Retro Fit for Steam Heating Coover Hall

Design Document

Team Members

Jevay Aggarwal	Technical Lead
Sarah Coffey	Weekly Report Lead
Thomas Devens	Project Plan Lead
Joseph Filbert	Client Contact Lead
Ken Wendt	Website Master
Liz Wickham Kolstad	Design Document Lead

Client: Leland Harker

Contents

1. Introduction

1.1 Acknowledgement

Leland Harker is our client, providing funding and guidance for this project. We would like to thank him for the opportunity to work on this project.

1.2 Problem Statement

Coover was built in the 1950s to be the Electrical Engineering Building. After over 60 years, it has only been renovated significantly in 1999 to include the addition of the Active Learning Complex. With that being said, there has not been a significant renovation of the steam heating system since it was built ("Coover"). While the steam heating system allows the heat to be generated off-site in a power plant across campus and transferred through pipes into the various offices, labs, and classrooms in Coover, the system lacks the ease of use that modern day technology offers. The steam is controlled through a number of valves distributed throughout the building. The system cannot react to changes in outside temperature or adjust to faculty's choice in temperature. According to the Room Temperature Policy, "University guidelines for space temperatures are 70 to 76 degrees during occupied hours and 63 to 83 degrees during unoccupied hours" ("Building"). Coover is in need of an upgraded system to control the steam and keep the temperature in the rooms stable in fluctuating outside temperatures. This will lead to more comfortable rooms to for occupants and use steam more efficiently. There are two main problems associated with the current steam heat system which we are faced with: the steam radiator valves are often hidden behind miscellaneous furniture or equipment making it very hard to adjust the temperature in the rooms. Additionally, there is no reliable way to control the temperature of the room. Currently, the only way to adjust the temperature is to open or close the radiator valve manually and wait to see if the room gets too hot or cold.

1.3 Operational Environment

Our product will be used in classrooms around Coover. Extreme temperatures will not be an issue for our product, nor will adverse weather conditions. However, dust may accumulate on our components and unintentional abuse may occur due to human interaction. Some preliminary testing has shown that the valve mount will need to withstand temperatures less than or equal to 150 degrees Fahrenheit. Additionally, the mount and motor should be made of corrosion-resistant material due to possible steam leakage expected on or near the radiator valves.

1.4 Intended Users and Uses

Our product will be used to control the temperature in classrooms and offices. People in those rooms as well as administrators to the system will be able to control the temperatures.

1.5 Assumptions and Limitations

Assumptions:

- The maximum number of people using the thermostat is one at a time
- The user will not move the valve by hand
- The user will have permission to change the temperature if they have access to the room

Limitations

- The valve controller will fit on the valve and have access to a wall outlet
- The thermostat will be battery powered
- The user will be able to change the temperature on the thermostat in the room

- An administrator of the system can access and change the temperature for any room from anywhere with access to campus WiFi

1.6 Expected End Product and Deliverables

First deadline: end of April 2018
- Ability to turn the motor
- Basic WiFi communication between the valve controller and thermostat
- Information on the temperature patterns in the rooms (for a baseline)
- Information on battery usage of the thermostat

Final deadline: end of December 2018
- Thermostat and Valve controller prototypes with user interface and
- Instructions on how to create the project from its components in order to duplicate for any room in Coover Hall
- Parts list of materials needed
- Working prototype and data to show improvement in temperature consistency in rooms where it's implemented
- Sends error reports if the valve is stuck or communication is lost

