RENEWABLE ENERGY SELF SUSTAINING SERVER MODEL

Deployable Prototype Documentation

SPRING 2017

CHENG, ARIK
JAMES, TAYLOR
MIETZ, ZACH
SHERWOOD, RYAN

***Abstract*—In looking at the societal problem of there being a finite amount of fossil fuels left, our group wanted to find a system that requires large amounts of energy and see if we could make it fully self-sustaining on renewable energy. Our goal is to create a fully self-sustaining, batch processing server, running only on power form a renewable source with battery backup. This document will focus on the design, funding, scoping, risk assessment, testing, status, and forecast of our product.**

***Index Terms*-Renewable energy sources, maximum power point trackers, dc-dc power converters, photovoltaic systems, energy efficiency, servers, global warming, solid state circuit design**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This report provides an analysis and evaluation of the project progress from winter 2016 through spring 2017. All aspects of the laboratory prototype have been expanded upon or refined to deliver a deployable prototype.

Methods of analysis include calculation, simulation, and execution of circuit design; drafting and revision of software; market research of potential market entry and viability; and rigorous testing of system components.

Results of the market data analyzed show that this project fits best in more specified markets. Data show that potential clients include green conscience small enterprises and businesses, schools, or governments of developing countries. In developing countries PV electricity generation is economical because it may be one of the cheapest sources of electricity.

Thorough testing of the solar charge controller revealed many holes in the team's approach to rapid prototyping. Many iterations of the MPPT/Buck circuit were created and tested to find a topology that would be sufficient for the desired power specifications of the project. Analysis of the circuit resulted in an inductive spiking phenomenon that would many times cause fatal temperatures in the circuit. This was finally corrected with the implementation of a snubber circuit across the inductor as referred to in Figure # [#]. The method of finding maximum power point has been changed to reduce complexity and minimize the chance of error.

Calibration of the measurement, voltage regulation, and gas gauging of the system were necessarily and fortunately smooth in implementing. The minor problem in power measurement was an erroneous offset that occurs when the microcontroller is powered with differing power sources. This was quickly remedied by powering the microcontroller with a fixed power source.

The software was expanded upon to include a database to allow a deeper analysis of the data, potentially to make calculations to predict by season or other time frames. A web based graphical user interface is implemented to show real time results on easy to read graphs.

The report finds the prospects of the projects in its current positions are positive and may have a strong viability in specified markets. Data analysis from results collected provide a stronger foundation to the working principals of the hardware electronics and well as more robustly handle electrical fault situations. Expansion of the project provides a more refined user interfaces and a more compact product ready for deployment.

The report also investigates the fact that the analysis conducted has limitations. Some of the limitations include: forecasting power generation based on long-term time frames (such as seasons), scalability, and heat management.

I. Introduction

With the rate at which we consume energy climbing and the finite amount of fossil fuels dwindling there is a great need to quickly increase our reliance on renewable energies. Unfortunately we cannot just say we want to switch to using only renewable energies as the current state of renewable energy technology isn't up to the task of fully powering the world. Furthermore, our demand for energy needs to at least remain idle but preferably decrease some. With these things in mind we decided to focus more on how we could demonstrate the ability to minimize the use of fossil fuels while showing the effectiveness of renewable energies. From there we looked into things that have a high energy need while also being high in demand now and for the foreseeable future. In narrowing our focus we found that a server would be the perfect model with which to work. Now with a product, we had to choose a form of renewable energy with which to power our server and decided to go with photovoltaic solar arrays. With a chemical based storage device to hold the power from the solar array, we believed we could run our server, with batch processing, over a full 24 hour period. In thinking deeper about the product we wanted to show the ability to control the flow of voltage and current into the storage device from the solar arrays and onto the server itself. To accomplish this accurately we needed a device called a solar charge controller. Accompanying the charge controller would be a microcontroller that measures current and voltages throughout the circuit between the solar arrays and the output of the chemical storage device. This microcontroller could then make calculations on this data and send that information to a mini-computer for data collection and display through a database and web GUI. The mini-computer, or load control server, would then be in charge of controlling the server's utilization based on perceived power availability from the chemical storage device. To procure the parts for this project we did not receive any outside funding. However, if not for one of our members having access to server hardware we would not have been able to afford them ourselves. Also we had originally budgeted to purchase solar arrays ourselves but Professor Tatro allowed us to borrow two of the schools arrays. Next we needed to break our project down into manageable sub tasks that would be assigned to certain group members. With our group structure consisting of two Electrical Engineers (EEE) and two Computer Engineers (CpE) we mostly split the project into two categories. The EEE members would do tasks that concentrated on hardware and circuit design. While the CpE members would handle the software programming. Some exceptions, programming of Arduino was done by EEE members and hardware install of server was done by CpE members. In addition we broke the project into four sub tasks with each sub task having either one or multiple activities. Member Arik, with help from Zach, handled the solar panel and regulator design circuitry. The storage system and providing power to the microcontrollers and server was handled by Zach. Member Taylor, with help from Ryan, handled the software programming of the server. While Ryan was in charge of programming the monitoring and control system. With this completed we were instructed to look at all sub tasks and activities to assess the possible risks of failure and find critical paths that would slow down the project drastically if not completed on time. Subsequently, we were asked to think of ways to mitigate these possible risks if they were to occur. In doing this task we discovered that our most critical juncture of the project revolved around the designing of the solar charge controller circuit. With this knowledge we decided to purchase an already build controller and leave the implementing of our own charge controller for the spring semester. During the spring semester we found out just how truly difficult it is to construct a solar charge controller. There were multiple some design changes that happened throughout our year long process of constructing this product specifically with the designing of the solar charge controller. We went through multiple microprocessors before we decided on the Arduino Mega to handle the current and voltage data collection as well as controlling the PWM signal for the maximum power point tracking algorithm. With the current status of our project hitting all our original goals or feature sets, we started to look at ways of improving the prototype if given more time. Lastly we went through and researched what we be the most effect way to market our product and what areas would we try and target. What follows is an in depth look at the process from finding a societal problem to a working deployable prototype.

II. Societal Problem

The future of energy lies in the proliferation of renewable energies, but what does that really mean and how do we accomplish such a task? What it means is that our reliability on fossil fuels as the dominate form of energy as a society cannot be sustained. Fossil Fuels are by nature a finite resource. As we will explore later in our paper this resource can be exasperated in our lifetime. With this information we, as a society, have no other option then to research and or create alternative solutions to fossil fuels for our energy consumption. While it is very difficult for certain aspects of our society to just quit using fossil fuels cold turkey, some are capable of moving to using more renewable energies. Nevertheless, there is very little stopping entities (residential, commercial, or industrial) in non-third world countries from utilizing some form of renewable energy to offset the use of fossil fuels. However we still have industries that refuse to see the writings on the wall and incorporate some aspects of renewable energy into the fold. One area that in recent years that has seen an acceptance of at least trying to incorporate renewable energies into their plans are tech companies. This is important specifically concerning data centers, as they take quite a bit of energy, and the need only increasing with so many products able to connect to the internet. To this end we believe a good solution would be to make data centers, more specifically a server, run only on renewable energies.

A. Increasing Demand

As heavily populated countries develop into first world nations, their energy consumption has skyrocketed. In the future energy demand will only be increasing at greater and greater rates due to our increasing population and continued reliance on non-renewable energies. There have been many strides towards meeting this demand of electricity through cleaner sources and renewable resources. Going forward we will have to be able to leverage technology to not only optimize the production of electricity but also the consumption of electricity. As technology has developed, along political incentives, have pushed companies to make more efficient devices with less power draw. Even with these improvements energy demand is skyrocketing and it must be managed in order to limit the environmental impact of non-renewable resources.

B. Why Are Fossil Fuels A Problem

The number one reason fossil fuels should be avoided is due to the environmental hazards that they provide, namely pollution. Fossil fuel consumption is the major contributor to the increasing concentration of carbon dioxide in the atmosphere and a key cause of global warming [1]. This same study shows that global warming reduces agricultural production and causes other biological problems [1]. If the burning of fossil fuels continue, the planet may face some very steep consequences in the near future. Fossil fuels are currently the major energy source being used in the world. However, it is only a matter of time before this finite source of fuel will run out. Globally, every year we currently consume the equivalent of over 11 billion tons of oil in fossil fuel and use crude oil at a net rate of 4 billion tons a year. Based off a CIA factbook study, oil is projected to last until 2052. Furthermore, gas would last us until 2060 and coal until 2088. Figure 1 [2] does not account for the amount of fossil fuel to be added to current supply. According to Tim Appenzeller of National Geographic's,

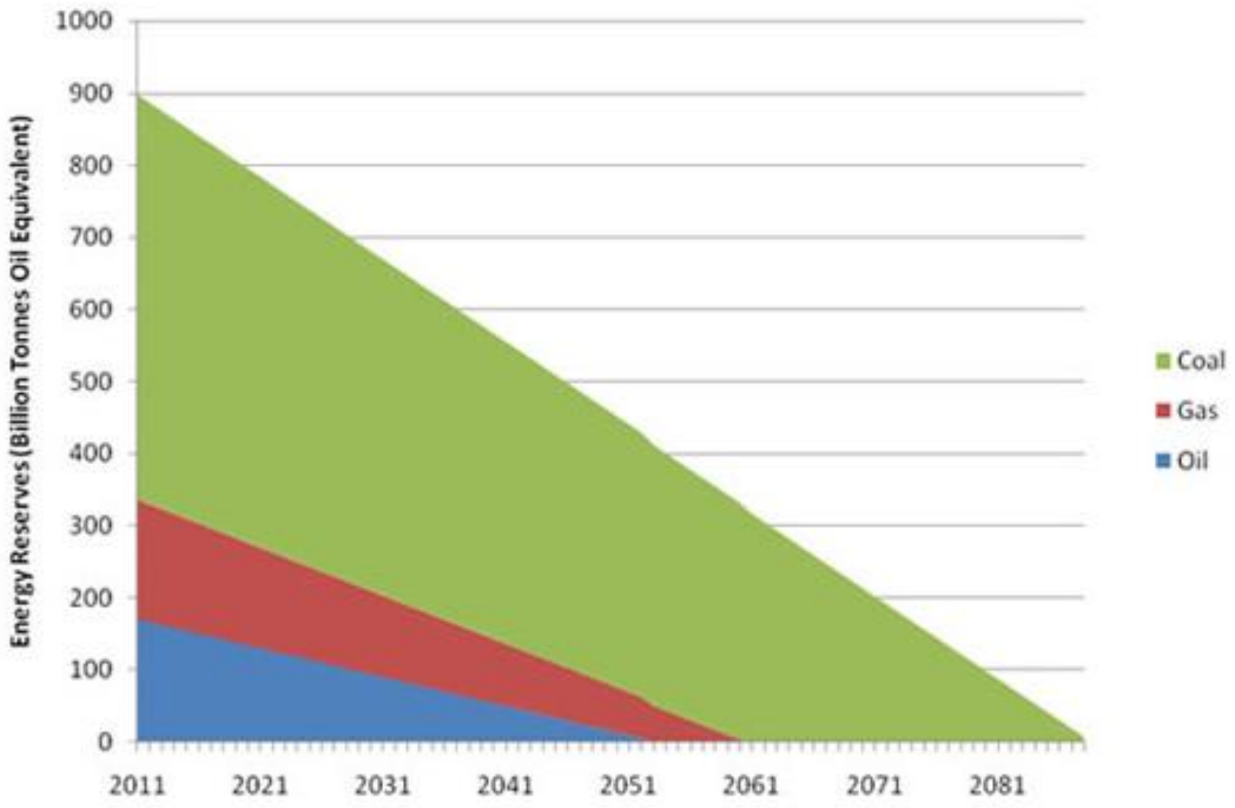

Fig. 1. Energy Reserves Left vs Time *[2]*

"more than a quarter trillion tons of coal lie underfoot, from the Appalachians through the Illinois Basin to the Rocky Mountains—enough to last 250 years at today's consumption rate. You hear it again and again: The U.S. is the Saudi Arabia of coal. About 40 coal-burning power plants are now being designed or built in the U.S. China, also rich in coal, could build several hundred by 2025."

Even still, this is not the practical way to continue as a source of energy. There is no easy way to capture all the carbon dioxide from the traditional burning of coal and we will eventually run out. The move to renewable energy is a beneficial alternative to that of fossil fuel. Although, it doesn't come without its own obstacles. A study done by the US Bureau of the Censor shows that Coal, petroleum, natural gas, and other mined fuels provide 75% of US electricity and 93% of other US energy needs [3].

According to the United States Energy Information Administration, the US total energy consumption in 2007 was over 27 trillion kWh. The majority of the energy consumed is generated by non-renewable and non-environmentally friendly fossil fuel sources such as coal petroleum, and natural gas.

Currently, the use of renewable energy (including hydroelectric energy) in the United States is only 6.8% of the total energy consumed, which is a level much lower than other developed nations, where fossil fuel prices are historically higher than the United States. In anticipation of the United States move towards greater renewable energy utilization, the ability to efficiently use these resources must be addressed. One major challenge to the widespread adoption of renewable energy is the ability to store and control the wide variety of different energy resources. Therefore, to enable the widespread utilization of long term, secure, sustainable, and environmentally friendly energy, the future electric distribution grid must address the issues of storage and complex control [4].

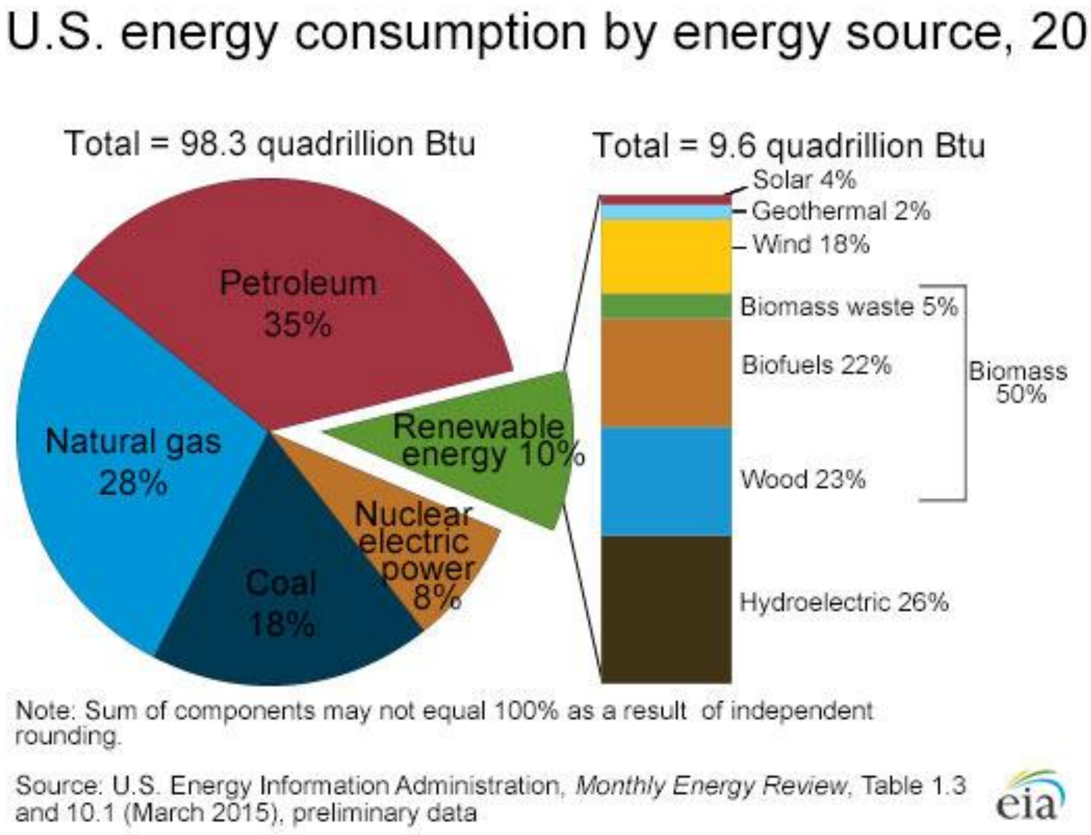

Fig. 2. U.S. Energy Consumption by Energy Source *[5]*

C. Integrating Energy Into The Future

In the current energy market the use of renewable energy resources is a small but rapidly growing industry. As of 2014 we were only generating 10% of consumed power by the U.S. with renewable energy sources, as can be seen in Figure 2 [5]. There is still a huge amount of energy we are generating from fossil fuels, and the reality of this is that there is only a finite amount of these resources left. This brings the question of how long until we run out of these fuels, and by when should we be reliant completely on renewable energy resources. As stated above in Figure 2 [5], at the current consumption of these fossil fuels we will be completely out of fossil fuels by the year 2088. This means that it's not a matter of if we will completely convert over to renewable energy resources, it's a matter of how soon we can and will be willing to do it. Fortunately there are many different methods that have been developed that are useful for producing renewable energy. Jacobsen predicts a reduction in fossil fuels and a rise in renewable energy resources shown in Figure 3 [6]. According to Jacobson, if we reduce the demand for energy by making energy usage more efficient and increase the use of renewable energies we could be using energy completely

