# SMART RENEWABLE ENERGY ALLOCATION AND CONSUMPTION IN AN OFF-GRID VERTICAL FARMING SYSTEM

by

Otto C. P. F. Randolph

A thesis submitted to the Graduate College of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
August 2020

Committee Members:

Bahram Asiabanpour, Chair

Heping Chen

Damian Valles

## FAIR USE AND AUTHOR'S PERMISSION STATEMENT

### Fair Use



### Duplication Permission

## DEDICATION

To my parents, for being there my whole life, to my friend Blake for his constant support and encouragement, and to all of the grandmas in my life, biological or adopted, past and present.

## ACKNOWLEDGEMENTS

This thesis would not have been possible without the constant support of the faculty and staff of the Ingram School of Engineering. I wish to expressly thank my thesis advisor, Dr. Asiabanpour, for the many hours he spent helping me to craft and write my thesis. Without his help this would have never been possible. Dr. Chen and Dr. Valles, my committee members, who gave insight into my project as well as Dr. Viswanathan the graduate. Many thanks to the departmental administrators, Ms. Torres, Ms. Rivas, and Ms. Batey, who were always gracious in answering my many questions. Finally, many thanks to Mr. Summers for being genuinely friendly and the first hand advice he gave.

I also wish to thank all of my coworkers on the EverGreen project. I could never have done the coding portions of this project without the hours of work put in by Jacob Pangonas, my partner in this project. His knowledge of computers and coding is the only reason the physical system was able to run. I also wish to thank Stephanie Lopez and the early work she put into this project. Special thanks to Alex Little, Riley Horner, Genisis Segundo, Daniel Ceballos, and Danielle Cortez. Their help, encouragement, and insights were very valuable and I value the friendships I made with them in working on this project. Thanks also to everyone else on the Evergreen project, and the countless hours of work they put in that helped make this possible.

Finally, many thanks are due to all of the family and friends who helped me along the way who were my foundation during this project. I am grateful to my mom and dad who spent hours proofreading rough drafts. I appreciate my friend, Blake, for his encouragement throughout the whole process. Many thanks to Joan

Shirley and the Bible study ladies. I am honored to have been adopted by the Bangladeshi community at Texas State that adopted me, especially Fatema, Orthy, Sazida. Finally, sincere thanks to everyone else who has touched my life over the past two years for there have been too many to count.

## TABLE OF CONTENTS

**Page**

## LIST OF TABLES

## LIST OF FIGURES

## ABSTRACT

Current electrical generation techniques are designed for fossil-fueled power plants and do not lend themselves for implementation with renewable solar energy sources as they rely on proper weather conditions. Vertical farming systems are an innovative farming technique that could solve world food shortages and improve food security by consistently producing food. However, they are generally energy intensive. Using off-grid renewable energy sources makes these systems more affordable and provides the flexibility to place them in areas that lack utilities and infrastructure. The challenge for off-grid renewable energy generation systems is their performance correlation with weather conditions and lack of consistency in energy generation. This paper proposes a smart system that combines renewable energy generation and vertical farming in an off-grid system. Inspired by a plant's natural ability to adapt to its surroundings, energy usage is tailored to fit the power production schedule of a renewable energy source. The system allocates energy and prioritizes the activation of subsystems and instruments based on their overall priority as well as the current and future needs of the system. A theoretical system was successfully modelled off of historical weather data. The maximum number of plants that could grow each season was calculated using a correlation between weather conditions and energy generation. These results were then used to create a physical system that successfully adapted to changing weather conditions without causing a power outage. Finally, it was proven that excess energy could be stored in the form of a thermal battery in the system.



Keywords-Vertical farming, renewable energy, off-grid, smart systems

# 1. INTRODUCTION

## Background

Current electrical generation techniques are unsustainable because they try to make supply match demand. Fossil fuel power plants are excellent at matching supply to meet the demand because they only consume fuel when they are in use (Martinez-Anido et al., 2016; van der Meer, Widén, & Munkhammar, 2018). Renewable energy sources do not work this way. Their "fuel" comes from natural sources, such as the wind and sun, and if it is not used immediately, the power is lost. This makes it virtually impossible for an electrical grid to entirely switch over to renewable energy sources and maintain a stable power supply (Khastieva, Dimoulkas, & Amelin, 2018). The two major solutions to this are impractical.

One solution, found in most off-grid homes, relies on a vast oversupply of electricity. This is wasteful and prohibitively expensive (Nassereddine, Rizk, Nagrial, & Hellany, 2018). The other solution relies on fossil fuel-powered backup generators to make up for any energy shortages (Morrow, 2019). Not only does this defeat the purpose of having a renewable energy grid, but these generators are also much less efficient than the power plants they replace (Martinez-Anido et al., 2016).

Virtually all renewable energy generation systems assume that demand is fixed. These systems then adjust supply to meet demand. This is true for many residential and industrial applications with fixed schedules. However, for applications such as vertical farming, there is room for flexibility because, in nature, plants survive, adapt, and grow in a wide range of weather conditions.

Currently, electricity costs keep vertical farms from being a viable alternative to traditional agriculture (Sarkar & Majumder, 2019). The benefit of a vertical farm is that it reduces pollution by using less water and pesticides than a traditional farm (Elsokah & Sakah, 2019; Labrador et al., 2019). These gains are canceled out

by the pollution created in generating the electricity used (Gentry, 2019). Plants thrive in a wide array of conditions and are not bound to a fixed schedule. Researchers have exploited this adaptability by tailoring their systems to run using energy that would typically be wasted (Gentry, 2019).

Inspired by the way that plants survive in variable weather conditions, this research proposes a smart, alternative solution. Instead of enforcing adherence to a strict power schedule, the system runs in sync with the power generated by solar panels. Unlike most systems where the power supply is adjusted to meet a fixed demand, the maximum supply capacity is treated as a fixed parameter, and the demand is altered to meet said supply. Machine learning algorithms predict the amount of power available to the system each day based on historical trends, current time, and weather conditions. Once the power production estimates are completed, a control algorithm decides how to allocate the available power based on the state of charge of the batteries, time of day, and the needs of the plants.

The EverGreen project and lab, is a testbed to implement and assess fully off-grid vertical farming operations utilizing different levels of automation (EverGreen, 2019). A 40 ft shipping container was used as the experimental and validation instrument for the vertical farming system, employing a hydroponic system. Water for the system came from multiple sources, including rainwater harvesting and atmospheric water generation (AWG) (Almusaied & Asiabanpour, 2017; Asiabanpour, Ownby, Summers, & Moghimi, 2019; Moghimi, Ghoddusi, Asiabanpour, & Behroozikhah, 2019). This system would serve as a model for more efficient food production and as a proof of concept for the benefits of an Industry 4.0 environment. In the future, fully automated smart factories could help balance out the grid and provide power stability. Working in sync with electric utilities, they could alter their energy usage based on power availability, increasing their power demands during oversupplies and reducing them when the grid is overloaded.

## Prior Work

Prior work comes from three areas of research, smart manufacturing, solar energy production, and vertical farms. Table 1.1 gives a summary of the different methodologies.

Table 1.1: Research Areas and Focuses.

| Research Area | Smart Manufacturing | Solar Energy | Vertical Farms |
|---|---|---|---|
| Focus 1 | Origin of Industry 4.0 | Prediction Models | Efficiency Improvements |
| Focus 2 | Data Management | Short-Term Forecasting | Automation |
| Focus 3 | Resource Allocation | Long-Term Forecasting | Off-Grid Systems |
| Focus 4 | Quality Control | Evaluating Model Errors | Industry 4.0 Applications |
| Focus 5 | Future Trends | Economic Analysis | |
| Focus 6 | | Energy Storage | |
| Focus 7 | | Control Algorithms | |

### Smart Manufacturing

Smart manufacturing, also known as Industry 4.0, is less an area of research and more an avenue of thinking. To understand this, one must go back to 2011, where the phrase was coined (Kagermann, Lukas, & Wahlster, 2011). In it, researchers touted Industry 4.0 as the next industrial revolution, one that had not yet happened. This concept exploded in popularity, but it took until 2016 before a consensus could be reached about what Industry 4.0 was (Hermann, Pentek, & Otto, 2016). Once that is understood, the principles of Industry 4.0 research, such as data management, allocation of resources, and quality control, can easily be applied to the EverGreen project. Finally, what the EverGreen project could look like in the future can be found in the direction of current Industry 4.0 research.

## Origin of Industry 4.0

Smart manufacturing, or Industry 4.0, is the next industrial revolution following the computer revolution of the early 1970s. Unlike other industrial revolutions, where the key attributes that led to industrialization were analyzed after the fact, Industry 4.0 was predicted in 2011 by individuals in industry, politics, and academia (Kagermann et al., 2011). The term has become muddied since then as people attempt to define this new revolution. Hermann et al., in their paper, Design Principles for Industrie 4.0, attempt to remedy this by sifting through the myriad of articles written about Industry 4.0 (Hermann et al., 2016). They found that Industry 4.0 is a revolution in communication. Instead of a centralized top-down manufacturing structure, where a single company plans a product's design, manufacture, and the logistics to get it to a customer, Industry 4.0 heralds a decentralized approach. In it, a customer orders a product, and the product controls how it is manufactured as well as logistics to the next machine and, ultimately, to the customer. Businesses are merely there to provide the factories used for production. They find this revolution is defined by greater interconnectivity of all steps of the manufacturing process (Hermann et al., 2016).

## Data Management

One of the struggles in Industry 4.0 is what to do with the tremendous amount of data these interconnected machines generate. Centrally controlling every device creates economies of scale when it comes to machine learning. This improved efficiency comes at the cost of speed and reliability. If the server is overwhelmed with data or breaks down, the whole system could go offline. Separating each machine into its island of autonomy would remedy this, but loses the interconnected benefits that Industry 4.0 is striving for. Villalonga et al. propose a different

approach. Their system is similar to the latter approach, where each machine in a system works autonomously (Villalonga, Beruvides, Castaño, & Haber, 2018). What makes their approach unique, is each machine in the system periodically sends its data to a central server that analyses it. The server uses various machine learning techniques to improve the control algorithm by analyzing the data. These improvements are then sent to each machine in the system (Villalonga et al., 2018). L. Zhang proposes a different approach.

L. Zhang chooses to organize the data presented by an Industry 4.0 factory in a way comparable to a USB port (L. Zhang, 2018). USB ports revolutionized computing because one port could perform a wide array of tasks. Similarly, Zhang's system incorporates a wide variety of subsystems without extensive modifications. To do this, Zhang generalizes subsystems into broad categories and creates a set of protocol standards to talk to the main system. While this generalization essentially creates a robust plug and play system for smart manufacturing, it does reduce the efficiency of the whole because it is harder to optimize the individual parts (L. Zhang, 2018). A third solution is similar, but proposes the adaptation of a blockchain-based trust mechanism.

One significant drain on computing power between systems is verifying the authenticity of the data traveling between the two. This puts unnecessary strain on smaller companies doing business in a smart manufacturing system. Blockchain could be a solution. If implemented, it could act as a third-party trust mechanism for the small and midsize suppliers who lack the industrial clout to form their credentials (Y. Zhang et al., 2019). Widespread adoption of blockchain technology would reduce the computing power needed and standardize data flow (Y. Zhang et al., 2019). Other researchers take this technology a step further.

The amount of data generated by the industry is skyrocketing as manufacturers prepare for an Industry 4.0 world. This data is often sent to

powerful, centralized servers to be analyzed. Once analyzed, the results are then sent back to the factory, and any new instructions are implemented. As the volume of data increases, latency becomes an issue, especially for time-sensitive calculations. Multi-access edge computing (MEC) could be an answer (Lee, Huo, Zhang, & Ng, 2020). MEC works by decentralizing the computing power of a system. This reduces latency by reducing the distance information must travel before it is analyzed. Combined with blockchain technology, it could revolutionize current smart manufacturing methods by reducing latency issues and lowering the amount of calculations required to determine trust in the system (Lee et al., 2020).

### Allocation of Resources

One problem facing smart manufacturing is the allocation of digital resources. The major resources are bandwidth, and processing power. When working in such a diverse environment as an Internet of Things (IoT), how does a computer decide which things are most important? One solution is an auction-style resource assignment. Each machine lists its computing needs, and a value is assigned based on items like the required computing power it will take to complete the task, task importance, and deadline to complete the process. These values are submitted as a bid. The highest bid wins the auction and is completed by the computer. Problems arise when machines, programmed by unscrupulous programmers, artificially inflate their bid (Jeong, Na, Kim, & Cho, 2018). This gives their tasks a higher priority (Jeong et al., 2018). To counteract this, Jeong et al. proposed a sealed bid style system where competing machines cannot see each other's bids. While machines can still inflate their bid, winning the auction is no longer a guarantee (Jeong et al., 2018). A second option was demonstrated by Angelakis et. al.

Instead of auctioning off resources to the highest bidder, Angelakis et al. places programs in tiers of importance (Angelakis, Avgouleas, Pappas, Fitzgerald, &

Yuan, 2016). From here, their solution splits into two different options. In option one, the program that takes the longest to run is chosen first. In option two, programs are given shares or tickets based on their tier of importance. Computer time is then raffled off. The most essential programs have the most cards and are more likely to be chosen than lower-tier programs (Angelakis et al., 2016). This reduces the amount of processing power spent allocating resources and improves performance. Additionally, the program's importance is hardcoded into the algorithm, so the computers cannot cheat to get more processing time. Another problem facing Smart Manufacturing systems is quality control.

## Quality Control

One of the goals of smart manufacturing is increased automation. This speeds up assembly lines and removes errors caused by tired or negligent workers. The problem is quality control. Commonly, industrial vision systems do this job (A. Ismail, Idris, Ayub, & Por, 2018). Current methods work by finding differences in what should be runs of identical parts and throwing out the product that does not conform to the pattern. In an Industry 4.0 world, every part coming down the assembly line could be unique. Additionally, systems that could sort out defective parts in this environment would require a disproportionate amount of computing time.

Shin and Park propose creating a set of guidelines for creating a quality control system in an Industry 4.0 factory (Shin & Park, 2019). When designing a product for a customer, the first step is taking in the customer requirements. Next, these customer requirements are transformed into a series of quantifiable specifications that can be measured. Shin and Park do this too, but take this process a step further. Once the process parameters are defined, their next step is to choose a process quality control strategy. They outline several strategies, each

building in complexity from the prior one. They range from simple industrial vision systems to a multi-factory network utilizing machine learning and a complex array of sensors and controllers (Shin & Park, 2019). Their system is then applied to a steel mill to improve the quality of the materials produced (Shin & Park, 2019).

Another solution combines cutting edge vision software with machine learning techniques (A. Ismail et al., 2018). Ismail et al. found that Spatial Pyramid Matching (SPM) combined with Support Vector Machine (SVM) could correctly identify defects 98% of the time in only 0.13 seconds on average (A. Ismail et al., 2018). The next best method, Convolutional Neural Network (CNN) with SVM, correctly identified defects 94% of the time in 0.08 seconds (A. Ismail et al., 2018). While this second method was faster, a 0.05 second reduction in time was inconsequential compared to the 4% increase in overall accuracy (A. Ismail et al., 2018).

## The Future of Smart Manufacturing

Tao and Qi took a step into the future at what smart manufacturing would mean for the industry (Tao & Qi, 2019). What they see is a shift in building into a service-oriented sector. This happens as a result of the increased flexibility smart manufacturing provides. Traditionally, consumers rely on large businesses to create large production runs of affordable products and work out the logistics to make them accessible to them. Smart manufacturing inherently provides a much more flexible factory layout that can accommodate small production runs, solving the first issue. Additionally, the interconnectivity of Industry 4.0 means that the logistics side of the equation can be automatically resolved through a cloud-based computing system (Tao & Qi, 2019). Others look even further.

The current trend is speculation on what Industry 4.0 will look like, and ways to get there. Ferrer et al. go further. They see beyond the fully automated and

interconnected factories of Industry 4.0 and see a Cyber-Physical System of Systems (CPSoS) (Ferrer et al., 2018). Like Smart Manufacturing, a CPSoS tracks and connects all consumption elements, from the extraction of natural resources to the purchase by a consumer. A CPSoS uses this data to improve efficiency by predicting demand (Ferrer et al., 2018). Industry 4.0 is reactionary at its heart, but a CPSoS would predict demand (Ferrer et al., 2018).

Finally, as the concept of smart manufacturing has matured, the overall industry has begun to shift. This is most apparent in the electronic products manufacturing industry. When smart manufacturing began, most manufacturers developed their technologies and standards. Now the individual manufacturers are working together to create a single industry standard (Chatterjee & Gamota, 2020). This improves efficiency and will allow multiple devices to connect much more easily than before (Chatterjee & Gamota, 2020).

### Conclusion and Application

Industry 4.0 is a term for a wide range of developing technologies with the potential to revolutionize the manufacturing world. While these technologies will be impactful in many ways, their concepts can be directly applied to the EverGreen system. Automating the EverGreen system will generate a tremendous amount of data. The data management techniques discussed let us compare several commercially proven methodologies and determine the best one. Because the system is relatively simple, it was determined that the Villalonga approach, where subsystems act relatively autonomously but receive periodic updates from a central controller, would be best (Villalonga et al., 2018).

In the EverGreen system, the main resource being allocated is power. Lights, pumps, and fans use fixed amounts of power when they are on, and the air-conditioning system uses a variable amount. Because the relative importance of

these subsystems was known, as well as their demands, the system will allocate power similarly to the way Angelakis et al. proposed. Each subsystem is given a tier of importance, and then power is allocated down each tier (Angelakis et al., 2016).

Quality control for the EverGreen system was more challenging to quantify (EverGreen, 2019). Plant growth is neither uniform nor constant, and the plant varieties will vary from season to season. One solution that could work though, is that proposed by Ismail et. al. Their solution combines machine learning with industrial vision systems to characterize a process (A. Ismail et al., 2018). In a future iteration, plants could be monitored by a static webcam. Generalized aspects of the plants could be monitored, and internal shipping container conditions adjusted based on these observations.

Finally, the future of Smart Manufacturing systems showed what the EverGreen system could evolve into. As the technology matures, increased standardization could allow multiple systems from different manufacturers to communicate with each other, while blockchain technology will improve the trust between these disparate systems (Chatterjee & Gamota, 2020). Industry 4.0 logistics improvements could reduce shipping costs to get plants to market or allow independent growers to work together (Tao & Qi, 2019). Finally, if there was widespread cooperation, the cyber-physical system of systems could be developed. This would help growers predict demand and plant crops most desired by consumers for each season or make up for shortfalls expected in traditional farming operations (Ferrer et al., 2018).

### Solar Energy Production

Solar energy research formed the basis for how energy was used in the system. Practical examples gave insight into how to set up the base system. Cities, such as Los Angeles, spend vast sums attempting to improve their power infrastructure and

make it "green" through renewable energy (Morrow, 2019). Another area of research is that of solar energy production prediction. Significant research has been done documenting the main factors affecting solar energy production (Das et al., 2018). Furthermore, research in applying machine learning principles to solar power production prediction gave valuable insight into which methods to look into for this application (Martinez-Anido et al., 2016). Finally, articles about energy usage and storage were valuable for creating the basic control algorithm (Khastieva et al., 2018; Walrath, 2010).

Because of the vast amounts of research done in this field, four focal points were chosen when researching solar energy production. The four focal points were energy prediction, economic analysis, energy storage, and control algorithms. The findings of this research are as follows.

### Prediction Models

Many weather forecasting models have been adapted for use with solar power stations to estimate the power production on a given day. Pedro et al. identify five main forecasting techniques for solar energy production (Pedro & Coimbra, 2012). Of the five main techniques, Auto-Regressive Integrated Moving Average (ARIMA), Artificial Neural Networks (ANNs), ANNs optimized by Genetic Algorithms. k-Nearest Neighbors (KNNs), and the Persistent Model, Pedro found that ANNs provided the most accurate results, but each has its own merits. Furthermore, the available techniques can be subdivided into two categories, short and long-term forecasting. More standardization is needed in evaluating errors in models.

### Short-Term Forecasting

Accurate short-term predictions of solar panel power production are critical for a reliable power grid. In many countries, power production figures must be

locked in 45 minutes to 1 hour before the period begins. Overestimation of solar power output leads to brownouts, whereas overproduction wipes out profits. Wolff et al. hope to improve these predictions using support vector regression modeling (SVM) (Wolff, Kühnert, Lorenz, Kramer, & Heinemann, 2016). SVM is a "learning" algorithm that relies on fewer data points and adapts itself over time. While less accurate over longer periods, their initial attempts show that SVM could be more accurate in the critical short-term prediction field.

Another short-term forecasting software proposed by Voyant et al. uses machine learning (Voyant et al., 2017). They show that machine learning is a viable means of interpreting data and applying it in data forecasting. The main drawbacks are its sensitivity to initial error. If the original data given has a bias, the resulting algorithm will also be biased. But this bias is reduced as the amount of initial data is increased.

### Long-Term Forecasting

Accurate short-term forecasts are critical for solar production during the day, but long-term profitability research into extended forecasting periods is vital. A wider time window of 24 hours would give energy producers ample time to operate backup power sources and reduce solar power costs. One of the most promising methods was proposed by Massidda and Morrocu in their paper about Multilinear Adaptive Regression Splines (MARS) (Massidda & Marrocu, 2017). The MARS model works in two steps. First, variables are added into a model using regression until the model is within a set error of the data, or the variables are exhausted. Next, the model removes the variables that increase the accuracy the least, resulting in the best sub-model. Using data from the Global Forecasting System (GFS) and the National Oceanic and Atmospheric Administration (NOAA), the author creates an improved model for solar power production using a MARS model. This increases

the prediction accuracy of solar panels in the 24-hour range and reduces the stress on the national power grid.

Another promising method is proposed by Pierro et al. (Pierro et al., 2017). This method focuses on specific regional input data based on satellite images and weather prediction software. Because computer power and satellite time are expensive, this method focuses on collecting data from strategic points. In this case, large solar panel installations. Accurate weather data is generated for these points and extrapolated for the region to create a regional weather forecast. This reduces the required computing power and reduces the error generated by up to 50%.

### Evaluating Model Errors

While load forecasting is an established science that routinely has prediction errors of less than 3%, this is not the case with solar power forecasting (van der Meer et al., 2018). Larson et al. consider it an accomplishment that their method routinely was within 13-23% of solar output. They only underestimated their power output by >40% twice in three years (Larson, Nonnenmacher, & Coimbra, 2016). Meer et al. believe that the best way to improve forecasting models is to standardize the error methods used to evaluate them (van der Meer et al., 2018). Currently, more than a dozen error methods are used, and the favored method is generally whatever makes an author's model look the best. While the forecasting period determines which errors are most important, Voyant et al. agrees that the number of error methods used can and should be reduced (Voyant et al., 2017).

### Economic Analysis

Finding an improved forecasting method has significant financial benefits. Martinez-Anido et al. performed an economic analysis of solar power production and the effects of improved solar forecasting (Martinez-Anido et al., 2016). They

found that improved forecasting has a tangible impact in the market, but that it suffered from diminishing returns past a 50% accurate forecast average. As prediction accuracy increases, the number of inefficient fast start plants required for daytime use decreases (natural gas, internal combustion, etc.). Where diminishing returns sets in is nighttime power generation. Most high-efficiency plants need several hours to ramp-up to full capacity. Therefore, these plants are started while solar panels are still producing adequate light, creating a glut of excess electricity. Meer et al. alsp mentions that there are significant government incentives for plants that improve solar forecasting accuracy (van der Meer et al., 2018).

Another problem affecting solar panels is their cost and reliability. Renewable energy sources are known for their high costs, and as these systems age, their efficiency goes down. This is especially a problem in tropical regions. The tropics seem like an ideal place for a solar farm due to their long, bright sunny days and their consistent number of light hours. This is not the case, though. The high heat and humidity combined with low wind speed put these panels to the test and cause them to fail quickly (Ogbomo, Amalu, Ekere, & Olagbegi, 2017). Therefore, researchers are using machine learning algorithms to predict when solar panels will fail based on the performance ratio of the individual cells (Bandong, Leksono, Purwarianti, & Joelianto, 2019). This prediction can then be used to schedule preventative maintenance on the system to improve stability.

### Energy Storage

Renewable energy is not a steady form of energy production. Wind and solar both produce power sporadically throughout the day based on environmental conditions. These excesses are currently absorbed into the grid through the powering up and down of traditional power plants. Still, this practice is untenable if a fully renewable power grid is desired. Khastieva et al. propose bulk energy storage

as a potential solution to this predicament (Khastieva et al., 2018). Energy storage decouples renewable energy from the grid by introducing power storage centers, allowing it to be used later. The best storage option depends mainly on the physical location of the reserve, so Khastevia et al. look at the problem from a different perspective. They outline two mathematical models, a centralized and decentralized version, that provides a guideline for when and where to place energy storage centers (Khastieva et al., 2018).

### Control Algorithms

For the EverGreen project, an ideal control algorithm would look at the problem from the demand side of the equation because the available supply is fixed. While some articles mentioned financial incentives for customers to adjust their usage, this was not useful for the project. Instead, a solution came from the world of batteries. Walrath patented an algorithm that slowly turns off devices based on a user-specified battery life preference (Walrath, 2010). If the device is plugged in, or there is enough battery life left, the system behaves normally. If the device is on battery and the user-specified battery life is less than available, non-essential subsystems are powered off until the battery life target is met. Because backup power is a finite resource, this would reduce the likelihood of a catastrophic complete power system failure in the EverGreen system.

## Vertical Farming

While many researchers have done studies on vertical farming systems, four areas of research were focused upon. These were efficiency improvements, automation, off-grid or renewable energy powered systems, and Industry 4.0 applications. Efficiency improvements were focused upon because energy is the limiting factor in the EverGreen system. Automation efforts gave insights into the

tools available to researchers automating their systems and their efficacy. Off-grid and renewable energy systems were applicable because the EverGreen system is powered by renewable energy. Finally, Industry 4.0 applications were a narrow subset of papers that focused on both smart manufacturing methods and vertical farming.

