Final Project Documentation -Team 8

Adrian Barrera, Brandon Buck, Kevin Crowe, Yutthachat Thao

Project Green Wall

April 27, 2020

Professor Tatro

College of Engineering and Computer Science, California State University, Sacramento
6000 J Street, Sacramento, CA 95822

AdrianValentino@csus.edu

BrandonBuck@csus.edu

KevinCrowe@csus.edu

YutthachatThao@csus.edu

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

We are making an automated indoor aeroponics system that will solve many of the issues with the current farming model.

Currently, farms for produce have gotten larger and larger, and farther away from where the produce is actually consumed. This is due to ever increasing populations. The problem with this is that a lot of money is spent on transporting and storing produce. We came up with an idea that would put food production right in the customers home. This would cut out all transportation costs and ensure that the customer gets the freshest produce possible. People who are trying to eat more healthy or cooks looking for fresh herbs would benefit greatly from such a system. We would have to address various issues when taking on this problem. We have to create a system that can fit in a room easily, be completely contained, quiet, help the user grow the plants, and create an overall system that is automated. We would create a contained system that has various sensors like temperature, humidity, and light. It would require minimal user interaction like being able to go at least one week between fill ups. Once we came up with our societal problem (the current farming model), and our design idea (an automated aeroponic system) we have a few more things to address.

We detail out the complete budget for the project. This includes all the parts we bought and what was donated which total up to about $1100. This would have been hard on us if we did not have donations from outside sources. As we did not predict accurately the cost of the whole system, we were left relying on those outside sources. In this report we detail our work breakdown. We had 5 major features that we wanted which are that the system can collect environmental data; it requires little user interaction; it can encourage community interaction; it is automated and has little impact on the room environment. We split this into small features and tasks that will allow us to divide up the work. This work breakdown structure is an analysis of the major things we had to accomplish and which group member worked on the task. Along with this we detail the total hours spent on each section by the group members. Comparing ourselves to other teams, we spent less hours in total. However, we were ahead in the first semester as we were able to easily assign everyone with the task that they were familiar with. In total, we spent 880 hours on the project which included our group meetings.

Next we analyze the risk associated with the project. Everything from minor things like component failure all the way to natural disasters shutting the school down. It is critical to plan for problems so that if they occur we will be ready and keep the project moving forward.Then we list out in detail every metric our design must satisfy. We were able to meet all the measurable requirements through careful selection of parts and the group's communication. All of our testing data is laid out to prove we accomplished everything we set out to do. We tested many of our components such as the sensors and apps to know that they worked as intended.

Finally we explain our entire design philosophy for every component and then analyze how our product fits in the market. Our system can fit pretty well into the market if we keep doing tests with our system. Although our system is viable, it still has a lot to go through to actually be a great product as they are still minor issues with bugs in the coding and factors that we did not account for such as the lighting issues and even leaks in our system. However, our system is able to meet all its requirements and with the small fixes, we can turn it into a viable product.

Abstract— **Team 8 has proposed a solution to the problem of access to fresh produce due to the ever growing urban population: a smartphone connected indoor aeroponics system to address the issue of accessibility to fresh produce in urban areas. This system will allow users access to their own personal garden. This will help reduce the need for groceries. In turn, it will reduce the amount of food miles that the produce would have to travel if it is mass grown in a rural area away from the urban population. This system is cost effective compared to similar systems on the market. As we spent about $1100 for the whole system, and it can grow up to 12 plants unlike the ones in the market right now. They cost in the upper thousands while only able to support a small plant. This is achieved through environmental sensors like humidity, temperature, and light are integrated with a raspberry pi to create an automated growing system. As we went through the process of selecting the sensors that are cost effective and materials that we knew would last, we are able to build a system that can compete in the market. The system requires little user interaction by having an app that facilitates the growing process. All in all, our system was able to keep plants alive for 2 weeks. With further tuning, we can fix our shortcomings and have our prototype compete in the global market.**

Keywords—aeroponics, urban agriculture, food desert, farm, food shortage

I. INTRODUCTION

As the urban population continues to increase rapidly, people are in need of a way to get fresh produce. Therefore, we decided that we will look to tackle this societal problem. The problem we decided on was the lack of availability of healthy food in urban areas. From there we thought of the necessary features our project would need to have to address that problem. We decided on an automated, indoor, aeroponics system that would be low-experience friendly. It would also have the capability, through an android application, to read relevant sensor information in close to real time and allow users to trade crops. With this system people will be able to grow their own fresh produce in their home. Although our design is small, we make up for it with our simple trading system. We plan for there to be many users of our system, and they can do some trading of surplus to help with their needs.

We described our funding for the project over the course of the two semesters as well as the simple total for parts on the final machine. We got funding of about $1100 in which $560 came from us. We were able to receive this funding from one of teammate's workplace. This was very helpful for us to complete the project because we did not anticipate that the cost was going to be near the thousands. However, we did plan on spending 150 each which with the outside fund allowed us to meet our budget.

With the design idea done we went on to plan the tasks that would need to get done over the two semesters to make sure our project was done in time. We first looked at the milestones our project would have over the time. These milestones included the important dates where the project would need to be done but also the assignments we would need to manage while we get the project work done. With this in mind we split the mechanical. hardware and software work between the four of us. Adrian designed the smartphone application and supporting database. Brandon designed the structure for and planned the physical layout of the device structure. Kevin was responsible for the lighting and automation programs. Yutthachat was responsible for the sensors and the sensor code. The planning for the first semester focused around getting our deployable prototype done for the senior showcase. We did our best to complete as much as we can because we want to prove that we can succeed in our project. The second semester was dedicated to fixing any

problems with, and upgrading, the design for the senior showcase. We redid our wiring and upscale our system. This allows the system to now support 12 plants. Furthermore, we got new lighting that can easily supply 12 plants with enough sunlight.

We next look at the risks of the assignment for the user and for ourselves and what we did to mitigate the harm if it exists. We planned for situations that are similar to the past such as the Paradise fire. This risk prevention strategy allowed us to migitage the Covid 19 pandemic as we had already moved all our material from the lab. From there we go into the reasons for designing our system the way it is and the reasons for using certain materials/components. This ranges from buying the correct sensor such as the DHT22 temperature and humidity sensor. Also, getting a submersible sensor of the Electrical Conductivity and pH sensor. We also picked building material such as corian as it is sturdy and will be perfect for our project. From there we look at the status of our project here at the end of the year and where a product like ours, given some changes in manufacturing, would sit in the market. We still need to fine tune the system, but it is still working as we have already grown a few plants.

II. SOCIETAL PROBLEM

Our design was created to address the issue of getting healthy food into urban areas and to join the trend toward urban farming as a means to fix other problems within the food industry. People in heavily populated urban areas suffer from being in what are called “food deserts”, places where healthy food is less accessible than cheaper and unhealthier options. Making healthy food more available in urban areas, places where people eat food that comes from far away from them, could help to tackle issues of obesity or lack of nutrition. There is also an upcoming trend toward the idea of urban farming, farming in areas inside a city to cut out the cost and pollution associated with transportation. Joining on this trend could help to make healthier food more available in urban areas with the added benefit of reducing pollution.

A. Urban Farming

The idea of urban farming has predated the now normal method of farming, industrial farming. People have been farming since the Neolithic Era, from seven to ten thousand years ago. The rise of farming allowed people to stay in one geographical area, as they could grow the food that they needed, leaving behind the hunter-gatherer lifestyle. At that time, humans were farming crops such as wheat, flax, and barley. For the following few millennia, agriculture saw only small improvements such as more efficient irrigation and crop cycling. However, things began to rapidly change in the 18th century, when the technology of agriculture grew to include tools such as the mechanical combine harvester. In the early 1900s, machinery completely replaced horse-drawn plows, with tractors becoming increasingly popular. Not only was this machinery making farmer’s lives easier, they were also allowing farms to be more productive, leading to the perfect conditions for population growth.

As time went on, and the population grew, so did our technology. We saw the rise in urbanization, and the development of vast cities. The rise of the automobile and other modes of transportation, along with refrigeration, changed agriculture forever. These technologies further allowed people who lived in cities to not need to rely on locally grown food as much, since transport was becoming a more viable option. With the growing efficiency of agriculture, more and more people were able to not have to learn any agricultural skills, and focus their attention elsewhere. Cities provide opportunities for people that had never been available to people before such as new professions, global trading, and new means of entertainment. With the creation of cities, though, came the reliance on others for food, and the idea of

making your own food, especially in an urban environment, resurfaced.

The usefulness of urban farming is well known, as seen during World War II, many resources and labor in the United states were put into the war, creating a shortage in food ration. To combat this, the government encouraged citizens to plant a "Victory Garden". These gardens used backyards, empty lots, and rooftops to grow their own food. At its height, the gardens were able to produce 40% of all vegetables in the United States [1]. As society continues to grow, many prominent people of our civilization are starting to embrace this idea of a personal garden that can provide food for the family. Former First Lady Michelle Obama started a garden in the White House. The United States Secretary of Agriculture encourages many citizens to join in what is called the "People's Garden" [1].

There's no such thing as a free lunch. Urban agriculture, while growing more popular, has many obstacles to overcome. The main issue that our team discovered is the barrier to entry when it comes to growing your own food. A person wanting to grow their own food must not only do extensive research on the plants that they want to grow, but also what they need and build their own system or to buy a complete system. Even this choice, though, is made uneasy because there are different growing methods: soil, aeroponics, and hydroponics. The second issue was the cost of many of the "all-in-one" systems available on the market, a problem we go more in depth on in the market review section of this paper. The last major issue is the one of physical space; urban farmers are in urban areas. In urban areas, especially dense city centers, space is expensive. Look no further than San Francisco, a Californian city where space is a luxury not easily afforded: the median price per square foot of space is $1,108 [2].

B. Food Deserts

One of the main problems that our team set out to tackle is the problem of food deserts; we saw the distance between consumers and producers of food to be a problem. Typically, as shown in Figure 1, a person who wants to buy produce has a few choices when it comes to how: farmer's markets, supermarkets, or from a shipping company directly. No matter the avenue, the process is cumbersome at the least. A customer can sacrifice convenience by going to a farmers market, or they can go to a supermarket where the produce may not be fresh or be overpriced. The problem is multiplied the larger the city in question gets, as the agricultural lands that support the city are even farther away. For example, New York City and its surrounding areas are highly urbanized, and have huge populations. To support these populations, consumers must rely on food being transported in from distant farms.

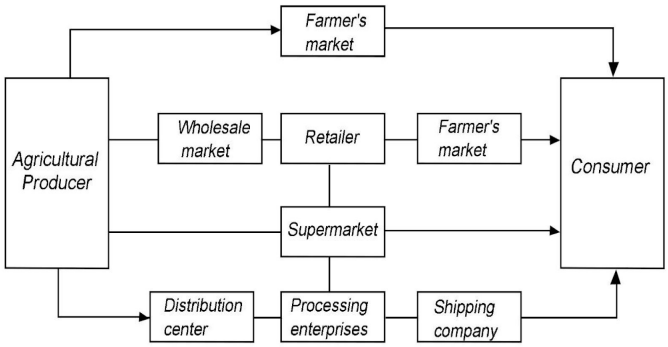


Figure 1: Agricultural Producer to consumer logistic cycle adapted from [3]

Regardless of the source, the traditional model of food production leaves at the mercy of the agricultural producer, transporter, supermarket, and any other intermediate entity. If there is a problem with any number of these intermediate entities, the consumer suffers. For instance, if the price of fuel skyrockets, it costs more to transport any food from the producer to the consumer's table. This can lead

the consumer to choose cheaper, more processed food in response to the increasing prices for fresh produce.

C. Environmental Benefits

Key benefits of moving food production closer to home include the reduction of food waste, the reduction of plastic waste, and reducing carbon dioxide emissions in already congested areas. Cutting out the transportation aspect of the food logistic cycle directly leads to less carbon dioxide emissions by transportation trucks, and industrial machinery used to process and package the food.

The reduction of carbon dioxide output of the food production industry and increase in plant life around a city has the added benefit of reducing temperatures in the city. Meharg [2] and Ackerman et. al. [3] refer to the "urban heat island" effect of big cities. Waste heat coming as a result of having such a dense population in a landscape of concrete and glass. This is a problem of cities that is a benefit to urban farming, "waste heat that all cities generate can be harnessed"[4]. All the activities in a city such as driving, production of goods, and usage of energy adds energy to the environment. These energies, instead of being released into the surrounding, are circulated in the city's microsystem. The microsystem consists of the greenhouse gases emitted in the city. In addition, all the buildings and materials in a city are absorbing the heat of the sun. Without a way to use up the 2 energy it is then trapped in the city for the day, "cities are typically found to be warmer than other areas, anywhere from 0.6°C to 12°C warmer"[4]. However, with gardens throughout a city, plants can absorb that sunlight with a way to convert that energy into food.

When there is an abundance of aged food at any point in the traditional logistic cycle, the entity tends to throw out the food because it either can't sell, won't sell, or to make room for another shipment. This incredible amount of food waste is not only bad for the environment through wasted water and pollution growing food that will be discarded, but also creates a huge amount of packaging waste. Items such as strawberries are kept in a plastic enclosure, where other items are wrapped, such as heads of lettuce. According to the United States Food and Drug Administration, food waste is estimated at between thirty to forty percent of the food supply, corresponding to approximately 133 billion pounds and $161 billion worth of food in 2010 [5].

D. Revision

With the start of the second semester of the Senior Design course, our team took the broad societal problem that we began with, and refined it. We took a step back and reevaluated the societal problem, and how our design best fits in the overall solution. We realized that there is no silver bullet to the many problems surrounding the agriculture industry, Our design, as mentioned in the next section, addresses the problem of accessibility to fresh produce to those in urban areas. Where we had to scale back our expectations, though, was the design's ability to supply enough food for a community to thrive; our design focus is on making the process of growing more accessible, and helping ease the reliance on supermarkets. Overall, the core of our goals have not changed, but the extent at which we expect the design to have an impact went through revision.

III. DESIGN IDEA

In order to address our societal problem our team created a set of features our design would need to be a viable solution. We started with the idea that the growing device would need to have sensors to gather all the relevant data required to grow food. We also decided that the process of growing the food should be primarily automated so people who aren't seasoned food growers can still work the device. The device would also have to require little interaction from the user so the device can run for long periods of time. Further we decided that the device should connect to other users through a smartphone

application. This could allow people to trade food, fostering community and adding the option for variety in the person's diet. Lastly we wanted the device to have little impact on the environment it would be put in, not causing noise or being confined to only certain allowable areas.

A. System Collects Environmental Data

After discussing what our system should be able to do, we decided that our design will have a way for the user to see the data to help grow the plants. We initially thought of 5 environmental data that we need to collect in order to help the user grow their plants. Also, the data can also be used to control the automation of the plant. These five environmental data include: the lights, humidity, temperature, pH, and electrical conductivity. After the first semester, we looked backed at our data collection and tried to see if we needed to change anything for the data collection. However, we believed that the data that we are collecting are sufficient for what we aimed to do with the design.

1) Light Sensor: For the first semester we just planned to use a general light sensor, something able to measure light intensity so that there could be a range of known light levels for the user to utilize. We also found it important that the sensor could measure UV light as the sun and many grow lights put off a good deal of it. To understand the amount of light our device would need to give off we looked to online sources to get an idea of the amount of light required for different types of foods. The most used metric used seemed to be Watts per square meter (W/m^2), Online sources give a recommendation of 32 (W/m^2) [6]. We decided to use the range of 25-40 (W/m^2), making our device suitable for growing lower light level food. This value would help us choose the correct power of light since it should just require knowledge of how big the grow space is.

There were several problems we encountered when trying to work this value into the design. Firstly most sensors output values in lux, the conversion from (W/m^2) to lux seemed to vary by the spectrum of the grow light used. Secondly, LED grow lights do not advertise the power required to operate them, they advertise their equivalent output to a High Pressure Sodium (HPS) grow light. The LED grow lights often operate at much lower power than is advertised. A third related problem was that distance from the grow space was not taken into account with the (W/m^2) metric. All of these facts made it hard to confidently design a lighting system that wouldn't burn the plants or starve them of light. Due to this, for the second semester the metric for the sensor was changed to a range of Photo-Synthetic Photon Flux (PPFD).

While this metric had the same problem as (W/m^2) in that it still had to be calculated from the lux value sensors, it was far easier to compare grow lights and to be sure plants were getting the right amount of life. PPFD measures the amount of light photons that hit a square meter of area [7]. It is a very direct way of measuring light that allows for considerations like distance and different lights. Most LED grow lights even give their tested PPFD values. We found after research that a good range would be 200-600 PPFD, again covering low light plants like lettuce and herbs.

2) Humidity Sensor: The second data that we plan to use and collect is the humidity. At the beginning of the semester, we believed that collecting the humidity of the ambient environment and the humidity of the inside of our roots will help us control the system. The humidity within an aeroponic on average should be near the 100% relative humidity. With this information, we can use the humidity data gathered to check if the plants are still getting their water and nutrients.

Therefore, when deciding what would be the appropriate humidity that our sensor should be able to measure, we can use what we know initially. Since we knew the humidity that we were trying to

measure, we can search for a sensor that can measure a relative humidity of 40%-100%. As stated in our measurable requirements, the humidity sensors should be able to measure a relative humidity of 40%-100% with the temperature between 60F - 90F. We put the temperature requirement with the humidity section because the humidity sensor is only able to measure the relative humidity of a surrounding and not the absolute humidity. Therefore, it needs a temperature requirement to accurately measure the relative humidity.

We decided that since our design is mostly for internal use within a household, the temperature of the environment will not change drastically. Therefore, the temperature will always be around 60F-90F. Furthermore, the design needed to measure two different humidity: one for the ambient environment and the one that is measuring the environment within the design box. After the first semester, we learned that our design contract for the humidity sensor is still valid as we did not change anything for the second semester. The requirements are still the same as the first semester meaning that we probably knew a bit about what we were doing for the environmental data collection.

3)Temperature Sensor: Another data that is closely aligned with the humidity data is the temperature data. At the start of the semester we knew that we needed to measure the temperature for our system because it is an easy data to collect and may be useful for the user if they plan to use this design outside. We decided that the design will be able to measure the temperature within the box and the ambient temperature. Since we agreed that our design is mostly for indoor use, we plan to fully use the fact that the temperature within a household is near room temperature which is around 70F. Therefore, we knew that the temperature will only change a little and created the measurable requirements that is from 40F-110 with a precision of +- 2F. This will give us a lot of headroom for the temperature within the box.

Within the box, we plan on running the pump constantly or at a quick interval to maximize the amount of water the plants will receive. Thus, the temperature within the box may drop to near 65 or 60F. Then, according to our design requirements, our temperature sensor should be able to measure that temperature accurately.

After the first semester, we reviewed our measurable requirements to see if we needed to change anything. However, looking at how our temperature sensor and the data it collected are, we believed that there was no need for a change order. The temperature that we collected was well within what we had had hope, so we didn't need to revise any requirements.

4) *pH Sensor:* For the pH sensor, we knew that we needed to measure the pH of the nutrient solution to know the effect it would have on the plants. At the beginning of the semester, we noted that our group does not have any one who is really familiar with farming. The average pH need is within 5.5-6.5. Plants like more acidic soil and solutions. With this information, we set out to find a sensor that will meet this requirement and is well within our budget.

Our requirement is that the pH sensor should be able to measure the pH of the nutrient solution from pH of 5 - 7.5 with a precision of .1. We decided to overshoot the average pH that plants needed because we wanted the sensor to help with the control of the pH of the nutrient solution. The pH sensor will have to be able to measure the pH accurately, and then, we can use that pH value to monitor and control the pH via a peristaltic pump.

After the first semester, we learned that our pH sensor and its data were within our requirements.

We did not need to implement any pH controls for the first semester, so we cannot determine if the pH sensor will be able to regulate the pH of the nutrient solution. However, we learned that the pH sensor was able to measure the pH accurately. This was all we needed to know to precede with the second semester. Therefore, looking at our requirements from the first semester, we agreed that the pH requirement is good for the plants and our design contract.

5) *Electrical Conductivity Sensor:* The electrical conductivity sensor or the parts per million sensor was used to see the amounts of nutrients left within the nutrient solution. In the first semester, we wanted our design to be able to regulate its own nutrient solution. This would be done if we could measure the amount of nutrients within the solution. We researched and found that the electrical conductivity of a solution is able to be converted to parts per million or we can just use the EC as a measurement. The EC of a solution was a standard unit of measurement for hydroponic solutions. Its units are millisiemens/cm. We also looked at the typical electrical conductivity of a solution and created our design requirement based on the typical value of EC. The measurable requirements for the EC is to be able to measure accurately with the range of .1 EC to 10 EC. Since the typical EC values are between 2 to 4 EC. This requirement will allow our sensor to measure outside the range of the typical EC and using that information, regulate the EC of our solution. Therefore, we decided to find a sensor that can measure the EC of a solution.

In the first semester, we knew that we wanted the project to be low-budget and the EC sensors that we found online were expensive nearing the 100s. Therefore, we decided to build our own using a voltage divider and two probes within the solution. This design was able to measure what we wanted but the amount of time spent calibration and setting up the sensor was far from what we wanted. After the first semester, the EC sensor was still working fine and it met all our requirements. However, it still needs more work to calibrate it everytime we change the structure of our system. Looking back, buying an off-the-shelf EC sensor would have been much easier and would have definitely taken less out of our group in terms of time spent.

B. Automated Growing Process

In our design, we wanted the system to automate the growing process of the plants. We would use the data collected from the sensors and control various aspects of the system to help with the plant's growth. We concluded on three primary regulations: the system should be able to regulate its nutrient solution; the plants are to be misted with nutrient enriched water; the system should be able to control the lights.

