6-15-2017

Smart and Sustainable Aquaponics

Ryan Toal
Santa Clara University, rtoal@scu.edu

Kevin Claggett
Santa Clara University, kclaggett@scu.edu

Justin Goh
Santa Clara University, jgoh@scu.edu

SANTA CLARA UNIVERSITY
Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Ryan Toal, Kevin Claggett, Justin Goh

ENTITLED

SMART AND SUSTAINABLE AQUAPONICS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

Thesis Advisor — Date 6/15/17

Department Chair — Date 6/15/17

SMART AND SUSTAINABLE AQUAPONICS

By
Ryan Toal, Kevin Claggett, Justin Goh

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical Engineering

Santa Clara, California

2017

Smart And Sustainable Aquaponics

Ryan Toal, Kevin Claggett, Justin Goh

Department of Electrical Engineering

Santa Clara University

2017

ABSTRACT

As the global population increases, new ways of sustainable and efficient food production need to be explored in order to meet the growing demand from society. In this paper we explore the methods and functionality of an off-grid, semi-autonomous aquaponics system. This system will serve as a proof of concept for a large scale aquaponics system which can be modelled in other parts of the world where arable land is scarce. It was found that though the initial startup costs are high, it will be able to reduce the cost of food in the long run. The system implements a solar array and battery system, lighting, temperature controls and a plumbing system.

Keywords: aquaponics, hydroponics, aquaculture, solar panels, off-grid, smart, sustainable, lead-acid battery, pumps, LED, greenhouse, food production, LECA, bluegill, tilapia, crawfish, shrimp

Acknowledgements

The authors would like to thank Santa Clara's School of Engineering for providing us with the funding needed to carry out this project. We would also like to thank Professor Michael Mcelfresh for his advice and guidance throughout this project. Lastly we would also like to thank SunPower for donating solar panels to our project, without which our project would merely have been "Smart".

Contents

1.0 Introduction

1.1 Problem Statement

As the world population continues to increase, people will need to find a way to produce larger quantities of food in a sustainable fashion. Aquaponic systems show potential in creating large quantities of food using relatively little space compared to conventional agriculture. However aquaponics can be potentially labor intensive for the internal environment of an aquaponics system needs to be carefully maintained. Our solution is to design and build a smart and sustainable aquaponics system which will be able to grow a substantial amount of food in a limited space.

1.2 Project Background

What is Aquaponics? Is always the first question that comes up when discussing this project. In short, aquaponics is a combination aquaculture and hydroponics. Aquaculture is essentially just fish farming, generally but not always referring to farming in controlled environments, in contrast to farming in the ocean, or fishing wild fish. That leaves the obvious What is Hydroponics? Hydroponics is farming without soil. Instead the plants roots are grown in a nutrient rich solution providing all their water and mineral needs. These plants can be grown in environments with either no substrate, just growing out of the water pipe, or an inert substrate such as gravel. Aquaponics combines these by taking the nutrient rich solution created by the fish and using that to water the plants, which in turn filter the water so it can be returned to the fish.

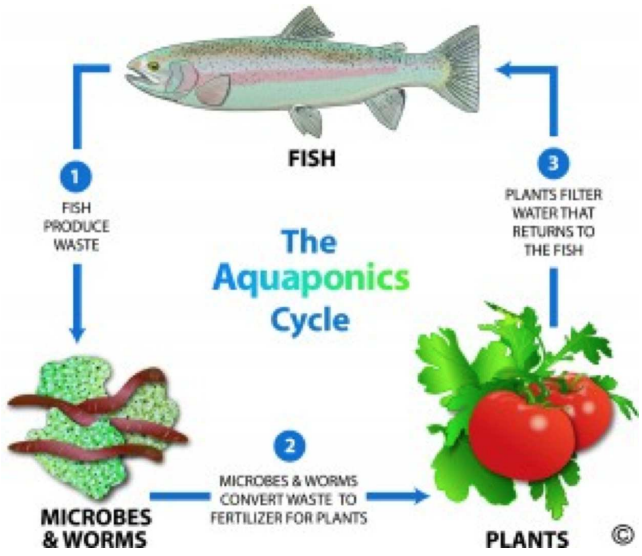


Fig 1. The basic flow of an aquaponics system (What is Aquaponics)

The figure above gives a good description of the basic symbioses that an aquaponics system creates. As you can see, the plants are fertilized by the waste of the fish, reducing the need for external sources of fertilizer for the plants. The plants act as a filter allowing the water to be recycled through the system back to the fish without need of external filtration. Thus the only inputs for the system are reduced to water, and food for the fish. This is a big savings when looking at aquaponics needs vs either hydroponics or straight aquaculture, each of which require more inputs to keep the food growing effectively.

