Fall 2016

Hydrobase: An IoT Gardening Application

Nandini Kant Goyal
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Other Computer Sciences Commons](#), and the [Systems Architecture Commons](#)

Recommended Citation

Goyal, Nandini Kant, "Hydrobase: An IoT Gardening Application" (2016). *Master's Projects*. 503.
DOI: <https://doi.org/10.31979/etd.kj96-myau>
https://scholarworks.sjsu.edu/etd_projects/503

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

SAN JOSÉ STATE UNIVERSITY

Hydrobase: An IoT Gardening Application

A Project Report

Presented to

Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Nandini Kant Goyal

December 2016

HYDROBASE

The Designated Project Committee Approves the Project Titled

Hydrobase: An IoT Gardening Application

By

Nandini Kant Goyal

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2016

Dr. Robert Chun	Department of Computer Science, SJSU
Dr. Katrina Potika	Department of Computer Science, SJSU
Ms. Sindhuja Jeyabal	School of Information, UC Berkeley

ABSTRACT

As the world is moving towards urban agriculture, there are many IoT Hydroponic Control Systems being introduced these days. There are many different platforms emerging, but they all suffer from the same flaw, the software and hardware are so tightly integrated that most of the times the individuals have no freedom of how to use the product, or they need to completely hack the system. This brings us to the need of a system that is designed to allow the user more flexibility in platform configuration, as well as assist in platform construction. We propose a model called Hydrobase to solve this problem. Urban Agriculture often uses hydroponic systems because of space constraints and environmental constraints. Subsequently, this project would assist efforts already underway. This system is specifically targeted at Small Scale DIY to Medium Scale Urban Agriculture customers who are interested in automating the gardening/farming experience. This system would also assist in agricultural experimentation and knowledge sharing. There are a number of different factors that contribute to a plant's success and exploring what is most beneficial has been the subject of much study. This system will help by allowing customers to contribute in a citizen scientist fashion, automating much of the experiment design and data collection. By automating the more monotonous aspects of experiment design and execution the system will empower more individuals to contribute to science. This report explains what work has been done in this domain and how Hydrobase is a better user friendly platform as compared to its IoT Hydroponic Control System counterparts.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor, Dr. Robert Chun, for his encouragement, patience, expertise and continuous guidance throughout my project and my graduate studies.

I am very grateful to the committee member, Professor Katrina Potika for being extremely helpful by giving her suggestions and for monitoring the progress of the project.

I would also like to thank Ms. Sindhuja Jeyabal for her valuable time and support through out the project.

I would sincerely like to appreciate the graduate students of iSchool at University of California Berkeley for coming up with this brilliant idea of Hydrobase. I thank them for all work and enthusiasm they put in to make this project happen.

I also appreciate the interviewees, survey respondents, and usability testers for sharing their time and valuable insights. I am grateful to my husband, who showed his support and exhibited patience during my late night and weekend hacks. Finally, a big thank you to the Computer Science Department at San Jose State University for giving me this opportunity. I couldn't have done it without you.

TABLE OF CONTENTS

1. Introduction	7
1.1. <i>Problem Statement</i>	7
1.2. <i>Solution</i>	7
1.3. <i>Scope</i>	8
1.4. <i>Target Users</i>	8
2. Background	9
3. Motivation	12
3.1. <i>Research Questions</i>	12
3.2. <i>Methodology</i>	14
3.2.1. Survey	14
3.2.2. Interview	16
3.2.3. Usability Studies	17
3.3. <i>Key Findings</i>	17
3.3.1. User Needs	18
4. Related Work	19
4.1. <i>MIT Open Agriculture Initiative</i>	20
4.1.1. Food Computer	21
4.1.2. Climate Recipe	23
4.2. <i>Grove Ecosystem</i>	25
4.3. <i>Hydrobase</i>	27
5. System Design	27
5.1. <i>Overview</i>	27
5.2. <i>System Architecture</i>	28
5.3. <i>Information Architecture</i>	30
5.3.1. Users	30

5.3.2. Devices	30
5.3.3. Grows	31
5.3.4. Plant Profiles	34
5.3.5. Data	35
5.3.6. CV Data	36
5.3.7. Pubnub Communication	36
5.4. <i>Physical Setup</i>	38
5.5. <i>User Interface</i>	40
5.6. <i>Information Visualization</i>	42
5.7. <i>Measuring Success</i>	42
6. Hydrobase Application Design	43
6.1. <i>Dashboard</i>	43
6.2. <i>Devices</i>	44
6.3. <i>Plant Profiles</i>	46
6.4. <i>Grows</i>	47
6.5. <i>Notifications</i>	49
6.6. <i>Emergency Stop</i>	49
6.7. <i>Data Export</i>	50
6.8. <i>Documentation</i>	50
6.9. <i>Signup/Login Page</i>	50
7. System Security	50
8. Conclusion	51
9. Future Work	52
9.1. <i>Permutation Testing</i>	52
9.2. <i>Crowdsourcing Grow Profiles</i>	53
9.3. <i>Modular Physical Kit Design</i>	53

10. References	54
11. Appendix	57
A. User Research	57
B. Code Implemented	64

LIST OF FIGURES

1. Simple Hydroponic system architecture	11
2. Geographical locations of survey respondents	15
3. MIT City Farm	21
4. Food Computer	22
5. Prototype of the Grove Ecosystem	26
6. Network Diagram	28
7. Electronic hook up diagram	29
8. Early concept drawings of the structure	38
9. Hydrobase finished structure	39
10. Annotated structure and electronic setup	40
11. Low fidelity mockup	41
12. Sensor visualization dashboard	44
13. Device management page	45
14. Plant profiles page	46
15. Hydrobase application being used to grow	48
16. Edit device and emergency stop page	49

1. INTRODUCTION

1.1 Problem statement

Increasingly there are a number of different systems and technologies being developed to support urban agriculture, ranging from Grove Lab's Ecosystem to the MIT OpenAg initiative, each attempting to support effective food production. Yet the majority of these systems suffer from the same central problems, that is they require tight integration between the hardware and software. While this is certainly understandable due to the complicated nature of hardware and software interactions, it creates substantial barriers for entry. In the case of Grove Labs' ecosystem, the vertical integration allows for a relatively easy and intuitive user experience, yet this comes at the cost of preventing users the ability to make their own choices about the plants they are growing and what is done with the data. On the other end of the spectrum, the MIT OpenAg initiative allows users to customize their system in whatever way they see fit, yet requires substantial knowledge of electrical engineering, system design and fabrication techniques. Subsequently, in this project the focus was on developing a system that would allow users the ability to customize their own set up, without requiring them to have an advanced degree. The lower the bar for entry into urban agriculture the more we can effectively support exploration and greater experimentation.

1.2 Solution

The solution was to create an open source cloud-based hardware agnostic control and data collection platform for hydroponic growing systems. We needed to build a hardware system that worked with the cloud-platform in order to both demonstrate the capabilities of the system as well as serve as a potential blueprint from which other users can build upon.

1.3 Scope

The scope of the project was to support situations where accurate plant data collection and environmental maintenance is critical, such as research, commercial or experimentation applications. This is not to say that these are the only situations under which the system can operate, but through the user research performed as part of this project, these were found to be the most applicable use cases for such a system.

1.4 Target Users

After doing some user research at the beginning of this project, we were able to identify that the target users for this system essentially fall into two different categories but both share a common characteristic of being heavily invested in growing plants.

Technically inclined, botanically disinclined: This individual is comfortable with information systems but has little experience working with plants and does not fully understand what they need or how to effectively support them. This user is predominantly focused on exploration and experimentation as they are comfortable with data extraction and analysis but need an effective entry point for hydroponic growing.

Botanically inclined, technically disinclined: This individual understands what plants need and is interested in accurately recording data about the plants but does not have the technical expertise to set up an automated system on their own. This user is predominantly focused on leveraging technology to support their existing practices in either research or commercial applications.

2. BACKGROUND

The aim of this project was to build an open source cloud-based hardware agnostic control and data collection platform for hydroponic growing systems. Before we start discussing the solution for our project, we need to first understand what is hydroponics. Hydroponics can simply be defined as a method for growing plants using a solution of mineral nutrient and water. No soil is used in this method for growing plants. Using this technique terrestrial plants are grown with their roots in the mineral solution, or in an inert medium, such as gravel or perlite.

One of the first work to be published about growing of plants without soil was found in 1627 in a book “*Sylva Sylvarum*” by Francis Bacon. This book was published one year after his death. After the book was published and became famous, water culture became a very popular research technique. With detailed research going on in the field, a report on experiment performed by John Woodward was published in 1699. The report explained how after an experiment it was concluded that plants grow better and faster in less pure water than pure distilled water. This came as a surprise to everyone and became a major research area. [1]

After many years of experimentation finally a list of nine elements which were found to be essential for plant growth was compiled in 1842. And further research on what is essential for plant growth resulted in the finding of soilless agriculture. This technique was first discovered by botanists Julius von Sachs and Wilhelm Knop, in the years 1859–1875. This technique of growing plants in minerals and nutrients without soil was known as solution culture. Research went on and many other ways of agriculture were introduced. Solution culture is still very commonly practiced in many areas and is usually considered a type of hydroponics. Hydroponics is agriculture by providing necessary nutrients to the plant by using water as an

inert medium. Whereas solution culture provides plants with the required nutrients but does not use any inert medium. [1]

Until 1929 solution agriculture was only being discussed and being experimented on. Then it finally was promoted at University of California Berkeley to be used as a method for agriculture. William Frederick Gericke introduced the concept of providing the required nutrients to the plants by using water as the inert medium. He first named the technique aquaculture but he later realized that aquaculture is a term used to describe culture of aquatic organisms. Gericke went up to grow twenty-five feet high tomato vines in his backyard by not using soil but instead using nutrient mineral solution. Later in 1937 he went on to name the technique Hydroponics (water culture). [1]

As reports and papers for Gericke's work were published, more and more people started becoming inquisitive and interested in Hydroponics. Most of people's questions were answered by Gericke in his book "Complete Guide to Soilless Gardening."

As the research went on it was proven that hydroponics has many advantages. Some of them being, giving access to required amount of water and oxygen to the plants. Giving the right amount of water to a plant is important because in most cases of gardening or even agriculture the plant is either over fed or under fed with water, which adversely affects the growth of the plant. Hydroponics prevents these mistakes by supplying ample amount of water to the plant. The plant uses up as much water as required and the rest is drained and re-collected in a reservoir placed in the system. This also helps to conserve water and the nutrients from being run-off into the soil. On the counterpart, when growing plants in soil, the grower needs to be very well informed about the water requirements for each given plant. If excess water is

supplied, it will cut down the plants oxygen accessibility and too little water will starve the plant of nutrients. [2]

One of the earliest successes cases of hydroponics was heard of on Wake Island, which is a rocky atoll in the Pacific Ocean. The Wake Island is used by Pan American Airlines as a re-fuelling stop. Hydroponics was used at this island in 1930s to grow vegetables for the passengers. Hydroponics was a necessity on Wake Island because there was no soil, and it was prohibitively expensive to airlift in fresh vegetables. [2]

Fig 1. A simple Hydroponic system architecture.

Figure 1 [1] helps us understand how a basic hydroponic system works. The plants to be used for hydroponic growing are placed in a pot with no soil. The roots are given direct access to water that is supplied using the water pump. Various nutrients and oxygen are also supplied to the plant using this water. All the extra water is drained out and stored back into the reservoir which can be re-used at some later point in time. We can see in the figure that there is no

wastage of water and the plant gets as much water as required. Some hydroponic systems may be simpler or more complicated compared to this one but all of them have a similar functionality. Now a days hydroponics is being used extensively in areas where it is difficult to grow food in the soil such as urban areas or areas such as food deserts where the soil is not fit for growing. Using hydroponics helps us to conserve water, grow faster, avoid weeds and pests and get better control. Considering all these factors we used hydroponics to build our system. [2]

3. MOTIVATION

The motivation for this project came from all the user research that was done for this project. Earing the issues people face and to be able to fix some of these problems by creating this system gave this project a start.

3.1 Research Questions

The initial research goals for this project were truly stemmed from an initial assumption; that individuals who are doing hydroponic urban farming would be interested in automating as much of their growing as possible. With this assumption the initial goals were developed focused on understanding user needs within the urban farming space. The initial goal was predominantly focused on identifying current solutions, how users leverage them, and identifying potential opportunities for the system to improve upon existing solutions.

This focus on user needs and opportunities for differentiation subsequently allowed us to focus on three major questions for the research:

❖ **What are the key metrics users are interested in?**

This question was focused on understanding what the users really care about from an informational standpoint. There are almost an infinite number of different aspects of a plant that can be tracked and recorded and to ensure that the information that was critical to the users is not missing. Additionally, to make sure that the solution does not include extraneous or confusing information that the user is unlikely to need or even want.

❖ **What are the major pain points for users when using existing systems?**

Here the focus was predominantly on developing empathy for the users. Running hydroponic grow systems is difficult and so we wanted to understand common problems users are running into with running and maintaining these systems. This serves the dual purpose of understanding where they are currently struggling and need help, and understanding what potential solutions can be provided via the Hydrobase system.