1) *Nutrient Solution Regulation:* In the first semester, we decided that our design should be able to regulate its own nutrient solution, so the user would not have to worry about the plants running out of food. The first thing we recognized was that aeroponics will have to use hydroponic solutions as they are basically the same thing. Since aeroponic is a subsection of hydroponic, we needed hydroponic solutions and a way to regulate it. We knew that we needed to regulate the amount of nutrients inside the solution to help the most with the plant growth.

This system control of the nutrient within the solution would be done via a pump using the data gathered from the EC Sensor. The user will be able to set a threshold for when the pump should start pumping in more nutrients. Since we are not expert in plants or farming, we cannot give precise detail to which EC the user should. However, the system will have a default value for the EC threshold. This value would be the average typical value we researched which is at 2 EC. The system then, will help water the plants with this solution and while the nutrient within the solution depletes. The system will be able to add more solutions. At first we planned on also

introducing a system that will control the pH of the solution. However, looking at our design requirements, we believed the pH of the system would not change drastically from what it is supposed to be. Also, the pH of the hydroponic solution is more acidic; therefore adding more to the nutrient solution would decrease the pH of the solution. This is okay as plants like more acidic solutions nearing 5.5 to 6.5. Furthermore, when the water is changed the pH will be up toward the pH 6.5-7.5. Adding the solution at the time will only help with the pH of solution as it will reduce the pH of the solution toward the 5.5-6. 5 mark.

After the first semester and into the second semester, we knew that our EC sensor was working, and now we can work on the nutrient regulation. We also knew that we can meet our design requirement as we can now read the EC and make a closed loop control to regulate the nutrition.

2) *Water Pump For Plants:* Another automation that we want is nutrient and water delivery systems. We planned to use a few pumps to spray the roots of the system with nutrient enriched water. This fulfills the plants' nutrition and water requirements which should help them grow. In our design contract, we stated that the user can control the interval for the water pump. This is exactly what we planned because we do want the user to have some knowledge about plants to actually grow them. The user would use an app to control the water intervals, and the system would a default value for the water interval. This default value is just an arbitrary value, however it should be able to keep the plant moist and the inside of the box at high humidity.

After the first semester prototype, we knew that our water and nutrient delivery systems work and we would just have to upscale. Furthermore, the system was adjustable, so it can easily be configured to work on a larger scale. We didn't need to change this requirement as it does not have an exact defined measurable metrics, but if the design is able to output water onto the plant, then the feature is met.

3) *Light interval Control:* The last big feature that we want our design to have is the ability to control its own grow lights. This would be an amazing feature to help the automation of the system as the lights would turn on and off in accordance with what the user wants. Again we let the user decide how much light the plants should get because every plant is different, and the user would know more than us.

Similar to the water and nutrient delivery system, we do not have an exact measurable metric for light control. However, as long as the system can control the light and the user can configure it, the feature should be met. In the first semester prototype, we were able to control the lights using the raspberry pi and relays. This demonstrated that our feature can be met, and in the second semester, we would be working on making this work with the user app.

C. Requires Little User Interaction

1) Water Tank Last for a Week: If the system is supposed to be in the home and user friendly, then it must be contained and not require the user to service it constantly. We decided that one week was a fair time span to go in between water and nutrient fill ups. Along with this requirement we decided that the system can't be hooked up to an external source. It had to be a reservoir inside the system. This way the user would have much more flexibility and be less of an intrusion in the home. We had to make sure the system was well sealed as well so we didn't lose and water and create a mess.

2) *User Access Status information in Under 60 Seconds:* A user will not be inclined to use the aeroponics system in our design if it is not easy to use. A large part of the user experience when it comes to software is how long it takes the user to access the information that they want to see. Our design calls for

a solution that allows the user to access the information on the aeroponics system in less than sixty seconds. This time period was measured from the time that the smartphone application starts, to the time that the user sees updated sensor information on the main landing page. Sixty seconds was chosen as it takes into account the time it takes for the user to be able to login to the smartphone application, and any network delays.

The user being able to access their system's data quickly is part of the larger user experience design for the smartphone application. Our team set out to make the smartphone application as simple and intuitive as possible, to avoid user frustration. While we spent time ensuring the ease of use, proper user experience and design research was deemed to be outside both the scope of our team's expertise, but also outside the scope of the project. Sixty seconds is an adequate qualitative measurement of how quickly the user can understand and navigate the smartphone application.

3) *Notification Of System:* If not cared for correctly, many plants will die very quickly. Because of this the system must be reliable and alert the user if anything goes wrong. Since the system can go over a week between fill ups there is the possibility that the user will not actively monitor it. To help with this the system will alert the user if it detects problems. If the sensors do not detect light when the light should be on it will display an error. This error will indicate that the light has burned out. Another error will be sent if the inside humidity is the same as the outside. This will indicate that the sprayers are not working. Both the light and sprayers are very critical so that's why we implemented these error notices.

D. Encourages Community Interaction

1) *Trading Contracts:* The idea behind community interaction for the purposes of addressing our societal problem is to help foster the popularity of urban agriculture and make it easier for users to come together to fill the nutritional gaps in their system. The aeroponics system, while able to be a standalone unit, also includes the ability for users of the system to contact other users of the aeroponics system in their respective area. With a way of interacting with other users, an individual can agree to cultivate a certain type of food, with other users doing the same. The users would then be able to trade their yield to get a more balanced food supply.

While this initial design was to create a user profile system, and a chat feature, time constraints led to the simplification of the feature. The final prototype has the ability to show which users are in a given zip code, and contact information. While not ideal, this solution still allows users to be able to connect with other users. This feature is also optional; the user of the aeroponics system can opt out of having their information available.

1) 10 Simultaneous Users: Since multiple users are able to have their own aeroponics system, our team decided that it is important to ensure that the database can support at least ten users simultaneously; the user experience would suffer if only one user or a handful of users can access their data simultaneously. The ability for at least ten simultaneous users to be supported was a driving motive behind the decisions regarding the database. We chose to go with the Firebase RealTime database. Firebase allows up to 100 simultaneous users on its free tier of service, which is perfect for the purposes of this design.

E. Minimal Environmental Impact

The system must not impact the area it is placed in. It must integrate well within a home setting and not disturb the user.

1) Maximum Noise Level: The system must produce no more than 70 decibels. The sprayers and

pumps must be set up in a way that quiets them. If the system is loud no one will want it in their home.

2) Floor Space Saving: To make the system fit in homes easier we decided it had to have a 2:1 wall to floor space ratio. Generally people have much more open vertical space than floor space. A 2:1 design is more efficient in our target setting.

3) Wireless Connection: Another feature that makes the system less intrusive is a wireless internet connection. It is a cleaner design with no ethernet cables required so the system can be placed anywhere it is in range of wifi.

IV. FUNDING

The funding from this project came from the group members. Each of us bought the necessary equipment for our separate aspects of the project, primarily in the first semester, and planned to split the cost evenly at the end of the semester. We kept track of the equipment we bought and prepared a seperate list of the equipment used in the final product. We further examined the funding to see the total cost of our project and several other interesting factors.

A. Funding Plan

The funding plan for the project in both semesters was whoever needed the piece of hardware had to get it, apart from some purchases just before the two showcases. This was not a big problem because most of the purchases were inexpensive but several essential components were expensive and had to be bought individually. Significant help in terms of resources came from the employer of one of our members, providing materials and tools that helped us with what would have been the most expensive part of the project.

In the first semester the primary purchases were the sensors and the actuators. The more expensive pieces of the project were not yet purchased due to fear of damage or realizing they're not what's needed. One of each of the sensors were purchased with the most expensive pieces purchased being the structure material and the analog pH sensor.

Several other important items were bought in the second semester but most hardware in the final project had already been purchased at this point. At the start of the semester the grow light was bought along with several more sensors but from there all of the pieces of the project had been assembled

B. Data

The total team cost is 558.42 which is the total personal cost to the team removing any materials that were donated or owned. We believe this cost is very acceptable given the final product we were able to get. Furthermore, we did expect to spend as least 150 per person for the project. Therefore, the team cost at 558 is perfect for us. For the funds, we allowed anyone to spend on anything they needed for the project. The only thing they need to do is to keep track of the purchase. This is done through online purchasing and receipt for brick and mortar stores. Since we did not want any one person to pay for more of the components, we added the total and split up the differences. Luckily for us, we also had outside sources that were able to donate the material and components that we needed for our project. Without these sources, we may have to pay some much for the project. This just shows that when planning a budget, we needed to think and account for much more than we had needed.

Table I. Total Cost of the Project adapted from [8]

Part Function	Cost/Unit	Quantity	Cost	Source
Structure Material (Corian)	500	1	500	Donated (Ansync)
Grow Light (KingLED)	100	1	100	Amazon
Microcontroller (RaspberryPi 3B)	39.5	1	39.5	Owned
Water Pump (Aubig)	22	1	22	Amazon
Chemical Pump	9.8	1	9.8	Amazon
Power Supply	32	1	32	Amazon
Relays	9.99	1	9.99	Amazon
2 - DC-DC Converters	10.98	1	10.98	Amazon
Mounting brackets	100	1	100	Donated (Ansync)
Plastic growing pots & plants	8	12	96	Amazon
Wiring hub	10	1	10	Donated (Ansync)
pH sensor (Gravity)	70	1	70	Amazon
Electrical conductivity sensor	15	1	15	Home-Built
Visible/IR lux sensor (TSL2591)	8.65	3	25.95	Amazon
UV index sensor (GUVA-S12SD)	9.6	1	9.6	Amazon
Temperature/Humidity Sensor (DHT 22)	11	2	22	Amazon
EC Callibration Solution	19.58	1	19.58	Amazon
			Total Cost:	Donated:
			1092.4	610

The table above shows a list of all parts and materials used in the deployable prototype. There are several things to note. First is that an equal amount of value in our project came from donations, we were able to create a very professional design because the resources we were afforded. Another point is that the personal cost that went into this final prototype was $482.40. Given that the numbers the team provided are close to accurate, there was very little spending, primarily hardware, that didn't make it into the final product. Another fact is that Amazon was the primary supplier of the parts used in the prototype even for the many Adafruit products. We found that most products available on company websites like Adafruit were also available on amazon with the benefits that come with being on the website.

V. PROJECT MILESTONES

Our project milestones are events that we personally feel that we have accomplished what we set out to do. This includes class assignments, work breakdown tasks, and building the actual structure. These milestones are encouraging to us because it helps prove that we are able to build the design and finish senior design.

A. Fall

In the first semester, the earliest milestone is when we knew what we wanted to build to help with our societal problems. At the time, we all have different ideas on how we would tackle the lack of fresh food in urban environments. However, we all decided that it was best if we would have an automatic aeroponic system. The next milestone was when our design requirements were accepted. We

spent a lot of time designing our requirements as we knew that it would be what we would be working for the next 2 semesters. Even though we were very excited that our design requirements were accepted, we were still cautious about how we would actually fill those requirements. Our whole group does not have a single person that is very familiar with gardening less aeroponic. This means that from the start after the design contract, we were still unsure if our design can actually work.

Jumping to the middle of the semester when the technical review was coming up real quick. We each presented our works and tasks to our whole group, and everyone was on track. The whole group was able to demonstrate what they did and how it will help with the final product. At this point, we knew that we could achieve the tasks that we set out for ourselves. This feeling carried on toward the December prototype where we met up at Kevin's house and finalized the setup for display. We worked on the setup for the whole day, and at the end, our design was pumping water, reading sensors' data, and communicating with the phone app. At this point, we all knew that if we continue this way, we will be able to meet all our measurable requirements. This finalized prototype brought out our confidence in our design.

B. Spring

This confidence from the first semester carries to the second. We met all our previous goals for the december prototype, and now all we need to do is to upscale the system and test its components. The first major milestone in the second semester was concurrent with the device test plan. We knew that we would have to test our components to see if they can meet our measurable metrics. We test each of our sensors, our database and phone, and even our previous structure. At the submission of this assignment, we were able to meet all the components testing requirements. We were really proud of ourselves as we knew that our first semester was a great jumpstart to finishing our final deployable prototype.

The next major milestone was when we got the structure built and we needed to integrate all the components. With the structure in front of our eyes, we knew that our design was coming to life. We continue during that day finishing and test the integration to where we had it running for 1 hour. After a week of the initial integration, we needed to record the feature breakdown for the review. This prototype helped us understand that our group was still on the track to finishing strong. On that day, we transfer a few plants and decide to start using our system to see if the plants are going to grow. This was an important milestone because of all the previous tasks, we have not tried to grow anything yet. The next important milestone was two week after the initial transplant. We documented the growing process of our system and acted how a user would to see if our system can help plants grow. In the first two weeks, we noticed a good progress from the plants. However, after the 2nd week, our system had some leaking issues combined with wifi connection issues. This occurred and like how the user would act, the person taking care of the plants did not notice it until 3-4 days had gone by with the system broken. This event demonstrated that our system still needed further testing to actually keep plants alive with minimal user interaction. We learned even though we met our measurable metrics, they are still a lot of our variables that we needed to be well aware of if we want to actually grow the plants.

VI. WORK BREAKDOWN STRUCTURE

After deciding on our design we planned the work that would need to be done to complete the project in time. The five overall features were split among the four of us and we came up with a plan for the work for the Fall and Spring semesters. Each of the five design features split into many sub-features that set all of the measurable metrics for our project. After completely breaking down the tasks that would

need to be accomplished to both complete our prototype and further to complete the final project we set out dates for when these would need to be completed. We each set the tasks that we would need to complete with several milestones to mark when certain parts of the project would need to be completed.

Below are the charts of how we spent our time for the project. Since we knew what we wanted to do early on, we did not spend a lot of time researching. Therefore, we were ahead of many groups when it comes to getting our project done. Furthermore, we splitted the tasks to each member and were well versed in those tasks. Brandon has connections to engineers, and he was familiar with construction and building materials. Therefore, most of the hours that he spent was on the structure of the system. Likewise, Adrian is a computer engineer. He understands how to program and how to get started on the app that we need. He got assigned to the software side of the project mainly the app and the database. For Kevin and Yutthachat, they were both used to microcontrollers and how to interact with them. Therefore, they were assigned various tasks that dealt with the hardware and components section of the project. This was the reason why our team spent less than average time, and this was how we stayed on top of our project.

Table II. Fall Hours Breakdown Adapted from [9]

Features (Fall)	Adrian	Brandon	Kevin	Yutthachat	
Collecting Environmental Data	0	0	8	39	
Automated and User Control Growing Processes	0	2	27	9	
System Requires Little user interaction	49	4	5	0	
System Encourages Community interaction	40.5	0	0	0	
System is not intrusive	0	37	0	0	
Class Assignments	31	28	40	40	Total
	120.5	71	80	88	359.5 HRS

Table III. Spring Hours Breakdown Adapted from [9]

Features (Spring)	Adrian	Brandon	Kevin	Yutthachat	
Collecting Environmental Data	0	0	15	88	
Automated and User Control Growing Processes	0	0	53	0	
System Requires Little user interaction	50	0	0	0	
System Encourages Community Interaction	30	0	0	0	
System is not intrusive	0	66	2	5	
ClassAssignments	40	35	40	36	Total
	120	101	110	129	460 HRS

Table IV. Group Meetings hours Breakdown Adapted from [9]

Group Meetings		Hours
	Fall	23.5
	Spring	31

A. System Collects Environmental Data

System collects environmental data including temperature, humidity, electrical conductivity, pH, and the amount of light that the plants are getting. This feature was broken down into 5 small features which included: temperature and humidity sensors, pH sensor, EC sensor, and the lights sensors. These tasks were further broken down to smaller tasks which included gathering the right sensors, adding the code to the pi, and testing the sensors.

These tasks were split between Yutthachat and Kevin. Yutthachat worked with the pH, EC, temperature, and humidity sensors from ordering parts, getting the parts to work, and testing the parts. For Kevin, he worked on the light sensors which include 1 UV sensor and 3 flux sensors. He created code for them and tested them to see if they met our measurable requirement.

These two people were perfect for this section of the project as the hardware requirements means that only these two people will only need to talk to one another to finish their task. This is also the reason why most of their time alloted was with this feature set. Next, we will look at the automation of our system.

B. Automated Growing Process

The automated growing process included the watering of the plants, the regulation of the EC in the nutrient solution, and the controls of the lights. Again, with how we splitted our task, Kevin was mostly working on this part. Since these tasks included using the microcontrollers and other hardware components, Kevin was the perfect groupmate for the job. These subfeatures were broken down into small tasks including gathering the components needed such as the lights, pumps, and relays.

Kevin coded and built the system to control the pumps and to control the lights. These were to be integrated with Yutthachat's sensors such as the EC and pH. Both Yutthachat and Kevin's tasks were closely aligned, and they worked closely with one another to finish the small components and the sensors. Furthermore, in this section, we have a few tasks that involved communicating with the database. This database communication is used to get data from the user to control the pi and its automation. Kevin also worked on these tasks. Overall, these first two features have a lot of dependence, so having two people only involved in these tasks allow for a smooth sailing toward the laboratory prototype and also deployable prototype. Something that does not have to do with these tasks were the app and its database which was the main focus of our computer engineer, Adrian.

C. Requires Little User Interaction

The System requirement little user interaction means that the user should not have to worry a lot about the system. This feature was broken down into sub features including: the ability to go one week without water fillup; users being able to access the environmental data in under 60 seconds; users being notified of system errors. The ability to go one week is easy to meet because our preferred plants are small and the final prototype was going to be pretty big. Therefore, this task was assigned to Brandon, and he knew that it can be fulfilled easily. For the other two, they were broken down into the smallest task that ranges from creating an android app with user authentications. These all included getting the app to work with our firebase database.

Like stated before, Adrian was a good fit for these tasks as he is the most knowledgeable with the

software side of our project. He learned to program an android app with Android Studio, and he also learned to work with the Firebase. He accomplished these tasks well and was independent from the rest of the group because he can test his own code with random data. Since our database acts as a buffer between the app and the Pi, he did not need to work with any other person to create the app. However, he also helps Kevin and Yutthachat with the communication between the Pi and the database. Overall, Adrian spends the most time learning about the software side of our project and that is why he has fewer hours on other parts of the project.

D. Encourages Community Interaction

Another feature that we wanted to implement was the ability to communicate with other users of the system. This was split into two smaller subfeatures which were that the system would be able to connect 10 simultaneous users and that users can make contracts with other users. These tasks were assigned to Adrian as he was already working with the phone app, and these tasks were to be implemented into the app. Therefore, Adrian was the only person working on the app. He spent the most time on the software.

E. Minimal Environmental Impact

Minimal environmental impact that we wanted the system to have was to save space for the user and allow accessibility for quick maneuver of the system. This feature was broken down into subfeatures which were that the system would not make noise above 70dB; it will be an indoor vertical design that will use more wall space than floor space, and that the system will be able to connect to wifi from a range of 100ft. This feature is meant to address the structure that we would be building. Since Brandon was familiar with materials and had working engineers as consulting, this feature was to be completed by Brandon. These features were split into smaller tasks which required Brandon to research a good material and designed our prototype. Brandon was able to achieve these tasks quickly and effectively. This is why he had the most hours on this feature as he was the one with the most knowledge on construction.

Another subfeatures within this feature was the ability of the system to connect up to a range of 100ft for the wifi. This was not a hard task, and we were able to purchase an wifi adapter and test it to meet this feature.

F. Group Meetings

Finally is the group meeting that we have every week. At the start of the first semester, we designate a time and place to meet every week. This was on Thursday around 3 pm. This was perfect for everyone because we are able to talk and complete many assignments and small tasks. This meeting was the most helpful in helping our team with keeping track of our progress. Furthermore, these meetings allow many of us to help each other with smaller tasks in other features that they were not assigned. In conclusion, meeting every week furthered our progress and reduced the need to spend more independent time working on our project and assignments.

VII. RISK ASSESSMENT

Our project involves wires being around water we had to take account for those and other important risks to make sure our device is safe to use. Electrical problems could theoretically come from being in the vicinity of water and water vapor but could happen just as easily from being dropped or other improper care. We took several factors into account to ensure that these risks are minimised.

Table V. Risk Matrix Adapted from [10]

		Impact				
		Minimum Impact	Tolerable Impact	Limited Impact	Serious Impact	Catastrophic Impact
	LV	1	2	3	4	5
Unlikely(> 0.01 - <= 0.10)	1			-Failure of Firebase Server		-Pandemic (Covid19)
Low Likelihood (> 0.10 - <= 0.30)	2	-User Data Breach	-Broken Temperature and Humidity sensor -Failure of Server Data Transmission -Chemical pump failure	-Broken pH sensor -Microcontroller failure	-Loss of Smartphone Application Code - Inability to Send Information From Smartphone to Server	
Likely (> 0.30 - <= 0.50)	3		-Light sensor failure -Grow light failure	-EC sensor affecting pH sensor	-Water affecting the electronics	-Power shutting down due to wildfire
Highly Likely (0.50 - <= 0.70)	4				-Water pump failure	
Nearly Certain (> 0.70 - <= 1.00)	5					

A. Physical Device

The main risks to the physical system were damage from being dropped, corrosion, and just mistakes during assembly. The system is made from corian and glue mostly so it is in danger of being broken if the system was ever dropped or hit. The main mitigation of this was to just be careful and to reinforce the system with aluminum extrusions. This made it incredibly robust and much less prone to breaking. Since we must use a nutrient solution which is quite corrosive we had to mitigate the risks associated with that. To mitigate the risk we only used real nutrient solutions when we had to, and no steel parts were used. Mostly corrosion resistant plastics and powder coated aluminum. Lastly any mistakes in assembly could lead to a weak design and could cause leaks. To mitigate this the system was observed over a period of a few weeks to make sure everything was well sealed.