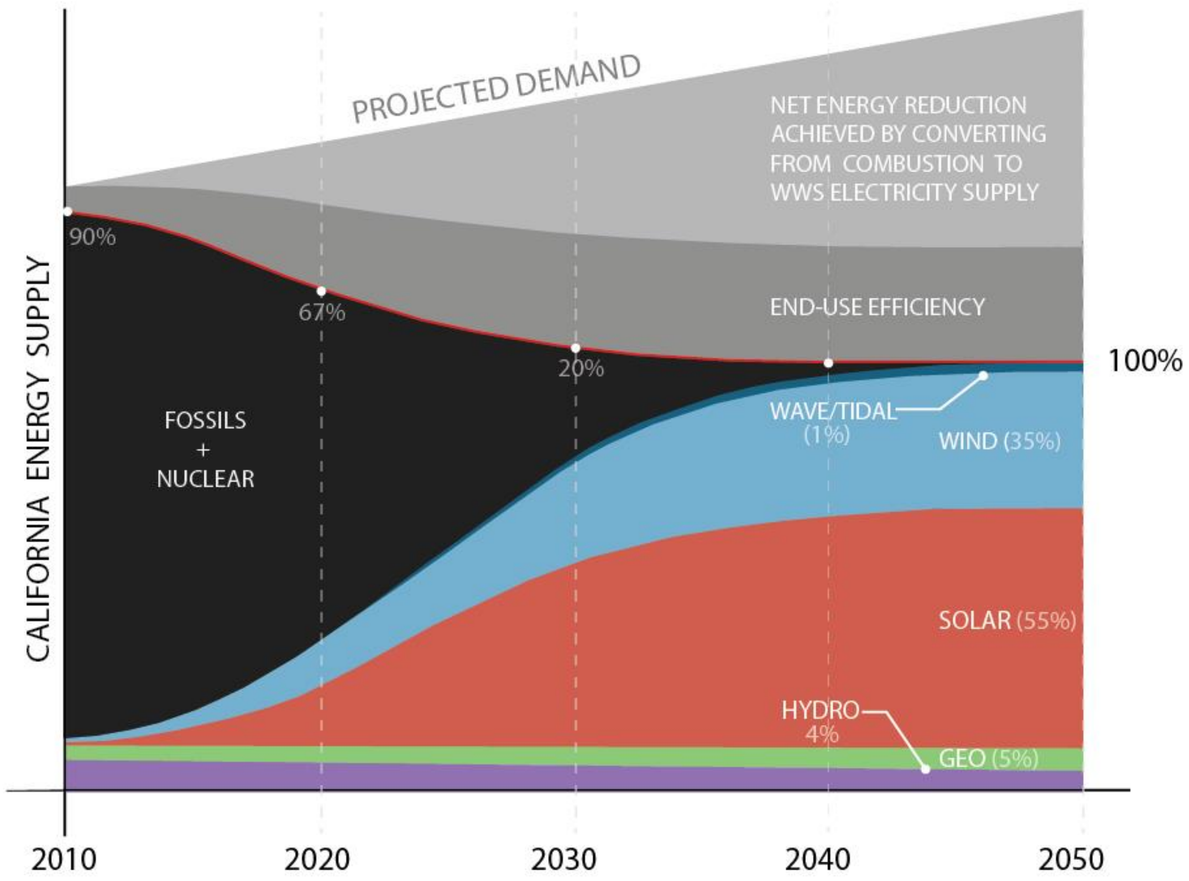

Fig. 3. Projected Timeline of Energy Usage in California *[6]*

from renewable resources in California by 2050. It shows the most efficient types of renewable energy that can be used in California and gives a realistic prediction on how much energy would come from each source. To get to this point would require a huge push to renewable resources, but in time it is completely feasible to expect the U.S. and eventually the rest of the world to run on renewable energy.

It's obvious that using renewable energy is going to be the way of the future, however there is many problems in implementing this types of energy resources into our already massive power grid in the United States. One huge problem is that there is going to be an enormous initial cost to create all these renewable energy resources. With only such a small percentage of renewable power being used today, we would have to greatly increase the production of these various types of renewable energy sources. One example of this, which many people are starting to implement in their own homes is the use of solar powered energy. There is a growing market for the use of solar power as an alternative of buying electricity from the power companies. As stated there is a large investment, however once the system can be implemented it eventually pays for itself in the amount that the homeowner is saving on their electric bill every month. According to energyinformative.org, the amount of time it would take for solar panels to pay for themselves could range from about 6 to 15 years depending on which state you live in and the cost of electricity in your area. The initial cost to retrofit an average size home can range from about $10,000 to $20,000 depending on how many watts the system needs to be capable of handling. These initial costs will decrease as solar panels become more affordable due to the increase in production of solar systems. Once solar use for residential buildings becomes more affordable it will become a much more prominent way of providing energy. Once this happens we will start to see major changes in how we generate and supply energy.

Once people start using renewable energy sources like solar to provide energy there will be less of a demand on the big power companies such as PG&E and SMUD, however there is a foreseeable problem for when we make this big shift to solar. As of now, most residents who use solar power are still connected to the main power grid. This allows them to pull energy from the grid when it cannot be provided by their solar panels. It also allows them to feed energy back into the grid to be used by other consumers of energy. In a way this acts like a large battery for people who use solar. With a massive shift over to solar this will become a problem, because more and more people will be getting their energy from solar and wind, which is an energy source that can fluctuate greatly depending on the environment it's in. This becomes a problem because the users of solar power will have no place to store their energy generated from the solar panels. Without any place to store energy it would be likely that they will run out of power during peak usage hours and at times when there is no sunlight. This is where technologies such as Tesla's Powerwall would come into use. According to IER [7] "One of the biggest problems with electricity from solar and wind power is that these sources of electricity are not reliable because of their intermittent nature. [. . .] Tesla claims that they have overcome much of these problems with its Powerwall battery". Technologies such as this would help solve the huge problem of intermittent energy supply from this renewable resources. It is likely that in the future age of renewable resources technology such as Tesla's Powerwall would exist in every home with them all running of renewable energy such as wind and solar. These technologies would also have to be introduced in the commercial application such as large buildings with high energy needs. With the incorporation of these energy storing technologies combined with the use of solar, wind, and geothermal energy the U.S. would be able to run completely off of renewable power once there was a big enough supply to meet the energy demand.

Another huge shift that will likely happen is the shift from AC to DC power. It would make sense that if homes are generating their own power from solar they would convert over to DC power. The reason behind this is because AC is only useful for sending long distances, however with solar it is on top of the roof so it does not have to travel far. This would be helpful for many things since DC power is a more useable power. Almost everything in our homes use DC power so it would make sense to switch to DC given the means. This will actually help with savings in a lot of ways because we can eliminate power lines traveling from home to home. In combination with new battery technologies to keep energy consistent we could use solar to completely detach houses from the grid. Overall this would actually save a lot of money in infrastructure costs, once the initial cost of solar was paid.

D. Servers and Power Efficiency

Knowing that we have a finite amount of fossil fuels left to produce energy, we need to immediately reduce our energy reliance on fossil fuels. In doing this we should also try and create technologies that do not rely on fossil fuels at all if possible. As of 2014 the estimated power data centers consumed was 70 billion kWh, which represents about 1.8% of total U.S. electricity consumption. This is the equivalent of the amount about 6.4 million American homes used in that same year [8]. That leads to close to a 4 percent increase in the total energy consumption from 2010 to 2014, while also being significantly bigger then the five years before which saw power consumption of data centers grow by about 24 percent [9].With the significant advances that have been made in power efficiency have played a role in the slowing growth rate of energy consumption in data centers. Specifically the current efficiency improvements have saved about 620 billion kWh between 2010 and 2020 as shown in Figure 4 [9]. These improvement are very good and show that we as a society are trending in the right direction but it doesn't paint the whole picture of how all servers are used. On the other hand there is still a large amount of inefficient servers out there being run at small enterprise IT centers and also in third world countries across the globe. It is important to provide incentives for these smaller companies or enterprises to upgrade their servers. Simultaneously we can help these smaller enterprises by showing the ability to incorporate some of the renewable energies we have discussed earlier in this paper.

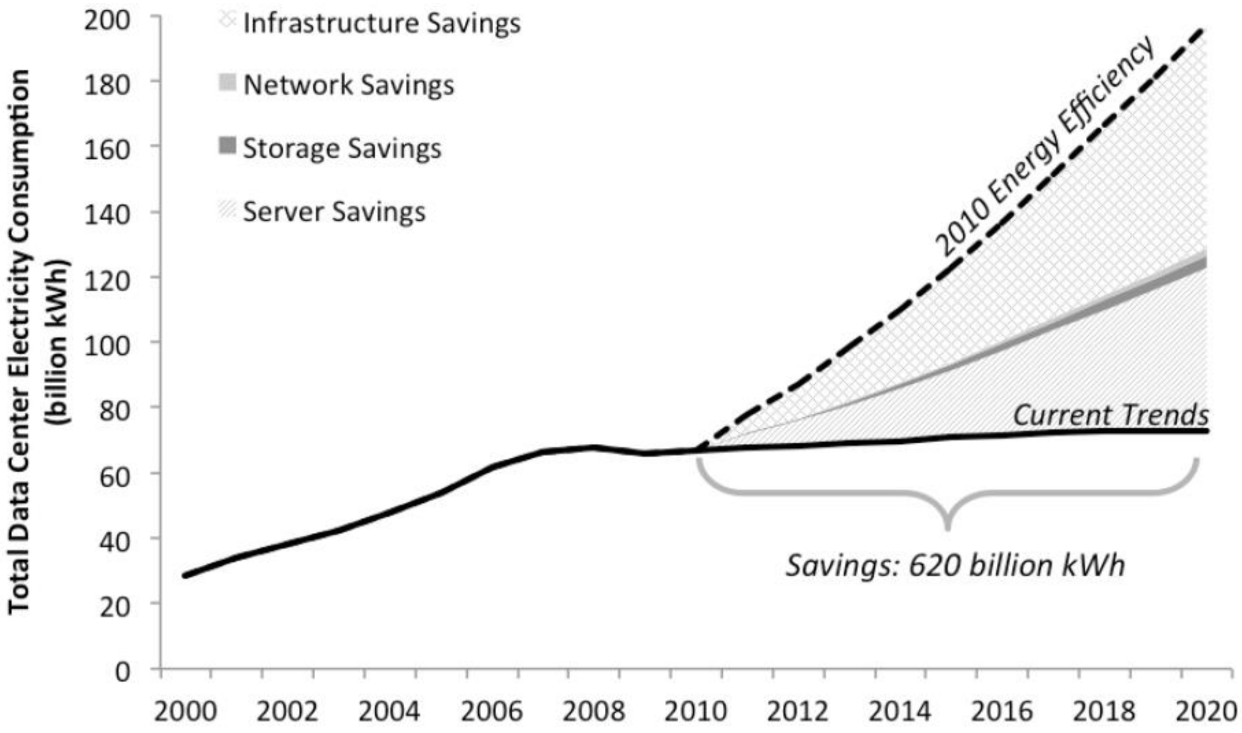

Fig. 4. Past and Projected Growth Rate of Total U.S. Data Center Energy Use *[9]*

III. Design Idea

Knowing that we have a finite amount of fossil fuels left to produce energy, we need to immediately reduce our energy reliance on fossil fuels. In doing this we should also try and create technologies that do not rely on fossil fuels at all if possible. As previously stated above the estimated power data centers consumed was 70 billion kWh. While this number does not look particularly damaging, the use of cloud storage will proliferate the need for data centers which will raise the percentage of U.S. electricity consumption felt by data centers. Without any advance in the energy efficiency of data centers it will only increase the reliance on fossil fuels. Our idea of having a batch processing server that runs completely off of the power grid will show the ability to lower the fossil fuel consumption felt by data centers. While it would be rather difficult to accomplish a real time data center that is powered completely off grid, our project can show that relying solely on fossil fuel energy is not necessary. Some of the technologies we will be incorporating into this project are: photovoltaic arrays, a chemical based storage device, a solar controller that will efficiently charge the storage device based off the power of the photovoltaic array and the current charge in the storage device, multiple microcontrollers and a high core low power draw processor. No major data centers are running on power created solely by renewable energies. The current state of renewable energy technology is just not capable of powering a major data center but there are many companies that begun to incorporate renewable energy into their data centers. For example, in 2009 Emerson Network Power installed a 7800 square foot solar array on the roof of its data center in St. Louis. Even with this large of a solar array at peak output it will supply only about 16% of the data center's power requirements [10]. A big problem that arises in using renewable energy to power data centers, specifically solar and wind, is that you do not receive stable power and need a device to store excess power that can be used win the renewable energy is not at peak performance, for a large data center this is not an easy task. Our project is different in this aspect as we will be running solely on renewable energy created by the photovoltaic array and chemical based storage device.

A. Resources Needed

The resources that we forecast a need for in this project will be divided into two categories. One category outlines the resources needed for the renewable energy harvesting, monitoring, storage, and consumption. The second category will include the hardware and software aspects of the data center. The renewable energy resources are comprised of the solar panel(s), the charge controller circuit, the microcontroller system, the chemical based storage, and the voltage regulator. The solar panel portion of this project is projected to be rated for around 200W as a single panel or as an array. The charge controller will require many circuit elements within the system. These elements include: transistors, capacitors, a buck converter, inductors, resistors, diodes, fuses, and wire (high enough gauge to handle ~20A). The microcontroller will require a serial cable, current sensors, and voltage sensors. We will be utilizing a chemical storage in either 12V or 24V bank system. Lastly, the renewable energy system will require a 12V regulator circuit comprised of circuit elements. The data center side of the project will require server hardware in order to run the workload. The server will require a small DC-DC power supply in order to directly power the motherboard power inputs. A Gigabyte mini-computer will be required in order to make a USB to USB connection to the other microcontroller that will be managing the power devices. A router will be required in order to facilitate the connection between the server and the Gigabyte.

B. Feature Set

1) Solar Array:
   Series of solar panels capable of outputting 150 to 200 watts of power for charging the chemical based storage device.
2) Chemical Based Storage Device:
   This storage device will be responsible for keeping the server running during the hours of the day when there is not enough sunlight to run the server off the power coming from the solar arrays.
3) Server Platform:
   The server platform will feature a 4+ core Intel CPU which will consume anywhere from 55W-105W of power.
4) Batch Processing:
   The server will be continuously calculating prime numbers, which can be benchmarked over a 24 hour period to gauge the performance of the server.
5) Monitoring System Power:
   The server will be able to monitor power production, power consumption, and battery storage level. These metrics will be used to calculate the server's consumption level using a series of microcontrollers.
6) Display Processed Data:
   The server will report the amount of prime numbers that have been calculated, the power generated by the solar panel/array, and power consumed by the server. This data will be displayed using a small display screen or a webserver on a small microcontroller.
7) Future Power Prediction:
   This feature will allow the server to take into account the sunrise and sunset times to predict future power production.

C. Goal

The goal of the project is to create a small scale model of a renewable energy datacenter. This is something that has been done in other places but in a different way. Most datacenters will pull a constant amount of electricity and then source as much of the power as possible from renewable sources. The design that we have described takes it a step further by using the latest technologies in renewable energy and changing the consumption of the server based on power availability. This ensures that when renewable energy is not able to fully power a datacenter, the datacenter will reduce load to remove dependence on nonrenewable resources. The implementation is mix of hardware and software solutions to create something that may not have been done before. There will still be limitations of the system. The system is purely dependent on solar power and only has a finite amount of energy storage, there is a potential that the system will have to be turned off for periods of time in order to increase the total amount of computation done in the long run. This limitation is being mitigated by focusing on batch processing and scientific applications that are not time sensitive. This project will be a challenge to our individual abilities and will be a good stepping stone to potential projects in our future careers.

IV. Funding

At the beginning of our project we agreed that our maximum budget would be $1000. With not having to buy the server processor and hard drive, because Taylor had access to both, we thought this was very feasible. As Table I [11] shows we expecting to come under our maximum budget by about $100. With help from Professor Tatro we were able to use two of the schools solar arrays which significantly helped cut out cost as seen in Table II [12].

TABLE I
Estimated Budget *[11]*

	Q	**Price**	**Sub Total**
Buck Converter	1	$25.00	$25.00
Current Sensor	3	$10.00	$30.00
Power Converter	1	$50.00	$50.00
Solar Charge Controller	1	$90.00	$90.00
Solar Arrays	4	$100.00	$400.00
Battery	2	$100.00	$200.00
Misc Elements		$100.00	$100.00
		Total	$895.00

TABLE II
Actual Budget *[12]*

Description of Part(s)	**Q**	**Price**	**Sub Total**
DC-DC Buck Step Down Converter	1	$25.80	$25.80
Module Current Sensor	3	$6.99	$20.97
LCD 2004 Display Module for Arduino	1	$12.99	$12.99
picoPSU-160-XT 12V DC-DC Power Converter	1	$44.50	$44.50
Dc Universal Regulated Switching Power Supply	1	$19.99	$19.99
MPPT Tracer2210RN Solar Charge Controller	1	$93.42	$93.42
12 Gauge Spade Connectors	1	$7.99	$7.99
12 Gauge Wire, 50' Blue	1	$10.95	$10.95
12 Gauge Wire, 50' Red	1	$9.95	$9.95
12 Gauge Wire, 50' Black	1	$9.95	$9.95
MCP3008 10-bit 8 channel ADC	1	$8.17	$8.17
75Amp Hour Deep Cycle Battery	1	$81.99	$81.99
IR2011 Mosfet Driver	5	$18.39	$18.39
Uxcell 8-pin DIP Socket	10	$4.81	$4.81
BNTECHGO Solder Wire	3	$9.98	$9.98
30A range Current Sensor ACS712	2	$8.99	$8.99
15SQ045 Schottky Diode	20	$6.99	$6.99
Magnet Wire 16 AWG 16' Length	1	$6.88	$6.88
Uxcell 36mm Torriod Inductor Core	2	$9.48	$9.48
		Total	$412.19
Items Previously purchased	**Q**	**Price**	**Sub Total**
Arduino Micro	1	$15.00	$15.00
Raspberry Pi 2 Model B	1	$89.99	$89.99
Misc Circuit Elements		$40.00	$40.00
		Total	$144.99
Actual Total			$557.18

V. Project Milestones

To best describe the milestones we have hit for our project we will be breaking it down by each task and describe the important accomplishments that have been achieved.

