Data Acquisition System for Autonomous Load Testing

User Guide

Issue 1

TABLE OF CONTENTS

0 PREFACE – PLEASE READ FIRST

0.1 PURPOSE OF THIS DOCUMENT

1. *This document serves as a user guide for the beta version of a computer based system to autonomously control a static load test while simultaneously logging data. This project was developed during my extended internship at Berkel and Company Contractors beginning in July of 2018 and ending in January of 2019. The system exists in a beta version and this document should assist whomever is assigned to complete the project.*

1. INTENDED READERSHIP

1. *This User Guide is intended for any Berkel employee who is assigned to continue development of this project, as well as anyone who has been given explicit permission from Berkel.*

2. *For each identified category of users:*

- *While software and hardware knowledge may be necessary to meet the demands of this project, this document attempts to explain critical functionality of the work that has already been done.*

2. APPLICABILITY STATEMENT

1. *This user guide applies to DAQ V 1.1*

3. PURPOSE

1. *This system was created to replace the current Data Logging software and hardware used by Berkel, as well as providing functionality to autonomously run static load tests. This document should explain key functionality and serve as a comprehensive guide to using the system.*

7. PROBLEM REPORTING INSTRUCTIONS

1. *If problems are discovered, feel free to contact me at chrismoranda@gmail.com. Include Berkel in subject line.*

2. OVERVIEW

1. *More often than not, before production piles can be installed on a job site, it is necessary to install test piles. These piles serve as an indication of soil to structure interface. The two common tests that are performed are dynamic load tests and static load tests. The system described in this guide focuses on the latter, although future developments may allow for a machine capable of acquiring data during a dynamic test.*

2. *While there are many electronic components at work in this system, the basic workflow is as follows:*

    A. *Electrical Signals are transmitted from various sensors to their inputs in the system.*

    B. *These signals are conditioned to 0-10 V*

    C. *The analogue outputs are converted to digital numbers*

    D. *These numbers are processed by the Raspberry Pi (Mini Computer) and converted to their symbolic values (Inches, PSI, Kips, etc…)*

    E. *The numbers are then logged and if the system is being ran autonomously, then the Raspberry Pi will control the Hydraulic Pump via Relay accordingly.*

3. *This system was designed to take readings from 3 sensor types. The specification followed includes 6 Linear Variable Differential Transformers (LVDTs) which output a 0-10V signal, 4 Pressure Transmitters which output a 4-20mA signal, and 4 Strain Gauge Load cells which output mv/v outputs. The sensors are all conditioned with different Dataforth 5B signal conditioning modules.*

3. INSTRUCTIONS

▪

4. HARDWARE DESIGN

4.0 Parts List

Part Name	Official Name	Qty.	Figure #
LVDT Panel Mounts	Amphenol Industrial MS3112E12-3S	6	1
Load Cell Panel Mounts	Amphenol Industrial MS3112E10-6S	4	2
Pressure Gauge Panel Mounts	Amphenol Industrial MS3112E8-4S	4	3
Pelican Case	Pelican 1600	1	4
AC/DC Power Supply (120AC/ 24DC)	Mean Well NDR-120-24	1	5
24-5V Converter	TOBSUN 15W DC-DC Converter	1	6
Din Rail	Erayco 7.5” DIN Rail	10	7
16 Channel Backpanel	Dataforth SCMPB01-2	1	8
Ribbon Cable	Dataforth SCMXCA004-01	1	9
Ribbon Cable to Screw Terminal	Dataforth SCMXIF-DIN	1	10
Strain Gauge Signal Conditioner	Dataforth SCM5B38-05	4	11
Pressure Gauge Signal Conditioner	Dataforth SCM5B42-01	4	12
A-D Converters	ADC1115 Breakout Boards	4	13
USB-USB Bulkheads	Startech 1ft. Panel Mount USB Cable A to A	3	14
Raspberry Pi DIN Rail Mount	DIN Rail Mount for Raspberry Pi	1	15
HDMI Bulkhead	Neutrik NAHDMI-W 568	1	16
AC Power Entry	Shurter 3-109-573	1	17
Resistors for load cell	4.7K ohm Resistors	12	18
Relay for Hydraulic Control	Sainsmart 4-Channel Relay Module	1	19

Part Name	Official Name	Qty.	Figure #
Multi-Colored Wire	Nano-Shield NS085 22AWG	1	20
Raspberry Pi 3b+	Raspberry Pi 3b+	1	21

FIGURE 0: EXTERNAL INPUTS TO DATALOGGER

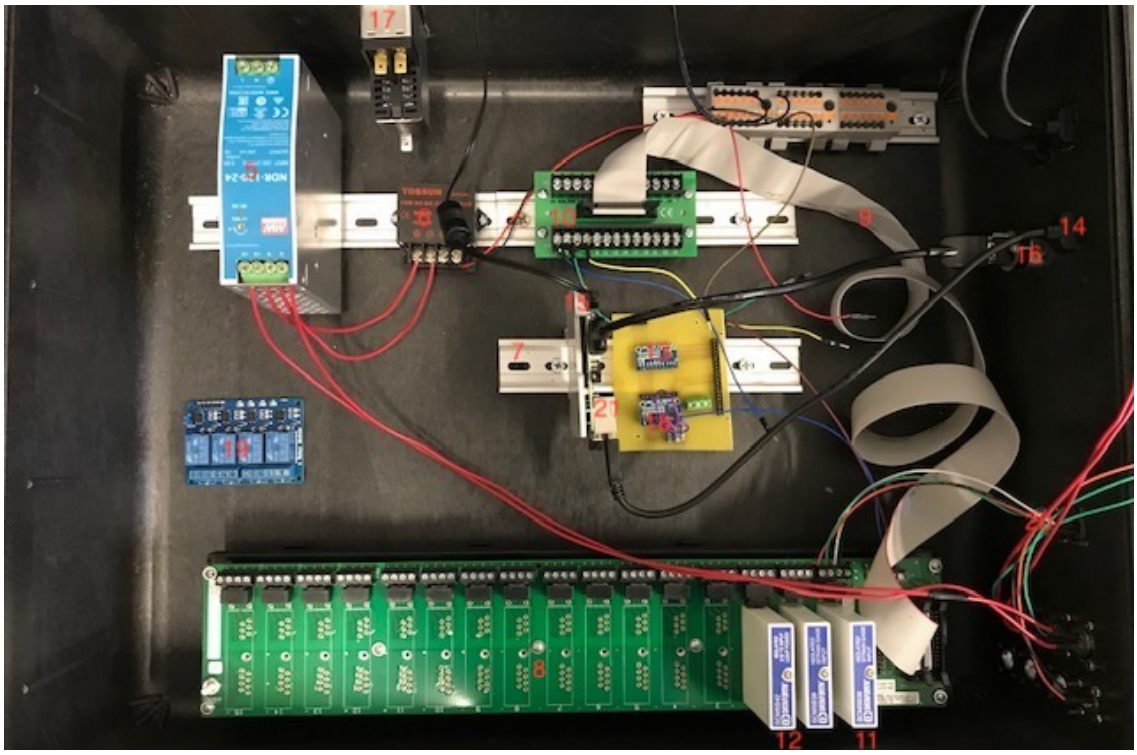


FIGURE 1: INSIDE COMPONENTS OF DATALOGGER

4.1 Discussion on parts

LVDT, Load Cell, and Pressure Gauge Panel Mounts:

All of these external panel mounts were selected to keep consistent with the original RST Systems Data Acquisition System as well as the current sensor connectors. The LVDT connector supports a 3 pin connection, 1 wire for 24V excitation, 1 wire for 0-10V output, and a common ground for both excitation and output. The Load Cell panel mount contains 6 pins, but only uses 4 of them. The red and black wires excite 10V to the load cell, and the Green and White wires transmit the mv/v output signal of the load cell to the signal conditioner. Finally, the Pressure Gauge panel mount is used to connect the 4-20ma signal outputted by the pressure transmitter to the signal conditioners. This 4 pin connector only uses 2 pins for a 2 wire transmission. The Red and Black wires serve to excite the pressure transmitter and serve as a current loop.

Power Supplies and Converters:

This system requires 24V and 5V and a common ground for operation. On the backside of the case, there is an input for AC power. This can take a power chord from wall power or generator power. This connects directly to a 120-24 AC/DC converter. The system runs entirely on DC power. The 24V output from the AC/DC converter plugs directly into a 24V-5V DC-DC converter. The 24V supply is used to provide excitation voltage to the LVDTs. The 5V supply powers the Raspberry Pi, the Relays, the A/D chips, and the Backpanel (which powers the remaining sensors). All systems share a common ground.

