# ESW Vaccine Project Final Report

## Executive Summary (10 pts)

This report summarizes the work of one of Stanford's Engineers for a Sustainable World international project teams in developing a remote vaccine refrigerator monitoring system for the global health nonprofit, PATH. We completed this work primarily from January to June 2015.

We begin by discussing the importance of improving the vaccine cold chain to global health, and explain how a system capable of remotely monitoring vaccine refrigerator can help both in testing different solar powered vaccine refrigerators and potentially serving in an alert system. We then review the specifications for such a system given to us by PATH, including requirements on the number and types of sensors, the presence of nonvolatile local storage of data, and ease of access of the data.

Next we provide our ultimate approach and results for meeting these specifications. Part of our approach was dividing our system into two halves. One half was an arduino based hardware device that would take measurements, store them locally, and transmit them by SMS. The other half was software that would received the SMS messages, parse them for the data, store the data in a database, and allow easy access as well as visualization of the data. For each half, we detail the subcomponents and discuss some design decisions.

We subsequently point to how consideration of sustainability influenced our design. The two facets of sustainability we particularly focused on were robustness and reproducibility. We considered sustainability of both the system itself as well as the sustainability of the project for further development.

Finally, we conclude by reflecting on the project as a whole. We include some of the challenges we overcame, our hopes for the impact of our project on global health, and the future of the project. Some us will travel to share the project with PATH in person. We also have suggestions on improvements that could be made in future iterations of this project.

## Introduction (10 pts)

According to the World Health Organization, 1.5 million children under 5 years old die each year from vaccine-preventable diseases. This project is part of a larger effort with PATH, a Seattle based global non-profit health organization, to help lower that number. Many people do not have access to these life-saving shots partly because of the difficulties of transporting and storing vaccines. One of the major problems with transporting vaccines is that they are temperature and time sensitive. Once the vaccines go out of their temperature range, they become ineffective.

To help address the problem of temperature sensitivity, we need technology to monitor the effectiveness of current vaccine refrigeration devices. The vaccine refrigerator remote monitoring device we've developed will be used by PATH to assess the performance of currently available solar powered vaccine refrigerators and potentially use the information gained to help improve the refrigerators. Initially our device will be used to test refrigerators in PATH's headquarters in Seattle. In the next stage, PATH will likely use our device to test refrigerators at U.S. laboratories such as Sandia Labs that allow testing in precisely controlled

environments. The final possible stage for the the device's deployment is to serve as a remote monitoring and alert system in under-resourced international communities. Proceeding to this final stage will depend on the superior functioning of the device and may require further iterations and improvement in the subsequent year.

The goal of this project was to create a vaccine refrigeration remote monitoring system meeting specifications given to us by PATH. The system needed to be capable of collecting data about the refrigeration system's current and voltage, as well as of making 8 separate temperature measurements. The system needed to be capable of both locally recording the data it collects and transmitting the data such that it can be remotely viewed and downloaded in a format readable by excel. The project aims are summarized in the following list:

*Project Aims*

- Develop a remote monitoring system for vaccine refrigerators
    - Create an device to read temperature, voltage, and current data from the refrigerators and transmit the data
    - Create a web application to receive and process the data
    - Connect the device and web application to make data available remotely
    - Develop a way to view the data online
    - Develop a way to download the data
- Test the monitoring system and fix as needed
- Write a clear user guide to allow others to use our system
- Deliver the monitoring system to PATH

We have succeeded in accomplishing all the initial aims of our project. Our final product consists of an Arduino platform capable of measuring temperature, voltage, and current data from the refrigerators and both saving this data on an SD card and transmitting this data to the database of a website. The website gives PATH users who know the general password the ability to view data visualizations and raw data and to download data over any desired time range in a .csv format.

**Results (40 pts)**

Our final deliverable to PATH will be a two-part deliverable. The first part is actual remote monitoring device, which is run by an Arduino microprocessor. The second part is the website (and the source code for the site), which provides the actual interface for people at PATH to examine the measurements of the monitoring device. Since our deliverable has these two separate layers, the presentation of our results will similarly be divided into the Arduino deliverable and the website deliverable.

**Arduino:**

Our team decided early on to use an Arduino to control the actual remote monitoring device. Our reasoning was that an Arduino is very capable of performing the calculations and tasks required of a simple temperature and voltage monitor, and that Arduinos are cheap and easily modified microprocessors, meaning that PATH can easily modify and create more of our device.

As requested by PATH, our device can process 8 simultaneous temperature readings using type T thermocouples. Thermocouples operate by outputting a voltage difference which varies based on the temperature and two metals used in the thermocouple. However, in order to measure the voltage difference output by a thermocouple and calculate the actual temperature based on that difference requires special circuitry. We opted to use a cold-junction compensated thermocouple-to-digital converter, the MAX 31855, which is an easily purchased electronic chip. The MAX 31855, when set up correctly, takes the voltage output of a type T thermocouple and writes the corresponding temperature to a digital output pin.

The MAX 31855 operates use Serial Parallel Interface, which initially seemed slightly complicated. First, the chip requires an electric power and electric ground. Next, each chip requires an electrical connection to the Arduino's clock, chip select, and digital output pins. The Arduino clock is simply a square digital wave output by the Arduino, which dictates how fast the Arduino reads bits from any digital input. The chip select pin, which is separate for each chip, is set to a low voltage when the Arduino requests a measurement from that chip. Lastly, the digital output pin is used to output the temperature reading in its bit representation to the Arduino. Finally, each chip requires two pins to read the inputs of two ends of a type T thermocouple.

Figuring out the function of each pin on the MAX 31855 and then setting up a working circuit took the Arduino sub-team a lot of time. Thankfully, though, there already exists an Arduino library which takes care of reading the measurements from the MAX 31855 once the circuit is set up properly. In addition, we learned that setting up multiple MAX 31855 chips is relatively easy once the first one is set up, because many electrical inputs can be shared; all the chips can share the same power, ground, clock, and digital output wires. In fact, each additional chip only requires one new unique connection, to the chip select pin, from the Arduino.