What makes our aquaponics system Smart and Sustainable is the use of automation and an off-grid power system added to a traditional aquaponics system. In most aquaponics systems today, there is a lot of manual labor to keep the system running properly. Pumps must be manually turned on and off, temperature control is generally manual as well, and it requires a lot of human input to keep running. Our system doesn't require any human input other than to refill the consumables of the system, like fish food, and harvesting the fish / vegetables when they are ready. The other big innovation is to make this system suitable for more environments than just the industrial world. We added and off-grid power system so that our aquaponics system could be implemented in areas without reliable grid power, many of which to day need

more farming capability**.**

1.3 Motivation

When looking at where aquaponics can be most useful in helping solve current and future issues surrounding food security, two areas become immediately obvious. The first of these are places that do not have enough arable land for traditional farming to keep up with a large or growing population. The second less obvious place is urban centers where fresh produce is not available, aptly named “Food Deserts”. In both of these areas the ability to have a continuous supply of fresh fish and vegetables without the need for large land areas or good soil would be a huge boon to their overall food security.

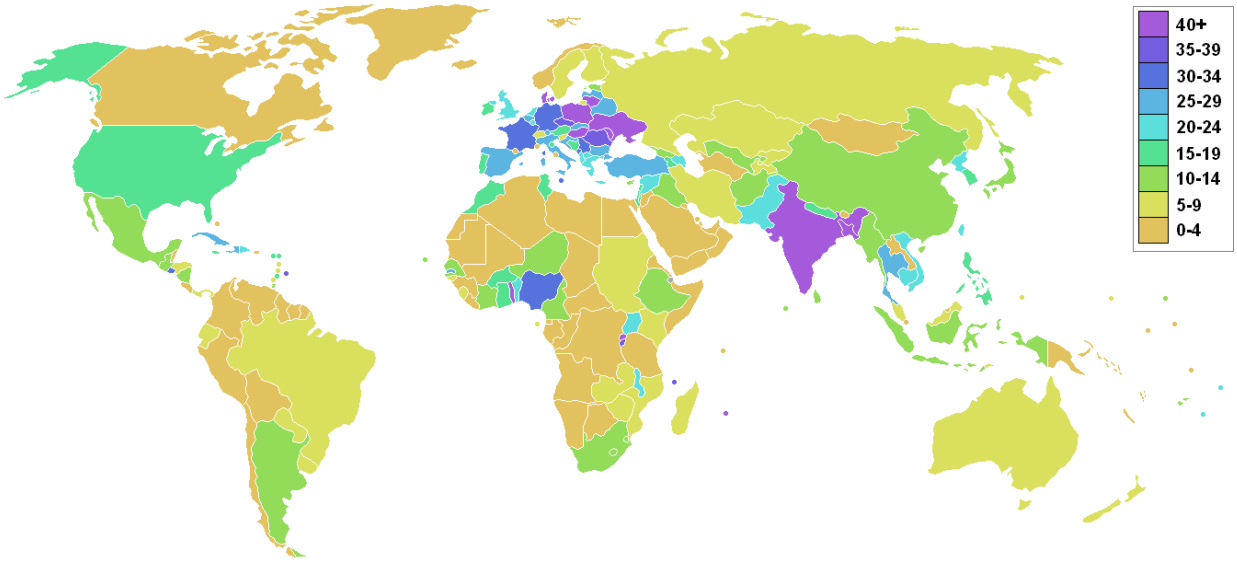


Fig 2. Percentage of Arable land by country (Roke)

There are numerous countries with small amounts of land available for farming, and many of these countries are projected to see large amounts of population growth in the future. As you can see from the map above, many of these countries are less developed so any system that is implemented to help alleviate food security issues needs to be functional in an off-grid environment. We designed the system with this constraint in mind because it allows our solution to be potentially more widely

deployed and could provide huge benefits for the areas. Aquaponics is a great solution for these countries because it doesn't require the use of any of that valuable farm land; the system can be built in otherwise wasted land currently not used by anything, effectively increasing the total available farm land for these countries.

In urban centers, the availability of fresh vegetables and fish can be very low. The FDA defines a "Food Desert" as an area where more than 33% of the people are more than 1 mile from a grocery store, and do not own cars (USDA Defines Food Deserts). Aquaponics can be a perfect solution for this environment. It provides large amounts of food for the space it takes us, and it can be installed in unused urban spaces such as rooftops. This feature allows it to be widely deployed where people need it, helping solve the Food Desert problem.

