5-1-2019

Emergency Water Station Final Design Report

Daniel Dahlinger
Trinity University

Sean Farrell
Trinity University

Bryan Granizo
Trinity University

Trevor Johnson
Trinity University

Kenneth L. Kusima
Trinity University

See next page for additional authors

Authors

Daniel Dahlinger, Sean Farrell, Bryan Granizo, Trevor Johnson, Kenneth L. Kusima, Caroline Kutach, Nathan Richter, Gabriel Righes, Jessica Rodriguez, and Emma Williams

Emergency Water Station Final Design Report

Daniel Dahlinger, Sean Farrell, Bryan Granizo, Trevor Johnson,
Kenneth Kusima, Caroline Kutach, Nathan Richter, Gabriel Righes,
Jessica Rodriguez, Emma Williams

Team Advisor: Mehran Aminian

May 1, 2019

Executive Summary

To:	Dr. Darin George
From:	Emergency Water Station Group
Subject:	Emergency Water Station Final Design Report
Date:	05/01/2019
CC:	Dr. Mehran Aminian
	Dr. Farzan Aminian
	Eddie Canales, South Texas Human Rights Center
	International Committee of the Red Cross

The following objectives were presented to the Emergency Water Station Group to create a station to aid South Texas Human Rights Center volunteers in providing water for dehydrated refugees:

- The system must sustain power to controllers, an LED, and a phone charger continuously
- The system should monitor GPS coordinates and the number of water jugs present in the station and upload this data to a website automatically
- The system should include a base to house the electronics and medical supplies, and also support a 27.5-ft flagpole while keeping all compartments watertight through common weather conditions

This report outlines the logistics of the created system as well as the tests conducted to evaluate its effectiveness.

The waterproofing tests concluded that seals on the electrical drawer, USB cover, and load cells successfully impeded water from entering any electrical compartments. The flagpole tests also proved successful because there was no plastic deformation, no base tipping, and no base roof shearing. Phone charger tests proved that the charger could successfully power three different phone brands. Weight sensor tests showed low drifting with temperature variations from 15 to 45 degrees Celsius and extended amounts of time. Similarly, the power system tests verified that the entire system was self sustaining for three days. Current tests showed little variation; however, there was significant noise present in the battery voltage readings. The communication system successfully transmitted the barrel weight and location of the station to the website over a full system test of three days.

It is recommended to improve the voltage reader to ensure more reliability and accuracy. There was drifting in the time readings for the Real Time Clock (RTC), which could be prevented using the Adafruit DS3231 RTC. It is also recommended that the satellite data sensing be optimized using an algorithm. Ideally, with further testing and research the entire system could operate without the need for a master Raspberry Pi, as it currently works as a middle-man for the communications system to collect data from all stations.

Introduction

Refugees along the Texas border are often subjected to critical dehydration. The primary goal of the Emergency Water Station project was to aid the volunteers of the South Texas Human Rights Center (STHRC) in providing water to refugees. Project objectives included creating an automated station with a means to protect sealed gallon water jugs and support a 27-ft flagpole. The system also needed to automatically determine the number of jugs present in the station and its GPS coordinates update a website using this data. Additionally, the flagpole needed to be topped by an LED which became lit at night and the station needed to include a phone charger, where all electronic compartments were completely waterproof. Constraints included the dimensions of the STHRC water barrels (20" diameter bottom, 23.5" diameter top, 38" height), as one barrel needed to fit in each station, and the dimensions of a standard truck bed of (78" x 64"), as the disassembled station needed to fit in a truck bed. The design must follow NEC grounding standards, FCC/wireless communication standards, and the IEC Code 60529: Ingress Protection Marking.

The base of the station was tested for waterproofing, stability, and flagpole support with successful reports on all accounts. The power system was tested for consistency over time in the weight sensors, current sensors, voltage sensors, and overall sustainability. While the power system was self sustaining for three days and showed steady results for the weight sensors and current sensors, the voltage sensors showed significant variations. Note that the temperature was varied from 15 to 45 degrees Celsius. The communications system was tested for an accurate flow of data from the RockBlock all the way to the BlueHost database. The system should be fully capable to accurately transmit the weight, location, time and temperature for each deployed station.

Overview of the Design as Tested

Power and Electrical Hardware

Designing the power system required first selecting the components required to fulfill the design requirements, as detailed in the Project Charter. These components, their respective current draw and their use time are given in Table 1. The combined current draw was used to size the solar panel and battery used in the station. Current draw was used to size the power production and storage requirements of the station (as opposed to power consumption of each component) because most of the components require lower voltages than provided by the battery. These voltages would need to be regulated down from battery voltage, which means there would be electrical losses. Thus, tracking current consumption provides a degree of safety in the sizing calculations.

Table 1: Power System Requirements.

Item	Number	Voltage Required (V)	Current (mA)	Use Time (hr)	Charge (Ah)
Arduino MEGA	1	10	50	24	1.20
Rockblock	1	5	500*	0.5	0.25
Weight Sensors	3	5	5	24	0.48
Phone Charger	1	5	1000	4	4
*Worst case current draw				Total Ah/day	5.93

Table 1 allowed the power system team to size the battery. In total, it was estimated the system requires 5.93 Ah worth of charge each day. Past iterations of this project used two 10 Ah batteries in parallel to provide the necessary capacity for the system [1]; however, these systems also had charging issues, which was most likely related to using two batteries in parallel. Based on this, a single 20 Ah, 112V, lithium-iron phosphate (LiFePO4) battery was chosen to fulfill the capacity requirements of the design. LiFePO4 was chosen because it can be recharged many times, retains the same voltage over most of its discharge cycle and was used in past project iterations with success. Meanwhile, 12V were chosen because the Arduino takes 10V of power to operate correctly, and 12V was the closest available battery voltage. With the previously stated charge requirements, the power system can power the station for three days without being charged.

The electrical system is powered by a 50W, 12V solar panel. This size was chosen because in previous iterations of the project (Phase 1), the station batteries would sometimes drain completely. Since this previous iteration was powered by a 30W solar panel and Phase 2 would also include a phone charger, the 30W was upgraded to a 50W panel to prevent any power issues and provide sufficient power for the phone charger. The SC-122420JUD solar charge controller was chosen to regulate the current from the solar panel and charge the station's battery. This charge controller has a maximum current rating of 20A[2], well above the 3.1A [3] maximum solar panel current.

The station is controlled by an Arduino MEGA. This microcontroller was chosen due to its large number of I/O pins and available program memory. This microcontroller has a max input voltage rating of 12V and since the battery is only nominally 12V (in reality it produces 13V) the UA7810, a 10V voltage regulator, was included to safely regulate the voltage from the battery. This microcontroller also contained a 5V power pin, which was used to power any devices running off 5V.

Users of the water station will charge their phones from a USB-A female port. The design team chose to implement this type of port over other types because most, if not all modern phones use a USB-A cable, meaning any user should be able to charge their phone from the station. The standard USB-A port has four pins, two of which are used for power, which the station will provide at 5V and a maximum of 1A, and two of which are for data transfer. These data pins may also be used to indicate how much current a charger can provide to a phone, which the phone uses to control the current it draws [4]. One of these data lines is held at 2.2V, while the other is held at 2.8V, to indicate to the phone how much current it can draw from the charger. To provide the 1A of current at 5V a R-78E DC-DC buck converter was chosen. This This converter is capable of converting a input of up to 28V down to 5V at an efficiency of up at least 70% [5]. Additionally, this converter can supply a current of up to 1A, which fulfills the design target. A DC-DC converter was chosen as opposed to a voltage regulator since regulating from 12V to 5V at 1A would have wasted a significant amount of power.

To keep track of the amount of water in each station, the power design team chose the FC2231 load cell. This load cell works by harnessing the changing resistance of a strain gauge due to the compressive force on a metal beam. In each of the FC2231's four of these strain gauges are used in a wheatstone configuration, which allows the small change in resistance to be translated into a measurable change in voltage. This change in voltage is amplified, output as an analog signal and interpreted by the Arduino. These load cells each support 100lbs and the design uses 3 of them, meaning the scale has a maximum rating of 300lbs. A self-contained scale was constructed and the designs are shown in Figure B1 of Appendix B.

These components and their connection to each other are summarized in the schematic of Figure 1. This schematic differs from the design proposed in Fall 2018, shown in Figure C1, in the following ways. First, the cell shield was replaced by a satellite communication unit. Since communication happens at specific times during the day, a PCF8523 real time clock (RTC) was added to the design to keep track of the time, even if the main battery is shut off. Because the cell shield was no longer included in the design, the 3.7 backup battery was also removed, along with the MAX642 DC-DC buck converter and power relay. The resistive load cells were replaced with FC2231-0100 load cells while data logging tools, non-invasive debugging/status lights and additional sensors were added to the design. These sensors consisted of a DHT11 temperature/humidity sensor and an ACS723 current sensor. This current sensor was chosen because it could measure currents up to 5A, higher than the maximum solar panel current. The DFRobot SD card module and 8GB SD card was added for data-logging.

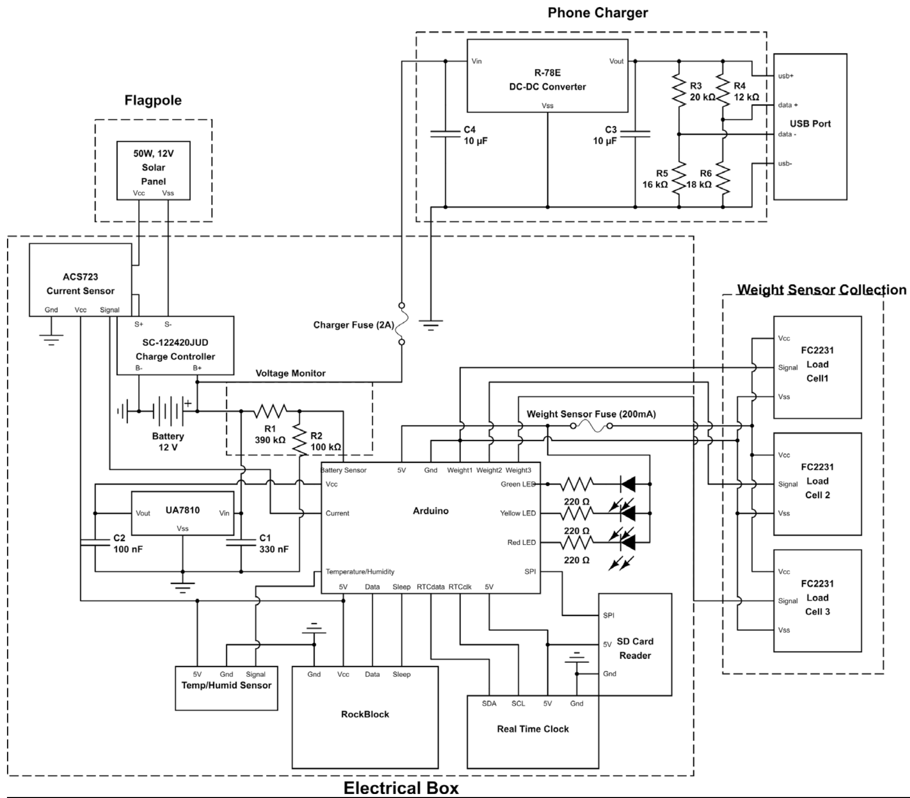

Figure 1: Updated electrical schematic.

The components within the “electrical box” of Figure X were mounted to a removable mounting plate which consisted of 0.5” thick HDPE. The design for this plate is shown in Figure B2. This plate was bolted to the bottom of the junction box to secure the electrical system. A special enclosure, shown in Figure B3 was designed for satellite module to ensure its antenna faced upwards to the sky. Three circuit boards, one to distribute power to all the electrical hardware, one containing the phone charging circuit and one containing debugging hardware were designed in eagle and routed using the Bantham routing machine. The board layouts for the power, phone charging and debugging circuit are shown respectively in Figure C2, C3 and C4. A photo of the finished electrical box, with the mounting plate installed, is shown in Figure C5.

Base Design

In order to design the base, all of the dimensions for the electronics compartment needed to be approximated as reasonable as possible. The electronics are a top priority because without safe housing, those components are rendered useless and the scope of the project would not be satisfied. The final design chosen was named ‘Pole Insertion’, due to how it supports the flagpole; namely, a separate mechanism that is inserted through the middle of the base. Figure 2 shows a Creo Parametric drawing of the final design.