### Efficiency Improvements

One problem with current agricultural practices is that food is not grown close to the consumer. One researcher found that, on average, food must travel 2400 km before reaching its final destination (Gentry, 2019). These transportation costs generate large amounts of emissions. Vertical farms could solve this problem, but their costs are prohibitive. In cold weather countries, one of the major costs for these systems is heating them. Gentry proposes heating these systems using waste heat from power plants (Gentry, 2019). Not only is this energy free, but it could help offset some of the carbon emissions generated by the power plant (Gentry, 2019).

Another group of researchers tried a different approach. Ideally, food should be grown close to where it will be consumed. This improves freshness, reduces transportation costs, and lowers the carbon footprint for transporting the goods to the market. This system fails though in cities where land costs are high. Instead of reducing costs by improving energy efficiency, costs were reduced by expanding the production vertically (Sarkar & Majumder, 2019). The researchers calculated the production costs of a six-story unit placed in an urban environment and found that the system could be economically feasible (Sarkar & Majumder, 2019).

A final, third approach to efficiency comes from the world of aquaponics. Traditional farming requires large amounts of fertilizer, water, and land. Furthermore, the process is slow, expensive, and the yields are relatively fixed. Elsokah and Sakah propose introducing aquaponics into the farming system

(Elsokah & Sakah, 2019). The fish act as a free form of fertilizer, but once grown, they can be harvested as a cheap source of protein. This solution can improve yields in poorer countries where the farmers lack access to fertilizer (Elsokah & Sakah, 2019).

### Automation

One of the primary costs incurred in vertical farming systems is that of labor. Unlike commercial farms that have benefited from hundreds of years of agricultural research, vertical farms are still a relatively new concept. They have not been optimized for an industrial agriculture system and require much more manual labor than in conventional farms. Labor could be reduced, though, if the watering system were automated. Belhekar et al. do this by applying a Raspberry Pi microcomputer (Belhekar, Thakare, Budhe, Shinde, & Waghmode, 2018). In this system, a wide variety of data points are collected, such as temperature, moisture, and pH level. These data points are then plugged into an algorithm to determine when to water the plants (Belhekar et al., 2018). This algorithm was further refined and used to balance the pH levels of the system. These pH levels were then optimized for the particular plants growing in the system (Thakare, Belhekar, Budhe, Shinde, & Waghmode, 2018).

Other researchers took a similar approach but on a larger scale. Most out of season fruits are grown in greenhouses at great expense. The greenhouses pollute the surrounding areas through fertilizer contaminated water runoff (Choi, Choi, Kim, & Lee, 2016). Researchers attempted to remedy this by creating an automated watering system for a commercial greenhouse. Their results showed that by more precisely controlling their watering schedules based on soil moisture, they reduced fertilizer costs by 41% (Choi et al., 2016).

A different method of automation instead focuses on aquaponics. In an

aquaponics system, fish and plants form a symbiotic relationship. The fish provide fertilizer, which is then absorbed by the plants (Aishwarya, Harish, Prathibhashree, & Panimozhi, 2018). This reduces fertilizer costs, and fish feed costs are offset from the selling of mature fish. Aishwarya et al. surveyed a number of proposed systems and found that the best systems incorporated an automated water supply and an automated fish feeder (Aishwarya et al., 2018).

A final approach was found in a patent application. In the patent application, daily maintenance is automated thanks to a network of sensors that monitor such things as humidity, carbon dioxide, pH level, etc. (Olesen, Smith, & Korn, 2019). These inputs are then fed into a control algorithm that determines the appropriate actions required to maintain an adequate growing environment. Where this system differs from the others is that it can grow multiple crops at once. The system mixes different concentrations of fertilized water and then stores it in designated reservoirs. These reservoirs are then used to water the different types of plants, with each plant getting its ideal fertilizer mixture (Olesen et al., 2019).

### Off-Grid and Renewable Systems

Because traditional farming methods rely on an abundance of cheap land, they are often located far away from cities and are not connected to the electric grid. Therefore, more scientific methods of drip irrigation and fertigation are impractical. Instead, fields are manually irrigated and fertilized based on fixed schedules. A potential solution to this is the creation of localized off-grid systems (Salih, Adom, & Shaakaf, 2012). These systems can remotely monitor the plants in the field and apply water and fertilizer as needed. An experimental fertigation system was installed, and it was shown that it could carefully monitor and accurately control the amount of fertilizer being delivered to the plants. If properly set up, it could increase the yields of plants farmed in remote areas (Salih et al., 2012).

Another off-grid system focused on aquaponics. This system used solar energy to power the pumps used to transport fertilized water from the fish holding tanks to the lettuce. Once filtered, the water was then sent back into the fish tanks (F. Ismail & Gryzagoridis, 2016). While the system added to the farming setup expense, it greatly reduced growth times and decreased the amount of spacing required for the plants. The plants were of a harvestable size in 35 days as compared to the 50-60 days in normal operation and could be planted closely together instead of requiring 120-150 mm of separation (F. Ismail & Gryzagoridis, 2016).

A final off-grid system was proposed by Labrador et. al. One of the main problems encountered by off-grid systems is that they still require a grid connection to maintain power stability. Daily and regional fluctuations can impact energy production and create shortfalls and surpluses that crash the system or allow energy to be wasted. Labrador et al. propose networking these off-grid systems to create a miniature grid (Labrador et al., 2019). In it, individual production shortfalls or excesses are shared among the systems in the network, significantly reducing their reliance on the grid and improving the efficiency of their renewable energy sources (Labrador et al., 2019).

### Industry 4.0 Applications

While many researchers are automating vertical farming systems, some are taking this a step further and applying Industry 4.0 techniques to their systems. One example of this is the work done by Valiente et. al. In it, they combine several systems to create an Internet of Things (IoT) to control an aquaponics system (Valiente et al., 2018). The data collected is taken in by an Intel Edison microcomputer, which adjusts conditions based on the outputs. One unique feature is the addition of a web camera to the system to allow visual confirmation of how the system was running (Valiente et al., 2018). This system was compared to a

traditional hydroponics and fishkeeping setup. The researchers found that for the first two weeks of lettuce production, there was no statistical difference between lettuce grown in the automated system and that of a traditional hydroponic farm. However, after two weeks, it was found that the IoT system had significantly better growth than a conventional setup (Valiente et al., 2018). The results were even more pronounced in the fish growing side of the experiment. After two weeks, the fish in the automated system were significantly larger than those in the control group (Valiente et al., 2018).

Another group of researchers focused solely on making an IoT system for a vertical farm. In this system, data is gathered and presented to the user in a web interface and is accompanied by a live feed (bin Ismail & Thamrin, 2017). Instead of relying on a complex algorithm to change the growing conditions, the user decides what should be done. This vastly reduces the system's complexity compared to an automated system and allows a farmer to monitor multiple fields at once (bin Ismail & Thamrin, 2017).

A final application of Industry 4.0 methods comes from the field of image processing. Even in industrial agriculture, much of the grading of fruits and vegetables, as well as the detection of disease, must be done by hand. Fruits and vegetables can take on a wide variety of shapes that fool image processors. Diseased plants must be identified in the field (Jhuria, Kumar, & Borse, 2013). To overcome this, Jhuria et al. trained a neural network machine learning algorithm to identify diseases on fruits and visually grade fruit based on weight (Jhuria et al., 2013). Not only could their algorithm detect the disease, but it could also categorize the severity of the outbreak, giving farmers better insight into how much insecticide should be used. Finally, their visual grading system was proven to guess the fruit's weight accurately and could help farmers determine the optimal time to harvest the produce (Jhuria et al., 2013).

## Applications for the EverGreen Project

The results of this research gave insight into what had already been researched by others and better approaches for this research. The efficiency improvements suggested by Gentry, using waste energy to heat the system, were adapted for the EverGreen system in the form of running lights at night during the winter (Gentry, 2019). This provided the plants extra light hours while also reducing the likelihood that they would freeze. This idea of capturing wasted energy was the nucleus of the idea of a thermal battery for the system. The adaptation of aquaponics into the system could then be used to eliminate the need for synthetic fertilizer in future system iterations (Elsokah & Sakah, 2019).

No researchers had tried to fully automate an off-grid vertical farm. Instead, only certain aspects of the farms were automated in most research systems, and these were mainly done in a laboratory environment (Belhekar et al., 2018). Those that were automated in a commercial environment focused on a single aspect, such as soil moisture (Choi et al., 2016). Fully automated commercial systems could exist in the near future, but they would be connected to the grid (Olesen et al., 2019).

Research into off-grid systems showed varying results. Some systems were designed to supplement traditional farming setups by more precisely controlling the amount of water and fertilizer given to the plants (Salih et al., 2012). Others were used in low power applications such as running pumps in an aquaponics system (F. Ismail & Gryzagoridis, 2016). Finally, some researchers combined multiple systems to create a more stable off-grid hydroponics system (Labrador et al., 2019). While the latter could be useful if multiple EverGreen units were in operation, what this shows is that the EverGreen research of creating an entirely off-grid indoor vertical farm is unique. No other researchers are attempting to automate an off-grid system using artificial light.

Research into Industry 4.0 applications showed that this avenue of work is still in its infancy and is mainly concerned with connecting various subsystems to control and monitor the system. It showed that a small micro-computer is powerful enough to monitor most systems and send out responses (Valiente et al., 2018). Furthermore, it gave insights into future system improvements such as an interactive online user interface that can let the user adjust the preset controls or incorporate a machine learning vision system to monitor plants for diseases (bin Ismail & Thamrin, 2017; Jhuria et al., 2013).

Finally, research in hydroponic farms gave a framework of possible ways to automate the system and its pitfalls. Many researchers have attempted to apply Industry 4.0 principles to hydroponic farms (bin Ismail & Thamrin, 2017). Others have sought to improve the efficiency of vertical farms and reduce their costs (Gentry, 2019; Sarkar & Majumder, 2019). Significant amounts of research have been done to automate vertical farms, but virtually all of these attempts have been in a small laboratory setting (Belhekar et al., 2018; Thakare et al., 2018). Few hydroponic systems use renewable energy sources. Those that are, are not fully automated or off-grid such as this project's system (Salih et al., 2012).

## Problem Statement and Hypotheses

Weather volatility, seasonality, and the system's energy storage capacity determine the number of vertical farming units that can run each month. Daily excesses and shortfalls of energy production show up in the quality of the growing environment. Similarly, to utilize 100% of the generated energy, the system should be automated to use whatever energy is available at any given time, turning on more vertical farming units or improving growing conditions when there is excess energy, and turning off units when there is a consistent energy shortfall.

This research aims to create harmony between a vertical farming system and

an off-grid renewable energy system. To achieve such harmony, well-balanced coordination between energy generation and the energy consumption is required. Balanced coordination can be designed, implemented, validated, and optimized through systematic steps, as shown in the following hypotheses:

- `Hypothesis 1:` It is possible to calculate the power production of the system based on historical weather conditions.
- `Hypothesis 2:` The maximum number of plants that can grow in the system can be calculated based on historical weather conditions and the power correlation generated in hypothesis 1.
- `Hypothesis 3:` An automated system can adapt to changing weather conditions, reducing power as necessary, without causing a power failure, or killing the plants.
- `Hypothesis 4:` An automated system can store excess energy by turning the system into a thermal battery.

## Research Roadmap

Fig. 1.1 illustrates the roadmap to complete this study. The acceptance or rejection of the hypotheses can give a clear insight into the system operation and optimization. The following list of methods and materials describe how each hypothesis can be assessed and evaluated.

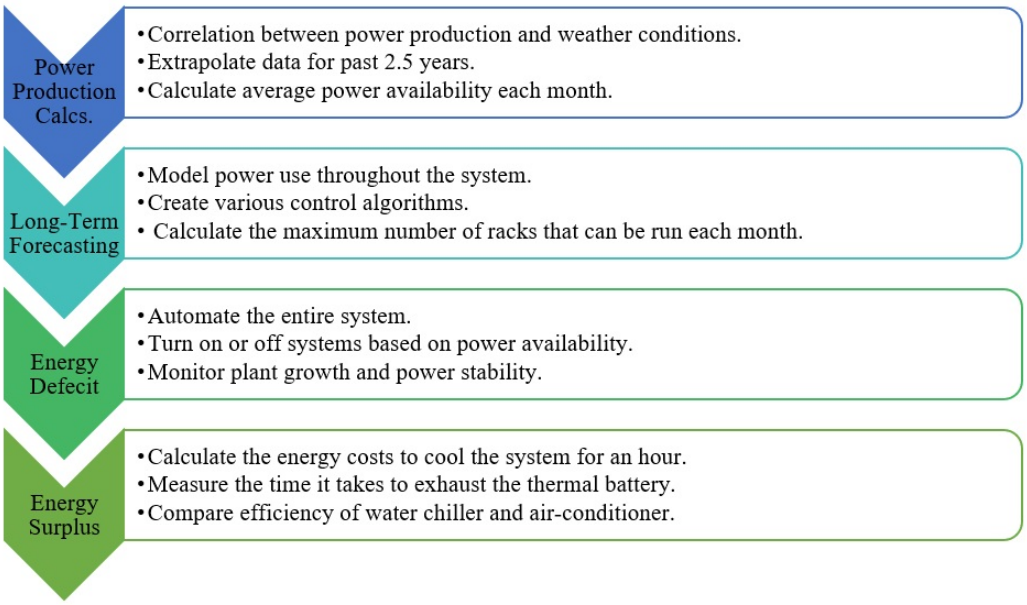



Figure 1.1: The roadmap for this study.

# 2. SYSTEM OVERVIEW

The experimental setup uses a 40 ft., climate-controlled shipping container located in an off-grid environment. This system is comprised of eight, three-level racks, for a total of six trays per rack, for growing fresh produce hydroponically, Fig. 2.1 and Fig. 2.2. Each tray is lit by LED grow lights. A communal reservoir, housing two pumps, contains the water for an individual rack. One pump pumps water to the top tray on the right side of the rack, the other the left. Siphon pumps are then used to draw water into the lower trays before being returned to the reservoir at the bottom. The water used in the system is primarily from two sources, rainwater, and atmospheric water. The primary source, rainwater collection, is gathered from the roof where the solar panels are located and stored in a large reservoir. The other source, atmospheric water generation (AWG), is an energy intensive system that generates water by condensing the water vapor in the air. This system is used primarily as an auxiliary source of water in times of drought and as a use for surplus energy.

In this system, power is provided by a 57-panel array of 350 W solar panels. Energy is stored in a 20 kWh lead-acid battery bank. There is no connection to the grid, and there are no backup generators. When the battery bank is exhausted, the system shuts down. The DC power generated by the panels is converted into AC power by an 8 A capacity inverter. Fig. 2.3 shows the entire power generation and storage system. A three-month trial was used to fine-tune the system and record the results. When monitoring the system, the system was split into two parts, power generation, and power usage.

When the power is generated is as important as how much power is generated in a day. Because the battery pack could be filled four times over in a day, it is critical that as much of the energy as possible is used as it comes into the system.

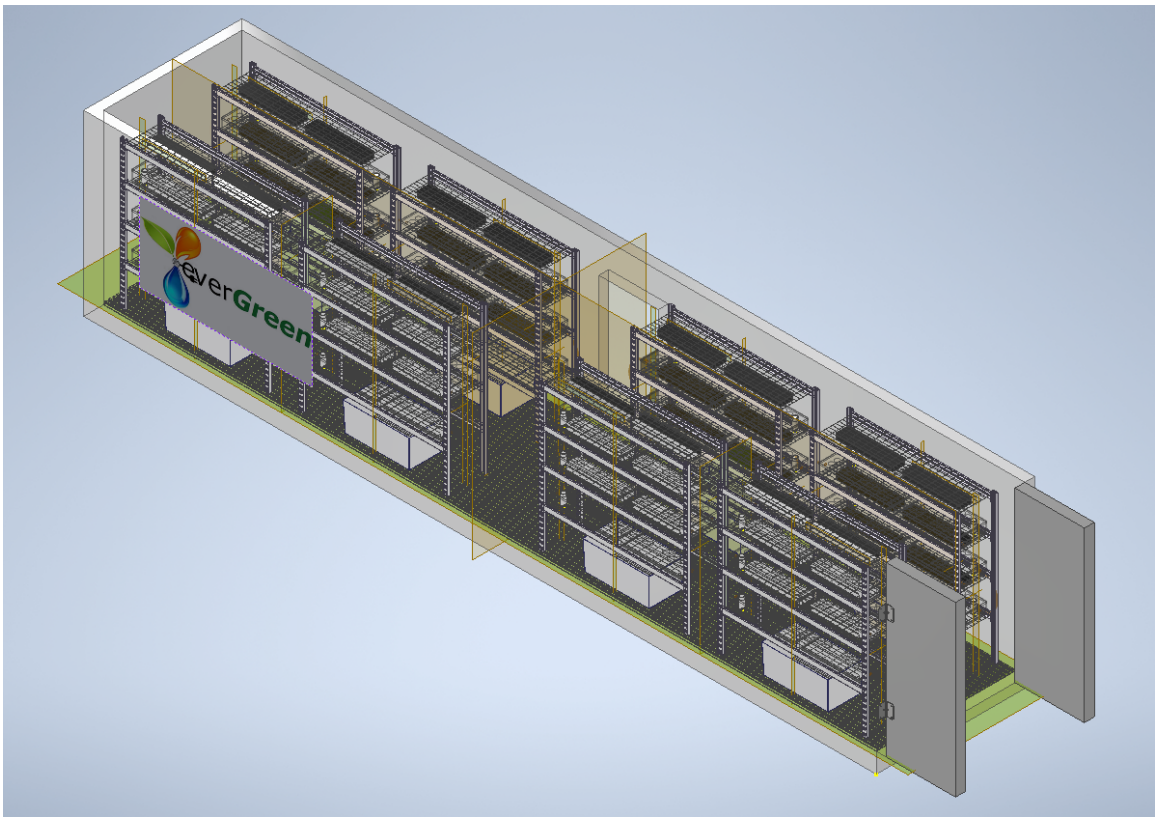



Figure 2.1: Indoor hydroponic farming, physical system model.

This was the driving force in molding the demand to meet the supply. Additionally, it was not economically feasible to expand battery capacity. Batteries have a high initial cost and a low lifespan. A typical battery for the system only has the capacity for 500 charge cycles before it needs to be replaced. This, combined with the energy conversion process's inefficiency, necessitated using the energy as it came into the system.

Figure 2.2: Indoor hydroponic farming, a close up of one rack.

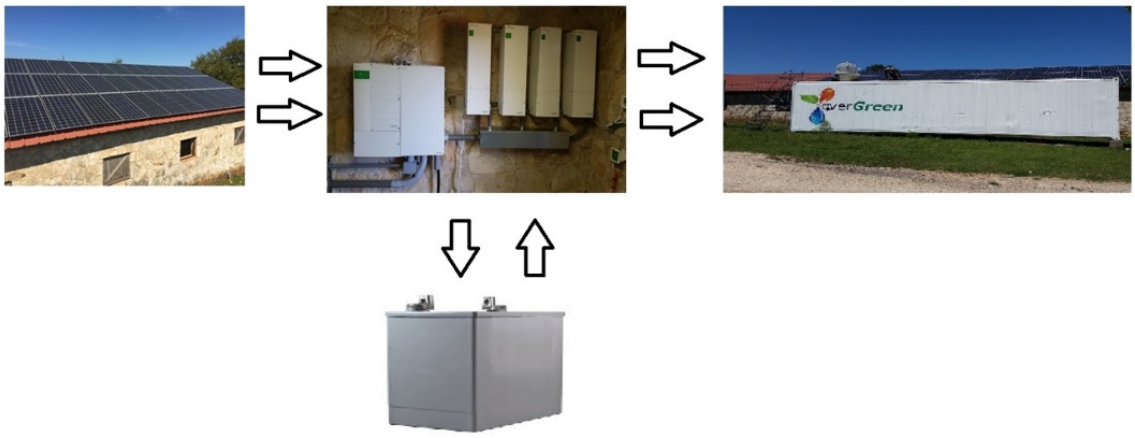


Figure 2.3: Flow of power in the system.

# 3. CORRELATING WEATHER CONDITIONS AND SOLAR POWER PRODUCTION

One unique trait of the off-grid system is that daily power generation depends on the season and weather conditions. For example, on a typical sunny, summer day, the EverGreen lab setting generates roughly 75 kWh of energy, Fig. 3.1. Only a small amount of this energy is stored in the batteries due to their limited capacity. To reduce energy waste, the vast majority of the generated energy should be consumed in real-time. Furthermore, excessive energy consumption during the day will drain energy from the batteries and lead to system instability and shutdown. Therefore, power demand must be roughly equal to the power supply.

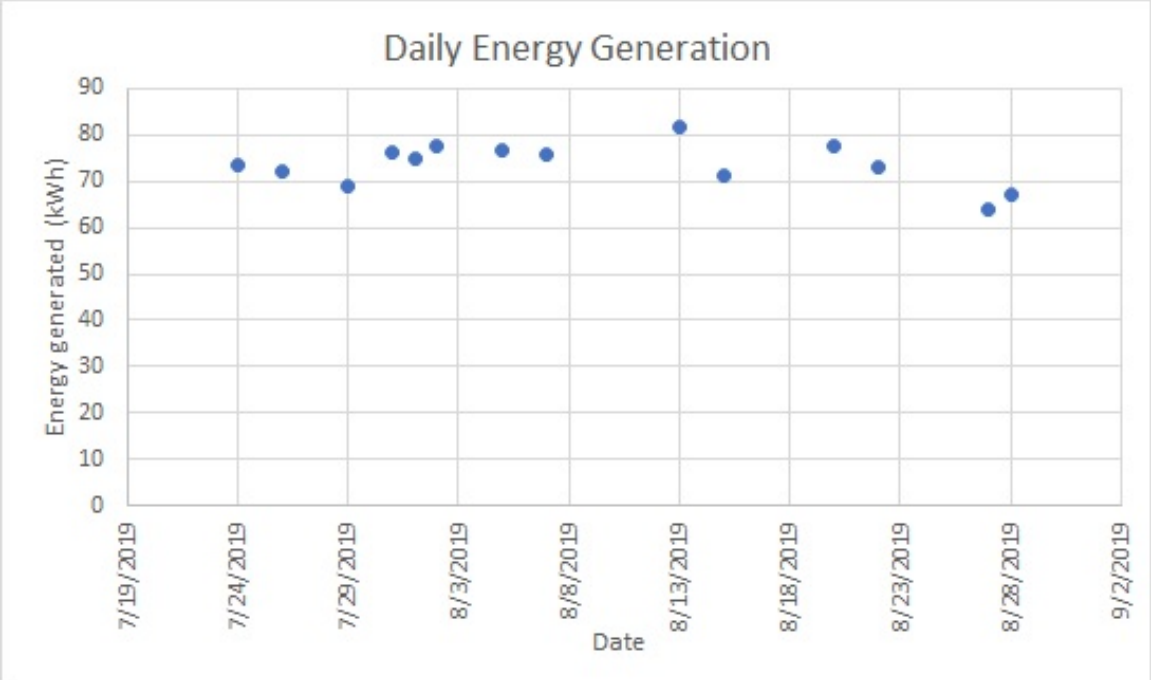



Figure 3.1: The average daily power generation in summer.

This chapter investigates the correlation between available energy (for a given time and weather condition) and the amount of usable energy generated by the off-grid Photovoltaic (PV) solar panel.

*Hypothesis 1: It is possible to calculate the power production of the system based on historical weather conditions.*

## Methods and Materials

While many systems rely on sophisticated machine learning (ML) algorithms to accurately predict solar panel output 1-72 hours in advance, the goal was to find the relationship between historical weather conditions and the power produced at any given time (Pedro & Coimbra, 2012). The data was analyzed using simple methods, then increasingly complex methods were used until a satisfactory relationship was discovered.

The data in question came from two sources, local weather reports and readings taken directly from the solar panel system. The weather station reports were hyper-local, made by a third-party provider, starting in September 2017. This data included solar irradiance, temperature, wind speed, and direction, for a total of twenty separate data points. For analysis, duplicate data points were removed from each hourly reading, including soil temperature, wet bulb temperature, etc.

Furthermore, irrelevant data points were removed as well as those data readings primarily filled with null entries. This reduced the number of potential points down to ten. For simple correlations done in Excel, only solar irradiance was considered. For more complex algorithms involving machine learning, all the data points were included.

Readings taken directly from the solar panel came in two varieties, manual and automatic. The manually entered readings were error-prone and not considered at first. Instead, the automatic readings were used. These readings were a selection of twenty total data points from the system, but they posed a problem in and of themselves.

The system did not directly record the amount of potential power available.

Instead, the system recorded the amount of energy captured at a given moment three separate ways. The first way involved summing the amount of power used by the EverGreen system with the amount of power entering the battery pack via the rate of change of ampere-hour battery capacity. The second way was similar, but summed the amount of power used by the system with the calculated number of watts flowing into and out of the system. The third way to measure power flow into the system was by taking readings directly from the solar panels.

Once the power readings were settled upon, a new problem became apparent. Power readings were taken at one-minute intervals, whereas weather condition readings were taken in one-hour intervals. This problem had three solutions. The first solution averaged the power readings over the preceding hour interval and used those when comparing to weather conditions. Solution two averaged readings over the previous five-minute interval and used that to compare with the current weather conditions. Finally, option three interpolated the weather readings over the hour. Because options one and two reduced the total volume of data present, they were used when large amounts of data were available. Option three preserved all the available data and was used when limited amounts of data were available.

When creating correlations, the primary strategy was to start simple and build complexity into the model. The literature showed that the primary factor for solar panel power production should be solar irradiance. Therefore, initial correlations simply relied upon an Excel correlation between solar irradiance and energy production. If this did not work, four simple machine learning algorithms were used with the full weather data set. These algorithms were Logistic Regression (LR), Perceptron, Support Vector Machine (SVM) with either a linear or RBF kernel and Decision Tree. To improve the correlation, the samples were divided into classes of data. To balance the classes, the data was up sampled. Samples were then reoriented using LDA and PCA data reduction techniques to reduce computational

load and lower the chance the model could be overfitted. Lastly, the most promising ML algorithms were run through a grid search to find the most suitable hyperparameters.