Raspberry Pi and Bulkheads:

The system is controlled by the Raspberry Pi 3B+ miniature computer. An earlier version was designed with the newly released Raspberry Pi 4, but due to unresolved issues with the new model, the project was reverted back to the Raspberry Pi 3B+. This serves as the lightweight “brain” of the Datalogger. While inexpensive and very portable, the Raspberry Pi is a powerful machine capable of handling the workload required of a load testing system. The Pi serves the following roles:

1. *Collects digital readings as the final stop for sensor data.*
2. *Runs the Graphical User Interface for the entire test.*
3. *Converts digital readings into meaningful sensor data that can be logically analyzed to control load test.*
4. *Controls load test by controlling hydraulic pump via relay.*
5. *Saves data file in flash memory that can easily be saved on external flash drive.*
6. *User can play Minecraft during downtime when load test is taking a long time :).*

The Raspberry Pi can be controlled via external USB bulkheads in which a mouse and a keyboard can be plugged into. Three USB ports are available on the outside of the box so the user can also insert a flash drive to collect data. An HDMI port is also available through

external bulkhead to connect the computer to a monitor. On a side note, a good future development to cut down on equipment would be to install a Raspberry Pi suitable LCD screen directly into the box so the system could exist as a contained unit.

DIN Rails:

For organizational and structural purposes, DIN rails have been mounted throughout the bottom of the pelican case. This guided the selection of components so that everything could be mounted to the rails. For example, the backpanel, power supply, Raspberry Pi, and screw terminal connector all have appropriate DIN mounting attachments. While seemingly trivial, it is easy to forget about this detail. Ensure that the models selected for contain DIN mounting options.

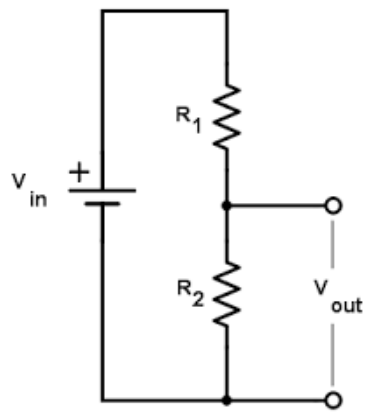
Signal Conditioning:

Possibly the most research intensive aspect of this project was delving into the intricacies of Signal Conditioning. I highly recommend the National Instruments Signal Conditioning Handbook to help understand some of these complexities. Essentially, every sensor used for this test transmits data as a different flavor of signal. These signals need to be "conditioned" in some way before they are ready to be interpreted by the program. Signal Conditioning is simply defined as manipulating the signal to prepare it for a further stage in processing. This could include amplifying the signal, converting it to digital, dividing the signal, or any number of things that make the signal acceptable for use.

LVDT

The simplest of these output signals comes from the LVDT. The LVDTs used by Berkel actually do not require any signal conditioning other than Analog to Digital conversion as they just transmit a 0-10V output directly proportional to 0-4 inches multiplied by some calibration factor. However, due to the selected A/D converters having a maximum input voltage of 5V, some signal conditioning was required. Taking inspiration from the RST system, I chose to use a voltage divider network. I think it is common for high precision A/D converters to have a maximum input of 5V, so this isn't an uncommon method. A voltage divider is as simple as it sounds. Take for instance the figure below:

If we take R1 and R2 to be equivalent (the actual system uses 2 4.7kohm resistors), then Vin will drop exactly half of its voltage across each resistor. Therefore, Vout will be Vin/2. If Vout is the value that is read by the converter, then we know it is exactly half of the input voltage. Using real values, if Vin is the output of our LVDT at 8V, then the ADC will see 4V. This is an acceptable value because our ADC has a Maximum 5V range. Adding some simple logic in our Data processing algorithm to simply multiply the output by 2 before converting to inches is all we need to do to make sense of this value. When the system sees 4V, it can then convert this value to 3.2 inches (assuming ideal calibration factor).



Pressure Gauges:

The pressure gauges output a 4-20ma signal. This is a very common signal for sensors that transmit data over long wires, as it degrades very little over long distances. This signal is directly proportional to whatever the range of the sensor it represents is. For instance, the hydraulic pump used for load tests is rated for 10,000 PSI. 0 PSI would be represented by 4ma and 10,000 PSI would be represented by 20ma (assuming a perfectly calibrated transmitter). Fun fact, a lot of people will refer to these electronic pressure gauge attachments as Pressure "Transducers". Technically they are Pressure Transmitters as transducers output a voltage and transmitters output a current in amps. Even the calibration sheets refer to them as Pressure "transducers", although the data is represented in milliamps. As is a common theme in signal conditioning, it's extremely beneficial to transform these signals to a small analog voltage that can then be converted and analyzed. This could be done, by just placing a resistor network on the output and using the voltage drop to get the signal voltage, but a much more sophisticated and reliable way to do this is to purchase a signal conditioning module developed by industry professional. For the pressure gauges, the *Dataforth SCM5B42-01* 4-20ma signal conditioning module proved to be the device for the job. This cartridge can excite the transmitter and process its output signal. It then outputs a 1-5V signal proportional to the 4-20 signal (i.e. 1V = 4ma = 0PSI and 5V = 20ma = 10,000 PSI). This signal is now acceptable for use by the ADCs and our conversion formula.

Load Cells:

The load cells used by Berkel are strain gauge load cells, which means that the signal outputted is a mv/v signal. The meaning of this was a bit of a puzzle in the beginning, but is actually pretty straightforward. All that is meant by mv/v is that a signal is outputted proportional to the voltage in which it is excited. For instance, the 600 Ton and 1200 Ton load cells are 2mv/v and 3mv/v respectively. This means that if the 600 Ton load cell was excited at 1V, then its maximum output would be 2mv. If it was excited at 10V, then the maximum output would be 20mv. Load Cells are a little more complicated than the other sensors, because they do not necessarily follow a linear/proportional correlation. A 10mv output may not directly correlate to half of the load of a 20mv output for instance. Instead, a 6 point and 2 point curve are given for the 600 Ton and 1200 Ton load cells respectively. All that this means, is that you must plug in the correct equation to translate mv/v to kips. The calibration sheets will generally provide coefficients C0-C5 and C1 and C2. If using the 600Ton load cell, you use this equation: Load(kips) = C0 + C1x + C2x^2 + C3x^3 + C4x^4 + C5x^5, where x is the output in mv/v. For the 1200Ton load cell, the equation is just: Load(kips) = C1 + C2x, where x is again the output in mv/v. This makes more sense if you plug in some of the values from the calibration sheet into this formula and compare the outputs. The software I have written takes care of all of this math, and expects a 5V maximum output that correlates to a maximum 2mv/v or 3mv/v output. This is handled by another Dataforth Signal Conditioning Module, the SCM5B38-05 Strain Gauge Module.

Analog to Digital Conversion:

The final step in the signal conditioning chain proved to be the most frustrating aspect of this project. All of the previously mentioned signal conditioning techniques are useless if the final

value cannot be read or processed. The Analog to Digital Converters therefore need to be precise and accurate (these do not mean the same thing), durable, and capable of handling the desired number of channels (6 LVDTs, 4 Pressure Gauges, 4 Load Cells). There were many candidates that I will mention below. However, first, some brief discussion on Analog to Digital Conversion.

There are a number of techniques to perform this conversion, but delving into them would take some knowledge of Electrical Engineering fundamentals and some Google searches. Here are the basics. Analog to Digital conversion “precision” is given in bits. This essentially tells the user what the smallest voltage you could expect to read is. The equation is essentially, smallest possible voltage = [range of voltages/2^#of bits]. If this seems confusing, it helps to think of it like this: Numbers are stored in binary as bits, there are 2^# of bits unique binary numbers for a given # of bits, and this is how many unique identifiers you have to represent the range of voltages. For instance, an ADC with 12 bits of precision and a 10V range (let’s say it can read -5V - 5V) yields a smallest hypothetical reading of 10/2^12 or 2.44mv. Now, it is very unlikely that a reading that small would be accurate, but it is the smallest possible representation. It is possible that an ADC with high precision has low accuracy in which the numbers that it outputs show many significant digits, but fluctuate constantly.

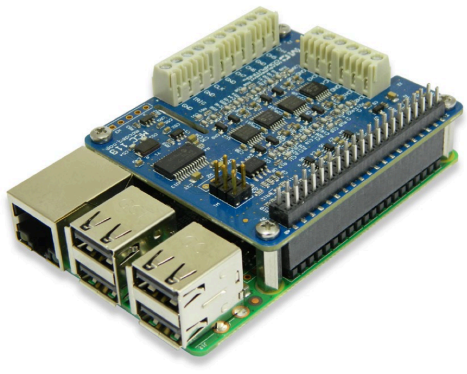
Now that there is some understanding of the metrics in which to judge an ADC, Design Qualifications can be discussed. Due to the constraints, the ADC needed to be able to read up to 14 inputs with a maximum voltage of 10V. This was scaled down to 5V, when it was decided voltage dividing networks were a viable option. The system also needed to read LVDTs accurately within 0.001 inches to meet the testing criteria. This means that ADC would need to be more accurate than 2.5mv as a change in 0.001 inches would correspond in a 2.5mv change in voltage. So, hypothetically an ADC with a 10V range and 12 bits of precision would just barely meet this qualification. Here are some of the candidates that were tested.

Candidate #1:

The Measurement Computing MCC118 daq for Raspberry Pi.

