presented at CDR. The tubing has been simplified further, and a switch has been added to divert control of the gas valves to the current gas controller, if necessary.

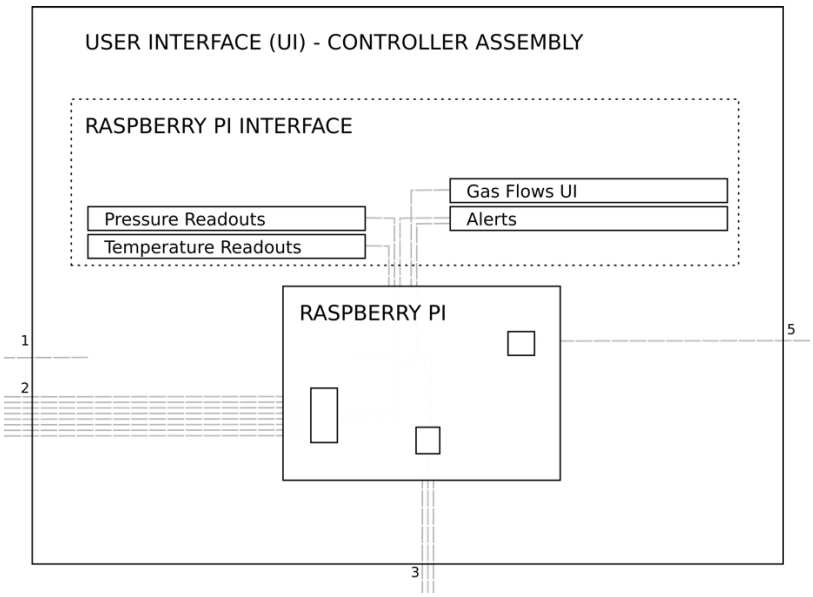


Figure 19. Second iteration controller-interface. At the request of the sponsor, furnace temperature will no longer be controlled by our system, but the system will make use of a Raspberry Pi single board computer.