This discovery that many of the electrical inputs to the MAX 31855 chips could be shared prompted us to design a PCB board. Our PCB board is designed so that the electrical connections needed to run the MAX 31855 chip are already build into the board, and the many shared connections are all simply connected in the PCB board. Our PCB board simplified our circuits significantly, and makes it possible for PATH to easily order the PCB boards and replicate our device. In fact, once someone receives the custom PCB board, they simply need to solder on the MAX 31855 chips in the correct places, and plug in the type T thermocouples and Arduino to get temperature readings.

Our device was also required to read the voltage and current of the solar panels powering the fridge. In order to measure the current from the solar panels, we selected a current sensing chip, the ACS713, which is capable of measuring up to the specified 30 amps of direct current. The circuitry for the ACS713 is also integrated into our PCB board, so that PATH can hopefully simply plug in the electrical wires carrying the current to the appropriate place on our PCB board. Lastly, the device uses the Arduino's analog-to-digital converter to read the voltage

output of the solar panels. However, since the Arduino can read at most 5V, we use a resistive divider to “divide” the voltage into a range that the Arduino can actually measure, then use simple do some math to convert back to the actual voltage.

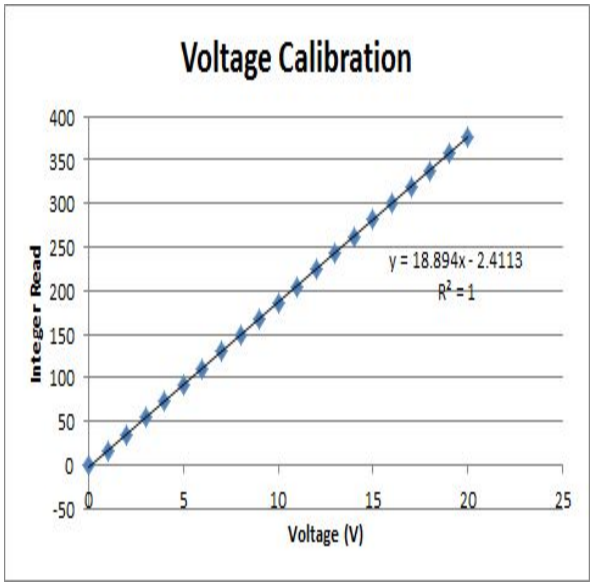
In addition to actually taking measurements, our device was required to store and transmit the measurements back to PATH. In order to do this, we chose to use existing microSD and GPRS shields (basically hardware add-ons for Arduino). The microSD shield allows the Arduino to write to a microSD card, which can store measurements if the power is interrupted or the device doesn’t have enough signal to send the measurements via SMS. The microSD shield is also very convenient, since the user simply needs to push a microSD card into the slot and plug the shield into the Arduino device; our code, which uses standard the Arduino SD library, handles the rest.

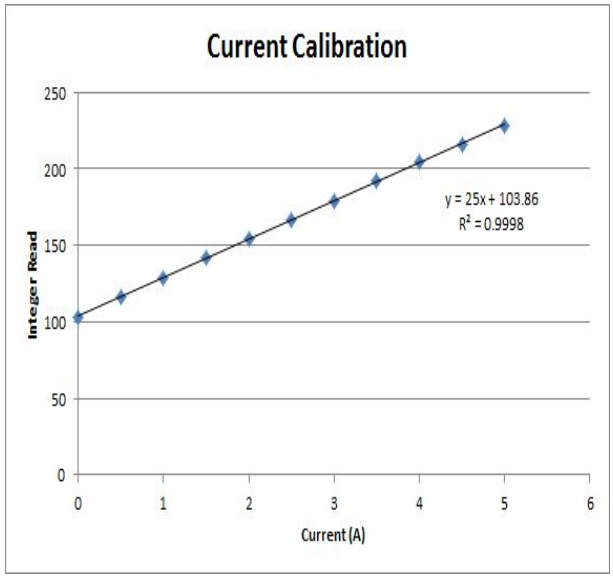
The GPRS shield works in a similar manner; the shield allows the device to send SMS messages to the website via GSM networks. Since almost every telephone network in the world uses GSM, this will allow the device to transmit measurements back to PATH almost anywhere in the world (as long as there is cell service). The GPRS requires only one thing from PATH: a valid SIM card registered to a GSM provider in the area. We have tested our device using a SIM card from a phone registered with AT&T; however, a fully operational device will require its own SIM card, which means purchasing a phone number for the device. Once the SIM card is placed in the SIM card holder, the GPRS shield registers to the network automatically. From there, the code on our device simply sends every measurement back to the phone number registered via Twilio to our website.

Last, but not least, we decided to add a real-time clock to our device. A real-time clock is powered by a small watch battery, which allows it to keep the actual date and time even if the device temporarily loses power. This will allow our device to include the date and time of each measurement in the messages sent to the website and files stored on the microSD card. This real-time clock is a safeguard in case the device loses power or cell signal; without the real-time clock, PATH would no longer be able to tell when each measurement was made, and the files stored on the microSD card would not be helpful in evaluating the potency of the vaccines stored in the fridge.

Overall, we are happy with how our device turned out. Our final device integrates our custom PCB board – containing the thermocouples, MAX 31855, current sensor, and voltage sensor circuit – and the GPRS shield, microSD card shield, and real-time clock. Our device is fairly robust – although there are some modifications we would like to make to the custom PCB board, discussed in the further work section – and has all the functionality originally requested by PATH. Just as importantly, almost all the parts are off-the-shelf parts which use standard Arduino libraries, so changing the design or manufacturing more of these devices should be fairly easy for PATH.