## Results

The first technique used to create a correlation between power production and weather conditions was done on Excel. In it, the daily energy production was compared to the solar irradiance of the prior 24 hours. These results did not give a clear correlation. There were too few data points and far too many weather variables. That is why the next technique attempted was machine learning.

Initial Machine Learning (ML) results proved inconclusive. To improve the results, several preprocessing and cleaning techniques were applied to the data before it was run through the ML algorithms, then the ML algorithm properties were fine-tuned using a grid search. First, the weather data points were organized. The original weather data represented over twenty variables. Some of them were removed because they were irrelevant, such as soil moisture. Others were removed because their data points were redundant. For example, the weather station calculated temperature three separate ways, but only one of those values, air temperature, was used. Some variable columns were mostly blank or had too few points to be considered applicable, such as wind chill and heat index. This reduced the number of features present in the data set to ten. This provided two benefits. First, it reduced the computational requirements of the Machine Learning algorithms. Second, reducing the number of features decreased the likelihood that the algorithms would overfit the data.

Once the weather station variables were organized, the solar panel data had to be analyzed. The data had several problems. One was calculating the amount of energy the solar panels generated. Ideally, the system itself would calculate the total

amount of solar power available at a given moment. This was not the case. Instead, the energy had to be calculated, given other usage factors. Of these, three were available. Options one and two summed the system's amount of power with the rate of change of energy stored in the battery packs. Option one did this by adding the rate of change of ampere-hours in the battery packs to the power use total. Option three added the system calculated flow of wattage from the batteries. Option three directly took the power generation data from the solar panels.

Of these three options, option two was found to be the best. Option one did not give a smooth power generation figure over time. The rate of change (ROC) in ampere-hours resolution was too low and was only calculated to the nearest whole ampere-hour. This was especially problematic at night. While power production at night averaged mostly 0 watts, the graph it generated showed mostly positive energy production followed by negative production, an impossibility, Fig. 3.2. Option two gave a smoother power production figure than compared to option 1, Fig. 3.3. This was largely because system load and battery power go down to the nearest watt of energy. The slight fluctuations can be attributed to error and energy losses. Option 3 records the data directly from the solar panels. While the PV output is always zero at night, and never records negative values, it did not appear to be accurate, Fig. 3.4. It routinely recorded watt values significantly less than what was being used by the system at that time.

Collection interval was the next problem that had to be overcome when analyzing the system data. Weather station data is taken once every hour, whereas the power system records data every minute. Because of the sheer volume of data and the volatility of weather at the ranch, power production data was down-selected rather than interpolating data points for the weather station data. This generated a new problem. When comparing the three methods of calculating power production, selecting a single point could result in a poor correlation. Therefore, several data

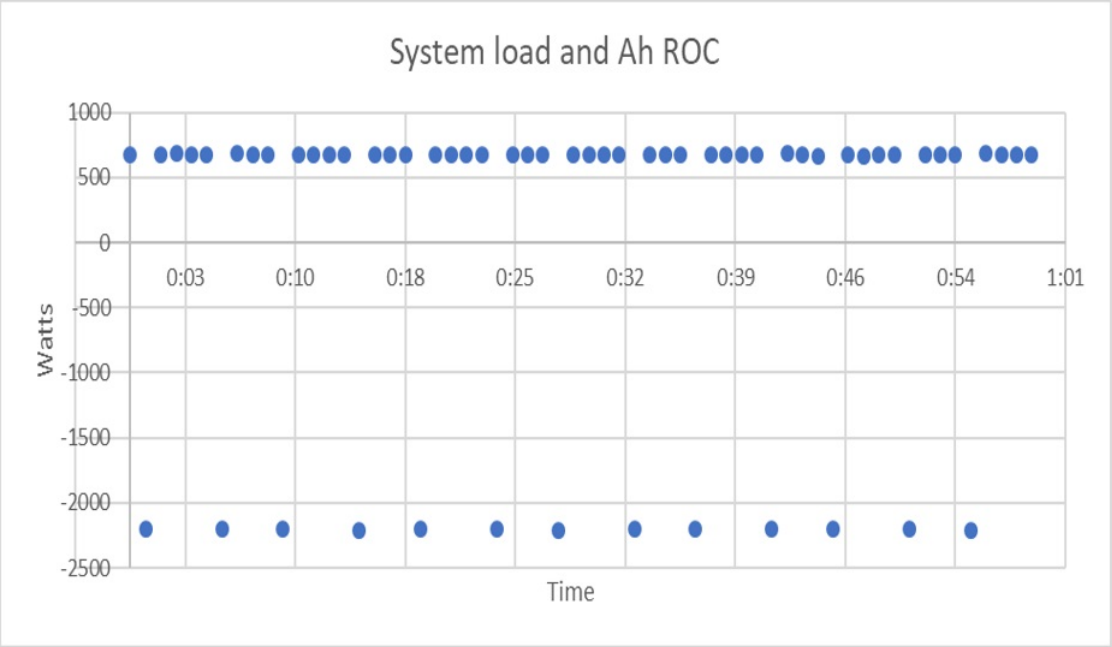



Figure 3.2: Option 1, system load plus battery ampere hour ROC.

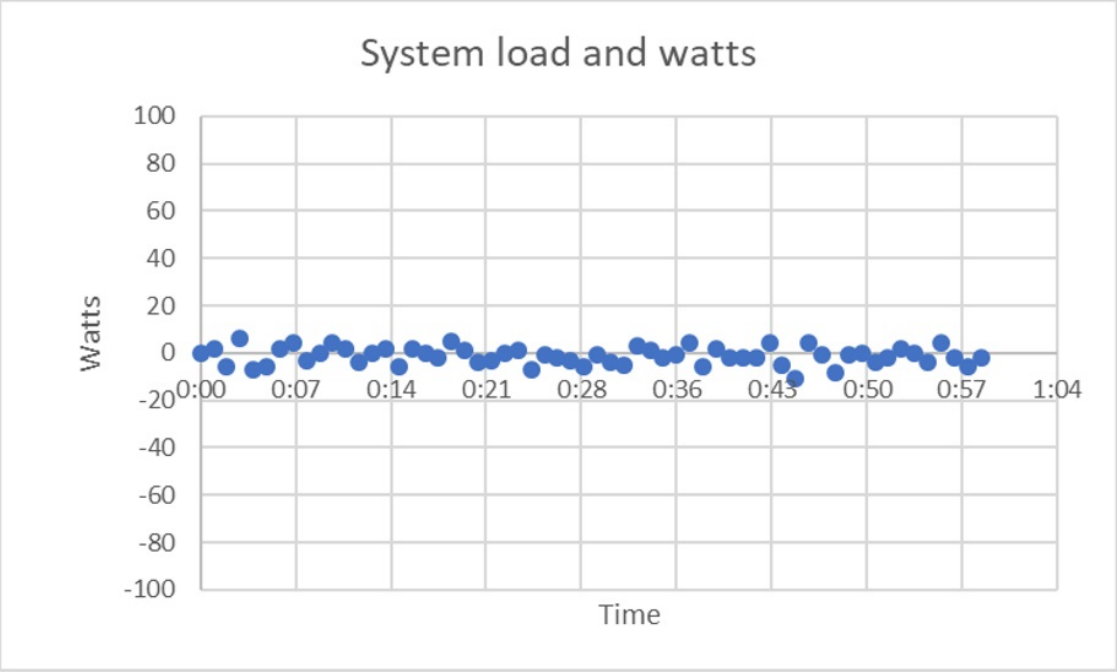



Figure 3.3: Option 2, system load plus battery watt flow.

points were averaged together and used as the value instead of one data point. Two averages were investigated: One average used every data point for the hour in

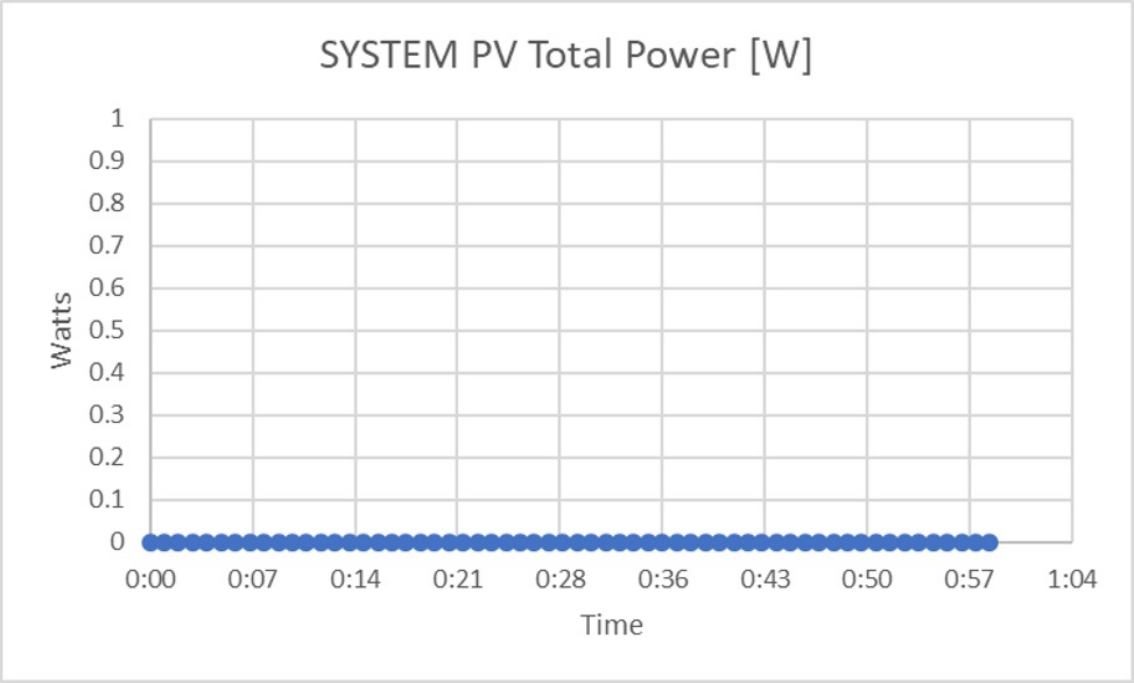



Figure 3.4: Option 3, system PV output.

question. The other averaged energy production over the next five minutes. When plugged into the machine learning algorithms, the five-minute averages gave better results, so this method was used.

The final problem for data organization, was classifying the power production levels. Power production is continuous, but most Machine Learning algorithms only work to output discrete values. Therefore, the values had to be rounded. Rounding into too small intervals results in too many distinct values, which reduces the accuracy of the correlation and increases the likelihood the system will crash. Rounding into too large of an interval reduces the quality of the correlation. After extensive testing, it was found that the best method was to round to the nearest kW of power.

Once the data collection and organization were completed, the data was run through PCA and LDA data compression algorithms. These algorithms find a better fit for the ML algorithms and reduce system complexity, without lowering the

quality of the data, Fig. 3.5 and 3.6. PCA algorithms reorient the axis of the data in question. Fig. 3.5 shows an example of this on a data set with two variables. The PCA algorithm reorients the data along a multidimensional plane for more complex correlations with multiple variables. A second program can then analyze the data and choose the two most relevant reoriented factors for the machine learning algorithm. LDA algorithms work similarly, but the variables must be fully independent and in the form of a normal distribution, Fig. 3.6. As the data was analyzed, it quickly became apparent that only the PCA method was applicable to the data set. The information was not sufficiently independent or in the form of a normal distribution, the two main LDA requirements.

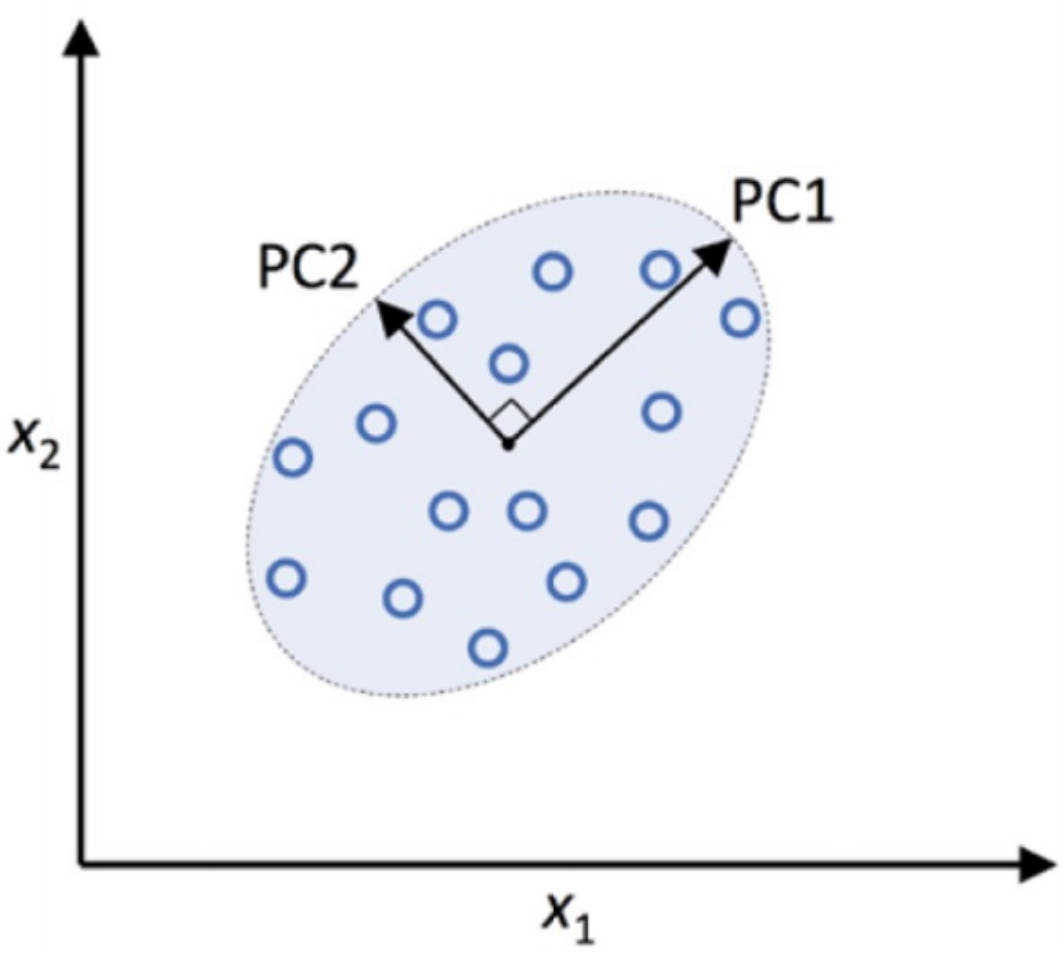



Figure 3.5: An example of a PCA data transformation (Valles, 2019).

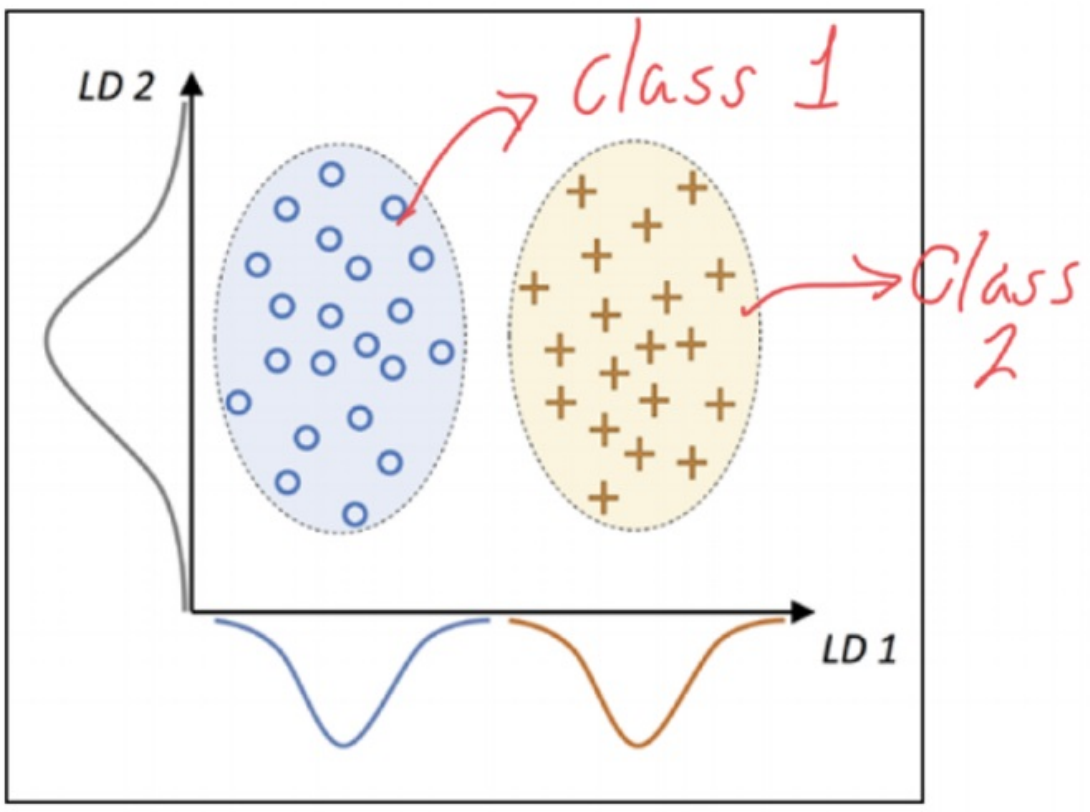



Figure 3.6: An example of an LDA data transformation (Valles, 2019).

The final data manipulation technique used before the ML algorithms were employed, was class balancing. The initial data set was highly imbalanced. Roughly 50% of the data points had a class of 0 because they were taken at night or during the early morning hours when energy production was negligible. Early attempts to run ML algorithms off this data resulted in skewed results. Testing and training accuracies routinely came in at 50-60%, but this was because the majority of the data points were classified as 0 or 1000W of energy production, the two most populous classes, Fig. 3.8. To remove this bias, the data points in each class were up sampled to match the number of points in the largest category, 0W. The only exception, was for outlier data points at or above 12000 watts. Because there were so few of these data points, these points were up-sampled together as a single group. This resulted in Fig. 3.7. While the training and testing accuracy plummeted to

16.2% and 15.6%, this was an improvement. Once the data was up sampled, no class of data accounted for more than 6%. Therefore, a 15.6% training accuracy could only be accomplished if multiple categories of data were correctly identified.

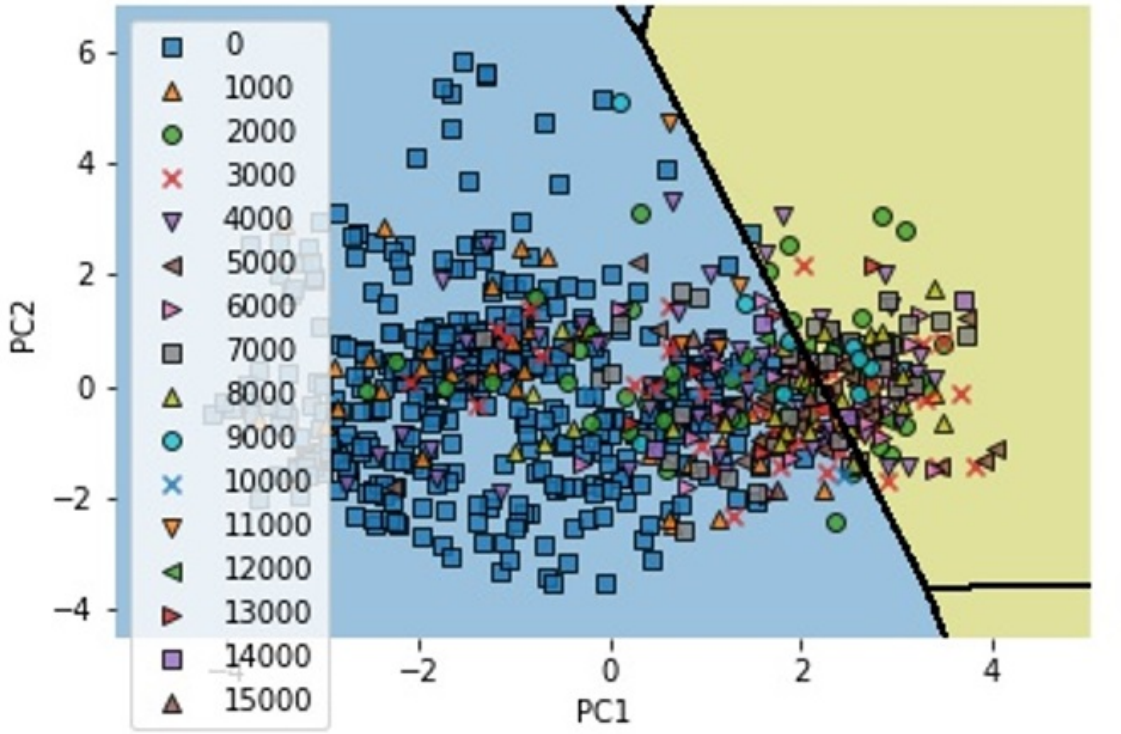



Figure 3.7: Initial Logistic Regression model with unbalanced classes.

Running the ML algorithms using this improved dataset gave testing accuracies of 15.6% for Logistic Regression (LR), 18.5% for Support Vector Machine (SVM) with a linear kernel, 59.8% for SVM with an RBF kernel, Fig. 3.9, and 64.4% for a Decision Tree model. At this point, research focused on Decision Tree models because they had significantly better results than LR and SVM with a linear kernel. The computing costs for SVMs with RBF kernels were prohibitive. Optimizing the hyperparameters using a grid search gave the following results. The best hyperparameters were a Decision Tree model using the Entropy criterion with a max depth of 20. This gave a 100% training accuracy and a 96.7% testing accuracy, Fig. 3.10.

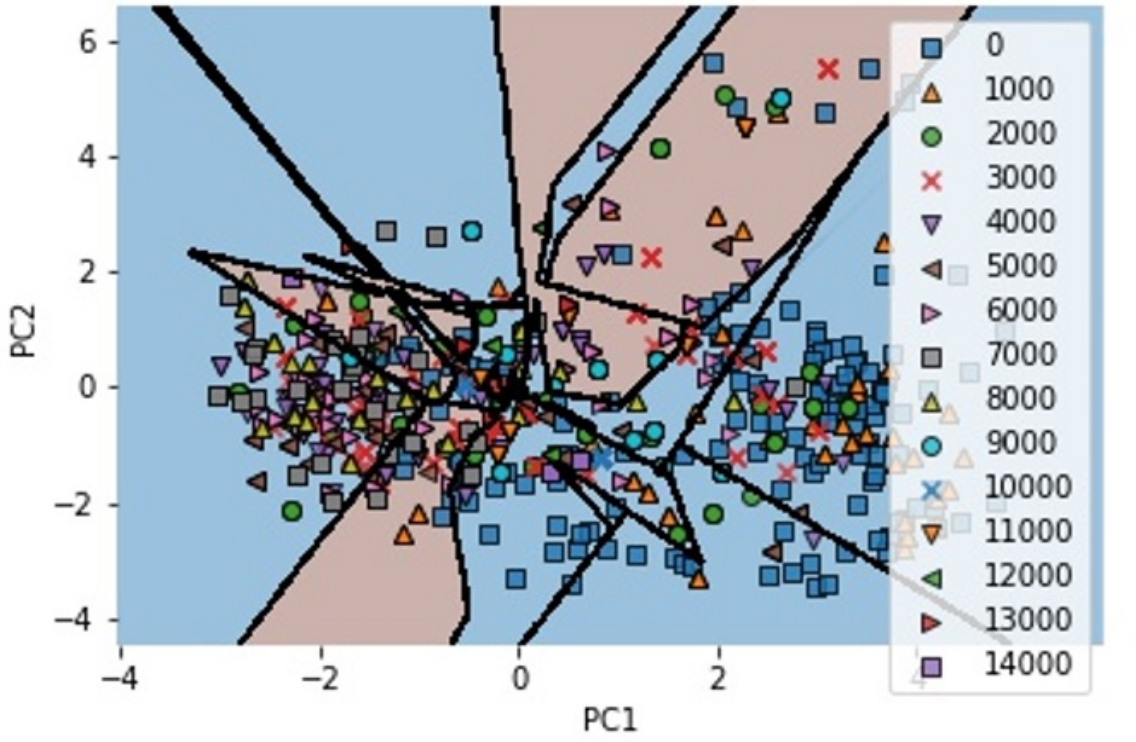



Figure 3.8: Improved Logistic Regression model featuring balanced data classes.

While these results were excellent, there were two problems with these results. First, the data was overfitted. While the accuracy did not drop significantly from the training to the testing phase, there was significant data overlap. This happened during the data cleaning phase when the data was up sampled. If the model were overfitted on duplicated datapoints, then the only samples that it could misidentify would be those points unique to the testing dataset. Running the model again with data points excluded from the up sampling procedure proved this, Fig. 3.11. Training and testing accuracy were almost identical up to a depth of 4. Beyond a depth of 5 the model was overfitted. At a max depth of 20, the ideal criterion found earlier, the training accuracy was 100%, but the testing accuracy was only 57.4%. Second, Machine Learning exists as a statistics tool. While the final model worked well in identifying the data, it did not explain why it did not match the theoretical models found in literature. Because of this problem, a different approach was tried.