A. Solar Panel and Regulator

The solar panel and regulator task revolves inevitably around the Maximum Power Point Tracking (MPPT) with a solar charge controller. Completion of this task was critical in order to accomplish efficient power generation for the system. To do so, a solar charge controller is required in order to route maximum power from the solar panel array safely to the battery storage system.

The maximum power point tracking circuit (MPPT circuit) contains a microcontroller which delivers a PWM signal to the circuit. The signal is then amplified by a MOSFET driver and sent to the gate of a high side MOSFET circuit. By adjusting the duty cycle of this PWM signal, the circuit determines how the voltage is regulated at the output.

The microcontroller also measures voltage and current at the solar panel array, into the battery, and into the load. By utilizing these values, the microcontroller calculates the power generation, power consumption, and battery level (utilized to perform gas gauging). The PV solar panel voltage measurement it also utilized to calculate the maximum power point from the solar panel array. Previously, the system used a Perturb and Observe algorithm in order to find the maximum power point. However, the method of finding maximum power was then replace by Constant Voltage Method. This method assumes the maximum power point is located when the voltage is adjusted to 76% of the open circuit voltage across the solar panel array. This change provided a more elegant method, less prone to error.

Once the solar charge controller was built, some safety aspects needed to be implemented. On the software side we have implemented different charging states based on a few aspects of the system. When the system sees a battery voltage attached to the charge controller and the solar panel is at a high enough voltage to charge the battery the charge controller will enter is bulk charging state. This is when the charge controller is in normal operation charging the battery from the solar array. If there is no battery voltage the charger enters sleep mode, which sets a digital pin low to the mosfet driver causing it to activate the shutdown mode. This allows no current to flow through the circuit shutting down all mosfet switching and the output regulation circuit.

One safety aspect on the hardware side is implementation of fuses on both the input and the output of the charge controller. This prevents any chance for excess current to flow through the charge controller. The last crucial safety feature that was implemented was the use of a snubber diode across our inductor. This was a necessary feature to reduce voltage spikes from the inductor as the magnetic field begins to collapse. Before the implementation of this we were seeing a lot of voltage spikes that was endangering our mosfets when switching at high voltages. The voltage spikes were exceeding our gate to source voltage of our high side switching fet causing it to burn out. This safety feature will help alleviate this issue.

Milestone: Solar Charge Controller

B. Battery Storage System

As stated in the solar panel and regulator section 'gas gauging' is also important to do with the battery. In regard to the battery we needed to know how much power could the battery pump out and for how long. At first we didn't have the server hooked up to the battery so we used a 2Ω resistor, supplied to us by Professor Tatro, as a load for preliminary tests on the battery. Simultaneously we started to work on a buck converter that would supply a steady 12V to the server, or load in this case, as the server needs a constant 12V supply. The design of the buck converter was becoming very laborious so with permission from our advisor we purchased an after-market unit and implemented in series with our boost converter. Lastly, as the title of our project states we needed devices to be powered from the battery as well and not by the grid. For this we used linear voltage regulators that could step down the 12V to the appropriate voltages of the individual devices.

Milestone: Gas Gauging of Battery

C. Server Running Batch Processing

The most time consuming activity on this task was by far designing the server load algorithm. This encompasses having to deal with the processors kernel directly to manipulate the availability of the different cores depending on our needs. In direct conjunction with that activity was implementing the load modulation. This activity is how the previous algorithm will be implemented on the processor. We could not complete this step until the load algorithm was finished. Next was the process wait states which actually reduce the processors utilization and also be used in case the server needs to shut down in the event the battery will not be able to supply it with power for the time needed. Also we needed

to connect the server to the load control server so that both units could communicate specified information between each other. We needed the server to send the amount of prime numbers that have been calculated to the load control server to eventually be displayed on the web GUI. Concurrently the server needed to receive a specified CPU target utilization from the load control server based on the estimated battery life.
Milestone: Server Algorithm

D. Monitoring and Control System

The monitoring system was the slowest moving task we had as it required progress from the other tasks before being able to implement. The first activity accomplished was interfacing the load control server with the Arduino Mega which was established by using a micro USB to USB cable. Next was the connection between the server and the load control server, which was done through a router using an Ethernet cable. For this we had to setup a UDP network protocol. We did this in Python on the load control server and through a JavaScript on the Server. With those complete and able to receive data we then developed our database and web GUI. The database we created using MySQL while the web GUI was constructed using JavaScript. Lastly was the algorithm that was used to send the CPU target utilization to the server based off the estimated battery life. This was originally made with the purpose of being fairly conservative before our field test, but during testing we found that it worked quite well and decided to leave as is.
Milestone: Server Utilization Algorithm

VI. Work Breakdown Structure

Each of our four sub tasks had a team leader but none were completed by only that person. The total hours our team has spent on the project over both semesters is approximately 1300 hours. This total includes both the documentation and implementation of our project so far. These findings along with the hours spent on each task by the group and by each member individually can be found in Table III [13].

TABLE III
TASK ASSESSMENT AND HOURS *[13]*

Feature	**Total Hours**	**Arik**	**Ryan**	**Taylor**	**Zach**
Problem Statement	12.5	3.5	2	2	5
Design Idea Contract	32	8	7	9	8
WBS	13	2	4	3	4
Timeline	7	1	1	4	1
Risk Assessment	15	5	5	3	2
Laboratory Prototype Document	40	8	14	9	9
Problem Statement Revision	10	3	4		3
Design Idea/Timeline Revision	4			4	
Device Test Plan	10.5	3	2.5	2	3
Market Review Report/Presentation	20.5	6	6.5	4	4
Test Results	22	10	5	2	5
Feature Set Report/Presentation	22	6	8	4	4
Deployable Prototype Document	55	10	30	5	10
Server Load Algorithm	100		5	95	
Power System Controller	20		10		10
Battery Storage System	80	30			50
Solar Panel Setup	20	10			10
Build Prototype Server Hardware	20		15	5	
Load Modulation	60			60	
Measure & Send Data to Pi	25	12			13
Interface With Solar Charge Controller	20	20			
Install/Test Purchased MPPT	8	4			4
Process Wait/Save States	28			28	
Get Data From Server	40		20	20	
Fine Tune Hardware Power Consumption	15		10	5	
Send Power Information to Server	30		25	5	
Voltage Regulator & Server Protection	40	25			15
Adjust Power State Based on Data	20	5	5	5	5
Build MPPT Hardware	241	101			140
Code MPPT Software	106	56	10	2	38
Adjust Power State Based on Data	36		30	6	
Database & Web GUI	35		5	30	
Interface With Solar Charge Controller	35		30		5
Gas Gauging	21				21
Build Component Enclosure	75	70			5
Total Hours	**1338.5**	**398.5**	**254**	**312**	**374**

VII. RISK ASSESSMENT

Being as this the first group project, of this size and magnitude, we have been involved with it can be difficult to assess the risk of our project. While it may be a difficult task it is something we realize that is extremely important for a large group project. When we talk about risk assessment we want to identify potential hazards and analyze what could happen if said hazards occur and how can you mitigate them [14]. To identify

TABLE IV
RISK ASSESSMENT *[15]*

Probability	**80-100%**	🟨	🟨	🟥	🟥	🟥
	60-79%	🟨	🟨	🟨	🟥	🟥
	40-59%	🟩	🟨	🟨 **Server Load Algorithm**	🟨 **Integrate Power System**	🟥 **Solar Charge Controller**
	20-39%	🟩 **Server Hardware**	🟩 **Process wait/save states**	🟨 **Measuring Power**	🟨 **Solar Array Overpower** **Adjust Server Utilization**	🟨
	0-19%	🟩 **AC-DC Server Power** **Load Control Server** **Database & Web GUI**	🟩 **Setting up Arduino** **Tune Hardware Power Consumption** **Reporting Calculations**	🟩	🟨 **Solar Array Reverse Power** **Power & Voltage Reporting**	🟨
		Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**
		Impact				

these risks we went through our Work Breakdown Structure (WBS) and discussed the possible risks that could occur for each activity. Once we had identified all our possible risks we used a risk assessment chart, Table IV [15], which shows the level of the risk versus the level of impact on the project. With this completed we were able to identify the critical path of the project. From there we looked at all the risks and decide how we could best mitigate them if they do occur. The risks that present themselves in our project can be divided into two clearly defined parts: hardware risks and software risks. Although the software risks may not have as big of an impact if they occur, they may occur more frequently and can cause problems in the progression of the project. The hardware risks do seem to have a bigger effect on the project if they occur and must be mitigated accordingly. In analyzing the activities of the project, our team is looking for all the potential risks that may occur so that we can account for and lessen the effects if/when something goes wrong.

A. Hardware

Many of the hardware risks are similar in the fact that they are potentially very high impact if or when they occur. The biggest and most important risk that has been identified in the hardware side is safely controlling and utilizing the magnitude of electrical power that will be output from the solar panels. This will apply to and from the battery and into the load (server).

1) Solar Array:

   Experimenting with solar panel arrays presents the risk of electrocution to team members and unsafe electricity to the other parts of the system. In this project, there is a risk that the current or voltage is more than the wire or other electrical elements are rated for and thus may cause damage to the system. There are also concerns of power going back into the solar panel during non-sunlight areas. Both of these risks have a relatively low percentage of occurring. We guess 20% chance of electrical components being overpowered and a 5% chance of reverse power to the solar panels if not mitigated. The impact of either of these risks will be 4 on the impact scale.

2) Solar Charge Controller:

   By building a solar controller from scratch, many risks are introduced into the project that would not be present otherwise. Again we will have to make sure that the circuit elements in the charge controller are matched and rated for the potential high current and voltage that will be received from the solar panel array. Next the circuit will have to protect the solar panel and chemical storage bank from many faulty conditions such as overcurrent, overvoltage, and reverse power. So in effect, the solar charge controller is a form of risk mitigation for the rest of the circuit as a secondary purpose to its charging efficiency. Because of the complexity of this circuit, there is a moderate chance (40%) that the circuit may not work in a timely matter. The impact of this risk would be a 5 and would bottleneck the project if not mitigated.

3) Measuring Power:

   To measure the voltage and current of the solar panel and chemical storage bank multiple voltage dividers will have to be added into the circuit. This is because the microcontroller input that we will be using can only take inputs voltages between 0 and 5 volts. We will run the risk of damaging the microcontroller or other parts of the circuit if the circuit is not built safety and accurately. We rate the probability of this risk as 10% and an impact score of 3. A secondary risk that arises from measuring the power of the hardware is in incorrectly reading the voltage and current of the solar panel and chemical storage device. This could damage the system or affect the efficiency of the charging. The probability of this occurring would be 20% if not mitigated and impact of 3.

4) Setting up Arduino:

   The risk that may occur in setting up the microcontroller is in the coding. This could cause the microcontroller to either read or send measurements inaccurately throughout the system that differ from the actual measurement. The percentage for these risks are relatively low (10% for both) and have an impact of 2.

5) AC-DC Server Power:

   This task involves connecting an AC-DC power supply to the DC-DC server power supply so that work can be done on the server before the renewable energy system is completed. The risk is a hardware risk involved around either of the power supplies not functioning. The approximate chance of this happening is 5-10% with a 1 on impact scale.

6) Prototype Server Hardware:

   This task involves the assembly of a consumer level computer hardware that will have similar power and performance characteristics to the server hardware. The risk of any component in the system not functioning is approximately 20% with an impact of 2.

7) Load Control Server:

   The load control server will be the primary device that will be integrated into the power management controller and the server. The primary risk is if the hardware in the controller is faulty, which has an approximate probability of 5% with an impact of 1. The other risk is if too much input voltage is put into the controller.

8) Integrate Power System:
This activity will play an integral part of our project as it combines the separate activities of the project into a single, uniform system. The big risk in this activity is in making sure that electricity flows in the way that we anticipate through analysis and calculation. There is a high possibility that the parts of the system do not work correctly when initially connected. The impact of this risk could potentially damage components of the system. The probability of this occurrence is guessed to be around 40% with an impact of 4.

B. Software

As stated above and reiterated below the risk for the software task having to spend more time on the problem. None of the programming we are doing has the ability to destroy any of our hardware.

1) Server Load Algorithm:
This is the primary software development task and its risk is inherently different from a hardware risk. The risk of underestimating the completion time is 50% because the estimate is only approximated and additional risks will be discovered throughout development. The impact level is a 3. Software development is an iterative process of software failure and debugging. Risk cannot be viewed as a pass and fail, but rather a fail then after time a pass.

2) Power & Voltage Reporting:
The risk in this is the Arduino measuring the current or voltage incorrectly, or calculating the power incorrectly. Either of these scenarios would lead to the load control server receiving incorrect data which would affect the reporting to the database and web GUI. The bigger issue would be that our server utilization level is based on the output voltage of the battery at a known state which is very important to our models self-sustainability. The risk of failure of sending incorrect data to the load control server for web GUI reporting is fairly low, approximately 10%. The impact is a 4 because if the data is incorrect it would have a big negative affect on our projects self-sustainability.

3) Tune Hardware Power Consumption:
This task will be done once the load algorithm and the hardware is built. The hardware itself will be adjusted in order to increase and decrease power consumption. The processor can be switched out to a lower power variant and the clock can be adjusted to reduce power consumption. If the system crashes after or during the process of fine tuning the hardware, the operating system can crash and in turn lead to the corruption of the operating system. This is a low likelihood of 5% with an impact of 2.

4) Process Wait/Save States:
This will allow the software to pause its state mid-way through calculation so that the server can reduce power consumption to its idle power state. The save state is slightly more risky because there is a risk of data loss throughout shutdown and reboot. This task also involved to connection of a relay to the power connectors on the motherboard. There is a potential that either the relay will break or the motherboard will function, but the risk of this happening is approximately 20% with a level 2 impact.

5) Reporting Calculations:
This activity is purely software and as such its risk is sensitive to time, not hardware or cost. I approximate that the activity will be not be completed in the budgeted amount of time to be 10%. The impact of not being able to report our calculations is a 2. It is important and in our feature set but will be worked on early in the project which will allow plenty of time for trouble shooting.

6) Adjust Server Utilization:

This activity is part of the critical path because it involved every measurement device in our project to be reporting to the load control server. It also requires the server's power consumption to be properly modulated. The step is purely dependent on software, but the risk of the task taking longer than anticipated is likely because the amount of software that this step is dependent upon. I rate the probability of failure 35% with an impact of 4.

7) Database & Web GUI:
The database and web GUI will be how we are displaying all pertinent information for testing and future analysis. This task is not on our feature set so if proven to be too difficult we can always scrape it. We see the probability of to be low at 15% with an impact of 1.

C. *Mitigation*

Though we have identified the risks involved in each part of the project, there are many things we can do to mitigate or lessen the effects of the risks to the system. We have taken an engineering approach to mitigating the risks that arise in the project by choosing the best course of action to each risk.

1) Solar Array:
In order to minimize the probability of overpowering circuit elements in our system we will have at least 2 people checking and calculating that connections made in the system are accurate and that elements are rated for the current and voltage that they will potentially encounter. In this way, we do not lessen the impact of the risk, but the probability of the risk occurrence. The second risk, reverse power to the solar panel array during non-sunlight hours, will be mitigated through the use of the solar panel controller. This will be done using diodes. Again by doing this, we lessen the probability of the risk occurring.

2) Solar Charge Controller:
The solar charge controller presents a high percentage, high impact risk to the project and also presents itself as a bottleneck. Because of this, we choose to temporarily substitute a purchased MPPT charge controller into the system for the first semester while we build a MPPT charge controller from scratch for our deployable prototype. This will allow us to spend extra time and care in assembling the rest of the circuit and significantly lowers the probability of faults in our electrical system. In accurately creating a MPPT charge controller, we take an iterative approach in building the circuit. First we will test each part of the circuit individually to ensure that the elements are working as specified and to learn about the characteristics of the elements in given conditions. We will then test the assembled circuit with reference voltages and currents to double check the behavior of the circuit. Then we will be able to implement the build MPPT charge controller into the system with a lower chance of risk to the rest of the project components. Lastly, at least two members of the team will be looking over any iteration of the circuit to ensure accurate and safely connected circuitry.

3) Measuring Power:
The mitigation of this risk is approached in a similar fashion as that of the solar charge controller above. The wiring of the voltage divider circuits will be first tested in a controlled circuit before being implemented with the solar panel array. The connections will also be double checked by Arik and Zach.

4) Setting up Arduino:
Setting up the microcontroller will require that hardware and software risks be addressed. The hardware circuitry will be double checked by Arik and Zach and the software will be double checked by Taylor and Ryan. This will ensure that the hardware connections are accurate and the analyzation of the measurements are accurately represented.