2.0 Design Considerations

2.1 Objectives

Our objective in the project was to build an aquaponics system that was self-sustaining as much as possible as well as applicable to as many environments as possible. To this end we built a system that incorporates automation for almost all the key features; the most important of these are the temperature control, water flow, and fish feeding. In order to make this system viable in a wide variety of environments we wanted to incorporate a power system that would work in and off-grid or unreliable grid setting. This system would need to be capable of powering the rest of the components for days at a time in the event of external power source loss. The last key element was the aquaponics system needs to actually keep its inhabitants alive. This means that it can't get too hot or too cold, that the water must be clean enough for the fish to survive and the plants get sufficient nutrients for the water, and light from the grow lamps to thrive.

2.2 System Level Requirements

The requirements for a functional aquaponic system that is self-regulating and sustaining over long periods are:

- Must maintain temperature within specified operational parameters in a bay area location. This range is generally going to be 70°F +/- 7.5°F based on using Bluegill fish.
- Keep the dissolved oxygen content (DO2) of the water above level fish need to survive or about 5 mg/L.
- Turn lights on and off at specified times to maintain an optimal 12 hour growth cycle for the plants and day/night cycle for the fish.
- Maintain proper water level in the aquatic tier. The fish are sensitive to water level changes so we will maintain the water level in the tanks within 15% of full.
- Dispense fish food several times daily.
- Harvest enough solar power to run the system.
- Store enough power to run the system without generation for several days.

2.3 Customer Needs

The most important need of the customers is a reliable source of fresh food. This food needs to be in an amount large enough to make a substantial portion of the daily calorie requirements for the group. The second most important need is that the

system not require a large amount of manual work. It needs to be mostly self maintaining with only occasional maintenance from the user. Lastly the system needs to be easy to use. It should be simple to gather the harvest, or make minor changes, or repairs without fear of breaking other parts of the system.

2.4 Solutions

- D02 will be maintained as long as the water continues to flow. The waterfall effect from the drop between the grow bed and the shrimp tank is enough alone to maintain D02 levels, this D02 infusion is again increased by the fall of the water over the netting of the top of the shrimp structure by breaking the water down and making it flow in smaller drips through a larger area.
- The two water pumps and overflow pipe allow for water to continue flowing through the system after any single pump failure.
- Staggered grow bed over the tanks to allow for access of fish and shrimp tanks.