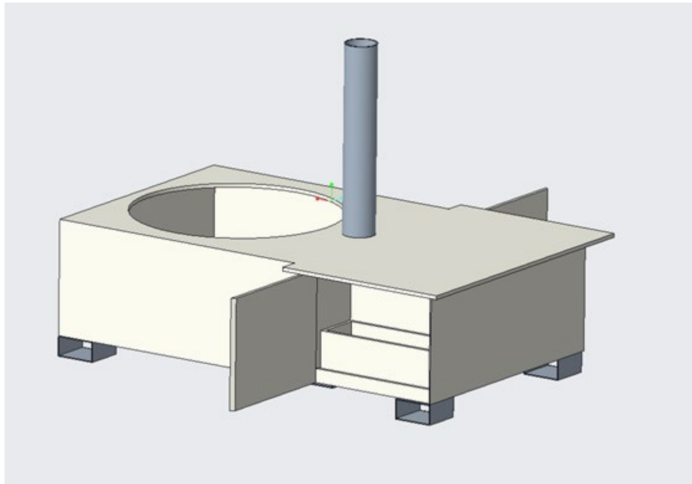

Figure 2: CAD model of the completed base apparatus

The flag pole will be placed in the pole support, displayed in Figure 2. The pole support is made from steel tubing that is 30 inches tall. Research revealed the industry standard for 30 foot flag poles is 30 inches, and even though the flagpole that will be used for our project is only 27.5 feet, we wanted to take extra steps to ensure the stability of the pole. The base is also elevated two inches off the ground by rectangular steel tubing, preventing the base from completely submerging in water in the event of heavy rain. There are two compartments in the base, one for electrolytes and medicine, and the other for the junction box. These two compartments were both fitted to allow room for workers in case of repairs or restocking. There will be a drawer in the electronics compartment that the junction box will be mounted on, and this drawer will be attached to guide rails on the side that allow the box to slide out for maintenance. One new feature on the prototype that is not shown in this design is that the door to the electronics compartment will not be attached with hinges but instead is attached to the drawer and will slide with it. This was done to make the compartment more water-proof and also to be locked easier.

Another defining feature of this design is the open area surrounding the water barrel, will be utilized by storing additional weight for the stability of the flagpole. Not featured in Figure 2 is the USB port that will be on the outside of the base. This feature is new to this iteration but it allows for station users to plug in and charge their phones, which can help save lives in the event of emergency. The USB port will be located on the outer face of the electronics compartment, covered by the extended overhang of the top face of the base. In order to protect the port from the elements, an external cover will be attached, similar to what is used for outdoor electricity ports.

Communication and Control Design

In order to design the communication system part of the water station a clear understanding of this system's requirements needed to be established. These requirements were to develop a reliable way of transmitting the amount of water jugs a station contains, its GPS location, and the stations electronic performance to the South Texas Human Rights Center bluehost website. A high level overview of the complete communication system is shown in Figure 3. Inside each station there is an Arduino Mega microcontroller connected to a RockBlock Mk2 satellite modem. The specific wiring connections can be found in Table D1. The Arduino preps an array of characters represented in hexadecimal, that it then sends to the RockBlock using UART communication. The RockBlock then establishes connection to several of

Iridium's global satellite network. It transmits the data from the satellite to email addresses recorded in the test group. To alter what emails are saved in each test group see Instructions for Accessing Each Satellite Modem (For Future Senior Designers). We used a RaspBerry Pi controller to automatically process the emails sent from the RockBlock. The processing comprises of decoding the data contained in the emails from hexadecimal to decimal values. These values are then stored in a .csv file and uploaded as a table onto the Bluehost website.

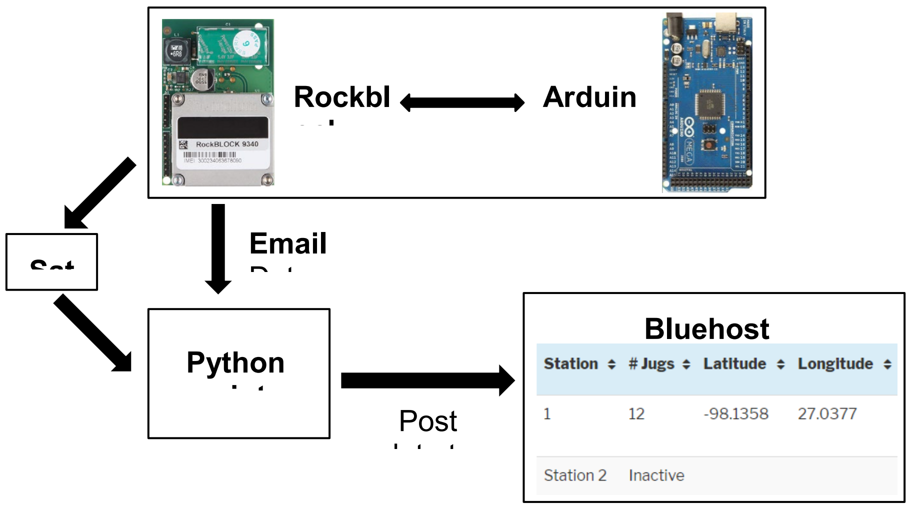

Figure 3. High-level communication flowchart.

The flowchart for the main program running on the Arduino Mega within each water station is shown in Figure. D1. The structure of the software is to first initialize all of the hardware modules: real time clock (RTC), SD card, temperature/humidity sensor, and RockBlock. Once the modules are initialized the program immediately reads the current barrel weight (lbs), battery voltage (V), and current from the solar panel in (mA). This data is first stored on the 8 GB SD card with the appropriate time stamp. This data logging was done in order to help in testing the water station and evaluating its overall performance.

After the data is stored on the SD card it is similarly saved into a character data array of size [0:47]. This type of array was used in order maximize the amount of information we could send with each transmittion. Before this array is transmitted by the RockBlock we developed a sending algorithm that checked to make sure there was a signal strength above 3 (scale 0-5). If the signal strength was below this threshold then the main loop will set a send error flag and continue performing other important tasks. Two minutes later it will check the signal strength again and perform the same logic. It continues this process until it reaches the maximum number of set attempts (20). If the maximum number of sending attempts is reached then it will not send data at that time and thus wait until the flag is set again either by an alarm triggering or the morning/evening time being reached. The very end of the main loop checks to see if it needs to save data to the SD card and if it should set any of the alarms. Currently, the software is set to only trigger alarms if the weight of the water barrel reaches a minimum or maximum value. However, the code has been developed to handle alarm triggers for panel current and battery voltage readings.

Within the main program running on the Arduino there are two very important subroutine functions. The first is the checktime function, which monitors all of the communication system time sensitive functions. The flowchart for this function is presented in Figure D2. It first records the current date and time saved on the RTC. After comparing it to pre-programmed timing thresholds it sets the sat_send_flag, sat_save_flag, and SD_save_flag accordingly. This function is time keeper for the whole system and is a critical component within the whole communication system.

The second very important subroutine function is the satellite sending data function. The general flowchart for this program is shown in Figure D3. As the name describes, the main goal of this program is to send the character data array from the water station to a satellite using the RockBlock. The program begins by initializing the RockBlock modem, then checking the signal quality again. It converts the data array into a binary array, which is needed for transmission, and then proceeds to send the data to the satellite. This function is how the data from the water station reaches the next part of the communication process which is the Python email processing.

The networking Python code running on the Raspberry Pi is essential for transferring the data sent from the RockBlock in an email to the Bluehost website. The general flowchart for the Python software is shown in Figure D4. The program begins by checking to see if any of the RockBlock modules had sent emails to the bot email address: trinitywaterstations@gmail.com (password: Water!Stations18, imap server: imap.gmail.com). If an email is in the inbox the program extracts the important data out of the email and decodes it into decimal values. The decimal values correspond to the barrel weight, battery voltage, and panel current recorded at the station. Once the barrel weight is converted to jugs by dividing the total weight (lbs) by the weight of one water jug (8.34 lb), all of the data is stored into a .csv file. The program then logs into the Bluehost website and uploads the file into a table on the webpage. Once the data has been uploaded it can be viewed by the employees of the STHRC.

Grounding

To observe the safety of the people using the station, and to adhere to the NEC codes and standards regarding the installation of electrical systems, priority was given to ensure an adequate grounding system was implemented while still meeting the set SMART goals of the project. Consideration was given for the use of arresters, grounding plates and grounding rods, of which we chose to move forward with the grounding rods. This system was chosen based on its superior effectiveness and ease in set up. We were however, tasked with making sure that the station will still be able to retain its portability. Due to the inexpensive nature of the grounding rods, we intend to have a grounding rod prepared for every station installation. Grounding rod available for use in the prototype was ⅝ inch in diameter and 8 ft tall. Driving in the grounding rod is dependent upon the type of soil. As a result, a pole driver and rotary hammer drill were purchased for installation in the softer and harder soil respectively.

The drill purchased allows for custom dewalt drill bits to be used and so, a rotary hammer adapter was obtained to allow for different drill bits to be used - this way any metal rod (1/16 to ½ inch in diameter) can be used as a drill bit. Use of a drill bit is beneficial since neither the adapter nor the rotary hammer drill are be able to directly fit the grounding rod, and so in hard soil, a bit would be used to prepare a ½ inch hole that would make it easier to drive the ⅝ inch grounding rod. Furthermore, the ideal grounding rod height to be used is 8 ft - as reported by a grounding professional, thus making it difficult for one person to drive the rod into the ground, without an initial hole prepared. Pushing in a drill bit will not only prepare a good start for the grounding rod, but also reduce the height needed to be able to reach the rod. A 12 gauge solid copper grounding wire was purchased to connect the electric box, pole holder, and the grounding rod, making sure they all have contact with the same material.

Coating

The steel used for the flagpole support is highly susceptible to corrosion and rust. To avoid this and to prolong longevity and structural integrity of the base, paint coating and primer were applied. The paint used was zinc coating due to its ease in application, its effectiveness, its inexpensiveness and 2 year lifetime. The primer was applied directly on top of the coating and provided better adhesion for the coats and overall corrosion inhibition. Research and testing validated to us that the use of primer was an absolute necessity.

Flagpole Light

To allow for the use of the station in the night time, a flagpole light was purchased. Red White & Blue Solar Beacon 106755-RWB is a solar powered light that flashes red, white and blue and is fit on top of the pole, with a 1 mile visibility length. Its independent power system removes the need of driving electrical wires to the top of the 27 ft tall flagpole and is able to therefore conserve the power consumption and assist with the waterproofing efforts of the station.

Webpage

All station data is be displayed in a password protected WordPress site hosted by Bluehost. The latitude, longitude and number of jugs for each station are shown on an interactive map using the WP Maps plugin. This data as well as the data timestamp are included in an automated table for each station.

This information is calculated within the Arduino code. Each station then sends the data to a master Raspberry Pi via satellite, which then transmits the data to a Bluehost database in a csv file. These files can easily be automatically accessed by the WordPress site.

If a website user wishes to add a new station, a form is available at the bottom of the page which prompts the user for a station number and approximate latitude and longitude. This information is stored in a table associated with the form plugin that is easily accessible by the website administrator. This form allows the administrator to keep track of requested changes without compromising the integrity of the data.

During three days of full system tests, the data was sent and successfully received twice per day with accurate data appearing in the website table.

Prototype Tests

Test Overview: Base

The team tested the waterproof capabilities of the base and the flagpole stability with a full assembly.

Objectives

1. Ensure no water is in the compartments
2. Confirm that the 'load cell in a bag' design is waterproof
3. Make sure the USB port cover allows no water to enter
4. The flagpole can handle the wind forces against the flagpole support
5. The base does not tip over while heavy winds are subjected to flagpole
6. The roof of the base does not shear off during testing

Test scope and plan

In order to test the waterproofing capabilities, the team used the hose located on the 5th floor of CSI near the Bee Alliance area. In order to figure about the flow rate generated by the garden hose, the team had gotten an empty one gallon jug of water and timed how long it took to fill. The one gallon of water took a total of 9.3 seconds to fill up, having an average of 6.5 gallons per minute. To test the efficiency of the base, all the test needed to demonstrate was that exactly zero water enters the base compartments, the load cell casing, and the USB cover. If any amount of water breached any of the listed the components, the test would be considered a failure.

The flagpole support system was tested next. This test was to prove that the flagpole would not break by the distributed load acting on the pole during a heavy wind. The performed simulations done in Creo revealed that the pole should be able to handle approximately 2900 in-lbm/s^2, or equivalently, a 27 mph wind. Of course, the purchase of the flagpole guaranteed that it can handle upwards of 90 mph winds. The second aspect of flagpole testing was to make sure that during the large moment generated by the wind, the base remains grounded and does not tip over. The last aspect tested was to ensure that the plastic welded roof of the base would not shear off from the wind force.

Acceptance criteria

An acceptable result consists of all components being completely waterproof. Also, if the flagpole shows minimal plastic deformation, the base does not tip, and the roof does not show any signs of weakening.