5) AC-DC Server Power:

The mitigation plan is to immediately order a new power supply to replace the broken one and return the broken power supply. Which would delay this task for 2 days. Temporarily we could also use the adjustable power supplies on campus.

6) Prototype Server Hardware:
If a component is broken the mitigation plan is to replace the component. A broken component in the system is easily replaceable because of the fact that the system is based on consumer hardware and can be replaced the same day for free because of the high availability of resources. The other mitigation is to build the operating system in a virtual machine where the software can be built, instead of relying on physical hardware. This alternate mitigation will be more work later because the software will have to be migrated to physical hardware.
7) Load Control Server:
If the hardware is faulty, then two other identical controllers can be sourced from the team the same day. The software on the controller will be saved on an external device at regular intervals for backup purposes. This will insure minimal down time if the load control server was to be deemed faulty.
8) Integrate Power System:
Integrating the whole system together will be an iterative process to make sure that the project works correctly. This will lessen the probability of risks occurring. We will do this be connecting only two parts of the system together at a time to ensure that they work together correctly. We will then add one additional part at a time to the system and test. This will we lessen the impact of risk to the whole system by only imposing part of the system at a time.
9) Server Load Algorithm:
This is a core step to the project and load modulation cannot be added to a piece of software that is not built from the ground up for this purpose. There is no third party software that can be temporarily used in place of this activity.
10) Power & Voltage Reporting:
If the power measurement device fails the mitigation is to order another one, which would delay progress for 2 days. There is also a small risk of putting too much voltage into the controller, approximately 2%. If this happens, the SD card from the controller can be moved to another identical controller on hand. The team has approximately 3 identical controllers at a cost of $30 apiece.
11) Tune Hardware Power Consumption:
To reduce risk in this process, the storage of the machine will be duplicated, so that if the OS crashes throughout the process, it can be quickly and easily restored to its previous software state. If the OS is crashed, the mitigation would only take up to 2 hours to restore to a known good state. There is also a very low risk of breaking the motherboard or processor in the process, 5%. If this happens, the mitigation is to get new hardware to replace the broken hardware. This can be done in as shortly as the following day.
12) Process Wait/Save States:
The mitigation if the hardware fails is to retrieve the identical hardware so that the software state will not have to change. The other mitigation is to operate within a virtual machine then we will be able to simulate every one of the events before testing on the physical hardware.
13) Reporting Calculations:
If this task falls behind schedule other individuals with particular knowledge in networking software can be involved to give insight into the task and accelerate its progress. This will be a critical path because this task needs to be completed for the product to hit our feature set. This activity will be watched closely for its progress.
14) Adjust Server Utilization:
This step of the software development needed to be watched closely. It was extremely vital to the project that the load

control server receive correct data of the output battery voltage in order to decide the target utilization for the server.

15) Database & Web GUI:
Since this step is neither critical to the project nor apart of the feature set if we come across to many problems we will just stop trying to implement it. We do not for see this to be a problem as Taylor has extensive knowledge in these two areas.

VIII. Design Documentation

To give the proper detail of how we designed our project we are going to break it up with sections corresponding to major sub tasks. All diagrams referenced too can be found in either the Hardware or Software appendices.

A. PV System

For our Solar Array we needed to choose a panel that would meet our power needs. The panel, shown in Figure 6 [16], must be able to

Fig. 6. Solar Array *[16]*

provide enough power to run our server and have enough extra energy to charge the battery during the daytime. Since our server uses about 50W when it is running at full power during the day, that means we needed an excess of that in order for our system to function properly. We ended up choosing a 100W panel with an open circuit voltage of 22.4V and a short circuit current of 5A.

The panel, even though rated for 100W does not realistically give us 100W in real scenarios. We were looking more in the range of about 60W to 70W depending on the weather. This meant that we were still able to charge our battery while running the server even at times when the server needs to run at full power. This panel was sufficient to provide power to our server, load control server, router, and microcontroller while still being able to charge our battery to a full charge for the night use.

B. Voltage Regulation

Voltage regulation is an important part of this project. We have four devices that needed to be powered by a regulated voltage and not all by the same amount of voltage. To power the Arduino we needed to regulate the battery voltage to about 8V. To do this we used a voltage regulating chip called LM317. This chip allows us to take the battery voltage and regulate it to a voltage that is safe for the Arduino to use. It uses a feedback resistor to adjust the amount of voltage that is then outputted by the chip. This same chip and feedback resistor setup is also used to power the Gigabyte and network router. Once we were able to provide power to our microcontroller, Gigabyte, and router, we needed to provide a steady stable 12V DC for the server to run on. To do this we used a boost buck circuit. We originally had planned on using a buck converter to regulate the voltage from the battery as seen in Figure 5 [17]. After lab testing we encountered a problem of losing about one volt across the buck converter. This meant that there would be no way to get 12V to the server if the battery was not outputting at least 13V. To alleviate this problem

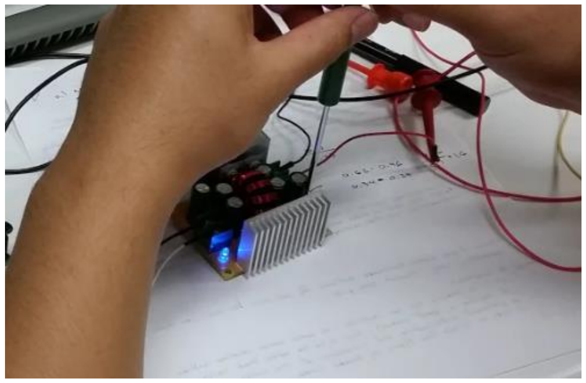

Fig. 5. Buck Experiment *[17]*

we connected a boost converter in series with the buck, which can be seen in Figure 7 [18]. The boost converter would boost the incoming voltage

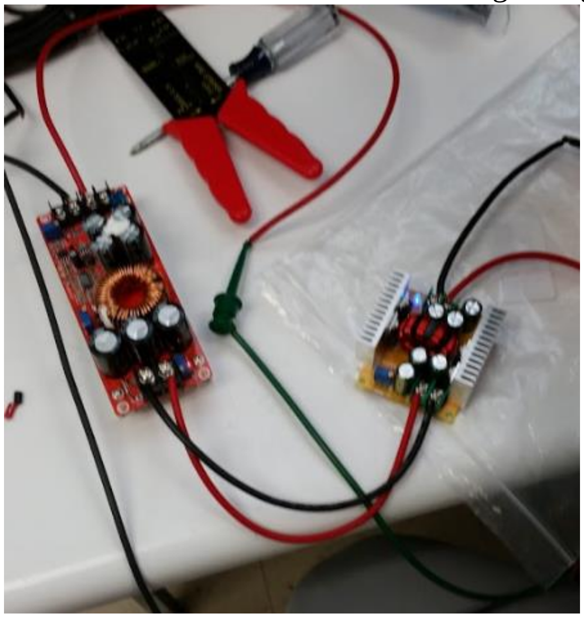

Fig. 7. Buck and Boost Circuit *[18]*

up to 18.5V which we then used the buck converter to regulate the voltage back down to a solid 12V.

C. Solar Charge Controller

The solar charge controller was the most difficult part of the design process on the hardware side. We needed to create a circuit that was able to output a regulated voltage for the battery from a range of about 12V to 13.5V and also was able to impedance match the solar panel to achieve maximum power point. The solution we came up with for this charge controller was using a DC to DC buck converter to step down the voltage of the panel to meet the voltage requirements to charge the battery. We used a microcontroller to control this buck converter with a PWM signal into a high side switching Nfet. This allows us to change the output voltage by adjusting the duty cycle of our PWM signal. As the duty cycle increases, the impedance of the circuit will decrease as well as the output voltage will increase. This allows us to actively track maximum power point using our microcontroller while keeping the charging voltage from the circuit in a safe range to charge our battery. When choosing the parts for our buck converter there were some specific formulas that we followed to size our inductor and capacitors accordingly.

$V_{in} = 24V$

$I_{out} \approx 6A * 2 = 12A$

$Switching\ Freq = 50kHz$

$Duty\ Cycle = \frac{V_{out}}{V_{in}} = 50\%$

$DL \approx 4.2A$

$L = (V_{in} - V_{out}) * D * \frac{1}{F_s} * \frac{1}{DL} = 2.9 * 10^{-5}H$

$C = \frac{DL}{8(F_s * 20mV)} = 525 * 10^{-3}F$

Once we had the right sized inductor and capacitor we needed to find a diode that would work for our circuit. We found out that we needed a fast switching Schottky diode that would also be able to handle the current we could potentially put through the diode. We ended up choosing the 15SQ045 Schottky diodes to meet these requirements. Through some initial testing we realized that our inductor capacitor combination that we chose did not allow our circuit to have a big enough impedance swing when adjusting the duty cycle which therefore meant it was not able to achieve maximum power point from the solar panel. We had to redesign our circuit. We ended up going with a much larger inductor and capacitor combination, the inductor being 248μH and the capacitor of 880uF. Once we implemented this change we were then able to get the voltage swing and impedance swing we needed to be able to achieve maximum power point from our panel while being at a safe output voltage for our battery to charge at.

D. Measurement Controller

The measurement controller we designed was actually built into our solar charge controller since most of the measurements could be taken from the solar charge controller. This measurement controller consisted of various voltage and current measurements in order for our microcontroller to both send data to the load control server and to regulate the PWM duty cycle for the solar charge controller. The measurement controller consists of three voltage dividers to get the voltage from the panel, the battery, and the load. We then had three current sensors measuring current from the panel, into the battery, and feeding the load. With the combination of both the current and voltage measurements the measurement controller is able to calculate the power provided by the solar array, and the power used by the load. This allows us to see that our system is actually achieving maximum power point. It also allows us to find the battery life that is left in the battery in order for the load control server to regulate the server utilization. This will be discussed more in the next section. The measurement controller is a crucial controller in our system because without it nothing will work correctly. It was a fairly simple design due to the fact it consists of current sensors and voltage dividers, but proved to take more time than first anticipated due to programming and tuning all of the sensors.

E. *Load Control Server*

The premise of the project pivots around the load control server, and is ultimately implemented in its software. The load control server is in charge of dictating what the CPU utilization of the server should be, along with running a database and web GUI. We use the database to store pertinent data for future analysis of the efficiency of the product. The web GUI is being used as a tool to display real time data of power production and consumption, calculated primes numbers, and estimated battery life. The load control server has four threads that run on its Linux CentOS; a main thread and three child processes. The main thread is where we spawn our three child threads as well as open a UDP socket for receiving data from the server. A flow diagram for the load control server can be found in Appendix C.

The first of the three child processes controls the updating of the server utilization. When the thread is created the function updateServer is called. This function waits for thirty seconds then calls another function idleServer, which will tell the server through the UDP connection to idle down to zero CPU utilization. For estimating our battery life we decided that best way to accomplish this was to measure the output voltage of the battery at a known load. The idea being that if you are checking the voltage at a constant load you should get a good estimation of the battery life. After the load control server has told the server to idle down it waits three seconds before calling monitorBattery which takes the latest battery output voltage measured and decides, by a series of if/else statements, what the new server utilization should be. With both the monitorBattery and idleServer dunctions complete, updateServer will now send the new CPU utilization through the UDP connection and start all over.

The second child process controls the sending data to the database by calling the function sendData. This function connects to the MySQL database and pushes the newest collected data to it. What we hold in the database is the power produced by the solar arrays, power consumed by the server, battery output voltage, CPU target utilization, and the calculated number of primes. The function then sends all the same data to the web GUI, sleeps for ten seconds, and starts all over again.

The final child process controls the serial connection with the Arduino microprocessor. The thread starts and calls the function updateSerial which proceeds to open the USB serial port and listen for incoming traffic. We utilize the imported function readline which is a blocking call, meaning the rest of the code will not run until the readline function has received new data. After receiving the data we save it to our global variables that are used in other threads.

F. *Database and Web GUI*

The database and web GUI are modeled after the primary way that server administrators monitor their systems. Servers are managed almost entirely remotely and the system that we have designed allows this. The load servers gets the current power data from the power controller and logs it into the database. The front end of the web GUI is designed in JavaScript and jQuery to allow asynchronous database transactions with the server. Every ten seconds the front end JavaScript runs a PHP callback script that queries that database for the latest power data. The power data is returned as an XML file to the JavaScript program. The different graphs then index into the XML file in order to get their appropriate values.

There are a total of four graphs on the web GUI: power generated, power consumed, battery voltage, and prime numbers generated. Each of the four graphs scales automatically so that as the numbers increase, the graphs can continue to be scaled relevantly. The JavaScript has been written in such a way that it can be tuned for graphing over a much longer period of time if needed.

The database logs the power information and power targets with timestamps to allow for further power analysis and optimizations. For ease of development and compatibility with the reset of the system Mariadb we utilized.

G. Server

The largest pieces of software exist on the server. The server is running CentOS 7. This is important because unlike the other two controllers, this operating system is preemptive. This means that if a process is in a wait state, the operating system can take a hardware thread away from the process and give it to a higher demanding process. Every piece of code is integrated into the same process, either through the use of sub processes or threads. This is done so that the entire system can easily be throttled in order to tune power consumption. There will be a total of four threads of execution. Flow diagrams for the four different threads can be found in Appendix C.

The first thread will handle calling a kernel module for limiting the processor utilization of a given PID (Process Identification Number). Calling the kernel module directly keeps the operating system from applying a nice value to the process, this then bypasses the scheduling algorithm. This is required to fine tune the utilization of the process. The limiting of CPU utilization will automatically reduce the voltage and frequency of the processor through the utilization of Intel SpeedStep technology. This technology will reduce the processors TDP any split second that the processor is not utilized through the modulation of the onboard VRMs (Voltage Regulation Module). In order to fully reduce the voltage and power draw, the frequency of the processor is reduced. CMOS physics dictates that a transistor fall and rise time is dependent on the square root of the voltage. Thus, by reducing the frequency, we are able to reduce the voltage exponentially. Thus, reducing the power draw drastically. The CPU can make these frequency adjustments as fast as 35ms. Further work will be done so that the kernel module creates time splices that are optimized for Intel SpeedStep Technology.

The second thread of execution on the server is the most complex. In order to keep the simplicity across the network APIs, and the controllers, we used Python. Python is not a very fast language. In order to get the simplicity of integration of Python and the speed of C, Cython was used. Cython allows a programmer to write cross language bindings from Python to C. The second thread of execution leveraged Cython to implement the sieve of Eratosthenes. This algorithm is one of the fastest in existence for calculating prime numbers. The worst-case time complexity of the algorithm is O(n(logn)(loglogn). This allows the algorithm to scale very well with large numbers. In the calculation of billions of digits, memory space is also important, the algorithm uses only O(logn) memory space. Designing a good way to benchmark the system had to be unique. It is not possible to record every prime number because over the course of hours, it would make 10s of terabytes to store. Thus, the simplest way to prove that the calculations are happening is to record the number of prime numbers of the total number of iterated numbers. The fastest known C implementation of the sieve of Eratosthenes was imported into the source code using a git module, to keep on sync with our source code management system. In order to call this code, the Python bindings had to be written and is included in the Appendix C. This allowed us to focus on power management and integration with the other controllers. The method that was used is to calculate the prime numbers in a range, then reevaluate all other variables and report back to the master controller. The load algorithm is fairly simple now that it has been abstracted away from the sieve of Eratosthenes. This process will take all available threads of execution from the processor, taking up to 100% of the CPU utilization. Every other process that runs on the machine besides this process, is considered overhead.

The third thread of execution is the thread to receive data from the load control server. The thread will be locked until it is triggered by a network event on the receiving port. The only thing that the system will receive is the desired CPU utilization. As stated on the previous block diagram, the first thread of execution waits for a receive signal from the master controller. This thread handles that process.

The fourth and final thread of execution is to update the load control server with the most updated prime number information. It will be locked until the second thread (load algorithm) sets a flag that it has outputted updated information. The process will then be sending a UDP packet to the master controller and wait for acknowledgement, then return to a locked state. Returning to a locked state ensures that the CPU cannot be taken away from the load algorithm, and less switching allows higher overall utilization.

By having a separate thread for sending and receiving it allows the software to be full duplex, which means to be able to send and receive simultaneously.

IX. Prototype Status

To best clarify the status of our prototype we split the product into two sections, hardware and software. In these sections we will discuss the test results we received along with the final implementations we chose based off the testing.