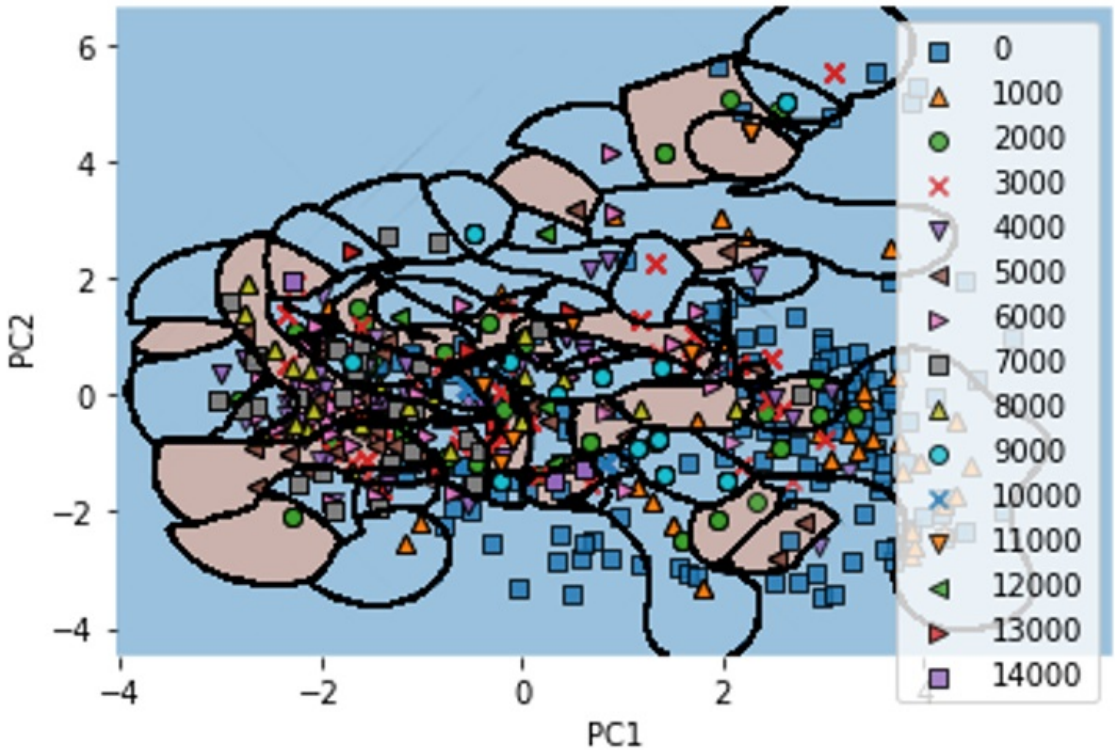



Figure 3.9: Testing results using an SVM with RBF kernel.

One flaw with the original data was that it did not show the total energy available at a given time. Instead, the system monitored the total energy harvested. This created problems when there was more energy available than could be collected at a given moment. An example of this is the dip in energy production found in Fig. 3.12. The initial peaks occurred when the system was using a large amount of power and storing excess energy in the batteries. Once the batteries were full, the excess energy was lost, and the only power being used was that going directly to the system, creating the dip. If the system were using all of the energy available, it would have looked similar to the graph of that day's solar irradiance, Fig. 3.13. While the system did not automatically record the transition between full energy usage and when available energy was wasted, it could be manually recorded. A week's worth of data was recorded, and only those points where the system was using all the available energy were considered. These points were then compared to

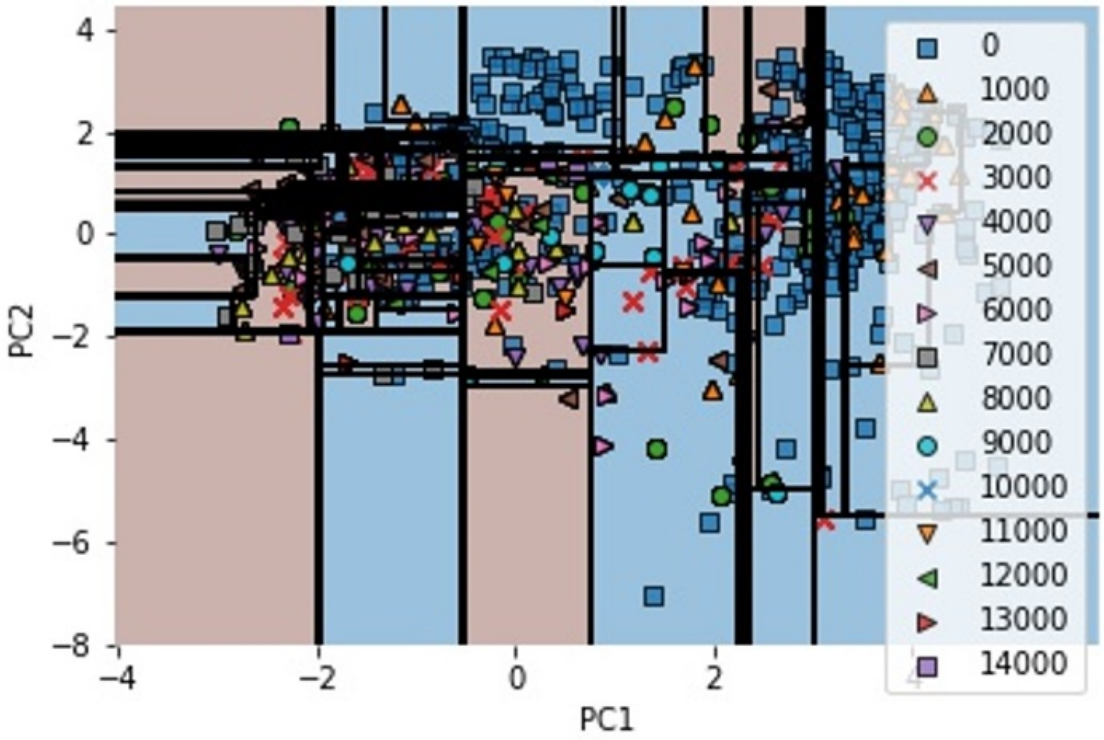



Figure 3.10: Testing results using a Decision Tree model.

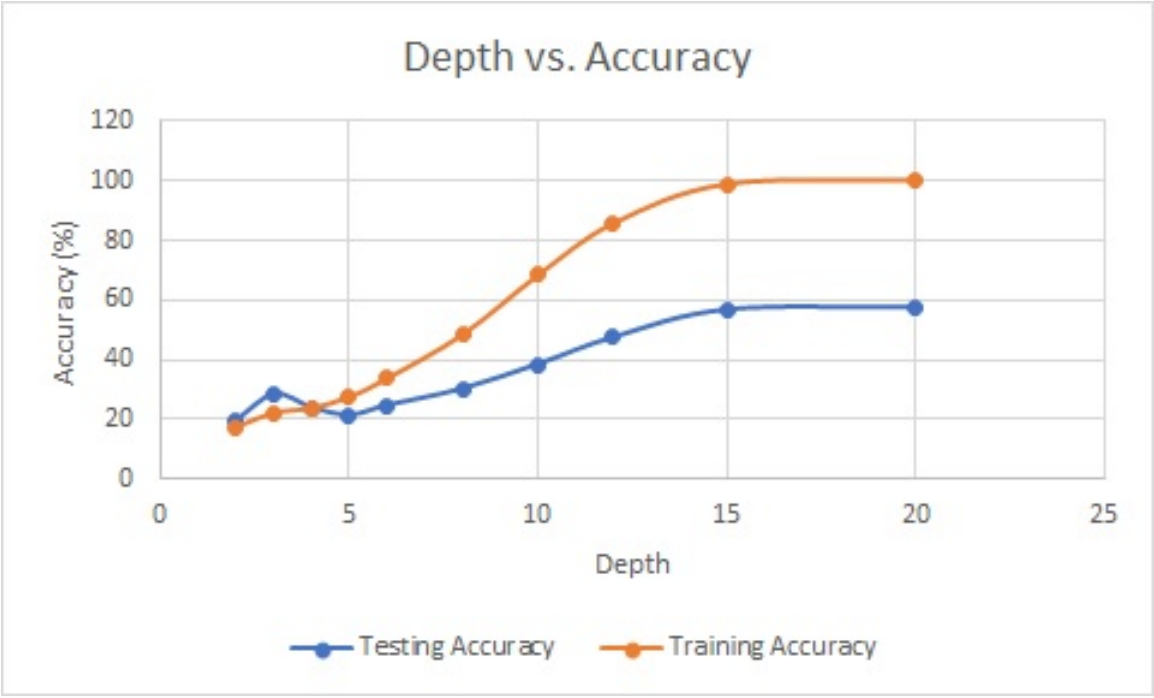



Figure 3.11: Accuracy of a decision tree model vs. the max depth.

solar irradiance. This resulted in a very strong correlation with an $R^2$ value of 0.8923 and a ratio of 22.136 watts of energy per watt of irradiance, Fig. 3.14. Because this method was simple, highly accurate, and matched the expected results found in literature, it was used when working on the second hypothesis.

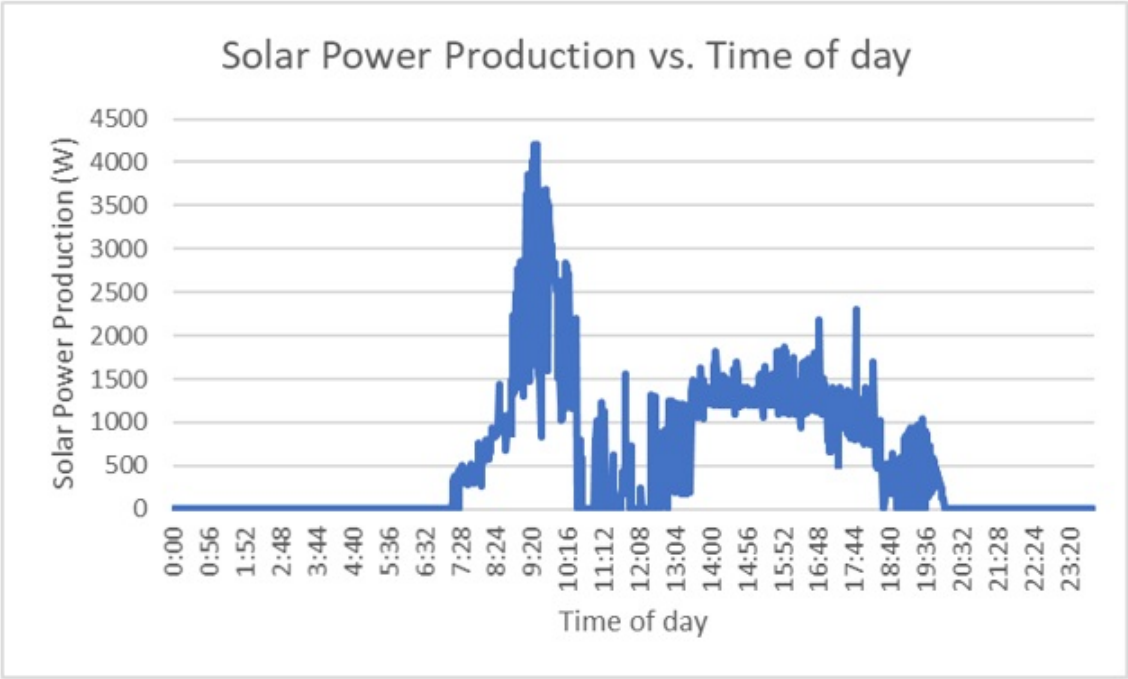



Figure 3.12: Energy production vs. time of day.

## Conclusion

Hypothesis 1 postulated that it was possible to calculate the power production of the system based on historical weather conditions. To prove this, a correlation between weather conditions and solar power production was created, Fig. 3.14. While the Machine Learning algorithms struggled to find a correlation based on the old, faulty data, the new data easily lent itself to a correlation. Because the correlation between solar irradiance and power output had an $R^2$ value of 0.9473, well above the 0.9 cutoff for a strong correlation, it is possible to calculate power production of a system based on historical weather conditions.

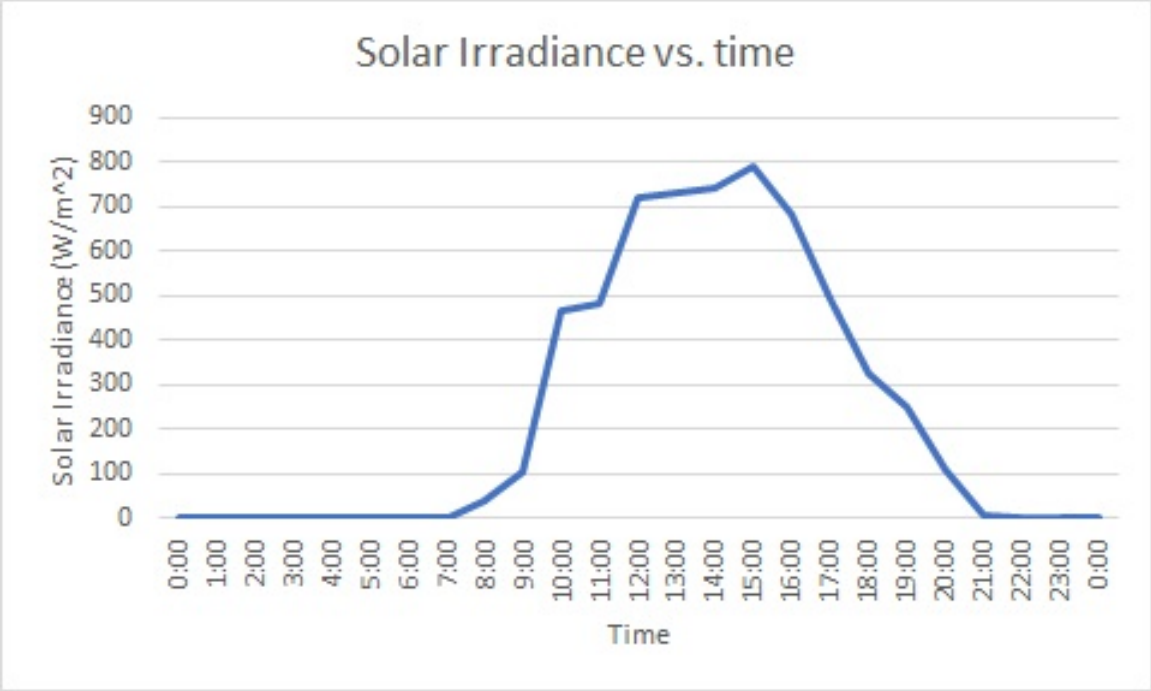



Figure 3.13: Solar irradiance vs. time of day.

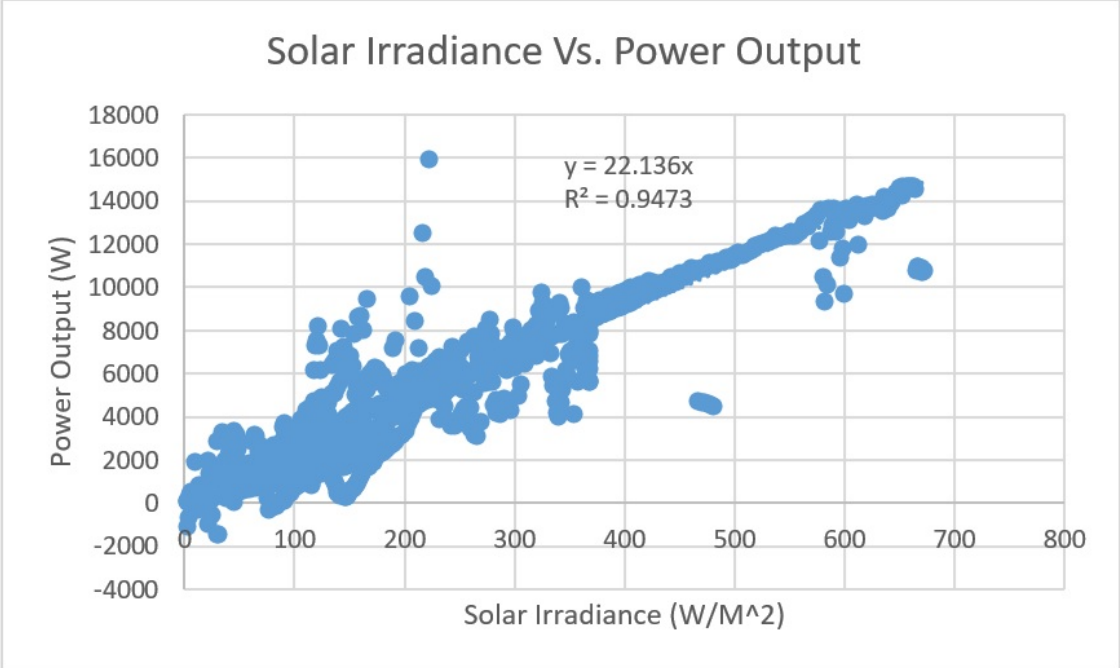



Figure 3.14: The graph of solar irradiance vs. power output.

# 4. LONGTERM PLANNING: CALCULATING THE MAXIMUM NUMBER OF GROW SYSTEMS THAT CAN BE ACTIVATED EACH SEASON

Sun-Earth position (azimuth angle and altitude angle) and daylight time for a given calendar day is repetitive year after year. This means that if weather condition is eliminated from the equation, each solar panel should generate an equivalent amount of energy on the same day each year. This chapter investigates the use of discrete event simulation modeling to predict the amount of energy that is expected on a given day. This model will allow users to plan energy consumption accordingly.

*Hypothesis 2: The maximum number of plants that can grow in the system can be calculated based on historical weather conditions and the power correlation generated.*

## Methods and Materials

This part of the system's goal was to create an Arena model that simulated energy production, storage, and consumption and could account for changes in outside temperature and days with low power consumption. The end goal for this system was to design something that used every possible watt of power. To do this, the best system should use all the available energy as it is coming in. This is illustrated in Fig. 4.1, which shows the theoretical potential energy generation curve of a typical day as a line and the energy usage for blocks of time as a bar graph.

The model was limited in the following ways. First, the system could not routinely exceed inverter capacity by a large margin. This limited the system to six active racks of plants at one time. Second, air-conditioning power uses roughly the same amount of energy as half of a rack filled with plants. Therefore, the system cannot run six full racks and the air-conditioner at the same time. Third, pumps

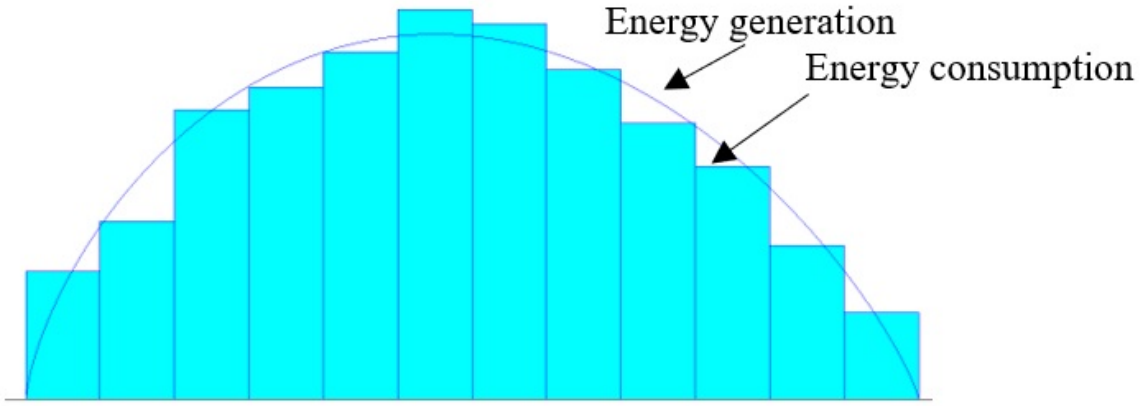



Figure 4.1: Theoretical energy generation vs. energy allocation/consumption.

must always have power available to them. Fourth, the cooling system must adapt to changing weather conditions. In hot weather, the AC should run at full power, whereas only the external fan should cool the system on more temperate days. Finally, the system must record how long it was out of power.

In addition to these limits, the characteristics of each subsystem were found. The inverter had a maximum sustainable capacity of 7200 watts with the option to provide up to 8000 watts for short periods. Lights consume 600 watts of power per half-rack, with a total of eight racks in the system. Running at full power, the air-conditioning system uses 700 watts of energy, while in fan-only mode, it uses 130 watts. The external fan uses 20 watts of power when it is running. Finally, the pumps run on a staggered schedule and pull a constant 20 watts of power. Baseline energy storage capacity is entirely from the reserve battery pack. Under normal operating conditions, the battery pack can provide up to 10 kWh of energy storage. The typical daily energy costs of each system, as well as the individual component costs, are shown in Table 4.1.

## Results

To fully maximize the power usage, a smart system was proposed that could adapt to changing energy conditions. Instead of managing the system on fixed

Table 4.1: Research Areas and Focuses.

| Device | Typical Energy cost per unit per day (kWh) | Hours used per day | Total daily Energy Usage (kWh) | % of whole |
|---|---|---|---|---|
| Lights | 1.81 | 9 | 65.28 | 85.1% |
| Air Conditioning | 9.35 | 11 | 9.35 | 12.2% |
| Fans | 0.85 | 12 | 0.85 | 1.1% |
| Pumps | 0.21 | 1.5 | 1.26 | 1.6% |

schedules, the system would run subsystems based on power availability. This had the benefit of running the system in the most efficient method possible. Limits in energy usage were put in place to account for the system's inverter and battery capacity. It was determined that a maximum of twelve high power devices could be used before the inverter reached its capacity. One device could be an AC unit or half a rack's worth of lights. Pumps and fans were negligible, so they were run irrespective of the amount of energy in the system. This meant that either six full racks of lights could be on at the same time, or five and a half racks plus the air conditioner. Battery capacity was restricted to 10 kWh worth of energy. This was done for two reasons. First, while the batteries have a theoretical capacity of 20 kWh, it was found that their usable capacity is much less than that. Second, fully discharging the batteries significantly reduces their lifespan. Therefore, a baseline of 10 kWh was used in the system model. Finally, to accurately model air condition usage, the outside temperature was considered. If external temperatures were low, the AC would be off; if high, the AC would turn on. Altogether, this resulted in the following model, Fig. 4.2.

The most straightforward portion of the simulation concerns the external temperature monitor, Fig. 4.3. Historical temperature data for the month is turned into a schedule. The schedule is read by a create node and creates one entity per hour for every degree the temperature rises. These entities are then held in a queue

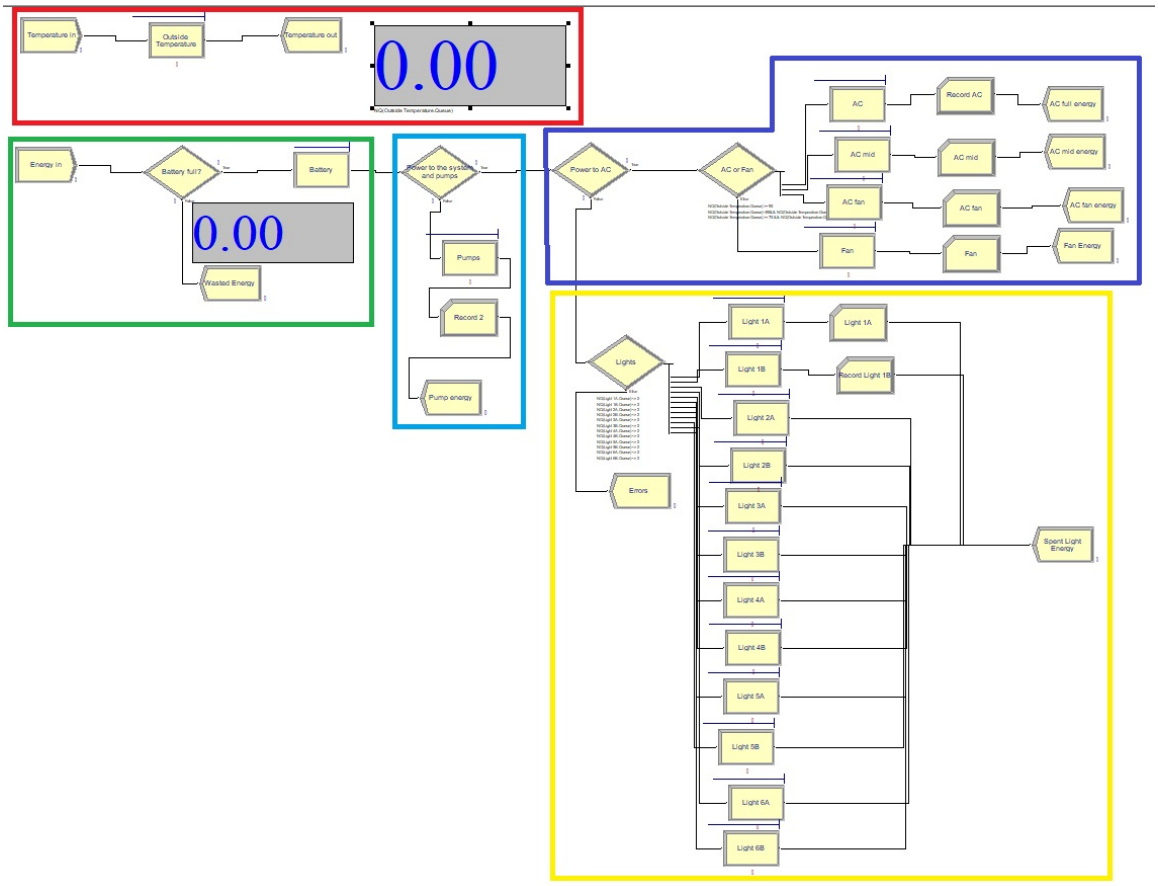



Figure 4.2: The full Arena model.

to be processed before they exit the system. The processing node changes in speed based on the number of entities in the queue. The goal is that each entity spends approximately one hour in queue. The temperature queue is read by the climate control part of the simulation to determine whether the air conditioner needs to be run.

