Measurement System Automation:

Tech Feasibility

4/1/25

Team Name: Thermo-Gen

Sponsor: Steve Miller

Mentor: Jeevana Swaroop Kalapala

Team Members: Olivia Vester, Kameron Napier, Gareth Carew

THERMO-GEN

Table of Contents

1. Introduction

Safety in the field is a priority in every industry– engineering, geology, space exploration, and more. To maintain safety, particular precautions must be taken and protective materials must be worn or used with necessary equipment.

At HeetShield, they create the materials that ensure the safety of researchers, engineers, scientists, etc. while they conduct their work. These materials range from thermal layers to protect people and objects (like scuba gear) from extreme cold temperatures to heat resistant materials that line spacecrafts. The materials hold a high level of importance for the protection of the individuals impacted by extreme environments. One factor that any individual entering extreme conditions would like to rely on is that the materials made to protect them have been rigorously and precisely tested. That is where the problem we are solving for our client HeetShield comes into play.

As stated above, HeetShield creates thermal materials used across several technical disciplines for purposes of protection. HeetShield has been working on manufacturing and testing these materials since 2020 under the leadership of Steve Miller with contracted work to organizations like NASA and the United States Marine Corps. The company creates these materials and has a custom-built in-house apparatus to test them, but, unfortunately, setting the condition parameters for this testing is extremely difficult and lacks much needed precision and sophistication an operation like this requires.

Right now, the company's temporary solution to these problems is to spend hours on data collection using manually set parameters that are not as exact as they could be. Our job is to create an interface for the measurement apparatus that allows technicians to precisely set temperature and pressure conditions.

Our solution is one that aims to be fulfilled by the creation of a standalone software that allows the apparatus operator to set the temperature for the testing environment and alert them of when undesired pressure changes occur. Our solution, though seemingly cut-and-dry, comes with a series of challenges that must be overcome in order to create a beta software.

2. Technological Challenges

As it is relatively early in the project, we are in the process of analyzing all of the important technological challenges, identifying possible alternatives, and selecting which of those alternatives provide the most promising solutions. In this document, we begin by analysing the major technological challenges we expect to face while trying to make the system more efficient and precise.

The main challenges are determining how to control both temperature and pressure digitally while ensuring that the automated system is faster and more accurate than manual controls. Yet another challenge is finding and implementing a solution for reading data from the thermocouple. A smaller but still just as important challenge would be setting up a GUI so that the system operator can easily enter the desired test environment and choose how they want to be notified. The final major challenge is

determining what type of notification and interface system to implement to ensure safety and maximize usability. We also want to emphasize the usage of Python to write the code for our system.

In the following section, we then analyze each of these challenges in turn: looking at alternatives, discussing how we explored them, and explaining our rationale for choosing a particular solution. The integration section following our analysis explains how we plan to combine all of the smaller solutions into a cohesive final product.

3. Technology Analysis

Our client faces a very specific issue: the product they test has qualities that outweigh the ability of their current technology. A current need in testing for the team at HeetShield is to get their testing apparatus to be set to 760C+ temperatures. We want to make a digital interface for their measurement system to allow temperatures and pressures to be set as a parameter and measured throughout the testing process. The other major issue is reading data from the thermocouples. We have decided to use National Instrument's Python API in order to pull data directly from the DAC.

Our ideal solution to this problem would be for us to build an application that would allow us to read the data from the thermocouples and then based on the data collected, control the temperature and pressure within the measurement system and then send notifications to tell the test operator that the parameters are met and ready for testing or that they can't be met.

3.1 Issue 1 - Language Integration and Thermocouple Reading

3.1.1 Intro

As computer scientists, the language in which we are conducting our work is important to establish early-on. A language that is known by all team members will be the base-line need for us to develop a piece of software with. With this in mind, the team has also looked into what language would allow us to easily create a nice looking GUI and be compliant with the mechanical parts used in this project.

3.1.2 Desired Characteristics

All of the components we are going to be using (temperature measuring, pressure measuring, etc.) can be integrated into a software system we created by using pre-made Python packages that we can use to complement our system. We want our language to not just be compatible with the team as programmers but with our client's mechanical needs as well. It is also significant that we can read the thermocouple data coming in from the DAC as well using National Instrument's APIs as this is where we will be getting our temperature information from in the testing environment. We also need to be able to show the user all of the information that we are gathering, so we need to be able to create a UI that is simple, yet shows all of the data that they need to see as well as places where they can adjust test parameters.