A. Hardware

In the hardware side of the system most of the work revolved around the solar charge controller and getting it to correctly regulate the voltage of the panel using the DC to DC buck converter configuration. With many redesigns of our system we were finally able to get the prototype working in the lab off of a regulated lab power supply. It correctly will convert the input DC voltage anywhere from 14 to 22 volts, which is what we are seeing from our solar array, down to the regulated voltage for the battery. Unfortunately we were having a lot of trouble with inductive voltage spiking. In the current configuration which can be seen in Figure 10 [19] the large inductor in the circuit caused large inductive spiking. This was causing a high voltage to be seen at the source of our high side switching n type mosfet. If this spiking exceeded 20V then it

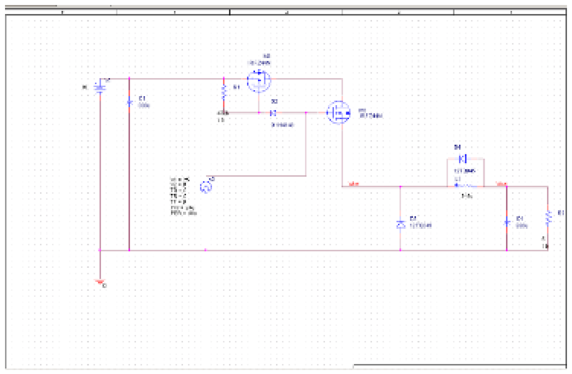
Fig. 8. Original Charge Controller Circuit *[19]*

could potentially exceed our gate to source voltage limit of 20V. This would in turn damage our mosfet which is what we kept seeing on the solar panel. At first we suspected overheating however when we upgraded the heat sync and tried the circuit again we quickly realized that heating was not the cause of our problem. After a lot of trial and error we realized our problem could be solved by introducing a snubber diode in

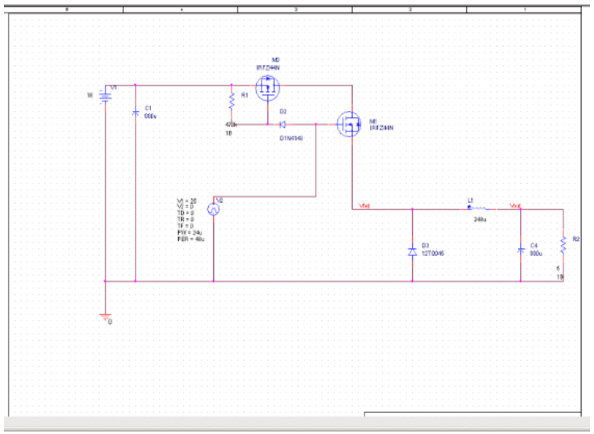
Fig. 9. Snubber Charge Controller Circuit *[20]*

the configuration that can be seen in Figure 9 [20]. Through the use of PSpice we simulated both the first configuration and the configuration with the snubber diode to see if it would alleviate our problem. The simulations can be seen below in Figure 11 [21]. From the first graph you can see large voltage spikes in excess of 25V. This is the simulation before the snubber diode was in place. The following Figure 8 [22] shows the simulation with the snubber diode. The simulations suggest that the snubber diode both cleaned up the signal immensely at the source of our high side switching mosfet and reduced voltage spiking to under 10V which is well within the safe region of the IRZ44N mosfets we were using. Due to lack of time we were unable to implement this solution into our charge

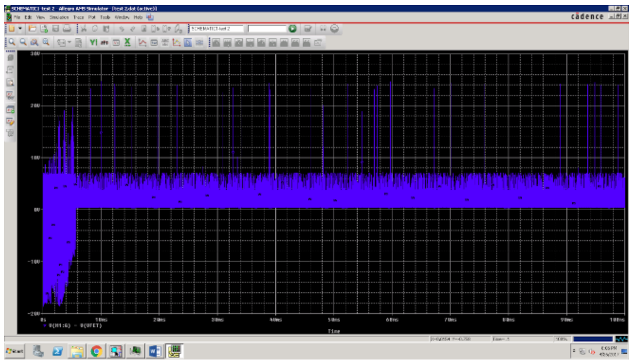
Fig. 10. PSPICE Sim Original Charge Controller *[21]*

controller, however through the research and simulations we have done we are confident that this would fix the problem with our circuit and would allow us to implement it into our system. At this time all other hardware has been completed and is fully functional. We have enclosed everything into a 3D printed case that can be viewed in Figure 12 [23]. With this case we have implemented two 120mm fans in order

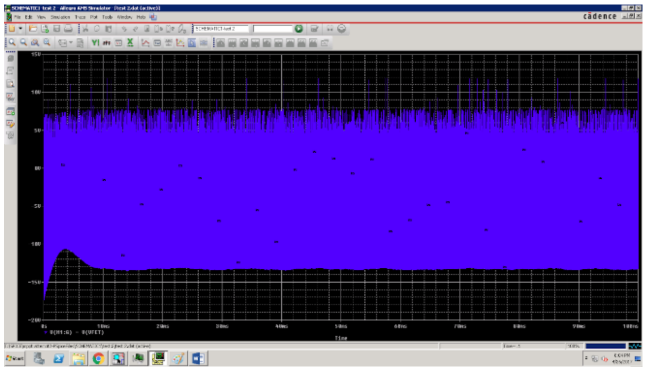
Fig. 13. PSPICE Sim Snubber Charge Controller *[22]*

to provide adequate cooling to our mosfets in our solar charge controller. These fans run off of a LM317 linear regulator circuit. Also we used an LM317 to power our router for the load control server and the load server to be able to communicate. The circuit layout for those can be seen in Appendix B. All of the hardware was able to reside inside of our 3D printed case, including our two fans, solar charge controller, measurement controller, and the boost buck

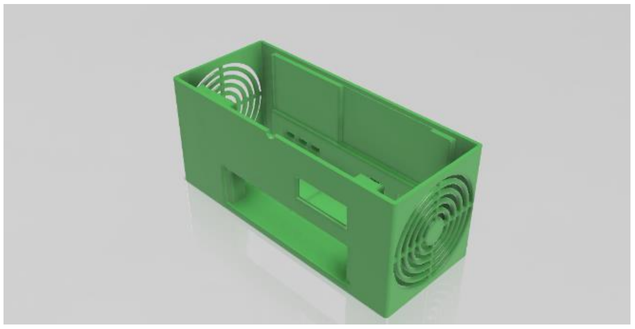
Fig. 11. 3D Model Case for Components *[23]*

configuration that we used to regulate the load voltage. This allows us to have an easy system to set up with quick connects on the case for outputs to the load server, load control server, router, battery, and the panel input. The fully assembled hardware system can be seen in Figure 13 [24].

Overall the hardware is complete with the exception of the fix to our solar charge controller. If time had allowed we believe that the implementation of the snubber diode into the

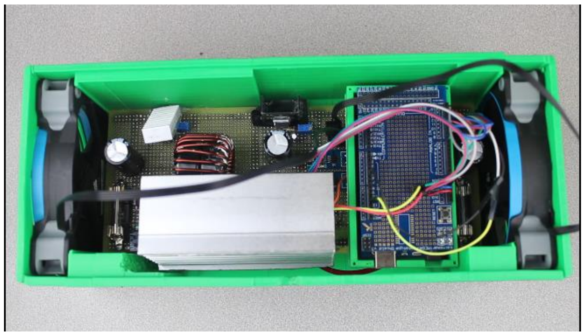
Fig. 12. Actual Component Case *[24]*

circuit would have alleviated our voltage spike issue and allowed the circuit to perform as expected.

B. *Software*

The performance characteristics for the sieve of Eratosthenes on the Intel i7-4790 to find all the prime numbers in the first billion digits seen in Table V [25]. When target

TABLE V
LOAD ALGORITHM PERFORMANCE DATA *[25]*

Target Utilization	CPU TDP	Time
100%	78W	82s
80%	63W	131s
60%	50W	286s
40%	38W	503s
20%	25W	983s

utilization is 100%, the software will always perform significantly faster because the operating system can avoid the cost of context switches and overhead involved with throttling the CPU.

A big part of what the load control server does is interface with other units. For this aspect there isn't really a test in particular, instead we coded try and except blocks into our python code around all connection interfaces. For instance, if there is an issue with the network connection between the load control server and the server then the 'except' block will catch this and try to make the UDP connection again. The main purpose of the load control server is to base the server CPU utilization off of the estimated battery charge level. This algorithm, which is a series of if/else statements that chose a CPU target based off output voltage of the battery, is very much testable. To test this we created a test bench program, on an Arduino, that creates different float values and serially sends the data to the Gigabyte. We then printed the input voltage data and the corresponding CPU target value. Table VI [26] shows that our actual CPU target value matches our expected value.

TABLE VI
LOAD SERVER TEST RESULTS *[26]*

	CPU Target	
Vout	Seen	Expected
12.31	100	100
12.26	90	90
12.21	90	90
12.16	80	80
12.11	70	70
12.06	60	60
12.01	50	50
11.96	40	40
11.91	30	30
11.86	20	20
11.81	10	10
11.76	0	0
11.71	0	0

X. MARKET FORECAST

While trying to solve a societal problem is a very noble cause, you will not get very far if you do not have a clearly defined market. During our market research there two different clients that we believe we could successfully market our product too; green conscience small enterprises and businesses, schools, or governments of developing countries. In doing the research we wanted to find similar projects like one done by a group at University of California, Berkeley that is similar but larger in scale. Lastly we looked at the opportunities in the market currently for renewable energies, specifically photovoltaic (PV) and possible challenges in integrating into existing markets.

While the prototype for our project uses solar energy as our renewable energy we do not believe we have to limit ourselves to only using that. Our true desire is that each potential customer could tailor the product to work with the renewable energy that best fits their needs. Moreover we would like to educate our customers that it is not just about limiting yourself to a certain renewable energy source or even just one renewable energy source. In other words to achieve a truly power efficient data center you should try to incorporate multiple forms of renewable energy. With this in mind we have a price point of about $200 per server, in the customer's setup that we believe we can easily achieve. This price point is if the costumer utilizes a solar array setup like our prototype.

A. Green Conscience Enterprises

We specify green conscience because in developed countries power is so easily and readily available that most small enterprises will want to go with what is easiest for them, which would be to plug their servers or computers straight into the power grid. Another issue in trying to market to small enterprises is their desire to limit upfront cost. While we believe our product will save companies money in the long run there is an upfront cost that would make it harder to convince smaller or startup companies that it is worth the investment. However, with green conscience companies they will be looking specifically for ways that will allow them to have a small carbon footprint like not using power from the grid. While so far this sounds like limiting ourselves to a small and to specific market, we believe that with the publicity of recent global companies (Apple, Google, and Amazon) to move toward a more renewable energy model for their data centers, this will lead to small or startup companies wanting to begin with a more green conscience approach to their energy usage.

B. Developing Countries

The other client we believe that truly fits with the goal of the senior design project of tackling a societal problem is businesses, schools, and government entities in developing countries. Understanding that catering to this market group will most likely not bring with it vast financial gains, we believe our product can help boost these economies. Stable and reliable power, like we have in the U.S., is not a fixture in most developing countries. Consequently there is not the push back from people saying renewable energies are not a reliable enough power source as their current options are not any more stable or reliable. In some places, like east/north Africa or west Asia, Sun and wind are in a particularly high abundance and can benefit from the use of renewable energies greater then we can here in the U.S. [27]. In addition countries like Costa Rica already produce most of their power off of renewable resources [28]. Tailoring or efforts two these to markets we believe that we can help solve our defined societal problem while also having a profitable business model.

C. Similar Projects

There are many corporations that are now operating completely on renewable energy. This is either through the direct acquisition of renewable energy devices, or through PPAs (power purchase agreements). A PPA is a contract with a company that generates electricity purely off of renewable energy sources. All the power generated by these sources is allocated to the company, even though they are not directly attached. Companies like Apple and Google have large teams in their datacenters dedicated to improving the efficiency and environmental sustainability of their datacenters. There are currently many research projects that focus on adapting this large scale model to other companies. A research group in the Department of Computer Science at Rutgers University has developed what they claim as a renewable energy powered data center. The goal of this research project was to further explore the potential of using renewable energy to reduce the carbon footprint of data centers [29]. They have developed a solar powered data center called Parasol. This has been built for research purposes to study co-location and self-generation. Parasol is small datacenter housing 2 racks of servers and network equipment. To power this data center they are using 16 solar panels which can produce up to 3.2kW AC. This project is similar to the Renewable Energy Server Model but one key component that differs between our renewable energy model and the Parasol model is that Parasol are converting the DC power from the panels to AC to be used by the power supplies on the servers. In our renewable energy model it is strictly DC power throughout the whole system. The server uses a DC to DC power supply so that it can be powered without the need for conversions. This allows our system to be more efficient because there are always losses when converting the power from DC to AC and back to DC. The way our project handles the power distribution of the system is unique in a way that there is no need for AC power in our system.

D. Opportunities

There are many opportunities for the integration of our product into the global market. As the price of solar panel installation drops, we see a continuing trend for PV installation. Globally, PV capacity is projected to grow in the coming decades and is expected to be more than 450 GW by 2020 [30]. The USA alone broke records in quarter three of 2016 for PV installations. Among U.S. states, California continues to lead the solar market due to the abundant solar energy and supportive solar policies. In it, nearly 57% of new PV installations in 2015 were in the utility. Extrapolation of this data shows that there is a growing market in which our product fits in the U.S. and especially in the state of California. The timing of our product will fall during a timeframe in which the cost of utility-scale solar photovoltaics is falling. According to [31], by 2020 the goal price of electricity will be around 6 cents per kWh. This is down from referenced 21.4 cents per kWh in 2010 as shown in Figure 14 [31]. We placing the product with favorable timing, we aim to take advantage of the opportunities of lower projected PV costs and increase demand of photovoltaics.

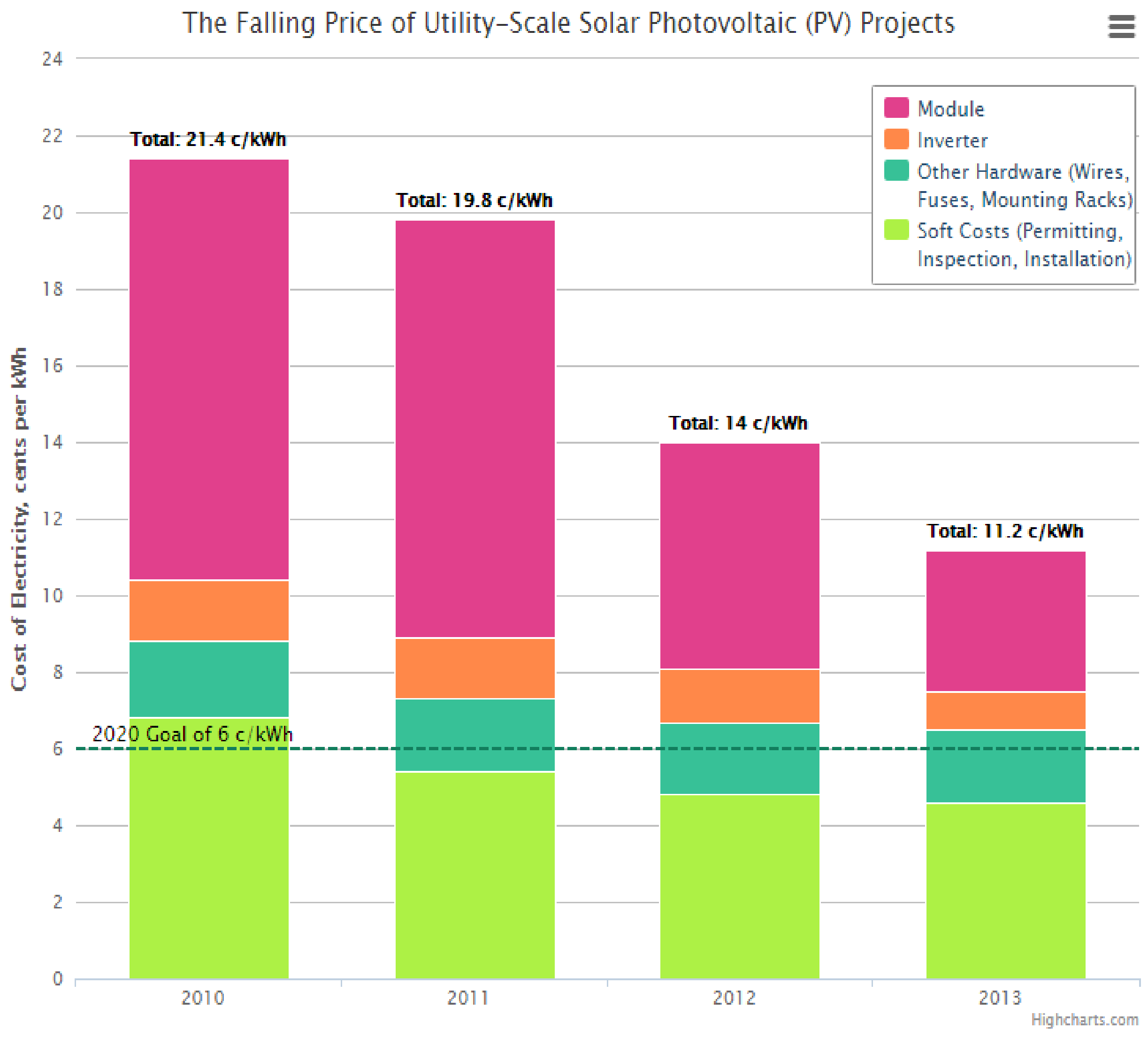

Fig. 14. Falling Price of Utility-Scale Solar Photovoltaic Projects *[31]*

E. Integration Challenges

Challenges that this product faces when entering the market come two-fold. The first challenge that we face is the reliability of the system with respect to time. This represents variable performance when the system is placed in changing location, events, weather, and/or seasons. Scenarios such as systems located in darker latitudes would be challenging and less effective than ones in sunnier regions. Thus, a larger system may be required in order to stabilize the reliability and only to a certain extent. The reality is there will be times in which solar will not be sufficient source of power alone and until technology advances, may only be supplementary to conventional power.

The second of the two challenges in which our product will be met is a lack in large-scale (utility scale) storage. With a varying degree of energy from photovoltaics, there is a need for more efficient or at least larger capacity energy storage. A standalone system will depend more heavily on energy storage when irradiance is less available. However, emerging technologies such as Tesla Powerwall and Powerpack (commercial and utility) shows promising innovation for the future of energy storage [32].