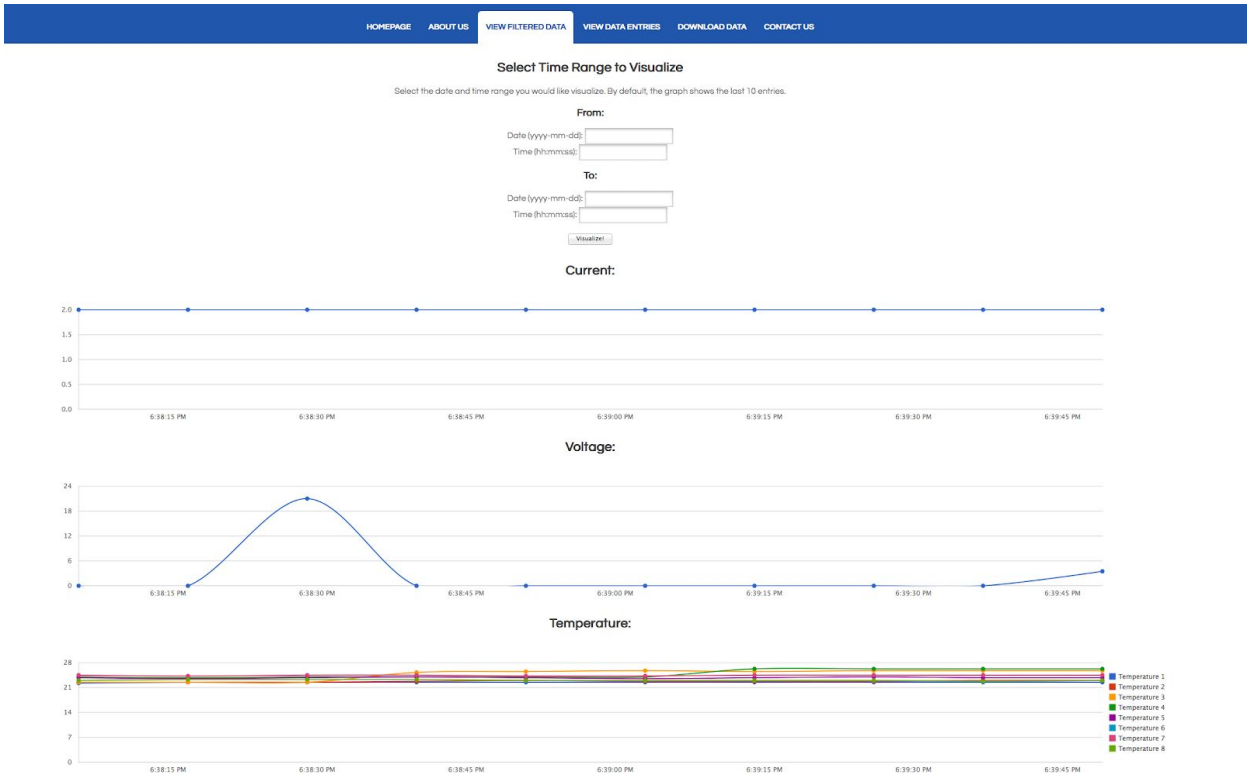
Included below are graphs showing the 10 bit integer read by the Arduino (0 to 1023) read for the current as well as voltage measurements. These graphs show that we have a linear relationship as desired and the equations fit to the graphs are used to calibrate the measurements.



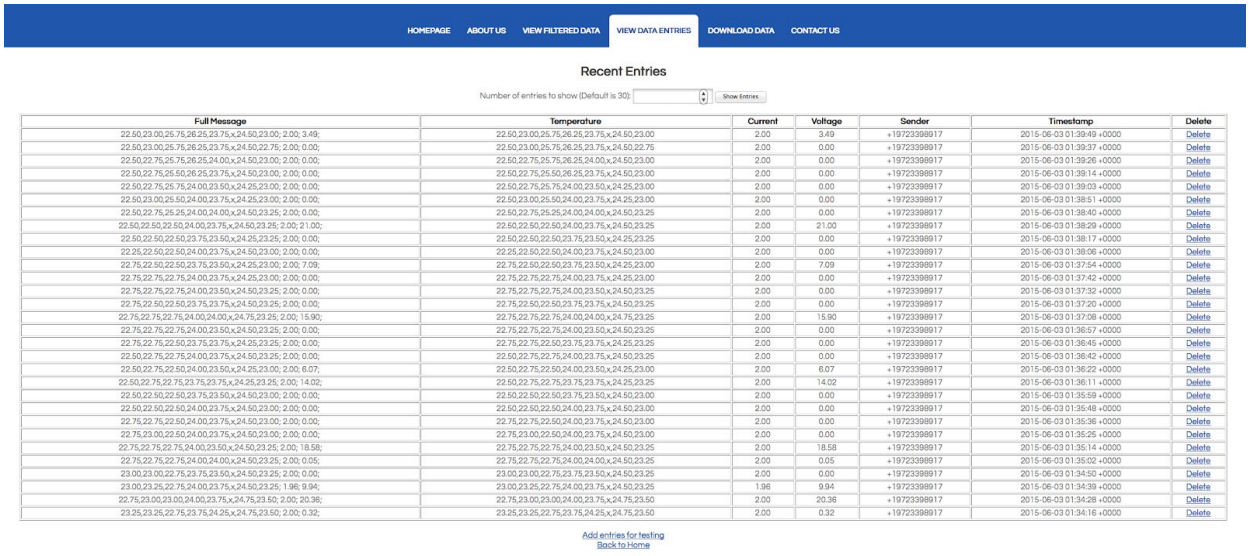




**Web Dev:** The web development team has created a web app that allows users to access our website (eswvaccineproject.herokuapp.com) and monitor, visualize and download the data from the vaccine monitoring device (powered by the Arduino). The app was built for hosting on Heroku's web servers. It is primarily coded in Ruby on the Sinatra framework, with HAML as the markup language and CSS for styling. There is a Postgres database that manages all data collection. Additional support came in the form of various gems, including the Chartkick gem for data visualization, and the Twilio API, for receiving SMS messages from the Arduino.