Appendix G. Drawings Package

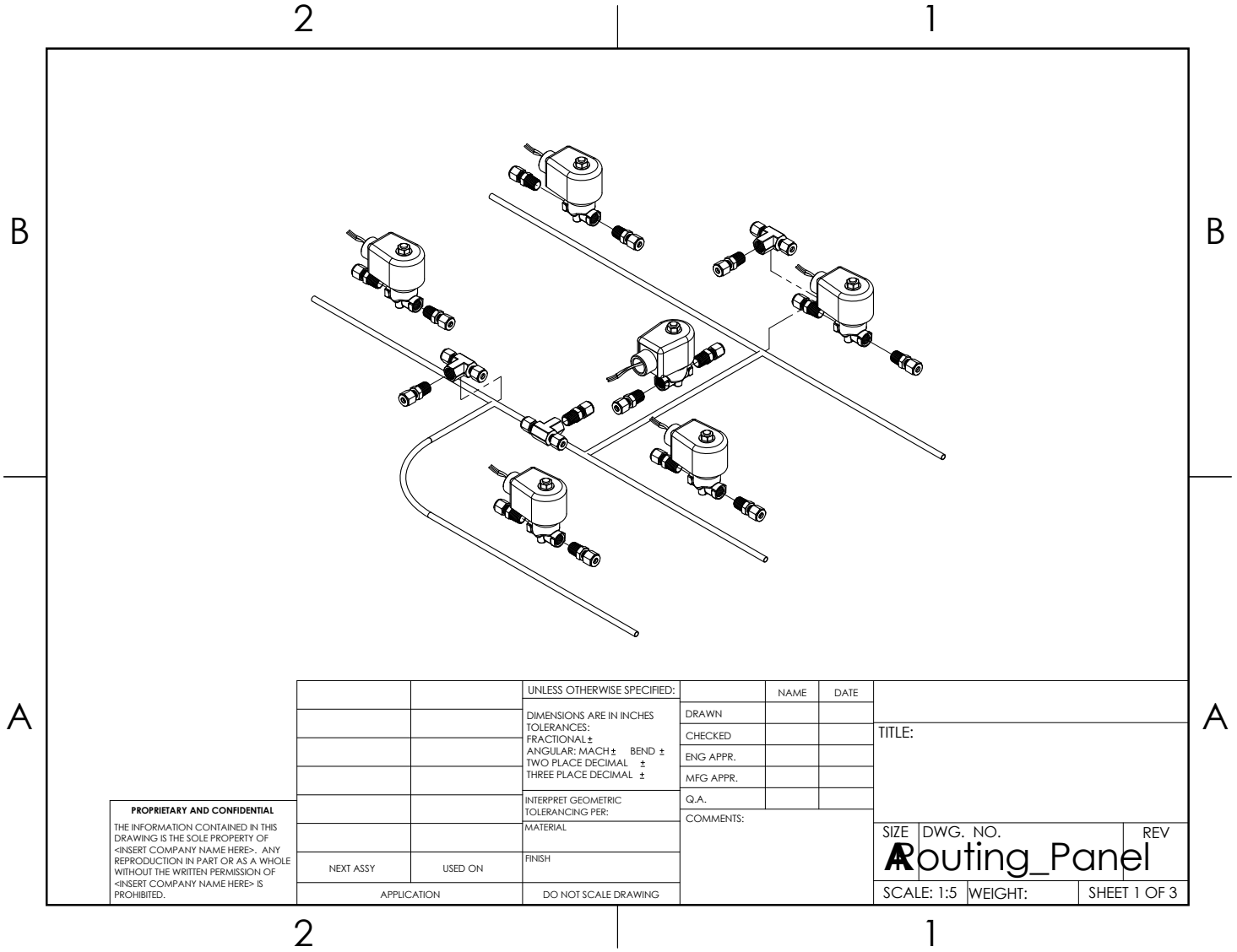


Figure 20. Routing Panel exploded view

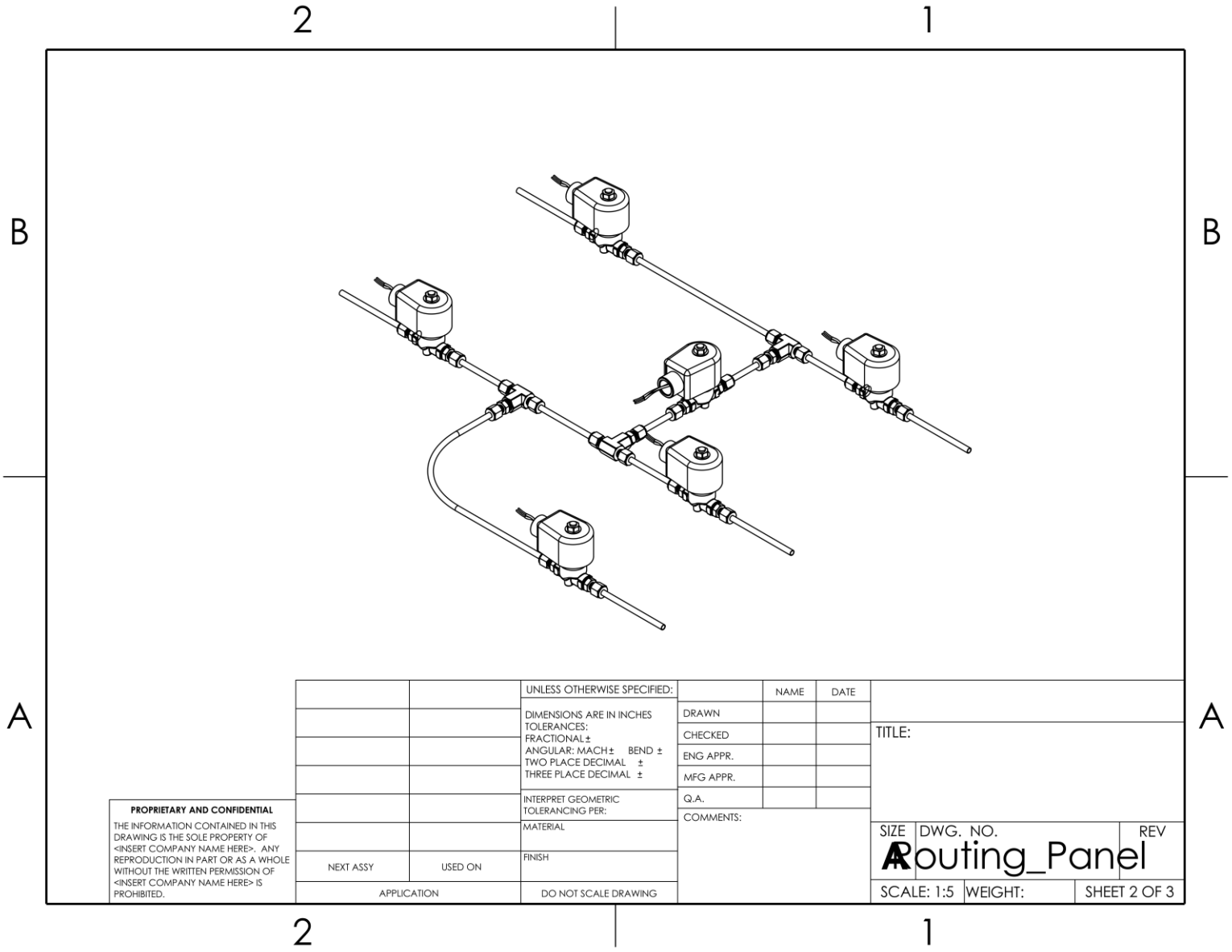


Figure 21. Routing panel isometric view

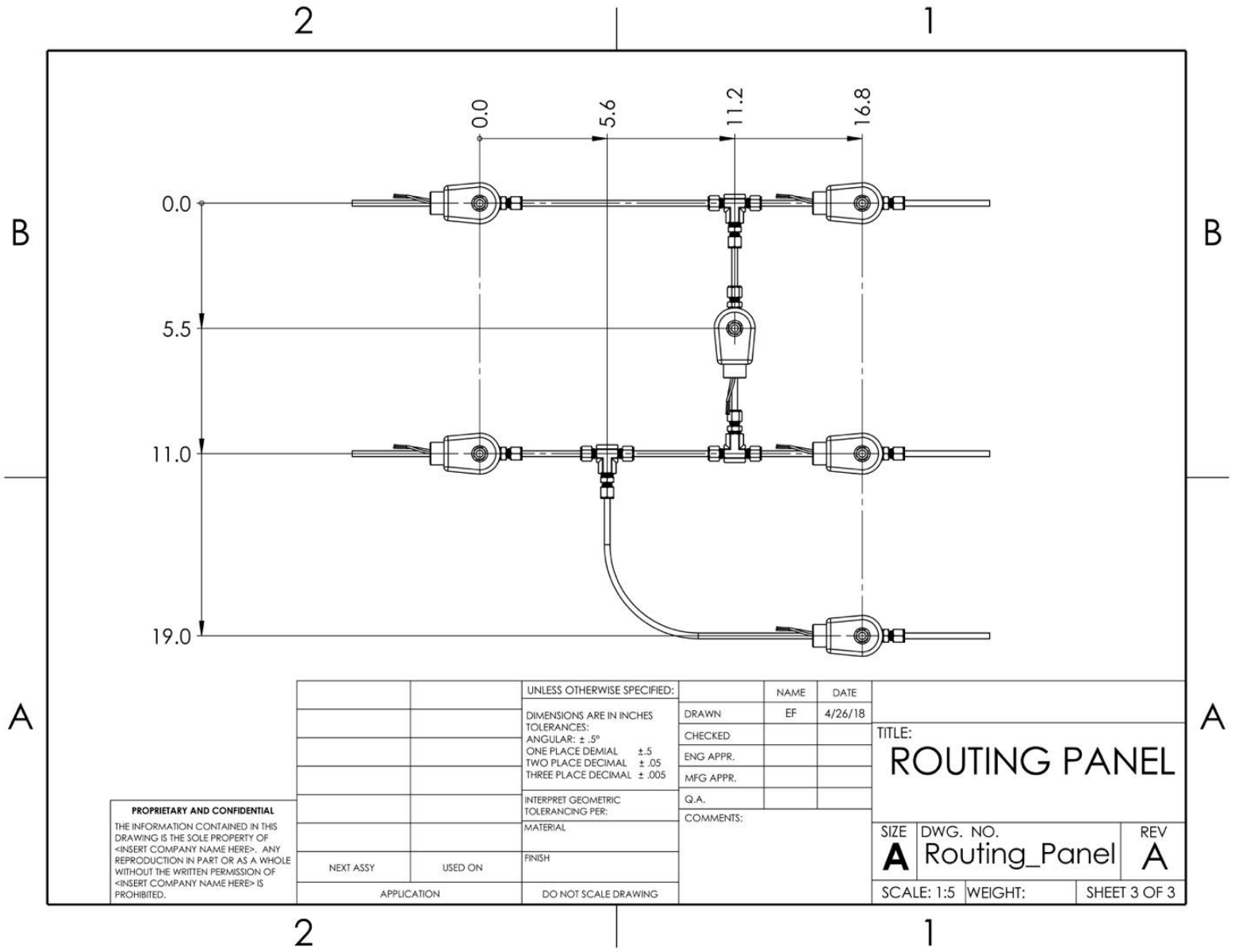


Figure 22. Routing panel top view

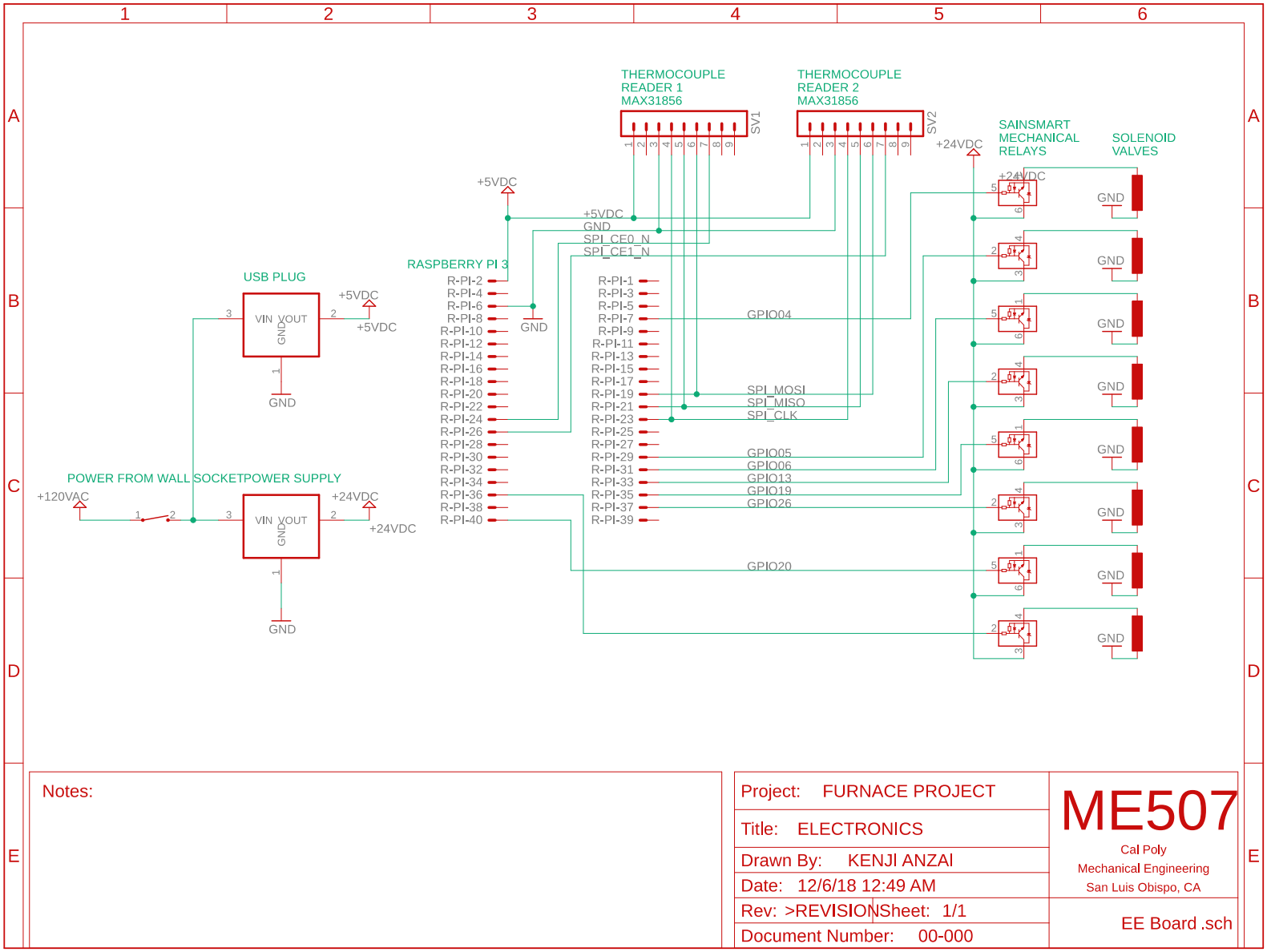


Figure 23. System wiring diagram

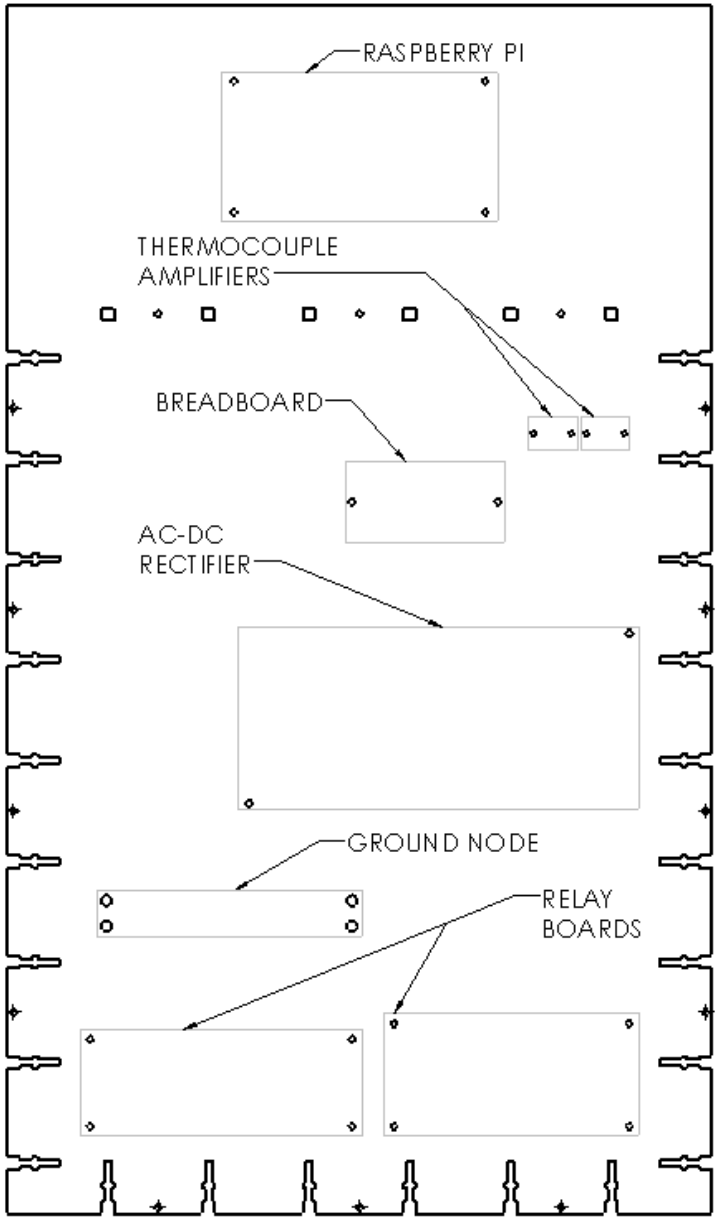


Figure 24. Annotated Electronic Housing Back Panel

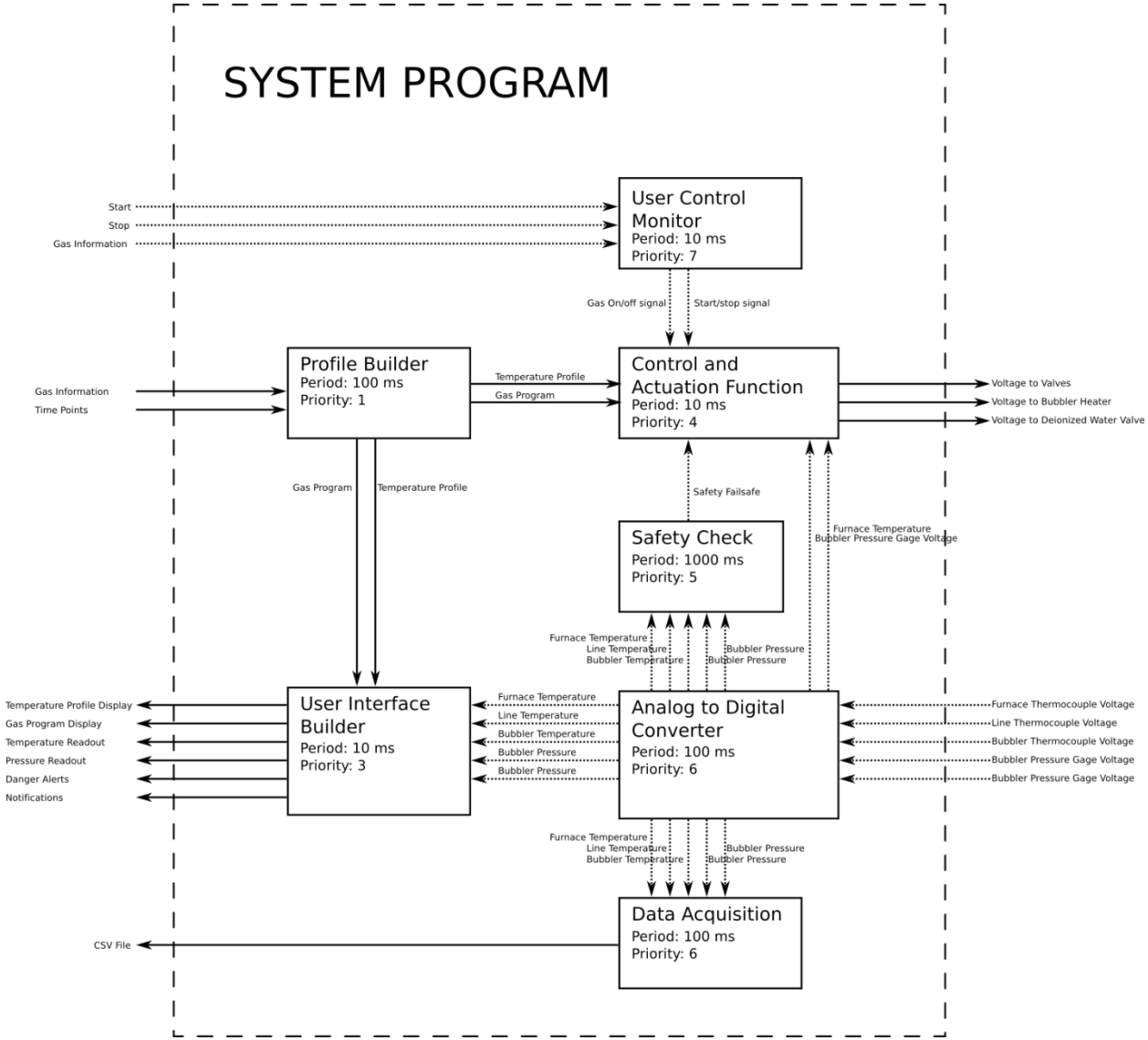


Figure 25. System software task diagram

Appendix H. Budget and Procurement List

Item	Vendor	Part NO	Unit Price	Quantity	Total
SSRs	Sainsmart	101-70-111	15.99	2	31.98
Acryllic Panel 1/8"x24"x48"	AcmePlastics	ACRYLIC -SHEET	24.14	1	24.14
TC amplifier	Adafruit	3263	17.50	2	35.00
Raspberry Pi 3	Adafruit	3055	35.00	2	70.00
Touchscreen	Adafruit	2718	79.95	1	79.95
Type-K TC	Adafruit	270	9.95	1	9.95
GPIO Extension	Amazon	N/a	9.99	1	9.99
M3x0.5x20mm standoff	FastenerFront	FM2155-3005-A	24.00	1	24.00
Teflon Tubing	Grainger	31231313	181.50	1	181.50
Bread Board	Jameco	WBU-202-R	9.95	1	9.95
Power supply	Jameco	RS-150-24	23.40	1	23.40
F-F jumper wire	Jameco	ZW-FF-20	4.95	1	4.95
m-m jumper wire	Jameco	ZW-MM-20	4.95	1	4.95
Yor-Lok fitting	McMaster	5182K111	8.73	15	130.95
T-Junction	McMaster	5182K222	36.27	5	181.35
Barbed Fittings	Mcmaster	5346K12	16.01	1	16.01
Band Clamps	McMaster	5388K14	6.02	1	6.02
Strut Channel, 6ft	McMaster	3310T57	27.27	1	27.27
M3x0.5x10mm screw	McMaster	92000A006	8.70	1	8.70
M3x0.5x5mm screw	McMaster	92000A003	11.72	1	11.72
M3x0.5x10mm nut	McMaster	90591A250	2.12	1	2.12
Valve	SMC	VXN22DS	47.00	6	282.00
male disconnect	Jameco	489731	0.35	50	17.5
female disconnect	Jameco	303327	0.29	50	14.5
ring terminals	Jameco	844570	0.29	50	14.5
wire (red)	Jameco	125736	15.95	1	15.95
switches	LiduidLevel	LS-11-075	53	2	106
Blow off	McMaster	4772K4	12.28	1	12.28
				Total	1356.63

Appendix I. Design Verification Plan and Report

TEST PLAN									
Item No	Specification #	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING	
						Quantity	Type	Start date	Finish date
1	Oxygen Line Safety	Examination	No risk of line puncture or ignition	Fedor	Incomplete	1	Sub	11/20	-