F. Final Forecast

There is a very large amount of resources dedicated towards research in renewable energy sources every year. The charter of this project is to adapt workloads that are not typically adaptable to renewable energy sources. The project does this through the use of efficient hardware and software that controls the efficiency and power consumption of a device that was not typically controllable. The technologies that we have begun to develop are a couple of years ahead of their time, and will become more and more important, the cheaper that PV electricity generation becomes.

The target consumer for this project is the company that does not have teams large enough to create a renewable energy datacenter on their own. Developing countries provide greater opportunity than any other region. In developing countries, the power grid may not be particularly mature, and as it develops it can be developed with renewable energy in mind. PV electricity generation is economical in these regions because it is one of the cheapest sources of electricity. The TAM (total available market) for renewable energy management technology will be rapidly growing in the coming decade.

In order to develop a larger MSS (market share segment) in renewable energy data centers, the unique properties of this solution must be demonstrated. Project Parasol has many similar properties to the project, but the biggest differentiating factors revolve around the fact that this project purely uses DC, and the power consumption on this project’s servers can be modulated. Broader society's dependence on renewable energy is rapidly increasing as the years go by. This project must evolve over the years to adapt to future technologies that are very quickly being developed. A large part of marketing is developing clear use cases of the product. The pace of innovation with renewable energy is fast, and we must demonstrate the value of our product.

There are many challenges to implementing our product. The current TAM for this product is limited but it will grow with time as solar panels become more cost effective and efficient. The fastest way to grow the MSS in the TAM is by keeping the ASP (average selling price) down, and justify the cost of the product by focusing on the ROI (return on investment) of this product.

XI. Conclusion

With a significant portion of the world’s energy demands still being met by a dwindling and finite resource of fossil fuels, it is on us as engineers to be proactive in finding solutions to this issue. Since there are many things that need power to run, as a group our senior design team wanted to narrow the scope from the broad issue into something we could really tackle. With the influx of cloud computing and more device having internet connectivity, the datacenter was a good area to focus our attention. Large server farms take up vast amounts of energy and are essential to the world we live in today. Our first idea was to supplement the energy consumption of datacenters with renewable energy. This process has already been started by major companies around the world like, Apple, Google, Facebook, and Amazon. Looking farther into the future just supplementing dome of the energy demands with renewable energies is not enough. We as a group needed to think bigger, which lead us to the idea of a fully self-sustaining renewable energy server model. Our idea was to have a series of solar panels in conjunction with a chemical storage device charge a server running batch processing for a twenty four hour period. To accomplish this we wanted to measure power production and consumption along with estimated battery life. These measurements were essential for the feature of self-sustaining. This was done by automatically changing the CPU utilization of the server to match the estimated life of the battery. With the design idea laid out budget realistic for our project to be able to achieve the features that we set for our system. We easily came in under budget because we were allowed to use solar arrays that belonged to the school. After this we broke down the workload into different sections for team members to tackle based on their skill set. This allowed us to tackle our design in the most efficient manner utilizing the different strengths of our team members. Zach and Arik were responsible for the hardware side consisting of voltage regulation of all powered components, proper and safe charging of the battery, and the measuring of power production & consumption, and estimated battery life. Taylor and Ryan were in charge of the software consisting of batch processing run by the server, CPU utilization control, and displaying of important data.

The hardware for this project was responsible for providing power to the server by only using solar energy from our solar panels. To do this we had to create a way to efficiently get the maximum power out of the panels to the server and a way to efficiently store energy for when the solar panels are unable to provide energy to the server. To get the maximum power out of our solar panels we set out to design a maximum power point tracking circuit to adjust the voltage of the panel in order to get the maximum power. This changing of voltages allows us to manipulate the current flowing from the panels into the battery and the connected devices. We then used an Arduino Nano to measure these voltages and currents to see what the power in and power out is over the max power point circuit. This allows us to see how much power the panels are providing and how much power the load will consume. These measurements are vital because the server needs to know how much power it has to utilize from the panels in order to correctly adjust the load it puts on the system. Also to make this system fully self-contained we needed to power our Arduino microcontroller and our Gigabyte off of our energy storage device, which in our case was a 12V deep cycle battery. To do this we used the LM317 voltage regulating circuit to bring the voltage down to an appropriate level that connected device could handle. Having these devices all powered by energy collected from the solar panel allows us to have a fully self-sustaining hardware system.

The Software part of the system encompasses a very large portion of our project, and many of the primary functions of the project are entirely reliant on it. The software utilizes the C programming language for all low level items. The power measurement controller uses C to collect the data off of it sensors, and convert it to units of power. The data is then sent to the load control server over USB using a serial data connection. The load control server then interprets this data to determine the desired CPU utilization of the server. Once the desired utilization is determined on the load control server, a UDP network packet is sent to the server so that the server side script can adjust its CPU utilization and in turn adjust the power consumption of the server. The server is using the Python programming language to handle all communications and high level interactions. The Python script creates four separate threads. One for managing the throttling of the process in the operating system scheduler. It does through the interfacing of an exposed kernel module from the Linux operating system. The throttling of the process is sufficient for reducing power consumption because Intel SpeedStep technology will reduce the frequency of the processor when the utilization is reduced. The second Python thread calls a git C repository for the Sieve of Eratosthenes. The software calls as many worker threads as possible to work on the sieve to calculate prime numbers. The software is benchmarked by the number of prime numbers found over the course of a 24 hours period. Any other software that is running on the server besides these worker threads, is considered overhead. The third and fourth thread provide full duplex support to the server and the load control server. The third thread receives data from the load control server to change the target CPU utilization. The fourth thread on the server sends data to the load control server every time 100 million digits are processed, this equates to about every 10 seconds. The load control server, which is a Gigabyte with a Linux OS, runs a Python script which utilizes a main thread and three child threads. The main thread spawns the three child threads then waits to receive prime numbers calculated from the server. The first child thread creates the serial connection with the power measurement controller and updates global variables when it receives. The second thread uses a thirty second counter then sends a signal to idle the server down for three seconds. This is done so that a measurement of the battery voltage at a known load can be taken and used to decide the new CPU utilization of the server. The final child thread sends all important measurements to a database for future analysis. This thread also sends power production, power consumption, battery output voltage, and calculated prime numbers to a web GUI that graph the measurements in real time.

After many different iterations of our maximum power point tracking circuit we still could not get it to work properly. Our problem in every iteration we came up with was that our FET's or FET drivers would break down. Through much of our testing we were under the impression that we were sending to much current in the components. With this hypothesis we spent a significant amount of time testing current throughout the circuit while also trying to add more safety precautions based around overcurrent. Finally at the request of our advisor Professor Tatro we ran simulations of our circuit in PSPICE and notices that the inductor was producing large amounts of voltage for very short periods of time that was breaking down the FET's and drivers. Professor Tatro instructed us to research snubber circuits for the inductor. By this time we had burned out all our FET drivers and could not get new ones ordered and delivered in time for further testing before the due date. We had to scrap our designed maximum power point tracker circuit and use our purchased solar charge controller for our project. While the maximum power point tracker was not a part of our feature set for the project it was an option we were very hopeful of accomplishing.

For the software portion of the project we had more success. We were able to model a batch processing server with the Sieve of Eratosthenes calculating prime numbers on the server. Our power measurement controller was successfully collecting data from our voltage and current sensor to find power production and consumption along with the output voltage on the battery. This data was then sent to the load control server would decide the CPU utilization of the server based on our estimated battery life. The load control server also housed our web GUI that allowed us to see real time graphs of our data. The load control server also housed our database that will help us do further analysis to help make our product even more efficient.

In forecasting the marketability of our project we decided that there was two markets we felt we could be successful in; green conscience enterprises and developing countries. We specified green conscience enterprises because power in developed countries is so reliable that we would need someone who cares about their carbon footprint to want to utilize our product. For developing countries we believed that because of their unreliable power sources they would be more interested in utilizing out system. We also looked at the future opportunities to enhance our product like utilizing a storage device like the Tesla Powerwall for greater energy storage capacity.

All in all we are very proud of the product that we put together. While we were not able to design a successful MPPT circuit ourselves, we do know the root of our trouble and believe we can fix this issue in a future iteration of the system. The proliferation of using renewable energy sources is extremely important to us and believe our product shows that self-sustaining renewable energy ran datacenters do not need to just be a plan for the future but instead a product for the present.

References

[1] S. H. Schneider, "Uncertainty and CLimate Change Policy: A Survey," Island Press, Washington D.C., 2002.

[2] "The World Factbook," CIA, [Online]. Available: https://www.cia.gov/library/publications/the-world-factbook/geos/xx.html. [Accessed September 2016].

[3] T. Appenzeller, "The High Cost of Cheap Coal," National Geographic, March 2006. [Online]. Available: http://ngm.nationalgeographic.com/2006/03/cheap-coal/appenzeller-text.html. [Accessed September 2016].

[4] A. Q. Huang, M. L. Crow, G. T. Heydt, J. P. Zheng and S. J. Dale, "The Future Renewable Electric Energy Delivery and Management (FREEDM) System: The Energy Internet," *Proceedings of IEEE,* vol. 99, no. 1, pp. 133-148, 2011.

[5] "What are the major sources and users of energy in the United States?," U.S. Energy Information Administration, 29 December 2015. [Online]. Available: http://www.eia.gov/energy_in_brief/article/major_energy_sources_and_users.cfm. [Accessed September 2016].

[6] M. Z. Jacobson, "A roadmap for repowering California for all purposes with wind, water, and sunlight," *Energy,* vol. 73, pp. 875-889, 2014.

[7] "How Long Does It Take to Pay Off a Tesla Powerwall?," Institute for Energy Research, 5 January 2016. [Online]. Available: http://instituteforenergyresearch.org/analysis/payback-on-teslas-powerwall-battery/. [Accessed September 2016].

[8] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo and W. Lintner, "United States Data Center Energy Usage Report," Ernest Orlando Lawerence Berkeley National Laboratory, Berkeley, 2016.

[9] Y. Sverdlik, *Here's How Much Energy All US Data Centers Consume,* Data Center Knowledge, 2016.

[10] R. Miller, "Emerson Looks to a Solar Future," Data Center Knowledge, 21 July 2009. [Online]. Available: http://www.datacenterknowledge.com/archives/2009/07/21/emerson-looks-to-a-solar-future/. [Accessed October 2016].

[11] R. P. Sherwood, *Estimated Budget,* Sacramento, 2016.

[12] R. P. Sherwood, *Actual Budget,* Sacramento, 2016.

[13] R. P. Sherwood, *Task Assignment and Hours,* Sacramento, 2016.

[14] "Ready," Department of Homeland Securty, [Online]. Available: https://www.ready.gov/risk-assessment. [Accessed 28 October 2016].

[15] Z. Mietz, *Risk Assesment Matrix.*

[16] Z. Mietz, *Solar Array,* Sacramento, 2017.

[17] Z. Mietz and A. Cheng, *Buck Converter Experiment.*

[18] Z. Mietz and A. Cheng, *Buck Boost Circuit.*

[19] Z. Mietz, *Original Solar Charge Controller Circuit,* Sacramento, 2017.

[20] Z. Mietz, *Solar Charge Coltroller with Snubber Circuit,* Sacramento, 2017.

[21] Z. Mietz, *PSPICE Simulation of Original Charge Controller,* Sacramento, 2017.

[22] Z. Mietz, *PSPICE Simulation of Charge Controller with Snubber Circuit,* Sacramento, 2017.

[23] A. Cheng, *3D Modeled Component Case,* Sacramento, 2017.

[24] A. Cheng, *Complete Component Case,* Sacramento, 2017.

[25] T. James, *Load Algorithm Performance Data,* Sacramento, 2017.

[26] R. Sherwood, *Target CPU Alogorthm Test Results,* Sacramento, 2017.

[27] A. Leach, "Race to renewable: five developing countries ditching fossil fuels," The Guardian, 15 September 2015. [Online]. Available: https://www.theguardian.com/global-development-professionals-network/2015/sep/15/five-developing-countries-ditching-fossil-fuels-china-india-costa-rica-afghanistan-albania. [Accessed 20 February 2017].

[28] L. Morias, "Costa Rica beats own record," Renewables Now, 13 August 2015. [Online]. Available: https://renewablesnow.com/news/costa-rica-beats-own-record-relies-solely-on-renewables-for-94-days-488370/. [Accessed 20 February 2017].

[29] I. Goiri, W. Katsak, K. Le, T. D. Nguyen and R. Bianchini, "Rutgers," March 2013. [Online]. Available: https://www.cs.rutgers.edu/~ricardob/papers/top-picks14.pdf. [Accessed 17 February 2017].

[30] A. Hobson, "In its Largest Quarter Ever, U.S. Solar Market Saw Nearly 2 MW of PV Installed Per Hour in Q3 2016," Solar Energy Industries Association, December 2016. [Online]. Available: http://www.seia.org/news/its-largest-quarter-ever-us-solar-market-saw-nearly-2-mw-pv-installed-hour-q3-2016. [Accessed 15 February 2017].

[31] "Progrss Report: Advancing Solar Energy Acroos America," U.S. Department of Energy, 12 February 2014. [Online]. Available: https://energy.gov/articles/progress-report-advancing-solar-energy-across-america. [Accessed 20 February 2017].

[32] "Tesla Energy," Tesla, 2017. [Online]. Available: https://www.tesla.com/energy. [Accessed February 21 2017].

[33] *LM317 Datasheet,* January: Semiconductor Component Industries, 2016.

[34] R. Sherwood, *Load Control Server Flow Diagram,* Sacramento, 2017.

[35] T. James, *Flow Chart for Load Algorithm,* Sacramento, 2017.

[36] T. James, *Flow Chart for the CPU Throttling Process on the Server,* Sacramento, 2017.

[37] T. James, *Flow Chart for the Network Receiving Thread,* Sacramento, 2017.

[38] T. James, *Flow Chart for the Network Sending Thread,* Sacramento, 2016.

[39] A. Cheng, *Back View Component Enclosure,* Sacramento, 2017.

[40] A. Cheng, *Front View of Component Enclosure,* Sacramento, 2017.

Glossary

MPPT – Maximum Power Point Tracking, optimizes power output of the solar energy source

USB – Universal Serial Bus is an IEEE certified serial interface for connecting peripheral devices to computers

GUI – Graphical User Interface, is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators

UDP – User Datagram Protocol is an Internet protocol suite used primarily for establishing low-latency and loss tolerating connections between applications

PID – Process Identification Number is a number used the kernel of an Operating System to uniquely identify an active process

VRM – Voltage Regulator Module, is a buck converter that provides a microprocessor with its appropriate supply voltage

CentOS – Is a Linux based Operating System

TAM – Total Available Market, used to reference the revenue opportunity available for a product of service

MSS – Market Share Segment, a company's share of a particular market segment

ASP – Average Selling Price, average price at which a particular product or commodity is sold across channels or markets

ROI – Return On Investment, the benefit to an investor resulting from an investment of some resource

APPENDIX A.
USER MANUAL

1. Start by flipping the switch just on outside of battery enclosure to the ON position.
2. Next turn on server and load control server. Wait approximately a minute for the units to boot up and connect.
3. Next connect into the router by Ethernet cable with your desired device, i.e. Laptop or Desktop computer.
4. To access the web GUI, in browser of choice type in the IP address 192.168.0.15 and it will come up
5. To retrieve the measurements stored in the database, in browser of choice type 192.168.0.15/phpmyadmin. Username root and password is password1. This tool will allow you to do database queries of all sorts.