3.1.3 Alternatives

As with every decision in a large-scale project, a backup plan must be put into place. Stated above, it is established that the team has chosen Python for the API features that integrate well with the work we wish to conduct, like the National Instrument's Python API, but there are other alternatives out there. There are also a number of GUI frameworks available for python such as Tkinter, PyQt or PySide.

Another possible solution would be for us to use a language like C; all the team members have knowledge in this language and since it is quite popular, finding integrative techniques is an easily researchable topic. A simple search gives results for several C language weather tracking APIs that could be molded to fit our system. The main GUI toolkit for C is GTK, which while usable, like the rest of C, requires a lot more effort and knowledge.

A final possible alternative for language integration would be using something like `lm-sensors` on Linux which is made to read hardware temperatures; we list this last just because Linux can be tricky and do not want to prioritize it as a main alternative.

3.1.4 Analysis

For Python, there is loads of documentation out there that helps make what we need compatible with what we have (like for Python and DAC for thermocouple reading).

For C, there is also plenty of similar documentation like that of Python. We do, as a team, know C, but find it to require slightly more work than a language like Python may require because of its manual memory management.

Lastly, is the Linux option. Those who prefer Linux, swear by it, and can frankly bend it at their will to make it function however they want, but those who do not often use it, find it to be a little tough to navigate. Our team, however, has no strong feelings towards Linux being our choice language. Linux is capable of interpreting data like temperature and pressure inputs but must do so with a chain of commands more complex than is necessary.

3.1.5 Chosen Approach

Our chosen approach, as previously stated, is to use Python as our language of choice for integrating the software component of our project. We chose this approach not only for the ease of use for the team itself but also because of the system compatibility with widely available APIs and features that can help us with reading in thermocouple data as well as the fairly straightforward GUI packages which allow us to easily show the user the data that has been collected.

3.1.6 Proving Feasibility

For the system we are working toward creating, information is going to be read in from a DAC, much like with HeetShield's current system. Conveniently, there is an entire site dedicated to the documentation of DAC and Python use that will allow us to know what we have, what we need, and what to change in regards to Python and integrating it into our software. There are also extensive docs for Qt as well as PySide which is the python wrapper for Qt. With this information we can use Python to get our thermocouple data read into our system and displayed to the user.

3.2 Issue 2 - Temperature control

3.2.1 Intro

Temperature is one of two main pieces of their system that our application has to control. The first problem our program aims to solve is being able to control the power supply to regulate the temperature. The other problem is reading in the temperature data from the DAC that is connected to the thermocouples. If our system can solve both of these issues, the system will be able to be used much more efficiently.

3.2.2 Desired Characteristics

The perfect system would be one that is able to accurately get to any pressure inputted by the user from 25 degrees C to 600 degrees C. The operator should be able to input the desired temperature in the GUI and be able to leave while the program works to bring the test chamber to the specified temperature. It must also be accomplished in a cost effective manner both in our solution and operation. The automation must also be faster and more accurate than if it was done by a person. This is important because the automation will speed up the process allowing them to do more tests in a day.

3.2.3 Alternatives

The most likely solution to automate the power supply is that the power supply needs to connect to a microcontroller via a 25 pin connector and cable that then connects via USB to a PC. This requires software for the microcontroller and PC. There are a variety of microcontrollers to evaluate. Our team will be working with Professors David Cole and Carlo daCunha who have the subject matter experience (SME). Our team did not have the SME contact information until March 27, and thus we are still working on the details.

3.2.4 Analysis

Our team will complete the analysis based on the information gathered from the subject matter experts.

3.2.5 Chosen Approach

The chosen approach will be determined in conjunction with recommendations from subject matter experts.

3.2.6 Proving Feasibility

In order to prove feasibility the team will write a small test program for the microcontroller to drive the 25 pin connector to ensure that we can drive the 25 pin connector to its specified limits and to its specified accuracy. The software is dependent on the specific microcontroller that is chosen. There may be electrical engineering tasks that are outside the expertise of the student team members.