2	Size	Measurement	Fits within a box of size 36" x 48" x 15"	Zailaa	Done	1	S	11/20	11/30
3	Number of control panels	Examination	Two or fewer	Anzai	Done	1	S	11/10	11/20
4	Voltage	Measurement	No exposed voltages greater than 40V	Fedor	Done	1	S	11/23	11/30
5	Gas regimen programming	Testing	Success	Anzai	Done	1	Sub	11/23	11/30
6	Unsafe condition detection	Unsafe Condition Simulation	No Fail	NA	NA	1	S	-	-
7	Temperature resolution	Measurement	Temperature reading accurate to within 10ºF	NA	NA	1	S	-	-
8	Pressure resolution	Measurement	Pressure reading accurate to within 1 kPa	NA	NA	1	S	-	-
9	Cost	Cost Analysis	Total cost less than $5000	Zailaa	Done	1	S	11/30	12/4
10	Notifies user of insufficient gas	Simulate insufficient gas conditions	No Fail	NA	NA	1	S	-	-
11	Automated bubbler refill	Testing	Success	NA	NA	1	Sub	-	-
12	Number of pneumatic valves	Examination	No pneumatic valves	Zailaa	Done	1	S	10/20	10/27

Appendix J. Failure Modes Effects Analysis and Design Hazard Checklist

The Failure Modes Effects Analysis (FMEA) analyzes the ways in which the system could fail and predicts the effects of such failures.

												Action Results			
System / Function	**Potential Failure Mode**	**Potential Effects of the Failure Mode**	**Severity**	**Potential Causes of the Failure Mode**	**Current Preventative Activities**	**Occurence**	**Current Detection Activities**	**Detection**	**Priority**	**Recommended Action(s)**	**Responsibility & Target Completion Date**	**Actions Taken**	**Severity**	**Occurence**	**Criticality**
User Interface / Allow user to control furnace	Interface inputs are unintuitive or ineffective	a) User is unable to control furnace	8	1) temperature regiment controller breaks 2) gas flows controller breaks	1) protect controllers in housing 2) code user interfaces properly	1	User experience	2	16	Design interface to be intuitive	Anzai, 12/4	Safe stop installed	8	1	1
User Interface / Alert user	Interface alerts in safe situation or fails to alert	a) User ignores or is unaware of a dangerous situation	10	1) alerting light burns out 2) alerting system is not told to turn on 3) wrong data is read by alerting system	1) use LED light bulbs 2) ensure bulbs are new 3) alerting system contacts with controller	2	Alarm tests	2	40	Create attention-getting alerts	Anzai, 12/4	Unaddressed	10	2	2