❖ **What role does technology play in their growing process?**

This question was focused on understanding user's current technical abilities, experiences and typical tasks. We attempted to understand if most users leverage technology to support their existing growing set-up or behavior and if so what they were using. This is important because we need to use this as a way to gauge both interest in technical solutions as well as understand proficiency with technology.

Each question was focused on attempting to understand a different aspect of a potential user's experience, the goal of which was to help develop different personas for Hydrobase.

We attempted to make sure that the individuals being recruited were technologically inclined or had substantial knowledge of urban farming, which meant that they would be a good fit for Hydrobase but as per the research we began to realize that there was a need to take a step back. So in addition to the original goals, we also began to try and understand at a more meta-level why people are growing plants in the first place.

3.2 Methodology

3.2.1 Survey

We first conducted surveys for a wide range of audience and these surveys helped us identify different personas based on needs and requirements. Since the aim of the project was to develop a platform to empower small to medium scale urban growers, feedback from non-growers was not valuable and so only participants with some growing experience were accepted. Since the target user base of enthusiasts and small-scale professionals is relatively niche, finding a large number of qualified respondents is difficult. Our survey questions were exhaustive and most of the questions were open-ended. This left us with a lot of valuable information. However, the disadvantage of this survey structure is that it was probably too time consuming for many people to fill out.

❖ Recruiting

To recruit for the survey, we turned to Reddit. On Reddit we found several relevant sub-reddits including:

- IndoorGarden - 13,832 subscribers
- Hydro - 7,523 subscribers
- Urbanfarming - 7,353 subscribers

- H2Grow - 479 subscribers
- UrbanAgriculture - 167 subscribers

We ended up only posting to the top three because they had the most subscribers.

Fig 2. Geographical locations of survey respondents.

Figure 2 shows a map with the locations of the survey respondents.

Reddit fortunately provided geographic diversity, which was helpful because people in different climates and urban areas typically have different growing habits which we wanted to learn about. The other unexpected benefit of Reddit were the comments on our post requesting survey responses. There people expressed concerns about modularity and pricing which are both important for us to understand.

❖ *Question Design*

The survey questions were set to start out easy to build up a rapport between the participant and the researcher. In the beginning there were close-ended questions about what they were growing. These are easy and quick to answer and allow some buildup of momentum. Below is the general flow of topic categories:

- What is the participant growing?
- What motivates the participant to grow?
- What challenges does the participant face?
- What is the role of technology in the participant's process?

We did our best to phrase questions with simple unambiguous words so the users did not have any unnecessary mental strain. The types of questions covered were characteristic, behavioral and attitudinal. All were either freeform answers or Likert Scale. A copy of the surveys sent out to participants is shown in Appendix A.

3.2.2 *Interviews*

One of the questions on the survey asked if the participant was willing to engage in an interview, and if they were, to provide contact information. We used this information to recruit interview subjects.

Interviews allow us to gain a deeper understanding of user needs. Although we used a question guide, purposely we let the conversation meander and take us where it might. As is typical with one-on-one interactions participants seemed to open up on a personal level more than they did on the relatively impersonal survey.

❖ *Interview Design*

We followed our interview guide and started by thanking the participant and then proceeded to give them context for the session. They were told that we are a graduate student research group developing technology to grow plants that are healthier to consume and have a smaller impact on the environment. We emphasized on how important it is for us to learn about real growers' processes, what motivates them and what their pain points are. Similar to the survey, we started out with easy close-ended questions, to build momentum, then proceeded to delve into more personal value based open-ended questions. We then concluded by profusely thanking them for their time and let them know how important their feedback was to our product. The growing needs interview guide followed by our team is shown in Appendix A.

3.2.3 *Usability Design*

The aim of all these surveys and interviews was looking for problems understanding how to:

- Instantiate a new grow
- Give the grow a name
- Associate the grow with a circuit board
- Associate sensors with the specific grow
- Set a condition that the system needs to respond to (pH getting below a certain point)

3.3 **Key Findings**

Overall respondents were supportive and enthusiastic about the Hydrobase system and were interested in beta testing future iterations. Below are the key findings from the survey, interviews and usability tests:

3.3.1 *User needs*

- ❖ **On Motivation:** Gardening generally made the survey respondents feel relaxed. They feel accomplished when something they've nurtured survives. Also many people, especially in California, are motivated by environmental concerns. This may be due to the recent drought. Several respondents expressed interest in using less water and energy. Furthermore, some find comfort in knowing that if the grocery store closed down, they would be able to survive on the food that they grew themselves.
- ❖ **On pain points:** Several people state that testing the conditions of their plants was time consuming and they wish they could offload it through some automation. People also shared that they sometimes get anxious about whether they are correctly taking care of their plants and that they wish there was a more centralized repository of ideal growing condition knowledge. Experimenting on one's own is difficult because there is a long delay between action and feedback. It is hard to isolate what is causing a plant to wither or thrive.
- ❖ **Feature suggestions:** Below are a few opinions users shared about specific system features:
 - Collecting data on the plants is more important than automation, though both are desired.
 - Modularity of the system is key. People want to be able to start small with a growing kit and scale up from there without starting over.
 - Growers want to be able to test daylight conditions to optimize the use of artificial light.
 - People want to know if they are ahead, behind or on schedule to harvest.

- It would be helpful to have tips about plant cleaning and maintenance. Also growers want reliable suggestions for getting rid of pests in a healthy way.

Based on all this information from a varied source of users we realized that there is a huge population who has interest in growing/ gardening but do not have the right set up available. These user interactions gave us the motivation to work towards building an open source system which is low in cost and is user friendly, such that a user can deploy one for their use and not run into small day to day growing issues.

4. RELATED WORK

Over the years' hydroponics has grown and developed. In today's evolving world, this technology has become so efficient and handy that it is now being used for agriculture world wide. It has helped remove all the constraints that once existed in farming. Now because of hydroponics we do not require huge lands and we do not require growth efficient soil for agriculture. Apart from that the growth results that we get from hydroponic agriculture are much better than what we get from traditional agriculture techniques. But in today's busy and digitalized world the problem still remains that agriculture still remains a manual job and not many youngsters in the coming generations are ready to take up this job. And because of this problem of shying away from manual labor, we are having a food crisis in so many parts of the world. Even if we do have people at places doing agriculture, the traditional agriculture techniques in most cases pollutes the soil and the water reservoirs hence leaving them unfit for future use. All these things urge us to build an automated system that does not require physical labor in the fields and does not cause harm to that nature. All these things have lead people to work towards building automated systems for agriculture. Most of these systems work on hydroponic agriculture.

4.1 MIT Open Agriculture Initiative (OpenAG)

One of the first such automated hydroponic agriculture systems to be built was **OpenAG**. It was founded in 2015 by Caleb Harper. It is a product that took birth at MIT Media Lab (Massachusetts Institute of Technology). The aim of this project is to develop a controlled environment agriculture platform. This project or these systems that are developed for this agriculture platform are called "Food Computers". These food computers operate on multiple scales. And these specialized systems can be used for a variety of things like experiments, personal use or even education. This whole project, i.e. the hardware and software are open source. The reason to make this platform open source is to be able to create a standard platform that can be used for urban agriculture and can also help with research and experimentation. [4]

The main and most essential product of Open AG project is "Food Computer". These food computers were first developed as part of the MIT City FARM project. Similar to our project Hydrobase, Food Computers also use soilless agriculture technology, hydroponics, to grow crops indoors. The Food Computer has quite a few sensors inbuilt into it that monitor various factors like the climate and other external conditions necessary for the plant. All this monitoring is then used to be able to control all the external and internal factors needed for the plant growth. Everything in the food computers is highly controlled and is tested very closely by researchers and scientists. [4]

Since the system is open source the contributors can track of their conditions, data and researches such that the fixed recipes created for each plant growth can be shared with anyone who may be interested. This seems to be a good approach to cut down on the climatic conditions and also helps save transportation cost as now anything can be grown in these

restricted environments without having to care about climate of any other such factors. We no more need to rely on finding big grounds for agriculture. Using these food computers, we can get fresh produce even in the middle of an urban city. [4]

Fig 3. MIT City Farm

In Figure 3 [4] we can see a MIT City Farm at the MIT Media Labs.

4.1.1 Food Computer

The Food Computer as discussed earlier allows us to build a controlled environment that can be used for urban agriculture that deploys multiple robotic systems to monitor and control external factors such as climate and energy. These food computers also monitor the plant growth in the chamber very closely. By controlling and monitoring climate variables we mean being able to keep a check on the levels of carbon dioxide, potential hydrogen, air temperature,

electrical conductivity, humidity, root-zone temperature, and dissolved oxygen. Apart from the above mentioned features, water level, energy level, and mineral levels are monitored using sensors and electrical meters. Most of these features can be adjusted using the sensors and chemical doses provided to the plants all through the research and experiment. [5]

Now considering all the above mentioned features we can have multiple combinations of the features and any such useful combination can be thought to be a climate recipe. And each such recipe will produce different agricultural results for a given plant. Given a same type of plant grown in different climate recipes might be different in color, size, yield, taste, nutrients, texture and growth rate. This helps the researcher to be able to judge the best recipe for a given plant. These food computers can also be used to induce drought or other such conditions and then judge the growth and health of the plant.

Fig 4. Food Computer

Figure 4 [4] shows a food computer which is being used currently.

We can build food computers in multiple sizes and based on the sizes the food computers can be classified in the following types:

- ❖ **Personal Food Computer** – this is the smallest size of the food computer and is mainly used to build small scale environments mainly for experimentation and learning biology, environment sciences, botany or programming. Such systems can be used by people for their hobby or by students in school for learning purposes. [5]
- ❖ **Food Server** –this is the intermediate size and is built mainly for people who want to start a small scale research. They can also be used by small scale growers like restaurant or cafeteria owners to grow small fresh produce that can be used as ingredients for the food they cook and serve. [5]
- ❖ **Food Datacenter** – This size is still in the development phase but the aim is to build huge food computers that can help replace agricultural fields. It is called the datacenter because it is expected to be more like a huge warehouse where food can be grown for industrial use without any restriction. The plan is to have multiple subdivisions to be able to grow different types of crops in each subdivision based on their optimal growing requirements. [5]

4.1.2 Climate Recipes:

As we move towards globalization, the food production and supply chain has also become a big industry. Today all food suppliers depend on mono crop farmlands that produce and supply huge amounts of produce all across the world. Most of the time we do not realize this but the food that we eat is usually produced thousands of miles away and reaches us days or may be even months later than when it was harvested. This concept of importing food from where it is produced adds cost and with that also affects the food quality. [5]

Many standard climate recipes are provided with the system and the user can import any such climate recipe. Most of these recipes have been researched and tested for the given plants to give best results but most of these recipes can be tweaked and changed as and when required by the user to get desired results. As discussed earlier, inside these chambers the climate can be strictly controlled and hence can be used to get better yields. Also the system logs the results of the growth and results achieved based on a new climate recipe. All the updated recipes are stored in an open source database and these new recipes can be used by new users all around the world as and when required. [5]

Vertical farming done with the help of hydroponics in a food computer helps to grow food in urban areas and helps to reduce carbon footprints. The amount of water being used for farming is also reduced greatly. The food supplied to the end user is much more fresh as the transportation time is greatly reduced. Also it helps to reduce risk from external factors like climate changes, droughts, etc. [5]

There are many features that food computers by OpenAG provide to their user, but the issue is that all the processing in the food computers is very controlled. Every thing they grow and experiment with is done in a controlled environment and after doing our user research we realized that it is not always possible to create such a controlled environment. Also Food Computers are not yet commercially available. As of 2016, there are six prototype Personal Food Computers operating in schools around the Boston area, and three Food Servers operating at MIT, Michigan State University, and Unidad Guadalajara (Cinvestav) in Mexico. Build directions and schematics are available for makers and hobbyists, while more-widespread availability is expected once manufacturing begins in coming years. [4]

4.2 Grove Ecosystem

Another set up that is now coming up in the market is the Grove Ecosystem. It is again a similar concept with few changes. It is not as controlled as OpenAg and it focuses more on a user's day to day needs. The Grove Ecosystem again was first developed at MIT. Its makers claim it to be the world's first intelligent, in-home garden.

The Grove ecosystem uses a process called aquaponics to implement an in-home gardening system. Aquaponics is similar to hydroponics, but instead of only using water with minerals for the growth of the plant, it uses water and minerals from an aquarium. The plants grow with the help of the water from the aquarium and the minerals come from the fish feces and microbes from the aquarium. This combination of aquarium and plant agriculture basically build an ecosystem where all the things in the system are interdependent on each other. The plants depend on the aquarium and its creatures for its water and mineral needs and the fish and other creatures in the aquarium depend on the plants grown in the system to get their food.