This was actually a fairly good Daqhat with some notable qualities. Firstly, it is developed by Measurement Computing which is a a subsidiary of National Instruments, a very reputable company. Secondly, it fits perfectly onto the Raspberry Pi and even allows for 8 total stacked hats for 64 total analog inputs. Finally, it was a very professional software library that is nicely documented, and well written.

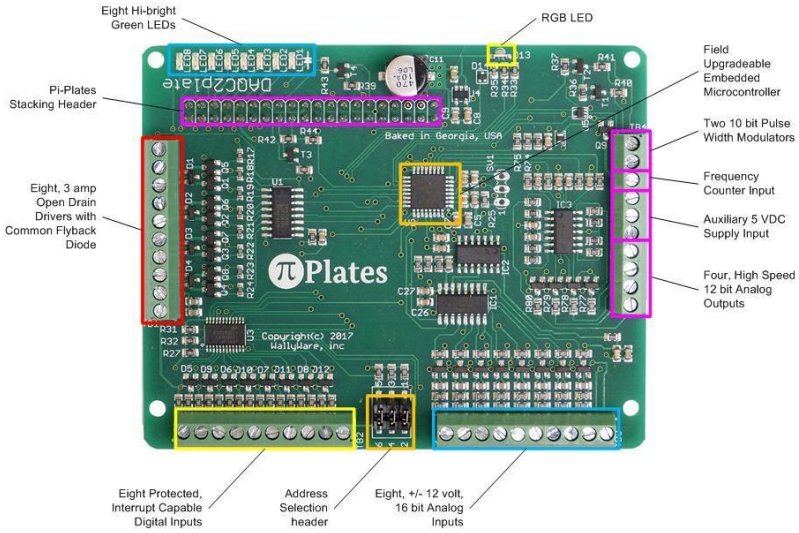
Unfortunately, it only has 12-bits of precision and wasn’t quite good enough to provide stable LVDT readings up to .001 inches.



Candidate #2:

PiPlates DAQ2PiPlate WallyWare:

This 16-bit ADC Hat was able to provide -10V - 10V readings and could also take up to 64 stacked inputs. It provided comprehensive software libraries in python and C, that worked smoothly with the Raspberry Pi. It was very easy to use and integrate and was half the price of the MCC118. They also had excellent customer service that promptly responded to my questions via email, although I suspect this was because they only had 1 or 2 employees. Unfortunately, this hat didn't quite allow for the precision that was desired to accurately read LVDTs.

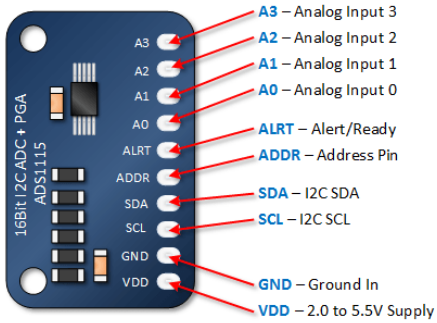


Candidate #3:

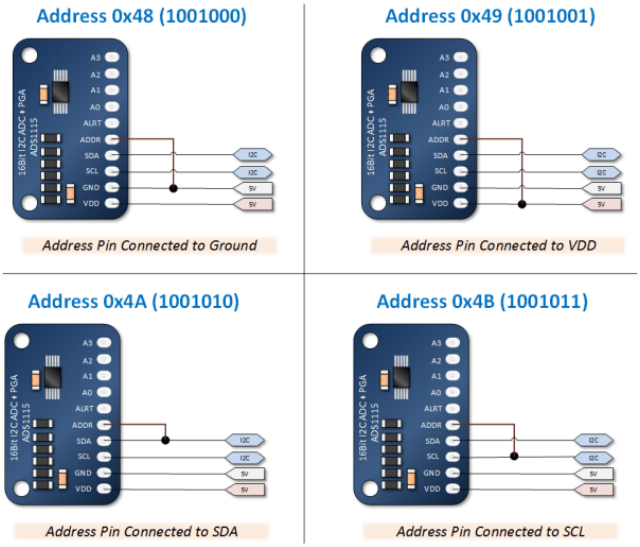
ADS1115:

The cheapest solution by far ended up being the best. The ADS115 is a very common ADC that has 4 channels, but configurable addresses to allow for up to 4 chips with 4 channels each totaling 16 inputs. This ADC has 16 bits of precision, with a range of -5 to 5 volts and ended up giving stable reading up to .001 inches on the LVDTs. It also allows for using 2 channels to read differential voltages. This was useful later when it was discovered the grounding on the backplane was different from that of the power supply (I never discovered why), so readings could be taken between Backplane output and Backplane ground instead of using the common ground that the ADCs shared. This makes more sense when looking at the

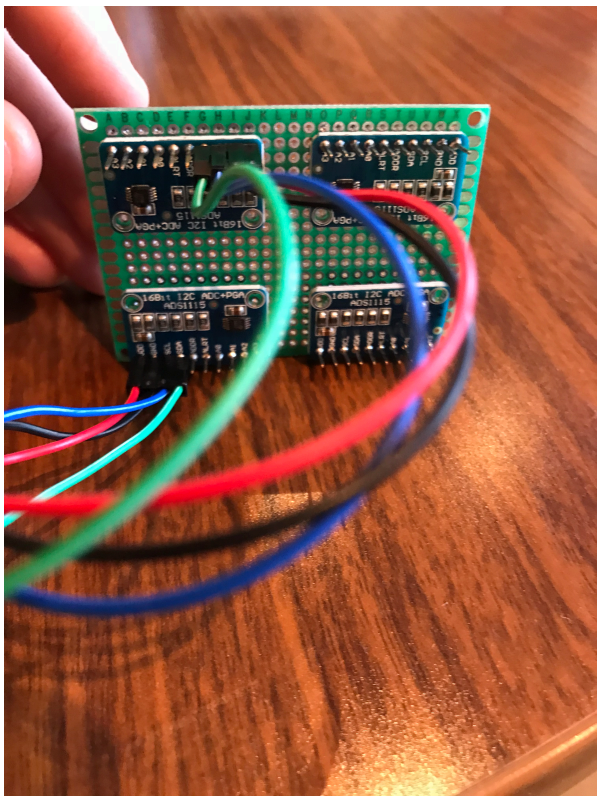
wiring diagram. Unfortunately, to my knowledge, this chip does not exist on a circuit board or hat that is configured with multiple chips. To allow for 14 inputs as was required for this system, it was necessary to design an elegant solution.



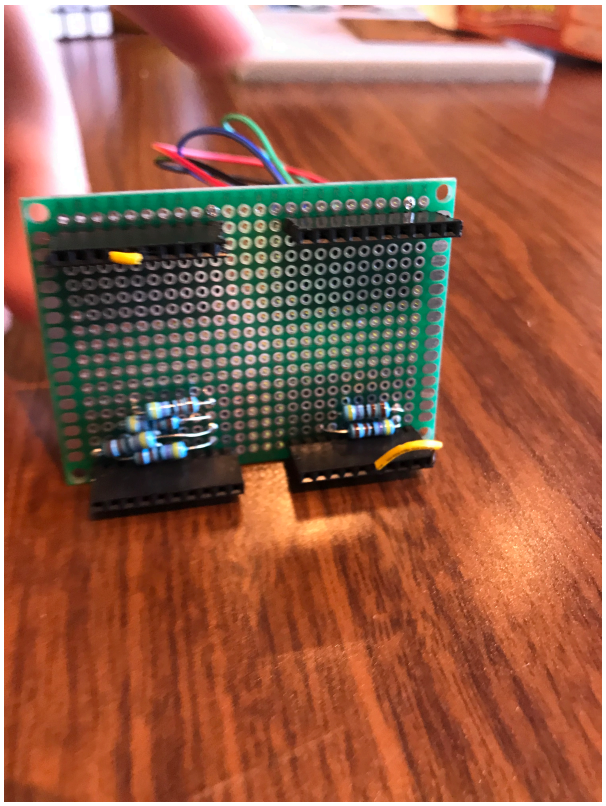
To configure 4 ADS1115 breakout boards, the address pin needs to be configured differently. This allows communication over I2C to refer to different boards when polling for values, as the address virtually changes in software. The addresses can be set using this guide:



Initial attempts at designing a 4 chip ADC resulted in workable, but wire infested messes that did not meet industrial standards of organization or durability. The second iteration consisted of a soldered protoboard that was slightly cleaner, but still not a great option. See below:



4



ADS1115 breakout boards are soldered to this through hole protoboard. The Power, GND, SCL, and SDA pins are then wires together using jumper cables. As is seen in the second picture, the back of the board uses female inputs for jumper cables. The addresses are tied to their respective pins using yellow solid core wire, and 6 resistors are soldered from analog input pins to ground to allow for 6 LVDT signals.

While this worked fairly well, it was not a good permanent solution as the “birds nest” of wires created an overwhelming environment for inputs, and the header pin input setup created a fragile device for industrial conditions. In short, it was not an “idiot proof” solution. As priorly mentioned, there is not, to my knowledge, a neatly packaged circuit board containing 4 accessible ADS1115 chips. Fortunately, etching a Printed Circuit Board (PCB) from scratch has always been a goal of mine. This project provided the perfect opportunity to break out the CAD software and the chemicals and get to work.