User Interface / Display data	User interface fails to display data	a) User does not know what is going on in furnace	4	1) readouts break 2) readouts lose connection to data source	1) use robust readouts 2) wire readouts securely to sources	2	Readout tests	1	8	Ensure proper coding and connection	Zailaa, 11/1	NA	-	-	-
Tubing/rout gasses	Gasses don't get routed	a) process gets ruined	6	1) Tubing deteriorates 2) tubes get broken	1) Material Selection 2) Oxygen safe valves	3	Volumetric flowrate observation	3	54	Ensure routing reliability	Fedor, 12/4	System built to design specifications	6	1	1

				3) Oxygen line sparks											
Tubing/rout gasses	Wrong gas gets routed	a) process gets ruined	6	1) wrong valves opening/closing 2) stuck valves	1) proper coding	3	Volumetric flowrate observation	4	72	Perform extensive testing	Anzai, 12/4	System tested	6	1	1
Tubing/measure	System does not accurately measure properties in the tubes	a) User does not know what is going on in furnace	7	1) thermocouple malfunction 2) pressure gage breaks	1)	2	Testing	2	28	Testing using known conditions	Fedor, 12/4	NA	-	-	-
Tubing / control	Gas not routed as specified by user	a) process gets ruined	6	1) valves are not properly actuated	1) secure wiring 2) proper coding	3	Periodic testing	3	54	Debug code	Anzai, 12/4	Code written	6	1	1
Electronics/ Translate instructions into actuation	Instructions dont get translated	a) no gasses get routed b) improper ramping	6	a) loose wires in relays b) Power outages c) relays fail d) software bugs	1) secure wiring 2) alternative power source 3) unlikely but a necessary evil 4) proper coding	2	User observation	7	84	Secure wiring and ensure robust operation before installation	Fedor, 12/4	Wires secured	6	2	2
Electronics/ Execute instructions	Hardware doesn't do its job	a) no logic gets output b) actuation fails	6	a) Circuitry gets damaged b) electrical contacts loosen c) software bugs	1) well designed enclosure 2) secure wiring 3)proper coding	2	User observation	2	24	Ensure output through testing	Anzai, 12/4	Tested	6	1	1