The web app provides three main features: data visualization, data logging and downloadable data, all of which can be filtered by date/time ranges, and are secured behind a password.

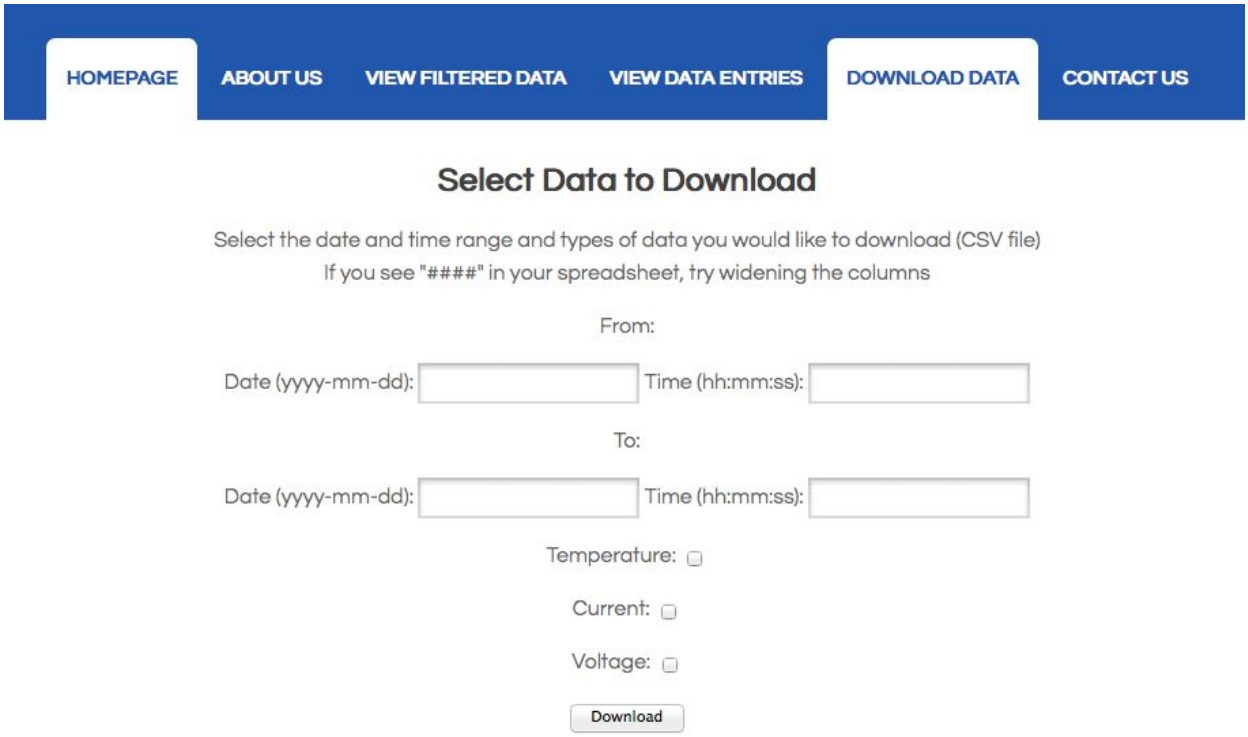



(Figure 1: Data visualization with the Chartkick gem with data received via SMS from the Arduino)