2. Specification and Analysis

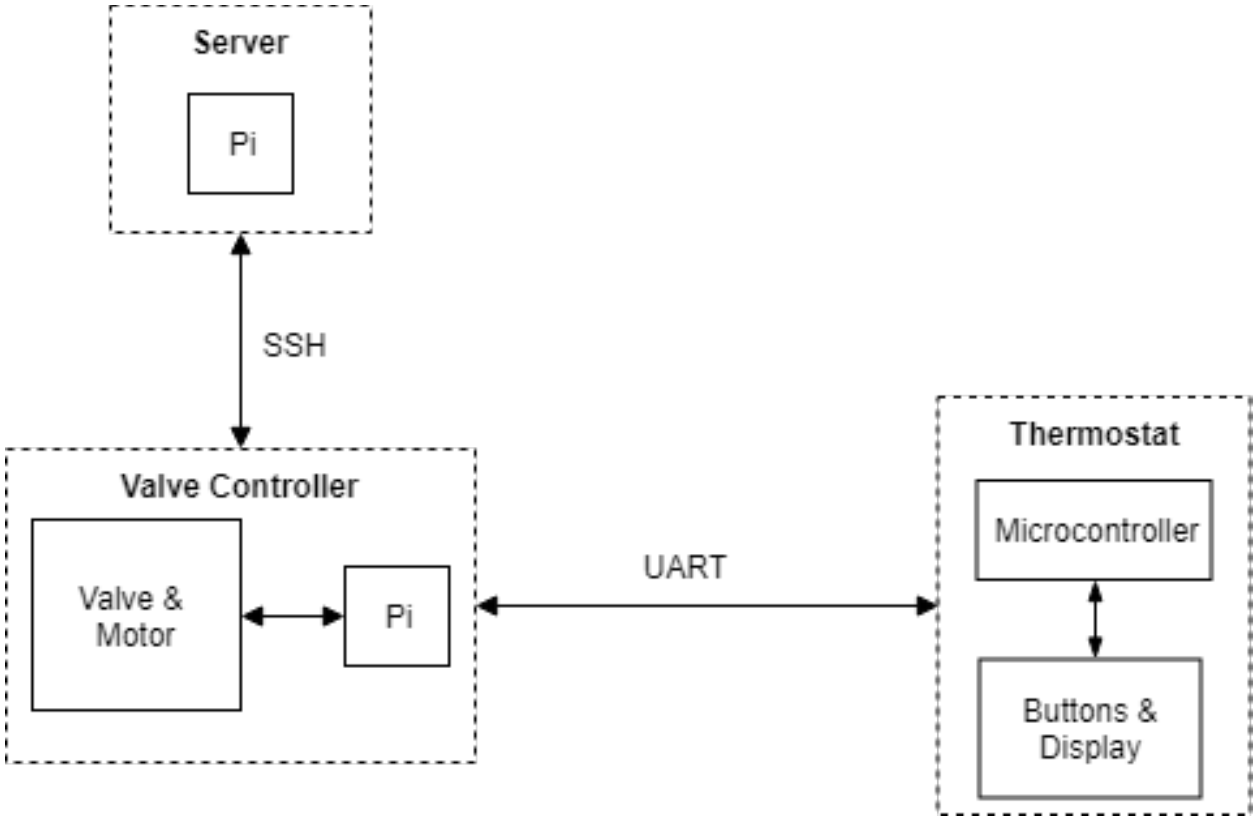
2.1 Proposed Design

The design for this project is constrained by our client's requirements. Specifically, these requirements define two systems: a thermostat and valve controller. The thermostat will be able to control the temperature of the room. While our group is only prototyping with a single valve controller and thermostat, the client would like to extend the solution to all rooms containing steam valves in Coover. This means that we have to design for numerous systems, while only prototyping one. We are developing our solution with these considerations in mind.

Our valve controller design starts with a Raspberry Pi 3 (or simply "Pi"). This Pi will communicate to the server and to the thermostat via WiFi to control the settings for the temperature. The desired room temperature will be stored in a file so that even upon reboot every night, the information will persist. Every 15 minutes (subject to change after functional testing) the Pi will read the temperature from a sensor (the MCP9808 High Accuracy I2C temperature sensor) connected to the I2C communication line on the Pi. Once this temperature is compared with the desired temperature, our algorithm (not yet designed) will decide whether to turn the valve and if so, how much. If our code decides to move the motor, our Pi will send a PWM signal to the motor controller (Adafruit DRV8871 DC Motor Driver Breakout Board) which will in turn, drive the motor (IG-42 24VDC 078 RPM DC) to turn to the desired location. With the encoder provided on the motor, we can monitor how much it has turned. With the software we will design, we can monitor if it has stuck, if it has reached the fully-opened or fully-closed state in order to protect the valve from over-torquing. If the software determines the valve to be stuck, it will send an error report via email to the necessary staff. We will also rotate the valve fully open and fully shut once a week to "exercise the valve" and minimize sticking.