B. Sensors

For our sensors, we took precaution when dealing with expensive sensors such as the pH sensor. However for the other sensors, temperature/humidity sensor and the lights sensors, we made sure to find sensors that can be delivered in under three days. This is where we look up on Amazon to see if the product quantities are high and if the item is on the Amazon Prime delivery system.

Although we did not experience any failures in our sensors, we did have a few moments where we were rushing to order new components. As stated before, we make sure to purchase the components that are high in quantity and can be delivered in under three days. The first scare was the DHT22 Sensor which is used to measure temperature and humidity. We have two of these sensors, one for the inside and the other is for the ambient environment. The one for the ambient environment was giving us an offset of 5% relative humidity. Also, the sensor tends to stop working in the middle of operations. Luckily as stated before, we purchased sensors that are able to be purchased and delivered fast. This was the case for the DHT22 sensor. We bought two extra; however, we really did not need to purchase the problem that the sensor was not the sensor but the wire connected to it. This way we avoid a potential disaster.

However, this way of mitigating the potential risk of the sensors would still be good if there was no pandemic to mess with the delivery. However luckily for us, we have about two of each sensor beside the pH sensor. This gave us peace of mind from worrying about the sensor failures.

C. Light/Pump Failure

Since we are growing plants, lighting and water are two of the most important things. Although we are using LEDs (which are quite reliable), it is not impossible that they burn out or some other type of problem that kills the light. If the light goes out and we are not aware of the problem the plants could die fairly quickly. To mitigate this we implemented code that would send the user an error if the light sensor does not detect anything when the light is supposed to be on. If the light did break we would be able to source one within three days.

If the pump goes out the plants would die even faster. Like the light, an error will be sent if the inside humidity is the same as the outside humidity. We chose an aquarium pump that is made to run continuously for years. In our usage case the pump should not have any problems.

D. Smartphone Application Failure

The smartphone application is the main way that the end user will interact with the system; the user will use the application to adjust parameters in the system such as light timing and watering intervals. The application is the window into the happenings of the device; the user will use the application to see the status of various sensors such as humidity within the structure. Additionally, one of the main components to the design is the ability for users to interact with one another. This user-to-user interaction relies heavily on the smartphone application.

The measurable metrics that will be affected by the failure of this subsystem are related to the system requiring minimal user interaction: the user can access status information in under sixty seconds, and the user is notified of system errors requiring attention. Without the smartphone application, there is no other way for the user to be able to do either of these tasks.

Because the smartphone application is such an integral part of the design as a whole, the failure of any of its individual parts can have a large impact on the end product. An all-out failure of this subsystem would be considered a catastrophic level on the risk matrix. However, the likelihood of the whole subsystem failing is low. Therefore, it is important to

look at the components that go into creating the smartphone application subsystem, and evaluate each with regard to their risks. From the work breakdown structure, the components and tasks associated with the smartphone application's critical path are the integrated development environment, user experience design, and connection of the application to the server.

The integrated development environment chosen was Google's Android Studio. The risks associated with the use of this application, and the related risks of how our team chose to develop the application, rank low on the likelihood scale (level 1), and can potentially have a tolerable impact (level 2). Since this development environment is from Google, who created the Android operating system, the risks of the development environment not having the tools needed for proper smartphone application development is nonexistent. The other main risk of this software is a risk of data loss.

The development environment was initially only installed on one teammates computer, and all files were hosted locally on that machine. The possibility of the failure of that machine posed a significant risk to the completion of the project, as the loss of the code would mean that time would have to be dedicated to rewrite it. This risk was deemed to be likely (level 3), and have a serious impact on the project (level 4). This risk was categorized as a schedule risk. To mitigate the impact of this event, we created cloud backups of the source code for the application, so that the application can be recreated from any machine. Since Android Studio is a free-to-use software, any team member can quickly and easily install it, so its failure on one machine was considered to be low on the impact scale.

The next portion of the smartphone application is the ability for it to send information to the server. This information can be things such as changes to the lighting timing, or data that pertains to an agreement to trade produce between two users in a given geographical area. If the smartphone application were to fail to send data, it could pose a level 4 range impact on the project. It would be a serious impact due to the user not being able to change any settings on the aeroponics system, and not being able to connect with other users in the way we intended. The likelihood of this risk was rated as likely, because there are many points of failure. These points of failure include loss of internet connection on the smartphone and bad coding on the host end of the application.

The risk of failure to connect can be mitigated by ensuring the code for the smartphone application is robust, and well tested. In the event of a connection failure, there will be code to ensure that the system can notify the user on where the connection failed. Also, the aeroponics system will have a base case for watering and light timing, so that the user doesn't need to immediately set those parameters to get started.

E. Database Failure

Similar to the smartphone application, failure of the database, or connection to the database, can pose a significant problem to the deliverables for the design. The critical path subtasks of the database are the data structure creation, connection to the server for both the user's smartphone as well as the microcontroller, login authentication, and transmission of data to the smartphone application and microcontroller.

The measurable metrics pertaining to this section of the risk assessment fall under the feature that the system encourages community interaction and local consumption, as well as the system requiring minimal user interaction. Under these features, the specific measurable metrics are that the system can connect ten simultaneous users, and the user will be able to make up to two simultaneous trading agreements with other users.

This path of the project has a relatively low probability to fail; outside of the code for the server and connection to the server on the client side, there is a low probability that the server will fail. The low probability of failure of the server was one of the key reasons for choosing to use Google's Firebase and associated tools; since the tools are backed up by one of the largest technology companies in existence, a system failure on their part is extremely unlikely. However, in the unlikely event that this failure does occur, it would have a significant impact on the completion of this path (level 3) due to how reliant our design is on Firebase. There are very few alternatives for moving our project. The other risk involved with this path is the event that the code developed is erroneous and not robust. While this can be likely if the code is not tested adequately before the deployable prototype, the impact is tolerable, as the code can be easily updated to include any bug fixes. The risk associated with the Firebase server service was deemed inevitable; we can only mitigate the effects.

Building off the risks that are unavoidable, a data breach regarding user authentication falls under this category. In the event that there is a breach in the database's security system, there is not much we can do on our end. This risk, although unlikely, still poses a potential impact on the project. The servers that our project resides on can be housed anywhere around the world, which itself can increase the risk associated with the use of Google's servers. In order to deal with this risk, our team can only use the tools available by Google to monitor any suspicious activity on the server.

The ability for the server to send data to both the microcontroller and the smartphone is pivotal in creating the system that was intended in the design. As with the server failure, the likelihood that the server will not be able to send data is low, and presents a significant impact on the measurable metrics for the design. The risk can be mitigated by testing multiple scenarios in which the server would transmit data.

F. Microcontroller Failure

Another important electrical device that we have is the microcontroller and all the other small components that connect to the microcontroller. This includes the analog to digital converter, the wires, and the soldering boards. Like the sensors, these devices are easy and quick to replace. We look for the devices and models that are easy to purchase and can also be delivered fast. Although the Raspberry Pi 3 can be expensive, it is still easy and quick to replace. Furthermore, for the connection and fear of short circuits, we have our team work mostly with 3.3v and 5v which reduces the risk of us frying our boards.

In addition, whenever we are working with the wall voltage and or 24v, we make sure to have all the team members there to check the connection. However this is rare and only when we are finishing the final touches on the project would be connecting to the wall. Like before, the risk for the board failure is minimal but still can affect us harshly as we are relying on the Pi to be the brain of the system. After working on the project for two semesters, we were able to minimize all the risk toward our completion of our project.

G. External Risks

For the first semester, we looked at what event that can occur and affect our project greatly. We have the Paradise Fire as an anecdote to how we should prepare if something were to occur and affect our school. In the report, we create a plan to make sure we do not leave important equipment and hardware at school. Therefore, when the Pandemic hit, all our stuff was already at home, and the structure was in the hand of one of our groupmates. This allowed us to finish our project in time, and we were able to follow our work breakdown structure.

Another obstacle we were able to overcome is the necessary delivery of our components. Although we mostly relied on Amazon, we also have group mates who have some of the components that we needed instead of purchasing from Amazon. Fortunately, we also purchased all our equipment and components beforehand. Even though the task of purchasing and working on a specific part of the project was not due, we make sure to purchase the components before especially if the component is less accessible like our pH sensor.

These external risks such as a world wide pandemic or local fire can greatly affect how we do our project and how our group would work under these new conditions. These risks have been in small consideration when we divided up the tasks. Firstly we group all the tasks that require the same components into the same group and give those tasks to one person. This way, if we were unable to meet, the groupmate would still be able to do their tasks. Although we can not see the future, we can prepare for disasters based on previous events and a risk analysis.

H. Engineering and Strategic Risks

The main strategic risks to consider is our timeline. We must consider what parts are going to take the longest and what to prioritize. The obvious risk in all of this is that we will focus on the wrong thing without knowing that another part is going to take much longer. Since this is our first time building such a system, these types of problems are almost unavoidable. Surprises are bound to pop up. The best we can do to mitigate it is to think through and plan the whole project to guess the best we can what will take the most time and in what order we must do things.

VIII. DESIGN PHILOSOPHY

The ideas behind our design comes generally from our design requirements and the vision of where our product would fit into a person's life. The goal of the design is simply to be an in-home gardening system that can make enough food to satisfy a person wanting to have one. Due to this our design had to take up as little space as possible while still growing enough food to make a difference to someone. The system had to be functional but at the same time had to have an attractive design. Making people want to have it in their home is critical. If the system works but no one is willing to place it in their home then we have failed. Another consideration was how the sensors would work with the device to give a person useful information. With that in mind we considered how an app could benefit a user in their growing pursuits while also helping a variety of healthy food to circulate through these communities. We also looked at what kind of plants could be grown and how to get quality food to a person without increasing the cost too high. There are also some non-engineering problems that we had to deal with in order to have a successful project such as algae growth, plant light requirements, and proper water-nutrient levels.

A. Spacing

For the spacing, we wanted our design to use its space effectively. This would be accomplished through building the structure 2 to 1 vertical to the floor space. This will allow our system to support more plants and still not clutter up the room where the system is set. We followed through with this design since the start of the design requirement for we knew that space utilization is very important for families that are starting their own personal gardens. Therefore, our first prototype was about 2 feet tall compared to 1 foot across. For our final this, we continue with this ratio to ensure that users are not inconvenienced by the size of our design.

B. Sensors

1) *Light Sensor:* The choice of the light sensors was very simple with choices both semesters. During the first semester it was clear the selection of cheap and premade light sensors was very slim. This

caused problems in the second semester that couldn't be fixed by replacing the part.

In the first semester we didn't know much about the proper measure of lighting but the sensor selection was simplified as most all the sensors output in lux rather than the prefered watt per square meter. With that only one sensor was capable of both high range and precision, the TSL2591. It far surpassed the lux range set with limited information during the Design Idea report and reached into what was seeming to be the actually needed range of light. The UV sensor was also simple because of the lack of choice, we went with an analog UV sensor that outputs a voltage proportional to the UV index. The UV index is a scale.

In the second semester problems came when trying to scale the number of visible light sensors. The sensor uses the I2C interface and has one unchangeable address so in theory only one could be used per device. We looked for another suitable light sensor such as the previous model, the TSL2561, which allowed for more addresses, but it was unavailable and other options would not work. The solution was adding an I2C multiplexer. It allows for up to 8 I2C connections with the same address and after some simple research was able to run without additional coding libraries. The team decided on three sensors because of the growing levels. The UV sensor was placed in the center to give a relative look at the amount of UV light since the precise intensity wasn't very important. It is more there to warn the user if the light is giving off too much dangerous light to the plants or worse yet them.

The only other problem came during testing when we found the light sensors would max out their readings if the light got close. This was a problem because the light was set to be much closer than the distance it was maxing out at. The solution was making sure the sensors were at a sharp enough angle to the light that they would continue to give values. One sensor still output a 0 when the full light's brightness was on it but it was able to be fixed with some code. The other problem was the test results for the commercial lux sensor were an order of magnitude smaller than the output of the sensors. We used the commercial lux sensor as the benchmark and lowered all of the signals by a constant to it's values. Lastly, the sensor readings from the commercial sensor were converted to PPFD with the help of an online calculator to agree that the range of 200-600 PPFD was successfully done.

2) *Humidity/Temperature Sensor:* For the humidity and temperature sensor, we went with the DHT22 sensor (See appendix B) as it meets all our requirements, and like stated in our risk management, it is easy to order and delivered quickly. We went for two DHT22 sensors, one for the outside and the other one is for the inside. We wanted to measure both the humidity and temperature of where the plants were going to be and where the roots would be also. Furthermore, we needed the data from these sensors to control or to be notified of failure. For example, the humidity inside the box is an indication that the water delivery system is working. The temperature that comes along with the humidity sensor allows it to measure the relative humidity.

Although at first, our ambient temperature and humidity sensor was just there to let the user know the temperature and humidity. It wasn't really used for any other thing until we got the new lights. The new light was 600 watts light that can easily heat up the ambient temperature if left for too long in a small enclosed space. Since our system is supposed to be within a household, some may put the system inside a closet. This will allow the system to heat up easily and the ambient temperature and humidity data will help notify the user.

Also since one of our groupmate was familiar with the DHT11 which is a smaller and weaker version of the DHT22, he would be able to control

this sensor easier. That was the case and this sensor was popular among hobbyists; therefore it has a lot of support such as libraries and even tutorials to control the sensor. When this was integrated into our system, we used the Adafruit library that easily helped us get the data we needed. Following the Adafruit tutorial, we got this setup quick and easy. This sensor is minimal in its design. It has 3 pins: one for ground, one for VCC, and one for data. All of these make the DHT22 a perfect choice for us to use and build with.

3) *pH Sensor:* The pH sensor that we chose was the Gravity Analog pH Sensor with a meter (See Appendix B). This sensor was within our budget, and it's the only sensor that was able to stay underwater for continuous reading. As we were searching for the right pH sensor, we were informed that most pH sensors do not work well continuously submerged. Therefore, we have to look for a sensor that was still within our budget, and it will work well. This sensor is within our budget and meets the measurable requirements that we had agreed.

Working with this sensor was a bit harder than expected as the documentation for this sensor was still lacking. The documentation contains only a simple way to start using the sensor. Therefore, we spent quite some time getting this sensor running. This sensor also needed to be calibrated. This is done using the packages that came with the product. There are premixed pH solutions. Using these solution packets, we calibrated the sensor, and it worked great. However, we knew that this sensor still needed to be calibrated every once in a while due to its intrinsic quality. This sensor will slowly drift off the accurate value because the KCl solution inside the pH sensor will slowly deplete. It will make a small change in the signal. This small change added a hundred times will cause an offset in the reading especially when the signal is amplified with the on board amplifier. However, since this sensor voltage is closely linear compared to the pH, we can add an offset to the sensor collection program to adjust for the offset.

Although the temperature of the water can affect the pH reading, this pH sensor does not take into account the temperature of the solution. This is because we believed that since the user is going to put this in their house. The temperature of the water will not change drastically from the tap water that is going to be used for the tank. Therefore, this sensor was adequate, and it was able to do what we wanted.

4) *Electrical Conductivity Sensor:* For the EC sensor, we wanted a sensor that is able to stay under water for a long time and measure the EC of the nutrient solution accurately. We search for EC sensors that meet our requirements and are within our budget. However, all the EC sensors that can do this are from china or they are very expensive. We did want to risk buying a defective cheap EC sensor, so we build our own (See Appendix B). This sensor is a simple voltage divider with one part of the resistance underwater. The unknown would be the resistance of the water, and we use an ADC to measure voltage across the unknown to get its conductivity. However, this was easier said than done as water behaved differently from a resistor. First, the distance and shape of the probes would affect the reading. In order to minimize this effect. We create a simple probe with a known solution of EC and calculate the cell constant. The cell constant is a constant that tells us how the distance between the probes and the shape of the probe would affect our reading.

After creating this program, we started off with a few cheap items to act as the probe. One time we got two walls connected and used that as our probe. However, we decided to build one that can look less ridiculous and that is where we settle with our EC sensor. With this new EC probe and an ADC, we were able to measure the resistance of the water and convert to EC. This was a huge step in our sensor design. We were able to measure EC for cheap. However, after researching even further, we learned that applying voltage across water can ionize the

water making our reading usable. Most EC sensors work with AC signals which aids in overcoming this obstacle. Since we do not have an AC signal, we decided to test the probes and learn that we only apply the voltage for a sec and have the water "rest" for a minute, then the reading is still accurate. So, that is what we decided to do. Overall, our EC sensor worked great until it doesn't like when the calibration goes off or when the pH sensor is near it.

A major problem that we had to overcome was that the EC sensor and pH sensor was affecting one another in the water. We wanted to get rid of this interference between the two sensors by alternating when the sensors are active. However, since the pH sensor is passively working, this still affected both the sensors. We decided that if the distance between the two sensors would be large about 1.5ft, then the interference would be minimal. We went with that the final deployable prototype worked very well.

C. Android Application

Our team wanted to make the plant growing process more easily accessible to the average person. As such, we decided to tie the monitoring and control of the system to a smartphone application. We took a look at how to best approach the problem of making the application, and the system, available to the most people possible. This is why we chose to go with Google's Android smartphone operating system, which has the largest market share in the smartphone market.

After choosing the platform, we had to choose which version of Android to develop for. Choosing a version was more complicated than choosing the operating system because there were more benefits and drawbacks to take into consideration. With each new version of Android, there comes more features, and better security. However, not all phones and service providers allow the update to the newest version of the operating system; The newest operating system tends to be adopted by less than 10 percent of all Android users. With this in mind, we decided to go with Android Nougat, or API 24. While a bit dated, choosing to develop for this version ensured that our application would be able to run on at least 75 percent of current Android phones.

D. Design layout

Building off of the idea that we want our system to be accessible, we also wanted the smartphone application to be as intuitive as possible. This motive was the driving force of the design of the appearance of the application. We decided to keep things as simple as possible, while still maintaining a good level of functionality. Not only were the elements of the application taken into account, but also the colors and presentation.

The login page was designed to be standard when it comes to smartphone login pages, with the added brand identity of California State University, Sacramento (CSUS). We used the official CSUS colors and logo so that the user recognizes the application as being tied to the school. There are only two buttons on this page, which are labeled clearly.

The main landing page displays all sensor data for the system. We decided to display all the data on one page as opposed to on different pages for each category so that the user will be able to see the information without searching for it; it keeps the amount of necessary touch inputs to a minimum. The sensor page has different categories differentiated by color: green for lighting, blue for humidity, red for temperature, orange for the nutrient solution, and green for controlling watering and light timing. There is also a button to switch pages to view the users in the area. This page is laid out in a simple list type fashion so the user has no problem viewing and entering the information they need.

The integrated development environment we used, Android Studio, uses XMLto code the appearance of the application. XML was new to us, so there was a learning curve we had to overcome

while simultaneously achieving our goal for a simple design.

E. Non-Engineering Problems

The biggest non-engineering problem is actually growing the plants and learning information about farming. Although we knew a bit about farming and plants from our research, we were still unable to get all the information to actually help us grow the plants. We decided that the user should be able to grow their own plants with their knowledge as we cannot provide all the necessary information about the plants. However, we do have some default settings that are there just to be the baseline for the and not the actual way to keep growing plants.

Another tough problem that we did not realize before it was too late is that the system light was bright, and it could possibly disturb the user. However, a fix is to put a towel over it. This is one of the oversight that we have managed to see after building the prototype. Another would be the waste management. We were excited about growing plants and maintaining them that we forget that plants will die and the system would have to notify us of something about. This is an oversight in our design contract and our prototype. However, with these oversights, we depend on the user to provide what is necessary for the plant that our system is lacking.

IX. DEPLOYABLE PROTOTYPE STATUS

The deployable prototype at the end of our project has successfully met the requirements set out in our feature set. Our testing has shown we have met all of the measurable metrics as we had them laid out for our features.

A. System Collects Environmental Data

1) *Light sensor:* The requirement for the light sensor is that it's able to measure a spectrum of 200-600 PPFD and separately that we are able to measure the UV light intensity. The light intensity spectrum was tested using a commercial lux sensor. It gave accurate lux values that could be used to calculate the PPFD and to test the accuracy of the sensor. The sensors gave data much higher than it should be but code was able to fix the problem. The calculated PPFD using the commercial light sensor though was 166-614 PPFD, just better than the desired range.

2) Humidity Sensor: Our humidity sensors worked great, and both the sensors were able to measure the humidity accurately after testing the sensors. To perform the test, we use the saturated salt method which when put within an enclosed space will create a relative humidity of 75%. After we left the saturated salt within a seal bag for a day, we tested the humidity with both of our sensors which resulted in both sensors being able to measure a relative humidity of 75% and 80%. We knew that the second sensor was offset by 5% relative humidity, so we tested to see if it was just a one time thing or if the sensor had some kind of defect measuring the humidity (See appendix B for data).

We then performed a stability test that showed that both the sensors are stable; however, the second sensor was offset about 5% relative humidity. This means that to have the correct humidity for the second sensor, we created an offset in the software to help this defect. All in all, our sensors are still working and measuring accurately during our growing process.

3) Temperature Sensor: Since our humidity and temperature sensors are conjoined, we did similar tests with sensors. For accuracy, we use a multimeter temperature sensor to measure the ambient temperature. We ran both the sensors and compared it with the multimeter. It shows that both the sensors are within +- 1F of the accurate temperature; therefore, we knew that both the sensors were accurate. However, since we had an issue with the relative humidity, we wonder if the temperature sensor has

the same problem as the humidity with an offset. However after testing the sensors for a day, both sensors' temperature were within one another. This means that both the sensors were working fine (See appendix B for Data).