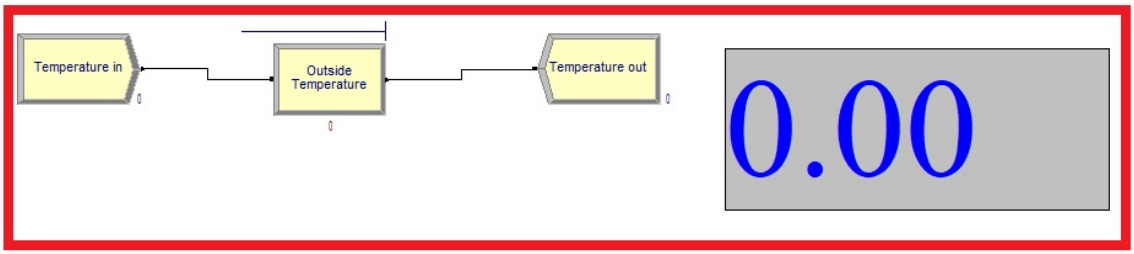



Figure 4.3: Temperature monitoring system.

Energy generation was simulated using actual weather data collected locally over the past 2.5 years. The data was broken down by month. The solar irradiance data was multiplied by the correlation data to get the amount of potential energy available each hour. Next, the potential energy data was turned into a schedule. Because the system starts without any reserve power, four hours of energy generation were added to the schedule to prime it when it was first started. The create node creates units of energy based on the schedule and sends them over to the "battery full" decision node, Fig. 4.4. Each energy unit is equivalent to 10 watt-hours of energy. Running the system in 10 watt-hour increments reduces the computational time necessary to run each simulation without significantly reducing the accuracy of the results. Once created, the decision node checks to see if there is room in the battery pack for another energy unit. If there is, the energy progresses towards the battery pack. If there is not, it exits the system as wasted energy.

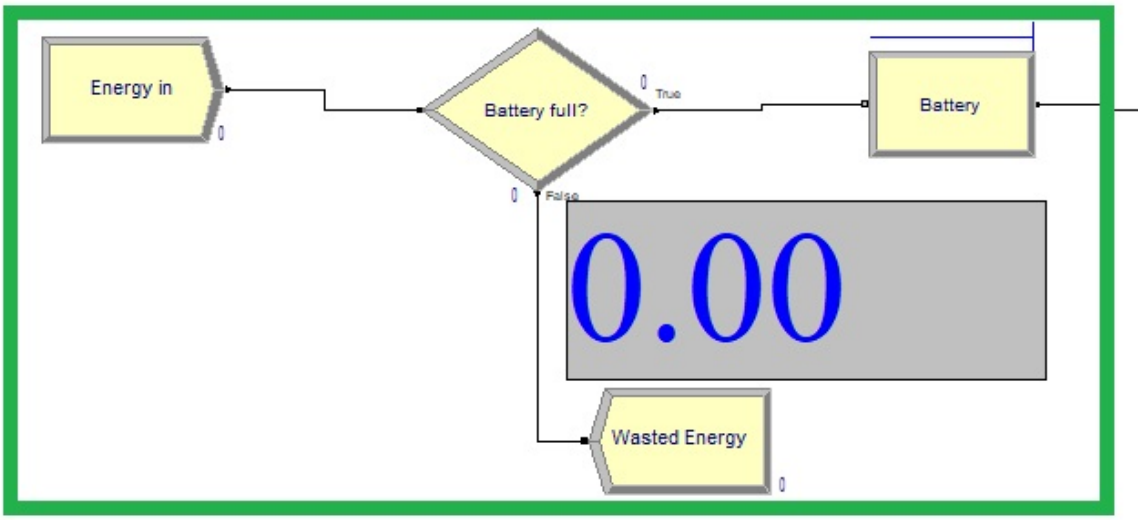



Figure 4.4: Energy entering the system.

The battery pack holds the units of energy until it sees a need for its use, after which it exits the energy portion of the system. The simulated battery differs slightly from the actual system where energy bypasses the battery if it is used in

real-time, but the change does not affect the accuracy of the simulation. Bypassing the battery merely reduces the wear and tear on the battery pack and slightly improves efficiency. The simulation assumes no wear on the components and that efficiency losses from the battery are negligible compared to the amount of power used directly by the system. Energy freely leaves the battery towards any subsystem when it has more than 3 kWh of reserve power stored. If the battery has between 1 and 3 kWh of stored energy, only the pumps, and climate control systems are run, this simulates the proposed reduced power mode. When the system has less than 1 kWh of energy, only the pumps are run, simulating the emergency power mode.

When an energy unit leaves the battery, it first passes a decision node to determine whether it is used to power the water pumps. If the pump queue is full, it continues down to the rest of the system, Fig. 4.5. If the queue is empty, it is sent towards the pumps. In the system, queues are used to prevent energy disruptions when energy is in transit from the battery to the subsystem. This is especially important for the pumps. The pump system accounts for energy being used to water the plants and make sure power is flowing in the system. If the pump servers ever stop working, this represents a power outage in the system. Therefore the battery is set up always to deliver energy to the pumps. The pumps, like all subsystems in the energy system, use the energy at a fixed rate. Once the energy is used by the pumps, it is recorded and leaves the system.

If the pumps do not use the energy unit, it goes to another decision node that determines if the temperature control system needs power or not. If it does need power, the energy unit goes to another decision node to determine how the system will be kept at an ideal temperature Fig. 4.6. If the temperature queue registers 75 or less, then the ambient outside temperature is less than or equal to 75 °F. This means that only the small external fan is required to keep the system in the ideal temperature range. This uses 20 Wh per hour. If the temperature is between 75 °F

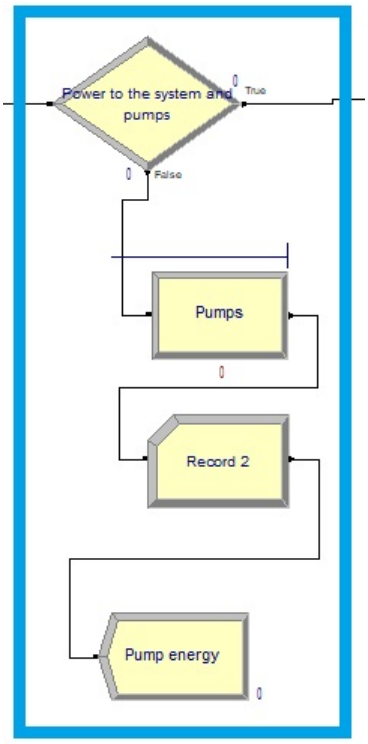



Figure 4.5: Water pump control.

and 85 °F, the air conditioner fan is run at 130 Wh per hour. Between 85 °F and 95 °F the AC alternates between on and off and uses an average of 400 Wh. Above 95 °F the AC runs at full power and uses 700 Wh per hour. A single resource manages both fan modes. When both have energy units in their queue, the AC fan is given priority until it is empty, after which it reverts to the stand-alone fan queue. If both queues are empty, it remains idle. The fan resource is idled because the outside temperature is greater than the system's ideal growing temperature. Running the fan would only draw in hot outside air and require the air-conditioner to run more. When temperatures are above the 85 °F thresholds, the air conditioner resource is used. It operates similarly to the fan resource. A single resource is used for both the AC mid and AC full process. AC full is given the highest priority, followed by AC mid, but when neither of these processes has a queue, the same resource is diverted to run the half-rack 1A. This is done because of the inverter limitations. As

mentioned earlier, the inverter is limited to a maximum of twelve high-power devices. Using the same resource to run the air conditioner or a light prevents the simulation from running simultaneously.

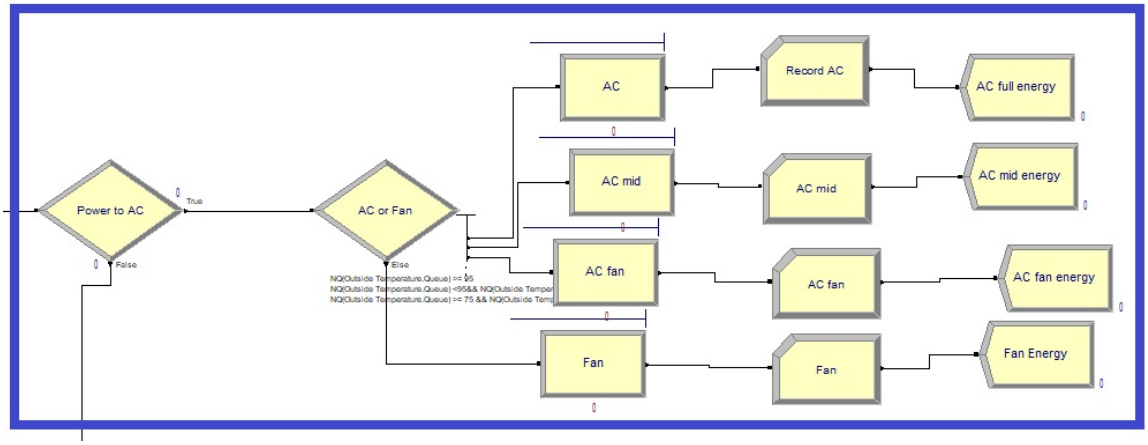



Figure 4.6: Cooling system diagram.

The last part of the simulation is the light control, Fig. 4.7. If power is not required by the pumps or temperature control system, it is sent to the lights. Except for the lights of half-rack 1A, each light has its resource and runs as power is available. The energy that arrives at this portion of the system is first directed to rack 1A. If the queue for rack 1A is full, it moves to 1B, then 2A, etc. down to 6B to find an empty queue. The resource utilization is recorded and determines the number of light hours each rack would have gotten on average. For lights on rack 1A, the number of light hours is determined by the ratio of energy processed through the light rack 1A over the total energy processed by that resource.

Running the data through the system gave the results listed in table 4.2. Each row represents the maximum number of light hours that could be given to each half a rack of plants if the system was running at full energy consumption. Because of the limited inverter capacity, it is impossible to run more than six racks of plants at once, hence the zero values for racks seven and eight. To get around this, table 4.3 was created. Table 4.3 averages the maximum number of light hours for each rack of

plants given X number of racks are in use. For example, in January, the first half-rack in use could run for 9.35 hours each day, the second 8.77, and the third 8.19. If the power were distributed equally, though, each rack could run for 8.77 hours per day on average. Using these averages and a cutoff of a minimum of 8 hours per day of light gave the results found in Fig. 4.8, which show the maximum number of racks of plants that can be run each month.

Table 4.2: Maximum number of light hours per half-rack.

| Month | Max number of light hours per rack | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Racks on | January | February | March | April | May | June | July | August | September | October | November | December |
| 0.5 | 9.35 | 9.44 | 9.82 | 10.39 | 8.29 | 6.86 | 6.25 | 4.89 | 6.38 | 8.26 | 9.03 | 8.91 |
| 1 | 8.77 | 8.85 | 10.09 | 11.29 | 11.59 | 12.00 | 12.20 | 11.57 | 10.63 | 9.76 | 8.93 | 8.28 |
| 1.5 | 8.19 | 7.89 | 9.55 | 10.79 | 11.22 | 11.76 | 12.09 | 11.41 | 10.35 | 9.39 | 8.32 | 7.65 |
| 2 | 7.64 | 6.99 | 9.05 | 10.36 | 10.83 | 11.46 | 11.95 | 11.30 | 10.08 | 8.97 | 7.80 | 7.12 |
| 2.5 | 7.15 | 6.33 | 8.53 | 10.01 | 10.44 | 11.16 | 11.79 | 11.18 | 9.81 | 8.56 | 7.36 | 6.75 |
| 3 | 6.72 | 5.79 | 8.04 | 9.74 | 10.12 | 10.88 | 11.64 | 11.05 | 9.53 | 8.17 | 6.98 | 6.41 |
| 3.5 | 6.33 | 5.32 | 7.64 | 9.47 | 9.86 | 10.63 | 11.48 | 10.88 | 9.26 | 7.85 | 6.66 | 6.04 |
| 4 | 6.00 | 4.94 | 7.28 | 9.19 | 9.66 | 10.37 | 11.27 | 10.70 | 9.01 | 7.60 | 6.40 | 5.71 |
| 4.5 | 5.78 | 4.64 | 6.96 | 8.94 | 9.46 | 10.15 | 11.09 | 10.52 | 8.83 | 7.41 | 6.16 | 5.45 |
| 5 | 5.61 | 4.42 | 6.70 | 8.70 | 9.25 | 9.95 | 10.91 | 10.39 | 8.70 | 7.23 | 5.93 | 5.28 |
| 5.5 | 5.50 | 4.24 | 6.47 | 8.55 | 9.09 | 9.79 | 10.78 | 10.29 | 8.61 | 7.07 | 5.73 | 5.16 |
| 6 | 5.42 | 4.11 | 6.31 | 8.43 | 8.96 | 9.67 | 10.70 | 10.21 | 8.53 | 6.92 | 5.57 | 5.08 |
| 6.5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7.5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 8 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Table 4.3: Average number of light hours per half-rack in use.

| Month | Average number of light hours per racks in use | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Racks On | January | February | March | April | May | June | July | August | September | October | November | December |
| 0.5 | 9.35 | 9.44 | 9.82 | 10.39 | 8.29 | 6.86 | 6.25 | 4.89 | 6.38 | 8.26 | 9.03 | 8.91 |
| 1 | 9.06 | 9.15 | 9.95 | 10.84 | 9.94 | 9.43 | 9.23 | 8.23 | 8.51 | 9.01 | 8.98 | 8.60 |
| 1.5 | 8.77 | 8.73 | 9.82 | 10.82 | 10.37 | 10.21 | 10.18 | 9.29 | 9.12 | 9.13 | 8.76 | 8.28 |
| 2 | 8.49 | 8.29 | 9.63 | 10.71 | 10.48 | 10.52 | 10.62 | 9.79 | 9.36 | 9.09 | 8.52 | 7.99 |
| 2.5 | 8.22 | 7.90 | 9.41 | 10.57 | 10.47 | 10.65 | 10.86 | 10.07 | 9.45 | 8.99 | 8.29 | 7.74 |
| 3 | 7.97 | 7.55 | 9.18 | 10.43 | 10.42 | 10.69 | 10.99 | 10.24 | 9.46 | 8.85 | 8.07 | 7.52 |
| 3.5 | 7.73 | 7.23 | 8.96 | 10.29 | 10.34 | 10.68 | 11.06 | 10.33 | 9.44 | 8.71 | 7.87 | 7.31 |
| 4 | 7.52 | 6.94 | 8.75 | 10.16 | 10.25 | 10.64 | 11.08 | 10.37 | 9.38 | 8.57 | 7.69 | 7.11 |
| 4.5 | 7.32 | 6.69 | 8.55 | 10.02 | 10.16 | 10.59 | 11.08 | 10.39 | 9.32 | 8.44 | 7.52 | 6.93 |
| 5 | 7.15 | 6.46 | 8.37 | 9.89 | 10.07 | 10.52 | 11.07 | 10.39 | 9.26 | 8.32 | 7.36 | 6.76 |
| 5.5 | 7.00 | 6.26 | 8.19 | 9.77 | 9.98 | 10.46 | 11.04 | 10.38 | 9.20 | 8.21 | 7.21 | 6.62 |
| 6 | 6.87 | 6.08 | 8.04 | 9.66 | 9.90 | 10.39 | 11.01 | 10.37 | 9.14 | 8.10 | 7.07 | 6.49 |
| 6.5 | 6.34 | 5.61 | 7.42 | 8.91 | 9.14 | 9.59 | 10.17 | 9.57 | 8.44 | 7.48 | 6.53 | 5.99 |
| 7 | 5.89 | 5.21 | 6.89 | 8.28 | 8.49 | 8.91 | 9.44 | 8.89 | 7.84 | 6.94 | 6.06 | 5.56 |
| 7.5 | 5.50 | 4.86 | 6.43 | 7.72 | 7.92 | 8.31 | 8.81 | 8.29 | 7.32 | 6.48 | 5.66 | 5.19 |
| 8 | 5.15 | 4.56 | 6.03 | 7.24 | 7.42 | 7.79 | 8.26 | 7.78 | 6.86 | 6.07 | 5.30 | 4.87 |

## Conclusion

The number of plants that can be grown each season varies widely, from two racks in winter to eight racks in summer. This is largely due to the variations in weather conditions. In the fall and winter, daylight hours are significantly reduced. This reduces the total amount of energy generation possible. Furthermore, as seen in Fig. 4.9, the angle the sun makes with the ground varies depending on latitude and time of year (Asiabanpour, Almusaied, Rainosek, & Davidson, 2019). Finally, adverse weather conditions such as rainfall and cloud cover significantly reduce energy generation. These adverse weather conditions explain why the system was more productive in January with its cloudless but short days vs. February's longer but rainy and overcast days. In spring and summer, the primary difference in system capacity comes from the length of days. During daylight hours the system operates at peak capacity and the length of the day determines how many light hours the plants are provided.

Hypothesis 2 postulated that the maximum number of plants that could grow in the system could be calculated based on historical weather conditions and the power correlation generated in hypothesis 1. To test this, a simulation was created. The simulation modeled an automated system's reactions to changing weather conditions based on historical data. The results were plausible, like those found during early experimental testing when the system was first set up. Therefore, the maximum number of plants that can be grown in the system can be calculated using a power correlation and historical weather data.

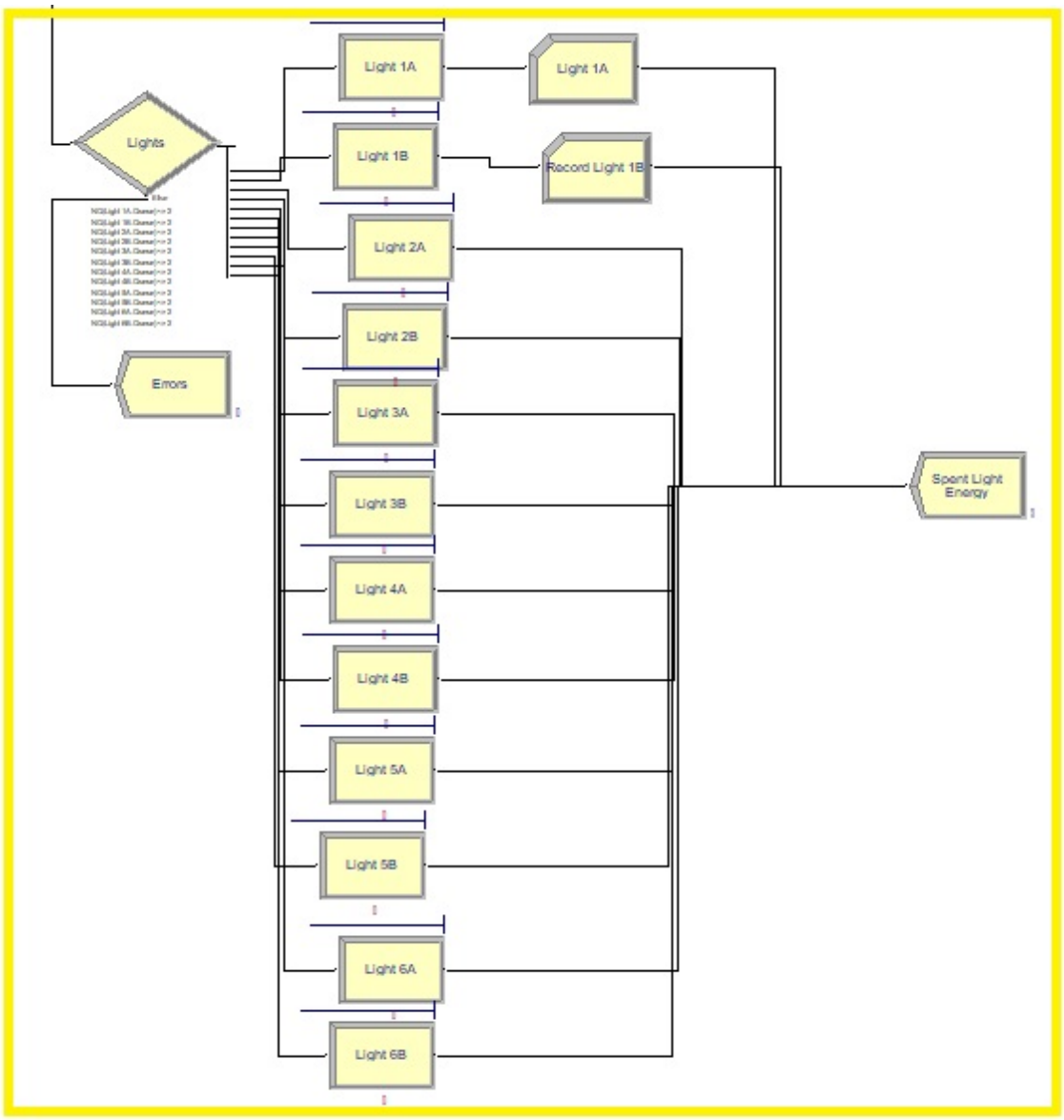



Figure 4.7: Light control system.

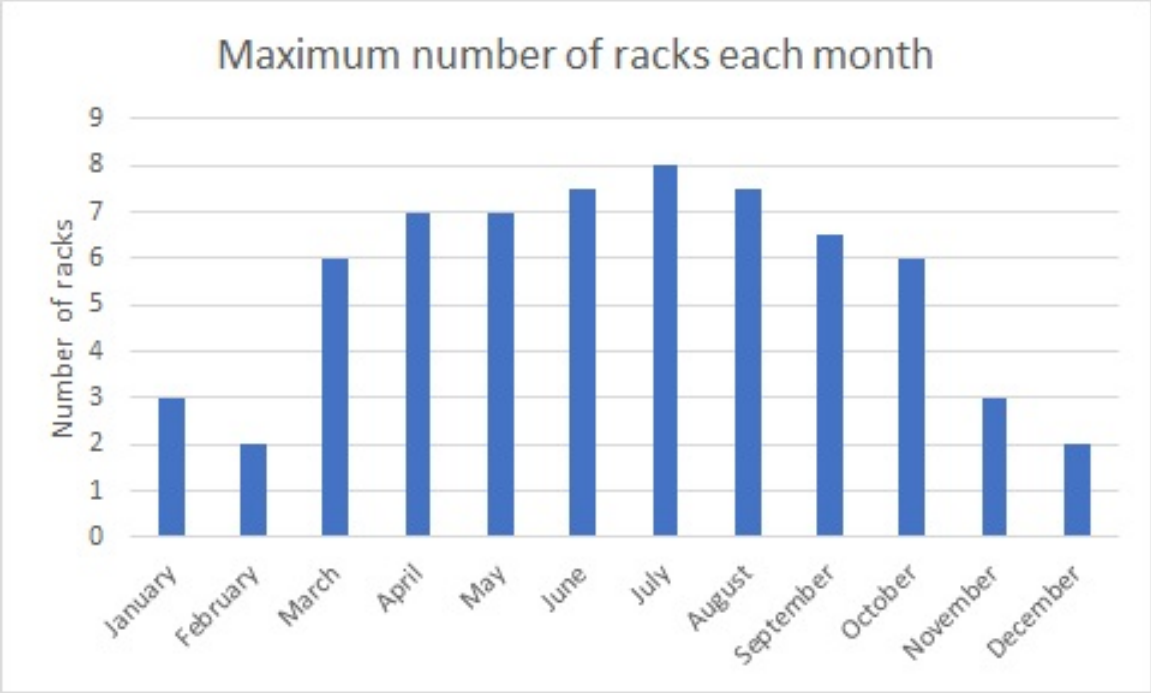



Figure 4.8: Maximum number of viable racks per month on average.

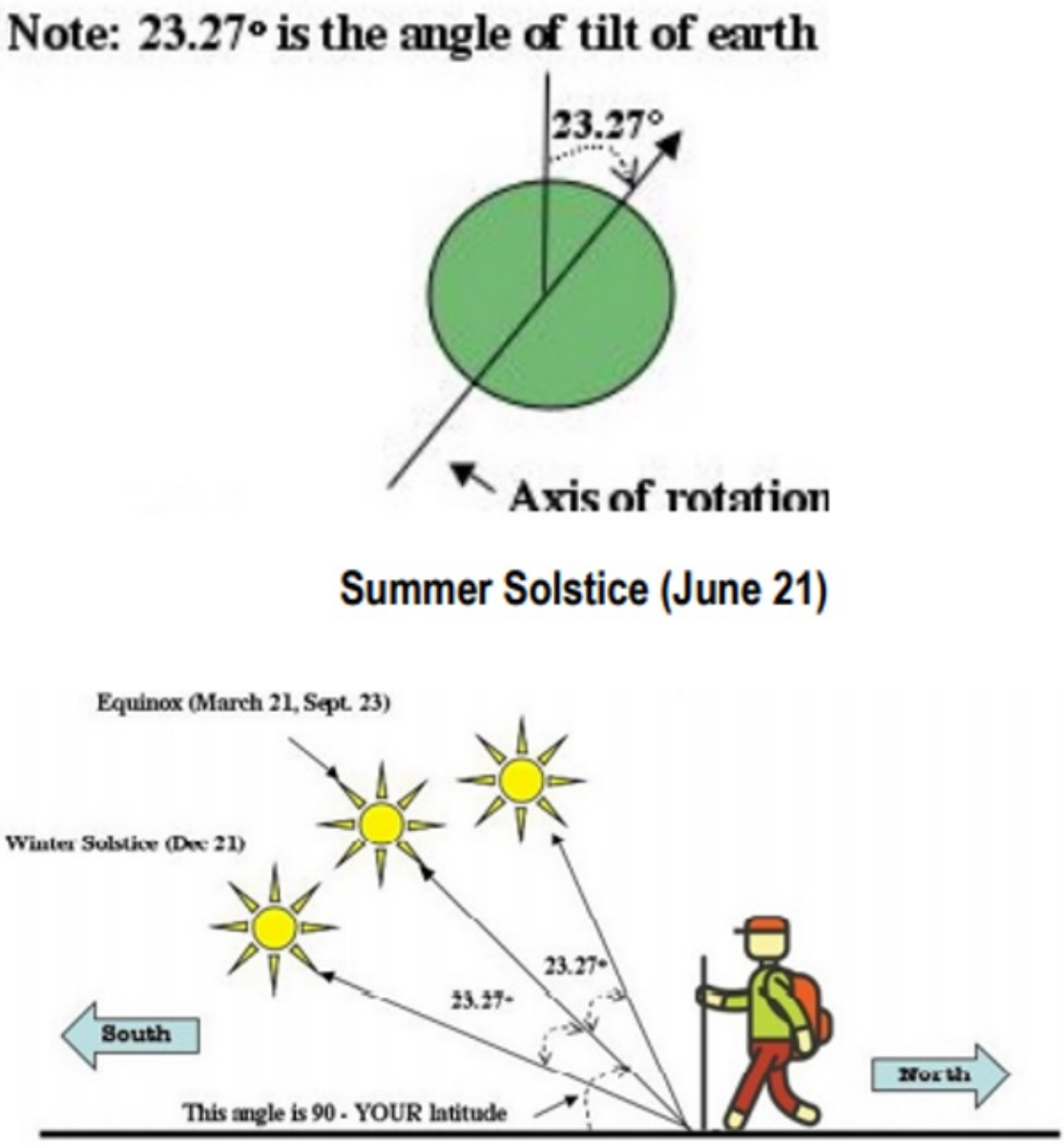



Figure 4.9: The effects of the earth's tilt (Asiabanpour, Almusaied, et al., 2019).