The thermostat will consist of 2 buttons and a display that the user will use to change the value of the desired temperature. One button will increment the desired temperature 1°F up, the other 1°F down. The display (Quad Alphanumeric Display with I2C backpack) will use I2C to communicate with the Feather. The two digits on the left will show the current room temperature, the two digits on the right will show the desired temperature. The buttons and display will be wired to the Feather HUZZAH board ("Feather"). This board will connect to the Valve Controller's Pi via WiFi protocol. Once the user has set the desired temperature with the buttons, the Feather will send the updated temperature to the Pi and shut itself off again. Our original design took advantage of the Feather's "deep sleep" mode. Even with all systems turned off, this required too much battery capacity. (See "Challenges - Battery Life" below) To fix this issue, we will implement a "power control system". This will consist of a 3 input OR gate into a MOSFET. Each button will be input into the OR gate along with a GPIO Pin from the Feather. When the user initially presses a button, the OR gate will output a logical "1" into the MOSFET, allowing the current to flow through the MOSFET and power the Feather. The Feather will then set the 3rd pin on the OR gate to "1", effectively controlling it's own power source. When the user has not pressed the button in 15 seconds (subject to change), the Feather will set the GPIO Pin connected to the OR gate to "0", and the OR gate will then output a "0" and turn off the current flow in the MOSFET. This will allow little to no power being used outside of user operations.

In order to facilitate building-side communication, we have a dedicated Raspberry Pi that will be set up as a server. We have been experimenting with SSH to control the Pi and designing what the network should look like.



2.2 Design Analysis

Our team has been divided into software and hardware teams. The software team is working on communication with the server, the Pis, and the feathers for their respective rooms. The hardware team is developing the valve controller.

Software: The first prototype of this device used an analog temperature sensor, which had to be relayed through an ADC unit on a microcontroller. To simplify this operation, we switched to a digital temperature built with an I2C bus capable of communicating with the Pi. We considered several options for the display, including a 16x2 character LCD and a Quad-alphanumeric LED display. Using a display with built-in I2C would simplify the design and use fewer pins on the Feather. This is preferable because there are only 9 available GPIO pins. We decided to use a Quad-alphanumeric character display after considering the alternatives; it uses I2C and because the thermostat will remain off for most of the time, the power consumption of the display will not be an issue. The buttons are momentary switches. They are easy to interact with, use little power, and are inexpensive. We are testing each component individually with the board to verify its functionality, and integrating once it's confirmed.

Hardware: We have been working on the physical mount that will hold the motor that turns the valve. We are using OnShape (a online CAD software) to model the valve and design ideas for the mount. We took measurements of the steam valves in Coover to use as a reference in our design. While measuring the valves, we made an important observation. Some of the valves are next to right angle turns in the pipes, so the initial design of the mount, an encased box that sits on the pipe, would not fit on the valve. To rectify this problem, we switched to a mount that supported itself by holding onto the valve, which actually made it easier to model. Our design now includes a ring that can be tightened around the valve to provide adequate support. We drew up this design in OnShape and presented it to our client, Leland Harker. The assembly drawing can be found in Figure 1 in the appendix. Through a discussion on this

design with our client, Leland Harker, we decided that the four-sided steel case would be overkill and changed the design to use only one side. We have received our manufactured final mount design and have verified its sizing to fits our motor and the valve. This took two attempts as we found the screw holes to be too small when we got the physical mount the first time. We fixed the design in OnShape and passed them on to Lee Harker to be corrected. Lastly, we have wired the motor, motor driver, and Pi together to rotate the motor according to simple PWM software running in Python on the Pi. We created a simple program to rotate the motor either way, at a given speed or for a given distance.

3. Testing and Implementation

3.1 Interface Specifications

We realize that testing will be a major part of the project. We started with logging temperatures inside room 1316 and kept track of the temperatures outside as to create a baseline for how impactful our system will be. Data collection occurs automatically every 15 minutes. The thermostat will have external buttons that will change the desired temperature. We will test the thermostat by changing the desired temperature and making sure a corresponding change happens on the valve controller. We will calculate a relationship between the temperature change and valve position that can be used in the code on the Raspberry Pi. For remote access to adjust the temperature, we plan on creating a webpage that can only be accessed by the staff within Coover. Logging in and authentication of users will be handled by the ISU Shibboleth system, and the webpage itself will start as two buttons similar to the physical thermostat, and work in the same way. The testing for this process will also be similar to the thermostat, with LED's to determine if the commands are being sent.