4) *pH Sensor:* For the pH sensor, we went with off the shelf sensors that have the required measuring depth for us. We followed the instruction and calibrated the sensor. It was working very well, and after testing its reliability and accuracy with a premixed pH solution, we learned that the sensor was within what we needed for our measurable metric.

However, a problem occured when you put both the EC and pH sensor into the same solution. They would interfere with one another, and so, we have to solve this issue. For this, we believed that if the distance between the two sensors were great enough, the interference will be minimal. Therefore, we did a test moving the two sensors 6 inches apart to 1 foot apart. From this test, we found that the interference drops dramatically when the sensors are 1 foot apart. Furthermore, our final prototype has the two sensors about 2 feet apart which even further reduce the interference.

5) *Electrical Conductivity Sensor:* The EC sensor was homebuilt, so we have to put a lot of effort to test its validity. We did initial testing for the sensor when we created the sensor, and it showed promises. We then created new probes for the sensors and did the accuracy test by using a known EC solution and measuring it. It was able to measure it within our required metric. Furthermore, we assumed that the sensor may require calibration every once in a while, so we did a stability test for a day to see how much change there would be. However, after the test, we can say that the calibration of the EC sensor will be quite long as it did not change in a day. According to the data, the sensors tend to stabilize after 15 to 18 hours.

B. Automated Growing Process

1) *Nutrient Solution Regulation:* The nutrient solution regulation consists of the EC sensor and a peralstic pump. The EC sensor will measure the EC and report back to the raspberry pi. From there, it will look at the EC threshold set by the user and add the nutrient mixture until the desired amount is reached. This is still working although the EC pump is very slow in adding the nutrient solution. However, this is to our advantage because it allows for more concentrated nutrients to be within the two bottles in the systems.

2) *Water Pump For Plants:* To control the water pump, we had set it up to where the pi will get the user input from the data and set its own interval. This interval was easy to test and just changing the number on the database changes the interval to which the water was to be sprayed on the roots. This is still working and able to be reproduced easily through the use of the relays.

3) *Light Interval Control:* The light interval control is similar to the water pump. The pi will read the data from the database and change its own interval to match the database. This allows the user to set their own interval that they would like. For right now, we have it to where the user would set a time, and the lights will turn on at that time for 3 hours.

C. Requires Little User Interaction

1) *One Week between fillup:* To meet this requirement, our system was able to hold 5 gallons of water. This water is able to go a week without being used up. As demonstrated when we grew 12 plants. However, there was a leak during our first test. This was quickly fixed but not before it did some damage to the plants. With this fix, the water lasts pretty long.

2) *User accessing information under 60 Seconds:* Our team was successful in achieving a below 60 second access time to the system's sensor information. This was possible through a simple login

system, as well as a layout design that ensured the user can access data in a timely manner.

3) *User is notified of system Errors:* Our system has the ability to notify users in the event that there is an issue with humidity: if the outside humidity is near the inside humidity, this indicates that there is an issue with the watering system. If there is no light sensor data when there should be, the system also throws an error.

D. Encourages Community Interaction

1) *10 Simultaneous Users:* The app must enable at least 10 people to be logged on at the same time. The Firebase real time database is rated to support up to one hundred simultaneous users. Our team was able to test 11 simultaneous virtual Android machines before running into performance issues with the host machine, as seen in the testing results in Appendix B.

2) *Trading with other Users:* The initial design idea behind the user being able to contact other users of the system was to allow the user to make user contracts of trading produce with other users. The user would have a user profile and be able to create a chat session with other users to discuss the trading of grown produce. With the campus closures and surrounding disruptions, our team scaled back our expectations regarding this measurable metric, with the solution being to allow users of our design to be able to contact other users within the same zip code, with an ability to opt out of the feature.

E. Minimal Environmental Impact

1) *Maximum Noise Level 70 dB:* After using a phone app to test the noise level of the system, we can conclude that the noise level of the was well below 70dB. We used a phone app that measured a level of about 40dB. This means we are well below the maximum noise level.

2) *Indoor Vertical Design:* For the indoor vertical design, we went with a 2 to 1 ratio where it will be two times as tall as it is long. This was achieved by our deployable prototype (See appendix D).

3) *System connects to Wifi for 150ft:* We did a quick test to see if the system will be able to reach up to 150ft. This was done by moving the raspberry pi relative to the router. The pi was able to meet this requirement if there are no big obstacles in between the two devices. However, for future proof, we added an additional wifi adapter module to the pi, and that was able to meet this requirement quite easily.

X. MARKETABILITY FORECAST

There are a few factors to consider when talking about taking our prototype to market. First, it is important to actually understand the market. What is currently for sale and what is and isn't successful in the market is important to consider. Market research would have to be done to determine what price people would actually pay. Without knowing the target price it is quite difficult to develop something. Once these things have been determined the next step is to make prototype revisions. These are all the changes that we need to make to bring our system to market.

A. Market

There are very few aeroponic systems on the market. Currently hydroponic systems outnumber aeroponics by far. This is very much a positive. This means that there is a large amount of possible growth in the market. The only aeroponic systems that are available are very expensive, sometimes over $2000, and they are very large generally. Our system that is completely contained and very efficient space wise with a smartphone app is quite unique. If we can keep the price reasonable there is legitimately some potential for success. What a "reasonable price" is would need to be determined through much more

market research. Most likely it would be well under $1000.

One area that our project is truly unique is the community aspect of it. No other product tries to foster a community of people growing their own produce. People talking to other growers and sharing tips and information along with trading produce is what could truly make this product a success. Since this aspect is built right into the app, it is accessible enough to actually be useful.

B. Prototype revisions

To truly bring our system to the market a few changes will need to be made. Corian is a great material but it is very expensive per unit compared to something like injection molded plastic. If many units were to be made a different material might be chosen. We would also have to refine the help within the application. We would add a robust help section that would guide people who are new to growing plants. Currently the system gives the user a great deal of control but the problem with that is that the user can very easily kill their plants. Currently it is possible to manually adjust the sprayers so if someone wishes to have more than one type of plant, and they have different watering needs, they can adjust the amount sprayed. This works but a more streamlined solution could be made. In the final product it should be easier to have plants with different water needs and possibly even different lighting needs.

XI. CONCLUSION

Our goal was to create a device that was able to supplement the diets of people in urban areas with more healthy food. People in heavily populated urban cities are generally in situations known as “food deserts” where unhealthy food is far more cheap and available than healthy alternatives. This is the focus of the growing trend toward urban farming where fruits and vegetables are grown within cities to cut down on the cost and pollution associated with transporting that food.

To achieve this goal we set out features we thought would be necessary and/or beneficial to achieving these ends. We decided there needed to be sensors to gather the data that might be relevant to someone who wants extra ability to control the food they grow. The sensors also allow for the system to react on it’s own to changes an uneducated user may not notice. We wanted the process to be automated for the same reason, to allow people less educated on growing food to get a foot in the door. It was also important that the system in general doesn’t require too much user attention, again it is time they might not be willing to spend. We decided the device should be connected to an app to foster a community of people wanting to get more healthy food in their diet. Lastly, since the design was meant for an urban area it should take up minimal space and not be noticeably loud. We then split these features into subtasks to be split amongst each other.

The funding and materials for this project came from the team members.Thanks to donations from the employer of one of our teammates we were able to get over $600 worth of materials, about the same amount of money we collectively spent both semesters, $558. It is also more than half of the $1092.4 total spent for the deployable prototype. In the end the budget stayed under control with none of our hardware failing and needing to be replaced. Almost all $558 can be seen as hardware or materials on the deployable prototype so little money was wasted in testing and development.

There were several milestones throughout the course of senior design, these included all of the group assignments but also several important dates. The lab prototype, midterm progress review and the end of project documentation were the three biggest milestones where our individual tasks had to be made to work together. The lab prototype made us get basic but thorough versions of our designs working together. The mid-term progress review had us finishing the project aside from small issues and thus

everyone's designs would have to work together. This end of project documentation had us compiling all of our work from the past two semesters into a representation of the time spent on this project.

The five sections of our feature set were divided by their subgroups among the four of us. Approximately 820 hours was spent on the project between all of us. Kevin was responsible for the automation programs and hardware and the wireless communication, most of his time was spent working on the automation aspects. Yutthachat was responsible for the sensor programs and hardware, most of his time was spent on wiring and acquiring data from the sensors. Adrian was responsible for the android application and operation of the database. He worked primarily on getting the Android application working. Brandon was responsible for the design and manufacturing and wiring of the chassis of the device. His time was spent designing and manufacturing the mechanics of the project.

Next we talked about the risks associated with our project and steps we've taken to avoid these situations. We wrote about the possibilities of hardware and software problems and the steps we've taken to address them such as ordering spare parts and using an online server, respectively. We ranked the danger and likelihood of various possible situations and used that to decide which problems would require the most urgent attention. Looking at the external risks became very important this semester with the spread of COVID-19 and the shutdown of the school and other major institutions such as vendors.

We went from there to talk about the design philosophy behind the various parts of the device. Choosing sensors came down to what was available at an inexpensive price. Still, the gravity pH sensor was an expensive necessity to meet the feature set. Software was designed with simplicity in mind as to get people with less agriculturally educated people into growing their own food. Lastly, hardware like the lights and pumps were carefully chosen to meet the feature set and to successfully grow food

Our project at this stage has met all of the feature requirements set out for it. Extensive testing has been done on the sensors to make sure they provide relevant and accurate information. The programs

We believe our device can compete with the other indoor growing systems on the market today. Cheaper systems lack the functionality ours has while similar devices are more expensive. Improvements could certainly be made to the design though such as the use of a cheaper structure material or optimizing the system to better care for the plants. This project has been a great learning experience for all of us in the issues that come during large scale and cooperative projects.

REFERENCES

[1] R. N. Amundsen, "Urban farming: Victory gardens for sustainable communities," *2013 Energy and Sustainability Conference*, 2013.

[2] Meharg, Andrew A. "Perspective: City farming needs monitoring." Nature: International Journal for Science, 17 Mar. 2016. Google Scholar, doi:https://www.nature.com/articles/531S60a. Accessed 16 Apr. 2020.

[3] Ackerman K, Conard M, Culligan P, Plunz R, Sutto, M Sustainable Food Systems for Future Cities: The Potential of Urban Agriculture*. The Economic and Social Review [Internet]. 2014 [cited 2020 Apr];45(2):189-206. (ScienceDirect)

[4] Zhange. He, Asutosh. Ashish, Hu. Wei, "Implementing Vertical Farming at University Scale to Promote Sustainable Communities: A Feasibility Analysis," Sustainability, 2018. [Online]. Available:https://doi.org/10.3390/su10124429.

[5] "Food Waste FAQs," USDA. [Online]. Available: https://www.usda.gov/foodwaste/faqs. [Accessed: 10-Apr-2020].

[6] "How many watts are required per square foot of grow space", LEDgrowlightdepot.com "https://www.ledgrowlightsdepot.com/blogs/blog/16326275-how-many-led-watts-are-required-per-square-foot-of-grow-space April 15,2020

[7] "Why is PAR Rating a Big Deal for Indoor Grow Light Systems?" GrowAce.com https://growace.com/blog/why-is-par-rating-a-big-deal-for-indoor-grow-light-systems/ April 15,2020

[8] Crowe, Kevin, “Total Cost Of the Project.” [Excel] 2020

[9] Thao, Yutthachat, “WBS Hours Breakdown.” [Excel] 2020

[10] Thao, Y., et al. “Risk Matrix” [Excel] 2020

[11] Thao, Yutthachat. “Project Documentation.” [Photos] 2020

[12] Thao, Yutthachat. “Electrical Connection.” [Draw.io] 2020

[13] “Raspberry Pi 3 – Model B.” *PiShop.us*, www.pishop.us/product/raspberry-pi-3-model-b-armv8-with-1g-ram/.

[14] Aosong Electronics Co. “DHT22 Datasheet.” [Datasheet]. Accessed on October 23, 2019. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[15] Adafruit Industries. “Analog UV Light Sensor Breakout - GUVA-S12SD.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1918.

[16] DiCola, Tony. “Raspberry Pi Analog to Digital Converters.” *Adafruit Learning System*, learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008

[17] MCP3008 - 8-Channel 10-Bit ADC With SPI Interface.” *The Pi Hut*, thepihut.com/products/adafruit-mcp3008-8-channel-10-bit-adc-with-spi-interface.

[18] “Gravity: Analog PH Sensor/Meter Kit V2.” *DFRobot*, www.dfrobot.com/product-1782.html.

[19] “SainSmart 4-Channel Relay Module.” *Amazon*, https://www.amazon.com/gp/product/B0057OC5O8/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1

[20] Thao, Yutthachat. “*EC Sensor Setup.” [*Draw.Io] 2020

[21] “Power Supply.” *Amazon*, images-na.ssl-images-amazon.com/images/I/61VB5ltH4zL._AC_SL1000_.jpg.

[22] “LM295 Buck Converter.” *Amazon*, images-na.ssl-images-amazon.com/images/I/41YI1K5raEL.jpg.

[23] “King Plus 600w Double Chips Led Grow Light Full Spectrum with UV and IR for Greenhouse and Indoor Plant Flowering Growing.”,*KingLED*, https://kingledlights.com/products/king-plus-600w-double-chips-led-grow-light

[24] Hilitand. “Peristaltic Pump Product Page.” *Amazon*, www.amazon.com/Self-Priming-Peristaltic-Aquarium-Chemicals-Additives/dp/B07J2JDT54.

[25] AUBIG. “Aubig DC 12V Brushless Magnetic Drive Centrifugal Submersible Oil Water Pump 500L/H 5M/16ft.” *Amazon*, www.amazon.com/DC-12V-Brushless-Centrifugal-Submersible/dp/B00C6XNB50/ref=sr_1_10?dchild=1&keywords=waterproof+12v+water+pump&qid=1587780102&sr=8-10.

[26] “Adafruit TSL2591 High Dynamic Range Digital Light Sensor” Adafruit.com https://www.adafruit.com/product/1980

[27] “TCA9548 I2C Multiplexer” Adafruit.com https://www.adafruit.com/product/2717

[28] Thao, Yutthachat. “Testing Data.” [Graphs]2020

[29] Crowe, Kevin, “Test Results - Light_Angle” 2020

[30] Crowe, Kevin, “Test Results - Light_lux/PPFD” 2020

[31] Crowe, Kevin “Raspberry Pi programs flow-chart”2020

[32] Crowe, Kevin “Pi programs” 2020

[33] “Subprocess” https://docs.python.org/3/library/subprocess.html

[34] Buck, Brandon “Construction” 2020

[35] Buck, Brandon “Tape compression” 2020

[36] Buck, Brandon “Sanding” 2020

[37] Buck, Brandon “Brace Cutting” 2020

[38] Buck, Brandon “Powder Coating” 2020

[39] Buck, Brandon “Dimensions” 2020

[40] Thao, Yutthachat. “Plants Documentation.” [Photos] 2020

[41] Barrera, Adrian“Login logic flow-chart”2020

[42] Barrera, Adrian “Login pseudocode flow-chart”2020

[43] Barrera, Adrian “User profile”2020

[44] Barrera, Adrian “Community Contact Logic”2020

[45] Barrera, Adrian “Layout pseudocode flow-chart”2020

GLOSSARY

Aeroponic - a plant-cultivation technique in which the roots hang suspended in the air while nutrient solution is delivered to them in the form of a fine mist.

Electrical Conductivity- A measurement of the amount of nutrients in a solution.

Risk - exposure to the chance of injury or loss; a hazard or dangerous chance

Polystyrene - a synthetic aromatic hydrocarbon polymer made from the monomer styrene; it can be solid or foamed.

pH - quantitative measure of the acidity or basicity of aqueous or other liquid solutions.

Food Deserts - places where healthy food is less accessible than cheaper and unhealthier options.

Photo-Synthetic Photon Flux (PPFD) - the amount of photosynthetically active photons (400-700nm) hitting a surface per unit area per unit time.

Android - an open-source operating system used for smartphones and tablet computers.

Firebase - a Backend-as-a-Service — BaaS — a next-generation app-development platform on Google Cloud Platform

APPENDIX A.
User Manual

IMPORTANT

READ THIS MANUAL CAREFULLY before attempting to operate the system.

DO NOT EVER LOOK DIRECTLY INTO GROW LIGHT. DIRECT OBSERVATION MAY CAUSE DAMAGE TO EYES.

BEFORE OPERATION: Ensure there are no cracks or holes in the Corion enclosure.

1.0 Overview:

The Green Wall is an automated aeroponic growing system. It has an Android smartphone application to monitor and control the environment to emulate the perfect growing condition for plants. It is equipped with various sensors and controls to help the user grow their selected plants.

1.1 Features:

- *Able to maintain 12 small plants:* After setup, system can support plants for a week without user intervention
- *Monitor relevant environmental information:* With the assistance of the Android smartphone application, users can monitor temperature (outside and inside system), light information (UV and LUX), nutrient solution information (EC and pH), and humidity (outside and inside system).
- *Control lighting and watering times:* With the assistance of the Android smartphone application, users can control the watering time and light times for the system. After initial setup, light and water timing are set to default times.
- *Interact with other Green Wall users:* Included in the Android smartphone application is the ability for Green Wall users to contact other users in their zip code for the purposes of trading produce.

1.2 What's Included:

- LED grow light x 1
- Corian enclosure x 1 [Figure 1]
- Two part nutrient solution x 1 [Figure 2]
- 2 inch net pots x 20 [Figure 3]
- Green Rockwool x 1 [Figure 4]
- Raspberry Pi x 1
- Light sensor x 3
- UV sensor x 1
- Temperature and humidity sensor x 2
- EC sensor x 1

- pH sensor x 2

Figure A1. Corian enclosure adapted from [11]

Figure A2. Two part nutrient solution adapted from [11]

Figure A3. Net pots adapted from [11]

Figure A4. Green Rockwool adapted from [11]

1.3 Additional Requirements:

- Android smartphone, API 24 or newer
- Plant seedlings
- Distilled water

2.0 Specifications:

Dimensions:

Depth: 12 inches
Width: 24 inches
Height: 30 inches
Weight: 35 pounds

Material:

Material: Corian/aluminum

Color: white

Thickness: .5 inches

Other:

Operating Systems:

Raspbian Linux ver. 4.19 (microcontroller)

Google Android API 24 (smartphone)

3.0 Getting Started:

To start the setup, we recommend the user purchase plant seedlings, as the Green Wall system has not been tested for the germination of the seeds. For example, a great and healthy plant to grow is Basil, mint (any type), and small herbs.

Pick a place where there is a lot of open room as it will allow the plants to have access to more air. The Green Wall is intended to sit flush against a wall; avoid using a well enclosed space such as a closet or small room.

After picking a designated place for your Green Wall, you will need to prepare your system for your new plants. First, fill the basin with 5 gallons of distilled water. Then, add a small 10ml of each of the two part nutrient solution. This is just a base value for your plants. We recommend that you research just how much your nutrient your type of plants require. We have an internal Electrical Conductivity Sensor to help with the nutrient in your water. After filling the water and adding an initial nutrient, fill up the two bottles within the system with the nutrients. Mix a 10 part water one part nutrient solution into the bottles. Those two bottles will act as a regulator which will automatically regulate the desired solution density.

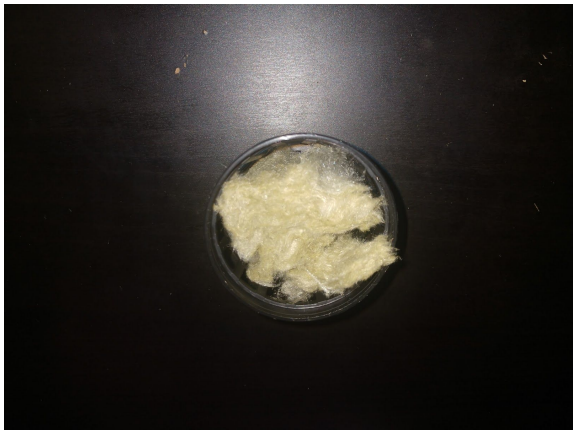


Figure A5. Rockwool in net pot adapted from [11]

After you are done, you are ready to transfer your plant. Remove your seedlings and gently remove most of the soil from its roots. Put your plant's root in first into the net on the system and then add rockwool around it to hold it in place. Similar to the Figure 5 below but you plant in the middle.

Now plug in the system and watch as the plants start to get water. To configure the light setting and the water interval, we recommend that you immediately use your provided credential and connect to our app.

3.1 Monitoring System:

After the initial setup, it is recommended that the Android smartphone application be downloaded and initialized. Once the application has been downloaded, open the application and sign up using your email and password. After sign up, the sensor data page is shown. At the bottom of the page, press the “users” button and enter in the provided credential into the respective text field, and click update. This completes the setup for the account. The user is now able to see the data that relates to the aeroponics system by returning to the sensor data page.

3.2 Other Green Wall users:

To view the other users in the same zip code, press the “users” button at the bottom of the sensor data page. Enter in the desired zip code and press “update.” This will provide the contact information for other users in your zip code. To opt out of this feature, enter in a value of “0”.

4.0 Troubleshooting:

In the event that the system is unresponsive, follow this procedure:

- Ensure that the overhead light is switched to the OFF position
- Unplug the Green Wall
- Plug in the Green Wall
- Give the system 5 minutes to reinitialize
- If open, close Android smartphone application
- Reopen Android smartphone application and login

This hard reset procedure will return the Green Wall to factory settings. Custom water and light timings will be changed back to default. Please update these timings to avoid disrupting plants.