Electronics/ Collect data	no data collected	a) no data for future reference	2	a) Bugged software	1) secure wiring 2) proper coding	1	User observation	1	2	Ensure proper design and manufacture of data collection systems	Anzai, 12/4	NA	-	-	-
Electronics/ Sense hazards and errors	hazards dont get sensed	a) Data doesn't get collected b) Process Fails c) Furnace damages itself	10	a) software bugs b) loose contacts c) failed sensors	1) proper coding 2) secure wiring 3) spec correct sensors	1	User observation	5	50	Design sufficiently sensitive equipment	Anzai, 12/4	NA	-	-	-
Electronics/ Translate users instructions for controller	Failure in translation	a) Furnace overheats b) Process fails	7	a) Software bugs	1) Proper coding	3	User observation	3	63	Debug software	Anzai, 12/4	Testing	7	1	1

Appendix K. Design Hazard Checklist

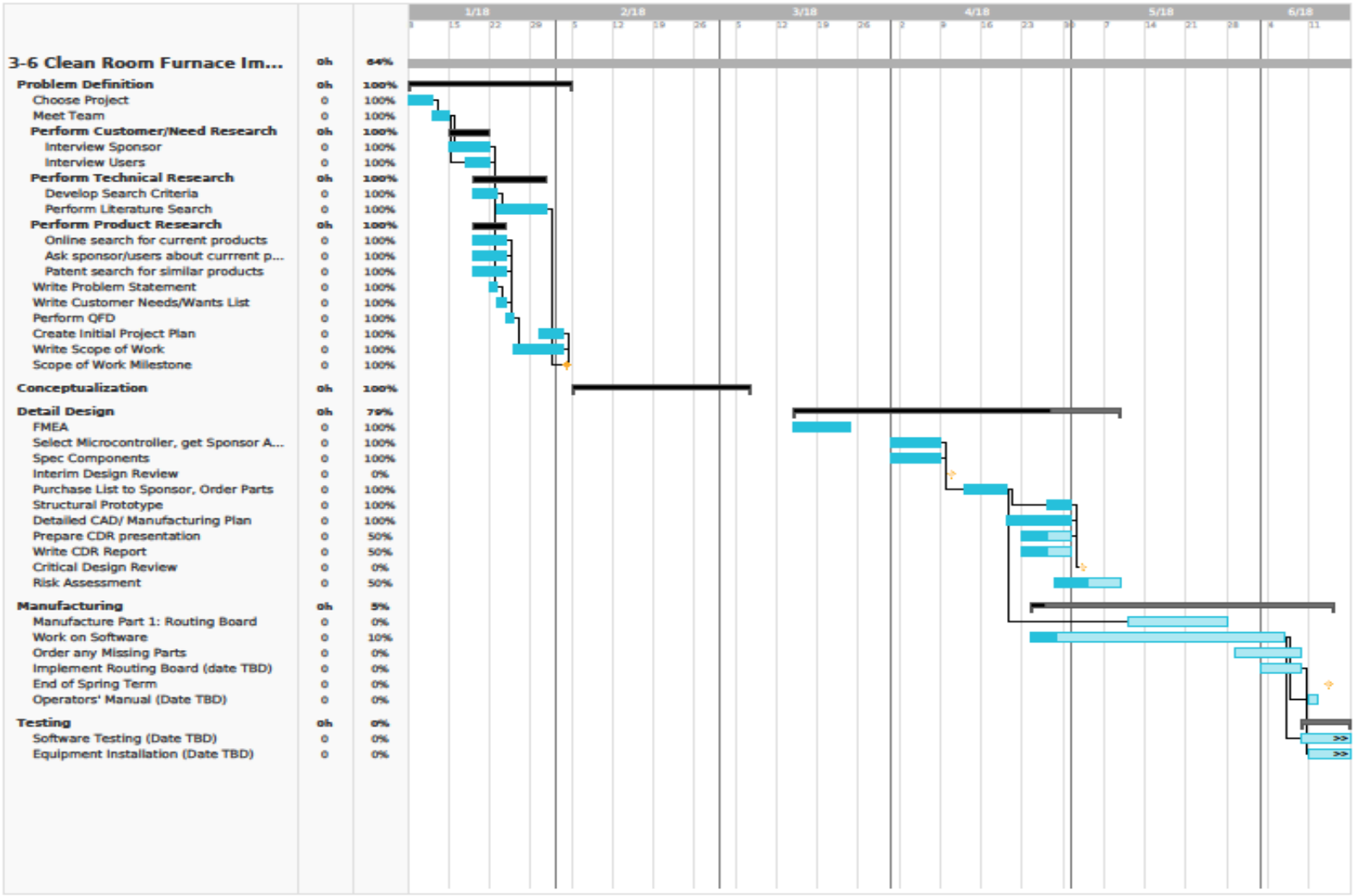
Team: _____ Advisor: _____ Date: _____

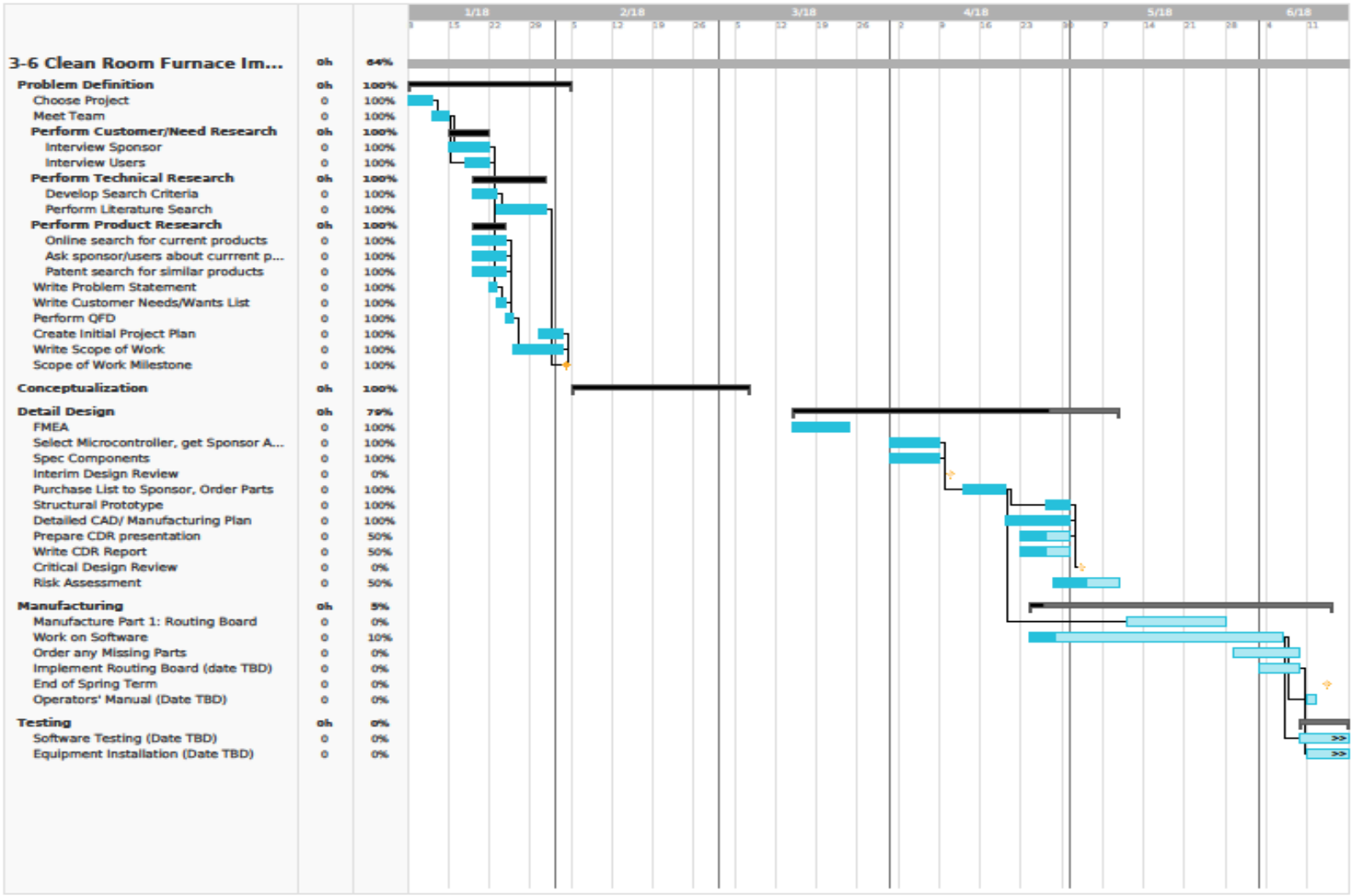
Y	N	
☐	■	1. Will the system include hazardous revolving, running, rolling, or mixing actions?
☐	■	2. Will the system include hazardous reciprocating, shearing, punching, pressing, squeezing, drawing, or cutting actions?
☐	■	3. Will any part of the design undergo high accelerations/decelerations?
☐	■	4. Will the system have any large (>5 kg) moving masses or large (>250 N) forces?
■	☐	5. Could the system produce a projectile?
☐	■	6. Could the system fall (due to gravity), creating injury?
☐	■	7. Will a user be exposed to overhanging weights as part of the design?
☐	■	8. Will the system have any burrs, sharp edges, shear points, or pinch points?
☐	■	9. Will any part of the electrical systems not be grounded?
☐	■	10. Will there be any large batteries (over 30 V)?
☐	■	11. Will there be any exposed electrical connections in the system (over 40 V)?
■	☐	12. Will there be any stored energy in the system such as flywheels, hanging weights or pressurized fluids/gases?
■	☐	13. Will there be any explosive or flammable liquids, gases, or small particle fuel as part of the system?
☐	■	14. Will the user be required to exert any abnormal effort or experience any abnormal physical posture during the use of the design?
☐	■	15. Will there be any materials known to be hazardous to humans involved in either the design or its manufacturing?
☐	■	16. Could the system generate high levels (>90 dBA) of noise?
☐	■	17. Will the device/system be exposed to extreme environmental conditions such as fog, humidity, or cold/high temperatures, during normal use?
■	☐	18. Is it possible for the system to be used in an unsafe manner?
■	☐	19. For powered systems, is there an emergency stop button?
☐	■	20. Will there be any other potential hazards not listed above? If yes, please explain on reverse.