3.2 Hardware and Software Tools

Our software team is using Python to program the Raspberry Pi for the thermostat and valve controller. The Raspberry Pi will have several physical inputs and outputs that are handled by our code, so we will be simulating these inputs and evaluating the output products as needed. The main tool we are using to evaluate outputs and debug is an oscilloscope. We have used two different usb oscilloscopes to monitor our Pi's signals, including an N-scope and a Pico-scope. Both tools work similarly and enable us to monitor the outputs of the pins on the Raspberry Pi.

In order to measure power, we have used a power supply and ammeter. The power supply controls the voltage that is given to the thermostat and the ammeter can measure the current the board is pulling from the supply. Another helpful circuit debugging tool is the waveform generator to check what waveforms a board, specifically the motor control board, works well with before connecting the Feather. Debugging every component individually in this way is more effective at pinpointing the problem and producing a solution.

We tested our motor driver by hooking up our raspberry pi. After researching how to create square waves with python on the microcontroller, we tested controlling the motor in both directions. We then tested running the motor and tracking the output of the encoder to see speed and distance traveled. With the output from the controller, we can tell if the valve is stuck or has been fully opened or closed.

Our mechanical team is using an online CAD software called OnShape. OnShape is preferred over other CAD tools for our team because all our files are stored online so the team can work on the same model from their personal computers and not have to worry about how to share the drawings with each other. We are using OnShape to model our motor mount design to test its dimensions before fabrication. We also modeled one of the steam valves to help with modeling our mount. We used a caliper to gather the valve dimensions because the valve manufacture, Ohio Brass Co., is no longer in business and therefore do not have their own CAD drawings online for us to use.

One test we conducted when we choose a motor was the torque required to turn the valve. Using a torque wrench, we found it takes around 1 ft-lb to turn the valve and around 8 ft-lb to close the valve. We had some brushed DC motors from a previous project that we are going to use. We took those dimensions and modeled it in OnShape with our mount design to model the system as a whole.

3.3 Functional Testing

As each part of the project is worked on, we will test expected functionality within the system. Once we have verified its functionality, we will build up our system incrementally, testing as we add components and functionality. This will allow us to debug the system in smaller parts, and catch edge-cases where our system may not respond according to our design specifications.

Our hardware team is been using a signal generator, oscilloscope, multimeter, and power supply to test the DC motor and the motor encoder. We have been able to control the motor using various DC voltages from our power supply and obtain output from the encoder on our oscilloscope. We then introduced our H-bridge to attempt to control the motor using a PWM signal from our signal generator and from a Raspberry Pi. We have been successful in controlling the motor using the Pi and have been able to read the encoder signals back into the Pi for analysis.

We will include evaluations of our software as it responds to different environments. The thermostat will need to be tested extensively as it is a user interface; this will include measuring the microcontroller's response to the button inputs and the screen's display. Each valve controller and thermostat are going to have wireless connections to each other and to the web server, so the communication protocols will also need to be verified.

Our hardware team is currently designing and testing the circuit that allows the Pi to control the motor and the motor to provide feedback to the Pi. This is being done in multiple stages to make our testing easier and more reliable, allowing us to find problems as new components are being added. We have been bread-boarding a prototype circuit and have successfully controlled the motor with the Pi. Our next step is to introduce and test our chosen power supply. The circuit will also be modeled in Modelsim, a SPICE program that will easily allow us to transfer the circuit to the program Ultiboard, a program that allows us to create our PCB layout and generally the Gerber files to be fabricated.

3.4 Non-Functional Testing

Scalability: Our project will need to be scalable, with the potential to be implemented in every room in Coover Hall. We will need to keep this in mind throughout our project, and set up a system to implement new modules in a room; automating the process as much as possible.

Server: The server has to give useful feedback to the logged-in users about the temperature setting in each room. It has to authenticate the user as they log-in correctly to prevent unauthorized people from gaining access.