Appendix B.
Hardware

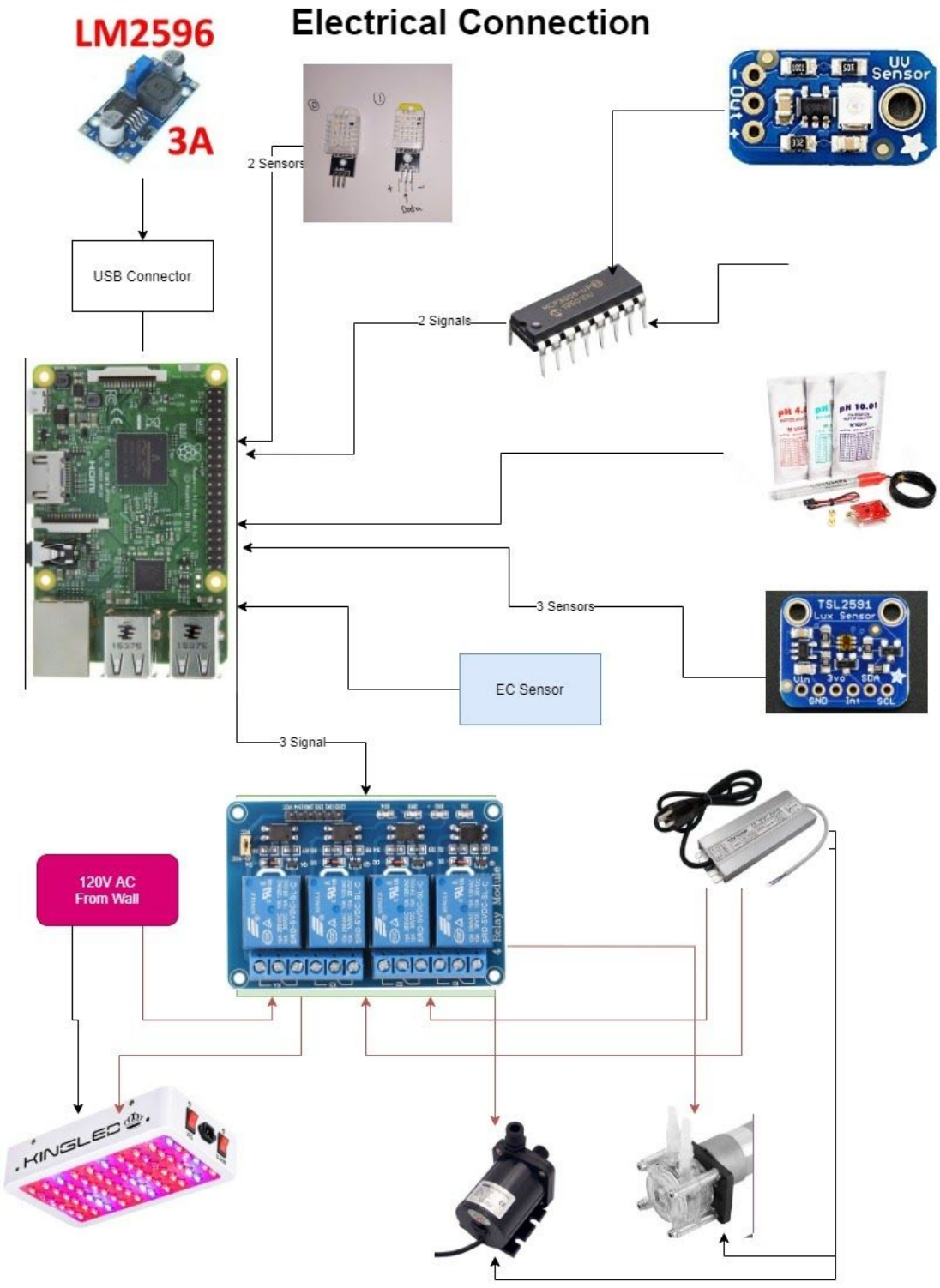


Figure B1. Electrical Connection Adapted from [12]

In this section, we will be listing all the hardware and components that we use for our project. Above is a simplified version of how we set up our overall electrical connections. We will be listing all these components. These are just the important components as the others are just minor details to get these components working. For the electrical components:

1. Raspberry Pi 3
2. 2 DHT22 Temperature and Humidity Sensor
3. UV Sensor
4. MCP3008 ADC
5. Gravity Analog pH Sensor
6. 4 Relays Module
7. EC Sensor
8. 100 Watt 12V Power Supply
9. LM295 Buck Converter
10. Kingsled 600 watt Led grow light
11. Peristaltic Pump
12. Waterproof 12v Pump
13. 3 TSL2951 Light Sensor

Why did we choose it?

Raspberry Pi Model 3

-Wifi Module
-Easy to Program
-Able to Run multiple programs
-Has multiple GPIO pins and specific pins

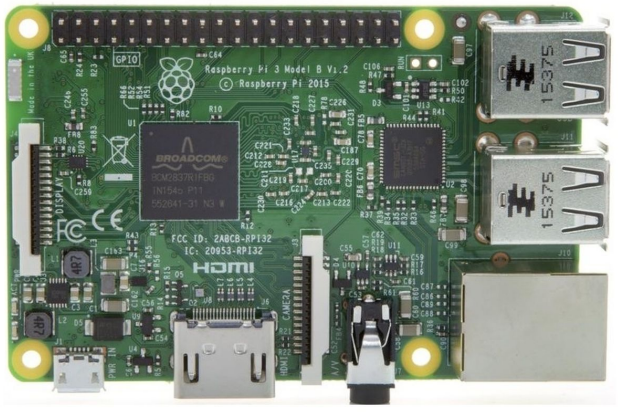


Figure B2. Raspberry Pi 3 Adapted from [13]

DHT22 Temperature And Humidity Sensor

-Fit Perfectly with our project requirement

Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Figure B3. DHT22 Specification Adapted from [14]

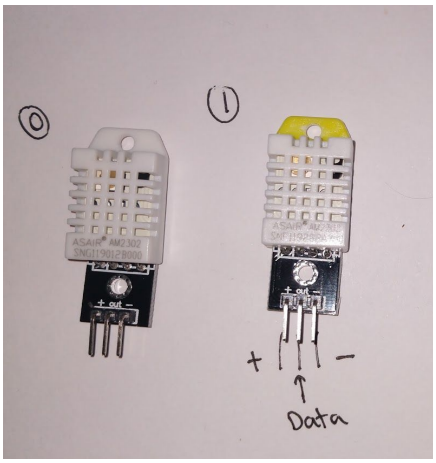


Figure B4. DHT22 Sensors Adapted from [11]

UV Sensor (Adafruit GUVA-S12SD)

-Able to measure UV index of 0-5

-Output Analog Signal

-.1v/step is equal to UV1/step

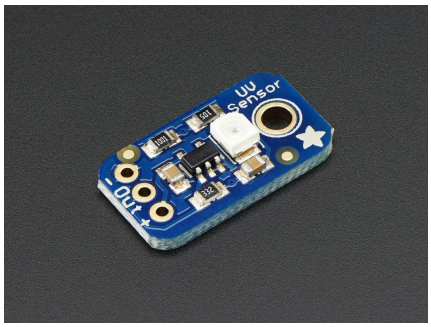
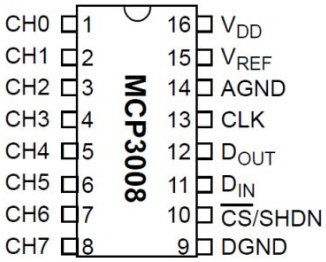


Figure B5. UV Sensor GUVA-S12SD Adapted from [15]

MCP3008 ADC

-8 Channel 10 bit ADC

-Able to be use with 3.3v-5v



- MCP3008 VDD to Raspberry Pi 3.3V
- MCP3008 VREF to Raspberry Pi 3.3V
- MCP3008 AGND to Raspberry Pi GND
- MCP3008 DGND to Raspberry Pi GND
- MCP3008 CLK to Raspberry Pi SCLK
- MCP3008 DOUT to Raspberry Pi MISO
- MCP3008 DIN to Raspberry Pi MOSI
- MCP3008 CS/SHDN to Raspberry Pi CE0

Figure B6. MCP3008 Wiring to Pi Adapted from [16]

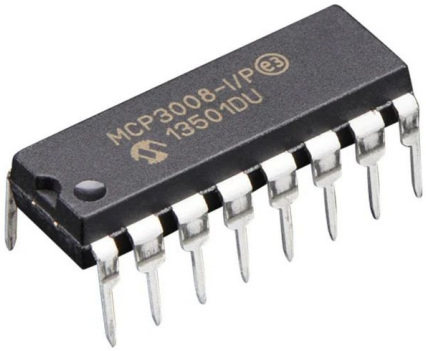


Figure B7. MCP3008 Adapted from [17]

Gravity Analog pH Sensor/Meter

Specifications

Reads: pH

pH: 0.1 - 14.0

Resolution: 0.1

Accuracy: +- 0.2

Response Time: Continuous Analog Output

Supported Probes: Any Type & Brand

Temp. Compensation: No (Unnecessary)

Data Protocol: Analog 2.7 - 0.2V

Figure B8. Gravity Analog pH Sensor Adapted from [18]

SainSmart 4-Channel Relay Module

- Logic levels controlled by Vcc: 3.3V
- Relay operates below 250V AC - 10A, or 30V DC - 10A
- Relays powered by JD-Vcc: 5V

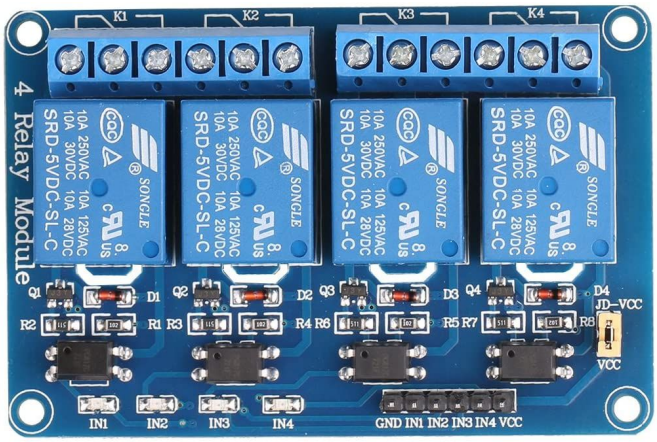


Figure B9. 4 Relay Module Adapted from [19]

EC Sensor

- Use stainless steel for the two probes
- 1Kohm for the known resistance

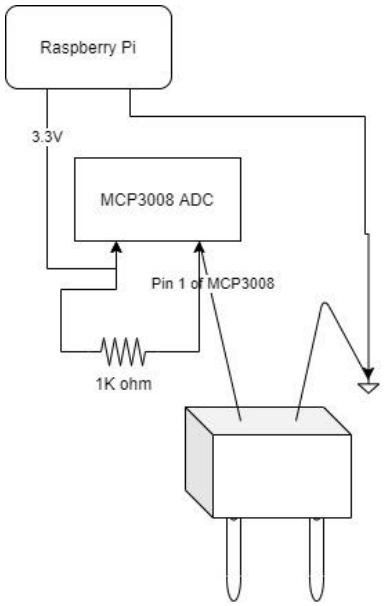


Figure B10: EC Sensor Setup
Adapted from [20]

100 Watts 12V Power Supply

Figure B11. 100 Watt 12V Power Supply Adapted from [21]

LM2596 Buck Converter

- (45 x 23 x 14) mm
- Vin: 3.0 - 40 V
- Vout: 1.5 - 30V
- Output voltage adjusted through potentiometer but Vout < Vin

Figure B12. LM295 Buck Converter Adapted from [22]

Kingled 600 Watt Grow Light

- (12.2 x 8.2 x 2.36) in
- 85 - 265 V
- effective 120 watt (equivalent to 600W HPS bulb)
- 60 10W equivalent LEDs
- Lifespan: >50,000 hours
- Recommended Distance: 2-3.5 ft

Figure B13. Kingled 600 Watt grow light Adapted from [23]

Peristaltic Pump

- Able to pump EC Solution
- Uses 12v

Figure B14. Peristaltic Pump Adapted from [24]

Main Pump 12V

- Able to be fully submerged
- Pump enough water for all our plants

Figure B15. Main Pump 12V Submersible Adapted from [25]

TSL2591 Visible/Ir Lux Sensor

- (19x16x1) mm
- Vin: 3.3V-5V
- Range: 188 uLux - 88 kLux
- Interface: I2C (7-bit address, 0x29)
- Temp Range: -30 - 80 (deg C)

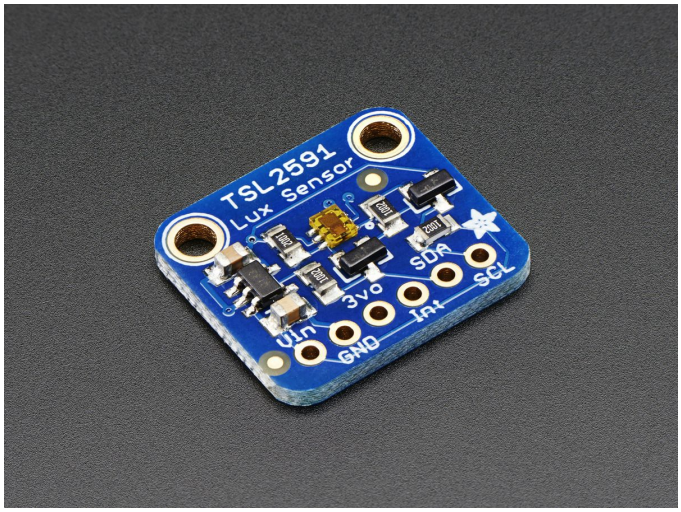


Figure B16: TSL 2591 Visible/IR lux sensor Adapted from [26]

TCA9548A I2C Multiplexer

(30.6 x 17.6 x 2.7) mm

Selectable I2C addresses: 0x70-0x77

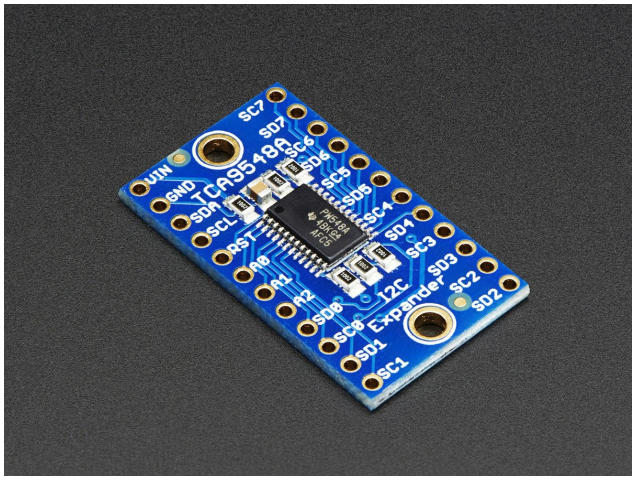


Figure B17: TCA9548A I2C Multiplexer Adapted from [27]

Testing Data

We will look at the most important components listed here and describe the test done to validate what the datasheet claims.

First is the pH, we wanted to see if the pH sensor can measure the pH accurately and stable through a day. Below is the stability inside a test solution of pH 6.82.

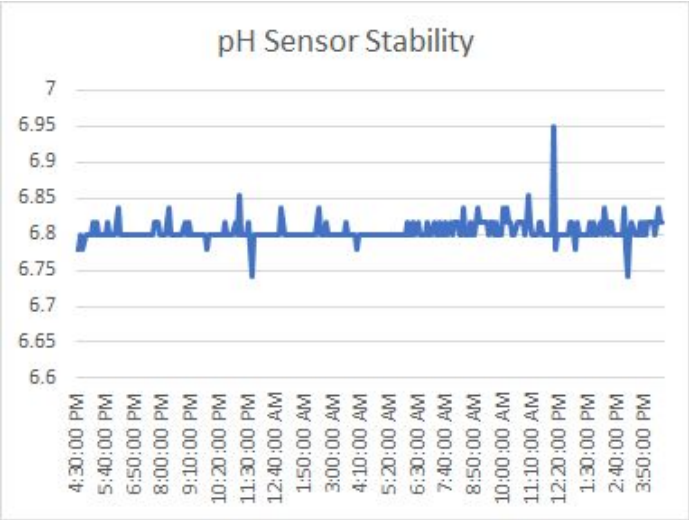


Figure B18. pH Sensor Stability over a day Adapted from [28]

Table BI. pH Sensor Test Adapted from [28]

Solutions	pH Sensor	Retail pH Sensor
30ml/100L	5.12	5.31
400mL/100L	4.6	4.55
pH Calibration Packet 4.01	3.99	4.05
7.01	7.04	7.12
10.01	10.00	10.15
Tap Water	8.07	8.01
30ml/100L solution	6.77	6.86
400mL/100L	5.77	5.64

Adapted from [27]

From both of the tests, we can see that the pH sensor is able to measure the pH accurately and the stability of the sensor over a day did not vary a lot. These are well within our measurable requirements. However, when both the pH and EC sensor are within the 1 ft of one another in the solution they would affect each other reading. Below is a test to see if we can figure out a way to minimize these effects.

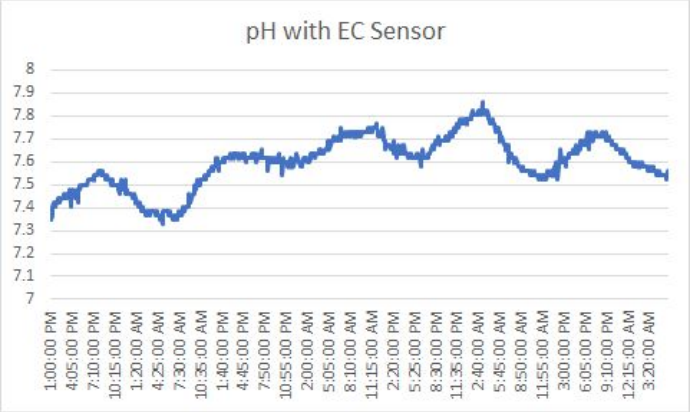


Figure B19. pH with EC Sensor Stability Adapted from [28]

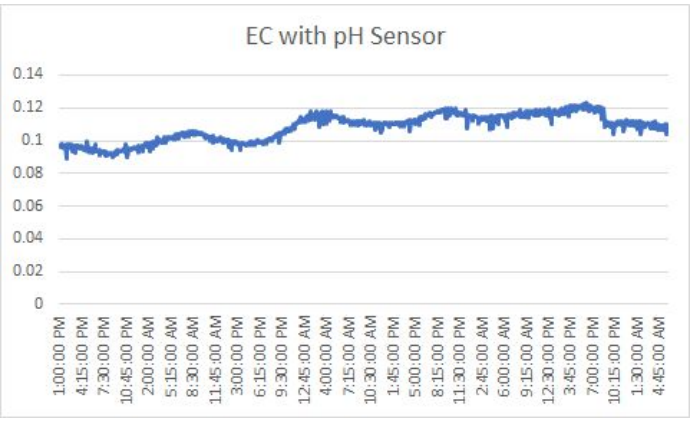


Figure B20. EC with pH Sensor Stability Adapted from [28]

In the first 6 hours, the two sensors were about 1 foot apart. Then the next data is when they are about 6 inches apart. At about 1 foot, the pH reads about 7.45 while the actual pH is 7.36. The difference is about .15 pH. For the EC Sensor, the EC read .09 mS/ cm which is close to the EC of .086 mS/cm. We can see that at one foot, the two sensors are already pretty closed to the accurate value. Therefore, when we built the deployable prototype we make sure they are two feet apart. This will greatly improve our readings.

Seeing that we are able to control the EC and pH interference, we needed to measure the EC stability and accuracy as well. Below is the stability in one day:

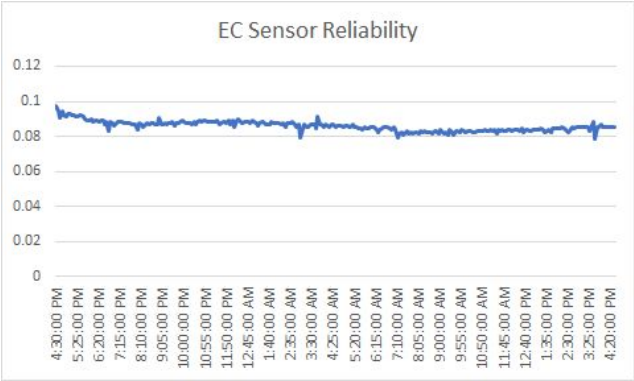


Figure B21. EC Stability over a day
Adapted from [28]

As we can tell from the data that we collected the EC tends to go off by about .3. The real EC was .86 while this graph shows that at first, the EC was still getting used to the environment. Later the EC sensor was able to settle near the actual value.

We will then also be testing the stability and accuracy of the DHT22 Sensors. We noticed that one of the sensors is offset by 5% relative humidity, so we are doing the test to see if we can actually see which one is the correct one. Below, we tested the stability of the two sensors.