| Full Message | Temperature | Current | Voltage | Sender | Timestamp | Delete |
|---|---|---|---|---|---|---|
| 22.50,23.00,25.75,26.25,23.75,x,24.50,23.00; 2.00; 3.49; | 22.50,23.00,25.75,26.25,23.75,x,24.50,23.00 | 2.00 | 3.49 | +19723398917 | 2015-06-03 01:39:49 +0000 | Delete |
| 22.50,23.00,25.75,26.25,23.75,x,24.50,22.75; 2.00; 0.00; | 22.50,23.00,25.75,26.25,23.75,x,24.50,22.75 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:39:37 +0000 | Delete |
| 22.50,22.75,25.75,26.25,24.00,x,24.50,23.00; 2.00; 0.00; | 22.50,22.75,25.75,26.25,24.00,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:39:26 +0000 | Delete |
| 22.50,22.75,25.50,26.25,23.75,x,24.50,23.00; 2.00; 0.00; | 22.50,22.75,25.50,26.25,23.75,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:39:14 +0000 | Delete |
| 22.50,22.75,25.75,24.00,23.50,x,24.25,23.00; 2.00; 0.00; | 22.50,22.75,25.75,24.00,23.50,x,24.25,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:39:03 +0000 | Delete |
| 22.50,23.00,25.50,24.00,23.75,x,24.25,23.00; 2.00; 0.00; | 22.50,23.00,25.50,24.00,23.75,x,24.25,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:38:51 +0000 | Delete |
| 22.50,22.75,25.25,24.00,24.00,x,24.50,23.25; 2.00; 0.00; | 22.50,22.75,25.25,24.00,24.00,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:38:40 +0000 | Delete |
| 22.50,22.50,22.50,24.00,23.75,x,24.50,23.25; 2.00; 21.00; | 22.50,22.50,22.50,24.00,23.75,x,24.50,23.25 | 2.00 | 21.00 | +19723398917 | 2015-06-03 01:38:29 +0000 | Delete |
| 22.50,22.50,22.50,23.75,23.50,x,24.25,23.25; 2.00; 0.00; | 22.50,22.50,22.50,23.75,23.50,x,24.25,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:38:17 +0000 | Delete |
| 22.25,22.50,22.50,24.00,23.75,x,24.50,23.00; 2.00; 0.00; | 22.25,22.50,22.50,24.00,23.75,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:38:06 +0000 | Delete |
| 22.75,22.50,22.50,23.75,23.50,x,24.25,23.00; 2.00; 7.09; | 22.75,22.50,22.50,23.75,23.50,x,24.25,23.00 | 2.00 | 7.09 | +19723398917 | 2015-06-03 01:37:54 +0000 | Delete |
| 22.75,22.75,22.75,24.00,23.75,x,24.25,23.00; 2.00; 0.00; | 22.75,22.75,22.75,24.00,23.75,x,24.25,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:37:42 +0000 | Delete |
| 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25; 2.00; 0.00; | 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:37:32 +0000 | Delete |
| 22.75,22.50,22.50,23.75,23.75,x,24.50,23.25; 2.00; 0.00; | 22.75,22.50,22.50,23.75,23.75,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:37:20 +0000 | Delete |
| 22.75,22.75,22.75,24.00,24.00,x,24.75,23.25; 2.00; 15.90; | 22.75,22.75,22.75,24.00,24.00,x,24.75,23.25 | 2.00 | 15.90 | +19723398917 | 2015-06-03 01:37:08 +0000 | Delete |
| 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25; 2.00; 0.00; | 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:36:57 +0000 | Delete |
| 22.75,22.75,22.50,23.75,23.75,x,24.25,23.25; 2.00; 0.00; | 22.75,22.75,22.50,23.75,23.75,x,24.25,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:36:45 +0000 | Delete |
| 22.50,22.75,22.75,24.00,23.75,x,24.50,23.25; 2.00; 0.00; | 22.50,22.75,22.75,24.00,23.75,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:36:42 +0000 | Delete |
| 22.50,22.75,22.50,24.00,23.50,x,24.25,23.00; 2.00; 6.07; | 22.50,22.75,22.50,24.00,23.50,x,24.25,23.00 | 2.00 | 6.07 | +19723398917 | 2015-06-03 01:36:22 +0000 | Delete |
| 22.50,22.75,22.75,23.75,23.75,x,24.25,23.25; 2.00; 14.02; | 22.50,22.75,22.75,23.75,23.75,x,24.25,23.25 | 2.00 | 14.02 | +19723398917 | 2015-06-03 01:36:11 +0000 | Delete |
| 22.50,22.50,22.50,23.75,23.50,x,24.50,23.00; 2.00; 0.00; | 22.50,22.50,22.50,23.75,23.50,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:35:59 +0000 | Delete |
| 22.50,22.50,22.50,24.00,23.75,x,24.50,23.00; 2.00; 0.00; | 22.50,22.50,22.50,24.00,23.75,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:35:48 +0000 | Delete |
| 22.75,22.75,22.50,24.00,23.75,x,24.50,23.00; 2.00; 0.00; | 22.75,22.75,22.50,24.00,23.75,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:35:36 +0000 | Delete |
| 22.75,23.00,22.50,24.00,23.75,x,24.50,23.00; 2.00; 0.00; | 22.75,23.00,22.50,24.00,23.75,x,24.50,23.00 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:35:25 +0000 | Delete |
| 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25; 2.00; 18.58; | 22.75,22.75,22.75,24.00,23.50,x,24.50,23.25 | 2.00 | 18.58 | +19723398917 | 2015-06-03 01:35:14 +0000 | Delete |
| 22.75,22.75,22.75,24.00,24.00,x,24.50,23.25; 2.00; 0.05; | 22.75,22.75,22.75,24.00,24.00,x,24.50,23.25 | 2.00 | 0.05 | +19723398917 | 2015-06-03 01:35:02 +0000 | Delete |
| 23.00,23.00,22.75,23.75,23.50,x,24.50,23.25; 2.00; 0.00; | 23.00,23.00,22.75,23.75,23.50,x,24.50,23.25 | 2.00 | 0.00 | +19723398917 | 2015-06-03 01:34:50 +0000 | Delete |
| 23.00,23.25,22.75,24.00,23.75,x,24.50,23.25; 1.96; 9.94; | 23.00,23.25,22.75,24.00,23.75,x,24.50,23.25 | 1.96 | 9.94 | +19723398917 | 2015-06-03 01:34:39 +0000 | Delete |
| 22.75,23.00,23.00,24.00,23.75,x,24.75,23.50; 2.00; 20.36; | 22.75,23.00,23.00,24.00,23.75,x,24.75,23.50 | 2.00 | 20.36 | +19723398917 | 2015-06-03 01:34:28 +0000 | Delete |
| 23.25,23.25,22.75,23.75,24.25,x,24.75,23.50; 2.00; 0.32; | 23.25,23.25,22.75,23.75,24.25,x,24.75,23.50 | 2.00 | 0.32 | +19723398917 | 2015-06-03 01:34:16 +0000 | Delete |

(Figure 2: A log of received SMS data from the Arduino - there is also the option to manually add and delete entries for testing)




( Figure 3: Data downloading screen, with the ability to filter by date and hour ranges)