This system can help grow small vegetables and fruits like tomatoes, peppers, lettuce, etc. The system has its own LED lighting and plumbing that is very efficient and keeps the system running perfectly. There is also a Grove OS and a mobile application that helps to monitor and control the Grove ecosystem and also helps guide the end users through the growing process. When used to the most of its utilization, this Grove Ecosystem can help the users to grow and fulfill at least all their salad needs for each day. This system has also been designed based on the aesthetic details and can be used by the users in their house as a decorative piece. It also allows fish lovers to have an aquarium and pets. [6]

Fig 5. A prototype of the Grove Ecosystem

Figure 5 [6] above shows a prototype of the Grove Ecosystem. As we can see it looks like a well-made cabinet, and the developers of this system claim that it is designed in such a way that it can be placed in the bedroom, kitchen or even living room. [6]

The lighting used in the system delivers wavelengths of light that is best suited for the crops. These lights match the wavelengths of lights based on the wavelengths the plants would receive in each season if they were grown outdoors. The timings of lighting and watering can be controlled through a smartphone app. There are also tutorials available on the applications for the user that can help them understand the other requirements of the plants and the system. The Grove Ecosystem is expected to sell commercially at \$4,500 when it is released in the market. The company is expecting to start the shipping by March, 2017. [7]

Although, the Grove Ecosystem seems to be very promising and user friendly, there are a few concerns. The cost for the system is \$4,500. Not everyone is looking to invest that amount into a gardening system. Also they provide a pre-defined kit and the user is not allowed to make changes or experiment with the system. [7]

4.3 Hydrobase

With the two up coming systems mentioned above there many challenges mainly focusing on the user friendly part of the applications and that is where Hydrobase started. OpenAG and Grove Ecosystem have their existing systems which they provide to their users. For this reason, when a user wants to start building his/her own little in-door garden with the basic steps, they might not be able to do so. These systems can not be expanded or contracted as and when the user's requires. Also the Ecosystem provides some hard coded predefined plant profiles for the user to choose from. Also most of the times the information required by the user to work upon the system is either missing or is not available due to propriety issues.

Hydrobase aims to provide the user with a low budget open source system to build their own indoor hydroponic garden. The user can start off with just the basic sensors and then can keep growing as required. The only cost involved in the system is that of the sensors and the set up.

5. SYSTEM DESIGN

5.1 Overview

The main goal of the system design is to support interoperability, and even though it seems like a trivial task, it was defined as one of the most important aspects of making the solution as open as possible. With that central goal in mind, all the other decisions were focused on facilitating flexibility in every aspect of the system. Subsequently it uses open- source hardware, a highly extensible database and micro-application framework to ensure that the

system is capable of adapting and evolving over time. This flexibility in the system turns out to be extremely important because the landscape for open agriculture technology has changed substantially in a very short period of time. Figure 6 below shows an overview of the network diagram for Hydrobase.

Fig 6. Network diagram.

5.2 System Architecture

The electronic system design focuses on using “off the shelf” components for a number of reasons, but the main one being to support user success. It ensures that users can build the same system themselves if they so desired. With this in mind some research was done on different platforms, their capabilities, availability, cost and also their support community to understand

what systems potential users are likely to be familiar with. From this research an Arduino Mega was chosen for use, as it was capable of working with a number of different peripherals in the Arduino suite of products, offered ample I/O and more than enough RAM for the applications that would be running on it. Additionally, Arduino is a very popular brand of open-source hardware with a strong community supporting it.

Fig 7. Electronic hook up diagram.

Figure 7 above shows the electronic hook up diagram for Hydrobase.

Once the main board was selected, the next aim was to choose applicable sensors and actuators. The project focused on what would be important information such as air temperature, electrical conductivity, pH, lux as well as important actuators such as water pumps and peristaltic pumps. The team specifically chose to keep the sensors and actuators as minimal as possible, again to try and ensure users would be successful without spending a fortune on their system.

5.3 Information Architecture

We used MongoDB to store all the data to power the hydrobase application. The database design for hydrobase was pretty complex and a lot of time was spent on working and reworking the database design so that we could optimize access to the database and with short queries that would allow the application to run faster. In the end the database was split into the following collections:

5.3.1 Users

The user collection contains the information about the user. Whenever a user signs up with Hydrobase, a document is created in the User collection for that particular user. Here is a sample document in the user's collection:

```
{  
  "email": "iotmims16@gmail.com", "username": "admin", "password":  
  "pbkdf2:sha1:1000$JifYdXt6$258a0e3729d059d292368dfd0ccdb290858285ce",  
  "zip": "94703"  
}
```

The username assigned to each user is unique and also serves as the channel name on pubnub for all the grows/devices associated to that particular user to communicate on.

We hashed the password at the backend before storing into the database in order to improve security. While signing in this password is converted back to plain text and then compared with the one entered by the user.

5.3.2 Devices

The devices collection contained the information about the devices. Whenever a device is added to the application, a document is created in the Devices collection. We assigned each device an “id” which was randomly generated GUID. A sample document from the devices

collection is shown below. This device document contains device information like name, identifier, kit, emergency stop, user the device belongs to and sensors and actuators associated with the device.

```
{
  "actuators": {
    "light_2": "31",
    "light_1": "30",
    "phUpper_pump": "35",
    "nutrient_pump": "33",
    "water_pump": "32",
    "phDowner_pump": "34"
  },
  "type": "Arduino",
  "device_name": "Wario",
  "device_id": "202d31e6-aa8b-4fe6-b479-285362ea6606",
  "kit": "standard",
  "emergency_stop": "false",
  "username": "admin",
  "sensors": [
    "Lux",
    "Water_Temp",
    "Air_Temp",
    "Humidity",
    "pH",
    "EC",
    "TDS",
    "PS"
  ]
}
```

5.3.3 Grows

The grows collection contains information about the different plants that belong to a particular user. Whenever a new grow is added through the application, a document in the grows collection is created. A sample document in the grows collection:

```
{
  "username": "admin",
  "experiment": "false",
  "grow_name": "Lily",
  "actuators": {
    "light_2": "31",
    "light_1": "30",
    "phUpper_pump": "34",
    "phDowner_pump": "35",
    "water_pump": "32",
```

```

    "nutrient_pump": "33"
  },
  "sensors": [
    "Lux",
    "Water_Temp",
    "Air_Temp",
    "Humidity",
    "pH",
    "EC",
    "TDS",
    "PS"
  ],
  "device_id": "68dc6c7e-8c19-499b-b167-5bd4fe37ff3d", "plant_profile":
  "capsicum_annuum_'jalapeño'", "controls": {
    "condition": [
      {
        "value": 6,
        "operator": "<",
        "action": "on",
        "actuator": "phUpper_pump",
        "sensor": "pH",
        "unit": "pH"
      },
      {
        "value": 6,
        "operator": ">",
        "action": "off",
        "actuator": "phUpper_pump",
        "sensor": "pH",
        "unit": "pH"
      },
      {
        "value": 6.8,
        "operator": ">",
        "action": "on",
        "actuator": "phDowner_pump",
        "sensor": "pH",
        "unit": "pH"
      },
      {
        "value": 6.8,
        "operator": "<",
        "action": "off",
        "actuator": "phDowner_pump",
        "sensor": "pH",
        "unit": "pH"
      },
      {
        "value": 600,
        "operator": "<",

```

```

        "action": "on",
        "actuator": "nutrient_pump",
        "sensor": "EC",
        "unit": "EC"
    },
    {
        "value": 700,
        "operator": ">",
        "action": "off",
        "actuator": "nutrient_pump",
        "sensor": "EC",
        "unit": "EC"
    } ],
    "time": [ {
        "action": "toggle",
        "actuator": "light_1",
        "value": 12,
        "unit": "hours"
    },
    {
        "action": "toggle",
        "actuator": "light_2",
        "value": 12,
        "unit": "hours"
    },
    {
        "action": "toggle",
        "actuator": "water_pump",
        "value": 15,
        "unit": "minutes"
    }
    ]
},
    "device_name": "Mario"
}]

```

A document in the grows collection contains information about a particular grow like name, device associated to it, sensors and actuators linked to that grow as well as the various controls both time and condition based for that particular grow.

5.3.4 Plant Profiles

The plant profiles collection contains the information about the various profiles that we sourced from USDA and other reliable sources and that are available to the user to create their grows from. A sample document from plant profiles collection:

```
{
  "fruit_nut_seed_product": "Yes",
  "fruit_nut_seed_color": "Green",
  "drought_tolerance": "Low",
  "shade_tolerance": "Intermediate",
  "precipitation_minimum": "Medium ",
  "ph_maximum": 5.8,
  "foliage_color": "Green",
  "height_mature_feet": 8,
  "precipitation_maximum": "High",
  "growth_period": "Spring/Summer",
  "bloom_period": "Autumn",
  "flower_conspicuous": "Yes",
  "ph_minimum": 5.2,
  "growth_rate": "Medium",
  "scientific_name": "Citrullus lanatus",
  "temperature_minimum_f": 45,
  "identifier": "citrullus_lanatus",
  "salinity_tolerance": "Low",
  "lifespan": "Moderate",
  "fruit_nut_seed_conspicuous": "No",
  "fruit_nut_seed_begin": "Autumn ",
  "fruit_nut_seed_end": "Fall",
  "common_name": "Watermelon",
  "fruit_nut_seed_abundance": "Medium"
}
```

5.3.5 Data

The data collection contains the data that is gathered by the various sensors on the devices. Each packet of data that is published by the device to pubnub is added to the data collection as a document. The timestamp is also appended along with the other information in order to identify the time at which that data was captured.

A sample document from the data collection:

```
{
  "airTemp": 75,
```

```

    "hour": 9,
    "min": 11,
    "lux": 45,
    "PS": "0.35",
    "EC": "723.8",
    "humidity": 33,
    "sec": 36,
    "year": 2016,
    "TDS": "390",
    "grow_name": "Lily",
    "waterTemp": 74.1875,
    "month": 5,
    "pH": "6.598",
    "day": 6,
    "device_id": "68dc6c7e-8c19-499b-b167-5bd4fe37ff3d"
  }

```

So the document contains all the sensor information along with the timestamp and the device and the grow information that is associated with it.

5.3.6 CV Data

CV data is published once a day and contains the data that we gathered using computer vision.

A sample CV document:

```

{
  "CV_Data": {
    "sample_3": {
      "time_stamp": "2016-05-06 05:03:04.340355",
      "foliage_density": {
        "fd_raw": 17.218166666666665,
        "fd_increase": 0
      },
      "grow_name": "Lily",
      "height": {
        "height_increase": -7,
        "height_raw": 78
      }
    }
  }
}

```

5.3.7 PubNub Communication

We use PubNub, “the global data stream network for IoT, mobile, and web applications,” to send data to and from our sensors. It is what controls the information flow. Devices can both subscribe and publish to channels, which enables “reading” and “writing,” respectively.

To store the sensor data, the application subscribes to the channel and sends what it receives to the database.

In order to automate actuator behavior based on user-specified settings, the device listens for messages in the following format:

```
{  
  
  "68dc6c7e-8c19-499b-b167-5bd4fe37ff3d": {  
  
    "30": 255,  
  
    "31": 255}  
  
}
```

This includes the device ID as well as pin numbers and associated values. The pin numbers correspond to actuators and the values instruct the device to turn them on or off. In our setup, off is represented by 0 and on is represented by 255.

These values are determined by the user-specified of the default settings. For example, a user can set the lighting schedule to something like 12 hours on and 12 hours off. We call these controls and work with two types: time-based and condition-based controls. Time-based controls only depend on time while condition-based controls depend on data.

Typically, lights and water pumps will be set up using time-based controls. Nutrient and pH pumps, on the other hand, will be set up using condition-based controls. With pH, an action, such as increasing or decreasing the pH, is triggered based on the most recent data. For example, a user may specify that pH not go below 6.5. In this case, if the most recent sensor read is below that value, the pH “upper” is triggered and the pH level of the solution is brought back into an acceptable range.

5.4 Physical setup

The physical set up was optimized for cost, strength and simplicity. For \$500 - \$1000 someone could replicate what has been built, the lion's share of that cost being the sensors. Early on in this project using 3D printed parts were discussed for the structure but later we decided against that since what was needed was accomplished for less money and time buying parts from Home Depot and Amazon. Figure 8 below shows an early concept drawings of the structure.

Fig 8. Early Concept Drawings of the Structure

The framing itself is built out of PVC pipe and PVC pipe connectors. Other parts of the system are common plastic buckets, rubber tubing and twine. The documentation on the Hydrobase web application provides detailed instructions on what to buy, how to assemble and even provides a downloadable 3D model which specifies drill hole locations.

We did have some issues with leaking when the aeroponic sprayers were on for extended periods of time but were able to resolve the problem for the most part with copious amounts of glue gun glue. In the future we would want to design a modular screw/snap-together grow system that lets growers scale up over time.

Figure 9 and 10 below show the finished structure and the annotated structure and electronic setup respectively.

Fig 9. Finished structure

Fig 10. Annotated structure and electronic setup.

5.5 User Interface

Based on the amount of information that should simultaneously be visualized and the complexity of the tasks being executed, desktop web seemed like the best primary platform. The first low-fidelity desktop mockup of the web application was created in Balsamiq. The “sketchy” nature of this prototype marked it ideal for evaluating what layout works best and which high level functions are important without getting caught up with color preferences, typefaces and other polishing touches. Furthermore, finally having a graphical representation of what we had been discussing in the abstract for weeks made our conversations more concrete

and also gave us a better sense of what scope we could handle for this project. Figure 11 below shows the low-fidelity desktop mockup of the web application created in Balsamiq.

Fig 11. Low fidelity mockup.

After agreeing upon an updated high-level organization, we transitioned to an interactive prototype in Justinmind. This prototype was useful for showing off what our web application would eventually be capable of.