3.0 Implementation

3.1 System Level Design

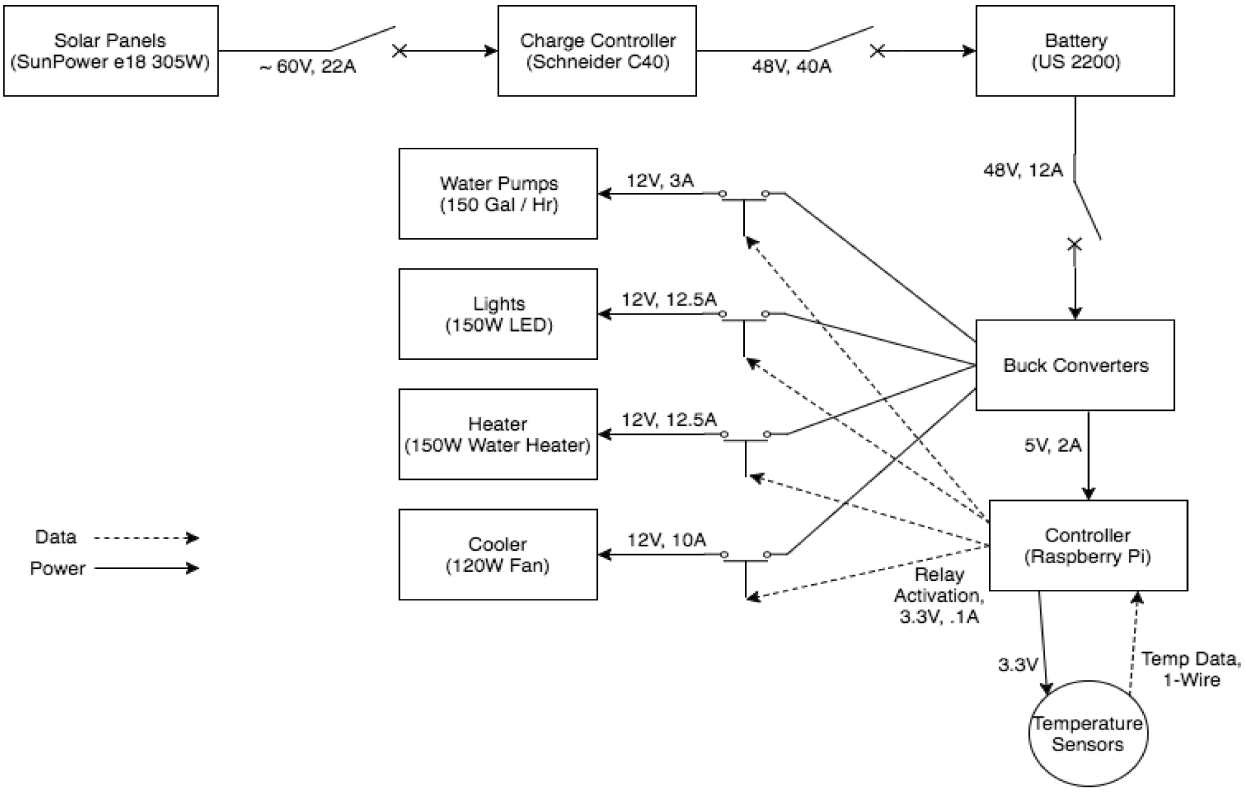


Fig 3. Design of the Power System

3.2 Software

The design goals for the control system were to be modular, lightweight, and easily configured or expanded to meet the changing needs of the underlying aquaponics system components. To meet these goals we created a system that uses an Raspberry Pi for the controller and an off-the-shelf relay board to do the actuation. The software we ran on the Raspberry Pi was written in Python so we could iterate quickly and allow new users to come up to speed easily.

When looking at what hardware to get for our control components, there were two choices that became immediately apparent. The first choice was to go with an Arduino board and use Arduino shields to get precisely whatever components you needed for the project. The second option was to go with a Raspberry Pi and have a more

general, and well rounded package to work with. We decided to go with the Raspberry Pi for a couple reasons, the first was that it has a larger user base than the Arduino projects, which means finding examples and libraries will be easier. The second reason was that in general Raspberry Pi's are programmed in Python while Arduino's generally are in C, and we had more experience with Python than C. The last reason was that many of the bundled features, like WiFi, were useful to us and allowed us to move more quickly making the Raspberry Pi a great choice.

The software design for the control system is to use a lightweight management module that runs any other configured modules included in the system. This module allows all the other modules to run once a minute, passing in and state that is needed for the module to make decisions. These modules are then responsible for all their own actions, such as turning on and off lights or pumps. After all the modules have run, the main module handles any reporting actions necessary before sleeping until the next iteration.

The software currently has 3 modules running that operate the rest of the aquaponics systems. These are all configured easily from a basic configuration module that reads a simple key value configuration file. The first module is the light module. It creates a 12 hour day night cycle by looking at the current time zone's wall clock time and turning the LED lights on from 7:30 to 19:30. The second module is the pump module, which is responsible for running the water pumps 45 min out of every hour. The last and most complicated module is the temperature module. It reads the temperature sensors in the system and decides whether to heat or cool as outlined later. This module could potentially be split into two modules if other components start needing access to the temperature data.

A module is typically operated by activating a relay to run the underlying subsystem. This starts with the software setting one of the gpio pins on the Raspberry Pi to either high or low. These pins are attached to relays on the external relay board. The pin going high activates the relay that is in series with the power supply to the underlying component. This powers on the component on; or in the case of going low, off. At this

point the underlying component should begin taking effect. With additional sensors is would be possible to then be possible to measure that effect in the next loop of the control software, and if the desired effect was not being achieved then to create an alert.

The control system as it is currently build is robust, lightweight, and easy to understand. It meets the needs of our current prototype aquaponics system well, as well as providing a solid platform to expand upon if that is needed later in a larger implementation.

3.3 Power

The basic constraints the power system was designed for was being able to continue providing power to the whole system for two days without any energy from the solar array, as well as meeting a very tight budget. Our estimated maximum power usage of 6.1 KW/h per day means that we should have around 12KW/h of battery storage in the system to meet that two day requirement. We were also donated 1200W of solar panels by SunPower, which lead to the rest of the design decisions for the power system.

The solar panels we received were members of SunPower's e18 series, indicating that they are more than 18% efficient at converting solar energy falling on their surface to usable power. Each of these panels are capable of producing around 5.5 Amps at between 50 and 65 Volts. The voltage varies because in most respects a solar panel can be considered a current source rather than a voltage source. We looked at using them in both a 2 series, 2 parallel (2S2P) configuration and a 4 parallel (4P) configuration, but ended up going with the 4P configuration because of the constraints the series panels would put on our choice of battery charge controller.