For any "Y" responses, add (1) a complete description, (2) a list of corrective actions to be taken, and (3) date to be completed on the reverse side.

Description of Hazard	Planned Corrective Action	Planned Date	Actual Date
System contains pressurized gas lines, which could rupture and produce a projectile	Choose tubes, valves, and fittings wisely to ensure ruptures do not occur, and perform testing	05/04	
System contains pressurized gas lines	Choose tubes, valves, and fittings wisely to ensure ruptures do not occur, and perform testing	05/04	
System contains high purity oxygen	Design all components which will be near oxygen with care to ensure they will not ignite the oxygen line	05/04	
Furnace has an emergency stop lever	Notify users of the presence of the emergency stop lever, so that they may use it if necessary	05/04	

Appendix L. Gantt Chart




1/18
2/18
3/18
4/18
5/18
6/18
3-6 Clean Room Furnace Im... 0h 64%
Problem Definition 0h 100%
Choose Project 0 100%
Meet Team 0 100%
Perform Customer/Need Research 0h 100%
Interview Sponsor 0 100%
Interview Users 0 100%
Perform Technical Research 0h 100%
Develop Search Criteria 0 100%
Perform Literature Search 0 100%
Perform Product Research 0h 100%
Online search for current products 0 100%
Ask sponsor/users about currrent p... 0 100%
Patent search for similar products 0 100%
Write Problem Statement 0 100%
Write Customer Needs/Wants List 0 100%
Perform QFD 0 100%
Create Initial Project Plan 0 100%
Write Scope of Work 0 100%
Scope of Work Milestone 0 100%
Conceptualization 0h 100%
Detail Design 0h 79%
FMEA 0 100%
Select Microcontroller, get Sponsor A... 0 100%
Spec Components 0 100%
Interim Design Review 0 0%
Purchase List to Sponsor, Order Parts 0 100%
Structural Prototype 0 100%
Detailed CAD/ Manufacturing Plan 0 100%
Prepare CDR presentation 0 50%
Write CDR Report 0 50%
Critical Design Review 0 0%
Risk Assessment 0 50%
Manufacturing 0h 5%
Manufacture Part 1: Routing Board 0 0%
Work on Software 0 10%
Order any Missing Parts 0 0%
Implement Routing Board (date TBD) 0 0%
End of Spring Term 0 0%
Operators' Manual (Date TBD) 0 0%
Testing 0h 0%
Software Testing (Date TBD) 0 0%
Equipment Installation (Date TBD) 0 0%

Appendix M. Graphical User Interface Designs

Appendix N. User manual

Microfabulous System Improvements User Manual

Kenji Anzai, Ethan Fedor, and Patrick Zailaa

This user's manual includes instructions for product use and important safety information. Read this section entirely including all safety warnings and cautions before using the product.

Important: This product is meant for use on the furnace system in the clean room at California Polytechnic State University. Before using this product, the user should be familiar with the operation and safety risks of and should be certified to work in the cleanroom with the furnace.

Using the furnace controller

All gas flow valves are controlled via the terminal window on the Raspberry Pi. Use the following steps to start and operate the interface.

How to start the user interface

1. Turn on the Raspberry Pi. Navigate to the terminal app, and enter the following commands:

```
cd desktop
python3 FurnaceControl.py
```

2. The furnace control program will begin running. The default start position is with all the valves shut. First, the program will ask the user which furnace is to be supplied with gas. Type a furnace choice. Upon pressing the "return" key, the program will recognize your choice. The program will only recognize "Oxidation" or "Diffusion" (case insensitive) as choices, so if the user input is misspelt, the program will ask again.
3. Type the desired gas and press the "return" key. Misspelt gases will not be recognized, and the program will continue to ask the user for gases until the user inputs an option is recognized. Appropriate gas choices are nitrogen, oxygen, wet oxygen, and off.
4. When an appropriate option has been selected, the program will confirm this with a message such as "Switching to oxygen" or "shutting off gas." You may hear the valves click to the appropriate configuration. The program will then ask for a gas. If the user does nothing, the same gas will continue to be routed. To switch to a different gas, simply type the new gas name and the appropriate gas will be routed.
5. This program will continue to run until the user stops it. To do this, type "stop." This will shut off all gases and exit the program. If this does not occur, first shut off all gases by typing "None," then hold the "control" key, then press "c" on the keyboard. The program will exit to the directory folder, and to run it again enter the command.

```
python3 FurnaceControl.py
```

NOTE: Do NOT exit the program without first shutting off all gases.

Safety Warnings – DO NOTS

This product is meant for use on the furnace system in the clean room at California Polytechnic State University. Do not leave it exposed to direct sunlight or use it in any other capacity outside the microfabrications lab.

1. **** HIGH VOLTAGE WARNING **** DO NOT attempt to stick any fingers or pointed objects such as screwdrivers, pens, knives, rods etc... inside electronic box slots on the top and bottom walls, or through the cooling holes on the front wall above the power supply. Electric shock from the high voltage side of the power supply could be fatal.
2. DO NOT unplug any wires connected to the Raspberry Pi or any other electronic wires. The program only works for the current wiring configuration and changes to the input/output connections causes the program to malfunction.
3. DO NOT under any circumstance disassemble the electronic housing box. If any electrical work or rewiring is needed, submit a request to have a certified technician do it.
4. DO NOT spill any food or liquids on the Raspberry Pi or any electronics associated.
5. DO NOT exit the program without first shutting off all gases. The valves will remain open and gases will continue to flow even if the program is shut, if the gases are not shut off using the “None” command.

Troubleshooting

If Raspberry Pi is not turning on

1. Make sure the Pi is connected.
2. Make sure the switch is turned ON.
3. Check to see if there is any loose wiring.
4. If the Pi still does not turn on, contact the instructor.