Test Results and Evaluation

The base underwent strenuous waterproofing tests to ensure the security of the electronics compartment. The first part of waterproof testing included spraying the entire base with a high velocity hose to simulate a massive rain storm. The base was sprayed at several different angles to guarantee no water would enter. Anticipated weak points were targeted with the high velocity water and it was confirmed that no water got inside the compartments. After approximately 30 mins of on/off spraying conditions while checking the components every 10 minutes, the compartment remained dry. Not only did the compartments remain dry, but the USB cover also blocked all water from enter the port. During all of the testing, the load cell carrier was left in the water barrel compartment getting submerged by the large amounts of water collecting. Even with complete submersion, the load cell remained dry. Submersion of the load cell was an essential test because that is a real life situation that the system may be exposed to in the field if heavy rains continue for a long period of time. Figure 4 shows the testing apparatus used to test the water resistivity of the aforementioned components.

Figure 4: Top image is electronics compartment and USB cover, bottom is the electrolyte powder compartment, and load cell in within the base getting submerged.

The next phase of testing included leaving the base on the CSI 5th floor rooftop to be exposed to real rain conditions. After being on the roof for over a week, the base was exposed to a severe thunderstorm with heavy rain. Even through an above average amount of rain directed to the base, there was still no water inside the compartments. The load cell also remained dry and no water entered the USB cover. After extensive water testing, the base proved that it can withstand heavy rainfall and still have all essential components remain dry.

The only way to test the series of flagpole objectives was to set up the entire system and observe. When the real assembly of the flagpole support occured, a 36 inch support was used instead of the previous specification of 30 inches. The extra 6 inches acted as an additional security measure. Figure 5 shows the full assembly of the base.

Figure 5: Base with flagpole and flag pole support

Upon full system set up, there were heavy winds in the San Antonio area, as well as a storm. Winds of approximately 20 mph were acted on the pole for almost an entire 24 hours. Within this time period, the flagpole showed no sign of deformations, the base did not tip, and the roof showed no signs of weakening. The flagpole support kept the pole upright with ease, even when there were no water jugs in the barrel. It was deemed that guide-wires were not necessary in order to keep the flagpole upright. All tests were passed.

Test Overview: Phone Charger

The team tested the charger by measuring the current provided to phones as they charged.

Objectives

This test evaluated the phone charger and port to verify that:

1. The phone charger provided 5W of power at 5V, for a full charge cycle.
2. The phone charger worked without overheating.
3. The phone charger could charge multiple types of phones.

Test Scope and Plan

The charger port was tested in the lab with a Samsung Galaxy S7 phone and in the field with a Samsung Note 9 and iPhone XR. The test examined the immediate power provided to the first phone when it was plugged in. Later, the power, along with the heat produced by the charger was examined as the phone charged to full. This test assumed the user would use the charger port correctly and would charge their phone with a standard USB-A charging cable. Once the station was set up, the Samsung Galaxy Note 9 and iPhone XR were tested with station’s phone charger to ensure they charged.

This test required a voltmeter, ammeter, the phone charger and the station’s main battery. The voltmeter measured the voltage across the power supply to the phone charger, while the ammeter measured the current supplied to the phones. This test required basic circuit knowledge to compute the power supplied to the phone by the charger. Once it was confirmed the charger provided the correct power to the phone, its battery was completely drained and charged to full by the charger. During charging, the voltage and current were recorded every 5 minutes, along with the battery capacity The test continued until the phone had been completely charged. This data would confirm the charger can supply necessary power for a length of time and that the phone is actually charging. In the field, the two other phones were tested with the phone charger

Acceptance Criteria

The phone charger can be validated if it provides 5W of power at 5V for the start of the charge cycle and steadily increases the charge of the phone.

Test Results and Evaluation

Figure 6 shows the current provided by the charger to the phone and the battery capacity over time. Based on these graphs, the phone took about 2 hours to be charged from 0 to 100%. The current provided to the battery did drop off as the phone battery charge increased. However, this is most likely due to the phone's internal circuitry regulating the power it received, rather than the charger not provided sufficient power. This hypothesis is further supported by the battery charge over time, which shows the battery capacity grew linearly with time. Furthermore, tests in the field show the charger was capable of charging an iPhone XR and Samsung Galaxy Note 9. These results validate the charging capabilities of the charger and since the charger provided 5W of power to the phone for a majority of charging cycle it meets the acceptance criteria. However, these three phones represent a small sample of the phones which could be potentially charged with the charger and tests should be conducted on a wider range of phones, in particular those most likely to be used by migrants.

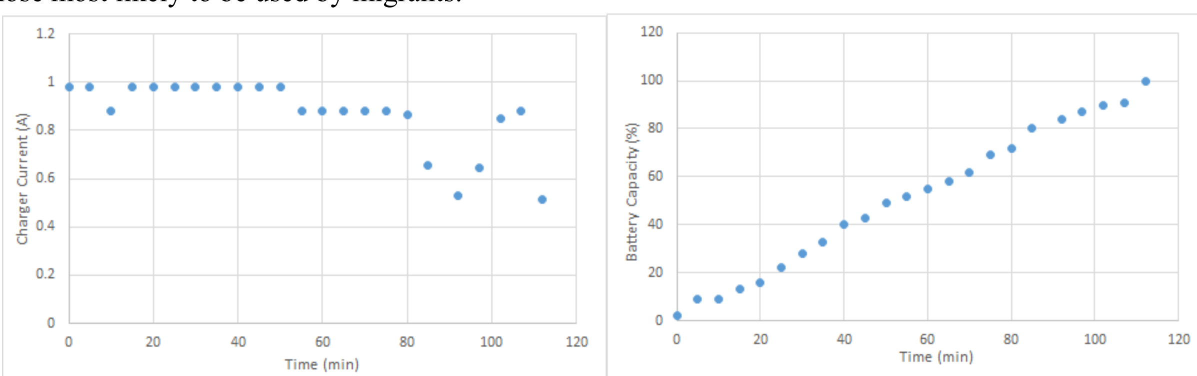

Figure 6: Current from charger (left) and battery capacity (right) over phone charging cycle.

Test Overview: Load Cells, Field Confirmation

The team conducted a drift test of the load cells in the field to evaluate their accuracy and precision when subjected to temperature variances.

Objectives

This test evaluated the self-contained load cells chosen after the initial testing done in the Fall and verified:

1. The station returned the correct number of water jugs.
2. The measurements the load cells produced did not drift over time in the field.
3. The measurements the load cells produce did not drift in response to temperature.

Test Scope and Plan

This test was conducted in a water station deployed in the field. The load cells were tested for 2 days and it was assumed the load cells were properly weatherproofed in the field, but otherwise no other assumptions were made.

The load cells were tested in the field with data saved to an SD card every minute. The load cells were placed on a plate underneath the water barrel, which was loaded with 15 water jugs. This test required basic programming skills to set up the microcontroller to relay the weight sensor measurements. To confirm the load cells work consistently, water weight were monitored over time as jugs were removed from the station. To confirm the weight sensors do not experience significant drift due to temperature, the temperature was saved as each weight sensor measurement was taken. A plot of the weight value vs. temperature was constructed to evaluate the drift,if any, due to temperature.

Acceptance Criteria

The load cells can be validated if they measure the correct number of jugs and produce readings which drift less than 2 lbs over time or in reaction to temperature.

Test Results and Evaluation

Figure 7 shows the weight vs. time plot and the temperature vs. time plot side-by-side. At all temperatures, there is no observable drift in weight reading as the temperature varies. There is a noticeable increase in the weight at the point where the station is 100 lbs and temperature remains constant at 20 degrees celsius. The temperature was held constant at this period so the variation cannot be attributed to the temperature difference. The station was not sending a status update or having a phone charged, and the weather was a clear sunny day so no influences of rain existed to provide extra weight. The station also recorded a constant weight of 55 lbs despite a quick temperature increase from 15 to 45 degrees celsius.

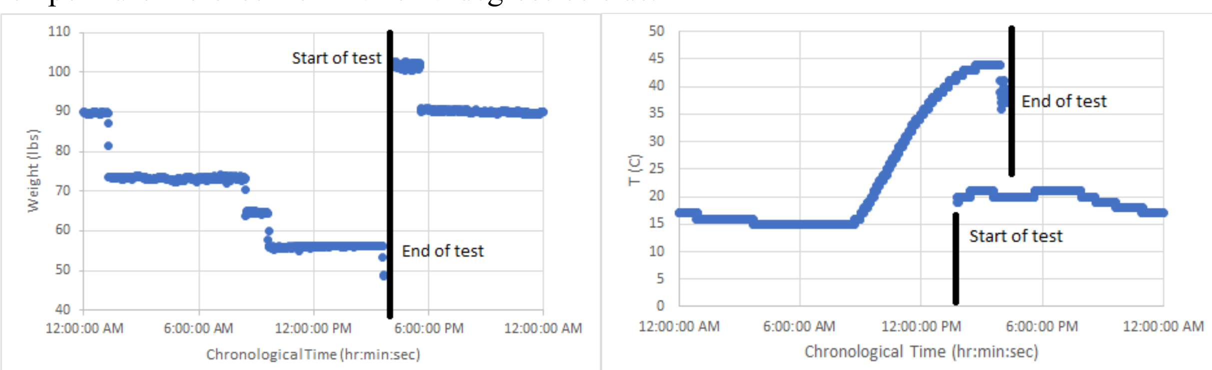

Figure 7: Weight of water in field station and ambient temperature over time.

Test Overview: Power Sensors, Sensor Accuracy

This test evaluated the voltage and current sensors ability to produce precise measurements within the expected range to be encountered in the field by calibrating them with known values and evaluating their consistency in response to field tests.

Objectives

This test evaluated the power sensors, which consisted of the ACS723 current and resistive voltage sensors. These were used to track the battery capacity of the station and ensured the solar panel was functioning correctly. This test sought to:

1. Calibrate the voltage and current sensors

Verify that:

2. The voltage sensor provided accurate battery voltage readings up to 14.7V
3. The current sensor provided accurate readings up to the maximum solar panel current, 3.1A

Test Scope and Plan

Calibration of the voltage and current sensors were limited to their operation limit, 0-15V for the voltage sensor and 0-3.1A for the current sensor. The limit on the voltage sensor represented the maximum possible voltage from the battery, while the limit on the current sensor represented the maximum current the panel could produce. Later, the measurements were during field testing to evaluate their precision.

These sensors were calibrated and tested using known voltages and currents which required a variable voltage source, voltmeter, the solar panel and an ammeter.. The voltage source was varied in 1 V increments from 0V to 9V, then in 0.25V increments from 9V to when the sensor saturated. To calibrate the current sensor, the angle of the solar panel was varied to produce current in 0.1A increments from 0A to 3.1A. The output of the two sensors were translated to calibration coefficients based on the ADC of the microcontroller. The accuracy of the sensors were evaluated by averaging the percent error between the measurement from the lab meters and the measurement from the station sensors. The measurements from these sensors was recorded every minute during a two day field test.

Acceptance Criteria

The operation of these sensors can be validated if the voltage sensor can measure battery voltages up to 14.7V within 3% of the true value and the current sensor can measure accurately currents up to 3.1A within 3% of the true value. These sensors must also be able to provide consistent readings of their phenomenon.

Test Results and Evaluation

The voltage and current sensors were shown to be reasonably linear during calibration, the graphs for which are shown in Figure F2. Meanwhile, the uncertainty associated with these sensors are summarized in Table 2. Both of these sensors had an average difference of less than 3% between the true and sensor values. Based on these results, these sensors should have performed well in the field. However, measurements in the field show mixed behavior from these sensors.

Table 2: Power Sensor Uncertainty

Sensor	Uncertainty (%)
Voltage	0.096
Current	2.816

The battery voltage and current from the solar panel over the course of 24 hours are given by the left and right side, respectively, of Figure 8. These graphs show that while the measurements from the current sensor were reasonably stable over the course of a sunny day, the voltage from the battery varied widely. The raw data, shown in blue, had a range of over 0.4V. The results from a 10-element averaging window filter, shown in orange, were not very stable either. This indicates that this particular sensor is fairly noisy and its measurements may vary widely from minute to minute. This is an issue, especially when one considers the operating voltage of the battery from 90% to 10% capacity only ranges from around 13.1V to 12.5V, as shown in Figure F3. As a result, while the ACS723 sensor is acceptable to measure current from the panel, more work is needed to create a reliable voltage sensor. Since this sensor is a resistive voltage divider, this issue could be fixed by selecting smaller resistors with higher tolerances and lower temperature dependance. Only when this problem is addressed can STHRC use the voltage data to alert them to a station with failing power.