APPENDIX B
HARDWARE

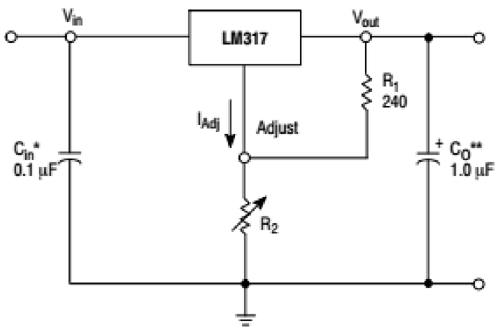

Fig. 15. Voltage Regulator Circuit *[33]*

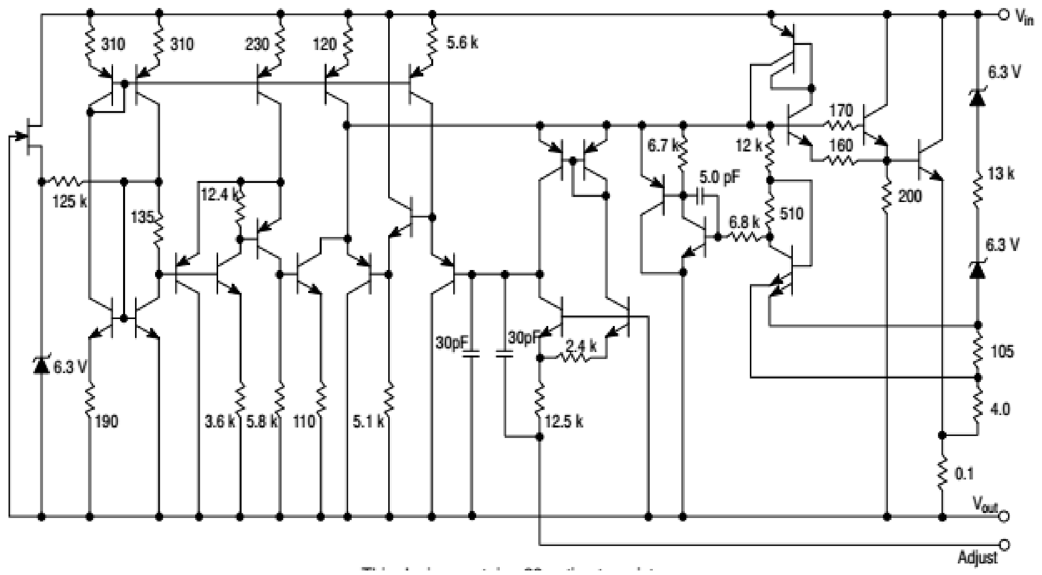

Fig. 16. LM317 Schematic Diagram *[33]*

APPENDIX C
SOFTWARE

A. Block Diagrams and Flow Charts

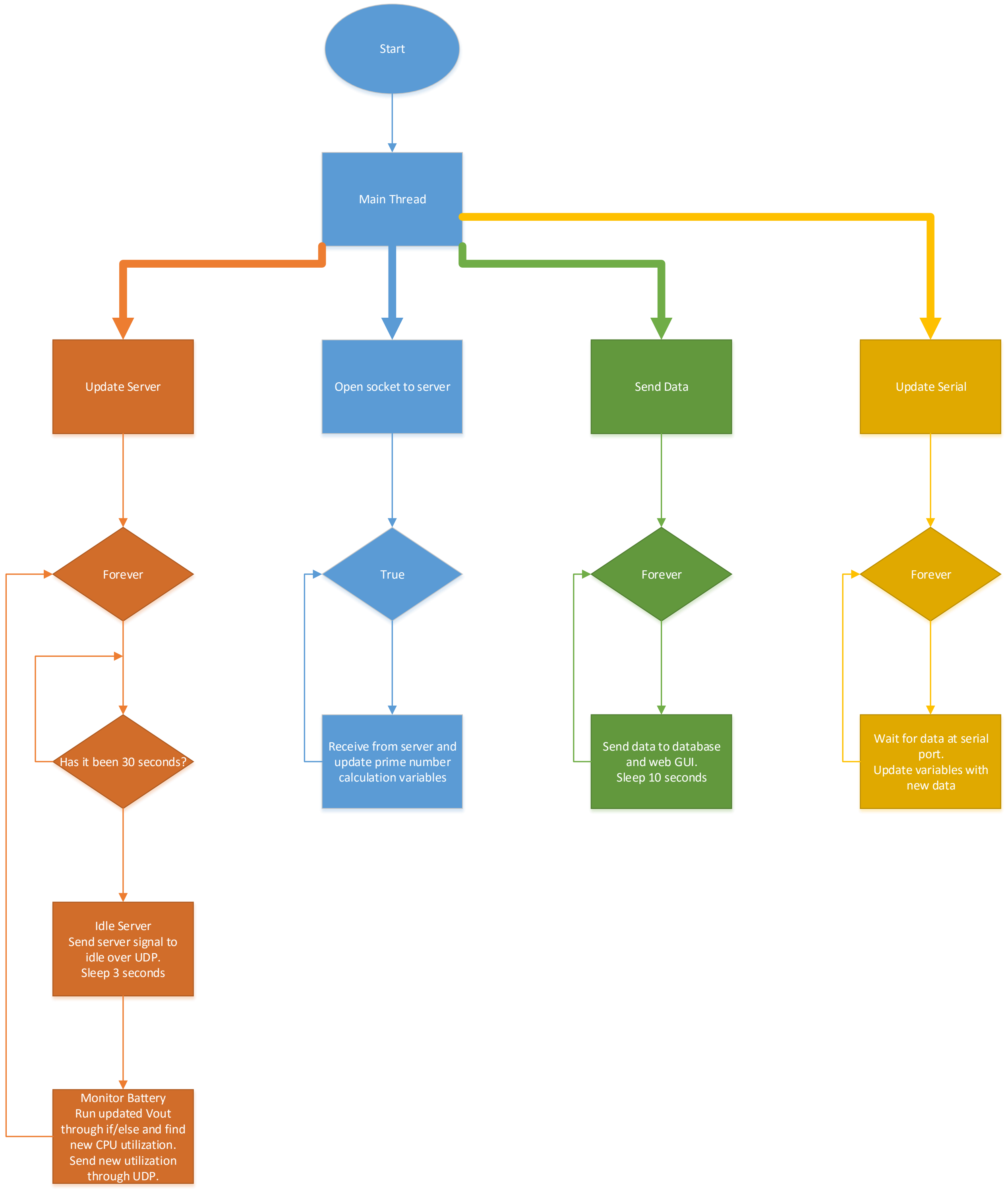
Start
Main Thread
Update Server
Open socket to server
Send Data
Update Serial
Forever
True
Forever
Forever
Has it been 30 seconds?
Receive from server and update prime number calculation variables
Send data to database and web GUI. Sleep 10 seconds
Wait for data at serial port. Update variables with new data
Idle Server Send server signal to idle over UDP. Sleep 3 seconds
Monitor Battery Run updated Vout through if/else and find new CPU utilization. Send new utilization through UDP.

Fig. 17. Load Control Server Flow Diagram *[34]*

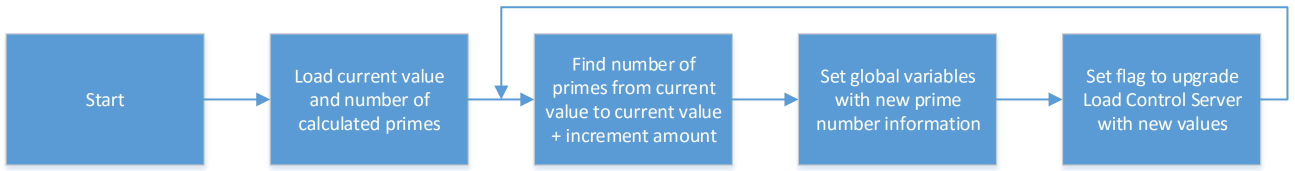
Start
Load current value and number of calculated primes
Find number of primes from current value to current value + increment amount
Set global variables with new prime number information
Set flag to upgrade Load Control Server with new values

Fig. 18. Flow Chart for Load Algorithm *[35]*

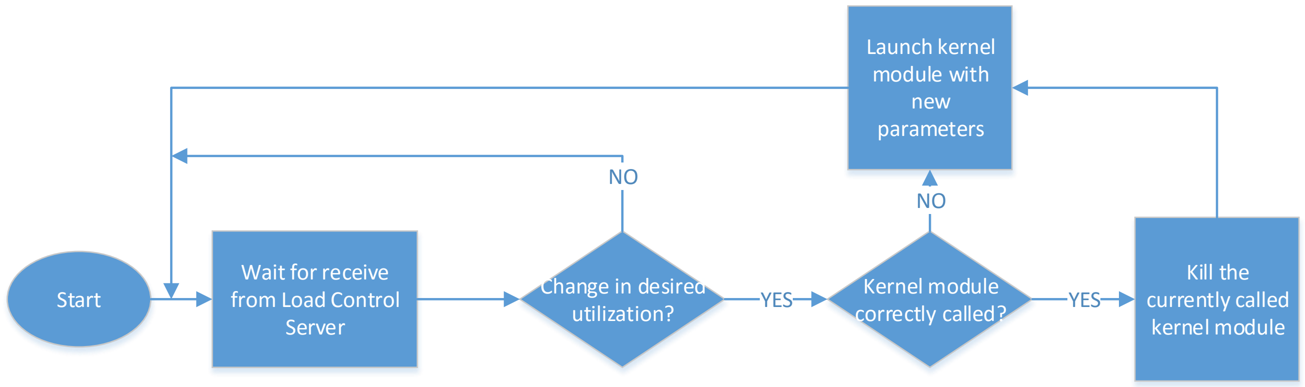
Start
Wait for receive from Load Control Server
Change in desired utilization?
NO
YES
Kernel module correctly called?
NO
YES
Launch kernel module with new parameters
Kill the currently called kernel module

Fig. 19. Flow Chart for the CPU Throttling Process on the Server *[36]*

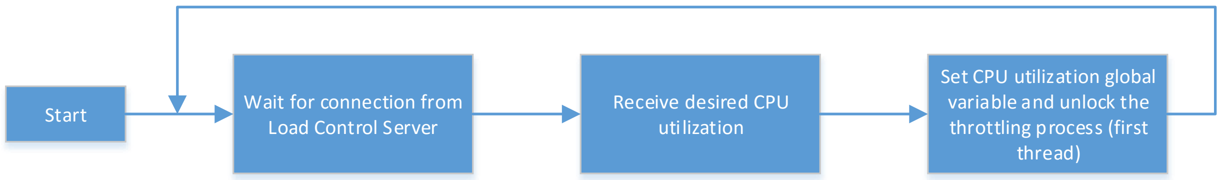
Start
Wait for connection from Load Control Server
Receive desired CPU utilization
Set CPU utilization global variable and unlock the throttling process (first thread)

Fig. 20. Flow Chart for the Network Receiving Thread *[37]*

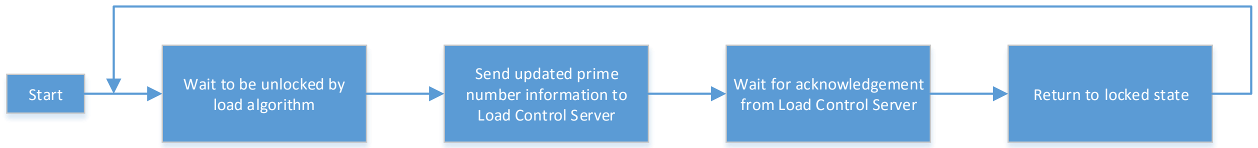
Start
Wait to be unlocked by load algorithm
Send updated prime number information to Load Control Server
Wait for acknowledgement from Load Control Server
Return to locked state

Fig. 21. Flow Chart for the Network Sending Thread *[38]*

B. The Sieve of Erathosthenes

```
from libc.stdint cimport uint64_t, int64_t
cimport cpp_primesieve

cpdef uint64_t parallel_nth_prime(int64_t n, uint64_t start = 0) except +:
    """Find the nth prime after start using multi-threading"""
    return cpp_primesieve.parallel_nth_prime(n, start)

cpdef uint64_t parallel_count_primes(uint64_t a, uint64_t b = 0) except +:
    """Count prime numbers using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_primes(a, b)
```

```
cpdef uint64_t parallel_count_twins(uint64_t a, uint64_t b = 0) except +:
    """Count twin primes using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_twins(a, b)

cpdef uint64_t parallel_count_triplets(uint64_t a, uint64_t b = 0) except +:
    """Count prime triplets using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_triplets(a, b)

cpdef uint64_t parallel_count_quadruplets(uint64_t a, uint64_t b = 0) except +:
    """Count prime quadruplets using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_quadruplets(a, b)

cpdef uint64_t parallel_count_quintuplets(uint64_t a, uint64_t b = 0) except +:
    """Count prime quintuplets using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_quintuplets(a, b)

cpdef uint64_t parallel_count_sextuplets(uint64_t a, uint64_t b = 0) except +:
    """Count prime sextuplets using multi-threading"""
    if b == 0:
        (a,b) = (0,a)
    return cpp_primesieve.parallel_count_sextuplets(a, b)

cpdef void print_primes(uint64_t a, uint64_t b = 0) except +:
    """Print prime numbers to stdout"""
    if b == 0:
        (a,b) = (0,a)
    cpp_primesieve.print_primes(a, b)
```

C. *Load Control Server*

```
#Author: Ryan Sherwood and Taylor James
#Load Control Server
#This script has two main goals; base server target utilization on
#measurements received from the microcontroller and sending desired
#information to the database and web GUI.

import serial
import socket
import cPickle as pickle
import threading
from time import sleep, time, strftime
import sys
import os
import MySQLdb
import json

PIN = 0.0
POUT = 0.0
VOUT = 0.0
VOUT_tmp = 0.0
CPU_TARGET = 100
TOTAL_PRIMES = 0
```

```
TOTAL_NUMS = 0


#Create function to shutdown server. Called when battery
#is to low
def shutdownServer():
    try:
        send_dict = {"shutdown":"yes"}
        send_data = pickle.dumps(send_dict,-1)
        sendsocket = socket.socket()
        sendsocket.settimeout(.2)
        sendsocket.connect(('192.168.0.10', 11293))#double check IP
        sendsocket.send(send_data)
    except socket.timeout:
        pass

#Create function to bring server to idle, wait, then
#call monitorBattery which will tell updateServer what
#CPU_TARGET should be based on VOUT
def idleServer():
    try:
        send_dict = {"cpu_throttle":0}
        send_data = pickle.dumps(send_dict,-1)
        sendsocket = socket.socket()
        sendsocket.settimeout(.2)
        sendsocket.connect(('192.168.0.10', 11293))#double check IP
        sendsocket.send(send_data)
        sleep(3)
        monitorBattery()
    except socket.timeout:
        pass

#Function called from monitorBattery which will tell the server
#what the new CPU_TARGET should be
def updateServer():
    global CPU_TARGET
    startTime = time()

    while 1:
        sleep(1)
        if (time()-startTime > 30):
            print("eval")
            startTime = time()
            idleServer();
            try:
                send_dict = {"cpu_throttle":CPU_TARGET}
                send_data = pickle.dumps(send_dict,-1)
                sendsocket = socket.socket()
                sendsocket.settimeout(.2)
```

```
                    sendsocket.connect(('192.168.0.10', 11293))#double check IP
                    sendsocket.send(send_data)
                except socket.timeout:
                    pass

#Function sends measurements to database and web GUI.
def sendData():
    global PIN
    global POUT
    global VOUT
    global CPU_TARGET
    global TOTAL_PRIMES

    try:
        connection = MySQLdb.connect (host = "localhost",
                                user = "root",
                                passwd = "password1",
                                db = "lcs")
        cursor = connection.cursor()
    except MySQLdb.Error, e:
        print(e)

    while 1:
        try:
            cursor.execute ("""INSERT INTO power VALUES (%s,%s,%s,%s,%s,%s)""",
         (strftime('%Y-%m-%d
%H:%M:%S'),PIN,POUT,VOUT,CPU_TARGET,TOTAL_PRIMES))
            connection.commit()
        except MySQLdb.Error, e:
            print(e)
            connection.rollback()

        ajax_dict = {'PIN':PIN, 'POUT':POUT, 'VOUT':VOUT,
'CPU_TARGET':CPU_TARGET, 'TOTAL_PRIMES':TOTAL_PRIMES}
        json.dump(ajax_dict, open('/var/www/html/ajax/data.json', 'wb'))

        sleep(10)


#Function monitors battery voltage and saves the new CPU target
#utilization value to the global variable
def monitorBattery():
    global VOUT
    global CPU_TARGET

    if VOUT >= 12.3:
        #keep server at 100%
        CPU_TARGET = 100
    elif 12.3 > VOUT >= 12.2:
        #drop server to 90%
        CPU_TARGET = 90
    elif 12.2 > VOUT >= 12.15:
        #drop server to 80%
        CPU_TARGET = 80
    elif 12.15 > VOUT >= 12.1:
        #drop server to 70%
        CPU_TARGET = 70
    elif 12.1 > VOUT >= 12.05:
        #drop server to 60%
        CPU_TARGET = 60
    elif 12.05 > VOUT >= 12.0:
        #drop server to 50%
        CPU_TARGET = 50
    elif 12.0 > VOUT >= 11.95:
        #drop server to 40%
        CPU_TARGET = 40
    elif 11.95 > VOUT >= 11.90:
        #drop server to 30%
        CPU_TARGET = 30
    elif 11.9 > VOUT >= 11.85:
        #drop server to 20%
        CPU_TARGET = 20
    elif 11.85 > VOUT >= 11.8:
        #drop server to 10%
        CPU_TARGET = 10
    elif 11.8 > VOUT:
        #drop server to 0%
        CPU_TARGET = 0
    else:
        #shutdown
        #shutdownServer()
        CPU_TARGET = 0
        pass

#Function reads the serial input from the microcontroller
#and updates the corresponding global variables
def updateSerial():
    global VOUT
    global PIN
    global POUT

    VOUT_tmp = [0 for x in range(10)]
    PIN_tmp = [0 for x in range(10)]
    POUT_tmp = [0 for x in range(10)]
    i = 0
    try:
        ser = serial.Serial('/dev/ttyACM0', 9600)
        while 1:
            data_in = str(ser.readline())
            print(data_in)
            power_data = data_in.split(',')
            VOUT_tmp[i%10] = float(power_data[0])
            PIN_tmp[i%10] = int(power_data[1])
            POUT_tmp[i%10] = int(power_data[2])
            i=i+1;
            VOUT = sum(VOUT_tmp)/len(VOUT_tmp)
            PIN = sum(PIN_tmp)/len(PIN_tmp)
            POUT = sum(POUT_tmp)/len(POUT_tmp)
            print("VOUT: %0.1f, PIN: %0.1f, POUT %0.1f" % (VOUT,PIN,POUT))

except serial.SerialException:
    print("Unable to connect to serial")
    sleep(5)
    updateSerial()

#Main function starts the threads for updating the server, sending data
#to the database and web GUI, and receiving updated data from the
#microcontroller. Then sets up the UDP connection to the server and
#receives the updated calculated prime numbers.
if __name__ == '__main__':
update_server = threading.Thread(target=updateServer)
update_server.start()

update_db = threading.Thread(target=sendData)
update_db.start()

update_serial = threading.Thread(target=updateSerial)
update_serial.start()

s = socket.socket()
port = 11294
s.bind(('', port))
s.listen(1)

try:
    while True:
        c, addr = s.accept()
        buf = c.recv(1024)
        dict = pickle.loads(buf)
        TOTAL_PRIMES = int(dict["total_primes"])
        TOTAL_NUMS = int(dict["curr_val"])

        c.close()

except KeyboardInterrupt:
    sys.exit()
print(e)
```