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Date 1 | 2015-06-01T | Date 2 | 2015-06-03T23:39 | | | | | | | | |
| 2 | Date | Current | Voltage | Temperature | | | | | | | | |
| 3 | 2015-06-03 ( | 2.04 | 0 | 24 | 24 | 21.5 | 25 | 24.75 | x | 25.25 | 24 | |
| 4 | 2015-06-03 ( | 2 | 0 | 23.75 | 24 | 21.75 | 25.25 | 24.75 | x | 25.5 | 24 | |
| 5 | 2015-06-03 ( | 2.11 | 0 | 23.75 | 24 | 21.75 | 25 | 24.75 | x | 15.75 | 24 | |
| 6 | 2015-06-03 ( | 1.96 | 0 | 23.75 | 24 | 21.75 | 25 | 24.5 | x | 25.25 | 24 | |
| 7 | 2015-06-03 ( | 2 | 0 | 23.5 | 24 | 21.75 | 25 | 24.75 | x | 25.25 | 24 | |
| 8 | 2015-06-03 ( | 2.04 | 0 | 23.75 | 24 | 21.75 | 25 | 24.5 | x | 25.25 | 24 | |
| 9 | 2015-06-03 ( | 2 | 0 | 23.5 | 24 | 21.75 | 25 | 24.75 | x | 25.25 | 24 | |
| 10 | 2015-06-03 ( | 1.96 | 13.54 | 24 | 21.25 | 21 | -32.5 | 27 | x | 4.25 | 24.25 | |
| 11 | 2015-06-03 ( | 2 | 0 | 23.75 | 24 | 21.5 | 25 | 24.5 | x | 25.5 | 23.75 | |
| 12 | 2015-06-03 ( | 1.96 | 14.13 | 23.25 | 24 | 21.5 | 25 | 24.5 | x | 25.25 | 24 | |
| 13 | 2015-06-03 ( | 2 | 0 | 23.5 | 24 | 21.5 | 25 | 24.75 | x | 25.25 | 24 | |
| 14 | 2015-06-03 ( | 2.04 | 0 | 23.75 | 24.25 | 21.5 | 25 | 24.75 | x | 25.25 | 24.25 | |
| 15 | 2015-06-03 ( | 2.04 | 0 | 24 | 24.25 | 21.75 | 24.75 | 25 | x | 25.5 | 24.25 | |
| 16 | 2015-06-03 ( | 2 | 0 | 23.75 | 24.25 | 21.5 | 25 | 25.25 | x | 25.5 | 24.25 | |
| 17 | 2015-06-03 ( | 1.96 | 10.74 | 24 | 24.25 | 21.75 | 25 | 25 | x | 25.5 | 24.25 | |
| 18 | 2015-06-03 ( | 2 | 0 | 23.75 | 24.25 | 21.75 | -35 | 25.25 | x | 25.5 | 24.25 | |
| 19 | 2015-06-03 ( | 2 | 0.05 | 23.75 | 24.25 | 21.75 | 27 | 24.75 | x | 25.5 | 24 | |
| 20 | 2015-06-03 ( | 1.96 | 8.16 | 23.75 | 24.25 | 27.25 | 28 | 24.75 | x | 25.25 | 24 | |
| 21 | 2015-06-03 ( | 2 | 0 | 23.75 | 24.5 | 25 | 26.5 | 24.75 | x | 25.75 | 24.25 | |
| 22 | 2015-06-03 ( | 2.04 | 0 | 23.75 | 24.25 | 27.5 | 28.25 | 25.25 | x | 25.5 | 24 | |
| 23 | 2015-06-03 ( | 2.04 | 0.16 | 23.75 | 24 | 26 | 27.75 | 24.75 | x | 25.75 | 24 | |
| 24 | 2015-06-03 ( | 2 | 0.05 | 24 | 24 | 28.25 | 29 | 25 | x | 25.5 | 24.25 | |
| 25 | 2015-06-03 ( | 1.96 | 5.8 | 23.75 | 24.25 | 28.25 | 29 | 25 | x | 25.75 | 24.25 | |
| 26 | 2015-06-03 ( | 2 | 0 | 24.25 | 24.5 | 28 | 29 | 25.25 | x | 25.75 | 24.25 | |
| 27 | 2015-06-03 ( | 2 | 0 | 24.5 | 25 | 28.25 | 29 | 25.5 | x | 26 | 24.75 | |
| 28 | 2015-06-03 ( | 1.96 | 9.78 | 24.25 | 24.5 | 28 | 29 | 25.5 | x | 26 | 24.5 | |
| 29 | 2015-06-03 ( | 2 | 0 | 24.25 | 24.5 | x | x | 25.5 | x | 26 | 24.75 | |
| 30 | 2015-06-03 ( | 2.04 | 0 | 26 | 24.5 | 22 | 28 | 25 | x | 26.25 | 24.75 | |
| 31 | 2015-06-03 ( | 2 | 0 | 24.5 | 24.5 | 20.75 | 26.25 | 25.25 | x | 26 | 24.75 | |
| 32 | 2015-06-03 ( | 1.96 | 8.43 | 24.25 | 24.75 | 20.5 | 26 | 25.25 | x | 25.75 | 24.5 | |

(Figure 4: A CSV file of the downloaded data)

**Project Sustainability (10 pts)**

Two key goals that we kept in mind when building our device and website were robustness and reproducibility. The goal of our project was to create a prototype for a remote vaccine monitoring system for PATH; however, if our prototype is successful, we hope that our device will be reproduced and used at PATH sites around the world. This motivated our group to make the design simple enough to easily replicate and create in large quantities. Secondly, the fact that our design might eventually be used in diverse climates required us to ensure the robustness of our device.

Many of the robustness considerations were actually handled by PATH, and provided to us in the specifications for the project. In particular, PATH asked that our device use 8 type T thermocouples, which provides redundancy in case one of the thermocouples fails or breaks. Since the device will eventually be powered from the same solar panels which power the vaccine refrigerators – which may provide intermittent power – PATH also asked us to store all the measurements on non-volatile memory. We decided to do this using a microSD card, which retains information written to it when it loses power, and is easy to access from a computer. Finally, PATH asked that our product be capable of communicating with PATH either via wireless internet or telephone networks. We decided to transmit messages to our website using GSM, the global system for mobile communications, since GSM is much more common than wireless internet, especially in some of the low resource settings where PATH operates.