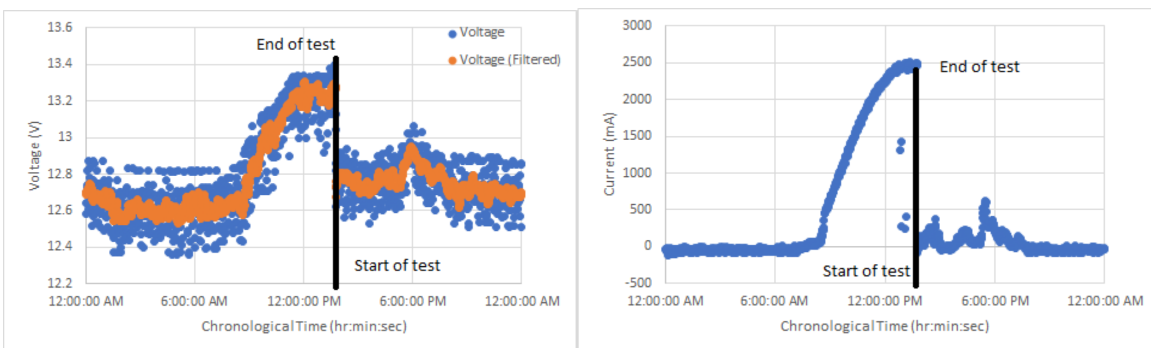

Figure 8: Battery voltage (left) and solar panel current (right) over 24 hr station deployment (sunny conditions). Start of 24 hr period is at approximately 1:30PM and ends at 1:30PM the following day.

Test Overview: Battery, Full Load Test

The team ran the entire power system at maximum loading to confirm the battery sustains operation without triggering its internal safety measures and shutting off power.

Objectives

This test evaluated the battery and its ability to provide power at the maximum possible electrical load and verified that:

1. Without shutting off, the battery provided power to all load devices: phone charger, load cells, communication module and power sensors while they were being used simultaneously.

And measured:

2. The current draw from the battery at this maximum load.

Test Scope and Plan

This test was conducted at maximum loading, with the load cells, solar controller, phone charger, microcontroller, power sensors and communication module active. However, this test assumed each of the components was working properly. This test was not meant to simulate extreme conditions such as accidental short circuits and/or damaged components.

Validating the battery's operation at full load required all of the various components and an ammeter. Each component was set up to draw power simultaneously and the team used a multimeter to measure the current draw from the battery. This draw was compared to the battery's data sheet, to confirm it operated within a safe range.

Acceptance criteria

The battery must be able to provide the necessary current at full load without being damaged and without shutting off.

Test Results and Analysis

The LiFePO4 battery was able to provide enough power to all of the components, even when they were all activated at once. Additionally, the Arduino Mega did not reset at any point, indicating the maximum current draw from the 5V power pin of this microcontroller was not reached. The current draw from the battery for various modes of operation are shown in Table 3. The maximum draw, 1.2A was at startup and is consistent with the fact that the Rockblock begins charging its supercapacitor as soon as power is applied. Worst case operation will most likely be limited to 650mA, since the Rockblock is not sending data continuously. This means the station would require at maximum 15.6Ah per day. At worst case, assuming the charger is being used all day, the station would be able to power itself for over day, even if there was no sun. If the charger is only used two hours a day, as the original estimates of battery capacity assumed, the station will require 5.46Ahr of charge, less than the original computation of 5.93A, given in Table 3.

Table 3: Current Draw From 12V Battery For Electrical System Operating States

Operating State	Current Draw (mA)
Station Startup w/ Charging Phone	1200
Sending Data w/ Charging Phone	790
Normal Operation w/ Charging Phone	650
Normal Operation	143

Test Overview: Solar Panel, Charging Test

The power systems team configured the station to automatically conduct measurements of the current and voltage output from the solar panel and battery during outside operation, ensuring the station power supply is self-sustaining.

Objectives

This test evaluated the 12V, 50W solar panel and the 12V LiFePO4 solar controller and verified:

1. The solar panel charged the battery

2. The station operated for two days without shutting down.

Test Scope and Plan

These objectives were tested once the full station was assembled and placed in a spot which accurately simulated the environmental conditions of where STHRC will set these stations up. The initial tests was not meant to simulate extreme conditions, such as incremental weather, though light rain was experienced during testing.

Validating the solar panel required the completed electronics box, which included the power sensors. The power sensors provided continuous monitoring of the battery and solar panel as the station was left outside for two days. This data was recorded along with the barrel weight, temperature and humidity every minute and was saved to an SD card. The charging system can be validated by examining the current from the panel and the battery voltage, since these data will confirm the panel is generating power and the battery is being charged.

Acceptance Criteria

During the course of independent operation, the battery cannot drop below 10% capacity. The solar panel must be of sufficient size that it can keep the battery charge at a reasonable level continuously, even if there is incremental weather.

Test Results and Analysis

The battery voltage over the course of a day is shown in Figure 9. Although this voltage sensor was fairly noisy, there is a general increase in voltage which correlates with the steady increase in current from the solar panel. Additionally, battery capacity did not drop below 10%, since if it had the battery protection module would shut the battery down, along with the rest of the station. Since no gaps in data can be seen in Figure 9, or in Figure F4 and F5, it can be concluded the battery had sufficient charge to operate normally. This particular 24 hour period of testing was performed on a sunny day; Figure 9 shows the solar panel was producing 2.5A around noon, close to its maximum current (3.1A). Based on these results, and those shown in Figure F4 and F5, which were gathered on a cloudy day, the station is shown to be self-sustaining since the battery charge did not drop below 10% and the solar panel kept the battery reasonably charged during the day. Based on these results, the solar panel and battery of the station were successful in keeping the station powered. However, it will be important to monitor the station over longer periods of time. Therefore, the SD data logging system will be kept with system when it is deployed by STHRC, in order to better understand how this station's power system operates over time and whether the solar panel and battery sizings were appropriate for the needs of this station.

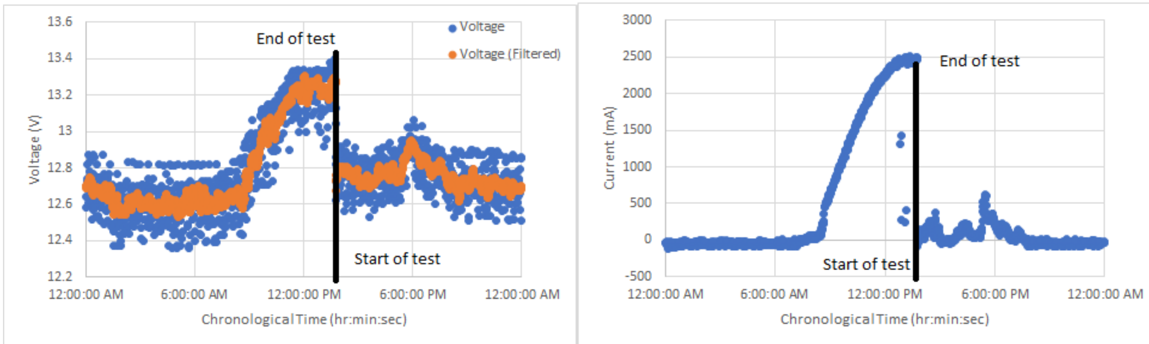

Figure 9: Battery voltage (left) and solar panel current (right) over 24 hr station deployment (sunny conditions). Start of 24 hr period is at approximately 1:30PM and ends at 1:30PM the following day.

Test Overview: Adafruit Real Time Clock

This test confirmed the communications system was able to accurately read the current time using a real time clock (RTC) in order to send out information at specific times.

Objectives

This test evaluated that the communication system could:

1. Connect the real time clock to work with the Arduino Mega
2. Calibrate the real time clock
3. Verify that no drift occurs overtime between the actual time and the RTC displayed time

Test Scope and Plan

The first objective was completed through the Adafruit RTC manuals on how to set up the device. The second objective, calibrating the clock, was tested once the RTC was connected to the Arduino. The scope of the third objective compared the time being displayed by the RTC and the actual time being displayed on the computer.

To test the RTC, a program that allows the real time clock to read the current time and be initialized was uploaded to it. This code is shown in Figure E1 of Appendix E. This code displays the current date and time, and these values were compared to the current date in time to evaluate the accuracy. Once initialized, this program did not have to download the time again, unless it lost power. To test for possible drift, the date and time were be read for periodically throughout the day, and the differences in the displayed versus actual time were compared. This confirmed whether the real time clock is accurate and if any drift occurs over time.

Acceptance Criteria

The operation of the real time clock must be within 5 minutes of the actual time. Drift is acceptable, as long as the values stay within the given 5 minute range.

Test Results and Evaluation

The accuracy of the RTC was evaluated for weeks as multiple tests involved capturing the current time. The RTC constantly was 3 seconds behind the actual time, and this difference did not seem to drift over time. This validated our RTC as an accurate component to use in our communication system. The RTC has an independent power source from the rest of the communication system, so it maintains its accuracy if the system has to be restarted.

Improvements for future communication designs in relation to using an RTC module would be to use a more precise real time clock. Our suggestion would be to use an Adafruit DS3231 Precision RTC breakout board. This specific model compensates for slight internal crystal timing drift that occurs in other RTCs when exposed to drastic temperature variations. The DS3231 has a temperature sensor right next to the integrated timing crystal that will add or subtract timing cycle based on temperature drift [6].

Test overview: Humidity and Temperature Sensor

This test confirmed the communications system was able to accurately read the current humidity and temperature using a DHT11 humidity and temperature sensor.

Objectives

This test evaluated that the communication system could:

1. Connect the DHT11 sensor to work with the Arduino Mega
2. Correctly accurately read both the humidity and temperature

Test Scope and Plan

The first objective was completed through the CircuitBasics manual on how to setup the device. The second objective, obtaining accurate readings, was tested once the sensor was connected to the Arduino.

To test the sensor, code was written to enable the sensors to capture data, shown Figure Ew of Appendix E.

Acceptance Criteria

The DHT11 sensor came with given rating for device accuracy, and the device was defined as acceptable as long as its reading stayed within this operating and accuracy ranges:

1. Temperature range = 0-50 (C)
2. Temperature Accuracy = +/- 2 (C)
3. Humidity Range = 20-90% RH
4. Humidity Accuracy = +/- 5% RH

Test Results and Evaluation

Using the code from Figure E2, the accuracy of the humidity and temperature sensor was evaluated. The results were compared to the known temperature and humidity values of the environment, and they met the acceptable criteria. However, when the sensor was sampled too frequency, in intervals less than 5 seconds, some of the values would miss these criteria. As long as the system does not sample this fast, the sensors maintained accurate readings.

Test overview: Complete System Transmission Test with RockBlock

This test evaluated the transmission capabilities of the RockBlock satellite module and the Arduino Mega Microcontroller. It tested that the module transmits the desired data independently and to the correct email addresses, and that this data is correctly uploaded to the BlueHost website.

Objectives

This test verified that:

1. The RockBlock can connect to the satellites properly and hold connectivity.
2. The data transmitted is correctly sent and uploaded to the set receiving email addresses.
3. The format for sending the data is compatible through transmission.
4. The RockBlock data sent from the module is uploaded correctly to the Bluehost website.
5. The Python code running on the Raspberry Pi extracts the satellite data and uploads it onto the website correctly.

Test scope and plan

A controller code was written in order to read the water weight and location at each of the deployed water stations. The code was written to meet the following criteria:

1. Read weight using Analog ports and GPS location using Satellite module
2. Keep track of real time using Real Time Clock (RTC) module
3. Send weight data array [0:47] at standard intervals (currently set to 7:00am and 7:00pm)
4. Read station parameters (battery voltage, solar panel current to battery)
5. Save station parameters to a removable SD card (acting like EEPROM)

The data collected through running this code was designed to automatically send in an email, shown in Figure E3 of Appendix E. In every email all of the data sent from the RockBlock is recorded in the body of the email. This allowed us to develop a program that could automatically extract this data and upload it onto the Bluehost website using the FTP protocol.

After the controller code finishes executing and an email is sent, Python code was written to decode these emails. The data extracted from the email is saved in hexadecimal format. The Python script converts this to decimal values. The data output after executing the Python script is shown in Figure E4 of Appendix E. This script was also written to upload the decoded data to the BlueHost website, as shown in Figure E5.

Acceptance criteria

The performance of the RockBlock module will be validated if it can send the test program, generated by the Iridium RockBlock website, correctly and to the designated receiving emails. It will also only be validated if the first part of this test is accomplished and if it can send the array generated from the Arduino Mega. The performance of this test will also be evaluated by the performance of the communication system to transmit the data from the station and then uploaded onto the Bluehost website. It will be acceptable if the data is uploaded correctly and continuously based on the transmission time interval.