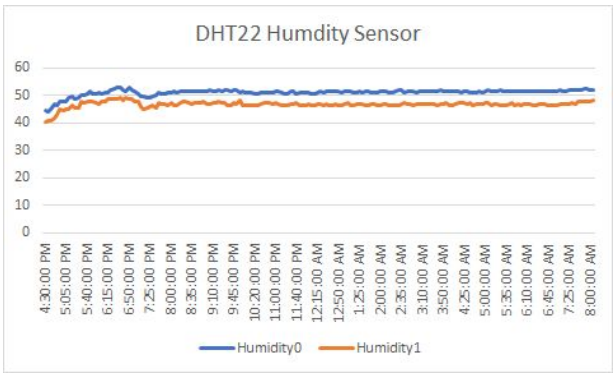


Figure B22. DHT22 Humidity Sensor Adapted from [28]

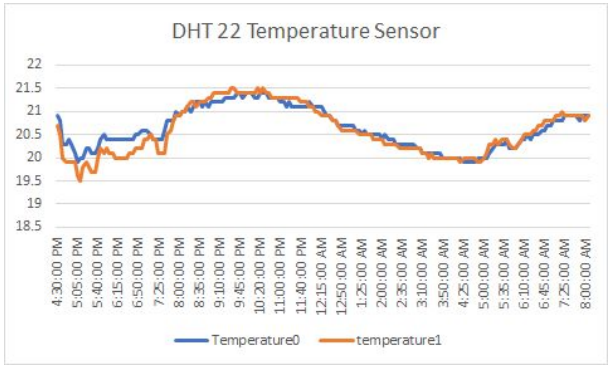


Figure B23. DHT22 Temperature Sensors Adapted from [28]

Looking at the result, we can see that the two sensors are able to measure the temperature and humidity at a stable rate. However for the humidity, one of them is offset by 5% relative humidity. To find out which one it is, we did an accuracy test. For the humidity, we create a saturated salt test which when placed within a sealed bag will create a humidity of about 75% relative humidity. We then put the two sensors in and measure the humidity to see if they are close to 75%. We found that the sensor1 was closest which means that sensor0 is offset 5% higher. For the temperature accuracy, we only needed to test if the temperature was within 2F of what it is. Therefore, we use a thermometer on a multimeter to measure the temperature and then using the two sensors, we measure the temperature. The real temperature is 71F while the two sensors measured 71.59 and 71.41F which are very close to what it should be.

Figure B24. Temperature Accuracy Test Result Adapted from [28]

Grow Light/Light Sensor:

Testing of the TSL2591 and the KingLED 600W grow light was a process of three tests. The first tested if the sensor required the light to be directly facing it or if the sensor could be angled and corrected with geometry later. The second tested if the light sensors were outputting the correct lux values, which was tested using a commercial lux meter the test was also used to see if the light range on the deployable was at least 200-600 on the PPFD scale.

The first test came back successful, results of the light shining on the parallel sensor can be seen in the figure below.

Angle = 90	Mode:Veg	Height: 24 in	
n	Time step	Time	Lux
1	10	0	63673.1328
2	10	10	63504.3024
3	10	20	63484.4736
4	10	30	63476.8848
5	10	40	63502.0992
6	10	50	63510.9936
7	10	60	63431.8416
8	10	70	63474.8448
9	10	80	63552.528
10	10	90	63426.2928
11	10	100	63545.6736
12	10	110	63432.4128
13	10	120	63435.1872
14	10	130	63479.496
15	10	140	63555.1392
16	10	150	63531.2304
17	10	160	63475.9872
18	10	170	63443.9184
19	10	180	63518.9904
20	10	190	63471.1728
21	10	200	63478.7616
22	10	210	63443.9184
23	10	220	63452.0784
24	10	230	63369.5808
25	10	240	63557.016
26	10	250	63414.4608
27	10	260	63524.9472
28	10	270	63405.5664
29	10	280	63405.5664
30	10	290	63422.0496
			Avg =
			63480.01824

Figure B25. angle_Veg-flat Adapted from [29]

The average value obtained from the data was used to compare to when the sensor was angled. I test it with angles from the ground of 45 and 60 degrees.

Avg =
63480.01824

Figure B26. Veg-flat average Adapted from [29]

This figure again shows the average of the flat sensor. Given another output the angle between those two outputs should be the inverse sine of the fraction made with the new number over the above average

Avg=	Angle=	Angle Error=
46760.2952	47.44380693	5.430682076

Figure B27. Veg-45 Angle Test Adapted from [29]

For example, here the average value of the angled sensor is 46760.2952. With the previous average, the angle should be arcsin(46760.2952/63480.0182) = 47.44, as it is. The "Angle Error" shows the percentage error from the ideal 45 degrees. The test setup itself was less than ideal so an error within 10% should be within reason.

Avg=	Angle=	Angle Error=
29172.05712	62.64211463	4.40352439

Figure B28. Veg-60 Angle Test Adapted from [29]

The error here is again low, supporting the idea that geometry can be used to overcome this sensor's inabilities. A similar test was done with the light in the other mode, bloom, the results were very similar.

Avg =
72756.9128

Figure B29. Bloom-flat Angle Test Adapted from [29]

Avg=	Angle =	Angle Error=
55142.01872	49.27887689	9.508615312

Figure B30. Bloom-45 Angle Test Adapted from [29]

Avg=	Angle=	Angle Error=
38185.64752	58.34261106	-2.762314906

Figure B31. Bloom-60 Angle Test Adapted from [29]

The angle error for this mode is also between 10%. This shows that angling the light can reduce the output without losing the accuracy of the sensor.

The second test was to both compare the light sensors to a commercial lux sensor to see how accurate their readings were. The results were very far off

		Bloom		
	n	Time step	Time	Lux
	1	10	0	19560
TOP	2	10	10	19570
	3	10	20	19550
	4	10	30	19560
	5	10	40	19580
	n	Time step	Time	Lux
	1	10	0	11280
	2	10	10	11300
MID	3	10	20	11250
	4	10	30	11330
	5	10	40	11240
	n	Time step	Time	Lux
	1	10	0	7640
	2	10	10	7660
BOTTOM	3	10	20	7600
	4	10	30	7590
	5	10	40	7620

Figure B32. Bloom Lux Test Adapted from [30]

This above table shows the measurements made by the commercial lux sensor. Measurements were taken at the three growing levels where the light sensors sit.

		Bloom	
	AvgT =	AvgM =	AvgB =
Lux	19564	11280	7622
PPFD	502.5901556	289.7780084	195.805672
		Range	
		196 to 503	

Figure B33. Bloom Lux Test 2 Adapted from [30]

This shows the averages for the data above. I used an online PPFD calculator to convert from lux. The calculator requests the spectrum of light and then converts your lux level to PPFD. Our light is “full spectrum” based on their product description. I thought the best fit would be the red blue and white LED option. The constant the website is using to calculate the PPFD was easy to find with some math and found to be .02569 PPFD/lux. Using this I was able to calculate the PPFD range from the bottom growing row to the top row.

		Veg	
	AvgT =	AvgM =	AvgB =
Lux	16784	9832	6476
PPFD	431.1732351	252.5795548	166.3654594
		Range	
		166 to 431	

Figure B34. Veg Lux Test Adapted from [30]

Using the same method as above I tested the results of the “Veg mode” The results were lower PPFD values in general. The range is not much thinner than the bloom option but it does reach lower into low light.

		260 to 614	
		Bloom+Veg	
	AvgT =	AvgM =	AvgB =
Lux	23900	14214	10,134
PPFD	613.98	365.15	260.34
		Range	
		260 to 614	

Figure B35. Bloom+Veg Lux Test Adapted from [30]

With both light modes enabled the system reaches its peak of 614 PPFD. This means the total range of light being shone on the grow space is 166-614, considering different modes, with a plenty wide range in any mode.

APPENDIX C.
Software

C.1. Raspberry Pi Program

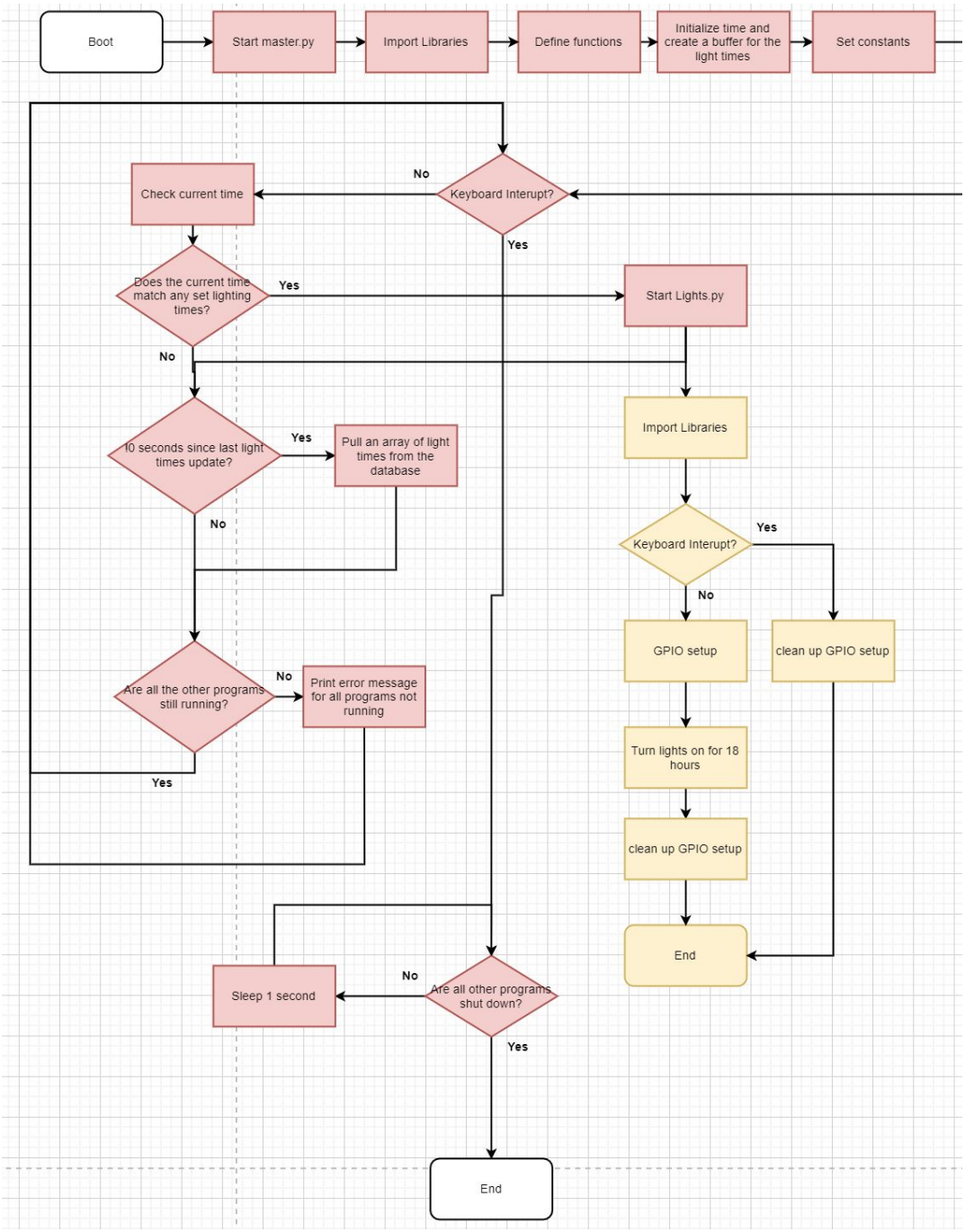


Figure C1. master.py and Lights.py Adapted from [31]

The flow charts above show how each program starts, what they do and how they are connected. The first shows the master program and the lighting program. The master program starts at boot and continues off to the right. In between then and when it returns for the start of the loop the four other programs are started.

This shows most of the process of the master program. It's main function is to check on the other programs. The keyboard interrupt block is used to show that a loop starts that can be exited by using the keyboard interrupt, CTRL + C on the pi. While it's not pressed the program goes into its' continuous loop. It first gets the current time, then it checks if that time is one of the lighting times, the first loop the values are empty so nothing is done. If it is one of those times the light is turned on. Continuing with the master program the next command is checking if the lighting times have been updated in the last 10 seconds, if they haven't an array of user set lighting times is pulled from the database. Lastly the program checks if the other programs are still running with the poll() command explained above. If any of the programs aren't running, aside from the light program, an error is displayed saying which program has stopped.

This flowchart shows the master program commands in red and color codes the rest of the programs uniquely. The yellow program path is the Light.py program, it simply runs the lights for several hours. It is the perfect example for why the subprocess library should be used. Since it's a seperate program it can sleep for the duration of the lighting, again several hours, and not interfere with the master program or any other for that matter. The Light.py program is the only one to run for a certain amount of time before stopping as the others all run continuously until stopped. Below I go into the other standalone programs.

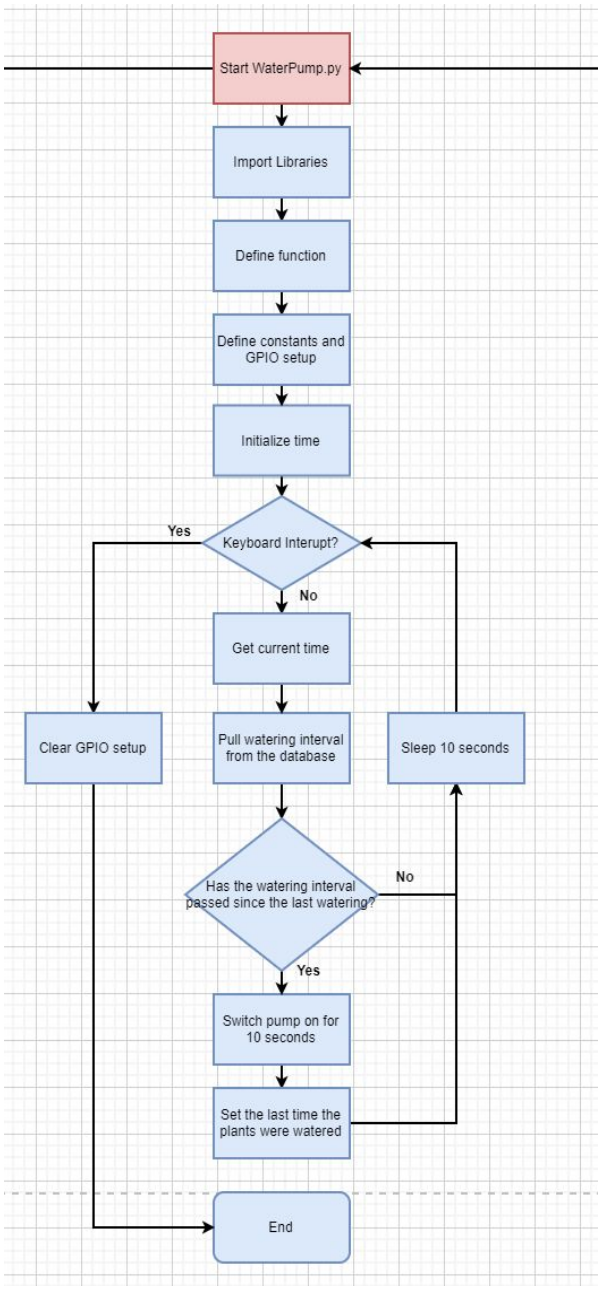


Figure C2. Water_Pump.py Adapted from [31]

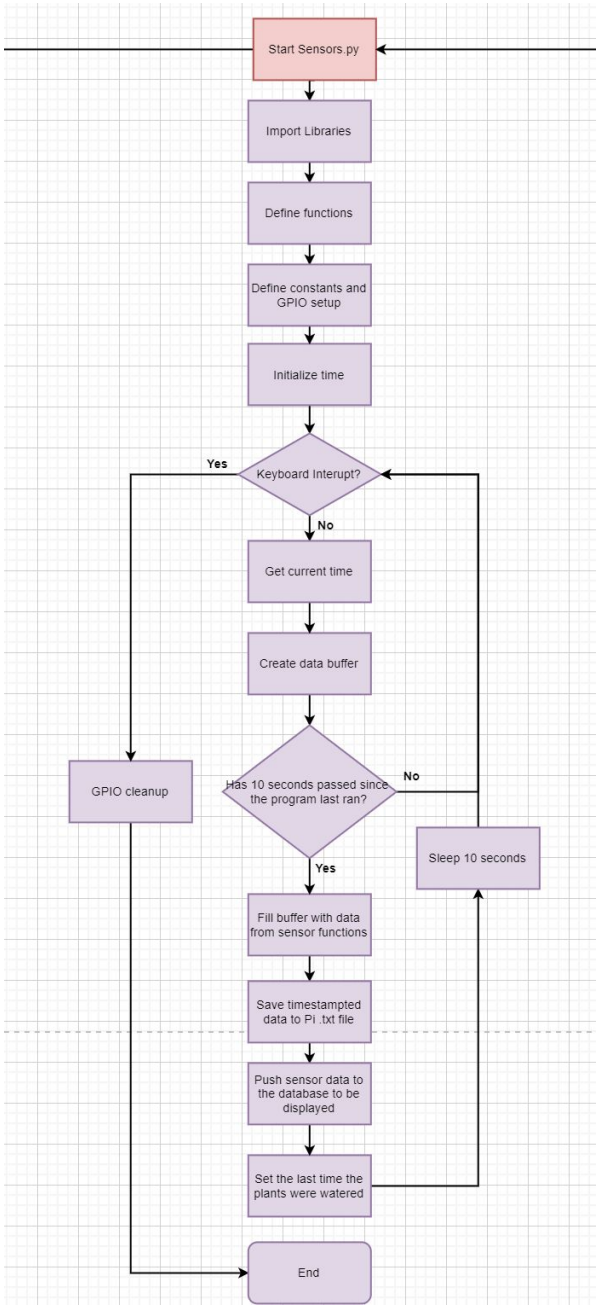


Figure C3. Sensors.py Adapted from [31]

The "Water_Pump.py" program controls how often the plants are sprayed with the nutrient solution. The program runs continuously until the keyboard interrupt is pressed. This program uses the requests library to pull the user set watering information from the application. The watering interval is set to revert back to the default 5 minutes if a time outside the range of 3 minutes to 7 minutes is entered. The two constants of the program set the default delay time just described and the other to set the duration of the watering, set to 10 seconds in the code. The program cycles in a loop, checking every 10 seconds if the delay time has passed since the last

watering time. When the keyboard interrupt is pressed the GPIO's are unset and the program ends.

The "Sensors.py" program controls the acquisition, saving, and sending of the sensor data. It notably uses the json and csv python libraries; the json library is used to format and send data to the database whereas the csv library is used to save chunks of past data to .txt files. The other notable libraries include those used for the sensor data acquisition; we had to get data from both SPI and I2C as well as a specialized digital interface for the temperature and humidity sensors. Each sensor's acquisition code has its own defined function for ease of use, the other two functions serve to save and send the data respectively as described above. The program runs on a simple timing loop like the others which, when triggered every 10 seconds, sticks the data from all of the sensors into a buffer after which the save and send functions get the data where it needs to go. A separate file is also created for the electrical conductivity value, a file containing the most recent EC value is created and overwritten every 10 seconds for use in regulating the amount of nutrients in the water.

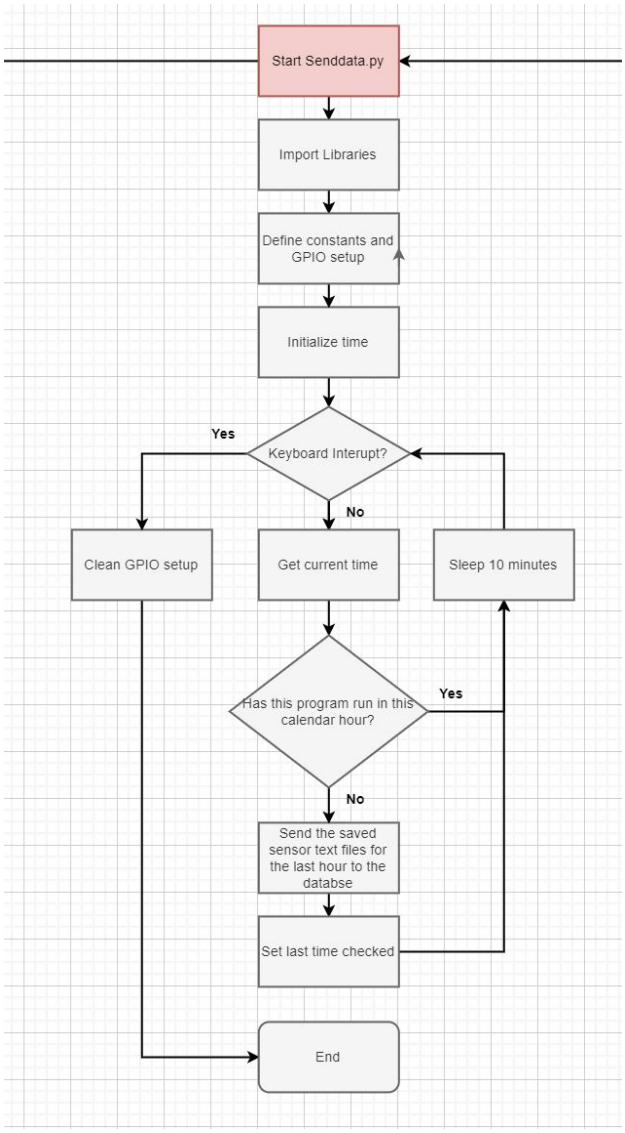


Figure C4. Send_Data.py Adapted from [31]

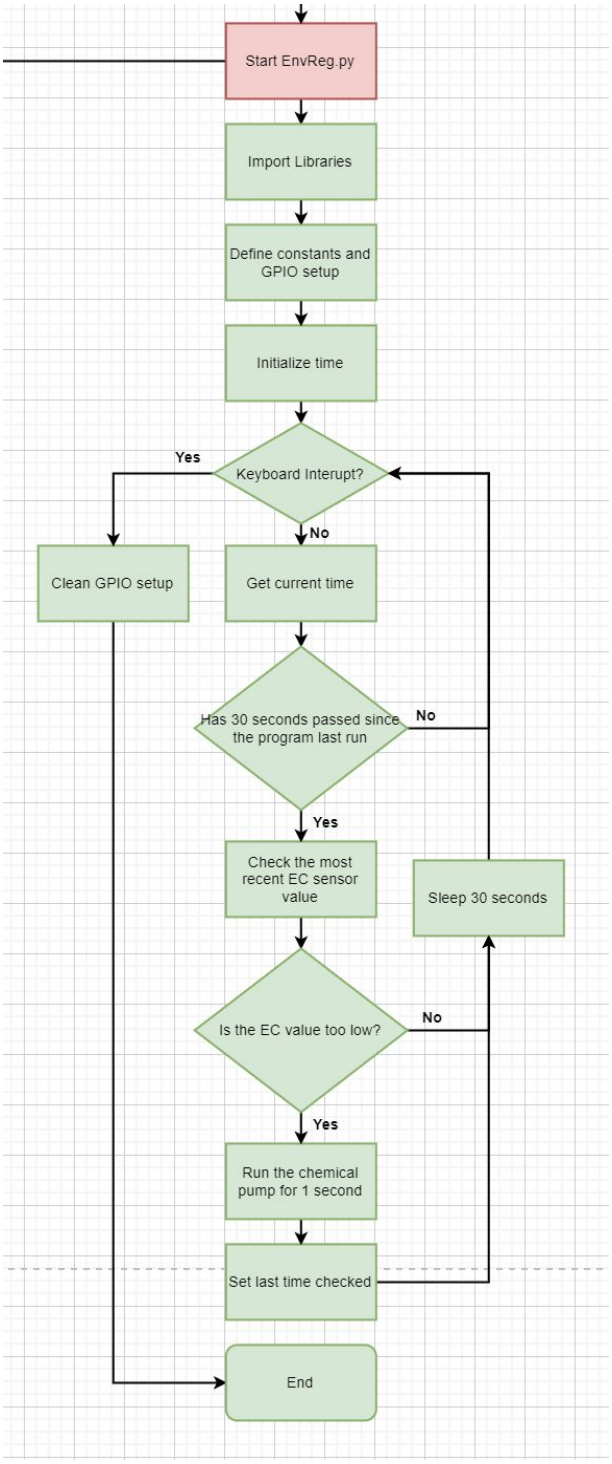


Figure C5. Env_Reg.py Adapted from [31]

The "Send_Data.py" program is a functionality that went unused for the most part with the lingering benefit of the sensor data being stored on the database. The program is very simple and was intended to allow for the plotting of past sensor data. The program pulls the time-stamped sensor data accumulated over the last hour and sends it as a json file to the

database. The program is asleep for most of the time, only checking if an hour has passed every 10 minutes and staying asleep to save the CPU the rest of the time.