3.3 Issue 3 - Pressure control

3.3.1 Intro

Pressure is the other main variable that our system has to control. The problem is twofold and stems from (a) the fact that the system cannot control the pressure from software and (b) there are two gauges – one for high pressure and one for low pressure. The low pressure dial is digital, and thus currently readable, but the high pressure dial is not digital, so not currently readable. Solving both of these problems will make the processes go faster.

3.3.2 Desired Characteristics

The perfect system would be one that is able to accurately get to any pressure inputted by the user from 580 torr (atmospheric pressure) to 0.1 torr (near vacuum). The operator should be able to input the desired pressure in some GUI and be able to leave and the apparatus gets brought to it. It must also be accomplished in a cost effective manner both in our solution and operation. The automation must also be faster and more accurate than if it was done by a person. This is important because the automation will speed up the process allowing them to do more tests in a day.

3.3.3 Alternatives

There are two main alternatives to solving this problem: buying a new vacuum pump or valve, either of which would need to be able to be electronically controlled. Additionally the hardware that controls the pressure needs to directly interface with the PC or have a simple enough interface that the chosen microcontroller can control it. Our team will be working with Professors David Cole, Carlo daCunha, and Constantin Ciocanel who have the subject matter experience. Our team did not have the SME contact information until March 27, and thus we are still working on the details.

3.3.4 Analysis

Our team will complete the analysis based on the information gathered from the subject matter experts.

3.3.5 Chosen Approach

The chosen approach will be determined in conjunction with recommendations from subject matter experts.

3.3.6 Proving Feasibility

With the expectation that the required hardware will be expensive, in order to prove feasibility the team will walk through the specifications of the chosen hardware to ensure that they are within specified ranges. There may be electrical engineering tasks that are outside the expertise of the student team members.

3.4 Issue 4 - Notifications

3.4.1 Intro

With the end goal being automation of their current system we need a way to inform the operator when the test parameters are set. We would also need it to inform the operator in case something goes wrong. This would be mostly a quality of life feature that helps alleviate the need to constantly check.

3.4.2 Desired Characteristics

The desired results would be a way to inform the user of the system being set up from anywhere at any time with customization so the operator can choose how to be notified. The list of options for notifications would be checkboxes, toggle switches or some other way to show which kind of notifications are turned on in the GUI. This is an important feature because it provides ease of use and allows for flexibility on the operator's end to do other tasks without worrying about letting the test environment sit there while ready.

3.4.3 Alternatives

One possible way to notify the operator would be a windows notification. This was brought up in a meeting as a possible solution when talking about how our system is going to work. This would be a part of our code base using the win10toast or win11toast library for python depending on what version of windows is running. Win10toast was made by Jithu R Jacob in 2018 and win11toast was made by Tomofumi Inoue in 2022. They have been used in many applications that want to have a pop up notification including but not limited to games, download progress, and input gathering.

A light notification is another option that can be used. This is a common solution for a situation like this and if we use a LED light we could use different colors to represent different states it is in. We would use gpiozero because of its Raspberry Pi support. It was made by Ben Nuttall and Dave Jones in 2015. It has been used by numerous projects due to it being installed by default in the Raspberry Pi OS desktop image.

Sound notifications are another possibility that could be done. This is another common way to notify people. We would accomplish this by using the winsound library. Winsound is provided by windows and has been since 1994. It is the default windows sound output library for python so has been used by many projects.

The last notification type we could do would be a phone notification. A phone notification would inform the operator of its completion from anywhere as long as they have their phone on them. This would be accomplished by using the notify-run library to send the notification. The notify-run library was made by Paul Butler and released in 2018. Some applications that use notify-run are Jupyter Notebooks, Keras and many other web services.

3.4.4 Analysis

All of these notification systems have their uses and could be used. The want for multiple that the operator can choose from would mean we could use all but with limited time and resources we would need to choose ones, at least initially, with little overhead and have many upsides with little overlap between them. Windows notifications are already a part of windows and the use of them requires no additional items besides importing a library in python. Light notifications have less additional elements than originally thought of since we are already going to be using a Raspberry Pi so the only additional element would be the LED light. Sound notifications can be accomplished using the laptops speakers however if the system is swapped to a computer with no speakers then an additional item would need to be purchased. Phone notifications would require the operator to have a phone which most people do but if theirs is dead or left at home then there is no way for them to be notified.