Our second main consideration while designing this project was reproducibility. In order to make our design easily reproducible, we focused on using off the shelf components, and providing thorough documentation. Using off-shelf-components makes it much more likely that other people will be able to successfully assemble our device. In particular, we also tried to use standard parts, which can be bought in most parts of the world, since this device may not only be used, but created, somewhere far outside the US. Using off-the-shelf components was especially important for the device sub-team (since the website only needs to be created once, while the device needs to be replicated for each fridge PATH intends to monitor). Our group was successful in using almost only off-the-shelf components: the device requires an Arduino Uno, a microSD and GPRS shield for Arduino, and a few electrical components readily available from large, well-established companies. The only custom part of our device is the custom PCB board, which contains the circuits necessary for reading the temperatures from the thermocouples; however, we plan to give the design to PATH, which can then order our custom PCB boards..

In addition, we documented our device and website. We created both a user guide and an assembly guide for the device, and documentation for the website. In particular, the user guide shows how to set up the device – from installing the SIM card in the GPRS shield so that the device can send SMS messages to turning on the device. The user guide basically troubleshoots many issues someone receiving an assembled device might encounter. The assembly guide contains much more detailed documentation, explaining both how to assemble the device and the reasoning behind decisions like the type of chip used to measure the current.

Overall, our group decided to approach project sustainability from two angles, robustness and reproducibility. Robustness required our device to handle different operating

conditions, and was mostly specified by PATH. The reproducibility aspect involved using easily acquired components, documenting our design and explaining how to use our device.

**Conclusion (10 pts)**

Our project had many difficulties this quarter, but in the end, we as a team agree that we are delivering a well tested, well put together project to PATH, and are very proud of the work we did this quarter. Our initial difficulties were with the individual components. We had difficulty setting up the GPRS shield because; much our to chagrin, we learned that we were trying to use a SIM card registered to CDMA carrier (Verizon), which does not use the standard GSM protocol to send SMS messages. Thankfully, there was an easy solution – finding and using a SIM card registered to a GSM carrier (AT&T and T-Mobile in the US). The 8 thermocouples resulted in messy wiring which difficult to reproduce correctly. This issue was eventually resolved by creating a custom PCB board, which contains most of the wiring in its traces. For the current and voltage sensors we were initially puzzled to have the measurements be slightly off from what we expected, but resolved this by realizing that we could simply correct the measurements empirically in software.

With the individual components working the next challenge was integration. We had to carefully map Arduino pins to each component, eventually having just enough pins for all the components. A significant challenge was making a PCB board for the first time. Once we had the PCB board, we spent many hours soldering and resoldering the components. Some of this difficulty could have been resolved by modifying the PCB board discussed in further work. However, much of the difficulty also appeared to be inherent in attempting to solder so many miniscule chip and our lack of experience. A final difficulty we faced was by having a higher current to be read specified in the last weeks of our project. This required swapping our previous current sensor chip with a new one and also having a mentor add copper traces above the board for us to carry the larger amount of current.

On the web development side, we faced issues with software installs, cross-browser compatibility, file writing with an ephemeral filesystem, and database connections. Early on in the quarter, when we decided to build the web application on Heroku, ⅔ team members on the web dev team could not download the needed Heroku software, which led to Emily being the only one with complete access to the database. We partially solved this problem by setting up the project through Github, so that we could all push code (though we could never test locally, so if we had errors, they were directly affecting the production site). We ran into cross-browser compatibility issues with our original graphing library (D3) in that the graphs wouldn't show up in Google Chrome. We remedied this by changing our visualizations to be generated from a Ruby gem called Chartkick. With file writing and our database, we simply struggled with the problems for a few weeks, doing a lot of researching and asking for help from more experienced web programmers. In the end, we solved both of the problems ourselves, but had spent more time on them than initially thought.

One of the big things that helped us this quarter was the meeting notes form and weekly meetings we set up at a team and as subteams. By breaking down tasks, deciding what we needed to get done each week, and then meeting to ensure we met our action items was quite beneficial to our team's productivity. Having regular meetings in subteams was very helpful for collaboration. There were many times when one person would be struggling with a feature or implementation, and the brain power, support and creativity from other team members helped them through it.

Our project is being sent to PATH in Seattle this summer, where they will conduct preliminary testing of our monitoring device and web application. Though it may take multiple iterations, we hope eventually our device will make it out into the field to monitor vaccine fridges. Children in countries from all over the world could potentially benefit from the information collected by our device. Hundreds of lives could be saved by keeping track of vaccine usability and fridge conditions. Additionally, this could also help companies commercially by giving feedback about their fridges and maybe showing them that they can make their products more efficient. Finally, our project also eases the job of health workers, by relieving them of some of their current monitoring and logging duties and giving them more time to work directly with patients.

**Further Work (5 pts)**

Web: The web application is fully functional as it is and does what we set out to do; however, in order to make the project more economical and convenient to use, there are some improvements we could make. First of all, the use of services such as Twilio and Heroku worked fine for our purposes, but they are not necessarily sustainable on a large scale if our project were to be implemented globally. Twilio SMS costs are very low in the US ($0.0075/message), while they are significantly higher in other countries (~$0.02/message), and although $0.02 does not seem like a lot, the cost quickly adds up if we are sending SMS messages every 30 seconds as intended. One potential solution would be to send messages less frequently and/or batch multiple data entries into a single SMS. Another solution would be to send SMS messages to email, and read data entries from email in order to bypass Twilio altogether; we could also consider other options such as using wifi when available. Additionally, although Heroku is a free service, the amount of database space is limited and the application requires some downtime which can lead to application failure if too many requests are received in a short amount of time. When PATH receives our project, they would probably need to either upgrade Heroku for approximately $10/month, or they could consider using their own servers to host the website to avoid Heroku costs. We might also need to think about how to handle large quantities of data -- for example, we could download the data and clear out the database at regular intervals.