Test Results and Evaluation

This full system test provided concrete evidence that the whole communication system was working properly. The email shown in Figure E3 verified that the RockBlock was able to transmit the correct data from the station to the satellite and then the satellite was able to correctly upload the data in an email. We then ran a Python script to decode the transmitted data presented in the email. The results of the decoded data are displayed in Figure E4, showing that the average number of water jugs was ~14, battery voltage was 12.8 V, and the station was not generating any panel current. These values were further verified by comparing them to the data saved on the SD card at the same relative timestamp this data was sent from the station. The last stage of the communication procedure was verified to have been working correctly by viewing the results posted to the Bluehost web page. The Bluehost table results shown in Figure E5 match the decoded values presented by the Python script. This verified that the process for uploading the transmitted data from the water station to the website was working as designed.

One improvement we suggest for future iterations of this communication system is to have an optimized data transmission algorithm running on the Arduino. Currently we perform ~20 transmission attempts and wait 2 minutes between attempts. If the signal quality read by the RockBlock does not reach above 3 (scale from 0-5) then the program will not send data at this time. This was done to improve the program's original busy-wait transmission protocol.

The second improvement we suggest is a redesign of the data transmission process outlined in the communication and control section. It would be ideal if the RockBlock could automatically upload the data to the Bluehost website, bypassing the need to run the Python email processing code on the Raspberry Pi.

Conclusion

The system proved to be successful on most accounts to meet the project objectives. The base of the station was successful in waterproofing, stability, and flagpole support. The power system was self sustaining for three days when testing the full system. Weight sensors and current sensors showed steady results. However, the voltage sensors showed significant variations. The communications system was capable of accurately transmitting the weight, location, time and temperature of the tested station.

It is recommended that the voltage reader should be improved to ensure reliability and accuracy. The Adafruit DS3231 RTC could potentially improve time drifts, but more testing would be required. The satellite data sensing could be optimized using an algorithm. With further testing and research, it is recommended that all stations communicate to the website without the need for a master Raspberry Pi.

Team Adviser Review		
Project Name: Emergency Water Station Project		
I have reviewed this project deliverable, and I have submitted comments and feedback to the Project Team.		
	Date Received	**Date Returned with Comments**
Team Adviser: Mehran Aminian Farzan Aminian F. Aminian	4/30/2019	4/30/2019

Appendix A: Setup, Operating and Safety Procedures

Instructions for Setting Up the Emergency Water Station

To Install Grounding Rod:

1. Position 8ft grounding rod over station site. Station site should be as flat as possible and clear of large debris
2. Using a fence post drive, drive the grounding rod into the ground, leaving approximately 1.5” of rod above the ground
3. Remove 0.25” of insulation from the 12 gauge grounding. Attach metal collar to top of grounding rod and place grounding wire between the collar and the grounding rod. Tighten the collar until there is a tight connection between the wire and the grounding rod.

To Install Base:

4. Place flagpole support on flat portion at the intended station location, as close as possible to the grounding rod.
5. Position base over top of flagpole support, ensuring long foot of the support is underneath the barrel compartment but is not directly underneath any of the drainage holes. Adjust ground underneath the feet on the corners of the base until the base has less than or equal to 1 degree of tilt (exact number does not need to be precise, but station will be more stable the more even the base is positioned).
6. Thread grounding wire through hole in bottom of base.

To Install Flagpole

7. Insert 3D printed part into flag support, ensuring hole is aligned with the hole in the base of the flag support. This piece will elevate flagpole when it is inserted into the support.
8. Attach solar panel to lowest section of flagpole, positioning it so that the hole in the flagpole is as close as possible to the solar panel wires.
9. Attach light to top of flagpole.
10. Insert flagpole into flagpole support. The second section of the pole will likely fall through the bottom of the pole, but will be stopped by the 3D printed insert. Align the lowest section of the pole so that the locking mechanism for the second section is within the lowest section (bottoms of the two sections are flush). Then, reaching through the hole in the bottom of the flagpole support, push the second section of the pole up using your fingers. Have another set of hands ready to grab the second section of the pole when it is elevated above the lowest section.
11. Erect the flagpole starting with the top section. Lock the section in by rotating counter clockwise and then fastening down into the locking mechanism. Carefully release the upper section to ensure that it is locked in securely. Do the same with the other sections until the pole is fully extended.

To Set Up Electronics:

12. Thread grounding wire through the small holes closest to the flagpole support hole in the bottom and top of the base. Attach end of grounding wire to flagpole, just above where the flagpole support ends.
13. Connect MC4 connectors from solar panel. The red wire should be connected to the MC4 connector from the solar panel marked with the “+”, while the black wire will be connected to the MC4 connector from the solar panel marked with the “-”. Drop wires through flagpole.
14. Using one’s fingers, reach into the flagpole support and pull out the wire ends. **Important Note:** Do not connect wires until battery has been connected.
15. Unlock and open junction box drawer,

16. Unscrew and remove lid of junction box
17. Ensure area is clear of wires and attach battery to velcro in junction box.
18. Connect the battery's barrel power connector to the barrel connector from the solar charge controller.
19. Re-attach junction box lid with screws and lock drawer.
20. Connect MC4 connectors located at the base of the flagpole.
21. Place barrel on top of weight sensors, being careful so the drainage hole in the barrel aligns with the drainage hole in the base. **Important Note:** Do not smash wires going to weight sensors.
22. Load the barrel with jugs of water using the shelves. Evenly distribute the jugs on each level.
23. If possible, confirm station has sent email (station is set to send information on station immediately when electronics are powered on)
24. To prepare station for transport, execute steps 1-23 in reverse.

Instructions for Operating the Emergency Water Station (For STHRC Volunteers)

To Replace Battery

1. Remove water jugs and shelves from water barrel. Then remove barrel.
2. Disconnect MC4 connectors located at the base of the flagpole.
3. Unlock and open junction box drawer,
4. Unscrew and remove lid of junction box
5. Disconnect the battery's barrel power connector.
6. Remove battery from the junction box.
7. Ensure area is clear of wires and attach fresh battery to velcro in junction box.
8. Connect the battery's barrel power connector to the barrel connector from the solar charge controller.
9. Re-attach junction box lid with screws.
10. Lock drawer.

Instructions for Accessing Each Satellite Modem (For Future Senior Designers)

1. Go to the login page for the RockBlock service https://rockblock.rock7.com/Operations
2. Login using username: faminian@trinity.edu and password: Water_Station18
3. Once inside the RockBlock website one can view the available transmission credits, how many modems are activated, and what email addresses are receiving data from the different devices. Each time the modem transmits data it uses one credit. A single credit can support 80 bytes of data. If there needs to be more credits purchased this is the website where this action can be performed.

Instructions for Logging onto Python (For Future Senior Designers)

1. Open Putty
2. Paste IP address specific for Pi module = 131.194.113.248
3. Log in to Pi using: username = pi
   Password = Water!Stations18
4. Use command $nano Phase2NC.py
   a. Save using Ctrl+O and exit with CTRL+X

5. Start running the python script using $sudo.sh launcher.sh
6. To exit the current running screen without closing down the program, type Ctrl+a,d (2 button presses: type a followed by typing d)
7. Type $screen -x <NC> to review the screen and make sure it is running
    a. Repeat step 6 to exit the screen

Instructions for Accessing Code on Visual Studio (For Future Senior Designers)

For access to the full code along with more details on how to run the scripts, this is a great resource

1. Go to https://trinitywaterstations.visualstudio.com/TrinityWaterStations
2. Log in using: username = trinitywaterstations@gmail.com
   Password = Water!Stations18
3. Click on "Repos" on the left, select Phase II folder, then select the Deployed Stations folder

General Guidelines for Safely Setting Up and Operating Emergency Water Station

1. Do not operate on water station electrical system unless power is off. To remove power, disconnect the MC4 connectors at the base of the flagpole and disconnect the main battery. **In that order.**
2. Do not have solar panel connected to electrical system without the main battery also being connected.
3. Do not set up station in incremental weather.
4. Do not apply more than 300lbs to the weight sensors.

Appendix B: Creo Drawings

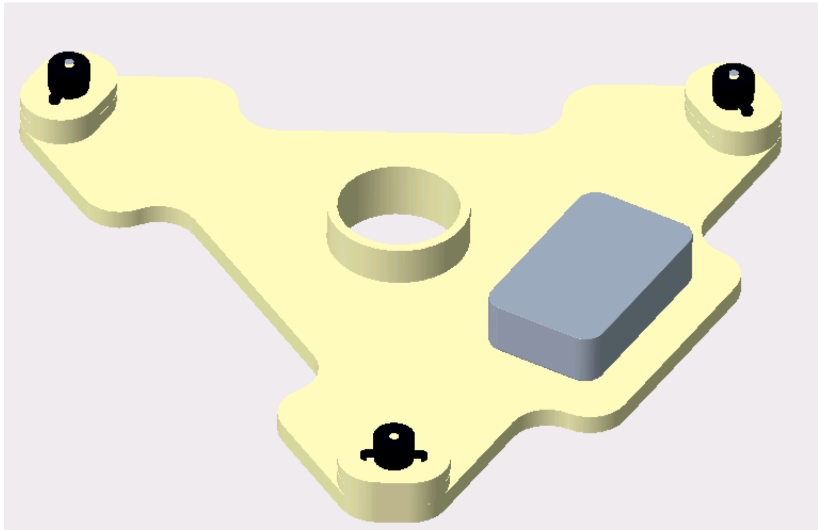

Figure B1: Design for self-contained weight plate.

Figure B2: Design for mounting plate for electronics.

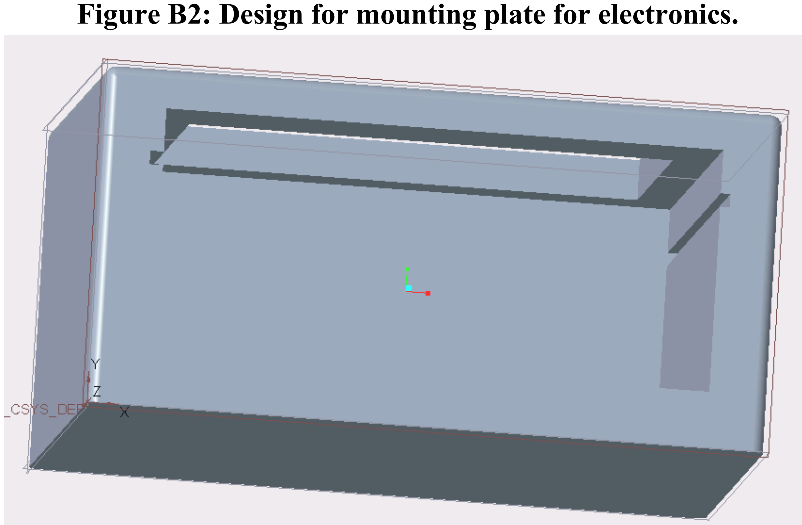

Figure B3: 3D printed mounting block for Rockblock satellite module

Appendix C: Electrical Schematics, Board Layouts and Photos

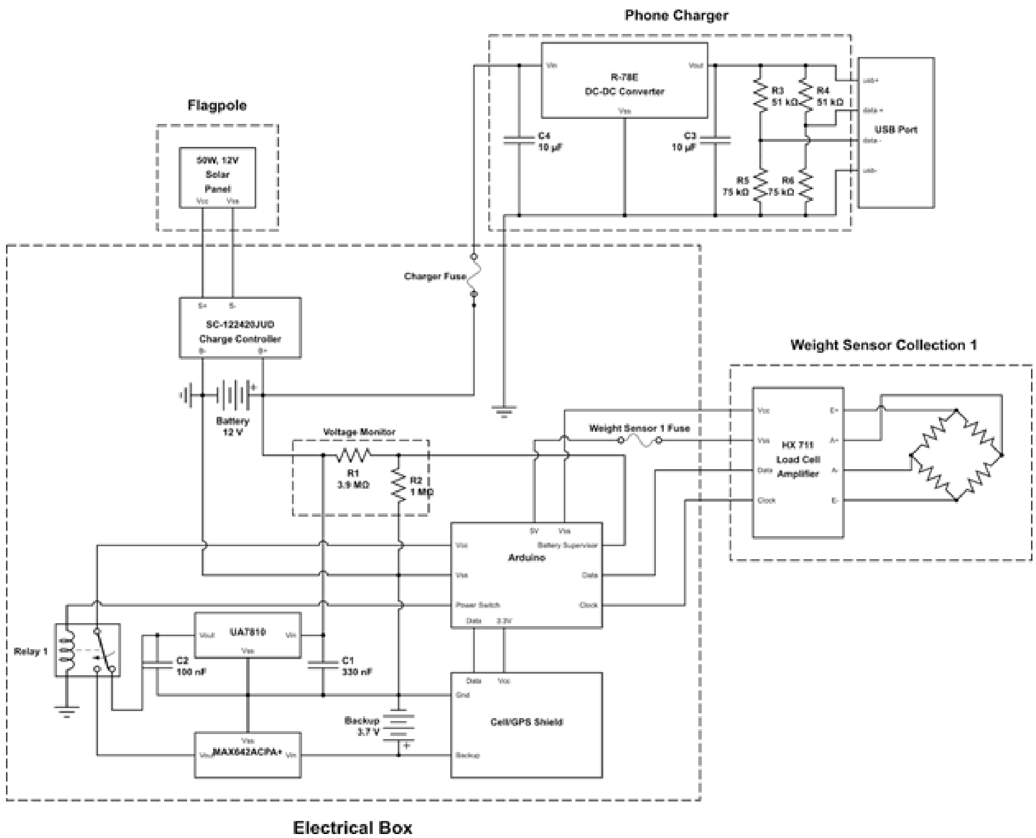

Figure C1: Emergency water station electrical schematic, circa. Dec 2018.