3.4.5 Chosen Approach

The pros of windows notifications are that they are simple, get the point across and highly customizable and could even take user input after it reaches the goal to change what parameters they want. The cons are that if you are not at the computer you would not be able to see that it is done testing and the notifications are kinda small and can be not realized initially since they are similar to regular windows notifications. Light notification solves the need to be there problem with the pros being: can be seen from a distance, can have different lights mean different things and is also able to utilize the Raspberry Pi. The cons are that it can break and leds tend to be small without some complex system. The pros of sound notification are they can easily get someone's attention and can be customized. The cons are it can be missed if there is a louder noise happening, teaching someone new what the sounds mean could be difficult, and could get complaints from other businesses nearby if done wrong. The pros of phone

notifications are they can be seen from anywhere, the user can customize how loudly it notifies them by changing the volume on their phone. The cons are that the user needs to have their phone on them and phones are used for other things so they might receive notifications for other things and think it is our system.

	Ability To Notify When Done	Ability To Notify When Goes Wrong	Use With Other Notifications	Lack of Additional Items	Additional Uses
Windows	3	3	5	5	5
Light	4	4	4	4	4
Sound	4	3	4	4	3
Phone	4	4	3	2	4

The table shows ratings of different uses wanted for the notification system. Windows notifications overall score a 21, the light notifications scored 20, sound notifications got a 18 and phone notifications scored a 17. Based on this our initial notification system will have both windows notifications and light notification. If time and resources allow we would next add phone notifications then lastly sound. We chose this because windows and light notification provide a reliable notification system in most cases. The reason why phone notifications would be added next instead of sound notifications is because most of the pitfalls of windows and light notifications relate to distance from the system which would be fixed with phone notifications the best.

3.4.6 Proving Feasibility

Feasibility of Windows notifications can be done easily with a small program showing them working however with a lack of a Raspberry Pi and a LED light it is hard to show that it actually works. The following code is a test of windows notifications to show they work. A similar program would also work for the LED light. Further tests will have a Raspberry Pi and the LED light to test and will take input from both a GUI and take input from the temperature and pressure readers. Further tests will also make sure the notifications stay until user input removes them.