Security: Cybersecurity is an important part of our project because all valve controllers, thermostats, and servers will be connected to the internet at Iowa State. While our team does not have specific backgrounds in cybersecurity, we will have each raspberry Pi update its software every night. This will keep the most updated operating system and programs on the Pis. We will also consider every transmission happening between components and make them as secure as possible. One example would be our server creating a secure shell to each valve controller that it wants to set a temperature and using a key instead of a password to allow this. Since the valve controllers will be accessible to students and faculty in any of our given rooms, we are limited in our ability to prevent an attacker from physical access to our valve controller and thermostat.

Usability: We will have a variety of users from students to professor, we will bring in various students and professors to have them use our thermostat and server interface to create a straightforward interface to change the temperature in the room. Our goal is to make an interface that a user intuitively knows how to turn the temperature up and down.

3.5 Process

Valve Controller: The test plan for this device includes the connections between the motor driver, Pi, and motor, as well as the temperature sensor and Pi. We will start by verifying that the Pi can drive the motor clockwise and counterclockwise. We will also test the software to make sure it can correctly detect when the valve is completely open or shut, or when the valve is stuck and needs repair. We have tested our temperature sensor by logging the temperature in 1316. We can then compare it to the outside temperature to establish a baseline for the current temperature patterns in the room. See figure 3 below for a graph of the data we have obtained so far.

Thermostat: We have tested the Feather to see the range of the current draw. This allows us to see the battery capacity we will need to run the thermostat on battery power for a semester. We will repeat our tests as our design changes to conserve as much battery power as possible. We will also test the button and communication of the Feather with a simple interface that will change and communicate to the Pi. Once we have verified the hardware, we will set up the valve controller in the workshop and verify that the temperature in the room remains steady, even with large temperature differences outside.

Server: Our server will be the last component we will test. We will test the user interface and verify that the valve controller is updated with the new temperature. We will also test the administrator functions that change the temperature in multiple rooms at once.

3.6 Results

While we are still developing the system, our tests for the temperature sensor are promising, offering consistent temperature readings over our 2 week test (see figure 3). We are also testing a Pi, verifying that the operation is consistent over long periods of time. We have concluded that with updates and reboots every day, our Pi is operating with nominal response times. We have also verified that our motor can be controlled by the Pi through a motor driver board. As of the writing of this paper, results for the initial phases of this project look promising. We will continue to test and modify our project as we develop the system and fix problems.

3.7 Challenges

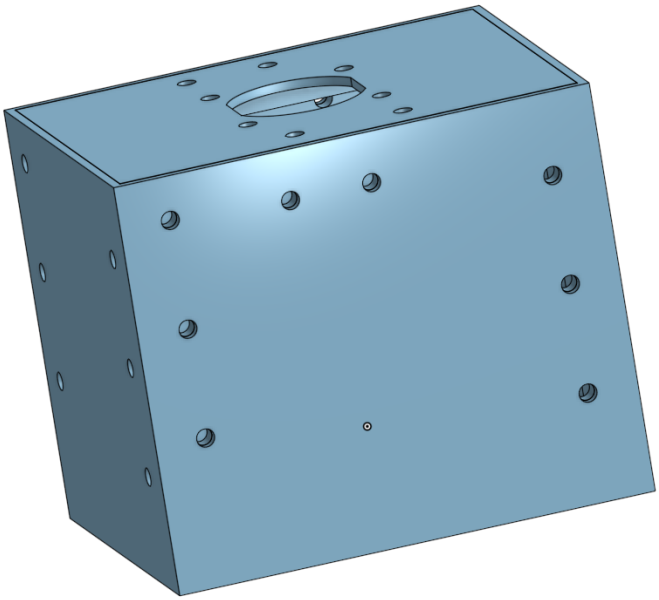
Battery

Our client would like the thermostat to be run off a battery to allow placement in any part of the room. This will maximize accessibility to users. Our initial research showed that the Feather has been run in a Deep-Sleep state for a year. However, after testing in our system, we found that the feather used 7 mA in it's lowest power state. Our client requires that the thermostat batteries last for an entire semester (2,856 hours), but even with minimal use, we would need a battery with at least 20,009 mAh capacity. While there are rechargeable batteries with this capacity on the market, they range from $15 dollars to $60 dollars. We can use a cheaper battery by turning off the Feather when the Thermostat is not in use. This will conserve energy, as none will be consumed when the thermostat is not being actively used.