The “Env_Reg.py” program is responsible for regulating the amount of nutrients in the water. It does this with information from our electrical conductivity PPM sensor, by controlling the floor level of electrical conductivity we are able to control the amount of nutrient solution in the water basin. The only relevant library this program uses is the csv library which is used to read the most recent EC value. There are no functions as the code is very simple. The loop gets the current time and checks if 30 seconds has passed since the last time the program got the EC value. If it’s been 30 seconds the program pulls the EC value from the most recent running of the “Sensor.py” program. If the value is lower than the set threshold then the peristaltic chemical pump is run for just a second. The pump time is so short compared to the wait time of 30 seconds because the nutrient solution is incredibly potent, it needs time to dissolve into the water.

Since each program runs independently of the others testing could be done by running the master program, assuming that works, and the programs that were incorrectly working would display errors specific enough to point them out. The main discovery of testing that prompted the addition of some code to every program is I was getting errors that the GPIO’s were already in use because I had exited the programs without doing any GPIO cleanup. The other fact discovered through testing was the effect of calling for a program to sleep. Running 6 programs at once was pushing the limits of the Raspberry Pi’s CPU and using the sleep command reduced the CPU percentage in use to below 50%.

The program “master.py” starts and monitors the others to make sure everything is running smoothly and to give specific errors as to which program isn’t working. This program is set to start on the boot of the Pi. From there several libraries are called, most important of which is the subprocess library. The library allows for programs to make command line executions such as starting another python program [30]. The function Popen() is what allows for the action.

```
#Runs Programs in "pi" folder
def runprog(file):
    p = subprocess.Popen('python '+ file ,shell=True)
    return p
```

Figure C6. master.py - Popen() Adapted from [32]

Assigning the function to a variable allows for the programs status to be monitored. Below I show a command used in the program to start one of the independent programs. I then show the command used to monitor those programs as an example.

```
w = runprog('Water_Pump.py')# Runs water pump on an interval
```

Figure C7. master.py - runprog() example Adapted from [32]

```
if (w.poll() != None):
    print('Water Pump error')
```

Figure C8. Master.py - poll() example Adapted from [32]

As seen in the function all that is needed as input for runprog() is the python file name as a string with the condition that the program is in the "pi" folder. The second command, found within the loop of the master program, checks the status of the water pump and outputs an error if it sees the program has stopped. It uses the variable of the runprog() command with the addition of the .poll() function. If poll() returns None the program is still running and returns 0 if the program has stopped (or can't be found) [33].

Another important library found in several programs is the request library, it allows for communication with the firebase database our android application uses. The library is used to get information both to and from the database. The following figure shows an example of the code in this case used to pull an array of user set lighting times.

```
req = requests.get(firebase_url + '/users/'+ user +'/sensors/LightTimes.json')
data = req.json()
```

Figure C9. master.py - requests Adapted from [32]

From that point the "data" variable can be treated as a dictionary in json format. A similar function is used to push the sensor data to the database for it to be displayed on the application.

```
        result = requests.put(firebase_url + '/users/3xL6MRsUvrN7I2H134ejPGglGgr2/sensors/' +
stp[j] + '/' + '.json', data=json.dumps(data[j]))
        print('Record inserted. Result Code = ' + str(result.status_code) + ',' + result.text)
```

Figure C10. requests - put Adapted from [32]

This command contains more syntax but is just pushing numbers to the database in json format rather than pulling from it. It is used for each sensor, string of names contained in the "stp" array, to push each value into its own named category on the database. With the correctly set user and firebase_url the correct categories will show up, named, on the database without the user having to set anything up on the website end.

The last important library is the datetime library which is used to control the timing of the device. It uses the RaspberryPi's internal clock which is updated through the internet. Other important libraries include the csv and json libraries. The csv library is used to create easy to read data files on the pi while the json library is used to send and receive data from the database/app. There are also the Adafruit libraries for the MCP3008, the DHT22, and the TSL 2591 used for the sensor functions. Other common libraries are used to a lesser degree in all programs.

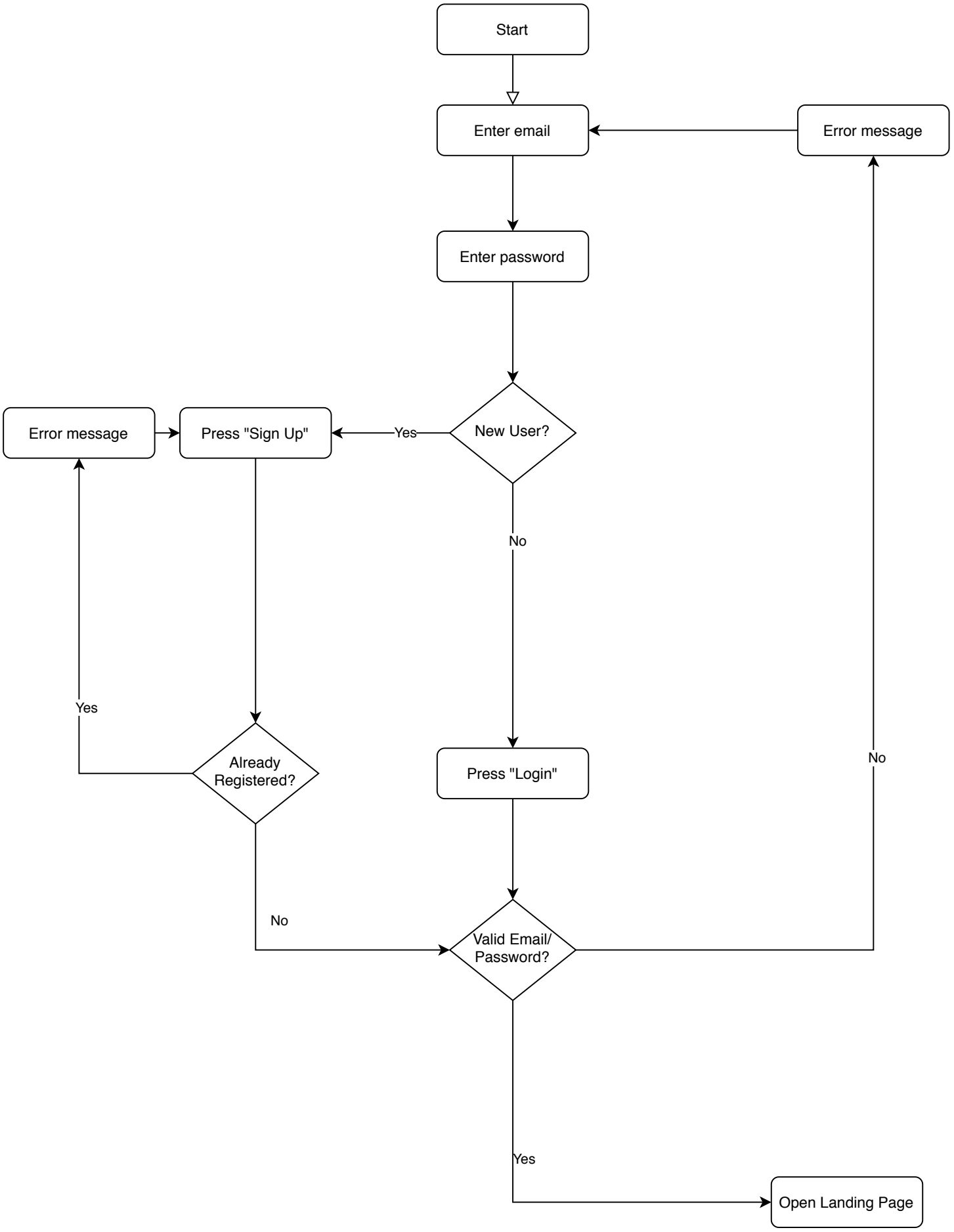


Figure C11: Login Logic Adapted from [41]

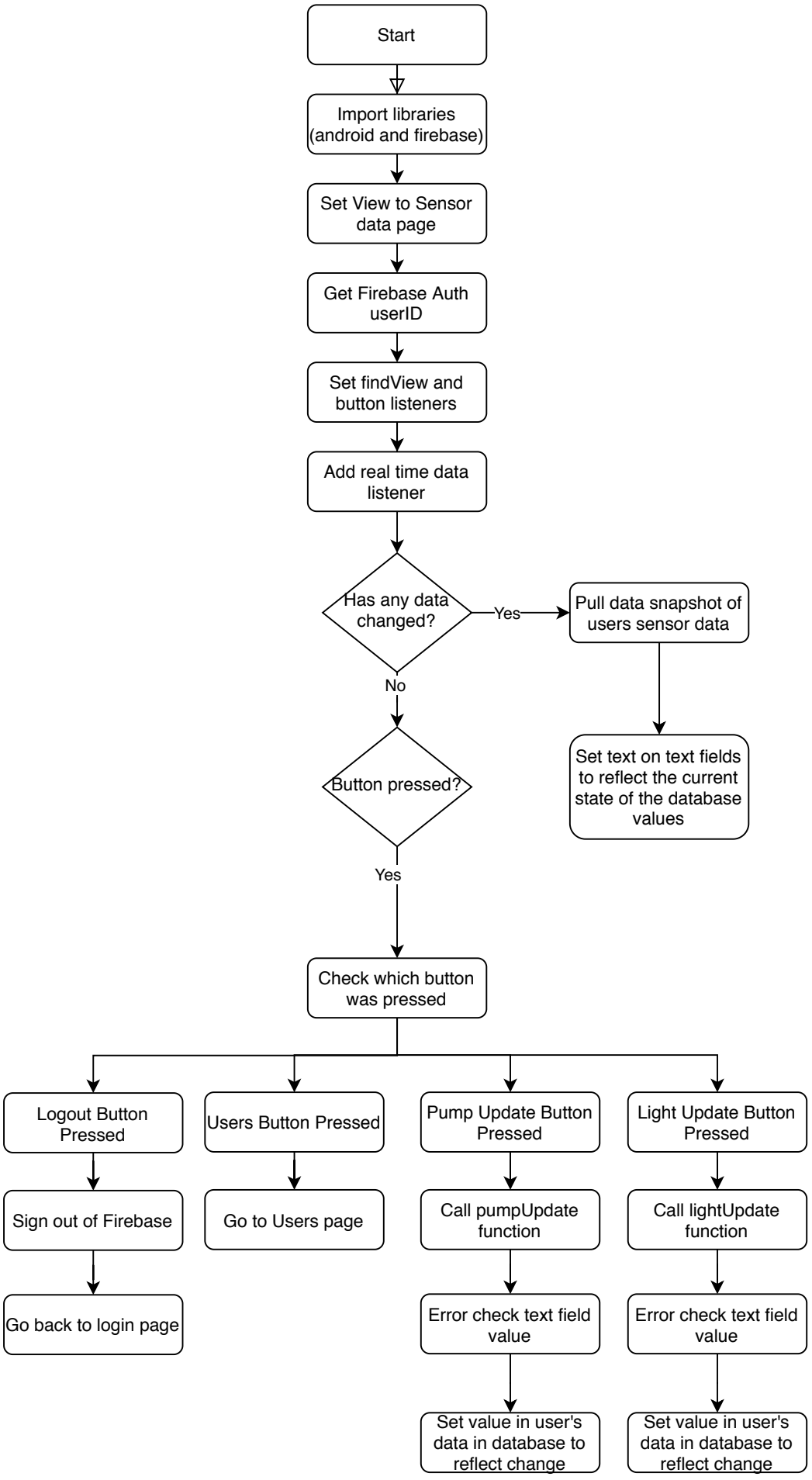


Figure C12: Login Psuedocode Adapted from [42]

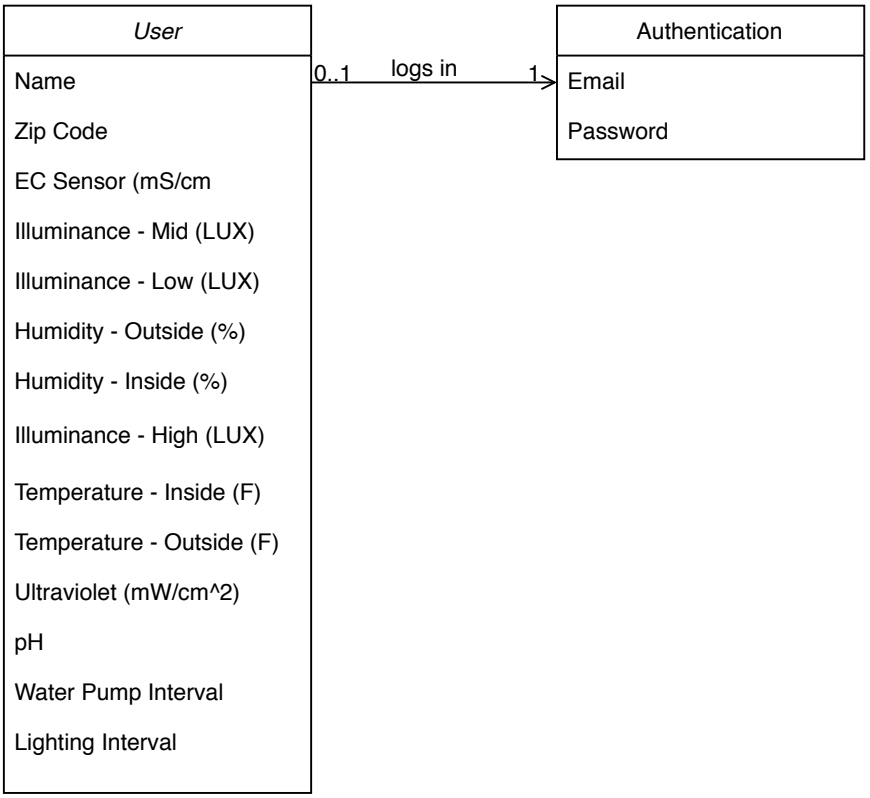


Figure C13: User Profile Adapted from [43]

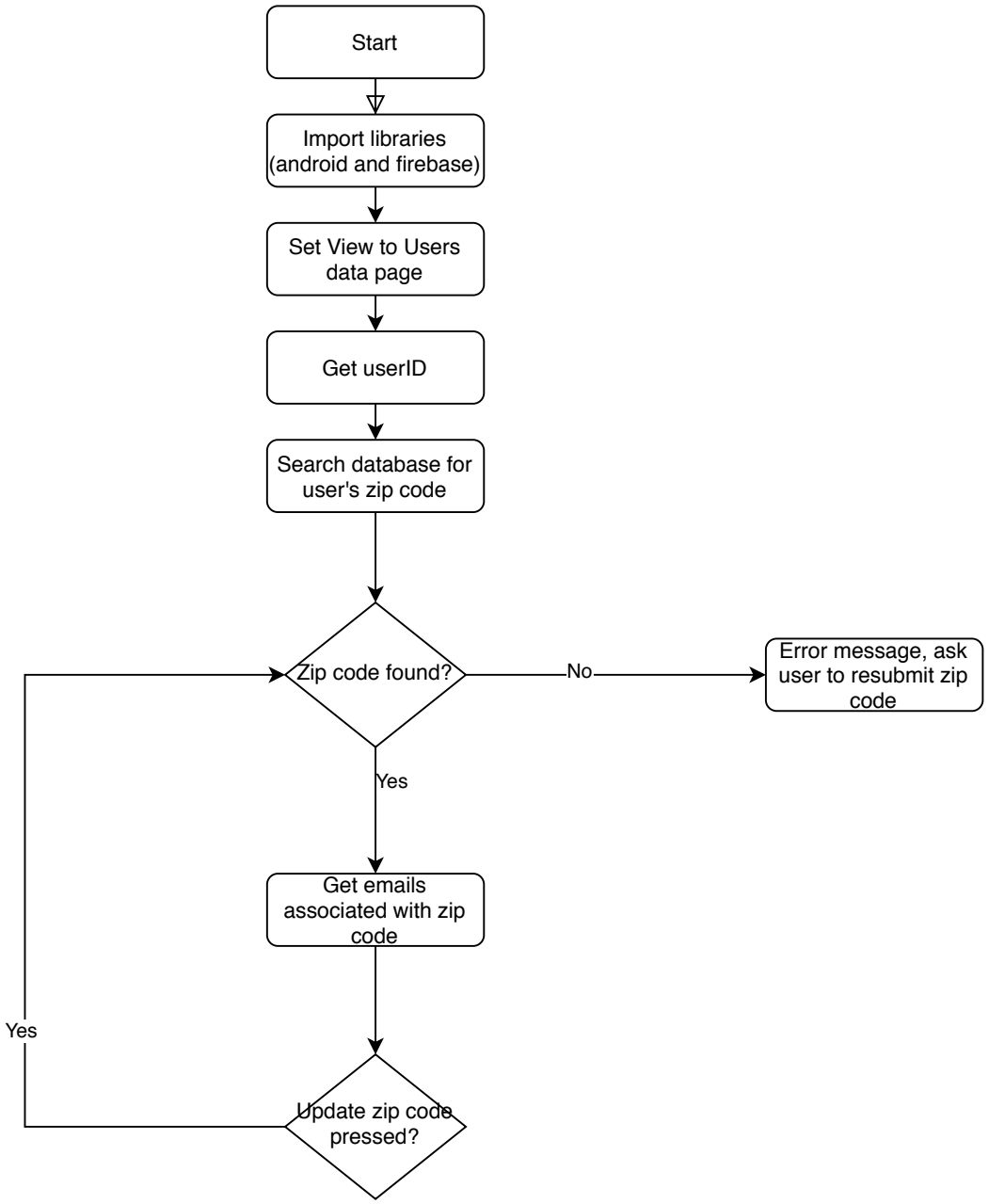


Figure C14: Community Contact Logic Adapted from [44]

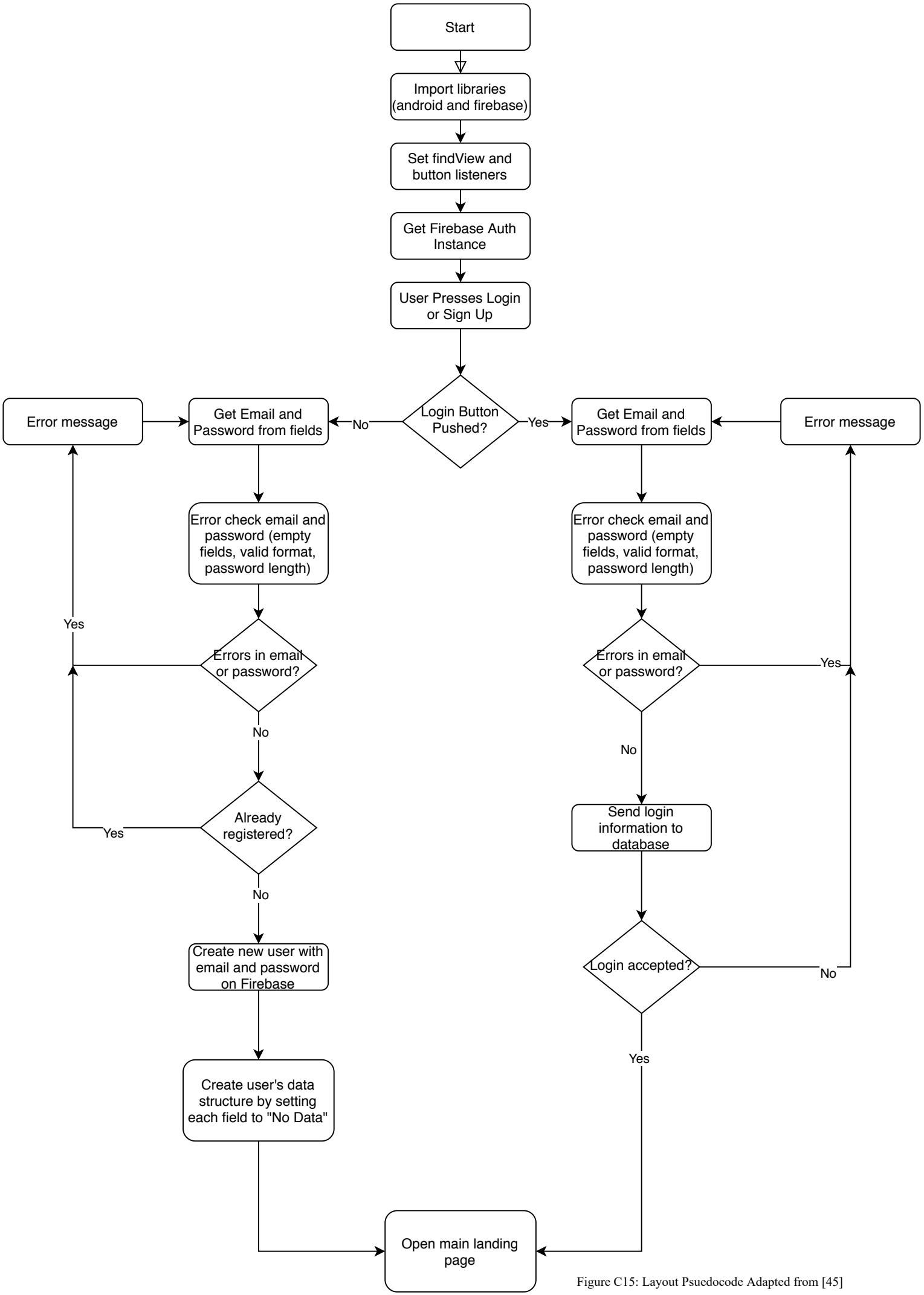


Figure C15: Layout Psuedocode Adapted from [45]

APPENDIX D.
Mechanical Aspects

I . Building Material

The first prototype was made from polystyrene. It is a synthetic aromatic hydrocarbon polymer. People use it everyday in food containers and utensils. It comes in many forms but the form we used was a solid that is ¼ inch thick. This was chosen for the first prototype because it is food safe and considered a cheap material. It is also very easy to cut and work with. The problems with it are that it is not stiff or particularly strong. This makes it almost impossible to create a structure that is perfectly straight. Another problem is that it starts to break down in UV light, something that could definitely happen in our case considering we are using a grow light. It also does not seal in water vapor well.