Candidate #4: The FrankenBoard

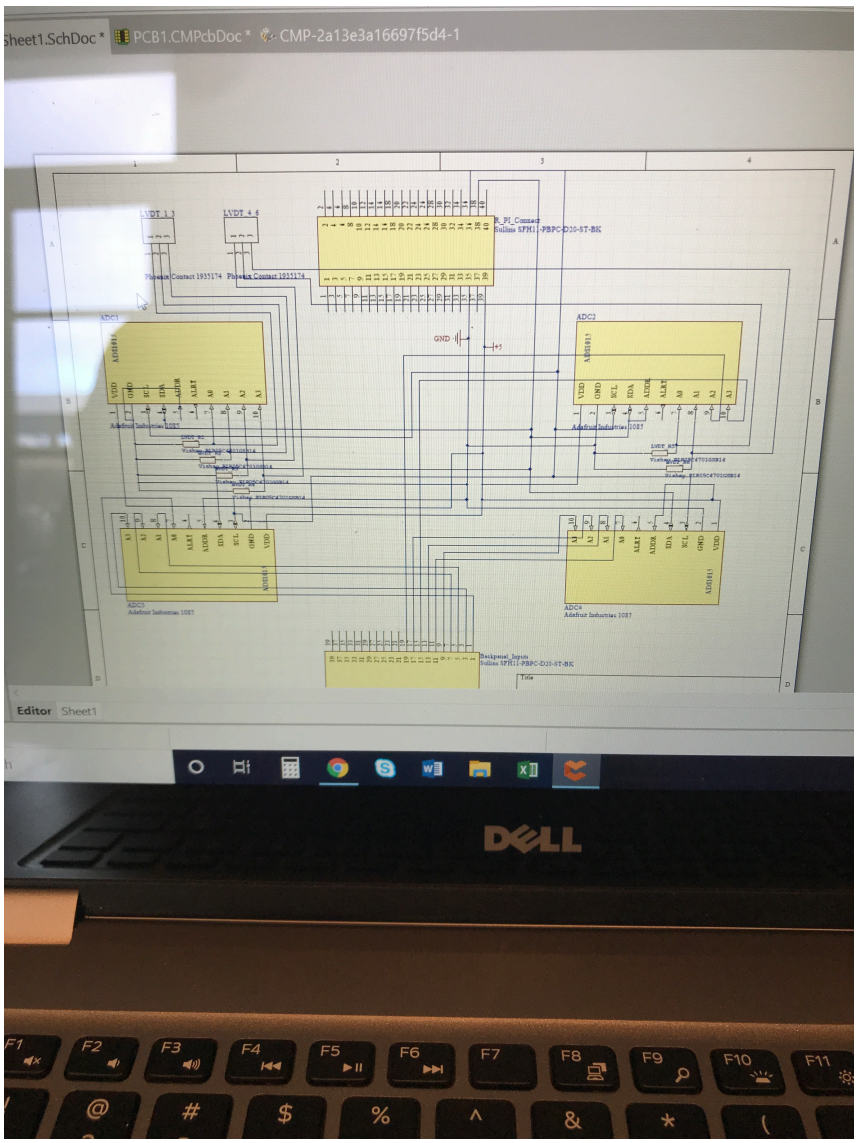
A quick Google search will yield many strategies for DIY circuit board printing/etching, and over the course of a few weeks I believe I attempted most of them. The one thing almost universally agreed upon is the need to design the circuit before putting your hands on any copper plates. Below is a summary of the process I followed, and the tips I have for creating PCBs. Much better guides can be found online, and my end result wasn’t really a better

solution than the aforementioned protoboard, but here are the steps I followed to develop what I will hereby refer to as the “Frankenboard”.

The Schematic:

For all, except perhaps a few brave and foolish souls, the circuit design process begins with creating a schematic. In actuality, one may start with a black box diagram listing the basic inputs and outputs and leaving out any notion of port numbers or signal lines, but since I had already been soldering chips to boards and had a pretty basic idea of my inputs/outputs I didn’t think this was necessary.

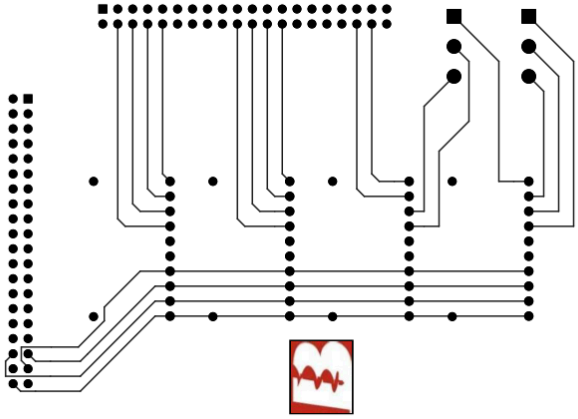
Creating a good schematic starts with selecting a good PCB software. I chose Altium’s design software and would recommend it to the budding hobbyist/enthusiast/Berkel employee that decides continuing this project may be a neat idea. Once you have your software picked, it may help to quickly sketch a simple circuit design with a pencil and paper to get an idea of layout and space. If you don’t, you may end up with something like this…



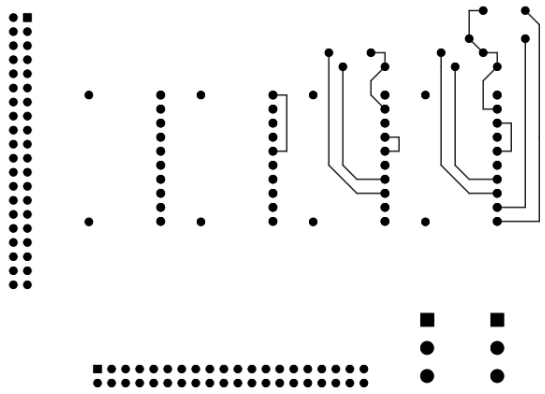
The schematic doesn't have to be perfect, but it should include all components and all of the proper connections. One should also be cognizant of physical limitations when designing the circuit (size of the board and components, placement of wires/traces etc…). My design, for example, includes 2 20x2 header pins, 2 3 pin screw terminals, 4 ADS1115 breakout boards, and 6 4.7kOhm resistors. The original idea was to use two 20x2 ribbon cables to reduce confusion and clutter. These cables would go directly from Raspberry Pi to the PCB and from the Signal Conditioner outputs to the PCB. The screw terminals would take all inputs from LVDTs. This way, there would be no confusion as to which pins went to which ports. After some extra organization, I was able to move on to the next step.

The PCB Sketch:

The next step in my whirl wind guide to homemade circuit boards is to create a PCB schematic. Using the previously drawn schematic as a template, it's time to actually draw the copper traces that will exist on the board. Usually, PCB design software will allow you to import the schematic and use the existing connections to draw traces. This is one reason why it is very important to spend a lot of time early so that creating the PCB sketch is as straight-forward as possible. In my case, it ended up being much simpler to use the schematic as more of a guide than a template and draw my sketch without prior connections made. I ended up greatly simplifying the connections that were needed. Here is my final schematic drawing, and they are free to use and share.



This is the front of the PCB and contains all connections from the Raspberry Pi and from the Backpanel/ LVDTs. The ADCs use shared lines that connect PWR, GND, SDA, and SCL.



The backside of the PCB connects the address ports to the necessary lines to ensure the use of multiple ADCs. This schematic also ensures that 6 resistors can be added to split the LVDT signals from input to ground.

Transfer Process:

Designing a schematic and extrapolating this design to create a PCB sketch are essentially universally accepted steps of the fabrication process. The next part is where “best practices” begin to diverge. The actual best practice seems to be to send your sketch to a company in China that can print 10 boards for $50 and have them back to you within 2 weeks, but while the end product may be professional and polished, something about this seemed to clash with the theme of this entire project. There was just something so exciting about the prospect of doing it from scratch and not relying on a third party to build components for the system. Therefore, in the spirit of idealistic inventors everywhere, I decided to attempt to transfer and etch my own circuit board. After a long and frustrating process with much trial and error, I can honestly say, sending your sketch to China is obviously the best option. Don’t be idealistic, or overly consumed with notions of engineering purity. Don’t reinvent the wheel. At a point in the not so distant past, it may have made economic sense to etch your own circuits, but today it is cheaper than ever to mass produce circuit boards by going through a company dedicated to this process. However, I will still briefly discuss the DIY processes that I attempted, if only to highlight the difficulties that arise when being idealistic.