Communication

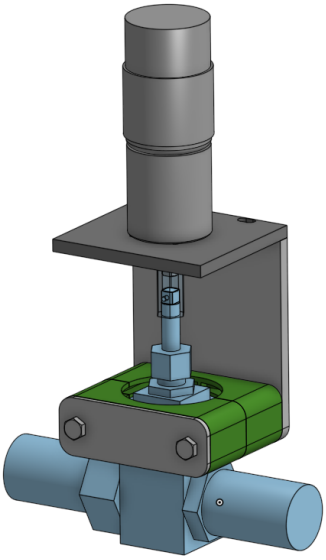
We considered two alternatives for communication between the thermostat and the valve controller. When battery power and energy consumption were trying to be lowered by using a deep-sleep mode, the clear answer was BLE (Bluetooth Low Energy). However, by turning the thermostat off for the times when it is not being used, BLE would be cumbersome and difficult to synchronize between the two devices. The better protocol to use for the current design is WiFi.

4. Closing Material

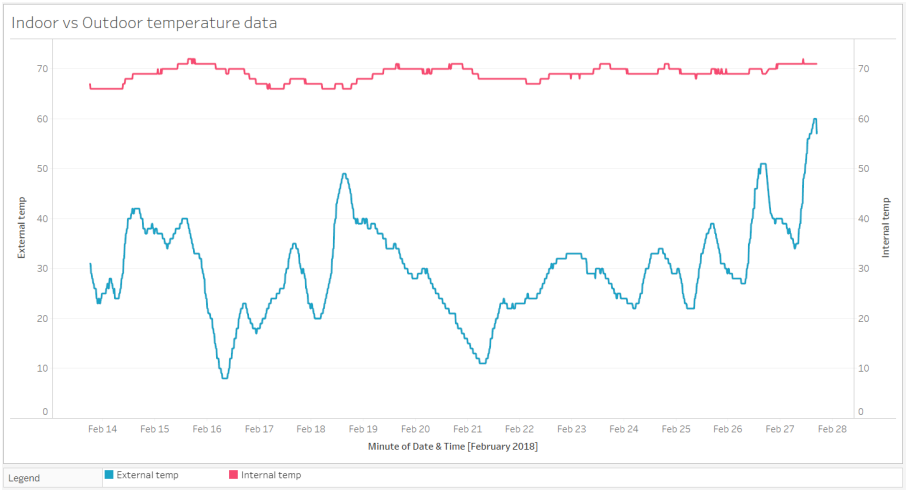


4.1 Conclusion

Software: Our main goal for the software side of the project is to set up and test the communication



between the thermostat, motor controller, and server such that we can adjust the temperature on the thermostat via a button interface and it will maintain the selected temperature in the room. Our secondary goal is to set up and test a simple webpage to change the desired temperature of any room from any connection to the campus WiFi from an authorized user. Our methods of testing this communication will be to create a command by pressing a button on the thermostat, and then send that command through the server to the motor controller which will light an LED to confirm that it has received the command. Once



we have basic communication set up, we plan to implement the motor control based on the command sent by the thermostat. To test this, we will again send a command to the controller and measure the change in position of the valve to see if it moves to the correct position. For the webpage portion, we plan to implement ISUs Shibboleth authentication system to limit access to the staff inside Coover. We also plan to design the remote access system such that an administrator has access to and can change temperature in any room, individually or simultaneously.

Hardware: Our team has designed a motor mount for the valve controllers in Coover Hall with help from our client, Leland Harker. We have also successfully tested the motor, motor driver, and encoder interaction with the Raspberry Pi. We have brainstormed ideas for how to test if the valve is stuck, fully opened, or fully closed, and will integrate a weekly valve check to exercise the valve to prolong the valve's life. We will be designing PCB layouts for our valve controller and thermostat in our next stage of circuit design.

4.2 References

"Building Energy Plans: Coover Hall." *Iowa State University Utility Services*. https://www.fpm.iastate.edu/utilities/bldg_energy_plans/building.asp?bldg=COOVER Accessed 6 February 2018.