There are two different types of charge controllers available on the market currently, Pulse Width Modulated and Maximum Power Point Tracking. MPPT controllers are a newer technology that is more efficient at using the available solar energy and generally capable of handling higher solar voltages. These controllers contain DC to DC

converters rather than pulse width modulation to ensure that the batteries are being charged at the correct voltage (MPPT vs PWM). Unfortunately these newer controllers are also significantly more expensive than PWM controllers, so they weren't going to be available with our limited budget. The charge controller we ended up choosing is the Schneider C40, a PWM controller capable of outputting 40A to the batteries. This controller has a maximum input voltage of 125V which is what caused us to go with a 4P configuration for the solar panels as the 2S configuration gets up near 140V. The controller also has a selectable 12 / 24 /48 V output to the battery, in order to get the most out of solar array's 1200W we realized we were going to need to go with a 48V battery bank, which theoretically should be able to use up to 1920W more than our solar output, while if we went with a 24V battery we would only be able to use around 960W of our battery or a 21% loss at this point.

The batteries were a major expense for the project and as such we needed to find a cheap solution that would meet our high storage requirement as well as be in a 48V configuration. In a production environment there would be significant advantages to going with and AGM battery. They tend to be more robust and able to stand the potential hazards of a deployment. We went with a set of flooded lead acid batteries because they provided us with the power we needed at the lowest possible price. The batteries, 8 US Battery US-2200s, were put in an 8S configuration to reach our 48V target. They are also 232Ah each at a 20Hr discharge rate giving us a total energy storage of around 11.1 KWh. This is slightly under our absolute maximum 2 day discharge of 12.2 KWh, but within reasonable distance of that goal. The 20h discharge rate is sane because that would mean a continuous discharge at 11.6A and our maximum daily usage of 6.1KWh would lead to a continuous discharge at 5.3A.

The last component of our power system is a series of buck converters that change our battery power from 48V to 12V and 5V. These are necessary because most aquaponics components expect a 12V input. Originally the power system was designed for these but in order to be maximally efficient with our solar power and budget though this had to change to a 48V battery system. These buck converters are

typically 95% efficient and have a minimum efficiency of 92%. They provide up to 60A at 12V allowing us to provide plenty of power to the system which at most can use ~40A. The 5V converter provides up to 3A to the Raspberry Pi, which has an internal converter allowing us to run 3.3V components on this rail as well.

The power system as a whole is a compromise of budget and available components. It allows us to run the aquaponics systems with plenty of overhead for expansion as well as meeting the current goals within the cost constraints we were working with.

3.4 Thermal

Our temperature system is set up using three temperature sensors, a fan, and a water tank heater. The three temperature sensors are distributed so that a waterproofed sensor is in the fish tank and another is in the shrimp tank. The third sensor is the air sensor that sits on the breadboard of the box. The two water sensors are averaged to yield the water temperature that we use, and the sensor on the breadboard yields our used air temperature. With setpoints of 67.5°F low and 82.5°F high we use the lookup table below to set the state of our system.

	Air Low	Air Norm	Air High
Water Low	H on	H on	H on F on
Water Norm	H on	All off	F on
Water High	All off	F on	F on

Table 3.4: Temperature Lookup Table H=Heat, F=Fan

The cooling system is a 120W, 12V, 10A, 16 inch fan that runs at the height of the fish and shrimp tanks so that the air blows across the falling water from the grow bed to the shrimp tank. This creates a swamp cooler effect that traps some of the heat in the water. This increases the humidity of greenhouse which helps lower the felt

temperature of the room.

The heating system uses the water as the thermal mass and is a 150W, 12V, 12.5A water tank heater that is screwed into the side of the fish tank beneath the float valve. This is so that it will always be fully submersed in water and not burn out. This is a quick solution to heating which we used to meet deadlines of the project. To make it more energy efficient and plausible for the winter where heat usage is high and solar power generation is lower we should change the system over to a solar thermal heating system. This will involve using water as the thermal mass where we heat water using sunlight, then pump this water through copper pipes in the fish and shrimp tanks to heat the water in the tanks. This reduces the energy cost of heat from 150W for the water tank heater down to about 8W for the solar thermal water pump.