# 5. SHORT-TERM PLANNING: ADJUSTING ENERGY CONSUMPTION BASED ON THE AMBIENT CONDITION

Long-term planning gives the approximate amount of energy available each season, but it is not suitable for use in the short-term. The amount of power produced each day can vary significantly. For example, a cloudy, rainy day means the system produces virtually no energy, whereas, on a bright sunny day, the system has an overabundance of available power. That is why a system must be in the place that can adapt to changing energy production. This chapter investigates hardware, software, and logic control algorithm needed to develop a system that quickly reacts to the ambient conditions to keep the energy supply stable.

*Hypothesis 3: An automated system can adapt to changing weather conditions, reducing power as necessary, without causing a power failure, or killing the plants.*

## Methods and Materials

A system-wide power outage could be catastrophic for the EverGreen system. The short-term lack of water causes the plants to wilt while the heat buildup can directly kill the plants or cause them to go to seed. Automating the system could stop a system-wide power outage by turning off non-essential services and preserve a reserve amount of battery power to keep the system running smoothly.

The main sources of energy use are the lights, air conditioning, Fans, and pumps. Lights use the most power and are least critical for the short-term health of plants. Air conditioning is vital to the plants during hot days but not used during the winter months. Running at full power uses the same amount of energy as one-half of a rack of lights. Fans are non-essential and serve as a supplementary cooling system if the outside weather is cold. They use a negligible amount of power. Water pumps are essential but use a negligible amount of power. Therefore,

an ideal system would turn off lights first, air-conditioning next if it were in use, and save fans and pumps for last. In the event of an unavoidable power outage, the system should have the ability to be easily restarted if not automatically restarted. The automation system has several requirements. First, it must integrate with the existing architecture. Any system chosen must be compatible with the existing fans, pumps, AC system, and lights. Second, it must be able to interpret data and make decisions off the data. Therefore, it must be compatible with a range of external sensors as well as the RS 422/485 connection of the solar panel system. Finally, it must be inexpensive, reliable, and have a large programming community in case problems arise.

The main methodology used to prove this hypothesis comes in the form of simulating the system. In this method, an automated system and a set timer system will be run using the same data. The primary method used to compare the two systems will be power reliability. Power stability determines the likelihood the plants would grow to a mature size. During a power outage, all environmental controls are off. In the short term, the system could overheat, stunting the plants and impacting quality. If the power outage lasted long enough, the plants would die from lack of water. Therefore, both the automated and timed systems will be judged based on their power reliability data.

## Results

Two strategies were used to compare the effects of automating the system vs. having a fixed schedule system. Strategy one simulated a system run by timers and compared the power stability to the automated system in the previous chapter. Racks were run for eight hours per day, and the number of racks each month was chosen based on the results found in chapter four. For instance, where the system was running more than six full racks at once, the run time of each rack in the

simulation was increased. This was done so that the same model could be used for the automated system as well as the fixed system. In general, instead of running the lights and air-conditioner as power was available, the lights and AC were run for eight hours per day at fixed times. Lighting schedules were staggered to better match when the power was generated and to reduce the likelihood of using up the battery reserves during non-peak energy production hours.

The data from four different months was run through the modified program to see how the timed system would react. The four months run were January, April, July, and October, representing one month from each season. In January, two racks were run. In April and October, six racks. In July, roughly eight racks were run. The results showed that while the timed system did not exhibit 100% power stability for all months, it did maintain stability most of the time. Fig. 5.1 shows the full results. While this is promising, the problem is much worse. When a power outage occurs in the physical system, the timers stop. When power is returned, the timers are now out of sync with energy production, creating even more power outages. This does not happen in the simulation. Instead, the timers are entirely in sync with the ideal schedule.

Strategy two automated the system and monitored the power stability for two months. Two strategies were considered for system automation, a network of Raspberry Pi microcomputers or an industrial-grade programmable logic controller (PLC). The PLC was a professional system and ideal for the final system. Still, the Raspberry Pi system was chosen because of its simplicity, low cost, extensive support database, and its ability to be integrated into existing hardware.

The Raspberry Pi network can be divided into two parts, the hardware and the control algorithm. The hardware consists of a central hub Pi that controls a network of servant Pis through a Wi-Fi connection, Fig. 5.2. The hub Pi runs the control algorithm, collects temperature data from the shipping container Pis, and

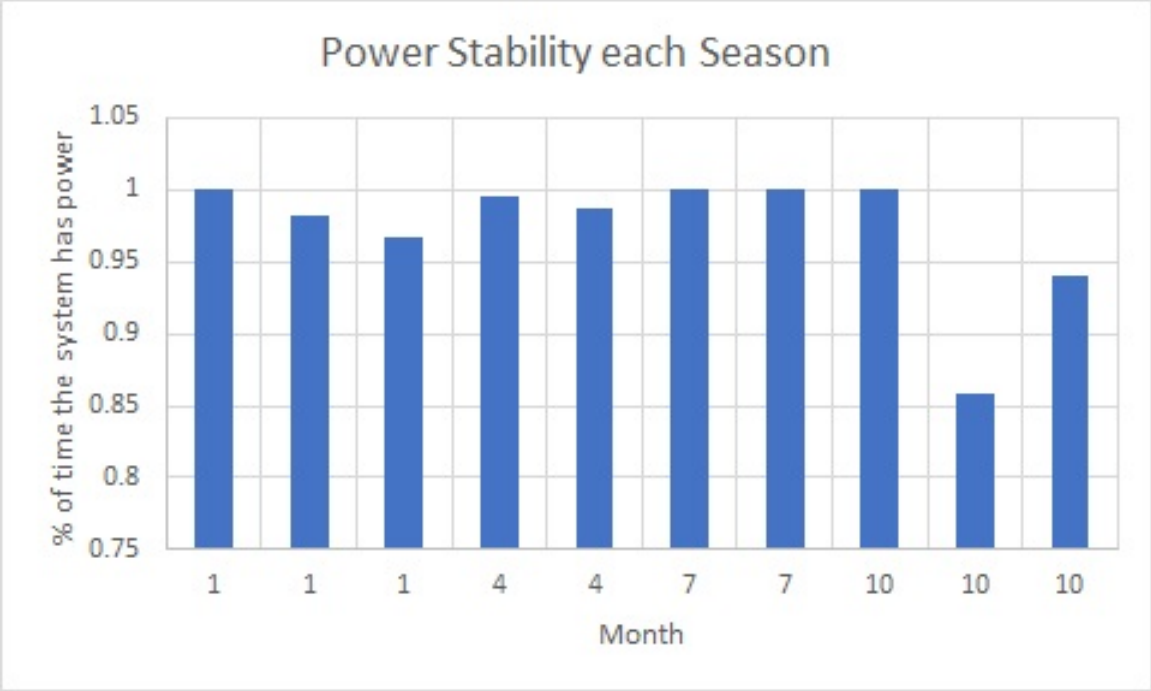



Figure 5.1: Power stability each season.

collects the state of charge (SOC) and net power flow of the batteries from the solar panel setup via a Modbus connection. This data is fed into the control algorithm to determine whether the AC should be turned on as well as the number of lights that can be turned on. Once the appropriate response is determined, the algorithm sends out the instructions to the servant Pis. Each servant Pi controls 1-2 power outlet relays via a hardline connection. In the event of a power disruption, the servant Pis are programmed to keep the outlets off until the connection is restored, preventing further power failures. The only Pi not networked to the system is the fan control Pi, Fig. 5.3. This Pi monitors the internal temperature of the container and the external ambient air temperature. If the internal temperature is higher than the external temperature and the internal temperature is greater than 65 °F, the fan is turned on to cool the system. This was done to reduce complexity, and because the external fan uses a negligible amount of power.

The control algorithm has two parts. Part one determines the number of

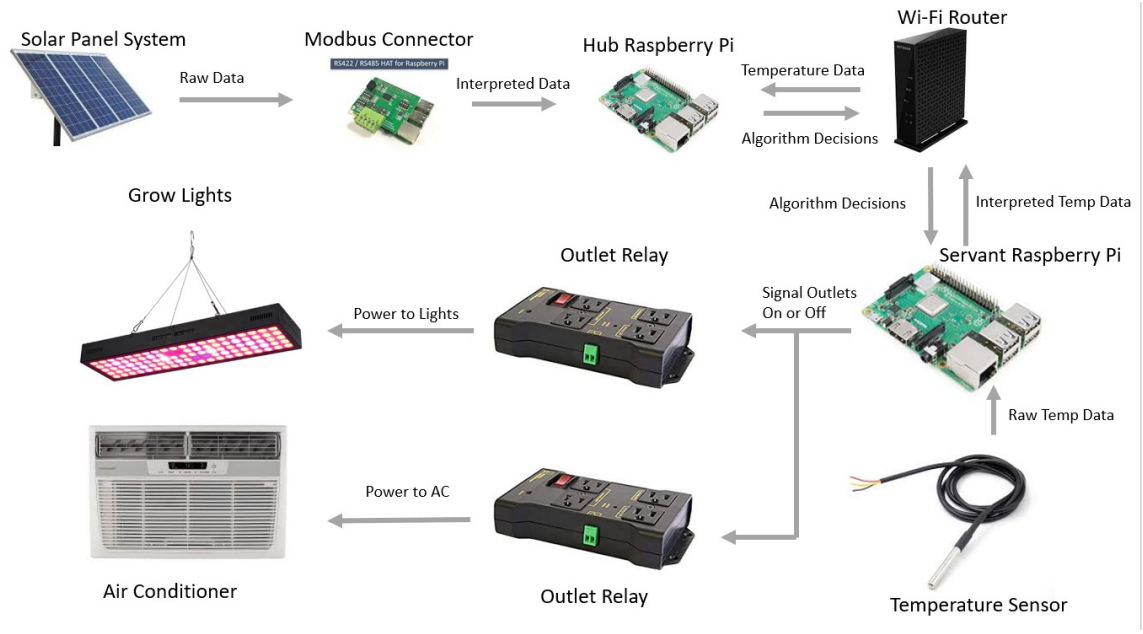



Figure 5.2: How the main system is connected.

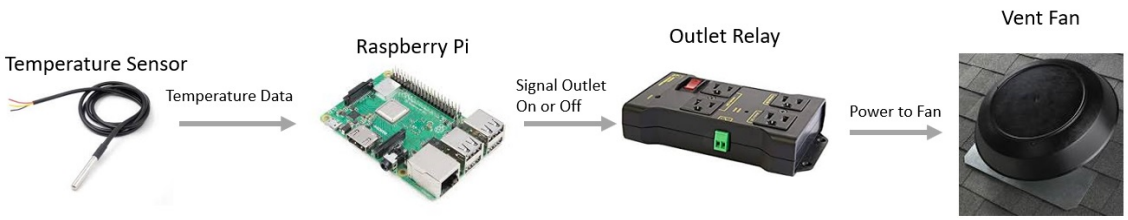



Figure 5.3: Fan control system.

devices to run, Fig. 5.4. Due to inverter limits, only 12 devices can be run at one time. One device is classified as a single air conditioning unit or half of a rack of plants. If excess power is available, limited energy flow out of the batteries (less than 2 amperes), the system will turn on an additional device. If excess power is not available, more than 2 amperes of power flowing from the batteries will turn off one device. This initial response is then modified based on which of the five levels the battery state of charge (SOC) is in. If the battery state of charge is below 10%, the system turns off all non-essential subsystems leaving only the fan and pumps running, as necessary. Between 10% and 30% power, the system runs the air conditioner as needed and can go up to running at half capacity if the batteries are

being actively charged. Between 30% and 50% battery SOC, the system will run up to half-power if excess power is available. Between 50% and 90% SOC, the system runs at least-half power, going up to full power if excess power is available. Because charging the batteries does not count towards inverter capacity, running down the batteries each day to 50% SOC increases the available energy in the system up to 5 kWh on sunny days. At 90%+ SOC, the system slows down charging the batteries, and excess power is wasted. To prevent power wastage, the system is run at full power irrespective of whether energy is flowing into or out of the battery pack.

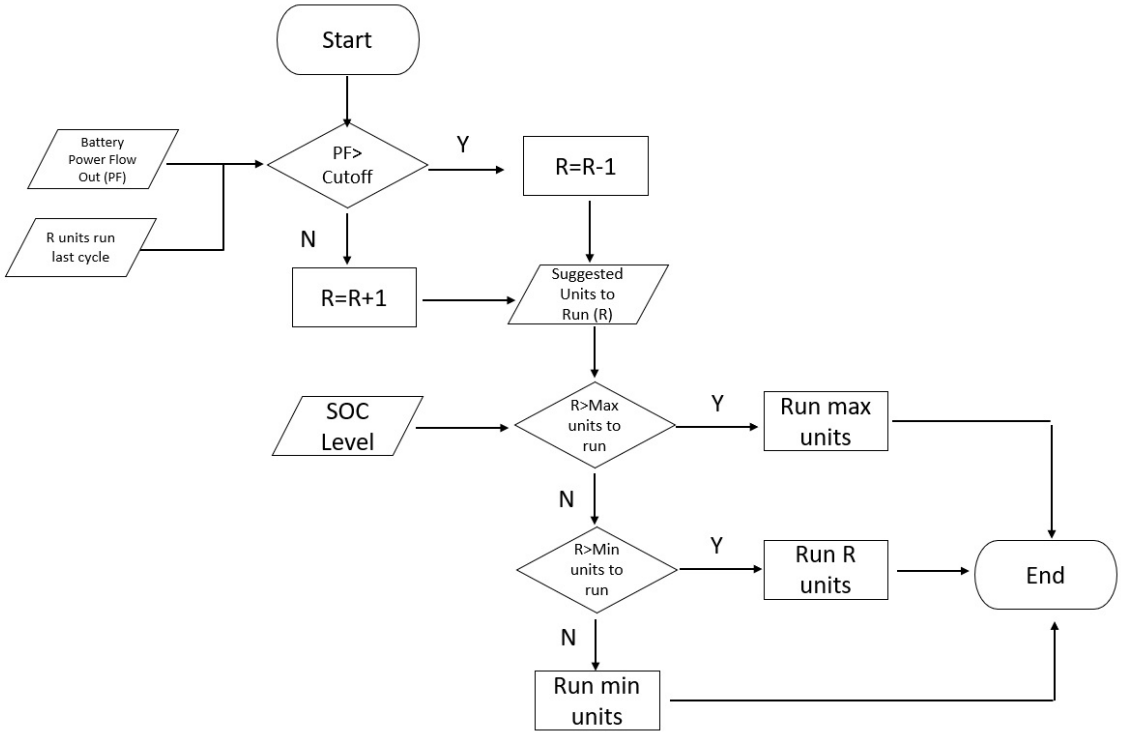



Figure 5.4: Algorithm to determine the number of units to run.

Once the system determines the number of devices that can be run, it goes down the list of available devices, Fig. 5.5. The air conditioner only runs when the shipping container temperature is above a certain threshold. If the temperature is below this threshold, this subsystem is not considered. Next, each half rack of lights is considered. The system tracks how long each half- rack has run that day. If it has

run longer than the user-specified time, it is ignored. If it has not, it is powered on. This process is repeated until every available device has been powered on or all available power has been allocated. Once this is done the algorithm sends out its decisions and waits sixty seconds before repeating the cycle.

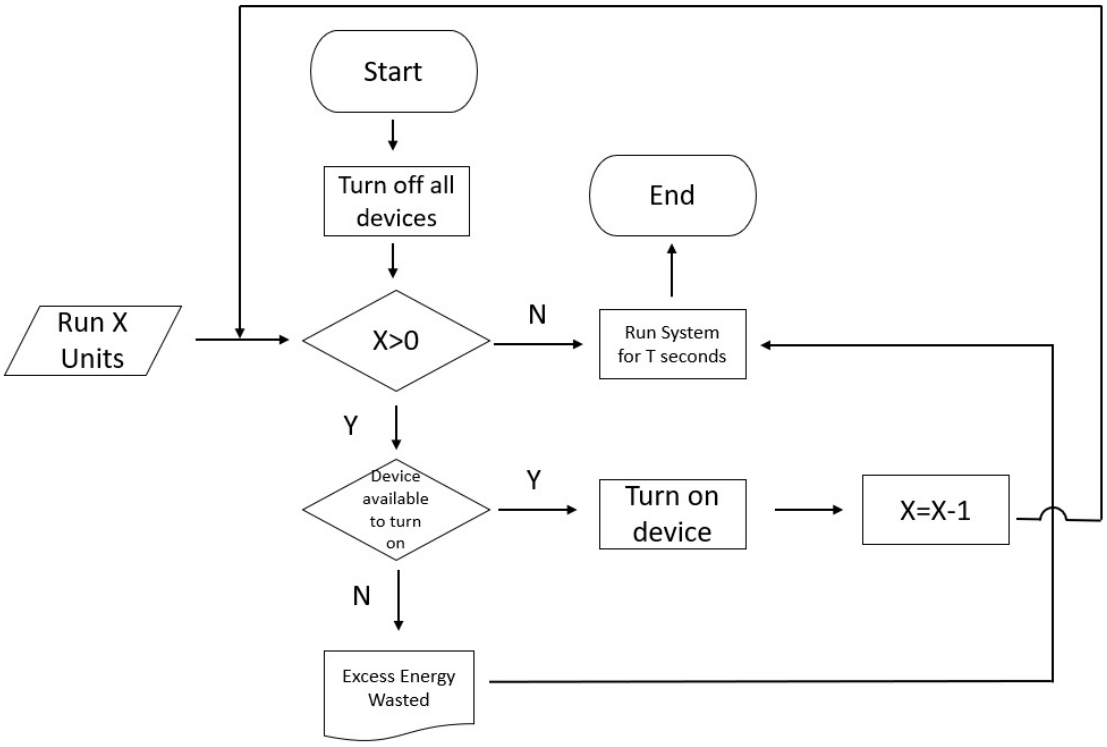



Figure 5.5: How power is distributed in the system.

The Raspberry Pi network was installed over a period of one month, and then run for two months. Over this time frame, the system only lost power twice, once when the system was partially implemented and once when the system was fully operational. The power outages were discovered several hours after the fact, but in both cases, power was restored within minutes, meaning any timers in the system were not out of sync. Furthermore, these power outages occurred after the worst possible weather conditions, days of cloudy, rainy weather where energy generation is minimal at best. If the system had not been automated, it would have been out of power for hours or days and required a full reset of all the timers.

## Conclusion

Hypothesis 3 postulated that an automated system could adapt to changing weather conditions, reducing power as necessary, without causing a power failure or killing the plants. It was assumed that the main way to kill plants was to remove their access to water, as found during a power failure. The simulation in hypothesis two showed that an automated system would never fully deplete the battery reserve in an ideal scenario. When running an altered simulation that modeled a timed system, it was shown that the system would not experience a shutdown at best, but at least one power failure would occur most months, Fig. 5.1. When a physical system was implemented, this was shown to be true. The fully automated system adapted to changing weather conditions without suffering a major power outage. If the fixed timed system had still been installed it would have suffered a major power outage and would have lost power for several hours until it was reset. Therefore, an automated system can adapt to changing weather conditions without causing a power failure or killing the plants, as was shown in the simulated and physical systems.

# 6. CREATION OF THERMAL BATTERIES FOR SURPLUS ENERGY

Plants survive and thrive in a wide variety of temperatures and conditions because they adapt to their surroundings. Because of this, the system does not need to be kept at exactly optimal conditions. Instead, the system can be kept within a range of growing conditions. The system can then take advantage of this by maintaining the most power-efficient conditions. This can be taken one step further.

Throughout the day, changing weather conditions create energy surpluses and shortages in the system. The control algorithm must then balance demands to make the system run as efficiently as possible. In an energy surplus condition, though, the energy is wasted. This is where the growing conditions come in. The system could store surplus energy in the form of a temperature differential. When the next energy shortfall occurs, power could be diverted from the climate control to other, more critical, subsystems.

*Hypothesis 3: An automated system can store excess energy by turning the system into a thermal battery.*

## Methods and Materials

System automation has several benefits beyond power stability. In addition to improving growing conditions when excess power is available, it could potentially store the energy in the form of a temperature difference or thermal battery. This would be done by cooling gas or liquid and then using that to cool the system later. If it worked, it would be a free form of energy storage.

For the system itself, two strategies were proposed. Strategy one involved cooling the system with a built-in air conditioner. Plants thrive in a temperature range from about 60-80 °F. Normally the system is cooled or heated to either end of

that spectrum. If excess energy were available, though, the air conditioner could cool the system several degrees. If the system experienced a power production shortfall, the air conditioner could be turned off, and the power diverted to another system. The benefit of this strategy is that the air conditioner could remain off until conditions improved or the temperature passed the ideal range. A second, similar option requires installing a water chiller. This chiller would cool the reservoir basins and store the energy there. The primary benefit of this system is that water has a higher density than air, meaning it could store more energy in a smaller space. A side benefit is that it would improve plants' growing conditions because plants prefer their roots to be cooler than the surrounding air. Once the system was set up, its efficacy would be tested.

The system's efficacy was tested three ways. First, the system was cooled for an hour and the amount of energy used to cool it, along with the final temperature, were recorded. Second, the energy needed to maintain the system at an acceptable temperature was recorded. This determined the energy savings for every hour the cooling system remained off. Third, the cooling system was turned off and the amount of time it remained off was recorded.

## Results

While two thermal battery options were considered, installing a water chiller or cooling the air using the air conditioner, only the latter option utilizing the air conditioner was used. This was done primarily due to time and energy constraints. Fig. 6.1 shows the modified control algorithm for creating a thermal battery. To test the efficiency of the air conditioner, two tests were run. The first test measured how cool the system got running the air conditioner at full power for an hour. In trial one, the starting temperature was 88 °F and the ending temperature was 90 °F In trial two, the starting temperature was 99 °F, and the ending temperature was 99

°F Over this period, the air conditioner used approximately 0.85 kWh of power. While the air conditioner failed to cool the system, this did show the energy required to maintain the system at a fixed temperature on an average day.

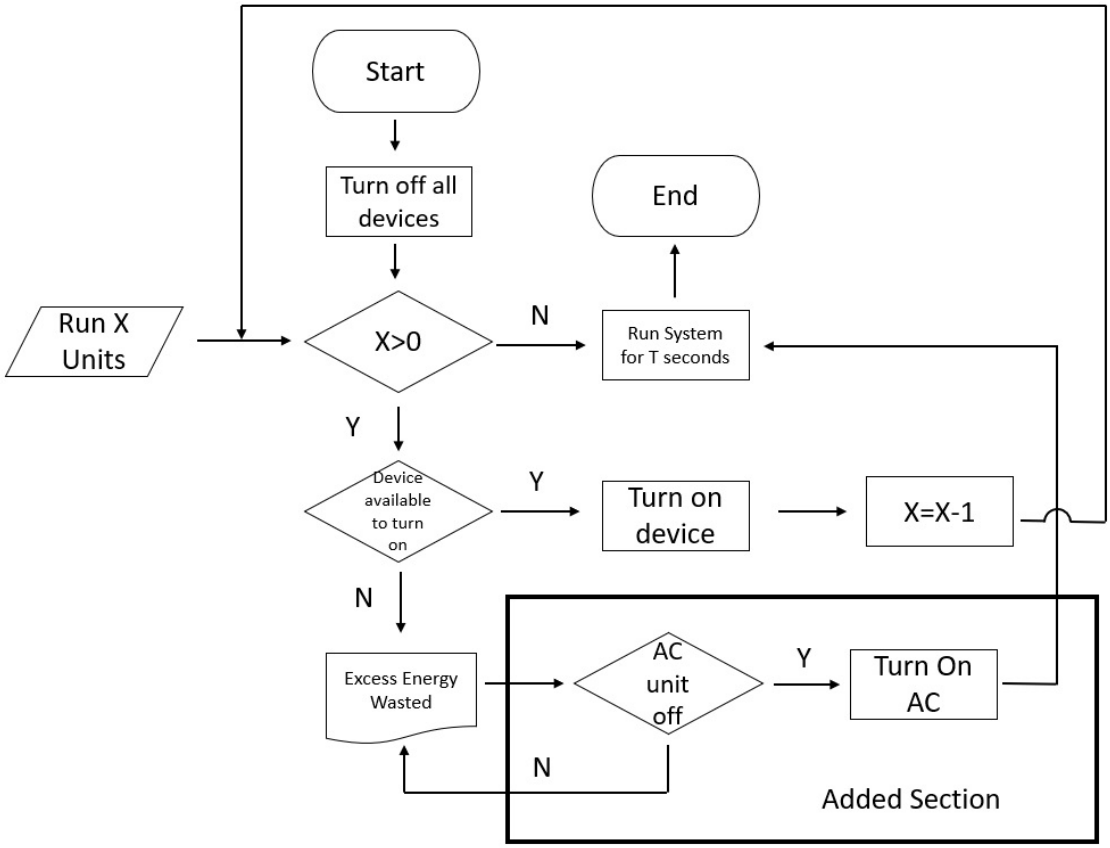



Figure 6.1: Modified power control system with a thermal battery.