"Coover Hall." *Iowa State University: College of Engineering*. https://www.engineering.iastate.edu/the-college/engineering-buildings/coover-hall/ Accessed 6 February 2018.

4.3 Appendices

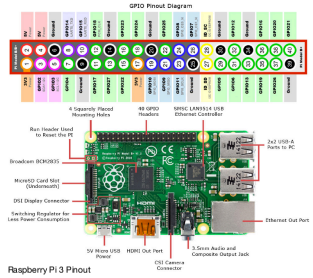
Figure 1. Scrapped motor mount assembly drawing

Figure 2. Current motor mount assembly drawing

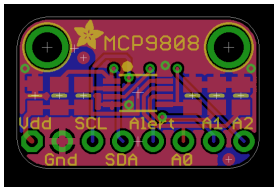
Figure 3. This is the data we logged in the first few months of this project in room 1316. The temperature is quite stable for the most part, and only drops after a drastic change in outside temperature. This does include changes in the valves control

Valve Controller

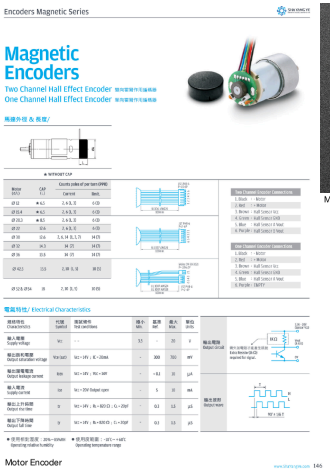
Pins	Raspberry Pi	Motor Driver	Motor	Motor Encoder	Temperature Sensor	Power Supply
Power	micro USB					USB to wall converter (2 Amp min)
	5V out	Vm in		Hall Sensor Vcc	Vdd in	
	GND out	GND in		Hall Sensor GND	GND in	
		Power +				24 V
		Power -				GND
Data	GPIO2 SDA1 I2C				SDA	
	GPIO3 SCL1 I2C				SCL	
	GPIO 13 CW	In1				
	GPIO 19 CCW	In2				
		Motor +	Red			
		Motor -	Black			
	GPIO 26			Hall Sensor A Vout		



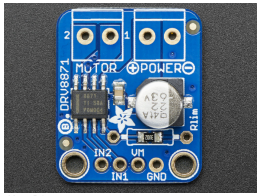
Raspberry Pi 3 Pinout



Temperature Sensor Pinout



Motor Encoder



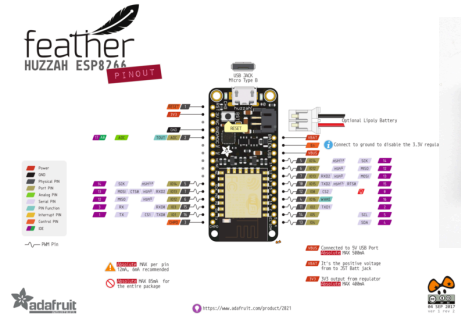
Motor Driver

Valve Controller

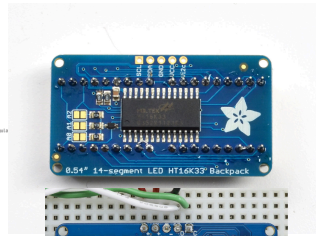
Thermostat

	Feather Board	Button Down	10 K resistor	Button Up	10 K resistor	Display	Power Control Circuit	Power Supply
Power							In +	5 V
							In GND	GND
	Bat +						Out +	
	Bat GND						Out GND	
	5V							
	3.3 V		in		in	Vl2C, VCC	Select 1	
	GND	pin 2		pin2		GND		
							Select 2	
Data	GPIO 14 SCL					SCL		
	GPIO 13 SDA					SDA		
	GPIO *						Select 3	
	GPIO 7	pin 1	out					
	GPIO 6			pin 1	out			

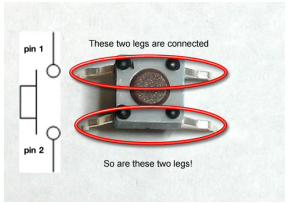
* Still to be determined



Feather Huzzah



Quad Alphanumeric Display



Momentary Switch

Thermostat