3.5 Water

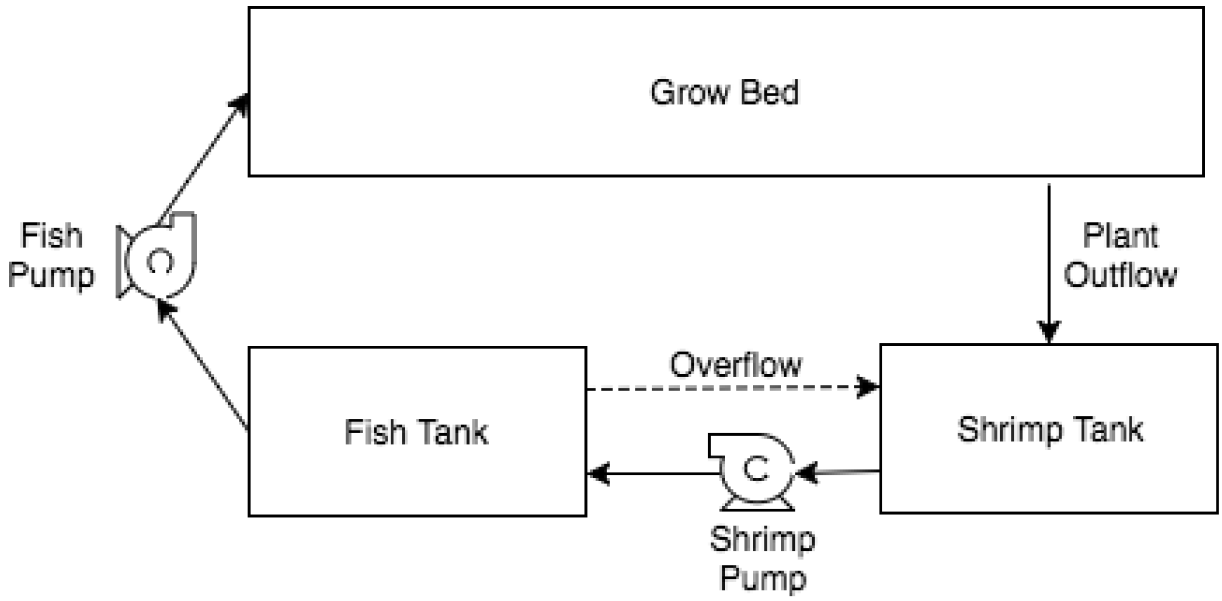


Fig 4. Design of the Plumbing System

The plumbing system is fairly simple. We use two tanks for two reasons, the first is that shrimp act as a natural filter in an aquaponics system. This allows for us to remove other forms of filters that need to be cleaned and maintained manually and replaced it with a food source. The second reason is that the fish are a volume determined animal, while the shrimp are a surface area determined animal, meaning that the shrimp do not

mind if their water level decreases as long as they're still underwater, but the fish do not want to be crowded. The two tank model allows us to pump water from the shrimp tank to the fish tank at roughly the same rate that we pump from the fish tank to the grow bed. This lowers the water in the shrimp tank while maintaining the water level in the fish tank.

Water is distributed through the grow bed by a PVC pipe array that encircles the grow bed and then crosses it at regular intervals creating separate rows of grow area. The "Fish Pump" is actually two pumps, one 180 GPH pump and one 75 GPH pump. These with the overflow pipe allow for water to continue flowing through the system even after a single pump failure. This ensures the dissolved oxygen content is high enough to keep the fish alive until the failed pump has been fixed.

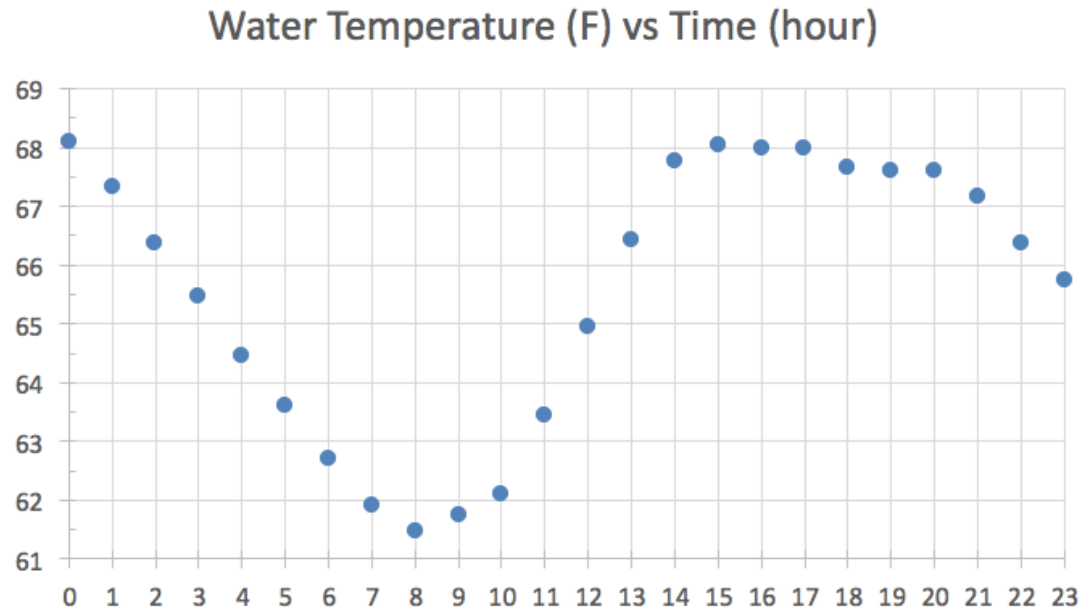
The pumps are tied to the clock on the raspberry pi and are on a fixed 45 min on/15 min off, flood/drain cycle.