We were able to quickly learn and iterate using the prototypes so by the time we started coding the actual web application, we had a much better sense of what we wanted to build and how it should be organized.

5.6 Information Visualization

The information visualization is extremely important. The problem with the information visualization is that if everything is working perfectly with the system, there is actually very little variation for displaying to the user via information visualization. Subsequently the system has two different types of visualizations. The visualizations on the “grow” page are the “sanity checks” to ensure that the system is doing what it is supposed to and that the user does not need to intervene. In order to accomplish this, we leveraged the data about the different plants to illustrate a “safe zone” within which the plant would be comfortable and displayed it as a range within which the data points should lie.

The real time dashboard on the other hand has a slightly different use case. The information visualization on the dashboard helps the user understand quickly what is happening at that moment with all of their devices. To have this functionality we developed a dashboard that displayed the 10 most recent data points and it continuously updates itself as new data comes in. This is effective to understand the current conditions of the system and potentially can aid in either setup or system modification as the user is able to get immediate feedback from the system.

5.7 Measuring success

The general factors that can be considered in order to measure success are the growth in height of the plant and increment in foliage density. However, these factors are qualitative and can be explored using image processing and computer vision techniques to measure these factors quantitatively.

The 2 factors that can be used to quantitatively define success over a certain period of growth time of the plant are:

- ❖ Increase in height
- ❖ Increase in foliage density

For this purpose, a raspberry pi board with a compatible camera that is fixed in its place is set up. The raspberry pi-camera and OpenCV libraries on the raspbian operating system are installed. The user needs to run the module (hydrobase_cv.py) and keep the camera fixed in a stable location. The camera is adjusted to focus on the image whose growth needs to be measured. The camera takes three consecutive pictures of the plant every 12 hours when the light is natural and the leaves seem to be in their normal color. At present, we are only considering green leaved plants for Computer Vision application, since more than 98% of the plants in the current database have green foliage. Since growing lamps are being used to grow the plants as a part of the hardware setup, pictures need to be taken of the plant in normal surroundings when there is no artificial glow on the plant.

Then the algorithms are run on the collected images to measure the growth in height and foliage density of the plant over a certain period of time.

6. HYDROBASE APPLICATION DESIGN

6.1 Dashboard

The dashboard page on the application is an aggregate page that allows the user to view all his grows in one go. The dashboard page list shows graphs for pH, electrical conductivity, temperature, humidity and lux which display the real time data that is coming in from the sensors. Thus the users know the status of their grow in real time. The dashboard also allows the user to compare different grows against each other with the help of graphs. Figure 12 below shows the sensor visualization dashboard on the Hydrobase application.

Fig 12. Sensor visualization dashboard

6.2 Devices

In order to effectively manage the different plants our users would be growing, we needed to ensure that we have an accurate representation of the devices (microcontrollers) that were controlling the plant's environment. Since we were allowing interoperability of different devices, we needed to create a way for users to explain their hardware to the system. Subsequently, we created a section of the application where users can manage the devices in great detail.

The devices page allows the user to add new devices and to identify a number of different critical aspects of the device such as the type, associated sensors and associated actuators. Additionally, for the actuators the users are able to identify which pins the actuators are connected to, thus allowing Hydrobase to activate them and either turn pumps / lights on or

off. Finally, for more technically inclined users, the devices section gives the user keys to the pub/sub system we are using and their specific channel, making it possible for them to connect many different devices to the system without Hydrobase needing to be specifically programmed for the device.

The devices page is also used to help users who are less technically inclined as it offers the option to use our pre-designed kit and gives the user the code required to connect to Hydrobase, send data and receive commands from the system. Again this was done in an explicit attempt to support both the technically inclined as well as the less technically inclined. Figure 13 below shows the device management page on the Hydrobase application.

Fig 13. Device management page

6.3 Plant Profiles

We collected most of our plant profiles from the US Department of Agriculture data that is available online. We have a collection of more than a thousand plants and there are several parameters that are placed in the database. The parameters that we consider significant are scientific name, pH range, precipitation range, temperature and drought tolerance, growth rate and period, average mature height, lifespan, relevant flower/fruit/nut/seed attributes, and foliage color. There were nearly 100 plant species that are quite commonly grown by people but were missing from the USDA database. We scraped the relevant data for these plants using online resources. All the data that we collected from USDA and various other reliable sources was cleaned and massaged to remove any irregularities and then inserted into the database. Each one of these profiles is available to the user to create a grow. Figure 14 below shows the plant profile page on the Hydrobase application.

Fig 14. Plant profile page

6.4 Grows

Grows corresponds to an instance of a plant that the user creates in the system. A grow can either be created from a predefined plant profile which will pick up the default values for various parameters like minimum and maximum pH, electrical conductivity and air temperature from the plant profiles derived from the USDA data. Or a grow can also be customized as per the user's needs, allowing them to set the ranges for the above parameters manually. This is done by setting up controls that are either time based or condition based. The time based controls perform an actuator action like turn the light on or turn the water on, in a timely manner like every 12 hours. On the other hand, the condition based controls perform an action only when a particular condition is met. For example, turn on the pH upper pump if the pH levels fall below the range specified by the user. This allows the user to create grows with custom profiles which can later be compared to the default profiles provided by USDA. The user can also opt to participate in an experiment while creating a grow. This option allows the user to set the range of values for various parameters like pH and electrical conductivity within a specified range of the defaults provided by the USDA for that plant. The grows that participate in experiments have a special flag that is set to true so that the system can identify those plants and hence use the data from those plants for testing and experiment purposes.

In addition to all these flexibilities provided to the user while creating a grow, the application also provides the user the flexibility to change the sensors and actuators linked to a grow as well as add, remove or change the time and condition based controls that have been applied to a particular grow.

For the experimentation purposes of Hydrobase, we grew an Avocado and a Jalapeno plant in our set up. Figure 15 below shows the growth of these plants in the Hydrobase set up over 65 days.

Fig 15. Hydrobase application being used to grow.

6.5 Notifications

We implemented a way to notify the user when any grow that belongs to the user, either the time based control or condition based control, is tripped. Although the system can rectify the condition that was tripped by turning on or off the associated actuator, the user is still notified so that they can dig deeper and find out the root cause for the same. In the application, the new notifications appear in bold and the old ones are not bold.

6.6 Emergency Stop

In the application we have provided the user with a capability to stop a device all-together. When this option is chosen, the actions that had to be performed by that device would be temporarily suspended until the user releases the device from the Emergency stop state. This was not planned in initially but later we thought that the users might want to perform certain actions once in a while that required the devices to be suspended and hence we implemented this feature. Figure 17 below shows the edit device and emergency stop page on the Hydrobase.

Fig 17. Edit device and emergency stop page

6.7 Data Export

During our user research one of the main feedback that we got from the users was that most of the existing hydroponics setups either did not capture and store the data at all or if they did the data was not exposed to the user. We provided the user with a way to download all the data for their grows as a JSON file that could be loaded into other data analysis and visualization tools.

6.8 Documentation

It quickly became clear based on the complexity of our system that good documentation was a must. Buying the wrong part or missing a single connection could be the difference between the system running smoothly or not at all. For the structure we created a 3D model so people could see clearly what the end product should look like. It also included all the drill hole locations for people to connect aeroponic tubes and hang the buckets.

6.9 Login/ Sign up

The login and signup pages are simple HTML forms that ask the user for various inputs like username, email, password, name and zip code. We have performed some error checking on the page so that the email was in proper format and password had some minimum requirements. The password was hashed and stored in the database for the purpose of security.

7. SYSTEM SECURITY

As discussed earlier in the report, we used PubNub to establish our IoT application, it helps to send data to and from the sensors. PubNub controls the information flow and using this channel the devices can both subscribe and publish to channels, which enables both reading and writing.

The reason for choosing PubNub for our application was simple. PubNub is a highly scalable publishing and subscribing messaging platform. It can automatically scale up or down to

handle millions of requests coming in every second. One of the best use cases that PubNub offers is "Remote Device Control for IoT". This feature helps prevent attacks like DoS. It helps make the system more secure and robust.

PubNub also ensures data security by providing end-to-end encryption with TLS/SSL and AES security protocols. In addition, with the help of Access manager and a combination of Publish key, Subscribe key, Secret key, and auth key, the users of our application can fine-grain permissions down to person, device, or channel level.

8. CONCLUSION

The entire process of building the software application based on the well created (but not perfect) hardware has been incredibly educational, certainly the most informative of my time at San Jose State University. We have been able to incorporate not only our technical skills but also our information organization skills, communication skills as well as socio-technical skills in an attempt to develop a different approach from what we are currently seeing.

Yet, none the less, the project has raised more questions than it has answered. We certainly feel that technology similar to this could be incredibly helpful and impactful in the future but it seems to be such early days that its exact form still appears quite uncertain. So, we felt it was best for us to explore a potential solution, as opposed to being so bold as to say that we have developed the ultimate solution. This light-hearted interpretation allowed us to meet with a wide range of individuals from local entrepreneurs to professors of botany, and truly emphasized the notion that a purely product-driven perspective is not always the best approach for understanding a problem and its potential solutions. Subsequently we are extremely pleased to have had the opportunity to explore and work hard on a problem we feel is worthwhile.

From a hardware perspective, the increasing availability of “off the shelf” open-hardware will only make it easier to build these systems, increasing the pace of exploration and discovery. For this reason, we feel it is critical to incorporate data capture and analysis systems to ensure that the collected data is not lost and so that knowledge and information can be effectively shared with a larger audience. Nonetheless there are strong economic forces at play that have an incentive to leverage lock-in, meaning that without strong advocates in the open-source community the information could be difficult or impossible to access for the greater public.

Our sole purpose of integrating this IOT application is to automate the process of taking care of your plants in a hydroponics setup in a seamless manner. As opposed to the existing systems, Open AG and Grove Ecosystem that not only cost the user a lot but also only provide their own fixed systems to the user that can not be expanded or contracted, our value proposition to the user is reduced hardware cost, coupled with an intelligent setup platform. Hydrobase provides better automation than any other solution available out there, and this translates to a better user experience with substantial user time savings.

9. FUTURE WORK

9.1 Permutation Testing

In order to determine the grow profiles that yield the best results, we have developed a permutation-testing-based method for assessing each profile’s efficacy. A benefit of permutation testing is that it does not rely on assumptions about how the data are distributed or about independence between observations. While there are several outcomes that users might care about, such as number of fruits or taste, we focus on biomass.

A challenge with comparing grow profiles is the number of possible combinations. For example, one user might specify particular lighting and water cycles in conjunction with

specific upper- and lower-bound pH levels while another might specify completely different settings. To alleviate this issue, we reduce the number of choices that can be made for each actuator. We are yet to fully implement and test the permutation testing module.

9.2 Crowdsourcing grow profiles

One of the challenges with agriculture is that machine readable data is not readily available. There is also a lot of misinformation about the type of conditions that produce the best results. Once our platform scales, we will have access to profile and outcomes data from large numbers of users, which will allow us to determine the settings that are most effective for plant grows. Similar to how large communities of people work together to make reliable Wikipedia articles, we hope to create the most comprehensive and reliable repository of plant recipes using the crowd.

9.3 Modular physical kit

As we have stated a number of times, a main focus of our system was to allow for exploration and experimentation in either the plants or the hardware and we feel that modular physical kits could play a critical role in this. We did choose our system to ensure people had everything they need but not more, yet as we did our user research we realized how substantial a barrier the “unknown” was. Modular physical kits would allow users to “get their feet wet” before committing to a more expensive system, thus mitigating to a degree this barrier of the unknown.

Additionally, our conceptualization of these physical kits is that they would be decomposable, allowing users to dismantle them and rebuild them in whatever fashion was most applicable to their needs. This serves a dual purpose of ensuring users are not being “locked in” to different products, thus further decreasing the uncertainty barrier, as well as allowing users to develop

systems we could never have imagined. Unfortunately, we were unable to accomplish much more in the way of kit development outside of our initial system, but this is an area I am eager to explore.

10. REFERENCES

- [1] M. H. Jenesen and W. L. Collins (2011), “Hydroponic Vegetable Production,” In: Jules Janick ed. *Horticulture Reviews*, Vol 7, The AVI Publishing Company, Inc. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/9781118060735.ch10/summary>.
- [2] H. M. Resh 2013, *Hydroponic Food Production*, Florida: CRC Press. [Online]. Available: <https://books.google.com/books>.
- [3] D. S. Domingues, H. W. Takahashi, Carlos A.P. Camara, S. L. Nixdorf (2012), “Automated system developed to control pH and concentration of nutrient solution evaluated in hydroponic lettuce production”, In: *Computer and Electronics in Agriculture*, Vol 84, Elsevier, pp 53-61. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168169912000361>
- [4] V. Protonotarios, “Open Source Agriculture? Meet the OpenAg Initiative!” News, Open Data. [Online]. Available: <http://blog.agroknow.com/?p=4330>
- [5] C. Harper and M. Siller, “OpenAG: A Globally Distributed Network of Food Computing,” in *Proc. 4th IEEE Pervasive Computing*, Vol 14, 2015, pp 24-27. [Online]. Available: <http://ieeexplore.ieee.org/xpl>
- [6] The Boston Globe, Grove Begins Selling Indoor Garden ‘Ecosystem’ on Kickstarter, [Online]. Available: <http://www.betaboston.com/news/2015/11/03/grove-labs-begins-selling-indoor-garden-ecosystem-on-kickstarter>.