If the system is unresponsive

1. Ensure proper wiring between the electronic components.
2. Ensure the gas commands are spelled correctly.

If you suspect there is a gas leak

1. Close all the solenoid valves by typing the “None” command into the terminal window.
2. Inspect the tubes and valves to find the source of the leak.

In the case of system malfunction

1. Shut off all the gases.
2. Turn off, then unplug the Raspberry Pi.
3. Revert to the old furnace control system.

If operating at unsafe temperatures

1. Close all the valves.
2. Turn off the Raspberry Pi.
3. Shut down the furnace.

Appendix O. Project Code

```
"""
Created on Tue May 22 12:50:17 2018
@b FurnaceControl.py
@c FurnaceControl contains a class which contains all the methods needed to
run
the furnace. There is a method which reads a .txt file to obtain preset gas
sequence instructions, a method which writes a .txt file with data gathered,
a method which controls valves based on the desired gas and furnace, a method
which allows the user to manually input these gases, and a method which shuts
off all valves and exits the program.

The furnace currently operates in manual mode. To operate the furnace,
navigate
to the directory containing FurnaceControl.py in Terminal and type
"python3 FurnaceControl.py".

@author: Kenji Anzai
"""

import RPi.GPIO as GPIO
# from Adafruit_GPIO.GPIO import *
# import Adafruit_GPIO.SPI as SPI
# import Adafruit_GPIO
# from .GPIO import *

# Local Imports
# from max31856 import MAX31856 as MAX31856
#import utime

class furnaceControl(object):
    def __init__ (self):
        '''
        @b __init__ is the initialization method for the class. It contains
the
        definitions of class variables which will be shared between all
methods
        '''
        ## @b CurrentData is a list containing the current measurements from
        #  the furnace
        self.CurrentData = []

        ## @b OldData is a list containin the previous measurements from the
        #  furnace
        self.OldData = []

        ## @b AllData is a list of lists containing measurements from the
        #  furnace at all times during the current run. @c CurrentData is
        #  appended to this list every time @c DataLog runs.
        self.AllData = []

        ## @b GUIStatus is a list containing the status of the GUI.
Information
        #  on gas setpoints, timing, and which buttons are pressed is
contained
        #  here. These values are obtained from a @c .txt written by the GUI.
```

```
        self.GUIStatus = [[], []]

        ## @b GoNogo is a variable which tells the @c Controller method
whether
        #  or not to continue operating. If set to @c 1, all is normal, and
        #  operation may continue. If it is @c 0, there is a safety concern,
        #  and the furnace must be shut down.
        self.GoNogo = 1

        ## @b stop is a variable which tells the @c ManualMode method whether
or
        #  not to continue looping. If set to 1, the @c stop method runs
        # self.stop = 0

        ## @b DataToGUI is a list containing all data to be sent back to the
GUI.
        #  This includes furnace condition data and warning messages.
        self.DataToGUI = []

        ## @b Warnings is a list containing all the warnings and other
        #  notifications for the user listed as strings.
        self.Warnings = []

        ## @b SafeCondition is a list containing the maximum allowable
        #  pressures and temperatures, as well as the maximum allowble change
        #  in pressures and temperatures
        self.SafeCondition = []

        ## @b desiredFurnace is the furnace to which gas will be routed
        self.desiredFurnace = None

        ## @b step is the step in the process which the furnace is currently
on
        self.step = 0

        ## @b gasOptions is the set of available gases. The program converts
all
        #  input to lower case and checks to see if it is in this set. If the
        #  typed option is not in the set, the furnace does nothing and asks
        #  again for a gas input.
        self.gasOptions = {'nitrogen', 'oxygen', 'wet oxygen', 'none'}

        ## @b furnaceOptions is the set of available furnaces. The program
converts all
        #  input to lower case and checks to see if it is in this set. If the
        #  typed option is not in the set, the furnace does nothing and asks
        #  again for a gas input.
        self.furnaceOptions = {'diffusion', 'oxidation'}


        self.valve0 = 7
        self.valve1 = 29
        self.valve2 = 31
        self.valve3 = 33
        self.valve4 = 35
        self.valve5 = 37
        self.valve6 = 40
```

```
        self.valve7 = 36
        self.valve8 = 38

        GPIO.setup(self.valve0, GPIO.OUT)
        GPIO.setup(self.valve1, GPIO.OUT)
        GPIO.setup(self.valve2, GPIO.OUT)
        GPIO.setup(self.valve3, GPIO.OUT)
        GPIO.setup(self.valve4, GPIO.OUT)
        GPIO.setup(self.valve5, GPIO.OUT)
        GPIO.setup(self.valve6, GPIO.OUT)
        GPIO.setup(self.valve7, GPIO.OUT)
        GPIO.setup(self.valve8, GPIO.OUT)

        GPIO.output(self.valve0, GPIO.HIGH)    # Insert proper GPIO pins and
states here
        GPIO.output(self.valve1, GPIO.HIGH)
        GPIO.output(self.valve2, GPIO.HIGH)
        GPIO.output(self.valve3, GPIO.HIGH)
        GPIO.output(self.valve4, GPIO.HIGH)
        GPIO.output(self.valve5, GPIO.HIGH)
        GPIO.output(self.valve6, GPIO.HIGH)
        GPIO.output(self.valve7, GPIO.HIGH)
        GPIO.output(self.valve8, GPIO.HIGH)

        print("Furnace control successfully booted. Type 'stop' at any time
to exit")



    def readFile (self, file):
        '''
        @b ReadFile reads GUI status information from a @c .txt file.
        '''
        DataArray = []
        desiredGas = []
        triggerTemp = []
        index = 0
        options = self.gasOptions
        with open (file, 'r') as afile:   #generates a list named lines,
consisting of a single column of strings
            lines = afile.readlines()

        self.desiredFurnace = lines.pop(0)

        for line in lines:                      #iterates over all elements
in lines
            line = line.split(',')
            DataLine = []
            for element in line:                #iterates over all elements
in line
                if self.is_number(element):          #Converts elements to
floats individually and adds them to a list
                    DataLine.append(float(element))
                else:
                    element = str(element)
                    DataLine.append(element)    #Deletes strings which cannot
be converted to floats and adds the nonetype to list
```

```
            if len(DataLine) >= 2:              #Adds the list to the array
if the length is at least 2 and none of the first two elements are nonetypes
                if (DataLine[0] in options) and
(self.is_number(DataLine[1])) :
                    DataArray.append(DataLine)
                    desiredGas.append(DataLine[0])
                    triggerTemp.append(DataLine[1])
                else:
                    if dataLine[0] not in options:
                        print('Error in line ', line , ': There\'s something
wrong with your desired gas')
                    if not self.is_number(DataLine[1]):
                        print('Error in line ', line , ': There\'s something
wrong with your desired temperature')
                    del DataLine[:]
            else:
                del DataLine[:]
        print(DataArray)
        self.GUIStatus = [desiredGas, triggerTemp]



    def is_number(self, s):
        '''
        @b is_number checks to see if a string can be expressed as a float.
Returns True if so,
        False if not. Sourced from
https://stackoverflow.com/questions/354038/how-do-i-check-if-a-string-is-a-
number-float
        '''
        try:
            float(s)
            return True
        except ValueError:
            return False


    def writeFile (self, fileName, array):
        '''
        @b WriteFile writes system data to a @c .txt file.
        '''
        file = open(fileName,'w')
        temp = []
        for line in array:
            line.insert(1, ',')
            line.insert(3, '\n')

            tempLine = ''.join(line)
            temp.append(tempLine)