## Conclusion

Hypothesis 4 postulated that an automated system could store excess energy by turning the system into a thermal battery. This thermal battery would keep the system cool during an energy shortage and allow the AC to be off. Initial experiments showed that the air conditioning system could not directly cool the EverGreen system. However, further experiments showed that the creation of a thermal battery was plausible if the air conditioning system was improved. A cooled

system stores a significant amount of thermal energy that is slowly used once the air conditioner is turned off. To further improve this system, two areas should be improved. First, the air conditioning system should be improved. This can be done by installing ductwork to route the chilled air throughout the system and by installing a second air conditioning unit. Second, the installation of a water chiller could store thermal energy in the water reservoirs of the plants, further cooling the system. Both of these suggestions are expanded upon in chapter seven.

# 7. FUTURE WORK

Future work will focus on two key areas of the system, system automation, and the cooling subsystem.

## System Automation

The EverGreen system is automated using a network of Raspberry Pi microcomputers. For a commercial system, a better solution would involve the use of industrial-grade PLCs. Industrial grade PLCs are more resilient to harsh environments, such as those found in the system. Furthermore, the PLCs could directly control the circuit breakers instead of individual outlets, reducing system complexity.

Another benefit of switching to an industrial-grade PLC is their ability to communicate wirelessly. While the system is mostly wireless, with individual components acting as islands of autonomy, it could go further. Wireless temperature sensors could be added to the system. Other wires would be removed when the PLCs were connected to circuit breakers instead of individual outlets.

One final improvement for system automation would be integrating it to cloud computing. Currently, the system is designed to be off-grid without any connections to the internet or electric grid. If internet connections were available though, data could be sent to a central facility to store and process it.

## Cooling Subsystem

The thermal battery system could be improved in three ways. First, installing a second air conditioner would increase the cooling capacity of the system and allow it to chill the system on demand. Second, installing ductwork on the air conditioning units would increase the efficacy of the system. Instead of cooling the

immediate area around the air conditioner, the chilled air would be more evenly distributed. Finally, a third way to improve the system's thermal battery capacity would be installing a water chiller. The water chiller would serve two purposes. First, it would improve the plants' growing environment by keeping their roots cool, something the plants prefer. Second, the water would act as a heat sink, helping to cool the container. Once these improved systems were installed, the control algorithm could be adjusted to take advantage of this and create a thermal battery.

Another potential way to create a thermal battery would be through passive heat radiation. Passive heat radiation systems work similarly to the cooling fins found on microchips and other electronic devices. In this setup, specialized panels are installed on the roof of the system to reflect heat into space. When done correctly, the panels maintain a temperature slightly lower than that of their surroundings. This allows them to wick away thermal energy from the system, reducing the load placed on the cooling system. Some systems build upon this passive heat radiation by turning the panels into a radiator. Low power pumps circulate water throughout the building, absorbing heat. This water is then pumped through the roof panels where the heat is dissipated.

A final cooling option could be done using an evaporative cooler. Evaporative coolers work by forcing air through a water-saturated filter. As the air passes through the filter, it causes the water to evaporate, cooling the air. While not as precise as a traditional air conditioning unit, it uses significantly less energy. Furthermore, evaporative cooling systems rely on drawing in fresh outside air. This ensures that adequate levels of CO2 are present in the system. Regular air conditioners recirculate the air in the system, meaning that CO2 levels can become depleted. Two shortfalls that must be addressed in implementing this system are that their effectiveness goes down as humidity levels increase and can only be used when excess water is available.

## APPENDIX SECTION

### Appendix A: Control Algorithms

#### A.1: Vent Fan Algorithm

```
#Program Start
import os
import glob
import time
import RPi.GPIO as GPIO
#The GPIO library is used to turn off and on specific GPIO pins
    n the raspberry pi
sensors ={"28-0114551239aa":0,"28-011455393eaa":0}
#The sensors listed are the specific sensors that we are using.
# if you switch them you will need to run a few commands in the
    terminal of the pi to
#to figure out the new serial numbers of the sensors
#Below is the link to the guide we used for setting up the
   temperature sensor
#https://thepihut.com/blogs/raspberry-pi-tutorials/18095732-
   sensors-temperature-with-the-1-wire-interface-and-the-ds18b
   20


import datetime


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
```

```
#The GPIO pin is 18 that we have in the set up. We turn this
   pin off/on during the program
counter = 0
#The counter switches between the two sensors. If we end up
   adding a third you will need to
# change the logic of the if statement below
while 1:
    for sensor in sensors:
        tempfile = open("/sys/bus/w1/devices/{0}/w1_slave".
           format(sensor))
        # this goes through the directory and goes all the way
           to the command that will read in the temperature
        temptext = tempfile.read()
        # temperature is read
        tempfile.close()
        # the file is closed to not cause errors in readings
        tempcelsius = temptext.split("\n")[1].split(" ")[9]
        temperature = float(tempcelsius[2:])
        temperature = temperature / 1000
        # temperature is converted into a readable float and
           then divided by 1000 to match the local temperature
        farenheit = (temperature * 9 / 5) + 32
        # temperature is then converted to farenheit
        # The if else statement below is to get a reading from
           the one sensor and storing it, then
        # adding one to counter to show that we have the first
           reading. During the second iteration in the loop for
```

```
    # sensor 2 it will not overwrite temp1 and will store
       it in temp2. That way we have both sensors updating
    # consistently
    if counter == 0:
        # OT is the outside temperature. That sensor is
           marked with black electrical tape on the end of
           it
        OT = farenheit
        counter = counter + 1
    else:
        counter = counter - 1
    sensors[sensor] = farenheit


print (' '.join(['{:2.4f}, '.format(v) for k,v in sensors.
   items()]))
# IT is inside temperature. The sensor is marked with a
   piece of blue electrical tape on the end of it.
IT = farenheit
# We then check if the outside temperature is greater than
   the inside temperature and if so we turn the fan off
# otherwise we run the fan. Afterwards we wait 5 minutes
   (300 seconds) till we get the next reading. We also
   record
# the temperature and time and put that into the file for
   debugging and data purposes
# The two elif statements are for when the sensors do not
   get a reading (32 and 185 being the errors)
```

```
    if OT < IT+3 and IT>55:
        currentDT = datetime.datetime.now()
        f=open("tempData4.txt","a+")
        f.write(str(currentDT))
        f.write(" OT: ")
        f.write(str(OT))
        f.write(" , IT: ")
        f.write(str(IT))
        f.write(" , off\n")
        f.close()
        GPIO.output(18,GPIO.LOW)
    elif OT == 32 and IT == 32:
        f=open("tempData4.txt","a+")
        f.write("OT: ")
        f.write(str(OT))
        f.write(" , IT: ")
        f.write(str(IT))
        f.write(" , no read \n")
        f.close()
    elif OT == 185 or IT == 185:
        f=open("tempData4.txt","a+")
        f.write("OT: ")
        f.write(str(OT))
        f.write(" , IT: ")
        f.write(str(IT))
        f.write(" , no read \n")
        f.close()
    else:
```

```
        f=open("tempData4.txt","a+")
        f.write("OT: ")
        f.write(str(OT))
        f.write(" , IT: ")
        f.write(str(IT))
        f.write(" , on\n")
        f.close()
        GPIO.output(18,GPIO.HIGH)
    time.sleep(300)
#Program Finish
```

A.2: Main Control Algorithm

```
#Program Start
import time #import libraries
import datetime
import minimalmodbus
import serial
import subprocess
import schedule



TimeOn=500 #Set how long you want each light to be on each day.
    In minutes
instrument=0 #set variables and counters for each process, pi
   and light
eCounter = 0 #Counts how many lights are on at once.


strawberryL = 0 #How long each light has been on. Initially set
    to 0.
strawberryR = 0


missMudL = 0
missMudR = 0


GCL = 0
GCR = 0


BBL = 0
BBR = 0
```

```
AC =0
MaxCur=2 #Sets maximum amount of current you want to flow form
   the battery pack. 2 is safe 0 is min.
MaxEcount=9 #Set maximum number of lights you want on. Max
   system can handle is 12, AC counts as one of 12 as well.
MaxPowerStor=70 #What's the maximum amount of power you want to
    store overnight(%of battery charge) Min 30 max 90
def reset():
    global strawberryL #ssh number 192.168.0.200
    global strawberryR
    global missMudL #ssh number 192.168.0.203
    global missMudR
    global GCL
    global GCR
    global BBL
    global BBR
    s=open("Daily Power use Overview.csv","a+") #reset
       procedure.
    s.write(str(currentDT)) #Records how long lights were on
       each day (in minutes).
    s.write(",")
    s.write(str(strawberryL)) #Done for each set of lights
    s.write(",")
    s.write(str(strawberryR))
    s.write(",")
    s.write(str(missMudL))
    s.write(",")
    s.write(str(missMudR))
```

```
    s.write("\n")
    s.close()
    strawberryL = 0 #reset all parameters
    strawberryR = 0
    missMudL = 0
    missMudR = 0
    GCL = 0
    GCR = 0
    BBL = 0
    BBR = 0
    print("RESET") #print confirmation that it is done

schedule.every().day.at("01:00").do(reset) #when we reset the
   parameters
print(instrument)
instrument = minimalmodbus.Instrument('/dev/serial0', 201)#201=
   combox 170=mppt0,173=mppt1, 171=mppt2, 172=mppt3

instrument.serial.port                     # this is the serial
    port name
instrument.serial.baudrate = 19200       # Baud ## Initially
   set at 19200, got better error when switched to 38400,
   normal baud rate for most stuff
instrument.serial.bytesize = 8
instrument.serial.parity   = serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout  = 2          # seconds
```

```
instrument.address                          # this is the slave
   address number
instrument.mode = minimalmodbus.MODE_RTU   # rtu or ascii mode
instrument.clear_buffers_before_each_transaction = True
print(instrument)

mppt=0
print(instrument)
mppt = minimalmodbus.Instrument('/dev/serial0', 170)#201=combox
    170=mppt0,173=mppt1, 171=mppt2, 172=mppt3

mppt.serial.port                      # this is the serial port
   name
mppt.serial.baudrate = 19200       # Baud ## Initially set at
   19200, got better error when switched to 38400, normal baud
   rate for most stuff
mppt.serial.bytesize = 8
mppt.serial.parity   = serial.PARITY_NONE
mppt.serial.stopbits = 1
mppt.serial.timeout  = 2          # seconds

mppt.address                          # this is the slave
   address number
mppt.mode = minimalmodbus.MODE_RTU   # rtu or ascii mode
mppt.clear_buffers_before_each_transaction = True
print(mppt)

mppt03 = minimalmodbus.Instrument('/dev/serial0', 172)#201=
   combox 170=mppt0,173=mppt1, 171=mppt2, 172=mppt3
```

```
mppt03.serial.port                          # this is the serial
   port name
mppt03.serial.baudrate = 19200       # Baud ## Initially set at
    19200, got better error when switched to 38400, normal baud
    rate for most stuff
mppt03.serial.bytesize = 8
mppt03.serial.parity   = serial.PARITY_NONE
mppt03.serial.stopbits = 1
mppt03.serial.timeout  = 2          # seconds

mppt03.address                              # this is the slave
   address number
mppt03.mode = minimalmodbus.MODE_RTU   # rtu or ascii mode
mppt03.clear_buffers_before_each_transaction = True
print(mppt03)
i=1


data=0
'''
try:
    print(instrument.read_register(0x0062))
except IOError:
    print("Failed to read from instrument", i)'''


data= instrument.read_register(0x0063,0)
data1= instrument.read_register(0x0049,0)
data2=mppt.read_register(0x0049)
data3=mppt.read_register(0x005C)
```

```
data4=mppt03.read_register(0x0049)
data5=mppt03.read_register(0x005C)
print(data1)
print(data2)
print(data3)

j = 1

currentDT = datetime.datetime.now() #write out the header for
   the data collection
f=open("testreadmodbusmppt2.csv","a+")
f.write("Time, mppt this is skewedPVHarvest(W), DC Charging
   Power (W), DC Inverting Power (W)")
f.write(", Load Output Power (W), Load Current (A), Battery
   Voltage, Battery Current Net (A)")
f.write(", Battery Power Net (W), Battery SOC (%), Battery
   Capacity Remaining (Ah),MPPT00State(section 6), MPPT00
   OutputPower(W), MPPT03OutputPower(W)\n")
f.close()

while i<=1: #infinite while loop, should probably use something
    else besides i
    schedule.run_pending()
    try: #Solar panel system resets at 3AM this is to prevent
       the system from freaking out.
        currentDT = datetime.datetime.now() #collecting all of
           the data for that moment
        PVHarvest= instrument.read_register(0x0044,0)
        DCChargeP= instrument.read_register(0x0046,0)
```

```
DCInvertP= instrument.read_register(0x004A,0)
LoadOutputP= instrument.read_register(0x005E,0)
LoadCurrent= instrument.read_register(0x0068,3)
BatteryV=instrument.read_register(0x0098,3)
BattCurrentNet= instrument.read_register(0x009C,3)
BattPNet=instrument.read_register(0x0158,0)
BatterySOC= instrument.read_register(0x03C8,0)
BattCapRem= instrument.read_register(0x03CA,0)
mpptstate=mppt.read_register(0x0049)
MPPT00output=mppt.read_register(0x005C)
data2=mppt.read_register(0x0049)
data3=mppt.read_register(0x005C)
MPPT03output=mppt03.read_register(0x005C)
BBinvertC=instrument.read_register(0x020A,3)
data5=mppt03.read_register(0x005C)
print("Estimated % of battery left")
print(BatterySOC)
print("Battery Current Net")
print(BattCurrentNet)
print("Battery Invert current")
print(BBinvertC)
f=open("testreadmodbusmppt2.csv","a+") #Saving data to
   a file
f.write(str(currentDT))
f.write(",")
f.write(str(PVHarvest))
f.write(",")
f.write(str(DCChargeP))
f.write(",")
```

```
        f.write(str(DCInvertP))
        f.write(",")
        f.write(str(LoadOutputP))
        f.write(",")
        f.write(str(LoadCurrent))
        f.write(",")
        f.write(str(BatteryV))
        f.write(",")
        f.write(str(BattCurrentNet))
        f.write(",")
        f.write(str(BattPNet))
        f.write(",")
        f.write(str(BatterySOC))
        f.write(",")
        f.write(str(BattCapRem))
        f.write(",")
        f.write(str(mpptstate))
        f.write(",")
        f.write(str(MPPT00output))
        f.write(",")
        f.write(str(MPPT03output))
        f.write("\n")
        f.close()
        if int(BBinvertC) <= MaxCur and eCounter < MaxEcount: #
           adjust counter for how many lights should be on at a
            given moment. Can also be used to control air-con.
            eCounter = eCounter + 1 #Max number of lights
               should be limited 12. Inverter has ~ 7kW cap
               each light takes 600W, AC 700W @ full blast
```

```
        print("Turn on one light") #Turn on one at a time
           until maxed out. 7.3 kW is kind of limit. Can
           run up to 8kW for short amounts of time
        print("Number of lights on, eCounter, ",eCounter)
    elif int(BBinvertC) > MaxCur and eCounter > 0: #before
       it overheats. Pumps and fan are negligible. Turn off
        lights to prevent system from
        eCounter = eCounter - 1 #fully discharging
           batteries and running out of power.
        print("Turn off one light")
        print("Number of lights on, eCounter, ",eCounter)
    else:
        print("No change in the number of lights on")
        print("Number of lights on, eCounter, ",eCounter)
    numLOn = eCounter #Use this to count off how many
       lights are to be on.
    #If statements regarding battery usage for the system.
       Hard limits for when and where power can be used.
       Could do as a nested if statement.
    if int(BatterySOC)<=10:
        numLOn=0 #Emergency power
        print("State 1")
    elif int(BatterySOC)<30 and int(BatterySOC)>10:
        #Can run AC if critical
        if int(BBinvertC)>0:
            numLOn=0
            print("State 2.1")
        elif int(BBinvertC)==0:
            numLon=eCounter/2
```

```
        print("State 2.2")
elif int(BatterySOC)>=30 and int(BatterySOC)<=
   MaxPowerStor:
    #run AC and fans. Regular numLOn for system if
       charging
    numLOn = eCounter
    print("State 3")
elif int(BatterySOC)>MaxPowerStor and int(BatterySOC)
   <90: #Want system to run down to 50% power each day.
    if MaxEcount>(2*eCounter): #If depleting battery
       run system at half power until user set battery.
        numLOn=MaxEcount/2
        print("State 4.1")
    else:
        numLOn = eCounter #If have energy coming in run
            at regular power.
        print("State 4.2")
elif int(BatterySOC)>=90:
    numLOn=MaxEcount
    print("State 5")
print("Actual number of lights on, numLOn, ",numLOn)


r=open("AC.txt", "r")
AC = r.read()
r.close()
AC = int(AC)
if AC == 1 and numLOn > 0:
    numLOn = numLOn - 1
    l = open("ACD.txt", "w+")
```

```
            l.write(str(1))
            l.close()
            #print("Output StrawberryL ON and ON for",
               strawberryL, "minutes")
        elif AC == 0 or numLOn<=0:
            l = open("ACD.txt", "w+")
            l.write(str(0))
            l.close()

        if numLOn > 0 and GCL < TimeOn+1:
            numLOn = numLOn - 1
            GCL = GCL + 1
            l = open("GCL.txt","w+")
            l.write(str(1))
            print("Output German ChoclateL ON and ON for", GCL,
                "minutes")
            l.close()
            if GCL >= TimeOn:
                l=open("workingGC.txt", "a+")
                l.write("German ChoclateL Full at,")
                l.write(str(currentDT))
                l.write("\n")
                l.close()
        elif numLOn <= 0 or GCL>=TimeOn:
            l = open("GCL.txt","w+")
            l.write(str(0))
            print("Output German ChoclateL OFF but was on for",
                GCL,"minutes today")
            l.close()
```

```
if numLOn > 0 and GCR < TimeOn+1:
    numLOn = numLOn - 1
    GCR= GCR + 1
    l = open("GCR.txt","w+")
    l.write(str(1))
    print("Output German ChoclateR ON and ON for", GCR,
        "minutes")
    l.close()
    if GCR >= TimeOn:
        l=open("workingGC.txt", "a+")
        l.write("German Choclate R full at,")
        l.write(str(currentDT))
        l.write("\n")
        l.close()
elif numLOn <= 0 or GCR>=TimeOn:
    l = open("GCR.txt","w+")
    l.write(str(0))
    print("Output German ChoclateR OFF but was on for",
        GCR,"minutes today")
    l.close()


if numLOn > 0 and BBL < TimeOn+1:
    numLOn = numLOn - 1
    BBL = BBL + 1
    l = open("BBL.txt","w+")
    l.write(str(1))
    print("Output Bumble BerryL ON and ON for", BBL, "
       minutes")
```

```
        l.close()
        if BBL >= TimeOn:
            l=open("workingBB.txt", "a+")
            l.write("Bumble BerryL Full at,")
            l.write(str(currentDT))
            l.write("\n")
            l.close()
    elif numLOn <= 0 or BBL>=TimeOn:
        l = open("BBL.txt","w+")
        l.write(str(0))
        print("Output Bumble BerryL OFF but was on for",
           BBL,"minutes today")
        l.close()

    if numLOn > 0 and BBR < TimeOn+1:
        numLOn = numLOn - 1
        BBR = BBR + 1
        l = open("BBR.txt","w+")
        l.write(str(1))
        print("Output Bumble BerryR ON and ON for", BBR, "
           minutes")
        l.close()
        if BBR >= TimeOn:
            l=open("workingBB.txt", "a+")
            l.write("Bumble BerryR full at,")
            l.write(str(currentDT))
            l.write("\n")
            l.close()
    elif numLOn <= 0 or BBR>=TimeOn:
```

```
        l = open("BBR.txt","w+")
        l.write(str(0))
        print("Output Bumble BerryR OFF but was on for",
           BBR,"minutes today")
        l.close()


    if numLOn > 0 and strawberryL < TimeOn+1: #List of
       lights and order of importance. Repeat for every set
        of lights that are installed.
        numLOn = numLOn - 1 #Subtract one if turning on a
           light.
        strawberryL = strawberryL + 1 #Counter for how long
            light has been on. Resets at 1am or user
           specified time.
        l = open("strawL.txt","w+") #Save on or off info to
            unique file. This is then sshed in to the
           appropriate pi
        l.write(str(1)) #Lights are set to be normally off.
            A 1 turns on the light, 0 turns off the light.
        print("Output StrawberryL ON and ON for",
           strawberryL, "minutes") #quick check of which
           lights should be on.
        l.close()
        if strawberryL >= TimeOn:
            l=open("workingStraw.txt", "a+")
            l.write("Strawberry L was Full at, ")
            l.write(str(currentDT))
            l.write("\n")
```

```
        l.close()
elif numLOn <= 0 or strawberryL>=TimeOn: #Turn off if
   it has been on so long or if there is insufficient
   energy coming in.
    l = open("strawL.txt","w+")
    l.write(str(0))
    print("Output StrawberryL OFF but was on for",
       strawberryL,"minutes today")
    l.close()

if numLOn > 0 and strawberryR < TimeOn+1: #List of
   lights and order of importance. Repeat for every set
    of lights that are installed.
    numLOn = numLOn - 1 #Subtract one if turning on a
       light.
    strawberryR = strawberryR + 1 #Counter for how long
        light has been on. Resets at 1am or user
       specified time.
    l = open("strawR.txt","w+") #Save on or off info to
        unique file. This is then sshed in to the
       appropriate pi
    l.write(str(1)) #Lights are set to be normally off.
        A 1 turns on the light, 0 turns off the light.
    print("Output StrawberryR ON and ON for",
       strawberryR, "minutes") #quick check of which
       lights should be on.
    l.close()
    if strawberryR >= TimeOn:
        l=open("workingStraw.txt", "a+")
```

```
            l.write("Strawberry R was Full at, ")
            l.write(str(currentDT))
            l.write("\n")
            l.close()
    elif numLOn <= 0 or strawberryR>=TimeOn: #Turn off if
       it has been on so long or if there is insufficient
       energy coming in.
        l = open("strawR.txt","w+")
        l.write(str(0))
        print("Output StrawberryR OFF but was on for",
           strawberryR,"minutes today")
        l.close()


    if numLOn > 0 and missMudL < TimeOn+1:
        numLOn = numLOn - 1
        missMudL= missMudL + 1
        l = open("mmL.txt","w+")
        l.write(str(1))
        print("Output Mississippi MudL ON and ON for",
           missMudL, "minutes")
        l.close()
        if missMudL >= TimeOn:
            l=open("workingMM.txt", "a+")
            l.write("Mississippi Mud L Full at,")
            l.write(str(currentDT))
            l.write("\n")
            l.close()
    elif numLOn <= 0 or missMudL>=TimeOn:
```

```
        l = open("mmL.txt","w+")
        l.write(str(0))
        print("Output Mississippi MudL OFF but was on for",
            missMudL,"minutes today")
        l.close()

    if numLOn > 0 and missMudR < TimeOn+1:
        numLOn = numLOn - 1
        missMudR= missMudR + 1
        l = open("mmR.txt","w+")
        l.write(str(1))
        print("Output Mississippi MudR ON and ON for",
           missMudR, "minutes")
        l.close()
        if missMudR >= TimeOn:
            l=open("workingMM.txt", "a+")
            l.write("Mississippi mud R full at,")
            l.write(str(currentDT))
            l.write("\n")
            l.close()
    elif numLOn <= 0 or missMudR>=TimeOn:
        l = open("mmR.txt","w+")
        l.write(str(0))
        print("Output Mississippi MudR OFF but was on for",
            missMudR,"minutes today")
        l.close()



    else:
```

```
        numLOn=0


    #print(currentDT)
   # l = open("sshData.txt","w+")
    #if int(BBinvertC) >= 2:
     #   l.write(str(0))
     #   print("Output 0 lights off")
    #elif int(BBinvertC) < 2:
       #  l.write(str(1))
        # print("Output 1 lights on")
   # else:
       #  print("Error")
    #l.close()
    subprocess.run(["scp","ACD.txt", "pi@192.168.0.206:/
       home/pi/Desktop/Evergreen/ACDC.txt"])
    subprocess.run(["scp","strawL.txt", "pi@192.168.0.200:/
       home/pi/Desktop/Evergreen/hi3.txt"]) #Sends results
       to individual pis.
    subprocess.run(["scp","strawR.txt", "pi@192.168.0.200:/
       home/pi/Desktop/Evergreen/hiR.txt"])
    subprocess.run(["scp","mmL.txt", "pi@192.168.0.203:/
       home/pi/Desktop/Evergreen/hi4.txt"])
    subprocess.run(["scp","mmR.txt", "pi@192.168.0.203:/
       home/pi/Desktop/Evergreen/hiMR.txt"])
    subprocess.run(["scp","GCL.txt", "pi@192.168.0.202:/
       home/pi/Desktop/Evergreen/hiGCL.txt"])
    subprocess.run(["scp","GCR.txt", "pi@192.168.0.202:/
       home/pi/Desktop/Evergreen/hiGCR.txt"])
```

```
    subprocess.run(["scp","BBL.txt", "pi@192.168.0.201:/
       home/pi/Desktop/Evergreen/hiBBL.txt"])
    subprocess.run(["scp","BBR.txt", "pi@192.168.0.201:/
       home/pi/Desktop/Evergreen/hiBBR.txt"])



    time.sleep(60) #Sleep for x amount of time.
except IOError: #Used if there is a problem to keep system
   from crashing.
    print("Failed to read from instrument")
    f=open("testreadmodbusmppt2.csv","a+")
    f.write(str(currentDT))
    f.write(",")
    f.write(str(PVHarvest))
    f.write(",")
    f.write(str(DCChargeP))
    f.write(",")
    f.write(str(DCInvertP))
    f.write(",")
    f.write(str(LoadOutputP))
    f.write(",")
    f.write(str(LoadCurrent))
    f.write(",")
    f.write(str(BatteryV))
    f.write(",")
    f.write(str(BattCurrentNet))
    f.write(",")
    f.write(str(BattPNet))
    f.write(",")
```

```
        f.write(str(BatterySOC))
        f.write(",")
        f.write(str(BattCapRem))
        f.write(",")
        f.write(str(mpptstate))
        f.write(",")
        f.write(str(MPPT00output))
        f.write(",")
        f.write(str(MPPT03output))
        f.write(", error on this reading \n")
        f.close()
        print(currentDT)
        time.sleep(60)
#Program Finish
```