Heat Transfer

There is much dispute regarding the best way to transfer your schematics to Copper Plated Substrate boards, and to be honest, I don’t have a conclusion on a best way to do this. It seems to depend on many different factors that are hard to account for, and like anything, it appears that experience and practice is the only way to achieve success. The first process I tried is known as the Heat Transfer method. This process consists of printing the schematics onto a special type of paper, and using an old iron to transfer the print to the board. Essentially just place the print ink side down onto the copper board, hold it in place, and run the iron over it for 10-15 minutes. Theoretically, the ink will be transferred perfectly onto the board at the end of this. This method alone has its fair share of disputes. Some “experts” recommend holding the iron in place and pressing down, others recommend ironing the board as if it is a shirt, and one PCB enthusiast recommended taping the board to the iron and using a rolling pin to roll over the board and press the design on. Even after one has decided on how to iron the board, there is still the question of the type of paper to use. Some recommend Glossy photo paper (which did not work well for me), while others recommend sticker backing paper (which definitely did not work for me), and apparently they make “toner transfer” paper which is designed specifically for this application. Another important thing to keep in mind if attempting this method is to use a laser printer that uses toner. An inkjet printer will not work. Even with this insight, however, the heat transfer method was not a success for me.

Cold Transfer

Another method that is popular is to forgo heat altogether, and instead use a chemical solvent to transfer the toner onto the copper. The general idea is print your sketch onto magazine paper (you can cut out a rectangle from a magazine that is large enough to fit the circuit onto

it and tape it to regular printer paper), and use some combination of Acetone and Ethyl Alcohol to transfer the print. My understanding, from very limited knowledge of chemistry, is that the Acetone will completely remove the toner from the paper, and the alcohol will keep it from dissolving completely so that the design remains intact for transfer. Different ratios of Acetone to alcohol were recommended, but none yielded the results that I desired.

Permanent Marker

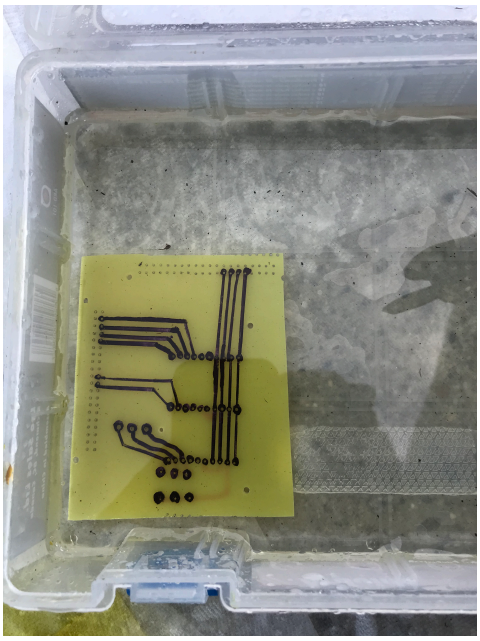
The simplest transfer method is simply to try and sketch your design by hand using permanent marker. This clearly only works if the design is relatively simple or you have surgical precision to draw all of the traces. I ended up using this method after failing to achieve workable results from the other methods. At this point I was desperate to get some result from this arduous task… seriously, just send the schematics to a company in China. This method is as straightforward as it sounds, just draw your circuit as it looks in the sketch. In my case, it helped greatly to pre-drill the necessary holes, and blot them with marker. Another thing to keep in mind when designing/etching circuit boards is that your sketch may be backwards when you actually press it down onto a circuit. Design software generally can account for this and mirror your circuit before or after it is sketched. Just ensure that your circuit isn't backwards before any etching is done.

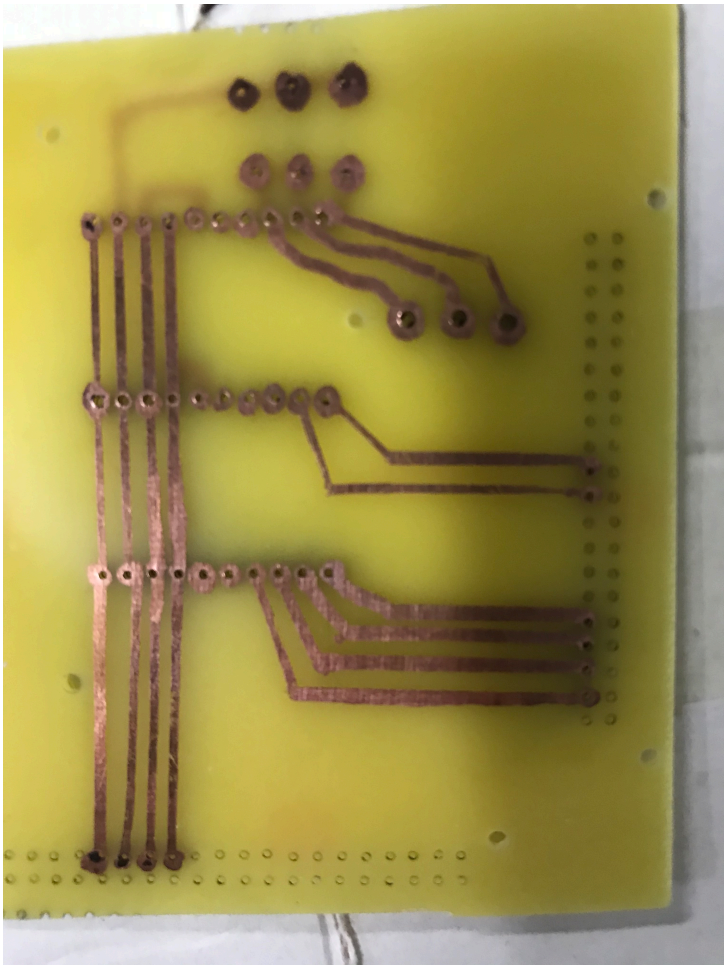
Etching

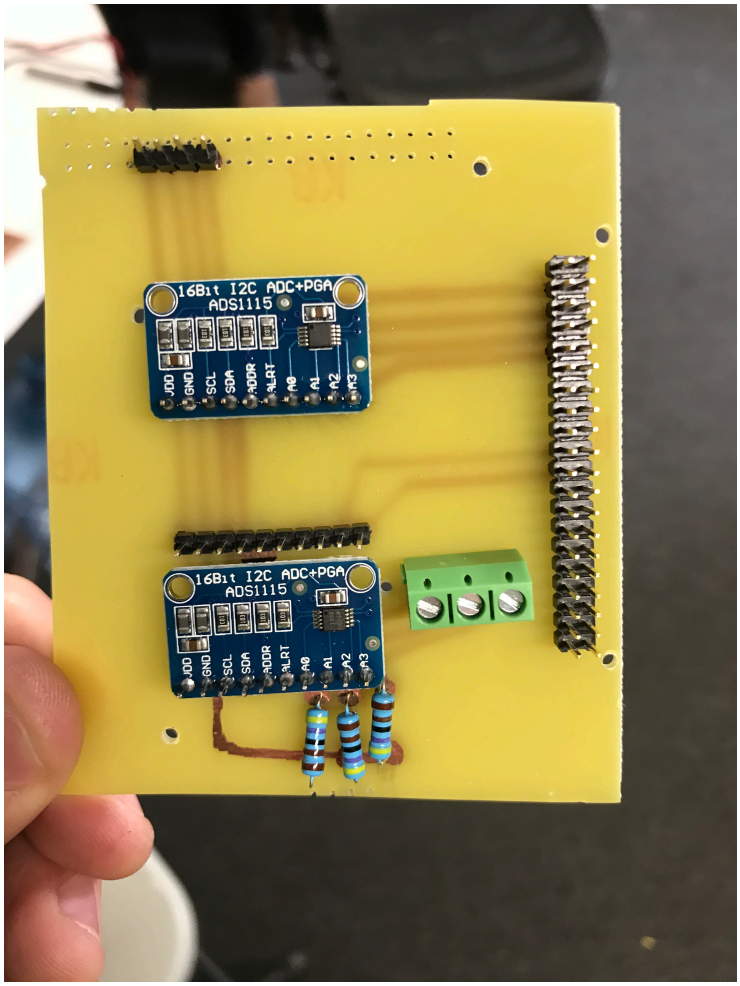
There are different strategies for etching circuit boards, but the most popular and budget friendly way to do this is to use a Ferric Chloride or FeCl3 solution. This chemical can be purchased from Amazon for around $15 as a powder that can be added to water, or a premixed solution. It is important to read about the dangers of this before using it. This chemical bath will remove all exposed copper from your circuit board. After approximately 15 minutes, the only copper that will remain is what was covered by your tracing that you painstakingly transferred to the board. The toner or ink remaining can be removed with acetone, nail polish, or a scrubbing with steel wool. For safety, follow these steps while etching.

1. *Use a plastic container to hold solution and board*
2. *Only use non-metallic items to grab circuit board from bath*
3. *Wear latex gloves and eye protection*
4. *Make sure work station is well ventilated*
5. *Rinse board thoroughly in water*
6. *Neutralize the solution with Baking Soda (Pour slowly and keep adding until the bubbling reaction no longer occurs. This may take a while and will expand so use a large container.)*
7. *Dispose of the remains responsibly (don't pour down drain)*

Below are the unimpressive results of this process:







5. SOFTWARE DESIGN

5.0 Overview

The software designed for this system needed to accomplish a number of things. First and foremost, it needed to replicate the crucial functionality of the existing RST software. This included the ability to read all data from the appropriate sensors, display this data in a user friendly interface, log this data with a timestamp in a file, and allow for changes to calibration factors and zeroing of devices (such as LVDTs). The current system handles all of this, and also allows for modular changes to easily be made if demands are changed. For example if more sensors are added to the system, this can be handled fairly easily with software updates to the GUI. Other improvements include the development of a master calibration sheet. This is essentially an excel sheet that keeps the most up to date record of all Berkel sensor calibrations and divides them by type of sensor (i.e. Load Cell, Pressure Gauges, and LVDTs) and stores calibration factors, coefficients, and serial numbers. This will be hosted on the Berkel web app and can easily be downloaded. The user can then simply enter the serial numbers of each device and have the calibration fields auto populate, eliminating the need to have calibration sheets on hand and the possibility of human error.