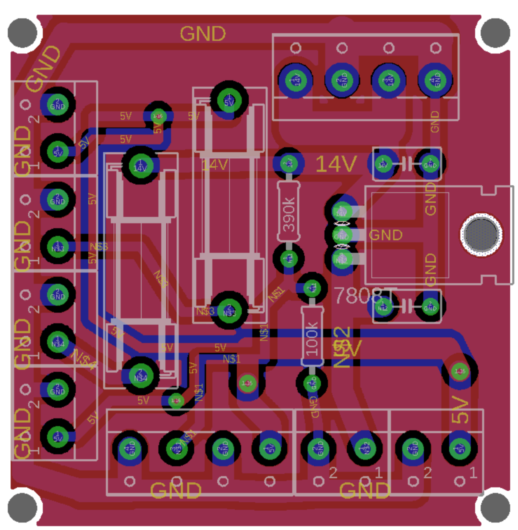

Figure C2: Eagle layout, power distribution board.

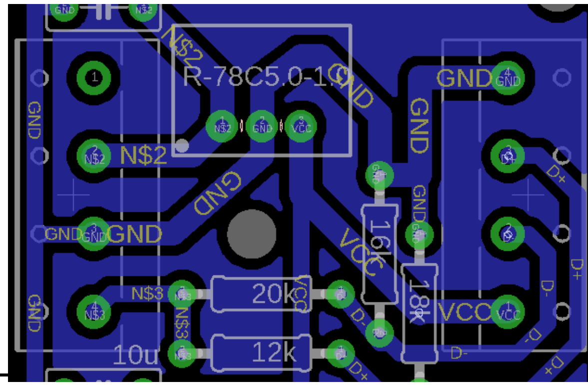

Figure C3: Eagle layout, phone charging board.

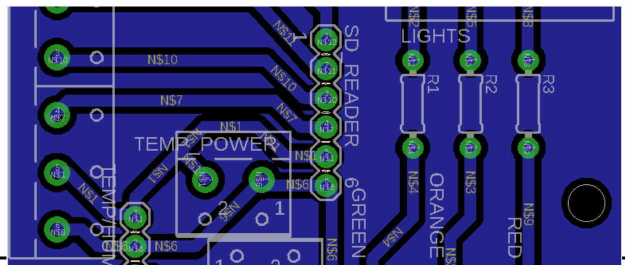

Figure C4: Eagle layout, debugging board.

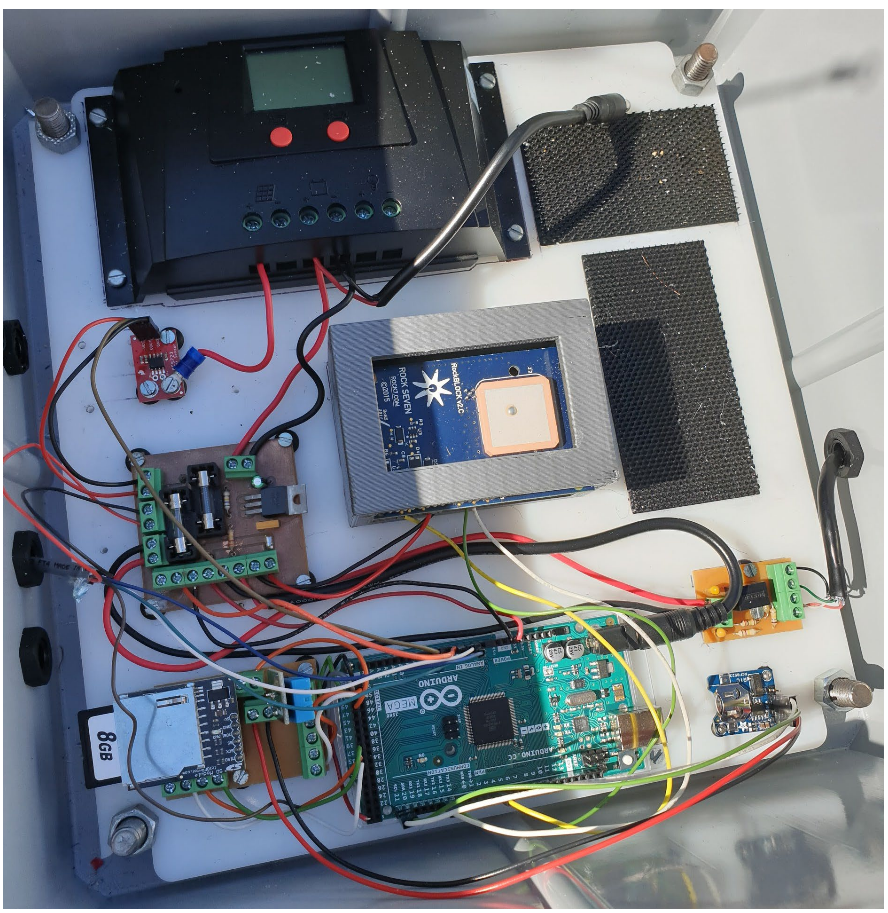

Figure C5: Photo of implemented mounting plate for electrical box.

Appendix D: Controller Constants, Flowchart and Program

Table D1: Arduino Mega Pinout

Type	Name	Pin
Analog	Weight Sensor 1	55 (A1)
	Weight Sensor 2	56 (A2)
	Weight Sensor 3	57 (A3)
	Battery Monitor	58 (A4)
	Solar Panel Current Monitor	59 (A5)
Digital	RockBlock Sleep	4
	Red Light	35
	Yellow Light	37
	Green Light	39
	Temperature/Humidity Signal	49
Communication	RockBlock Data (TX)	14
	RockBlock Data (RX)	15
	RTC Clock (SDA)	20
	RTC Clock (SCL)	21
	SD Reader (MISO)	50
	SD Reader (MOSI)	51
	SD Reader (SCK)	52
	SD Reader (SS)	53

Table D2: Arduino Constants and Calibration Coefficients

TAG	VALUE	Notes
FC2231_ADC_TO_LBS_SLOPE	0.37	slope for ADC to lbs calibration curve
FC2231_ADC_TO_LBS_INTCPT	-41.5	y-intercept for ADC to lbs calibration curve
I_SNSR_ADC_TO_A_SLOPE	12.299	slope for ADC to milliamps calibration curve
I_SNSR_ADC_TO_A_INTCPT	-6290	y-intercept for ADC to milliamps calibration curve
V_SNSR_ADC_TO_V_SLOPE	0.0131	slope for ADC to volts calibration curve
V_SNSR_ADC_TO_V_INTCPT	0.047	y-intercept for ADC to volts calibration curve
BARREL_WGHT	36.4	weight of water barrell and shelves (lbs)

Table D3: Arduino Alarm Triggers and Resets.

ALARMS	VALUE	Notes
LOW_WGHT_TRIG	60	low weight trigger (lbs)
LOW_WGHT_RST	110	reset for the low weight alarm (lbs)
HIGH_WGHT_TRIG	350	high weight trigger (lbs)
HIGH_WGHT_RST	300	reset for the high weight alarm (lbs)
HIGH_I_TRIG	4000	high current trigger (mA)
HIGH_I_RST	3200	reset for the high current alarm (mA)
LOW_V_TRIG*	12.45	low voltage trigger (V)
LOW_V_RST*	12.71	voltage required to reset the low voltage state (V)
HIGH_V_TRIG*	15.2	high voltage trigger (V)
HIGH_V_RST*	14.7	reset for the low voltage alarm trigger (V)

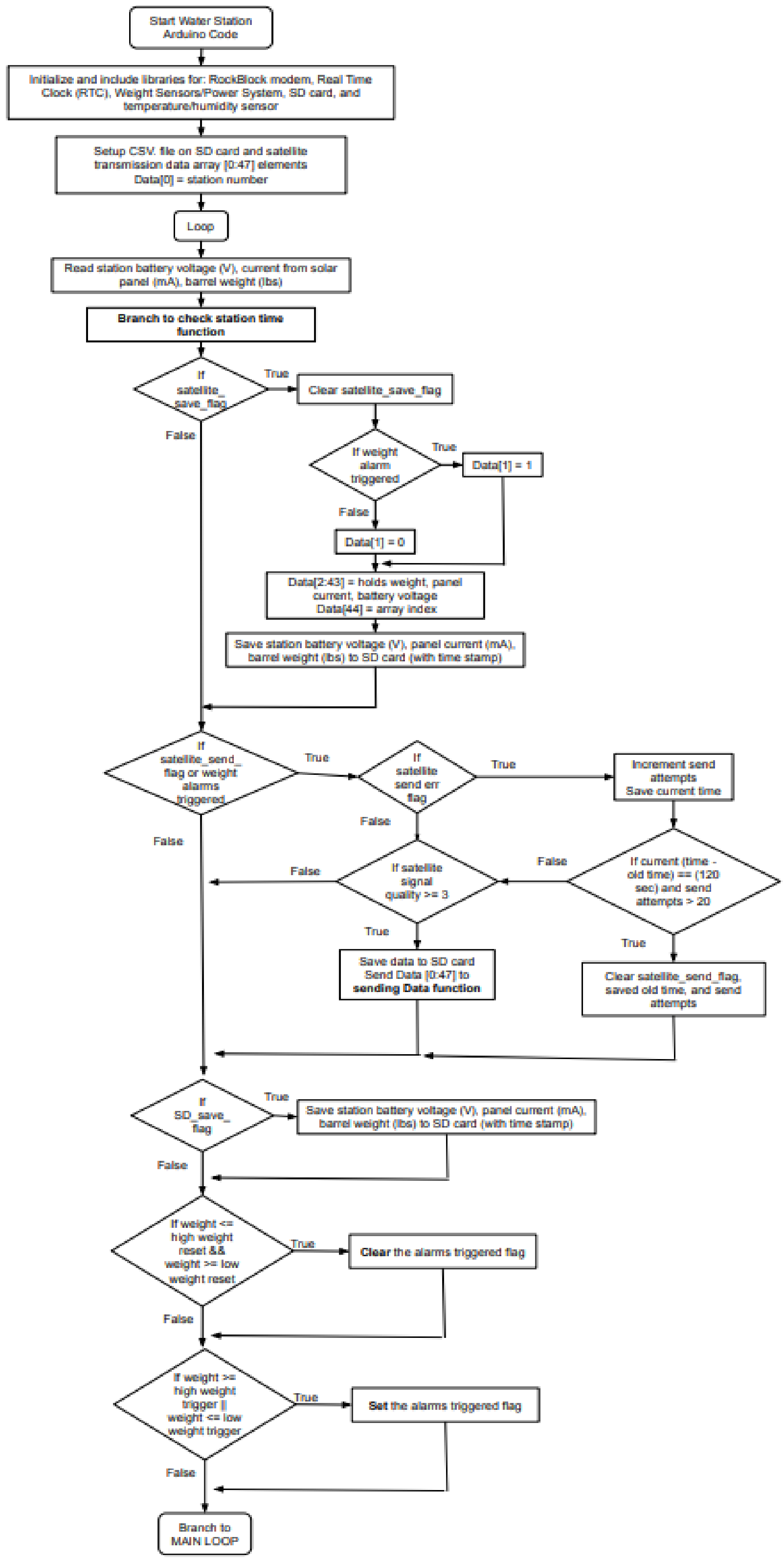

Figure D1. This is the flowchart for the main program running on the Arduino Mega within the water station.