A.3: Light Control Algorithm

```
#Program Start
import os
import glob
import time
import RPi.GPIO as GPIO
import datetime


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)


hi = 0
hiR = 0
fRun = 0
f = open("hi3.txt","w")
f.write("0")
f.close()
while 1:


    f = open("hi3.txt","r")
    hi = f.read()
    f.close()
    print(hi)
    hi=int(hi)
    if fRun == 0:
        GPIO.output(18,GPIO.LOW)
```

```
        GPIO.output(27,GPIO.LOW)
        fRun = fRun + 1
    else:
        if hi == 0:
            GPIO.output(18,GPIO.LOW)

        elif hi == 1:
            GPIO.output(18,GPIO.HIGH)

    f = open("hiR.txt","r")
    hiR = f.read()
    f.close()
    hiR=int(hiR)
    if fRun == 0:
        GPIO.output(18,GPIO.LOW)
        GPIO.output(27,GPIO.LOW)
        fRun = fRun + 1
    else:
        if hiR == 0:
            GPIO.output(27,GPIO.LOW)

        elif hiR == 1:
            GPIO.output(27,GPIO.HIGH)



    time.sleep(8)
#Program Finish
```

A.4: AC On/Off Control

```
import os
import glob
import time
import RPi.GPIO as GPIO
import datetime


GPIO.setmode(GPIO.BCM)https://www.overleaf.com/project/5ecfd
   9771a84a80001772be5
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
hi = 0
counter = 1
fRun = 0
f = open("ACDC.txt","w")
f.write("0")
f.close()
while 1:


    f = open("ACDC.txt","r")
    hi = f.read()
    f.close()
    print(hi)
    hi=int(hi)
    if fRun == 0:
        GPIO.output(18,GPIO.LOW)
        fRun = fRun + 1
    else:
```

```
        if hi == 0:
            GPIO.output(18,GPIO.LOW)
            counter = counter - 1
        elif hi == 1:
            GPIO.output(18,GPIO.HIGH)
            counter = counter + 1


    time.sleep(8)
```

A.5: AC Control Algorithm

```
#AC code
import os
import glob
import time
import subprocess
import RPi.GPIO as GPIO
#The GPIO library is used to turn off and on specific GPIO pins
    n the raspberry pi
sensors ={"28-01191c219af2":0}
#The sensors listed are the specific sensors that we are using.
# if you switch them you will need to run a few commands in the
    terminal of the pi to
#to figure out the new serial numbers of the sensors
#Below is the link to the guide we used for setting up the
   temperature sensor
#https://thepihut.com/blogs/raspberry-pi-tutorials/18095732-
   sensors-temperature-with-the-1-wire-interface-and-the-ds18b
   20
# BW, MW, DR, RS
import datetime


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
#The GPIO pin is 18 that we have in the set up. We turn this
   pin off/on during the program
counter = 0
```

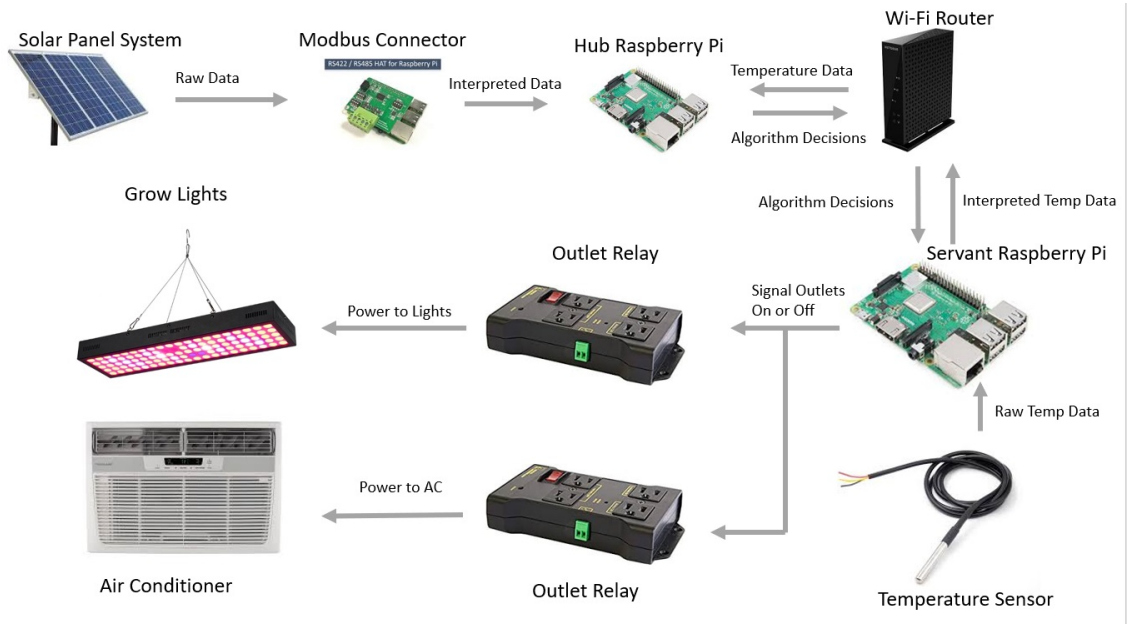
```
count = 1
#The counter switches between the two sensors. If we end up
   adding a third you will need to
# change the logic of the if statement below
while 1:
    for sensor in sensors:
        tempfile = open("/sys/bus/w1/devices/{0}/w1_slave".
           format(sensor))
        # this goes through the directory and goes all the way
           to the command that will read in the temperature
        temptext = tempfile.read()
        # temperature is read
        tempfile.close()
        # the file is closed to not cause errors in readings
        tempcelsius = temptext.split("\n")[1].split(" ")[9]
        temperature = float(tempcelsius[2:])
        temperature = temperature / 1000
        # temperature is converted into a readable float and
           then divided by 1000 to match the local temperature
        farenheit = (temperature * 9 / 5) + 32
        # temperature is then converted to farenheit
        # The if else statement below is to get a reading from
           the one sensor and storing it, then
        # adding one to counter to show that we have the first
           reading. During the second iteration in the loop for
        # sensor 2 it will not overwrite temp1 and will store
           it in temp2. That way we have both sensors updating
        # consistently
```

```
#print (' '.join(['{:2.4f}, '.format(v) for k,v in sensors.
   items()]))
# IT is inside temperature. The sensor is marked with a
   piece of blue electrical tape on the end of it.
AC = farenheit
print(farenheit)
if AC>75 and AC!=185:
    l=open("sData.txt","w+")
    l.write(str(1))
    l.close()
else:
    l=open("sData.txt","w+")
    l.write(str(0))
    l.close()
subprocess.run(["scp","sData.txt", "pi@192.168.0.116:/home/
   pi/Desktop/AC.txt"])



currentDT = datetime.datetime.now()
f=open("tdata.txt","a+")
f.write(str(currentDT))
f.write(str(AC))
f.close()
# We then check if the outside temperature is greater than
   the inside temperature and if so we turn the fan off
```

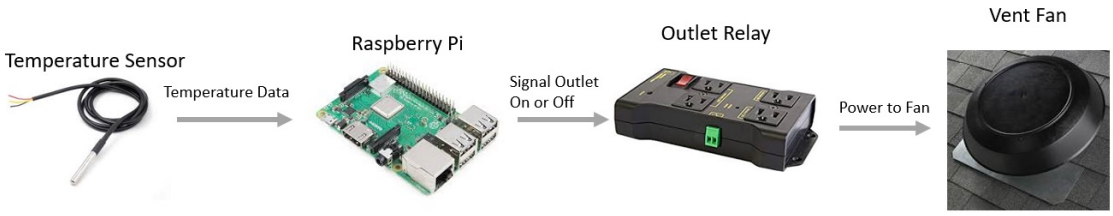
```
    # otherwise we run the fan. Afterwards we wait 5 minutes
       (300 seconds) till we get the next reading. We also
       record
    # the temperature and time and put that into the file for
       debugging and data purposes
    # The two elif statements are for when the sensors do not
       get a reading (32 and 185 being the errors)
    #if OT < IT+3 and IT>55:



    count = count+1
    time.sleep(300)
```

# Appendix B: Hardware Layout

## B.1: How the Main System Is Connected

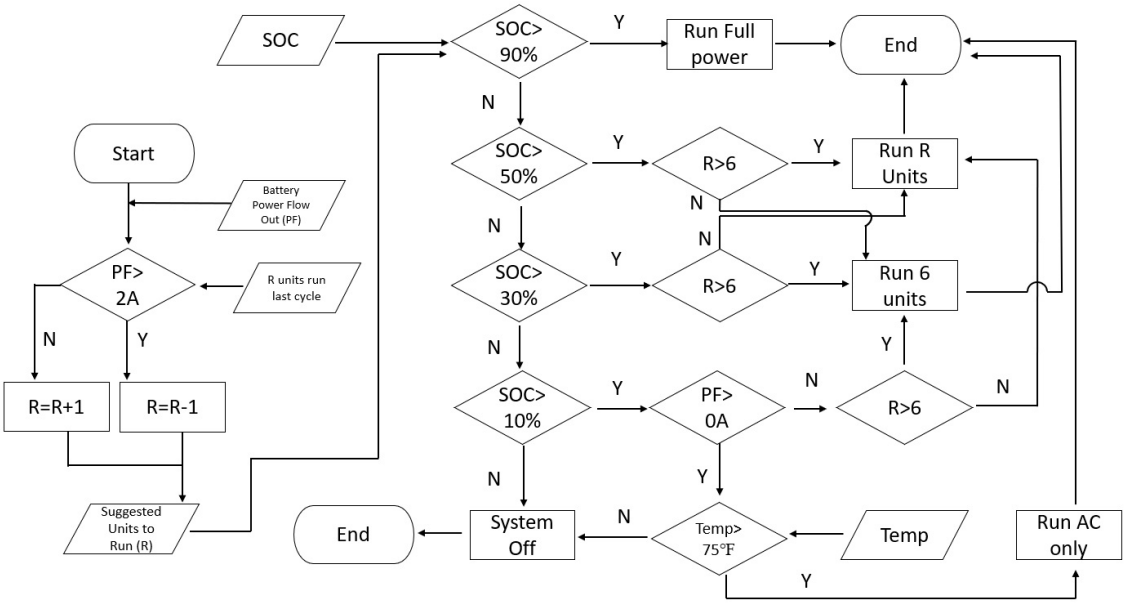



## B.2: Fan Control System

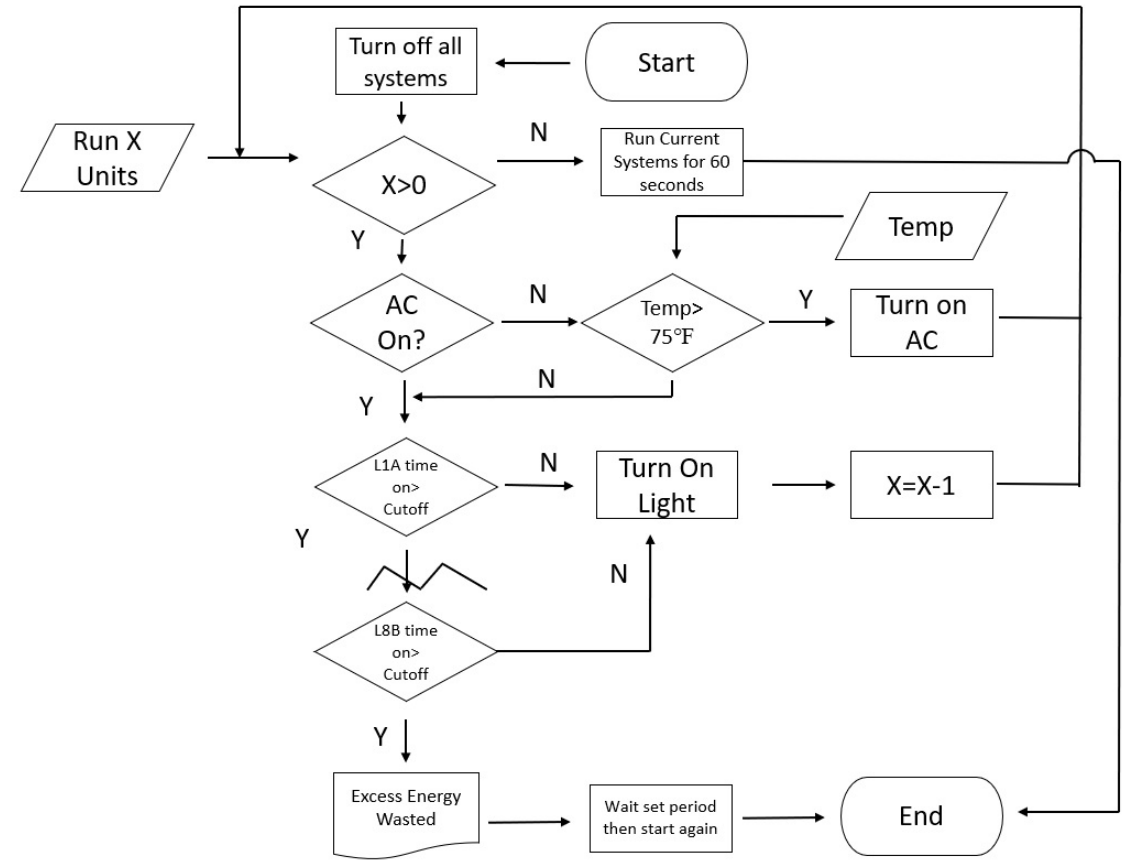


# Appendix C: Full Flow Charts

## C.1: How Many Subsystems To Run



C.2: Which Subsystems To Run



## REFERENCES


Aishwarya, K. S., Harish, M., Prathibhashree, S., & Panimozhi, K. (2018). Survey on Automated Aquponics Based Gardening Approaches. Paper presented at the Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018, Coimbatore, India.

Almusaied, Z., & Asiabanpour, B. (2017). Atmospheric Water Generation: Technologies and Influential Factors. Paper presented at the 2017 Industrial and Systems Engineering Conference, Pittsburgh, Pennsylvania.

Angelakis, V., Avgouleas, I., Pappas, N., Fitzgerald, E., & Yuan, D. (2016). Allocation of Heterogeneous Resources of an IoT Device to Flexible Services. In (Vol. 3, pp. 691-700).

Asiabanpour, B., Almusaied, Z., Rainosek, K., & Davidson, K. (2019). A comparison between simulation and empirical methods to determine fixed versus sun-tracking photovoltaic panel performance. International Journal of Computer Applications in Technology, 60(1), 37-50. doi:10.1504/IJCAT.2019.099504

Asiabanpour, B., Ownby, N., Summers, M., & Moghimi, F. (2019). Atmospheric Water Generation and Energy Consumption: An Empirical Analysis. Paper presented at the 2019 IEEE Texas Power and Energy Conference (TPEC), College Station, TX.

Bandong, S., Leksono, E., Purwarianti, A., & Joelianto, E. (2019). Performance Ratio Estimation and Prediction of Solar Power Plants Using Machine Learning to Improve Energy Reliability. Paper presented at the 2019 6th International Conference on Instrumentation, Control, and Automation (ICA), Bandung, Indonesia.

Belhekar, P., Thakare, A., Budhe, P., Shinde, U., & Waghmode, V. (2018). Automated System for Farming with Hydroponic Style. Paper presented at the Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India.

bin Ismail, M. I. H., & Thamrin, N. M. (2017). IoT implementation for indoor vertical farming watering system. Paper presented at the International Conference on Electrical, Electronics and System Engineering (ICEESE), Kanazawa, Japan.

Chatterjee, R., & Gamota, D. (2020). The Convergence of Technologies and Standards Across the Electronic Products Manufacturing Industry (SEMI, OSAT, and PCBA) to Realize Smart Manufacturing. Paper presented at the 2020 Pan Pacific Microelectronics Symposium (Pan Pacific), HI, USA. Conference retrieved from

Choi, K.-Y., Choi, E.-Y., Kim, I. S., & Lee, Y.-B. (2016). Improving water and fertilizer use efficiency during the production of strawberry in coir substrate hydroponics using a FDR sensor-automated irrigation system. In (Vol. 57, pp. 431-439).

Das, U. K., Tey, K. S., Seyedmahmoudian, M., Mekhilef, S., Idris, M. Y. I., Van Deventer, W., . . . Stojcevski, A. (2018). Forecasting of photovoltaic power generation and model optimization: A review. Renewable and Sustainable Energy Reviews, 81(Part 1), 912-928. doi:10.1016/j.rser.2017.08.017

Elsokah, M. M., & Sakah, M. (2019). Next Generation of Smart Aquaponics with Internet of Things Solutions. 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 106. doi:10.1109/STA.2019.8717280

EverGreen. (2019). EverGreen Texas State. Retrieved from https://www.evergreen.txstate.edu/

Ferrer, B. R., Mohammed, W. M., Martinez Lastra, J. L., Villalonga, A., Beruvides, A., Castaño, F., & Haber, R. E. (2018). Towards the Adoption of Cyber-Physical Systems of Systems Paradigm in Smart Manufacturing Environments. Paper presented at the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal.

Gentry, M. (2019). Local heat, local food: Integrating vertical hydroponic farming with district heating in Sweden. Energy, 174, 191-197. doi:10.1016/j.energy.2019.02.119

Hermann, M., Pentek, T., & Otto, B. (2016). Design Principles for Industrie 4.0 Scenarios. In (pp. 3928): IEEE.

Ismail, A., Idris, M. Y. I., Ayub, M. N., & Por, L. Y. (2018). Vision-Based Apple Classification for Smart Manufacturing. SENSORS, 18. doi:10.3390/s18124353

Ismail, F., & Gryzagoridis, J. (2016, 2016 / 10 / 21 /). Sustainable development using renewable energy to boost aquaponics food production in needy communities. Paper presented at the Proceedings of the Conference on the Industrial and Commercial Use of Energy, ICUE, Cape Peninsula University of Technology Bellville, South Africa.

Jeong, S., Na, W., Kim, J., & Cho, S. (2018). Internet of Things for Smart Manufacturing System: Trust Issues in Resource Allocation. IEEE Internet of Things Journal, Internet of Things Journal, IEEE, IEEE Internet Things J.(6), 4418. doi:10.1109/JIOT.2018.2814063

Jhuria, M., Kumar, A., & Borse, R. (2013, 2013 / 01 / 01 /). Image processing for smart farming: Detection of disease and fruit grading. Paper presented at the 2013 IEEE 2nd International Conference on Image Information Processing, Shimla, India.

Kagermann, H., Lukas, W., & Wahlster, W. (2011). Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. VDI Nachrichten, 13.

Khastieva, D., Dimoulkas, I., & Amelin, M. (2018). Optimal Investment Planning of Bulk Energy Storage Systems. Sustainability (2071-1050), 10(3), 610. doi:10.3390/su10030610

Labrador, C. G., Ong, A. C. L., Baldovino, R. G., Valenzuela, I. C., Culaba, A. B., & Dadios, E. P. (2019, 2019 / 03 / 12 /). Optimization of power generation and distribution for vertical farming with wireless sensor network.

Larson, D. P., Nonnenmacher, L., & Coimbra, C. F. M. (2016). Day-ahead forecasting of solar power output from photovoltaic plants in the American Southwest. Renewable Energy, 91, 11-20. doi:10.1016/j.renene.2016.01.039

Lee, C. K. M., Huo, Y. Z., Zhang, S. Z., & Ng, K. K. H. (2020). Design of a Smart Manufacturing System With the Application of Multi-Access Edge Computing and Blockchain Technology. IEEE Access, Access, IEEE, 8, 28659-28667. doi:10.1109/ACCESS.2020.2972284

Martinez-Anido, C. B., Botor, B., Florita, A. R., Draxl, C., Lu, S., Hamann, H. F., & Hodge, B.-M. (2016). The value of day-ahead solar power forecasting improvement. Solar Energy, 129, 192-203. doi:10.1016/j.solener.2016.01.049

Massidda, L., & Marrocu, M. (2017). Use of Multilinear Adaptive Regression Splines and numerical weather prediction to forecast the power output of a PV plant in Borkum, Germany. Solar Energy, 146, 141-149. doi:10.1016/j.solener.2017.02.007

Moghimi, F., Ghoddusi, H., Asiabanpour, B., & Behroozikhah, M. (2019). Atmospheric Water Generation (AWG): Performance Model and Economic Analysis. Paper presented at the APMS 2019: Advances in Production Management Systems. Towards Smart Production Management Systems, Austin, TX.

Morrow, D. (2019). Toward a 100% renewable future - Lessons from the city of Los Angeles. Paper presented at the Texas Power and Energy Conference (TPEC), College Station, TX.

Nassereddine, M., Rizk, J., Nagrial, M., & Hellany, A. (2018). Battery sustainable PV solar house: Storage consideration for off grid. 2018 Third International Conference on Electrical and Biomedical Engineering, Clean Energy and Green Computing (EBECEGC), 34. doi:10.1109/EBECEGC.2018.8357129

Ogbomo, O. O., Amalu, E. H., Ekere, N. N., & Olagbegi, P. O. (2017). A review of photovoltaic module technologies for increased performance in tropical climate. Renewable and Sustainable Energy Reviews, 75, 1225-1238. doi:10.1016/j.rser.2016.11.109

Olesen, A., Smith, G., & Korn, S. (2019). USA Patent No.16/100795

Pedro, H. T. C., & Coimbra, C. F. M. (2012). Assessment of forecasting techniques for solar power production with no exogenous inputs. Solar Energy, 86(7), 2017-2028. doi:10.1016/j.solener.2012.04.004

Pierro, M., De Felice, M., Maggioni, E., Moser, D., Perotto, A., Spada, F., & Cornaro, C. (2017). Data-driven upscaling methods for regional photovoltaic power estimation and forecast using satellite and numerical weather prediction data. Solar Energy, 158, 1026-1038. doi:10.1016/j.solener.2017.09.068

Salih, J. E. M., Adom, A. H., & Shaakaf, A. Y. M. (2012). Solar Powered Automated Fertigation Control System for Cucumis Melo L. Cultivation in Green House. APCBEE Procedia, 4, 79-87. doi:10.1016/j.apcbee.2012.11.014

Sarkar, A., & Majumder, M. (2019). Economic of a six-story stacked protected farm structure. Environment, Development and Sustainability: A Multidisciplinary Approach to the Theory and Practice of Sustainable Development, 21(3), 1075. doi:10.1007/s10668-018-0088-0

Shin, K.-Y., & Park, H.-C. (2019). Smart Manufacturing Systems Engineering for Designing Smart Product-Quality Monitoring System in the Industry 4.0. Paper presented at the 2019 19th International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea (South). Conference retrieved from

Tao, F., & Qi, Q. (2019). New IT Driven Service-Oriented Smart Manufacturing: Framework and Characteristics. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Systems, Man, and Cybernetics: Systems, IEEE Transactions on, IEEE Trans. Syst. Man Cybern, Syst.(1), 81. doi:10.1109/TSMC.2017.2723764

Thakare, A., Belhekar, P., Budhe, P., Shinde, U., & Waghmode, V. (2018). DECISION SUPPORT SYSTEM FOR SMART FARMING WITH HYDROPONIC STYLE. International Journal of Advanced Research in Computer Science, 9(1), 427-431. doi:10.26483/ijarcs.v9i1.5292

Valiente, F. L., Garcia, R. G., Domingo, E. J. A., Estante, S. M. T., Ochaves, E. J. L., Villanueva, J. C. C., & Balbin, J. R. (2018). Internet of Things (IOT)-Based Mobile Application for Monitoring of Automated Aquaponics System. Paper presented at the IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM), Baguio City, Philippines.

Valles, D. (2019). PCA, LDA, Kernel PCA. Paper presented at the EE 5331: Machine Learning, San Marcos, TX.

van der Meer, D. W., Widén, J., & Munkhammar, J. (2018). Review on probabilistic forecasting of photovoltaic power production and electricity consumption. Renewable and Sustainable Energy Reviews, 81(Part 1), 1484-1512. doi:10.1016/j.rser.2017.05.212

Villalonga, A., Beruvides, A., Castaño, F., & Haber, R. E. (2018). Industrial cyber-physical system for condition-based monitoring in manufacturing processes. 2018 IEEE Industrial Cyber-Physical Systems (ICPS). doi:10.1109/icphys.2018.8390780

Voyant, C., Notton, G., Kalogirou, S., Nivet, M.-L., Paoli, C., Motte, F., & Fouilloy, A. (2017). Machine learning methods for solar radiation forecasting: A review. Renewable Energy, 105, 569-582. doi:10.1016/j.renene.2016.12.095

Walrath, C. A. (2010). USA Patent No. USPTO: 7834585B2.

Wolff, B., Kühnert, J., Lorenz, E., Kramer, O., & Heinemann, D. (2016). Comparing support vector regression for PV power forecasting to a physical modeling approach using measurement, numerical weather prediction, and cloud motion data. Solar Energy, 135, 197-208. doi:10.1016/j.solener.2016.05.051

Zhang, L. (2018). Specification and Design of Cyber Physical Systems Based on System of Systems Engineering Approach. 2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES). doi:10.1109/dcabes.2018.00084

Zhang, Y., Xu, X., Liu, A., Lu, Q., Xu, L., & Tao, F. (2019). Blockchain-Based Trust Mechanism for IoT-Based Smart Manufacturing System. IEEE Transactions on Computational Social Systems, Computational Social Systems, IEEE Transactions on, IEEE Trans. Comput. Soc. Syst., 6(6), 1386-1394. doi:10.1109/TCSS.2019.2918467