- [7] Inhabitat, Amazing Grove Aquaponic Ecosystem Can Grow You a Salad Everyday, [Online]. Available: <http://inhabitat.com/amazing-grove-aquaponic-ecosystem-can-grow-you-a-salad-every-day>
- [8] J. Zhao, J. Zhang, Y. Feng and J. Gua, "The Study and Application of the IOT Technology in Agriculture," IEEE, China, 2010.
- [9] W. Zhang, "Study about IOT's Application in Digital Agriculture Construction," IEEE, China, 2011.
- [10] M. Anderson, "MIT's Food Computer: The Future of Urban Agriculture?" IEEE Spectrum, January 2016, [Online]. Available: <http://spectrum.ieee.org/computing/embedded-systems/mits-food-computer-the-future-of-urban-agriculture>
- [11] P. Wahba, "Target to Partner with MIT on City Farming, other Food Science Projects," Fortune, October 2015, [Online]. Available: <http://fortune.com/2015/10/20/target-cityfarming>
- [12] J. Temperton, "In the future, even broccoli will have an IP address," *in Conf.* Wired 2015, October 2015, [Online]. Available: <http://www.wired.co.uk/news/archive/2015-10/15/caleb-harper-urban-farms-mit-wired2015>
- [13] E. Price, "Open-sourced food production – the future of urban diets?" Open Source, The Guardian, November 2015, [Online]. Available: <https://www.theguardian.com/technology/2015/nov/10/open-sourced-food-production-future-urban-diets>
- [14] S. Brands, "Team at MIT CityFARM Growing Food 3-4x Faster, with 90% Less Water," Cleantech, October 2014, [Online]. Available:

http://www.sustainablebrands.com/news_and_views/cleantech/sustainable_brands/team_mit_cityfarm_growing_food_3_4x_faster_90_less_water

- [15] H. Hansman, “What Is a Personal Food Computer?” Smithsonian.com, August 2015, [Online]. Available: <http://www.smithsonianmag.com/innovation/what-is-a-personal-food-computer-180956085/?no-ist>
- [16] M. McFarland, “Inside an MIT Researcher’s Grand Plan to Create the Personal Food Computer,” Innovations, The Washington Post, June 2015, [Online]. Available: <https://www.washingtonpost.com/news/innovations/wp/2015/06/17/inside-an-mit-researchers-grand-plan-to-create-the-personal-food-computer>
- [17] J. Byron, “An Ecosystem Approach to Growing Food,” Grove Lab Blog, May 2016, [Online]. Available: <http://blog.grovelabs.io/an-ecosystem-approach-to-growing-food-part-1>
- [18] E. Kolsun, “Aquatic Bliss: The Grove Aquarium,” Grove Lab Blog, May 2016, [Online]. Available: <http://blog.grovelabs.io/aquatic-bliss-the-grove-aquarium>
- [19] USDA, “United States Department of Agriculture: Data and Statistics,” October, 2015, [Online]. Available: http://www.usda.gov/wps/portal/usda/DATA_STATISTICS
- [20] USDA, “United States Department of Agriculture: Foreign Agriculture Services,” Data Analysis Databases, October, 2015, [Online]. Available: <http://www.fas.usda.gov/data>

APPENDIX A

User research

1. Recruiting flyer

Interested in automating or optimizing your hydroponics setup?

We'd love to talk to you!

We're a group five of UC Berkeley graduate students developing a user-friendly hydroponic system that's both automatically controlled and cross-platform compatible. We're looking for small to medium scale growers to interview for 45 – 60 minutes to better understand our users' needs. Our goal is to advance urban agriculture and agricultural experimentation, so you'd be contributing to a system that enables growers to work more locally and more efficiently.

Please e-mail japple@berkeley.edu if you are willing to contribute an hour of your time.

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

hydroponics automation
japple@berkeley.edu

2. Survey

Growing System User Needs

Hydrobase is a flexible and visual platform for growing plants that are healthy for you and sustainable for the environment. With intuitive controls and analytics, our aim is to make the urban hydroponics growing process smarter and more fun! Right now we're focused on learning more about our prospective users so we can better serve their needs. We appreciate you carving out the time to provide thoughtful answers to this survey.

*** Required**

Name

Your answer

E-mail *

Your answer

Location

City, State

Your answer

What most interests you about hydroponics?

e.g. it uses less water than growing in soil, it uses less energy than traditional farming and distribution, high nutrient levels in plants, consistency of plant output, growing year round, advancing knowledge of ideal growing conditions, technology is fun

Your answer

What kind of plants are you interested in growing?

e.g. tomatoes, basil, kale, orchids

Your answer

For what purpose are you growing?

e.g. personal use; sell to grocery stores; use in my restaurant

Your answer

How much do you want to grow?

e.g. enough spinach for 3 salads per month; 3 plants; as much as my apartment can hold

Your answer

Are you currently growing plants? If so, what are you growing and what is your setup?

Your answer

How much space do you have to grow indoors?

Provide length, width, height estimates if possible

Your answer

If you have indoor space that could be used for growing, does it have access to sunlight?

Your answer

How much space do you have to grow outdoors?

Provide length, width, height estimates if possible

Your answer

How much gardening experience do you have?

Your answer

How does gardening make you feel? Why?

e.g. frustrated, relaxed, happy...

Your answer

Hydrobase is a web-based application that connects to a physical hydroponics (growing plants without soil) system. Would you be more interested in connecting this platform to a standard kit or linking it to your own customized system?

Your answer

Rate These by level of importance

	Not important	----	Somewhat important	---	Very important	No opinion
I want to use automated controls to give light to my plants	☐	☐	☐	☐	☐	☐
I want to use automated controls to provide a nutrient solution to my plants	☐	☐	☐	☐	☐	☐
I want to use automated controls to provide carbon dioxide to my plants	☐	☐	☐	☐	☐	☐
I want to use automated controls to maintain the pH of my nutrient solution	☐	☐	☐	☐	☐	☐
I want to use automated controls to maintain the ratio of nutrients to water	☐	☐	☐	☐	☐	☐
I want to be able to collect data on pH	☐	☐	☐	☐	☐	☐
I want to be able to collect data	☐	☐	☐	☐	☐	☐
on humidity						
I want to be able to collect data on light levels	☐	☐	☐	☐	☐	☐
I want to be able to collect data on temperature	☐	☐	☐	☐	☐	☐
I want to be able to collect data on solution nutrient levels	☐	☐	☐	☐	☐	☐
I want to be able to collect data on bacteria levels	☐	☐	☐	☐	☐	☐
I want to be able to compare the success of similar plants under different growing conditions for optimization purposes.	☐	☐	☐	☐	☐	☐
I want this system to be modular so I can gradually scale it.	☐	☐	☐	☐	☐	☐
I want to be able to browse what other people are growing and the conditions they grew in.	☐	☐	☐	☐	☐	☐
I want to be able to browse a library of						

standard plant profiles (e.g. for chili peppers or lettuce) that apply optimal settings to my growing controls

Are there features that have not been mentioned that you think this system should have?

Your answer

How would you describe your level of comfort with technology?

Your answer

How much experience do you have working with microcontrollers like Arduino or Raspberry Pi?

Your answer

How much would you be willing to pay for a physical hydroponics growing kit that allows you to automatically control lights, pH, nutrient levels, water pumps for multiple plants?

Don't worry you don't need to commit to this. :)

Your answer

Assuming most of the features mentioned in this survey were enabled, how much would you be willing to pay for a monthly subscription to the Hydrobase control and analytics platform?

Your answer

Would you be willing to be interviewed about your growing needs?

If so, provide a phone number, or we could do a Google Hangouts call.

Your answer

Would you be willing to be a beta tester of our software?

Don't worry, even if you say yes now, you can opt out at any time.

Your answer

Other comments?

Your answer

3. Growing needs interview guide

***Note to interviewer:** If interviewee responded to our survey, be sure to familiarize yourself with their responses before the interview and adapt your questions accordingly. Make it seem like we value their previous input.*

Intro: Hi, I'm <your name> Thanks so much for taking the time to do this interview. How's everything going today?

I thought I'd start by giving an introduction to our project to give some context to the questions I'm going to be asking. After that, we'll spend most of the time talking about you and your growing process. Does that sound good?

It will help me better focus on the conversation if I don't need to focus too intensely on taking notes. Would you mind if I recorded our conversation?

<Great, thank you> or <no worries, that shouldn't be a problem> <start recording>

OK, I've started the recording. Thanks for agreeing to that <interviewee name>. So I'm <your name> and I'm working with a team of UC Berkeley graduate students on developing web-based platform for analyzing sensor data and controlling plant growth on a small to medium scale. It's called Hydrobase. Our aim is create technology that enables people to grow plants that are healthier to consume and have a smaller impact on the environment.

As we develop this system, we want to be sure we are addressing actual user needs, and are not just making up imaginary problems, which is where you come in. This interview serves to help us understand a real grower's process and how it could be

improved upon. We want to learn about your pain points and also what motivates you.

With that, I'll jump into the questions. Sound good?

Question guide (remember to go with the flow and ask follow up questions)

- ❖ What city are you based out of?
- ❖ Tell me about your grow setup.
- ❖ How much time each week do you spend gardening?
- ❖ Why do you grow? What are your primary goals?
- ❖ What challenges do you face while growing?
- ❖ Tell me about a recent positive experience you had while growing.
- ❖ Tell me about a recent negative experience you had while growing.
- ❖ What role, if any, does technology currently play in your growing process?
- ❖ If you could have one technology, and it doesn't need to exist yet, added to your growing process, what would it be and why?
- ❖ Do you ever run experiments with your grows? If so, tell me about that.
- ❖ How much money have you invested in your current setup?
- ❖ How do you measure success with your grows? Yield? Taste? They didn't die?
- ❖ How do you track growth? E.g. harvest time, when to water, pH levels?
- ❖ How do you do record keeping around your grows?
- ❖ Do you do any data analysis currently?
- ❖ Would you be willing to be a beta tester?

- ❖ How do you find out the ideal growing conditions for your plants? E.g. water, pH, light, nutrients
- ❖ How important is the aesthetic quality of your growing setup?

Conclusion: Those are all the questions I had. Is there any other information you think my team should consider as we develop this technology? Is there any questions you have for me? Thank so much for taking the time to do this interview. You were super helpful and we appreciate it! Feel free to follow up with any questions or comments by email. Take care, goodbye.

APPENDIX B

Code Implemented

1. Config.py

```
import os
_basedir = os.path.abspath(os.path.dirname(__file__))

DEBUG = True

SRF_ENABLED = True
SECRET_KEY = 'you-will-never-guess'

MONGO_URI = 'mongodb://admin:admin@ds011268.mlab.com:11268/analytics-hydrobase'
DB_NAME = 'analytics-hydrobase'

PUBNUB_PUBLISH_KEY = 'pub-c-93c6c384-e1a0-412f-87cf-e626aaab6a00'
PUBNUB_SUBSCRIBE_KEY = 'sub-c-8ec9d89e-e4aa-11e5-a4f2-0619f8945a4f'
PUBNUB_SECRET_KEY = 'sec-c-YzQyMTU3NmYtMDNhMS00MzM5LTg3MTgtZjA2N2U0N2IyNGY3'
PUBNUB_AUTH_KEY = '40ed6434-1991-4f7a-8034-20a072abde43'
PUBNUB_CHANNEL_GRP = "hydrobase"
```

2. Db_writer.py

```

from pymongo import MongoClient
from pubnub import Pubnub
import logging
import datetime

def _callback(message):
    logging.info(message)

def _error(message):
    logging.info(message)

def sub_callback(message, channel):
    # print(channel)
    if "CV_Data" in message.keys():
        logging.basicConfig(filename='hydrobase_cv.log', level=logging.INFO)
        logging.info(message)
        db.cv_data.insert_one(message)
    elif 'sender' in message.keys():
        message_array = []
        utc_datetime = datetime.datetime.utcnow()
        message['year'] = utc_datetime.year
        message['month'] = utc_datetime.month
        message['day'] = utc_datetime.day
        message['hour'] = utc_datetime.hour
        message['min'] = utc_datetime.minute
        message['sec'] = utc_datetime.second
        if 'EC' in message.keys():
            message['TDS'] = message['EC'].split(",")[1]
            message['PS'] = message['EC'].split(",")[2]
            message['EC'] = message['EC'].split(",")[0]

        device_id = message['sender']['device_id']
        message['device_id'] = device_id
        message.pop('sender', None)
        related_grows = db.grows.find({'device_id' : device_id})
        date_now = datetime.datetime(year=message['year'], month=message['month'], day=message['day'], \
            hour=message['hour'], minute=message['min'], second=message['sec'])
        for grow in related_grows:
            message_temp = message.copy()
            message_temp['grow_name'] = grow['grow_name']
            message_array.append(message_temp)
            data_points = db.data.find({'grow_name' : grow['grow_name']}).sort('_id', -1).limit(1)
            for data_point in data_points:
                date_last = datetime.datetime(year=data_point['year'], month=data_point['month'], day=data_point['day'], \
                    hour=data_point['hour'], minute=data_point['min'], second=data_point['sec'])
                time_diff = date_now - date_last

        # insert in display
        logging.basicConfig(filename='hydrobase_display.log', level=logging.INFO)
        logging.info(message_array)
        db.display.insert_many(message_array)

        # insert in data db if last entry was more than 4 hours back
        if int(time_diff.seconds) >= 14400:
            logging.basicConfig(filename='hydrobase_data.log', level=logging.INFO)
            logging.info(message_array)
            db.data.insert_many(message_array)