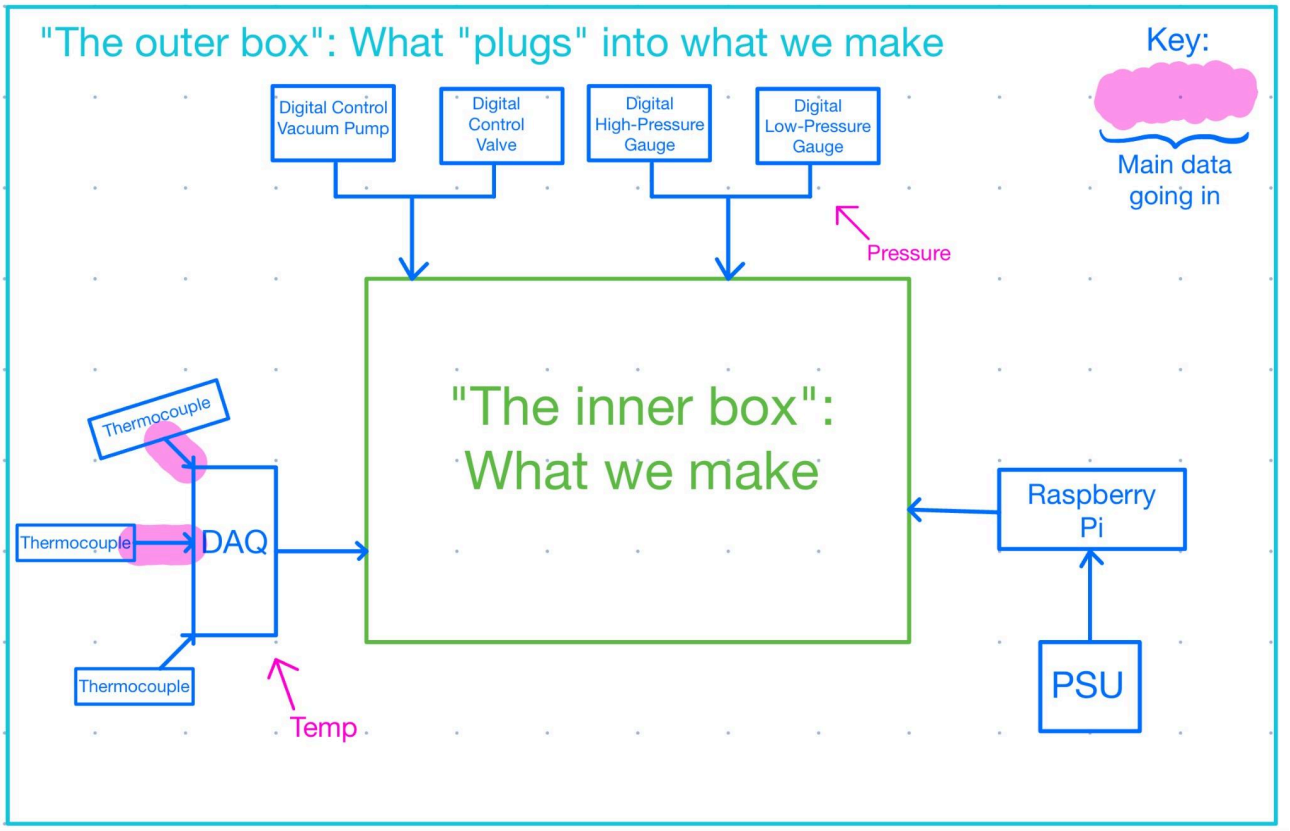
```
# Windows notification test
import win10toast
def check_condition():
    current_temp = 0
    current_pres = 0
    desired_temp = 10
    desired_pres = 10
    while current_temp != desired_temp and current_pres != desired_temp:
        # increment pressure and temp
```

```
        current_pres = current_pres + 1
        current_temp = current_temp + 1
        # check if conditions met
        if current_temp == desired_temp and current_pres == desired_temp:
            return True
    return False
# create a toast notifier
notifier = win10toast.ToastNotifier()
# check the condition and display the notification
if check_condition():
    notifier.show_toast("SUCCESS!", "temp and pressure met")
```