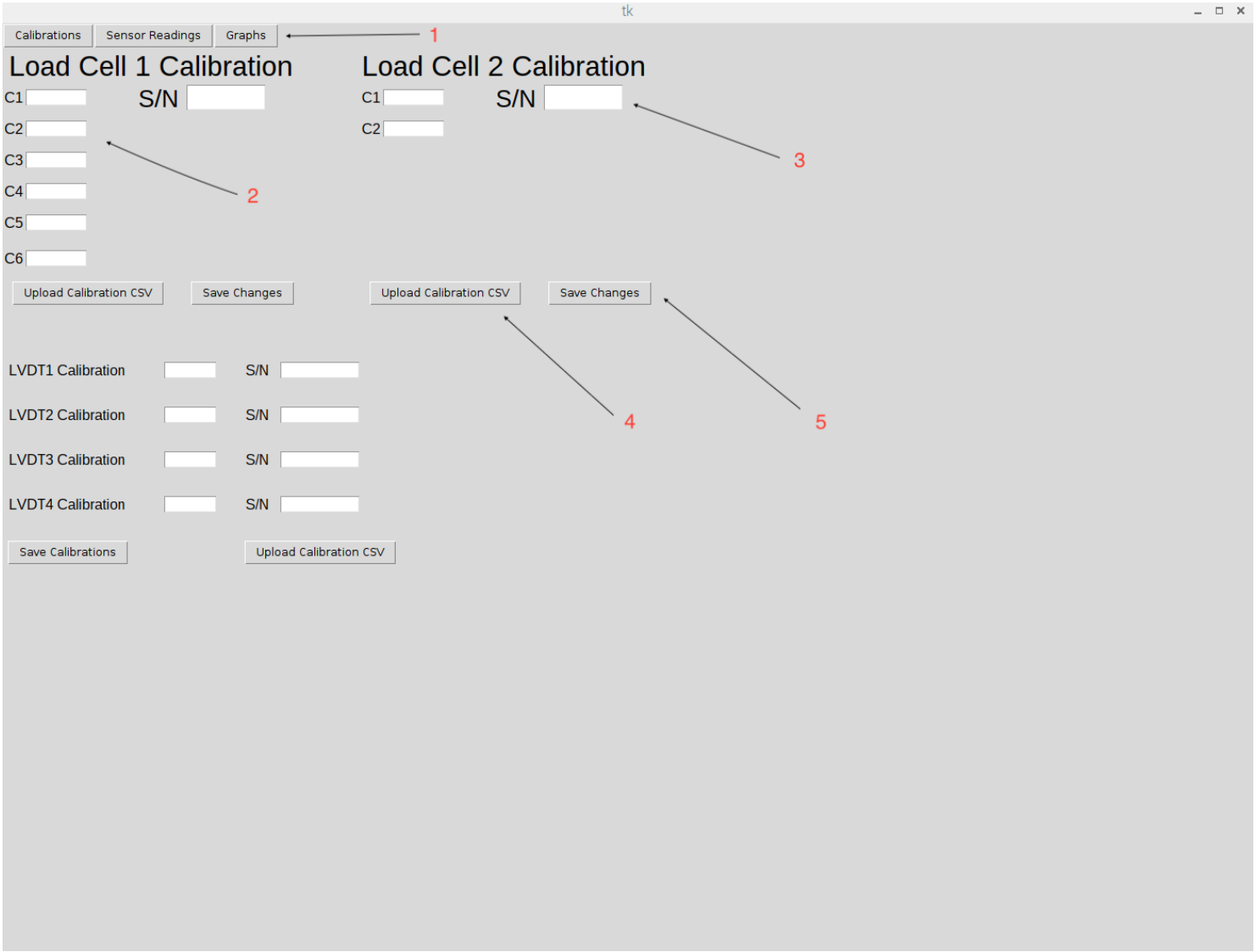
The other requirement, and main motivation for creating this system, was to allow for autonomous load tests. This resulted in the addition of software controlled relays that can control the hydraulic pump, eliminating the need for constant human oversight. Currently in this version of software, buttons have been added to the Interface to allow for control directly from the computer instead of needing to use a remote. There also currently exists a simple scheduler that can take the Desired Load, the Load Increment %, and the Time Increment and run a very simple load test. More testing and logic must be added to this part of the software to ensure safety in the event of an emergency. Most of the remaining work to be done resides in this part of the software.

The software was written using Python and the Python GUI library TKinter. TKinter allows for user interaction with buttons, and display of the interface for sensor readings and graphing capabilities. The GPIO functionality of the Raspberry Pi was taken advantage of to allow for I2C communication from the ADCs as well as the output control of the relays. The on board SD card is utilized to store the data files from each test, although extra memory units could be added to ensure data is not lost.

5.1 Software Guide

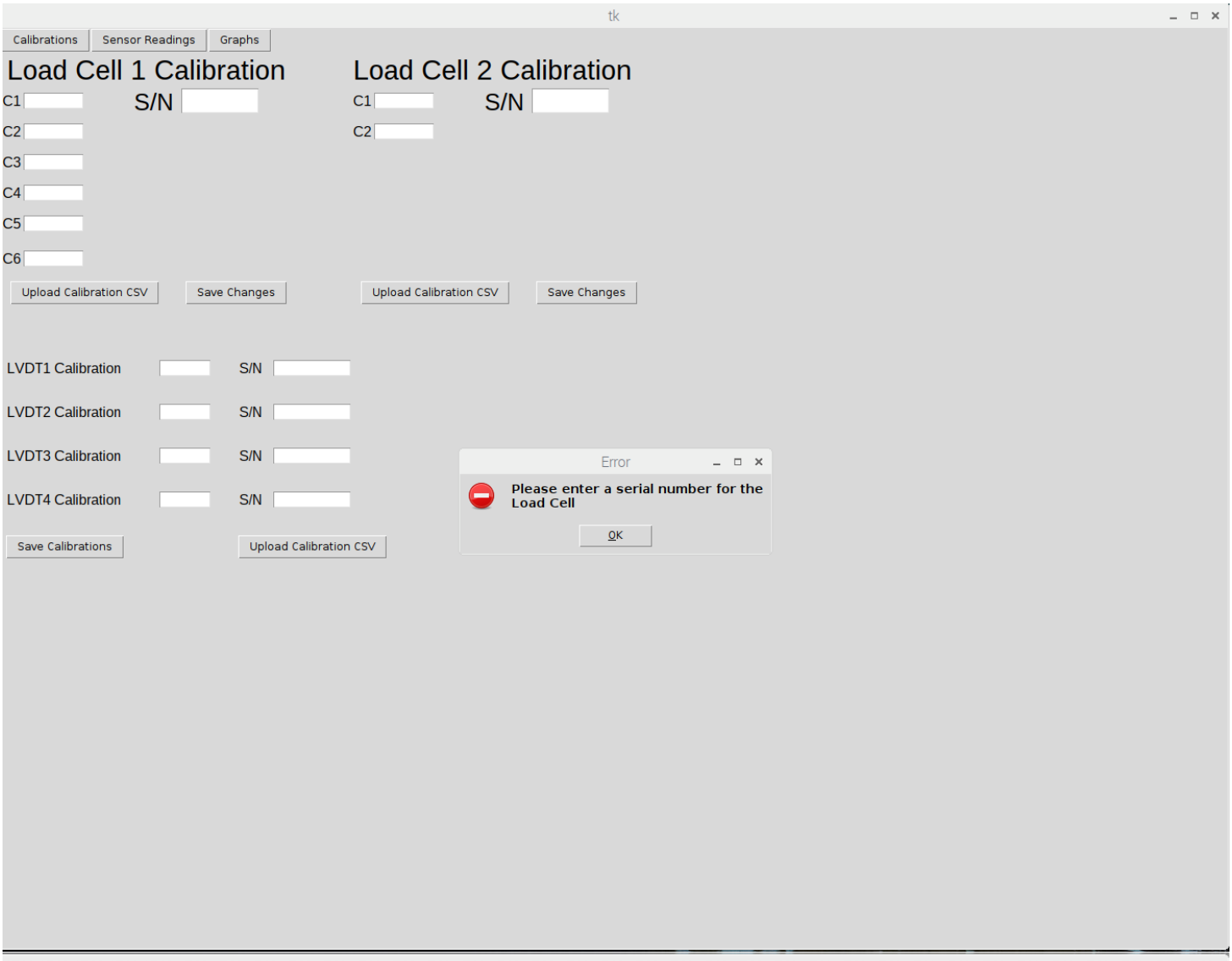
In the future, the software will just exist as a .exe application. This will allow for the user to just click on the app to start it running. For now, to run the app, follow these steps:

1. *Open a terminal*
2. *cd to correct directory (e.g. cd Desktop/DAS/examples/ADC__ enter)*
3. *Run application (python layered_app_debug.py)*

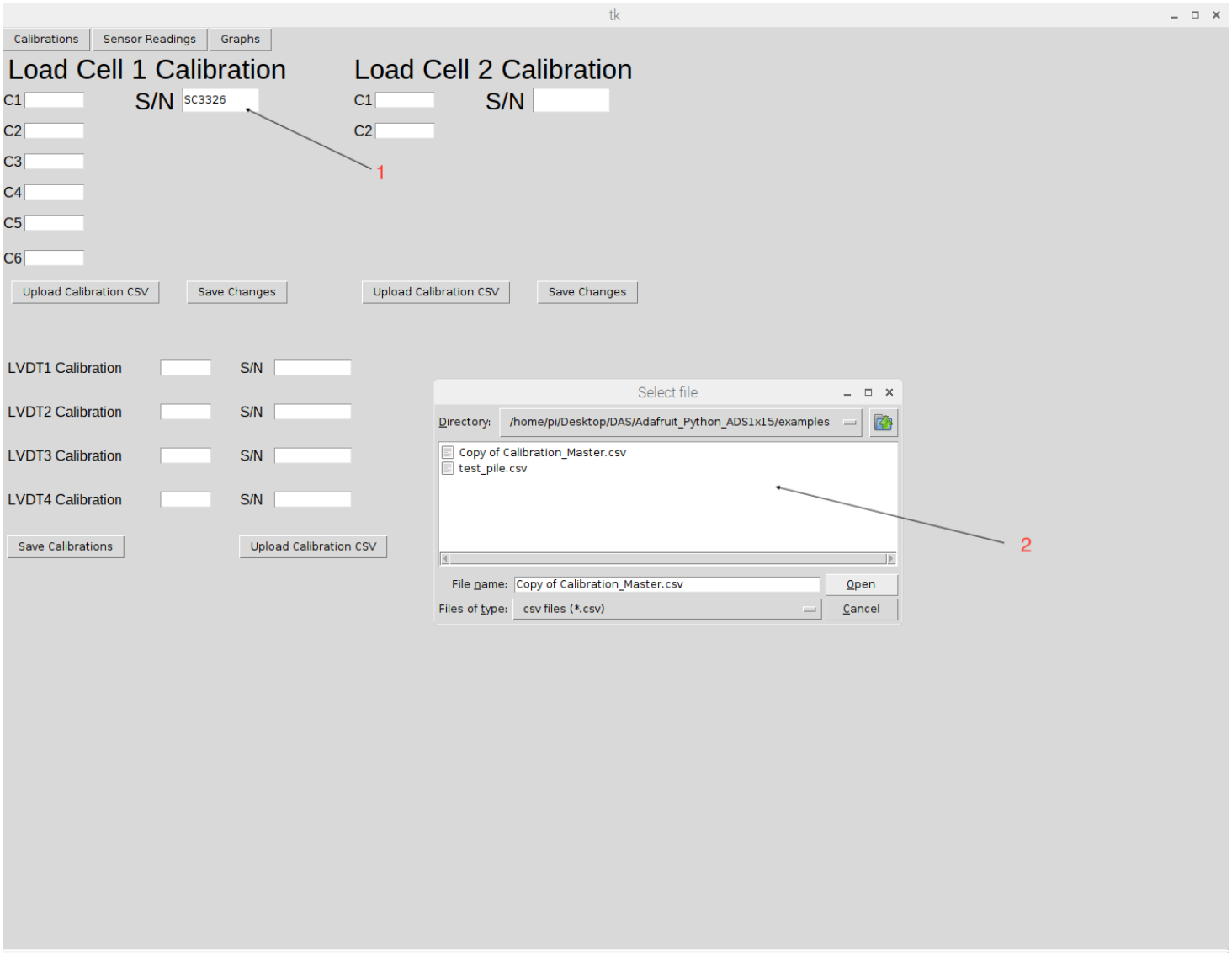


Key:

1. *Selection between pages. User can seamlessly transition between calibrations page, live sensor reading, and a graphing utility.*
2. *Calibration factors/coefficients can be entered manually as is done in the current system or auto populated with new calibration sheet feature.*
3. *Entering a serial number allows for the software to locate device data in the calibration sheet.*
4. *Uploading a calibration csv opens a menu where the user can look through files within the app to locate the master calibration sheet.*
5. *The user can save calibrations if entered manually. If calibrations are auto populated through the calibration sheet, they are automatically saved.*



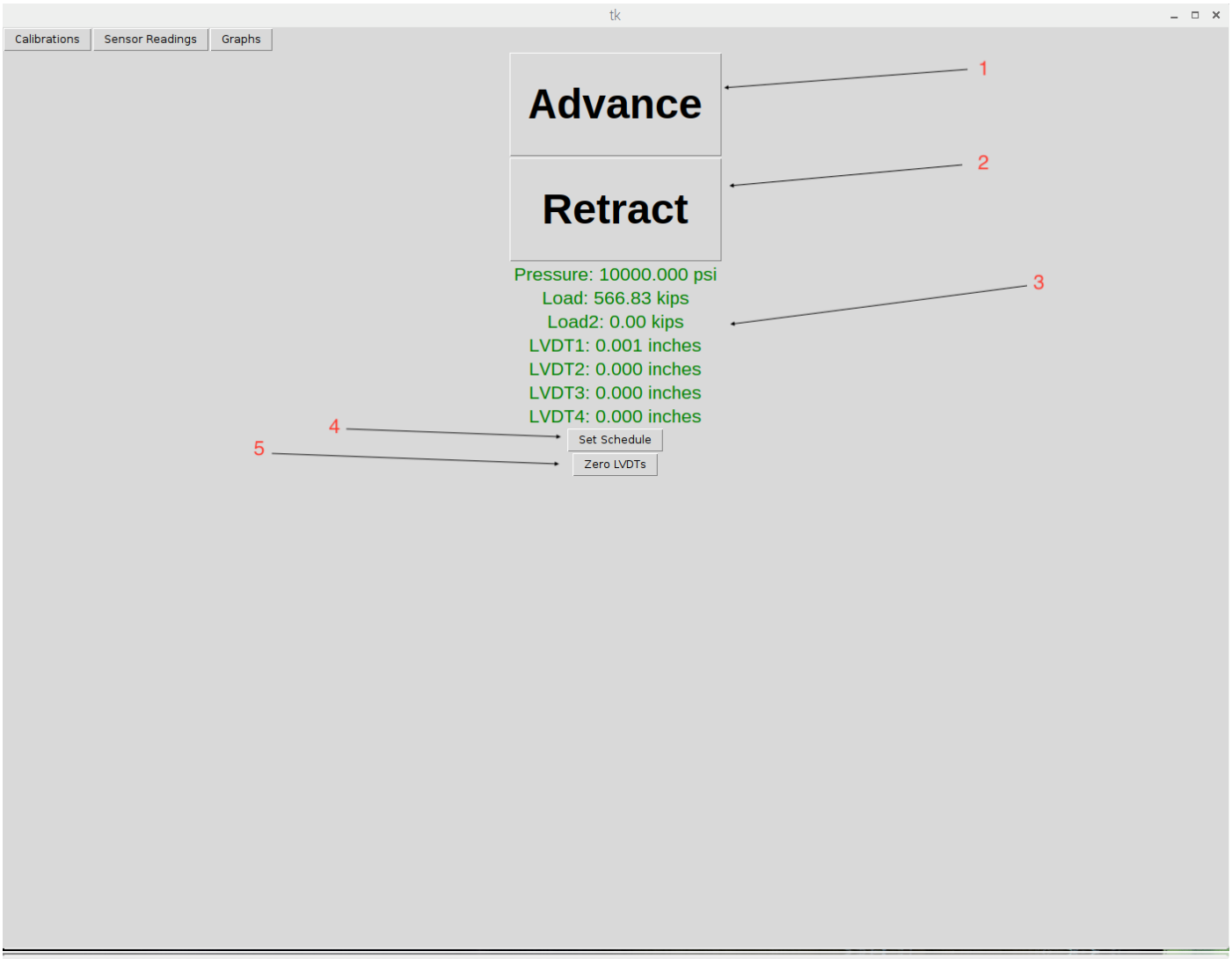
Attempting to upload a Calibration Sheet without providing a serial number will result in an error. Attempting to save calibrations when no calibrations have been entered will result in an error, and attempting to upload a calibration for a device who's serial number doesn't exist in the calibration sheet provided will also result in an error message.



Key:

1. *Attempt to upload calibration coefficients for Load Cell with serial number SC3326.*
2. *Picker menu to select file from within the app.*

The load cell calibrations have auto populated and are saved for use in the program.



Key:

1. "Soft" Button to control the advance side of the hydraulic pump from the Interface.