D. Power Measurement Controller

```
#include <PWM.h>

float panelAMeter = 0;
float loadAMeter =0;
float panelMeter = 0;
float batteryMeter = 0;
float loadMeter = 0;

float currentMath = 5/1024.0;
float arduinoMath = 1;

float currentPOffset = 2.5;
float currentLOffset = 2.5;

float panelAMath = 1;
float batteryAMath = 1;
float loadAMath = 1;
float panelMath = 0.0351;
float batteryMath = 0.0354;
float loadMath = 0.0351;
float panelAmpMath = 15.79455204;
float loadAmpMath = 15.15762;

float panelCurrent = 0;
float batteryCurrent = 0;
float loadCurrent = 0;
float panelVolts = 0;
float batteryVolts = 0;
float loadVolts = 0;

float panelPower = 0;
float loadPower = 0;
float test = 0;

void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A5, INPUT);
  pinMode(A4, INPUT);
  pinMode(A6, INPUT);
}
void loop(){

  measure();
  print_data();

}
```

```
void measure(){

    test = analogRead(A5);
     panelVolts = analogRead(A4);
     batteryVolts = analogRead(A5);
     loadVolts += analogRead(A6);

    panelCurrent = 0;
    for(int i=0;i<100;i++){
       panelCurrent += analogRead(A1);
    }

     loadCurrent = 0;
    for(int i=0;i<100;i++){
     loadCurrent += analogRead(A2);

    panelVolts = panelVolts/100;
    batteryVolts = batteryVolts/100;
    loadVolts = loadVolts/100;
    panelCurrent = panelCurrent/100;
    loadCurrent = loadCurrent/100;

    panelVolts = (panelVolts * arduinoMath) * panelMath;
    batteryVolts = (batteryVolts * arduinoMath) * batteryMath;
    loadVolts = (loadVolts * arduinoMath) * loadMath;

    panelCurrent = ((panelCurrent * currentMath) - currentPOffset) * panelAmpMath;
    loadCurrent = ((loadCurrent * currentMath) - currentLOffset) * loadAmpMath;

    panelPower = panelVolts * panelCurrent;
    loadPower = loadVolts * loadCurrent;
    }
}

void print_data() {
/*
  int Pin = (int) panelPower;
  int Pout = (int) loadPower;
  Serial.print(batteryVolts);
  Serial.print(",");
  Serial.print(Pin);
  Serial.print(",");
  Serial.println(Pout);
  */

  Serial.print("Panel: ");
  Serial.print(panelVolts);
  Serial.print("\t");
  Serial.print("Battery: ");
  Serial.print(batteryVolts);
  Serial.print("\t");
  Serial.print("Load: ");
  Serial.print(loadMeter);
  Serial.println("\t");
/*
  Serial.print("Panel: ");
  Serial.print(panelCurrent);
  Serial.print("\t");
  Serial.print("Load: ");
  Serial.print(loadCurrent);
  Serial.println("\t");
*/
}
```

Appendix D
Mechanical

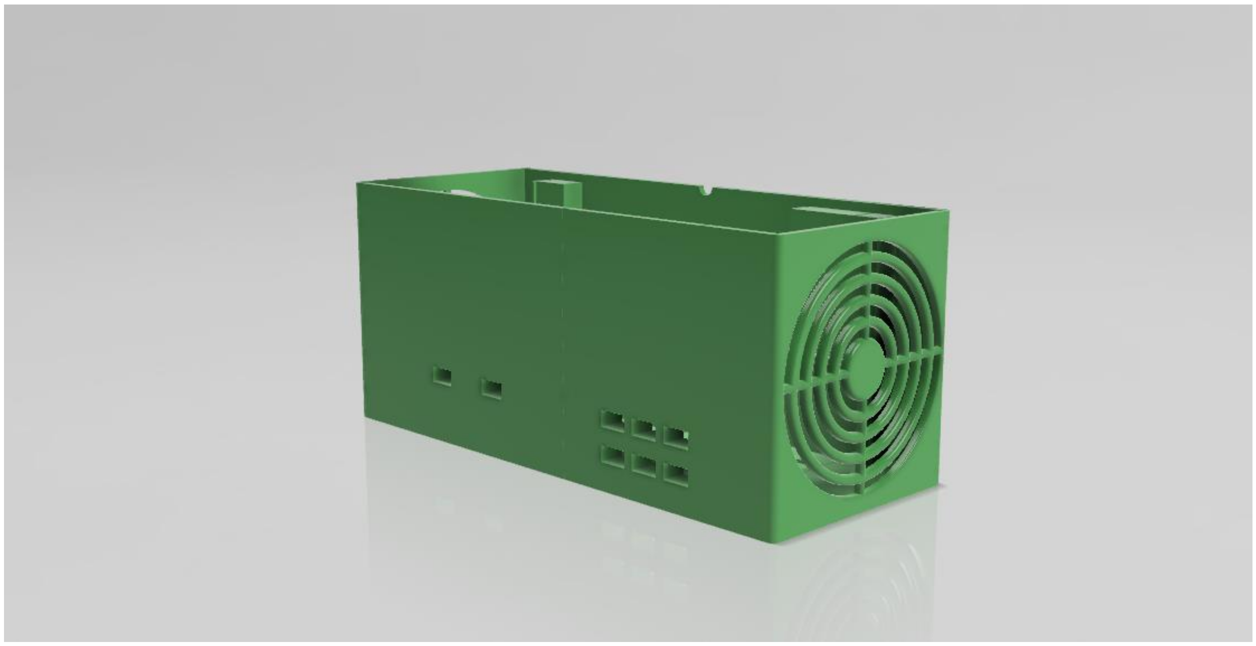

Fig 22. Back View Component Enclosure *[39]*

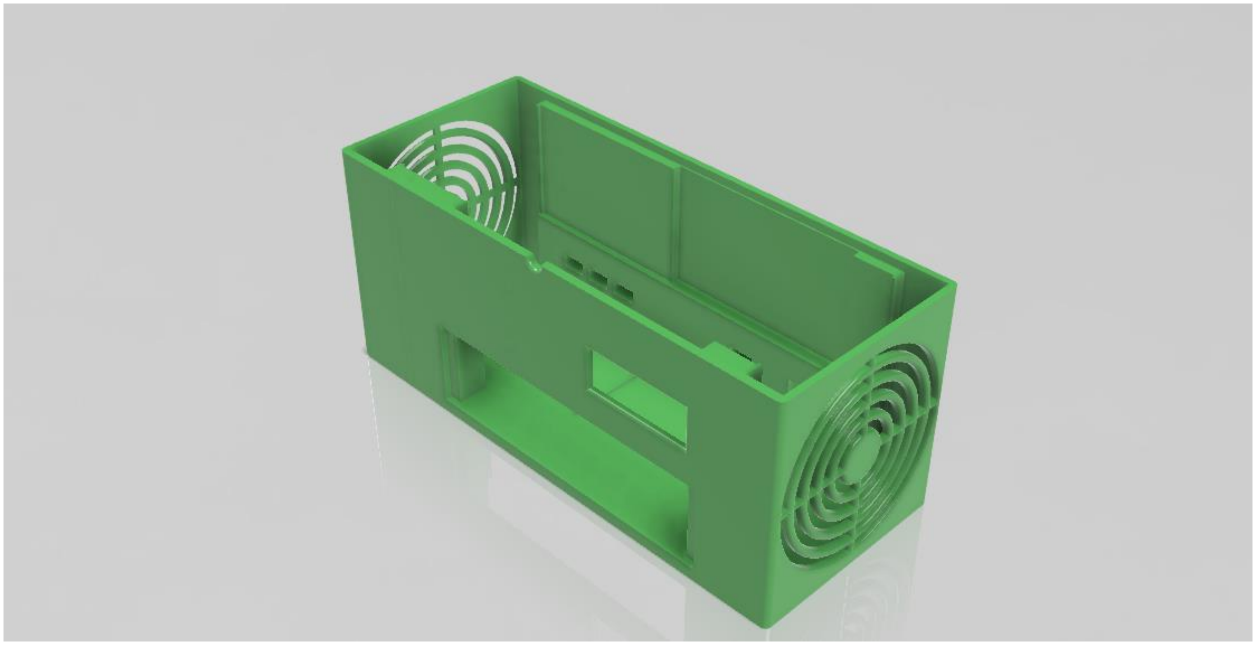

Fig. 23. Front View Component Enclosure *[40]*

Appendix E
Vendor Contacts

We had no outside vendors help us with this project.

Appendix F
Resumes

Ryan Sherwood

Experience

Intel Corporation (Software Solutions Group) — **Folsom, CA**
Hardware Modeling & Simulation Intern — *June 2016-Present*

- Modeled both pipelined and sequential iterations of Intel's GPU Texture Sampler
- Obtained 16x simulation speedup with POSIX generated models of pipelined units

Aerospace Corporation — **El Segundo, CA**
Metrology Representative — *November 2008-July 2013*

- Performed compliance audits on engineering laboratories each month to insure government standards were met
- Worked with scientists and technicians to manage inventory of over 100 types of scientific equipment totaling over 6,000 units

United States Coast Guard — **San Diego, CA**
Electronics Technician — *February 2002-February 2006*

- Maintained, debugged, and repaired ships radar, navigational, and communication equipment for use in Drug Interdiction and Search & Rescue
- Specialized in 5 different types of radar and navigational equipment
- Lead a team of 15 colleagues responsible for emergency protocols on ship while in port

Projects

Renewable Energy Self Sustaining Server Model

- Group senior design project where we designed a single server to run for a 24 hour period on power generated from 2 solar arrays and a battery backup
- Designed and implemented the server load controller with algorithm to control the server utilization based on the voltage measurements solar array and battery, while sending all data to a database for monitoring

RISC Pipelined CPU

- Designed, modeled, simulated and verified a 16-bit RISC CPU in Verilog that can decode and execute instructions in a pipelined datapath
- Design considerations included Hazard Detection, Branching, Register Forwarding, and Exception handling

Vending Machine Controller

- Designed, modeled and verified the CMOS layout mask on the 0.24 micron process for a Vending Machine Controller in Cadence Virtuoso

Education

Sacramento State University — **Sacramento, CA**
Bachelor of Science 3.3 GPA – Deans Honor List — *Graduating May 2017*

- Major: Computer Engineering

Skills

Computer Languages

- C – C++ – Python – Java – x86 Assembly – Ruby

Hardware Descriptive Languages

- Verilog – VHDL

Tools

- Cofluent – Quartus – Synopsys VCS – Synopsys DVE– VIM – BASH– PSpice – Multism – Cadence Virtuoso – Perforce – Git – MS Visual Studio – MS Office

ARIK CHENG

SUMMARY OF QUALIFICATIONS:

- Working knowledge of MatLab, PSpice, AutoCAD, Quartus II, VMware, and Advanced Design Systems.
- Strong advanced computer background, including C++ and structured programming knowledge.
- Friendly, approachable and energetic worker with self motivation and positive attitude.
- Initiative to perform multiple tasks and work efficiently under stress and pressure.
- Systematic, flexible, tolerant, organized, detail-oriented, and goal-directed.

EDUCATION AND TRAINING:

Bachelor Degree Program – Electrical Engineering (To Complete May, 2017)
University of California, Sacramento – Sacramento, CA
Associates Degree Program – Mathematics/Physical Sciences (Completed *May, 2014*)
Associates Degree Program – Mathematics Transfer, Behavioral Sciences (Completed *May, 2015*)
American River College - Sacramento, CA
National Pharmacy Technician Certification (*Completed 2015*)
Pharmacy Technician Certification Board - Washington, DC
California Pharmacy Technician License (*Completed 2015*)
Board of Pharmacy – Sacramento, CA

PROFESSIONAL EXPERIENCE:

Electrical Engineering Student

- Junior level electrical and electronics engineering student with emphasis on power and energy.
- Currently holds officer position for CSUS Power and Energy Society.
- Keep records of events and meetings as well and significantly contribute to club events.
- Theoretical and experimental knowledge of Verilog and transistor level logic design.
- Apply educational knowledge of AC/DC circuit analysis, electromechanical conversion, power system analysis, microprocessors, signals and systems, and transmission line analysis.

Pharmacy/Medication Technician

- Apply profession knowledge of medication and pharmacy law.
- Administer medication to residents as well as assist with tasks of daily living.
- Specialize in interactions with residents with varying degrees of dementia.
- Frequently recognized for high level of expertise in senior care environments. Customer Service/ Management Skills
- Greet customers and ascertain personalized customer wants or needs.
- Maintain knowledge of current sales and promotions, policies regarding payment and exchanges, and security practices.
- Assist management with daily operations.
- Assist in training new staff to ensure that applicants are qualified to perform their delegated tasks.
- Monitor and recognize potential security risks and thefts, and know how to prevent or handle these situations.

WORK HISTORY:

Certified Pharmacy Technician	CVS Pharmacy, Lincoln, CA	*(06/14-Present)*
Medication Technician	Atria Senior Living, Carmichael, CA	*(04/13-06/14), (05/11-01/13)*
Medication Technician	Somerford Place, Roseville, CA	*(12/12-04/13)*
Customer Service Associate	JCPenny's, Roseville, CA	*(10/10-07/11)*
Manager in Training	Little Caesars, Antelope, CA	*(10/08-10/10)*

TAYLOR JAMES

WEB DEVELOPER/PROGRAMMER

Profile

I'm a web resource professional and a Computer Engineering major at Sacramento State. While only 23 years old, I have had my own business designing and managing websites for the past 6 years and developing PHP applications for the past 4 years. I am creative in not only tuning my skills to adapt to the ever demanding
and
changing field, but also constantly searching out better and more efficient ways to get the job done. I am self-taught in the web developing discipline and this passion drives me to look for new challenges and push myself past web developing and further into the future of my field.

Skills

Programming/Languages:
Python(Qt), Java, PHP (eCommerce, Custom Systems), C (Systems Programming,Visual Studio), MySQL, JavaScript (JQuery), AutoIt, Assembly, HTML/CSS,

Education:
Algorithms & Data Structures, Network & Internet (LAN, WAN), Signal & Systems, Electronics (Basic), Operating System Principals. MS Office, Solidworks

Hardware Description:
Verilog (VCS Design Tools), VHDL, Computer Organization & Data Pathing, Advanced Logic Design

DevOps:
CentOS/RHEL (Programming/CLI), SELinux, Windows Server (Basic), Vagrant, GIT

Experience

Intel Corporation 2015-Current

Technical Marketing Engineer Intern - Channel Desktop i5+

My job role included verifying hardware problems that exist with intel customers, expanding channel customer sales through platform enablement, and technical development of marketing materials. Beyond this I learned about Intel architecture and internal systems in depth.

Tjames Web Design 2008-2014

Freelance Web Designer and Developer

This business started very young with my basic web design skills. Over the years I have worked with numerous companies and have extended my skillset to include backend web development, Python apps and Linux server management. Running this business has consistently been a challenge to best fit my client's unique needs and has developed my problem solving ability.

More information: tjameswebdesign.com

Zach Mietz

Objective Seeking an internship in the Electrical Engineering field to further enhance my knowledge of applicable skills in the field

Education *In progress:* BS in Electrical Engineering, CSU Sacramento, May 2017

Related Courses

- Electronics I
- Electronics II*
- Introduction to Feedback Systems
- Circuit Analysis
- Network Analysis
- Robotics Explorations
- Fundamentals of Engineering
- Computational Methods & Apps
- Introduction to Logic Design
- Power Systems Analysis
- Introduction to Microprocessors
- Electromechanical Conversions
- Transmission Lines and Fields
- Signal and System Analysis
- Interdisciplinary Topics of Engineering
- Personal Computing
- Algorithm Design/Problem Solving
- Introduction to Computer Science
- Senior Design*
- Modern Communication Systems*

Skills

- **Programing Language:** C, C++, Assembly Language
- **Software Tools:** Microsoft Visual Studio, Microsoft Office, Autodesk Inventor
- **Operating Systems:** Windows, Mac OS

Employment History

Cashier Raley's Supermarket, Galt, CA 8/14-9/16

- Provide high quality customer service both in person and over the phone
- Deal with customer inquiries and complaints
- Process sales payments
- Assist with product display and pricing
- Restock ordered product
- Take payment in exchange of items sold
- Balance money in cash register with sales data
- Order merchandise for store sales

Courtesy Clerk Raley's Supermarket, Galt CA 11/11-8/14

- Provide high quality customer service both in person and over the phone
- Greet customers and assist them with item location
- Bag groceries and other items
- Carry packed bags to customers' vehicles
- Return grocery items left at checkout to their rightful places
- Collect shopping carts and baskets and deliver them to their proper location
- Maintain parking lot cleanliness when required