```

```

if __name__ == '__main__':

    client = MongoClient('mongodb://admin:admin@ds011268.mlab.com:11268/analytics-hydrobase')
    db = client['analytics-hydrobase']

    #initialize the pubnub instance
    pubnub = Pubnub(publish_key='pub-c-93c6c384-e1a0-412f-87cf-e626aaab6a00', \
        subscribe_key='sub-c-8ec9d89e-e4aa-11e5-a4f2-0619f8945a4f', secret_key='sec-c-YzQyMTU3NmYtMDNhMS00MzM5LTg3MTgtZjA2N2U0N2IyNGY3', \
        auth_key='40ed6434-1991-4f7a-8034-20a072abde43')

    # Grant read, write and manage permissions to the pubnub instance that we initialized
    pubnub.grant(channel_group='hydrobase', auth_key='40ed6434-1991-4f7a-8034-20a072abde43', \
        read=True, write=True, manage=True, ttl=0, callback=_callback, error=_error)

    # Subscribe to the channel group 'hydrobase' that contains the channels for all users to get the data
    # coming in from different devices and put that into the DB
    pubnub.subscribe_group(channel_groups='hydrobase', \
        callback=sub_callback, error=_error)

```

3. Views.py

```

from flask import Blueprint, render_template, redirect, url_for
from flask.ext.login import current_user
from app import db

# Define the blueprint: 'app'
mod_app = Blueprint('app', __name__)

# Set the route and accepted methods
@mod_app.route('/')
@mod_app.route('/index')
def index():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard.dashboard'))
    else:
        return render_template('index.html')

```

4. Usermessaging.py

```

import datetime
from app import db

def addMessage(username, subject, content):
    temp_dict = {}
    users = db.users.find({'username': username})
    for user in users:
        total_msgs = user['message_dict']['total_count']
        print user['username']
        print total_msgs

    temp_dict['subject'] = subject
    temp_dict['content'] = content
    temp_dict['time'] = datetime.datetime.now().strftime("%I:%M%p %b %d")
    temp_dict['id'] = total_msgs+1
    temp_dict['unread'] = 1

    db.users.update_one(
        { "username" : username},
        {
            '$push': {"message_dict.message_list" : temp_dict},
            '$inc' : {"message_dict.total_count" : 1, "message_dict.unread_count" : 1},
        }
    )
    return

def deleteMessage(username, id):
    users = db.users.find({'username': username})
    for user in users:
        messages = user['message_dict']['message_list']
        for message in messages:
            if message['id'] == id:
                if message['unread'] == 1:
                    unread_msg = 1
                else:
                    unread_msg = 0

```



```

if unread_msg == 1:
    db.users.update_one(
        {"username" : username},
        {
            '$dec' : {"message_dict.total_count" :1, "message_dict.unread_count" :1},
            '$pull' : {"message_dict.message_list" :{ "message_dict.message_list.id" : id}}
        }
    )
else:
    db.users.update_one(
        {"username" : username},
        {
            '$dec' : {"message_dict.total_count" :1, "message_dict.unread_count" :1},
            '$pull' : {"message_dict.message_list" :{ "message_dict.message_list.id" : id}}
        }
    )
return

def markAsRead(username, id):
    db.users.update_one(
        {"username" : username, "message_dict.message_list.id" : id},
        {
            '$dec' : {"message_dict.unread_count" :1},
            '$set' : {"message_dict.message_list" :{ "message_dict.message_list.unread" : 0}}
        }
    )
    return

def returnMessageList(username):
    users = db.users.find({'username': username})
    for user in users:
        # total_count = user['message_dict']['total_count']
        # total_unread = user['message_dict']['total_unread']
        # messages = user['message_dict']['message_list']
        return user['message_dict']

```

5. Auth – Forms.py

[illegible]

6. Auth – Models.py

```
class User():

    def __init__(self, username):
        self.username = username

    def is_authenticated(self):
        return True

    def is_active(self):
        return True

    def is_anonymous(self):
        return False

    def get_id(self):
        return self.username
```

7. Auth – Views.py

```
from flask import Blueprint, render_template, redirect, url_for, request, flash
from flask.ext.login import login_required, login_user, logout_user
from werkzeug import check_password_hash, generate_password_hash

from app import db, login_manager, pubnub, app, _callback
from .models import User
from .forms import LoginForm, SignupForm

from app import usermessaging

mod_auth = Blueprint('auth', __name__)

@mod_auth.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm(request.form)
    error = None
    print(request.method)
    if request.method == 'POST':
        user = db.users.find_one({'username': request.form['username']})
        if not user:
            error = 'User does not exist'
        elif not check_password_hash(user['password'], request.form['password']):
            error = 'Invalid credentials. Please try again.'
        else:
            user_obj = User(user['username'])
            login_user(user_obj)
            return redirect(url_for('dashboard.dashboard'))
    return render_template('auth/login.html',
                           title='Log In to Hydrobase',
                           form=form,
                           error=error)

@mod_auth.route('/signup', methods=['GET', 'POST'])
def signup():
    form = SignupForm(request.form)
    error = None
    if request.method == 'POST':
        existing_user = db.users.find_one({'username' :
                                           request.form['username']})
        if existing_user:
            error = 'Username already exists'
        else:
            new_user = {'username' : request.form['username'],
                        'email' : request.form['email'],
                        'zip' : request.form['zip'],
                        'password' : generate_password_hash(request.form['password']),
                        'message_dict' : {
                            'total_count' : 0,
                            'unread_count' : 0,
                            'message_list' : []
                        }
            }
    }
```

```

        db.users.insert_one(new_user)
        subject = "Welcome to hyderbase"
        content = "..... XYZ....."
        usermessaging.addMessage(request.form['username'], subject, content)
        user = db.users.find_one({'username': request.form['username']})
        pubnub.channel_group_add_channel(channel_group=app.config['PUBNUB_CHANNEL_GRP'], channel=user['username'])
        pubnub.grant(channel=user['username'], auth_key=app.config['PUBNUB_AUTH_KEY'], read=True, write=True, manage=True, ttl=0)
        return redirect(url_for('dashboard.dashboard'))
    return render_template('auth/signup.html', form=form,
                           title='Sign Up for Hydrobase', error=error)

# @mod_auth.route('/googlelogin', methods=['GET', 'POST'])

@mod_auth.route("/logout")
@login_required
def logout():
    logout_user()
    flash("Logged out.")
    return redirect('/login')

@login_manager.unauthorized_handler
def unauthorized_callback():
    return redirect('/login')

@login_manager.user_loader
def load_user(username):
    u = db.users.find_one({'username': username})
    if not u:
        return None
    return User(u['username'])

def callback(message, channel):
    db.data.insert_one(message)

def error(message):
    db.data.insert_one(message)

```

8. Dashboard – Views.py

```

import flask
from flask import Blueprint, render_template, Response, url_for, redirect
from flask.ext.login import login_required, current_user
from app import db
import json

from app import usermessaging

# Define the blueprint: 'dashboard'
mod_dashboard = Blueprint('dashboard', __name__)

@mod_dashboard.route('/dashboard')
@login_required
def dashboard():
    device_list = []
    grows_list = []
    range_list = []
    seed_data = []
    username = current_user.get_id()
    devices = db.devices.find({'username': current_user.get_id()})
    grows = db.grows.find({'username': current_user.get_id()})
    if devices.count() == 0 or grows.count() == 0:
        return redirect(url_for('devices.list_devices'))
    else:
        for device in devices:
            device_list.append((device['device_name'], device['type'], \
                                device['sensors'], device['actuators'], device['kit'], device['device_id']))
        grows = db.grows.find({'username': current_user.get_id()})
        for grow in grows:
            grows_list.append((grow['grow_name'], grow['device_name'], grow['sensors'], grow['actuators']))
            ph_min = 0
            ph_max = 0
            ec_min = 0
            ec_max = 0
            for condition_control in grow["controls"]["condition"]:
                if condition_control['actuator'] == 'phUpper_pump':
                    ph_min = condition_control['value']
                if condition_control['actuator'] == 'phDowner_pump':
                    ph_max = condition_control['value']
                if condition_control['actuator'] == 'nutrient_pump' and condition_control['action'] == 'off':
                    ec_max = condition_control['value']
                if condition_control['actuator'] == 'nutrient_pump' and condition_control['action'] == 'on':
                    ec_min = condition_control['value']
            range_list.append({"grow_name": grow['grow_name'], "ph_min": ph_min, "ph_max": ph_max, "ec_min": ec_min, "ec_max": ec_max})
        grows = db.grows.find({'username': current_user.get_id()})

```

```

for grow in grows:
    pH_list = []
    lux_list = []
    EC_list = []
    TDS_list = []
    PS_list = []
    humidity_list = []
    airTemp_list = []
    waterTemp_list = []
    data_points = db.display.find({'grow_name' : grow['grow_name']}).sort('_id', -1).limit(10);

    for data_point in data_points:
        pH_list.append(data_point['pH'])
        lux_list.append(data_point['lux'])
        EC_list.append(data_point['EC'])
        TDS_list.append(data_point['TDS'])
        PS_list.append(data_point['PS'])
        humidity_list.append(data_point['humidity'])
        airTemp_list.append(data_point['airTemp'])
        waterTemp_list.append(data_point['waterTemp'])
    pH_list.reverse()
    lux_list.reverse()
    EC_list.reverse()
    TDS_list.reverse()
    PS_list.reverse()
    humidity_list.reverse()
    airTemp_list.reverse()
    waterTemp_list.reverse()
    seed_data.append({'grow_name': grow['grow_name'], "pH" : pH_list, "lux" : lux_list, "EC" : EC_list, \
        "TDS" : TDS_list, "PS" : PS_list, "humidity" : humidity_list, "airTemp" : airTemp_list, \
        "waterTemp" : waterTemp_list})

message_dict = usermessaging.returnMessageList(username)

return render_template('dashboard/dashboard.html', username=username, my_devices=device_list, \
    my_grows=grows_list, range_list=range_list, seed_data=seed_data, message_dict = message_dict)

@mod_dashboard.route('/dashboard/markRead/<id>')
@login_required
def markRead(id):
    username = current_user.get_id()
    usermessaging.markAsRead(username, id)
    return redirect(url_for('dashboard.dashboard'))

@mod_dashboard.route('/get_data/', methods = ['GET'])
@login_required
def get_data():
    concatenated_data = {}
    username = current_user.get_id()
    grows = db.grows.find({'username' : current_user.get_id()})
    for grow in grows:
        data_points = db.data.find({'grow_name' : grow['grow_name']})
        for data_point in data_points:
            data_point.pop('_id', None)
            concatenated_data.setdefault(grow['grow_name'], []).append(data_point)

    return flask.jsonify(**concatenated_data)

```

9. Devices – Views.py

```

from flask import Blueprint, render_template, request, url_for, redirect
from app import db, login_manager, pubnub
from flask.ext.login import login_required, current_user
from app.auth.models import User
import uuid

mod_devices = Blueprint('devices', __name__)

@mod_devices.route('/devices', methods=['GET'])
@login_required
def list_devices():
    print "HEREEEE 22222"
    device_list = []
    grows_list = []
    UUID = str(uuid.uuid4())
    username = current_user.get_id()
    devices = db.devices.find({'username': current_user.get_id()})
    for device in devices:
        device_list.append((device['device_name'], device['type'], \
            device['sensors'], device['actuators'], device['kit'], device['device_id'], device['emergency_stop']))
    grows = db.grows.find({'username' : current_user.get_id()})
    for grow in grows:
        grows_list.append((grow['grow_name'], grow['device_name']))
    return render_template('devices/devices.html', my_devices=device_list, my_grows=grows_list, username=username, uuid=UUID)