        print(temp)
        file.writelines(temp)
        file.close()


    def switch (self):
        next = self.step + 1
```

```
        furnaceTemp = self.CurrentData[1]   # change this number later
        if next not in range(len(self.GUIStatus[0])):
            return



    def controller (self):
        '''
        @b Controller uses data and GUI status to send control signals to the
        furnace.
        '''
        print(self.step)
        print(self.GUIStatus)

        DesiredGas = self.GUIStatus[0][self.step]      # Insert proper
GUIStatus index
        Furnace = self.desiredFurnace


        if not self.GoNogo:
            print('Furnace shut down for safety reasons.')
            GPIO.output(self.valve0, GPIO.HIGH)    # Insert proper GPIO pins
and states here
            GPIO.output(self.valve1, GPIO.HIGH)
            GPIO.output(self.valve2, GPIO.HIGH)
            GPIO.output(self.valve3, GPIO.HIGH)
            GPIO.output(self.valve4, GPIO.HIGH)
            GPIO.output(self.valve5, GPIO.HIGH)
            GPIO.output(self.valve6, GPIO.HIGH)
            GPIO.output(self.valve7, GPIO.HIGH)
            GPIO.output(self.valve8, GPIO.HIGH)
            return

        '''Setting appropriate gas valves'''
        if DesiredGas == 'oxygen':
            if Furnace == 'oxidation':
                GPIO.output(self.valve0, GPIO.HIGH)    # Valves for sending
oxygen to the oxidation furnace
                GPIO.output(self.valve1, GPIO.HIGH)
                GPIO.output(self.valve2, GPIO.HIGH)
                GPIO.output(self.valve3, GPIO.LOW)
                GPIO.output(self.valve4, GPIO.LOW)
                GPIO.output(self.valve5, GPIO.HIGH)
                print('Oxygen routed to the oxidation furnace')
            else:
                print('oxygen is not an allowed gas for the diffusion
furnace')

        elif DesiredGas == 'wet oxygen':
            if Furnace == 'oxidation':
                GPIO.output(self.valve0, GPIO.HIGH)    # Insert proper GPIO
pins and states here
                GPIO.output(self.valve1, GPIO.HIGH)
                GPIO.output(self.valve2, GPIO.HIGH)
                GPIO.output(self.valve3, GPIO.LOW)
                GPIO.output(self.valve4, GPIO.HIGH)
                GPIO.output(self.valve5, GPIO.LOW)
```

```
                print('Wet oxygen routed to the oxidation furnace')
            else:
                print('wet oxygen is not an available gas for the diffusion
furnace')

        elif DesiredGas == 'nitrogen':
            if Furnace == 'oxidation':
                GPIO.output(self.valve0, GPIO.LOW)    # Insert proper GPIO
pins and states here
                GPIO.output(self.valve1, GPIO.HIGH)
                GPIO.output(self.valve2, GPIO.LOW)
                GPIO.output(self.valve3, GPIO.HIGH)
                GPIO.output(self.valve4, GPIO.LOW)
                GPIO.output(self.valve5, GPIO.HIGH)
                print('Nitrogen routed to the oxidation furnace')
            else:
                GPIO.output(self.valve0, GPIO.LOW)    # Insert proper GPIO
pins and states here
                GPIO.output(self.valve1, GPIO.LOW)
                GPIO.output(self.valve2, GPIO.HIGH)
                GPIO.output(self.valve3, GPIO.HIGH)
                GPIO.output(self.valve4, GPIO.HIGH)
                GPIO.output(self.valve5, GPIO.HIGH)
                print('Nitrogen routed to the diffusion furnace')

        elif DesiredGas == 'none':
            GPIO.output(self.valve0, GPIO.HIGH)    # Insert proper GPIO pins
and states here
            GPIO.output(self.valve1, GPIO.HIGH)
            GPIO.output(self.valve2, GPIO.HIGH)
            GPIO.output(self.valve3, GPIO.HIGH)
            GPIO.output(self.valve4, GPIO.HIGH)
            GPIO.output(self.valve5, GPIO.HIGH)
            print('Gas has been shut off')



    def readData (self):
        '''
        @b ReadData reads sensors to obtain the state of the furnace.
        '''
        Pressure1 = 1
        Pressure2 = 2
        Pressure3 = 3
        Temp1 = 1
        Temp2 = 2
        self.OldData = self.CurrentData
        self.CurrentData = [Pressure1, Pressure2, Pressure3, Temp1, Temp2]

    def DataLog (self):
        '''
        @b DataLog creates a growing matrix of data values and saves the data
        to an SD card
        '''
        self.AllData.append(self.CurrentData)
```

```
        '''Figure out how to save things to SD cards!'''

    def SafetyCheck (self):
        '''
        @b SafetyCheck takes the data from the furnace and tests whether the
        state is within acceptable bounds. If the furnace is in an unsafe
        condition, sends a signal to stop the furnace.
        '''
        i = 0
        for Measurement in self.CurrentData:
            if Measurement >= self.SafeCondition[i]:
                self.GoNogo = 0

    # def UserInput (self):
    #     '''
    #     @b UserInput checks for user intput to the terminal. If the user
inputs
    #     a new gas, the program wipes the current automated run and runs the
user
    #     specified gas until the user specifies a new one or stops the
program.
    #     '''


    def stop(self):
        '''
        @b stop turns off all valves and exits the program, so the user has a
        safe way to exit the program.
        '''
        GPIO.output(self.valve0, GPIO.HIGH)    # Insert proper GPIO pins and
states here
        GPIO.output(self.valve1, GPIO.HIGH)
        GPIO.output(self.valve2, GPIO.HIGH)
        GPIO.output(self.valve3, GPIO.HIGH)
        GPIO.output(self.valve4, GPIO.HIGH)
        GPIO.output(self.valve5, GPIO.HIGH)
        GPIO.output(self.valve6, GPIO.HIGH)
        GPIO.output(self.valve7, GPIO.HIGH)
        GPIO.output(self.valve8, GPIO.HIGH)
        GPIO.cleanup()
        endMessage = "Gas has been successfully shut off"
        return endMessage


    def T_get(self):
        '''
        @b T_get reads the thermocouple. This does not yet work, so it is not
        used in this code. However, if the thermocouple can be made
operational,
        this is how it would be read.
        '''
        spi_port = 0
        spi_device = 0
        sensor = MAX31856(hardware_spi=SPI.SpiDev(spi_port, spi_device))

        temp = sensor.read_temp_c()
        print("We have a temperature!", temp)
```

```
    def ManualMode(self):
        '''
        @b ManualMode runs the program in manual mode, where the gas is
inputted
        based on user input written directly to the terminal. This program
waits
        until the user types in an option and presses enter. If the user
typed a
        valid response, it runs @c controller() to switch the appropriate
valves
        on.
        '''
        while not self.desiredFurnace:
            furnace = input('Which furnace? Options are diffusion or
oxidation \n').lower()
            if furnace not in self.furnaceOptions:
                print(furnace, 'is not an available furnace. Please try
again.')
            else:
                self.desiredFurnace = furnace

        while True:
            desiredGas = input('Which gas? Options are oxygen, wet oxygen,
nitrogen, or none \n')
            if desiredGas.lower() == "stop":
                print(self.stop())
                return
            elif desiredGas.lower() not in self.gasOptions:
                print(desiredGas, 'is not an available gas. Please try
again.')
            else:
                self.GUIStatus[0].append(desiredGas.lower())
                self.GUIStatus[1].append(float('inf'))
                if desiredGas.lower() == 'none':
                    print('The gas has been shut off.')
                else:
                    print('Gas has been changed to ',
self.GUIStatus[0][self.step])
                self.controller()
                self.step += 1



if __name__ == "__main__":
    GPIO.setmode(GPIO.BOARD)
    runFurnace = furnaceControl()
    runFurnace.ManualMode()
    # fileName = 'FurnaceData.txt'
    # data = [['0','1'],['1','5'],['2','1'],['3','3'],['4','3']]
    # runFurnace.writeFile(fileName,data)
    #fileName = 'routine.txt'
    #runFurnace.readFile(fileName)
    #print(furnaceControl.GUIStatus)
    # while true:
    #     runFurnace.readFile('example.txt')
```

```
#     yield
```


CALIFORNIA • POLYTECHNIC • STATE • UNIVERSITY • MECHANICAL • ENGINEERING • 1941 •