The solution to all of this was to switch to corian. It is an acrylic polymer and it's much denser than polystyrene. It is often used outside for countertops and similar applications so it is very UV resistant. The most common usage for it is kitchen countertops and cutting boards. It is used because it is food safe and very stable. It comes in very precise sheets so building a structure with it is straightforward.

The other primary building material is 6061 aluminum. It can be bought in any shape and is fairly inexpensive. For our use we got it at an extruded angle. It is simply an L shaped bar. On its own it is a fairly resilient material, but to make it even more resistant to corrosion it was powder coated. Powder coating is the process of using static to make small beads of plastic cling to an object, and then baking that object until the plastic melts and forms a hard layer.

II. Building Process

To build the structure all the pieces were cut from a large sheet of corian. Then, with color matching glue specifically made for corian the pieces were glued together. As seen in Figure D1.

Figure D1. Construction Adapted from [34]

In figure D3 blue tape is used to provide compression. In general adhesives create a much stronger bond when the pieces are pressed tightly. This was also done so that everything would stay perfectly aligned while the glue cured.

Figure D2. Tape compression. Adapted from [35]

Once the main structure was glued together, everything needed to be sanded to give the

Figure D3 shows the system during its sanding phase.

Figure D3. Sanding Adapted from [36]

Once the sanding was completed the aluminum bracing had to be cut, debured, and then powder coated. Figure D4 shows the aluminum being cut in a band saw.

Figure D4. Brace cutting. Adapted from [37]

Once cut and debured, the aluminum bracing had to be powder coated. That process is shown in figure D5.

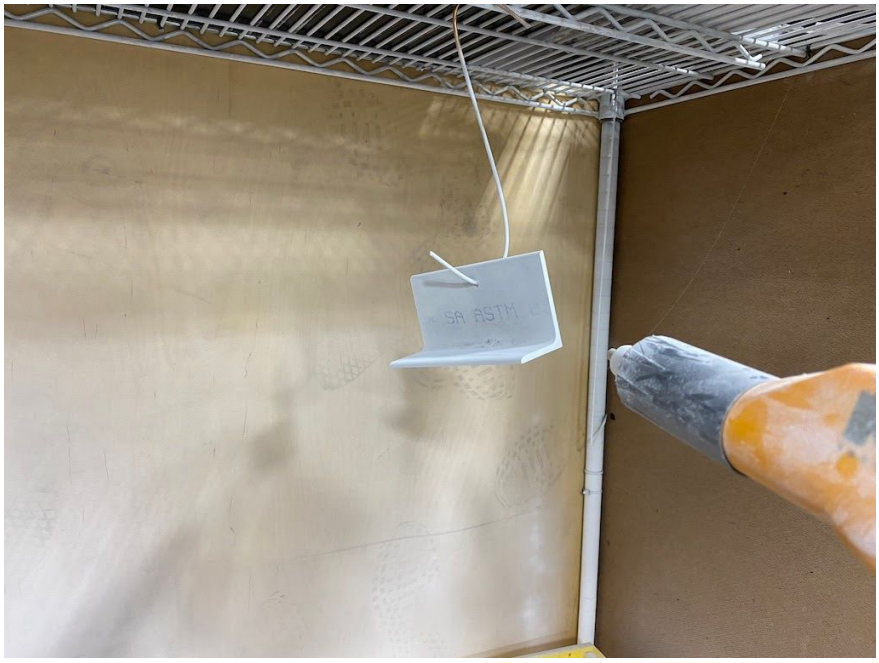


Figure D5. Powder Coating. Adapted from [38]

Once all these steps were completed the main structure was complete. Figure D6 shows the dimensions of the finished system.

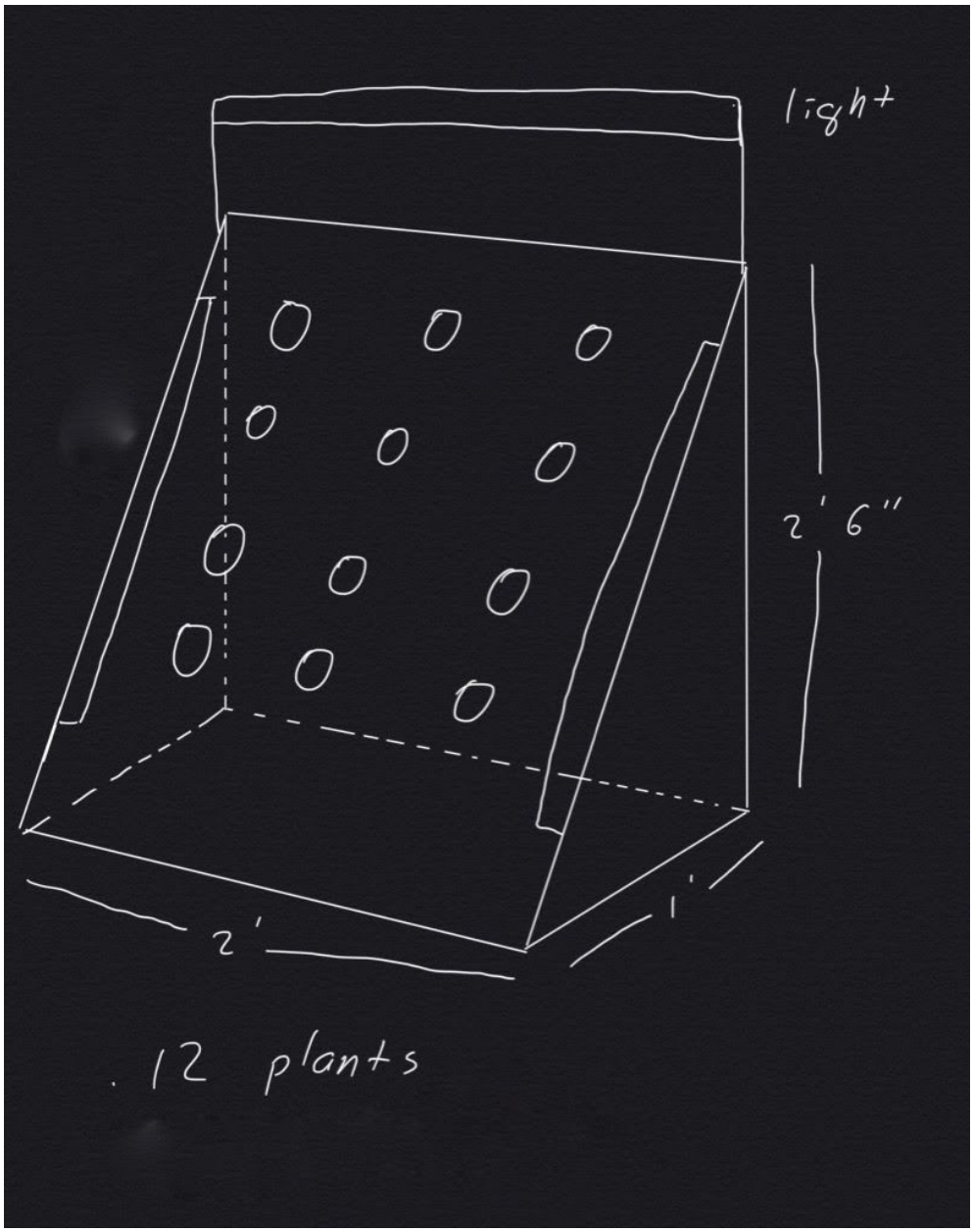


Figure D6. Dimensions Adapted from [39]

III. Integration

The final part of the physical structure was integrating all the parts. To accomplish this holes had to be cut anywhere a sensor was placed. This was done so that the wires could run directly into the system. Using aluminum bracing a mount was made for the light. The light is over 10 pounds so we chose not to hang it. Instead it simply sat on top of an aluminum shelf.

APPENDIX E.
Vendor Contacts

Ansync Labs
5090 Robert J Mathews Pkwy, Eldorado Hills, CA 95762
916-933-2850

Kevin Crowe

EDUCATION

California State University; Sacramento, CA
Bachelor of Science, Electrical and Electronic Engineering, May 2020

Los Rios Community Colleges; Sacramento, CA
Dec 2017

EXPERIENCE

Selland's Market-Cafe; Sacramento, CA — March 2017 - Now
Dishwasher

- Work as a dishwasher and doing food preparation

Opa Opa; Sacramento, CA — March 2017 – Jan 2018
Busser

- Worked as a busser, food runner, and dishwasher

SES Tutoring; Sacramento, CA — Jan 2016 - Dec 2016
Tutor

- Tutored in K-12 English and Math

PROJECTS

Senior Product Design

- Designed and built an automated aeroponic system for indoor farming which received an A grade

Yutthachat Thao

EDUCATION

California State University, Sacramento
Bachelor of Science, Electrical and Electronic Engineering, May 2020

EXPERIENCE

KFC Crew Member, Elk grove, CA — Nov 2015 - Sep. 2017
Project Manager

- Responsible for greeting customers and taking orders.
- Responsible for package them and delivering them

Macy Receiver, Sacramento, CA — May 2018 - Oct 2018
Engineering Intern/Assembly Tech

- Help unload new stocks and prepare them for inventory.
- Loaded old shipment of unneeded clothes to RBA.

PROJECTS

Senior Product Design

- Designed and built an aeroponic system for indoor farming which received an A grade

LANGUAGES & INTERESTS

- **Code development,**
  - Love to learn and automate many functions of life through programming

Adrian Barrera

EDUCATION

California State University, Sacramento
Bachelor of Science, Computer Engineering, May 2020

EXPERIENCE

California Department of Water Resources July 2017 – Jan 2020
Electrical Engineering Intern

- Assist electrical engineers with ongoing projects
- Review functional trip test reports of protective electrical systems for motors, generators, transformers, etc. to ensure WECC compliance
- Analyze electrical single-line, three-line, connection, dc schematic, and switching diagrams for compliance purposes

California Department of Community Services and Development July 2014 - July 2017
Student Assistant

- Act as administrative assistant for the Human Resources Office (HRO)
- Troubleshoot IT issues for the HRO

California Department of Public Health December 2013 - April 2014
Volunteer

- Conduct black box testing of the California Reportable Disease Information Exchange (CalREDIE) system, with supporting documentation
- Provide effective customer support to users of the CalREDIE system
- Effectively control the release of confidential information to health care providers

Brandon Buck

EDUCATION

California State University, Sacramento
Bachelor of Science, Electrical and Electronic Engineering, May 2020

Folsom Lake College
Associate of Science, Mathematics, May 2016

EXPERIENCE

Ansync Labs, Eldorado Hills, CA — June 2019 - Present
Project Manager

- Organize and facilitate relationships with vendors
- Schedule and create timeline for projects including material acquisition and department assignments
- Facilitate communication between teams and clients

Ansync Labs, Eldorado Hills, CA — June 2018 - June 2019
Engineering Intern/Assembly Tech

- Consulted on projects especially relating to conductivity and signal integrity
- Constructed prototypes

J A Frasca & Associates, Sacramento, CA — February 2017 - June 2018
Consultant

- Contracted for the California Public Retirement System
- Coded programs for data auditing to ensure data quality
- Created a search engine to locate records impacted by production change requests

PROJECTS

Senior Product Design

- Designed and built an aeroponic system for indoor farming which received an A grade

LANGUAGES & INTERESTS

Fluent in Afrikaans

Cycling Development — November 2014 - September 2016

- High school mountain bike program coach

APPENDIX G.
Using The System

This section is a documentation of our first use of the system. We decided to grow 3 different plants: spearmints, lemon mints, and basil. We chose these plants because we wanted a variety to test how far our system can keep the plants alive and growing. In addition, the spearmint and lemon mints are much easier to maintain and are more resilient to environmental changes. They required about 2-3 hours of full direct sunlight while the basil requires 6 hours of direct sunlight. These requirements can be met by our led growth light.

The first step was the transfer. We wanted to have a few small growing plants already, so we can test our system better. We transfer the plants one by one from their pots to the holder in our systems. This transfer may have caused some stress in the plants. Although not fully realized, the plants may be struggling. In addition, the plants did not receive any water or nutrition after 3 hours after the transplant because we were working far away from the plant we wanted to test the system. However after we transplanted, the plants were still looking good.

Figure G1. The plants after the transplants Adapted from [40]

We transfer the plants at one of our group member's workplace.

Figure G2. After Setting the System At the Designated place Adapted from [40]

After we decided who would take care of the plants, we set up the system and had it running. This system is going to be run by one of our group members, and he will treat it like a user would. This means that he will not be taking care of the system everyday. He will run the programs and tried to see if our system can actually grow some plants.

Figure G3. Day 4 after the Transplant Adapted from [40]

After 3 days, we noticed that the plants were doing well, and our system is working as intended. It was watering the plants, and the plant actually a bit. The system so far has not had any issue.

Figure G4. Day 5 Adapted from [40]

We decided that when we set up the system, we should prepare for any mishaps. Therefore, we have a towel underneath to make sure if there is a small leak, the carpet will be safe for a while.

Figure G5. Day 6 Adapted from [40]

Day 6: The plants are growing and still doing fine. Again there is no issue so far with the system. The lights were still on and the water was still pumping.

Figure G6. Day 9 Adapted from [40]

Day 9: After a week, we were sure that the system and the plants were going to be fine and left it with much attention just as the user would have. However, our system did have a leak in the back. Most of the water was leaking out on the back. This left the plants with a few days of no water and nutrition. We did not notice this leak until it had already left the plants stressed.

Figure G7. Day 10 Adapted from [40]

Day 10: We decided to put a plastic sheet underneath and the towel just in case there was another leak. We sealed up the previous leak, and now the water plants are still doing ok. Therefore, just like before, we intend to leave the plants growing without much attention from the user. This was an idea because our system did not have a way to act if the wifi was intermittent. This was the case after 10 days. The wifi at our designated testing area was working intermittently, and we did not notice it immediately and left the system running for 7 days without access to instruction on when to run the lights and the interval of the watering.

Figure G8. Day 18 Adapted from [40]

Day 18: After another 8 days, we noticed that the plants were dying and struggling to survive. We checked the system, and found that the program was all out when the wifi disconnected. This left the system not working for a whole 1 week. This caused plants to stress and mostly the basil all died. The basil required much more sunlight to survive and there was barely any from the window near and no light from our grow lights. We removed the basil and decided to keep testing the system after fixing the issue. Again, we left the system with little attention from its users and tried to see what would happen.

Figure G9. Day 25 Adapted from [40]

Day 25: After a week, we did a checkup on the plants and noticed that they were still doing okay. They may have grown a bit different, but it looks like they were still growing even if just by a little. We checked the system for any error and have found nothing.

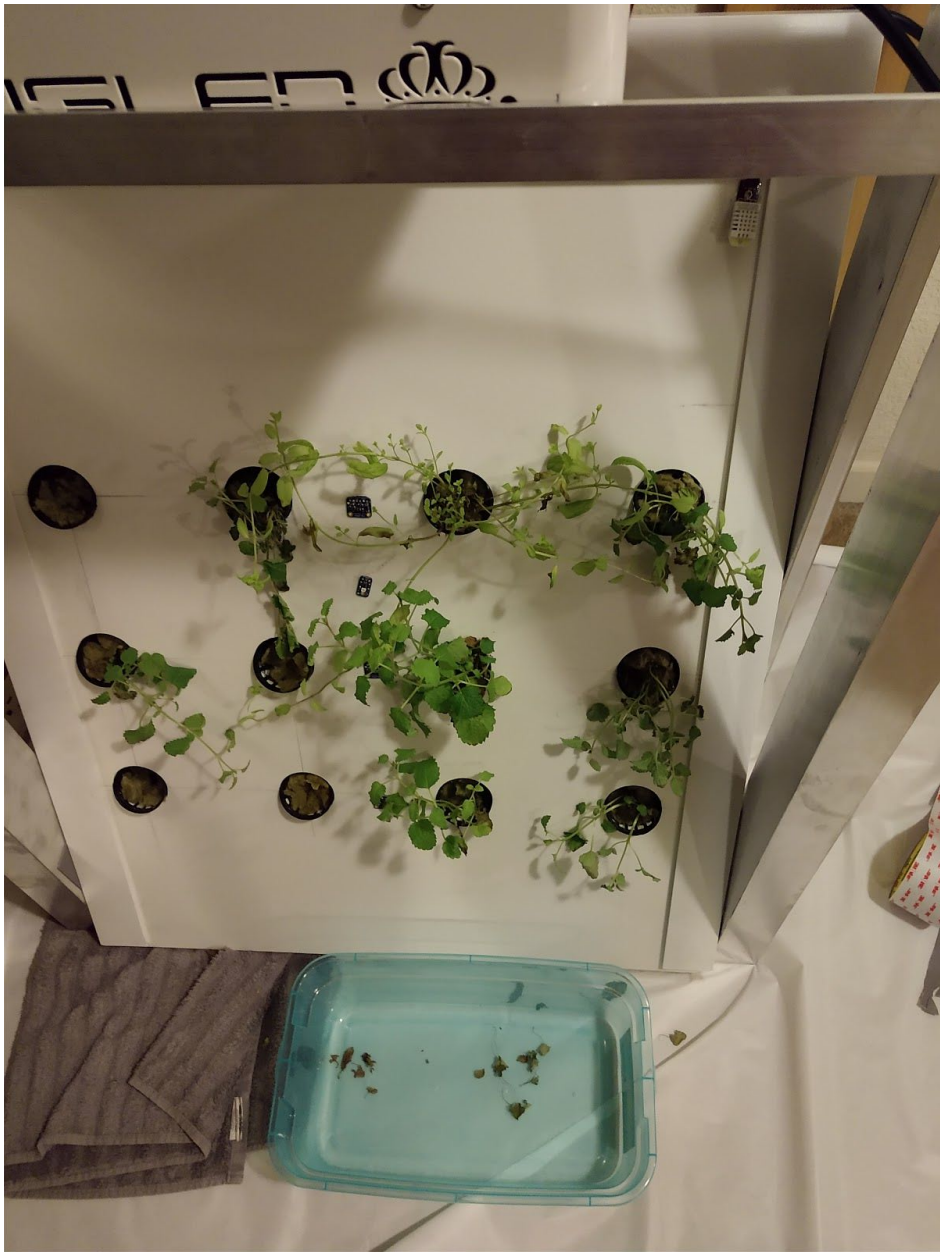


Figure G10: Day 27 Adapted from [40]

Day 27: We noticed that the plants were slowly dying, and again there was another leak in our system. Although the plants were getting water, they were only getting a little now. This may be the reason why our plants are struggling to grow. Even though we fixed this error, it seems that the plants were going to die. Slowly we watched the plants die one by one.

Figure G11: Final Day Adapted from [40]

This is the final day. It is the day in which we are recording the video for the senior project. We knew we did not have time to retry our test. However with our first experiment, we learn that the plants are able to be kept alive for a while without any care. Stills, our design was not perfect and there were some leaks and mishaps. This ultimately caused our plants to slowly die. In conclusion, we still needed to test the actually growing process even further before actually marketing this product. There are still a lot of minor details that needed to be corrected before we can actually call this a real product.