```

```

@mod_devices.route('/add_device/<new_device_id>', methods=['POST'])
@login_required
def add_device(new_device_id):
    username = current_user.get_id()
    existing_device = db.devices.find_one({'device_name' :
                                           request.form['device_name']})

    if not existing_device:
        if request.form['kit'] == "standard":
            new_device = {'username' : username, 'device_id': new_device_id, 'device_name' : request.form['device_name'], 'type' : 'Arduino', 'kit' : request.form['kit'], \
                          'sensors' : ['Lux', 'Water_Temp', 'Air_Temp', 'Humidity', 'pH', 'EC', 'TDS', 'PS'], \
                          'actuators' : {'light_1' : "30", "light_2" : "31", "water_pump" : "32", "nutrient_pump" : "33", "phUpper_pump" : "34", "phUpper_pump" : "35"}, 'emergency_stop': 'false'}
        else:
            sensors = []
            if request.form['Lux'] == 'on':
                sensors.append("Lux")
            if request.form['Water_Temp'] == 'on':
                sensors.append("Water_Temp")
            if request.form['Air_Temp'] == 'on':
                sensors.append("Air_Temp")
            if request.form['Humidity'] == 'on':
                sensors.append("Humidity")
            if request.form['pH'] == 'on':
                sensors.append("pH")
            if request.form['EC'] == 'on':
                sensors.append("EC")
            if request.form['TDS'] == 'on':
                sensors.append("TDS")
            if request.form['PS'] == 'on':
                sensors.append("PS")
            actuators = {}
            if request.form['light_1_pin'] != "":
                actuators['light_1'] = request.form['light_1_pin']
            if request.form['light_2_pin'] != "":
                actuators['light_2'] = request.form['light_2_pin']
            if request.form['water_pump_pin'] != "":
                actuators['water_pump'] = request.form['water_pump_pin']
            if request.form['nutrient_pump_pin'] != "":
                actuators['nutrient_pump'] = request.form['nutrient_pump_pin']
            if request.form['phUpper_pump_pin'] != "":
                actuators['phUpper_pump'] = request.form['phUpper_pump_pin']
            if request.form['phDowner_pump_pin'] != "":
                actuators['phDowner_pump'] = request.form['phDowner_pump_pin']
            new_device = {'username' : username, 'device_id': new_device_id, \
                          'device_name' : request.form['device_name'], 'type' : request.form['device_type'], \
                          'kit' : request.form['kit'], 'emergency_stop': 'false', 'sensors' : sensors, 'actuators': actuators}
            db.devices.insert_one(new_device)
    return redirect(url_for('devices.list_devices'))

@mod_devices.route('/edit_device/<device_id>', methods=['POST'])
@login_required
def edit_device(device_id):
    sensors = []
    if request.form['Lux'] == 'on':
        sensors.append("Lux")
    if request.form['Water_Temp'] == 'on':
        sensors.append("Water_Temp")
    if request.form['Air_Temp'] == 'on':
        sensors.append("Air_Temp")
    if request.form['Humidity'] == 'on':
        sensors.append("Humidity")
    if request.form['pH'] == 'on':
        sensors.append("pH")
    if request.form['EC'] == 'on':
        sensors.append("EC")
    if request.form['TDS'] == 'on':
        sensors.append("TDS")
    if request.form['PS'] == 'on':
        sensors.append("PS")
    actuators = {}
    if request.form['light_1_pin'] != "":
        actuators['light_1'] = request.form['light_1_pin']
    if request.form['light_2_pin'] != "":
        actuators['light_2'] = request.form['light_2_pin']
    if request.form['water_pump_pin'] != "":
        actuators['water_pump'] = request.form['water_pump_pin']
    if request.form['nutrient_pump_pin'] != "":
        actuators['nutrient_pump'] = request.form['nutrient_pump_pin']
    if request.form['phUpper_pump_pin'] != "":
        actuators['phUpper_pump'] = request.form['phUpper_pump_pin']
    if request.form['phDowner_pump_pin'] != "":
        actuators['phDowner_pump'] = request.form['phDowner_pump_pin']
    result = db.devices.update_one(
        { "device_id" : device_id },
        {
            '$set': {'device_name': request.form['device_name'], 'sensors' : sensors, 'actuators' : actuators}
        },
        upsert=True
    )

```



```

grows = db.grows.find({'device_id' : device_id})
for grow in grows:
    g_sensors = grow['sensors']
    g_actuators = grow['actuators']
    g_controls = grow['controls']
    for s in g_sensors:
        if s not in sensors:
            g_sensors.remove(s)
    for key in g_actuators.keys():
        if key not in actuators.keys():
            g_actuators.pop(key, None)
    for time_control in g_controls["time"]:
        if time_control["actuator"] not in actuators.keys():
            g_controls["time"].remove(time_control)
    for condition_control in g_controls["condition"]:
        if condition_control["actuator"] not in actuators.keys() or condition_control["sensor"] not in sensors:
            g_controls["condition"].remove(condition_control)
    result = db.grows.update_one(
        { "grow_name" : grow['grow_name'] },
        {
            '$set': {'device_name': request.form['device_name'], 'sensors' : g_sensors, 'actuators' : g_actuators, 'controls': g_controls :
        },
        upsert=True
    )

    return redirect(url_for('devices.list_devices'))

@mod_devices.route('/delete_device/<device_id>', methods=['POST'])
@login_required
def delete_device(device_id):
    username = current_user.get_id()
    result = db.grows.update_many(
        { "device_id" : device_id },
        {
            '$set': {'device_name': "", 'device_id':"", 'sensors' : [], 'actuators' : {}, 'controls':{}}
        },
        upsert=True
    )

    device = db.devices.delete_one({'device_id' : device_id})
    return redirect(url_for('devices.list_devices'))

@mod_devices.route('/emergency_stop/<device_id>', methods=['POST'])
@login_required
def emergency_stop(device_id):
    result = db.devices.update_one(
        { "device_id" : device_id },
        {
            '$set': {'emergency_stop': request.form['emergency_stop_status']}
        },
        upsert=True
    )

    return redirect(url_for('devices.list_devices'))

```

10. Grows – Views.py

```

from flask import Blueprint, render_template, request, url_for, redirect
from app import db, login_manager, pubsub
from flask.ext.login import login_required, current_user
import datetime

mod_grows = Blueprint('grows', __name__)

@mod_grows.route('/grows/<current_grow>', methods=['GET'])
@login_required
def list_grow(current_grow):
    pHChartData = []
    ecChartData = []
    luxChartData = []
    tempChartData = []
    humidityChartData = []
    range_list = []
    user_devices = []
    user_grows = []
    device_list = []
    grows_list = []
    assoc_device_name = ''
    username = current_user.get_id()
    grows = db.grows.find({'grow_name' : current_grow})

```

```

for grow in grows:
    assoc_device_name = grow['device_name']
    experiment = grow['experiment']
    grows_list.append((current_grow, grow['device_name'], grow['sensors'], grow['actuators'], grow['controls'], grow['plant_profile']))
ph_min = 0
ph_max = 0
ec_min = 0
ec_max = 0
grows = db.grows.find({'grow_name' : current_grow})
for grow in grows:
    for condition_control in grow["controls"]["condition"]:
        if condition_control['actuator'] == 'phUpper_pump':
            ph_min = condition_control['value']
        if condition_control['actuator'] == 'phDowner_pump':
            ph_max = condition_control['value']
        if condition_control['actuator'] == 'nutrient_pump' and condition_control['action'] == 'off':
            ec_max = condition_control['value']
        if condition_control['actuator'] == 'nutrient_pump' and condition_control['action'] == 'on':
            ec_min = condition_control['value']
    range_list.append({'grow_name': str(grow['grow_name']), "ph_min": ph_min, "ph_max": ph_max, "ec_min": ec_min, "ec_max": ec_max})
data_points = db.data.find({'grow_name' : grow['grow_name']}).sort('_id', -1).limit(42);
for data_point in data_points:
    date = datetime.datetime(year=data_point['year'], month=data_point['month'], day=data_point['day'], \
                             hour=data_point['hour'], minute=data_point['min'], second=data_point['sec'])
    pHChartData.append({'year': data_point['year'], "month":data_point['month'], \
                       "day": data_point['day'], "hour": data_point['hour'], "minute": data_point['min'], \
                       "second":data_point['sec'], "value" : float(data_point['pH'])})
    ecChartData.append({'year': data_point['year'], "month":data_point['month'], \
                       "day": data_point['day'], "hour": data_point['hour'], "minute": data_point['min'], \
                       "second":data_point['sec'], "ec_value" : float(data_point['EC']), "tds" : float(data_point['TDS']), "ps": float(data_point['PS'])})
    luxChartData.append({'year': data_point['year'], "month":data_point['month'], \
                        "day": data_point['day'], "hour": data_point['hour'], "minute": data_point['min'], \
                        "second":data_point['sec'], "value" : float(data_point['lux'])})
    tempChartData.append({'year': data_point['year'], "month":data_point['month'], \
                         "day": data_point['day'], "hour": data_point['hour'], "minute": data_point['min'], \
                         "second":data_point['sec'], "waterTemp" : float(data_point['waterTemp']), "airTemp": float(data_point['airTemp'])})
    humidityChartData.append({'year': data_point['year'], "month":data_point['month'], \
                             "day": data_point['day'], "hour": data_point['hour'], "minute": data_point['min'], \
                             "second":data_point['sec'], "value" : float(data_point['humidity'])})

if assoc_device_name == "":
    device_list.append(("No Device Linked", "No Device Linked", [] , {}, "", ""))
else:
    devices = db.devices.find({'device_name': assoc_device_name })
    for device in devices:
        device_list.append((device['device_name'], device['type'], \
                             device['sensors'], device['actuators'], device['kit'], device['device_id']))

devices = db.devices.find({'username': current_user.get_id()})
for device in devices:
    user_devices.append((device['device_name'], device['type'], \
                         device['sensors'], device['actuators'], device['kit'], device['device_id']))
grows = db.grows.find({'username' : current_user.get_id()})
for grow in grows:
    user_grows.append((grow['grow_name'], grow['device_name'], grow['sensors'], grow['actuators'], grow['controls']))

pHChartData.reverse()
ecChartData.reverse()
luxChartData.reverse()
tempChartData.reverse()
humidityChartData.reverse()
print range_list
return render_template('grows/grows.html',
                       username=username, current_grow=current_grow, experiment=experiment, current_device=assoc_device_name, \
                       device=device_list, grow=grows_list, my_devices=user_devices, my_grows=user_grows, pHChartData=pHChartData, \
                       ecChartData=ecChartData, luxChartData=luxChartData, tempChartData=tempChartData, \
                       humidityChartData=humidityChartData, range=range_list)

@mod_grows.route('/link/<current_grow>/<link_device>', methods=['POST'])
@login_required
def link(current_grow, link_device):
    devices = db.devices.find({'device_name':link_device})

```

```

for device in devices:
    device_id = device['device_id']
    result = db.grows.update_one(
        { "grow_name" : current_grow},
        {
            '$set': {'device_name' : link_device, 'device_id' : device_id, 'sensors': [], 'actuators' : {}, 'controls' : {"time":[], "condition" :[]}}
        },
        upsert=True
    )
    return redirect(url_for('grows.list_grow', current_grow=current_grow))

@mod_grows.route('/edit_grows/<current_grow>', methods=['POST'])
@login_required
def edit_grow(current_grow):
    grows = db.grows.find({'grow_name':current_grow})
    for grow in grows:
        device_id = grow['device_id']
        devices = db.devices.find({'device_id':device_id})
        for device in devices:
            d_actuators = device['actuators']
            d_sensors = device['sensors']
        sensors = []
        for sensor in d_sensors:
            if request.form[sensor] == 'on':
                sensors.append(sensor)

        actuators = {}
        for key in d_actuators.keys():
            if request.form[key] == "on":
                actuators[key] = d_actuators[key]
        grows = db.grows.find({'grow_name':current_grow})
        for grow in grows:
            device_id = device['device_id']
            g_controls = grow['controls']
            for time_control in g_controls["time"]:
                if time_control["actuator"] not in actuators.keys():
                    g_controls["time"].remove(time_control)
            for condition_control in g_controls["condition"]:
                if condition_control["actuator"] not in actuators.keys() or condition_control["sensor"] not in sensors:
                    g_controls["condition"].remove(condition_control)
        result = db.grows.update_one(
            { "grow_name" : current_grow},
            {
                '$set': {'sensors':sensors, 'actuators' : actuators, 'controls':g_controls}
            },
            upsert=True
        )
        return redirect(url_for('grows.list_grow', current_grow=current_grow))

@mod_grows.route('/edit_grow_controls/<current_grow>', methods=['POST'])
@login_required
def add_grow_controls(current_grow):
    username = current_user.get_id()
    existing_grow = db.grows.find({'grow_name' : current_grow})
    for grow in existing_grow:
        g_actuators = grow['actuators']
        g_controls = grow['controls']
        g_sensors = grow['sensors']

    edited_grow_controls = {"time":[], "condition" : []}
    condition_control_num = 0
    time_control_num = 0
    for key in request.form.keys():
        if key[-1] == "if_this_condition_":
            condition_control_num+=1
        elif key[-1] == "time_int_":
            time_control_num+=1
        if time_control_num == 1:
            if request.form['time_int_1'] == "":
                time_control_num = 0
        if condition_control_num == 1:
            if request.form['if_this_condition_1'] == "":
                condition_control_num = 0

    for i in range(time_control_num):
        if str(request.form['actuator_control_'+str(i+1)+'_actuator']) in g_actuators.keys():
            edited_grow_controls["time"].append({'action': "toggle",
            "actuator": request.form['actuator_control_'+str(i+1)+'_actuator'],
            "unit": request.form['time_hour_min_'+str(i+1)],
            "value": int(request.form['time_int_'+str(i+1)]),
            "start_date" : request.form['startdate_activity_'+str(i+1)],
            "finish_date" : request.form['finishdate_activity_'+str(i+1)]})

    for i in range(condition_control_num):
        if request.form['if_this_condition_'+str(i+1)] in g_sensors and \
            request.form['do_this_condition_'+str(i+1)].split("-")[0] in g_actuators.keys():
            edited_grow_controls["condition"].append({'action': (request.form['do_this_condition_'+str(i+1)].split("-")[1]),
            "actuator": request.form['do_this_condition_'+str(i+1)].split("-")[0],
            "sensor" : request.form['if_this_condition_'+str(i+1)],
            "operator" : request.form['is_condition_'+str(i+1)].split("%")[0],
            "unit": request.form['if_this_condition_'+str(i+1)],
            "value": float(request.form['to_this_condition_'+str(i+1)])})