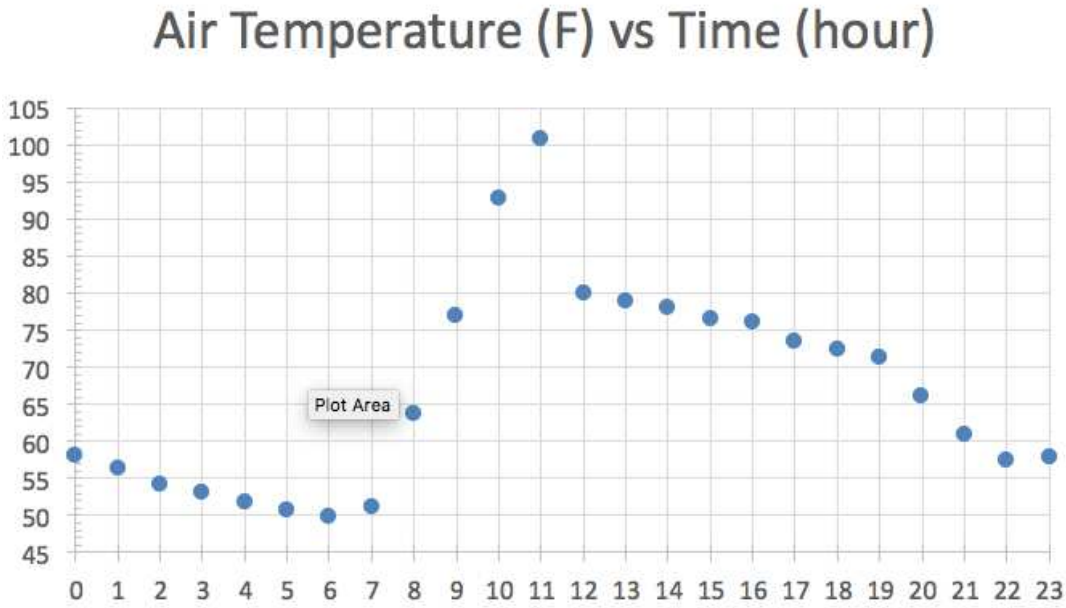
4.0 Testing and Analysis

4.1 Thermal Functionality

After assembling our system and debugging our code, we allowed the system to run for one full day. This was to test that our temperature sensors was gathering data and logging it into the Raspberry Pi. The table below show the temperature of the air and water in degrees fahrenheit, (F) and the state of our heating and cooling system throughout the day. The graphs in the following page shows the relation between air and water temperature vs time.

Time (Hour)	Temperature (F)		Temperature Control	
	Water	Air	Heating	Cooling
00:00	68.1116	58.2116	ON	OFF
01:00	67.3241	56.3	ON	OFF
02:00	66.3683	54.275	ON	OFF
03:00	65.4683	53.15	ON	OFF
04:00	64.4558	51.8	ON	OFF
05:00	63.6116	50.675	ON	OFF
06:00	62.7125	49.775	ON	OFF
07:00	61.9241	51.125	ON	OFF
08:00	61.4741	63.8366	ON	OFF
09:00	61.7558	76.8866	ON	ON
10:00	62.0933	92.8616	ON	ON
11:00	63.4433	100.85	ON	ON
12:00	64.9625	80.0366	ON	ON
13:00	66.4241	78.9116	ON	ON
14:00	67.7741	78.125	ON	ON
15:00	68.0558	76.55	OFF	ON
16:00	68	76.1	OFF	ON
17:00	68	73.625	OFF	ON
18:00	67.6616	72.3866	ON	ON
19:00	67.6058	71.2616	ON	OFF
20:00	67.6058	66.0866	ON	OFF
21:00	67.1558	61.025	ON	OFF
22:00	66.3683	57.5366	ON	OFF
23:00	65.75	57.9866	ON	OFF