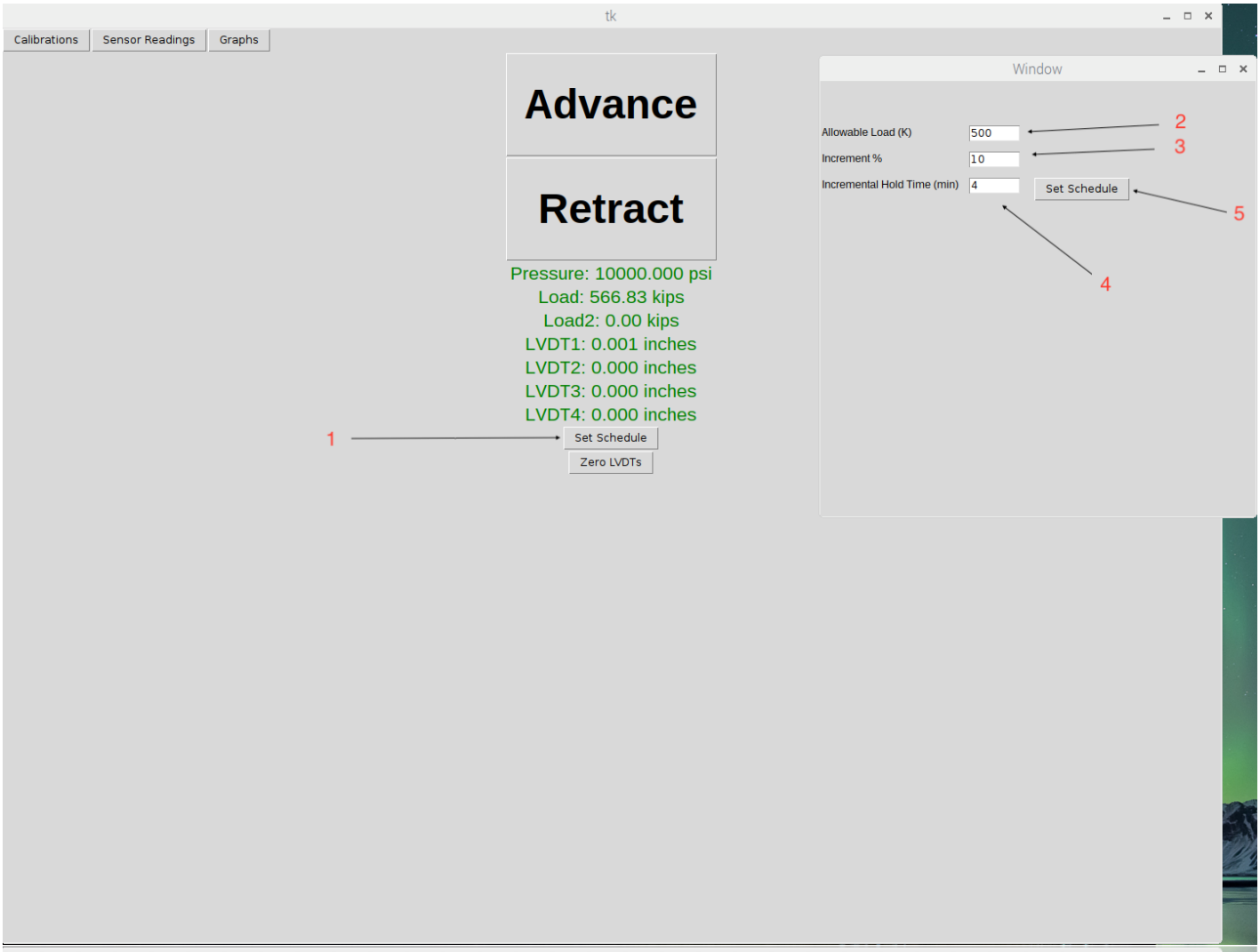
2. "Soft" Button to control the retract side of the hydraulic pump from the Interface.

(These buttons are a replacement for the physical remote which can be connected or disconnected via panel mount connection)

3. The current readout of all attached sensors (if more are required, they can be added to the interface with simple software changes).

4. A button to set a simple Load Schedule (see next page).

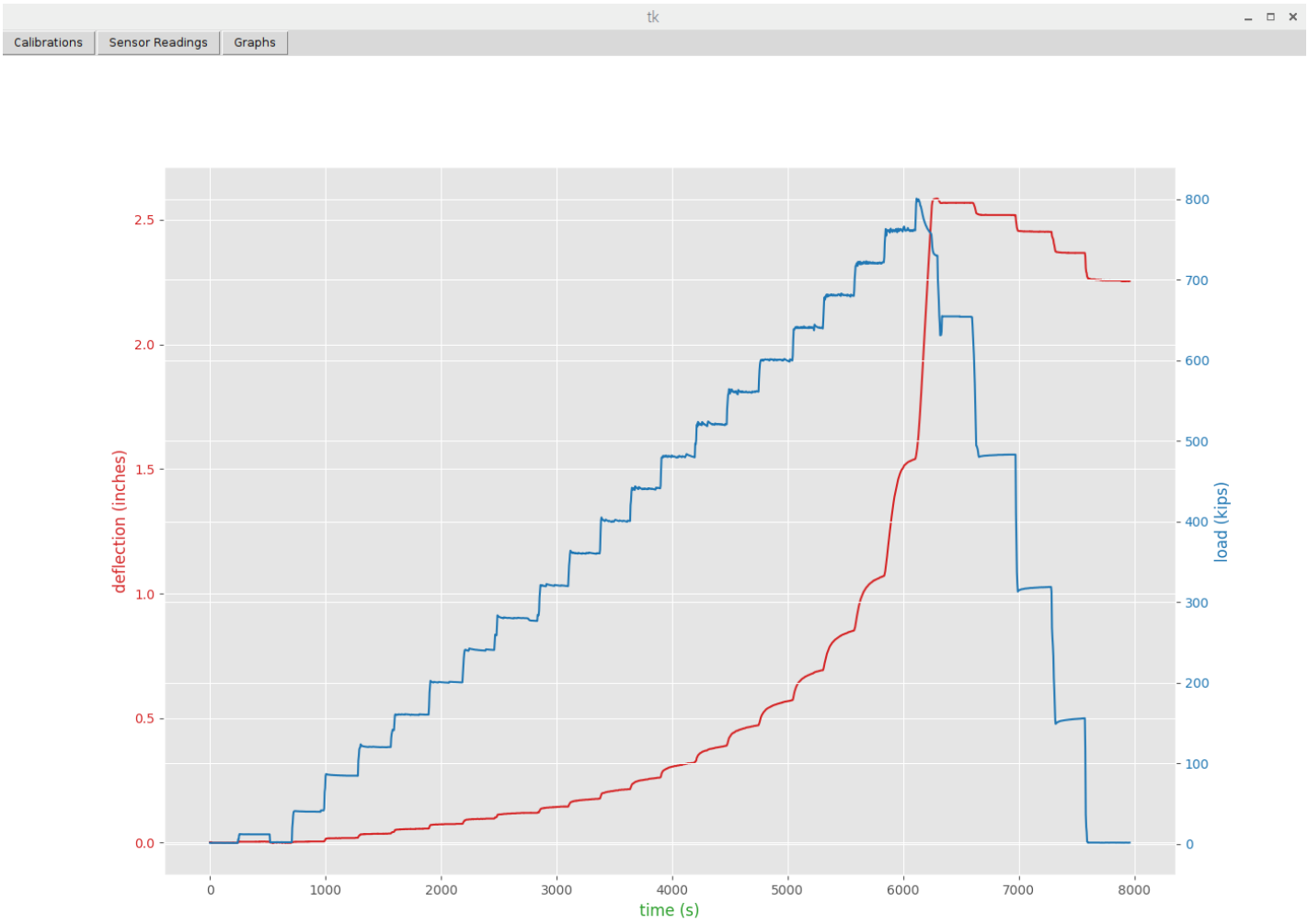
5. This button will zero all LVDTs by subtracting their starting position so that they can all begin at 0 inches.



Key:

1. *Clicking this button will cause a popup window to appear for setting a schedule.*
2. *This text field allows a user to enter the highest allowable load (in this example, the load test will not surpass 500 Kips)*
3. *This text field specifies the percentage of the maximum load to increment by at each step (in this example, the incremental percentage is 10%, which means the test will run in 50 Kip Increments)*
4. *This text field corresponds to the amount of time to hold each increment in minutes (in the example, each step will be held for 5 minutes)*
5. *Clicking the Set Schedule button will begin autonomous control.*

Note: This is the part of the software with the least amount of testing. It is not complex enough to run an entire load test. It will not pump down autonomously and will not follow logic specified by load test criteria.



An example of the graph for an entire load test using 1 load cell and 1 LVDT. To remain consistent with the RST Systems Logger Plus software, there 2 Y-Axes. The leftmost Y-Axis in red pertains to the deflection in inches of a single LVDT (more could be added). The rightmost Y-Axis in blue pertains to the load in kips. The green labeled X-Axis pertains to the time in seconds.