```



```

new_grow = {"grow_name" : request.form['new_grow_name'], "plant_profile": "user defined", "experiment" : "false", "device_name" :
            "username":username, "sensors":sensors, "actuators":actuators, "controls": new_grow_controls }
db.grows.insert_one(new_grow)
return redirect(url_for('grows.list_grow', current_grow=request.form['new_grow_name']))
else:
    return redirect(url_for('grows.list_grow', current_grow=current_grow))

@mod_grows.route('/add_grows/', methods=['POST'])
@mod_grows.route('/add_grows/<num>/<first>', methods=['POST'])
@login_required
def add_grow(num=1, first=1):
    username = current_user.get_id()
    existing_grow = db.grows.find_one({'grow_name' : request.form['grow_name']})
    if not existing_grow:
        devices = db.devices.find({'device_name': request.form['device_name']})
        for device in devices:
            device_id = device['device_id']
            sensors = device['sensors']
            actuators = device['actuators']
        plant_profiles = db.plant_profiles.find({'identifier' : request.form['plant_type']})
        for profile in plant_profiles:
            if 'ph_minimum' not in profile.keys():
                ph_min = 6.75
            elif profile['ph_minimum'] is "":
                ph_min = 6.75
            else:
                ph_min = profile['ph_minimum']
            if 'ph_maximum' not in profile.keys():
                ph_max = 7.25
            elif profile['ph_maximum'] is "":
                ph_max = 7.25
            else:
                ph_max = profile['ph_maximum']
            if 'salinity_tolerance' not in profile.keys():
                ec_min = 600
                ec_max = 700
            elif profile['salinity_tolerance'] == "":
                ec_min = 600
                ec_max = 700
            elif profile['salinity_tolerance'] == "none":
                ec_min = 200
                ec_max = 300
            elif profile['salinity_tolerance'] == "low":
                ec_min = 400
                ec_max = 500
            elif profile['salinity_tolerance'] == "medium":
                ec_min = 600
                ec_max = 700
            elif profile['salinity_tolerance'] == "high":
                ec_min = 800
                ec_max = 900
            else:
                ec_min = 600
                ec_max = 700
        new_grow_controls = {"time":[], "condition" : []}
        if request.form['experiment'] == "false" and request.form['default_profile'] == "false":
            condition_control_num = 0
            time_control_num = 0

            for key in request.form.keys():
                if key[-1] == "if_this_condition_":
                    condition_control_num+=1
                elif key[-1] == "time_int_":
                    time_control_num+=1
                if time_control_num == 1:
                    if request.form['time_int_1'] == "":
                        time_control_num = 0
                if condition_control_num == 1:
                    if request.form['if_this_condition_1'] == "":
                        condition_control_num = 0

            for i in range(time_control_num):
                if str(request.form['actuator_control_'+str(i+1)+'_actuator']) in actuators.keys():
                    new_grow_controls["time"].append({"action": "toggle",
                                                         "actuator": request.form['actuator_control_'+str(i+1)+'_actuator'],
                                                         "unit": request.form['time_hour_min_'+str(i+1)],
                                                         "value": int(request.form['time_int_'+str(i+1)]),
                                                         "start_date" : request.form['startdate_activity_'+str(i+1)],
                                                         "finish_date" : request.form['finishdate_activity_'+str(i+1)]})

            for i in range(condition_control_num):
                if request.form['if_this_condition_'+str(i+1)] in sensors and \
                    request.form['do_this_condition_'+str(i+1)].split("-")[0] in actuators.keys():
                    new_grow_controls["condition"].append({"action": (request.form['do_this_condition_'+str(i+1)].split("-")[1]),
                                                             "actuator": request.form['do_this_condition_'+str(i+1)].split("-")[0],
                                                             "sensor" : request.form['if_this_condition_'+str(i+1)],
                                                             "operator" : request.form['is_condition_'+str(i+1)].split("%")[0],
                                                             "unit": request.form['if_this_condition_'+str(i+1)],
                                                             "value": float(request.form['to_this_condition_'+str(i+1)])})

```

```

if request.form['experiment'] == "false" and request.form['default_profile'] == "true":
    if "light_1" in actuators.keys():
        new_grow_controls["time"].append({"action": "toggle",
            "actuator": "light_1",
            "unit": "hours",
            "value": 12 })
    if "light_2" in actuators.keys():
        new_grow_controls["time"].append({"action": "toggle",
            "actuator": "light_2",
            "unit": "hours",
            "value": 12 })
    if "water_pump" in actuators.keys():
        new_grow_controls["time"].append({"action": "toggle",
            "actuator": "water_pump",
            "unit": "minutes",
            "value": 15 })
    if "pH" in sensors:
        if "phUpper_pump" in actuators.keys():
            new_grow_controls["condition"].append({"action": "on",
                "actuator": "phUpper_pump",
                "operator": "<",
                "unit": "pH",
                "sensor": "pH",
                "value": ph_min})
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "phUpper_pump",
                "operator": ">",
                "unit": "pH",
                "sensor": "pH",
                "value": ph_min})
        if "phDowner_pump" in actuators.keys():
            new_grow_controls["condition"].append({"action": "on",
                "actuator": "phDowner_pump",
                "operator": ">",
                "unit": "pH",
                "sensor": "pH",
                "value": ph_max})
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "phDowner_pump",
                "operator": "<",
                "unit": "pH",
                "sensor": "pH",
                "value": ph_max})
    if "EC" in sensors:
        if "nutrient_pump" in actuators.keys():
            new_grow_controls["condition"].append({"action": "on",
                "actuator": "nutrient_pump",
                "operator": "<",
                "unit": "EC",
                "sensor": "EC",
                "value": ec_min})
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "nutrient_pump",
                "operator": ">",
                "unit": "EC",
                "sensor": "EC",
                "value": ec_max})

if request.form['experiment'] == "true":
    if "light_1" in actuators.keys():
        value = int(request.form['lights'].split("H")[0])
        new_grow_controls["time"].append({"action": "on",
            "actuator": "light_1",
            "unit": "hours",
            "value": value })
    if "light_2" in actuators.keys():
        value = int(request.form['lights'].split("H")[0])
        new_grow_controls["time"].append({"action": "on",
            "actuator": "light_2",
            "unit": "hours",
            "value": value })

```

```

    if "water_pump" in actuators.keys():
        value = int(request.form['water'].split("min")[0])
        new_grow_controls["time"].append({"action": "toggle",
            "actuator": "water_pump",
            "unit": "minutes",
            "value": value })
    if "pH" in sensors:
        if "phUpper_pump" in actuators.keys():
            if request.form['pH'] == "pH_below_default":
                value_min = ph_min - .25
            elif request.form['pH'] == "pH_above_default":
                value_min = ph_min + .25
            new_grow_controls["condition"].append({"action": "on",
                "actuator": "phUpper_pump",
                "operator": "<",
                "unit": "pH",
                "sensor": "pH",
                "value": value_min })
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "phUpper_pump",
                "operator": ">",
                "unit": "pH",
                "sensor": "pH",
                "value": value_min })
        if "phDowner_pump" in actuators.keys():
            if request.form['pH'] == "pH_below_default":
                value_max = ph_max - .25
            elif request.form['pH'] == "pH_above_default":
                value_max = ph_max + .25
            new_grow_controls["condition"].append({"action": "on",
                "actuator": "phDowner_pump",
                "operator": ">",
                "unit": "pH",
                "sensor": "pH",
                "value": value_max })
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "phDowner_pump",
                "operator": "<",
                "unit": "pH",
                "sensor": "pH",
                "value": value_max })
    if "EC" in sensors:
        if "nutrient_pump" in actuators.keys():
            if request.form['ec'] == "ec_below default":
                value_min = ec_min - 100
                value_max = ec_max - 100
            elif request.form['ec'] == "ec above default":
                value_min = ec_min + 100
                value_max = ec_max + 100

            new_grow_controls["condition"].append({"action": "on",
                "actuator": "nutrient_pump",
                "operator": "<",
                "unit": "EC",
                "sensor": "EC",
                "value": value_min })
            new_grow_controls["condition"].append({"action": "off",
                "actuator": "nutrient_pump",
                "operator": ">",
                "unit": "EC",
                "sensor": "EC",
                "value": value_max })

    new_grow = {"grow_name" : request.form['grow_name'], "plant_profile": request.form['plant_type'], "experiment"
        "device_id":device_id, \
            "username":username, "sensors":sensors, "actuators":actuators, "controls": new_grow_controls }
    db.grows.insert_one(new_grow)
    return redirect(url_for('grows.list_grow', current_grow=request.form['grow_name']))
else:
    return redirect(url_for('plant_profiles.list_plant_profiles', num=num, first=first))

@mod_grows.route('/delete_grow/<grow_name>', methods=['POST'])
@login_required
def delete_grow(grow_name):
    username = current_user.get_id()

    device = db.grows.delete_one({'grow_name' : grow_name})
    return redirect(url_for('plant_profiles.list_plant_profiles'))

```

11. Plant Profiles – Views.py

```

from flask import Blueprint, render_template, request, url_for, redirect
from app import db, login_manager, pubnub
from flask.ext.login import login_required, current_user

mod_plant_profiles = Blueprint('plant_profiles', __name__)

@mod_plant_profiles.route('/plant_profiles/', methods=['GET'])
@mod_plant_profiles.route('/plant_profiles/<cur>/<first>', methods=['GET'])
# @mod_plant_profiles.route('/plant_profiles/<cur>/<first>/<shift>', methods=['GET'])
@login_required
def list_plant_profiles(cur=1, first=1):
    search = False
    num_profiles = db.plant_profiles.find().count()
    skip = (int(cur)-1) * 8
    lim = 8
    if num_profiles%lim == 0:
        pages = (num_profiles/lim)
    else:
        pages = (num_profiles/lim)+1
    device_list = []
    grows_list = []
    plant_list = []
    username = current_user.get_id()
    devices = db.devices.find({'username': current_user.get_id()})
    for device in devices:
        device_list.append((device['device_name'], device['type'], \
                             device['sensors'], device['actuators'], device['kit'], device['device_id']))
    grows = db.grows.find({'username' : current_user.get_id()})
    for grow in grows:
        grows_list.append((grow['grow_name'], grow['device_name']))
    plants = db.plant_profiles.find().skip(skip).limit(lim)
    for plant in plants:
        plant_list.append(plant)
    return render_template('plant_profiles/plant_profiles.html', username=username, my_devices=device_list,\
                           my_grows=grows_list, my_plants=plant_list, pages=pages, cur=int(cur), first=int(first), search=search)

@mod_plant_profiles.route('/plant_profiles_next/<cur>/<first>', methods=['GET'])
@login_required
def next_plant_profiles(cur=1, first=1):
    first = int(first) + 1
    cur = int(cur) + 1
    return redirect(url_for('plant_profiles.list_plant_profiles', cur=cur, first=first))

@mod_plant_profiles.route('/plant_profiles_prev/<cur>/<first>', methods=['GET'])
@login_required
def prev_plant_profiles(cur=1, first=1):
    first = int(first) - 1
    if int(cur) > first+2:
        cur = first+2
    return redirect(url_for('plant_profiles.list_plant_profiles', cur=cur, first=first))

```

```

@mod_plant_profiles.route('/plant_profiles_first', methods=['GET'])
@login_required
def first_plant_profiles():
    first=1
    cur=1
    return redirect(url_for('plant_profiles.list_plant_profiles', cur=cur, first=first))

@mod_plant_profiles.route('/plant_profiles_last', methods=['GET'])
@login_required
def last_plant_profiles():
    num_profiles = db.plant_profiles.find().count()
    lim = 8
    if num_profiles%lim == 0:
        pages = (num_profiles/lim)
    else:
        pages = (num_profiles/lim)+1
    first=pages-2
    cur=pages
    return redirect(url_for('plant_profiles.list_plant_profiles', cur=cur, first=first))

@mod_plant_profiles.route('/search_plant_profiles', methods=['GET'])
@login_required
def search_plant_profiles(cur=1, first=1, shift="no change"):
    search = True
    num_profiles = db.plant_profiles.find().count()
    skip = (int(cur)-1) * 8
    lim = 8
    if num_profiles%lim == 0:
        pages = (num_profiles/lim)
    else:
        pages = (num_profiles/lim)+1
    device_list = []
    grows_list = []
    plant_list = []
    username = current_user.get_id()
    devices = db.devices.find({'username': current_user.get_id()})
    for device in devices:
        device_list.append((device['device_name'], device['type'], \
            device['sensors'], device['actuators'], device['kit'], device['device_id']))
    grows = db.grows.find({'username' : current_user.get_id()})
    for grow in grows:
        grows_list.append((grow['grow_name'], grow['device_name']))
    plants = db.plant_profiles.find({"common_name": request.args.get('search')})
    for plant in plants:
        plant_list.append(plant)
    return render_template('plant_profiles/plant_profiles.html', username=username, my_devices=device_list,\
        my_grows=grows_list, my_plants=plant_list, pages=pages, cur=int(cur), first=int(first), search=search)

```