4. Technology Integration

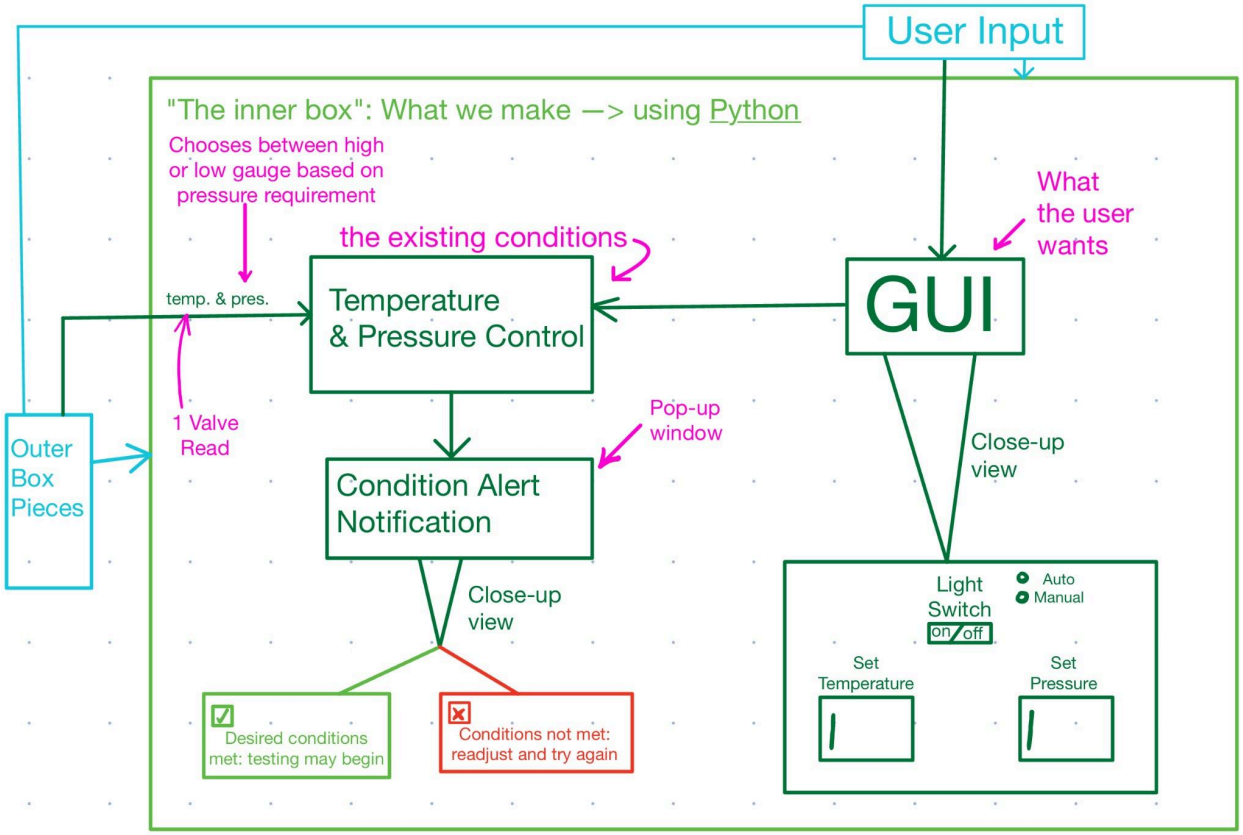
To solve HeetShield's challenges of the automation of precise temperature and pressure control, we have designed a system where hardware and software components work in conjunction with each other. Our system consists of three main parts: a user interface for input, a control system to automate the process and a notification system for feedback. By creating an extension for LabView and ways to regulate temperature and pressure, we can enable precise temperature and pressure measurements while automating the adjustments needed throughout the process eliminating the need for manual input. This would ensure the testing apparatus will reach the test environment with minimal human interaction while still being as precise as needed.

Figure 1 demonstrates "the outer box" which is all the external components that would be needed for this. It illustrates what data we would be intaking that being pressure and temperature in real time multiple times a second. This data will then be used to control other external components like the pressure valve or pump and a Raspberry Pi which would control the PSU. The system is a closed loop system so it will be able to correct itself every time it reads data that is not the desired test environment.



4.1 Figure 1: Display of outer components that will influence the software we create

Figure 2 shows “the inner box” which is all the parts that are in the user machine. It shows that the GUI takes user input then that input is sent to the temperature and pressure control. The temperature and pressure control also intakes data from “the outer box” to compare to the user input to match them to get the desired test environment. The temperature and pressure control will output notifications using the notification system to indicate whether the test environment was set up correctly or failed to. The GUI will have fields for the user to input for desired temperature and pressure and how they want to be notified about the state of the environment.



4.2 Figure 2: Display of the inner elements that compose the software that we make

By unifying these elements into a single architecture we deliver a solution that is not only faster and more accurate than manual processes but also leaves room for further improvements. The architecture allows for further improvements, such as remote monitoring, while addressing the current problems of manual intervention and data handling.

5. Conclusion

In closing, the development of an automated testing system for HeetShield's thermal protection materials addresses a critical need for precision and efficiency. The challenges of manual temperature and pressure control which are time consuming and prone to human error have been systematically addressed by our three layer architecture combining a Python control software, real time data monitoring and context based notification system. Our system uses National Instruments' Python API for precise thermocouple data gathering, implements a closed loop control for maintaining extreme temperature and pressure conditions, and incorporates a multitiered alert system to maximize the operator's knowledge of what state the system is in. The system design not only solves immediate problems but has been deliberately designed to allow for future expansion. With further prototyping and SME consultation as our

immediate next steps, this project represents a significant advancement in the setup of test environments that will enable HeetShield to deliver even more reliable protective materials for extreme conditions. The integration of these technologies creates a robust foundation that meets current operational requirements while positioning HeetShield for future growth.