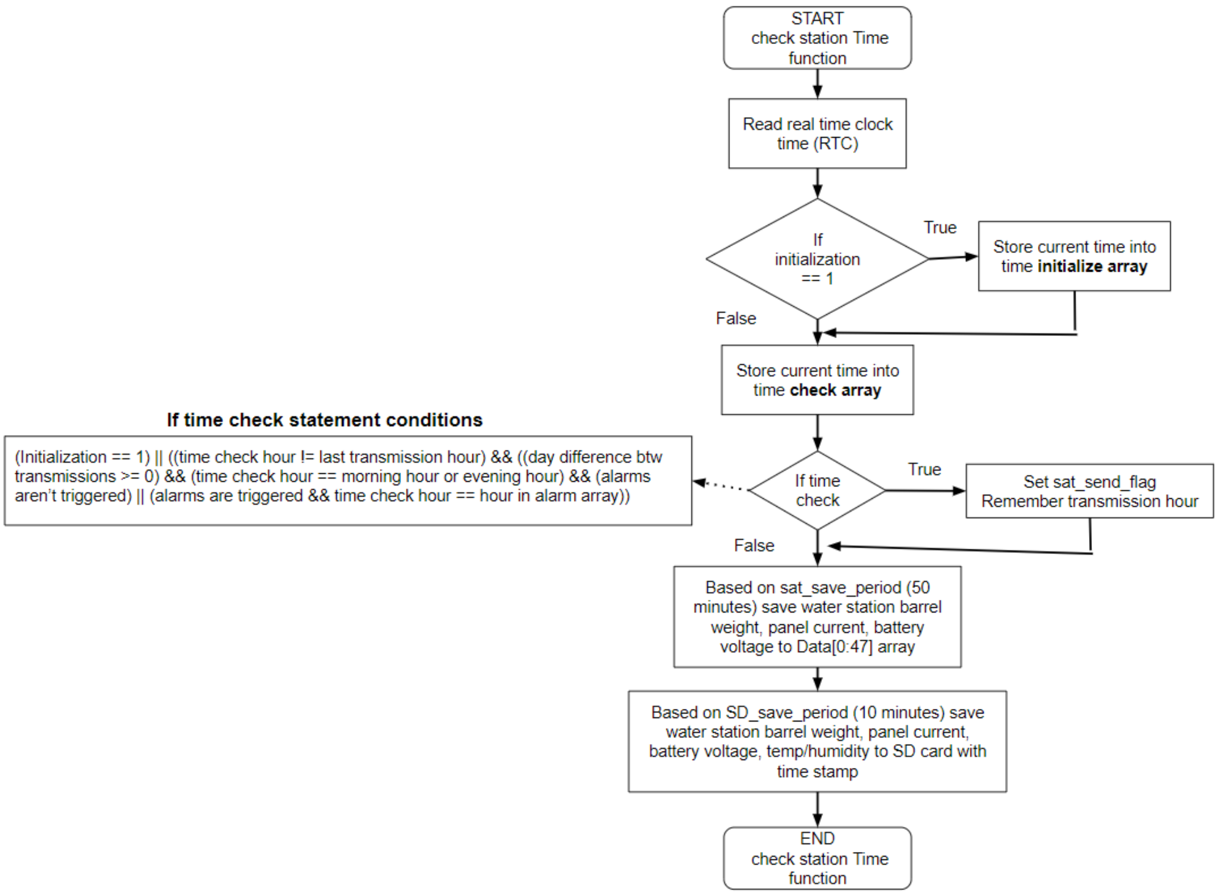

Figure D2. This is the flowchart for the checkTime function within the main program on the Arduino Mega. Its purpose is to set flags based on real time when the station needs to send data over satellite, save data to the transmission array, and save data to the SD card.

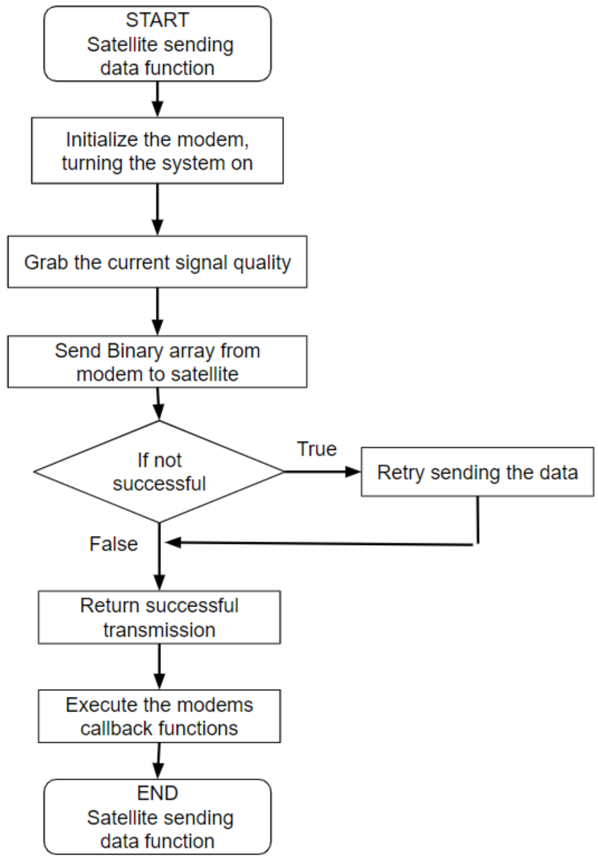

Figure D3. This flowchart represents the RockBlock function within the main program on the Arduino Mega that sends the data array [0:47] to the satellite.

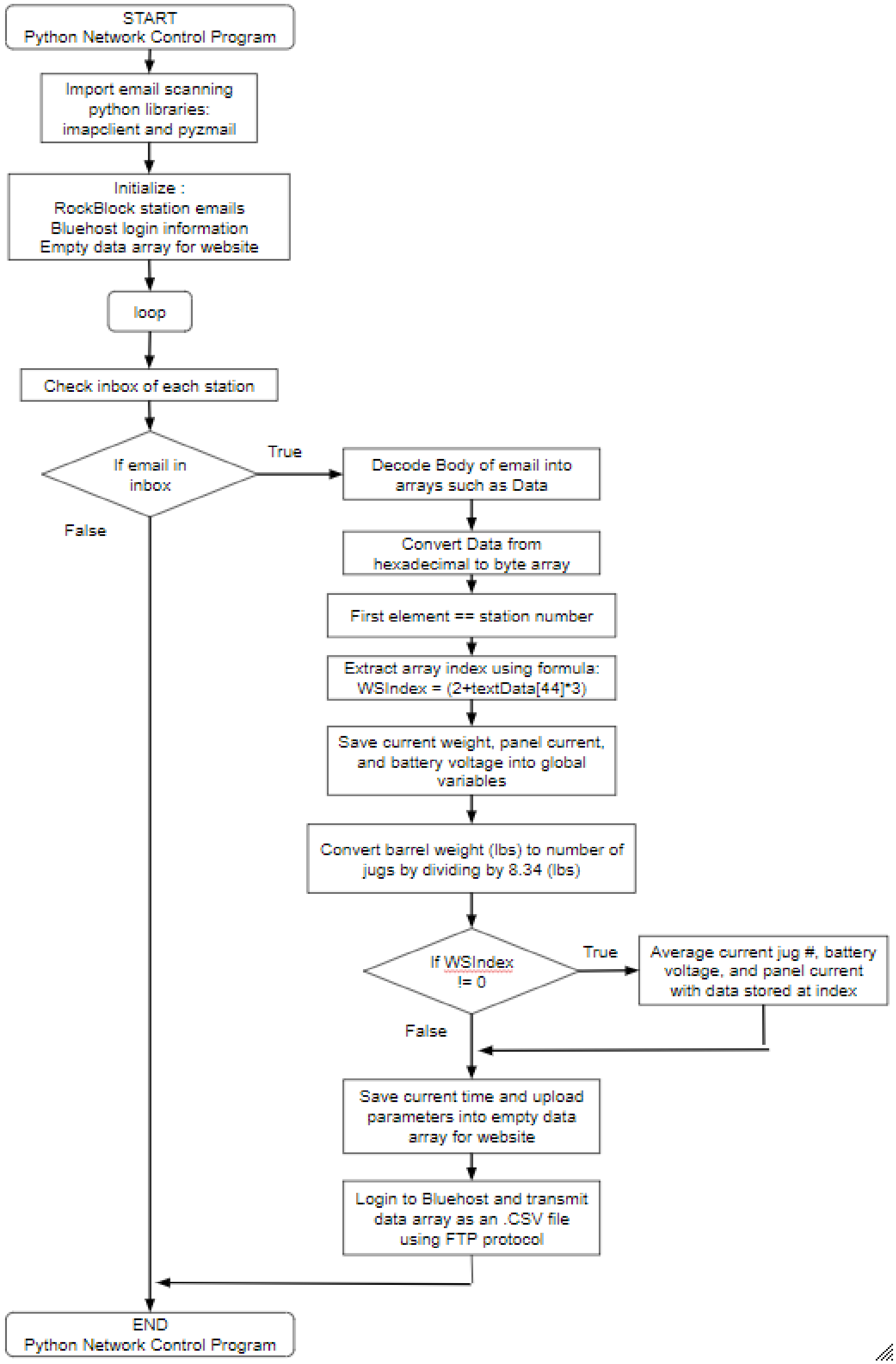

Figure D4. This flowchart represents the python network code running on the Raspberry Pi. It scans and processes the RockBlock data sent from each water station and then loads it onto the Bluehost website.

Appendix E: Communications Testing, Additional Figures

RTC_test §

```
/*
 * This is a RTC PCF8523 test code. It displays the current data and time along with
 * the computers initial reference time. Use this code to initilize the RTC module by
 * uncommenting (1) setting the date and time on the computer.
 * The time stamp is ~5-6 seconds off.
 *
 * Written by Caroline Kutach and Sean Farrell
 */

#include <Wire.h>
#include "RTClib.h"

RTC_PCF8523 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

void setup() {

  Wire.begin();
  pinMode(22, OUTPUT);
  digitalWrite(22, LOW);
  // put your setup code here, to run once:
  while (!Serial); // for Leonardo/Micro/Zero

  Serial.begin(9600);

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }
 //rtc.begin();
  //******************************
  //Only uncomment (1) when you want to intialize the new RTC module
  //after the first intilization MUST RECOMEMNT (1).
  //rtc.adjust(DateTime((__DATE__),(__TIME__)));  //(1)
  //******************************

  if (! rtc.initialized()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    //rtc.adjust(DateTime((__DATE__),(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    //rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
     // following line sets the RTC to the date & time this sketch was compiled
  }
}
```

```
void loop() {
  DateTime now = rtc.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" (");
    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
    Serial.print(") ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.print(" since midnight 1/1/1970 = ");
    Serial.print(now.unixtime());
    Serial.print("s = ");
    Serial.print(now.unixtime() / 86400L);
    Serial.println("d");

    Serial.println();
    delay(3000);
}
```

Figure E1: Real time clock test code

temp_humidity_sensor §

```
/*
 * This program is used to test the DHT11 temperature and humidity sensor.
 * The pin connections are (-) -> GND, (+) -> +5V, and (S) -> digital data
 * pin (49) defined as DHT11_PIN.
 *
 * Created by Caroline Kutach and Sean Farrell
 * Last Modified: 4/6/19
 */
 // Below is the ratings for the device accuracy:
 /*
  * 20 meter signal transmission
  * Power supply voltage = 5V
  * Temperature range = 0-50 (C)
  * Temperature Accuracy = +/- 2 (C)
  * Humidity Range = 20-90% RH
  * Humidity Accuracy = +/- 5% RH
  */
#include <dht.h>

dht DHT;

#define DHT11_PIN 49
float tempFahrenheit = 0.0;
float tempCelcsius = 0.0;
float humidity = 0.0;
void setup(){
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Checking temperature and humidity now");
  int chk = DHT.read11(DHT11_PIN);
  Serial.println((String)"Check "+chk);
  tempCelcsius = DHT.temperature;
  tempFahrenheit = (tempCelcsius*(1.8))+32;
  Serial.println(tempFahrenheit);
  humidity = DHT.humidity;
  Serial.println((String)"Temperature (C) = " + tempCelcsius);
  Serial.println((String)"Temperature (F) = " + tempFahrenheit);
  Serial.println((String)"Humidity = "+ humidity);
  delay(5000);
}
```

Figure E2: DHT11 Temperature and Humidity Sensor Code

From: 300234067929930 <300234067929930@rockblock.rock7.com>
Subject: Message 25 from RockBLOCK 300234067929930
Date: April 29, 2019 at 7:02:08 AM CDT
To: sfarrel1@trinity.edu