The web application itself could also be improved to be made more user-friendly and convenient. Currently, the time in the database is stored in UTC time, which is somewhat inconvenient to account for if one is not in that time zone. We could add additional support for time zones to improve usability. Additionally, we could add support for a time entry from the Arduino and allow data to be visualized based on the time it was measured rather than the time

it was received. Furthermore, we could also potentially make our data downloading or graphing pages more convenient to use by adding additional error-checking and adding default settings instead of forcing the user to retype dates or times that are commonly used. We might also make a more secure password, or add different levels of login permissions so to further secure areas of the website that allow users to add or delete data. Nonetheless, our web application is completely functional as it is now, so any next steps would serve to reduce costs and improve usability.

Arduino: Although the Arduino sub-team is fairly happy with our final device as we feel we were able to meet our core objectives. There are some changes we would make if we had time. For our PCB board we initially designed our board to be very compact since we wanted it to be the same size as an arduino. For several of the components, the thermocouples in particular, we designed the holes to be just large enough for the wires to fit. In practice this meant the holes were a little too small and it took effort to push the thermocouple wires through the holes. In future iterations we would expand the size of our board, and make the holes larger and more spread out. With a larger PCB board we would also like to add a neater interface for connecting current and voltage sensing wires to the device, possibly with a screw terminal. Additional possible improvements include having Wi-Fi capability as well SMS capability to reduce data transmission costs where possible and having the Arduino be powered from a solar panel rather than a wall outlet

If doing another iteration of this project, we would also consider ways of lowering the total cost of our device. The components we choose including the Arduino and several of the shields have the advantage of being easy to use, reliable, easy to expand, and off-the-shelf. Unfortunately, off-the-shelf components tend to be associated with a higher cost so it would be interesting to consider cheaper alternatives. As others in ESW have pointed out, developing a low cost but easy to use microcontroller and sms transmitter, would be of use to numerous projects, and would be a useful broader problem to consider.

## APPENDIX

**Budget and Funding (5 pts)**

We estimate that the total cost of a single unit of the Arduino device is close to 150 dollars. The ongoing costs that would be associated with the website are attached. We've also attached the estimated cost for this project, though this is more than the cost of a single Arduino. PATH initially shipped us any components that we requested. Later we obtained funding from The Stanford Fund and the School of Engineering and began to purchase components directly. We expect PATH to fund the website maintenance costs for their use.

**Team Management (5 pts)**

Emily: This quarter, I was a member of the web development team, where we focused on creating a web application that could receive data from the Arduino and parse it so that the data could be visualized and downloaded. Jinhie, Nathalia, and I worked closely to create the website from start to finish. Due to technical difficulties with Ruby and Heroku, my computer was the only one that could actually host the application, so I was responsible for initial set-up of the application, managing the various accounts, and setting up the application so that others could also contribute. I also worked on connecting the web application to Twilio, setting up the database, parsing SMS messages to go in the database, and implementing a CSS template for the website; later on, I focused on data visualization with Chartkick and data customization. Throughout the project, the web development team all worked very closely on all aspects of the web app to implement new features and debug any issues that arose.

Jinhie: As a member of the web development team, I worked very closely with Emily and Nathalia through many group coding sessions to write the web application that receives, stores, and processes the data from the Arduino. At the beginning of the quarter, I focused on writing scripts we could run on the Raspberry Pi to download and parse emails to recover the data sent from the Arduino. However, after our team decided to use Twilio instead to receive and parse the data sent from the Arduino, I shifted my focus and primarily worked on the data visualization, first with D3 and later with chartkick, and customization of the data shown in the visualizations and downloaded file as well as preliminary error checking.

Julien: My role this quarter was to help the Arduino subteam finish the Arduino controlled device. In particular, I focused on working with the GPRS shield to get the Arduino to send SMS messages. Once the GPRS shield was working properly, I worked with Moosa and Helen to set up the thermocouple circuitry, plan out our PCB board, and assemble all the pieces - microSD card reader, GPRS shield, real-time clock, thermocouples and current sensors - into a single working device.

Helen: As a member of the Arduino team, I was primarily responsible for the temperature sensors. As part of this job, I selected, purchased, mounted, and found code for the thermocouple detector library, and researched the proper IC chips to purchase to integrate the thermocouples with the Arduino. Later in the quarter, I worked with Moosa and Julien to

integrate the code for several sensing components into one file for the Arduino. I also assisted in selecting casing for the hardware component, debugging the GSM shield, and briefly assisted the web development team in data visualization.

Moosa: As part of the Arduino team, the subcomponents I primarily focused on were the current sensor, the voltage sensor, the SD card reader, and the RTC clock. I helped Julien with some of the debugging for the GPRS shield. I also designed the PCB board on which we ultimately placed the temperature, voltage, and current sensing chips on to connect to the arduino. This project was my first time designing a PCB board. Once the PCB boards arrived I worked with Helen and Julien to assemble on the components together. We spent a significant amount of time resoldering and debugging when the components did not work initially. As project coordinator I completed the duties that would be expected of a project coordinator, including communicating with PATH, ordering and picking up the majority of components, leading meetings, helping plan out our project, and getting us access to lab space in Packard.

Nathalia: This quarter I was on the web development team, which focused on creating a web application to view and download the data being sent back from the Arduino via SMS. Along with Jinhie and Emily, I researched everything from Twilio integration to file system writing to overall web app architecture. We worked very closely together to bring this project online, often researching, implementing and debugging features while consulting each other in the same room. In particular for this project, I worked on researching Twilio and introducing it as a basis for our project (effectively overhauling the Pi's function), first stage data visualization (with D3, which was later overhauled with the Chartkick gem) and working on the file writing and downloading of data.

Mentors: Professor Boris Murmann, Steven Clark (Packard Lab Manager).

**Gantt Chart/Timeline (5 pts)**

attached

**Weekly Group Meeting Attendance (Graded separately)**

attached

**Relevant Links (Web App)**

https://github.com/eling8/eswvaccineproject
https://eswvaccineproject.herokuapp.com/

**Hardware User Guide**

attached