IMEI: 300234067929930
MOMSN: 25
Transmit Time: 2019-04-29T12:00:28Z UTC
Iridium Latitude: 29.5039
Iridium Longitude: -96.7870
Iridium CEP: 181.0
Iridium Session Status: 2
Data: 060073007f74007f73008073008073008274007f72008173007f74007f71008172008173008073008273008003730080

Figure E3: Email from RockBLOCK containing the water station data

```
Seans-MacBook-Pro-2:Python_Code seanfarrell$ clear
Seans-MacBook-Pro-2:Python_Code seanfarrell$ python myprogram.py
Length of the string:  48
[6, 0, 115, 0, 127, 116, 0, 127, 115, 0, 128, 115, 0, 128, 115, 0, 130, 116, 0, 127, 114, 0, 129, 115, 0, 127, 116, 0, 127, 113, 0, 129,
 114, 0, 129, 115, 0, 128, 115, 0, 130, 115, 0, 128, 3, 115, 0, 128]
Water Station current weight (lbs):  115.0
Water Station current Jugs:  13.7889688249
Water Station current panel current (A) 0.0
Water Station current battery voltage (V) 12.8
Array index 11
Data at array index 128.0
Raw float values in average function
Water Station old Jugs:  115.0
Water Station old panel current (A) 0.0
Water Station old battery voltage (V) 12.8
After Average function
Water Station Jugs:  13.7889688249
Water Station panel current (A) 0.0
Water Station battery voltage (V) 12.8
Seans-MacBook-Pro-2:Python_Code seanfarrell$
                                                   LF  UTF-8  Python  GitHub  Git (0)
```

Figure E4: Output from Python code used to decode the email data from RockBLOCK

Phase 1 Version 2 Stations Table

Show 10 **entries** **Search:**

Station	# Jugs	Latitude	Longitude	CEP	Upload Time	Current	Battery Voltage
1	12	-98.1358	27.0377	2.0	2019-04-29 16:17	0	0
Station 2	Inactive						
Station 3	Inactive						
Station 4	Inactive						
Station 5	Inactive						
6	14.0	-96.7870	29.5039	181.0	2019-04-29 07:05	0.0	12.8

Showing 1 to 6 of 6 entries Previous Next

Figure E5: Water station data posted on Bluehost database from the Python code

Appendix F: Power Systems Testing, Additional Figures

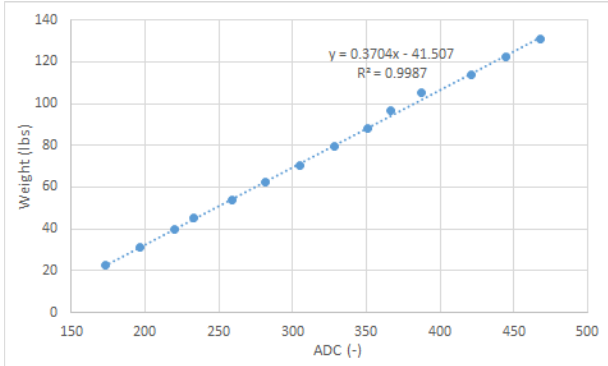

Figure F1: Weight sensor calibration curve.

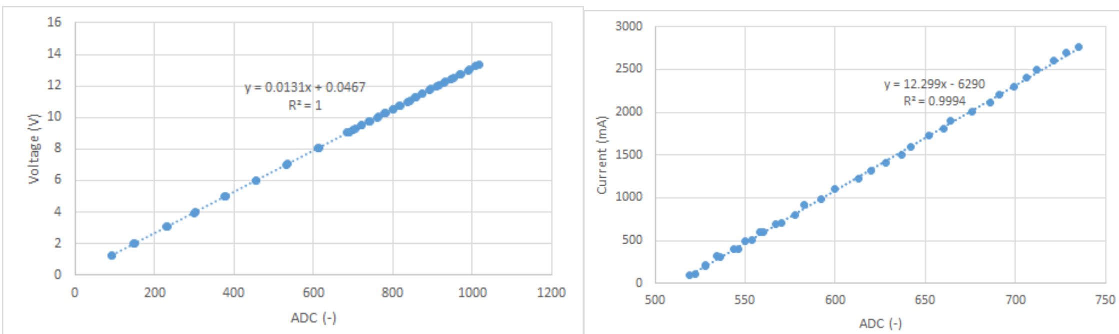

Figure F2: Voltage and current sensor calibration curves.

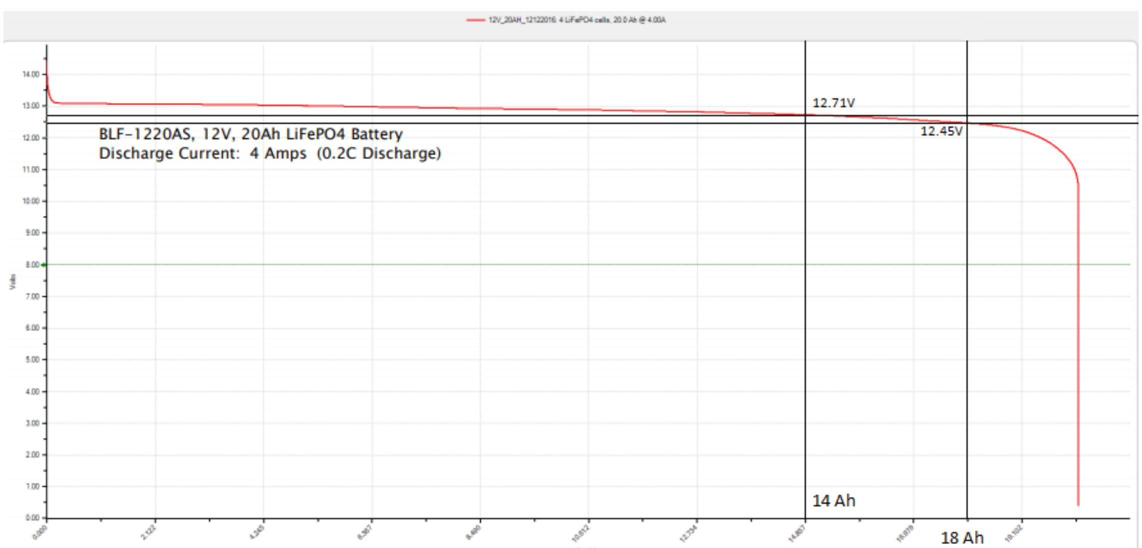

Figure F3: Discharge curve for 12V Bioenna LiFePO4 battery.

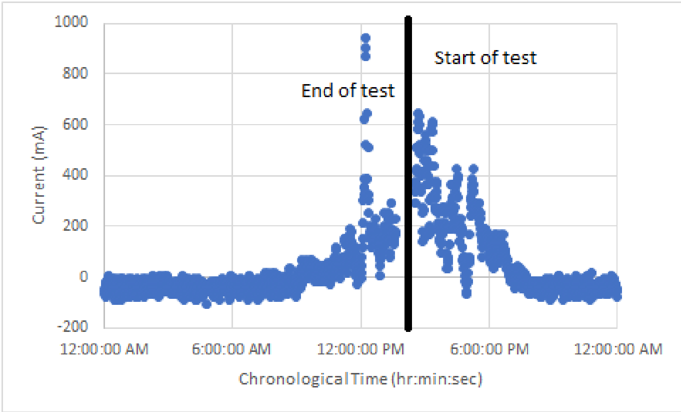

Figure F4: Plot of measured current (mA) vs. time collected on SD card for cloudy day test

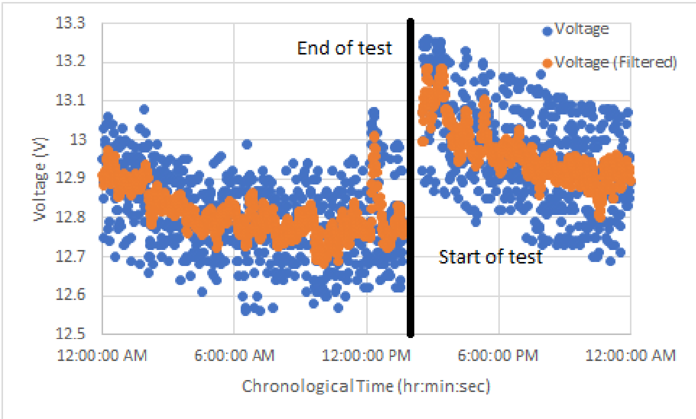

Figure F5: Plot of measured voltage (mA) vs. time collected on SD card for cloudy day test

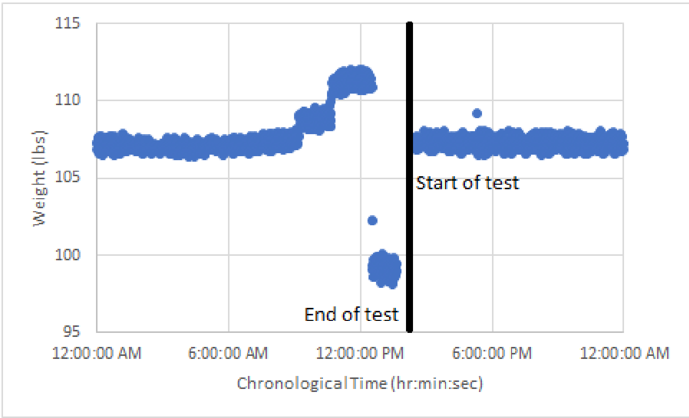

Figure F6: Plot of measured weight (lbs) vs. time collected on SD card for cloudy day test

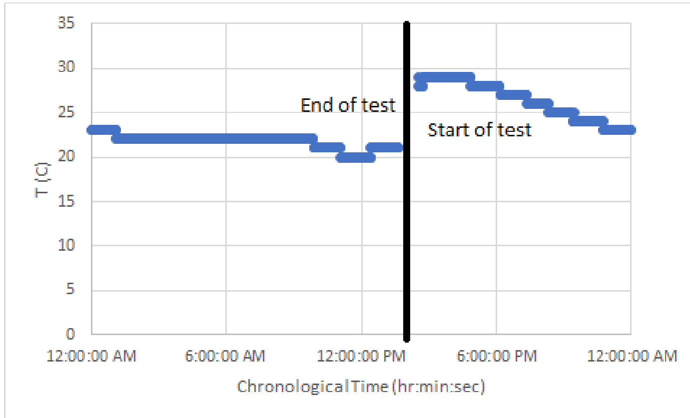

Figure F7: Plot of measured temperature (C) vs. time collected on SD card for cloudy day test

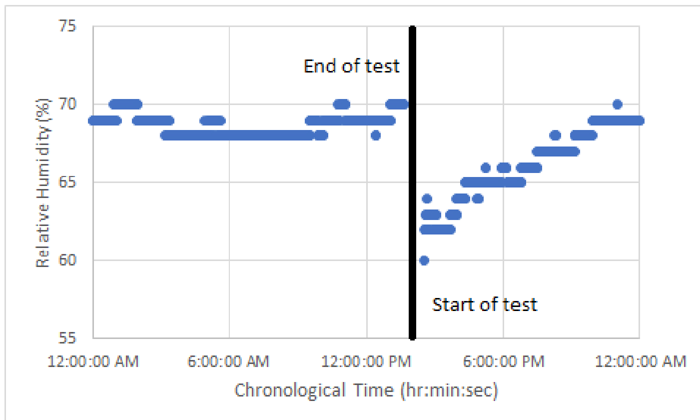

Figure F8: Plot of measured relative humidity (%) vs. time collected on SD card for cloudy day test

Bibliography

[1] R. Bond, J. Bradley, C. Campbell, A. Dinh, E. Edrington, Y. Farra, B. Hood, C. Lansdale, S. Pottian, J. Regan, K. Schoer, K. Walls, A. Zavala, "Final project report: emergency water supply stations," Trinity University, San Antonio, Texas, 27 April, 2017.

[2] Bioenno Power, *Bioenno Power SC-122420JUD CC/CV Series User Manual*. Santa Anna, California: Bioenno, 2019, pp. 1-4.

[3] Renogy, "RNG-50D-SSl," 50W Monocrystalline Solar Panel datasheet

[4] L. Ada, "The mysteries of Apple device charging," *learn.adafruit.com*, para. 15, June 11, 2013 [Online]. Available: https://learn.adafruit.com/minty-boost/icharging [Accessed: April 27, 2019]

[5] Recom, "R-78E-1.0," DC-DC Converter datasheet, [Revised April 2017].

[6] A. Industries, "Adafruit DS3231 Precision RTC Breakout", *Adafruit.com*, 2019. [Online]. Available: https://www.adafruit.com/product/3013?gclid=EAIaIQobChMIloDd0v324QIVAZ7ACh1vtgAQEAQYASABEgJGavD_BwE.