4.2 Plumbing Functionality

The plumbing was tested by filling up the fish tank from the float valve and running the fish pumps on a portable 12V battery to test the functionality of the PVC halo. Upon first running we found that the pump head height was too high for the pump and the PVC diameter was too large. We then added a pump to the fish tank and remade the

PVC halo to a smaller diameter and reran the test. Once that functionality was established we ran the full system while watching for an hour to see if any imbalances occurred. It seemed that the shrimp pump was overpowering the two fish pumps, but the overflow valve could handle the difference so the system was in equilibrium. We then checked on the system every several hours, then twice a day, and finally daily. The system runs in equilibrium unless it is worked on and air gets into the pump lines causing them to lose their prime.

4.3 Component and Wiring Testing

Components were tested as they arrived across a portable 12V battery and voltmeter to check how close to their specifications they were working. They were then installed into the greenhouse and wires were run from the point of the raspberry pi to the components. They were then tested again across the portable 12V battery from the raspberry pi end to make sure the component had not been damaged during transportation or installation and that the wiring was still working. Volts and currents were again taken to check specifications.

5.0 Future Works

5.1 Solar Thermal

Considering that we had the theme of sustainability in mind, one of the initial designs for this project included the installation of a solar thermal system. The purpose of this system would be to eliminate the need to rely on a conventional water heater and instead be able to heat the system sustainably. However this aspect of the project was not able to be implemented due to resource and time constraint.

If one were to implement a solar thermal system, the first step would be to calculate the total U-value or thermal transmittance of the structure that will be fitted with the solar thermal system. After that, in order to size the furnace for the solar thermal system we need to calculate the heat transfer, q using the formula below.

$$q = [(UA) \times (Tset - Tdesign) \times (pickup\,factor)] \div \eta$$

After calculating the q value, we can then look for furnaces and select the smallest and most efficient ones that will have at least that q value.

5.2 Insulation

Proper insulation of the aquaponics shrimp structure could be done in order to minimize heat loss within the system. This in turn will enable preservation of energy by not requiring additional heating elements to maintain the internal temperature of the aquaponics system. One way this could be done is by caulking the structure. By sealing any air leaks in the structure, it will minimize the amount of heat loss during the winter months and prevent hot air from entering during the summer months. Another possible way to insulate the aquaponics structure is by rebuilding the shed with walls that have fiberglass insulation inside the wall cavity.

5.3 Painting

Painting the aquaponics shed would improve the longevity of the wooden structure. This is done because the wood will be able to last longer without rotting.

5.4 Solar Tracking

Solar tracking could be added to the system. This would increase the amount of electricity produced since the system will be able to maximise the amount of sunlight the solar panels receive. However it should be taken into consideration as to whether or not the additional electricity would make the system more efficient. Depending on the geographical region the system is located in, the addition of solar trackers may be redundant. Solar trackers are designed for warmer climates where it can maximize the amount of sunlight it receives. In harsher environments such as snowy regions, a fixed solar array may be more preferable due to the increased rigidity of a fixed solar array compared to a solar tracking system.

5.5 Alarm

A good addition to this system would be an alarm. It could either be an onboard alarm on the raspberry pi, or a better but more complicated solution would be a smartphone application that can push update the alarm. This alarm should be used for when the air temperature reaches extreme highs or lows and if the water temperature reaches more moderate highs or lows since the plants can handle a higher temperature range than the fish.

6.0 Ethical Analysis

6.1 Sustainability

Our project requires a number of different resources in order to perform as intended. We will explore the different resources required by examining the different components that are crucial to our project. The first item of concern are the solar panels. Solar panels are typically made from varying silicon compounds. The process of manufacturing solar panels can be potentially hazardous for the workers involved. This is because during the manufacturing process, one of the steps required is sawing silicon discs. This process will result in the build up of silicon dust also known as kerf. If inhaled by workers, the kerf may result in respiratory problems. Another concern is the use of silica gas during the manufacturing process. It is known to be highly combustible and may spontaneously combust, potentially harming workers. However solar panels typically have a shelf-life of about 25 to 30 years and is relatively easy to recycle if done correctly. If not disposed of appropriately the panels can potentially be detrimental to the local environment due to the exposure to the silicon compound.

Another crucial component in our project are the AGM batteries. Though it was the safest option among the type of batteries we considered, it can still be detrimental to the environment if not disposed properly. If the chemicals within the battery were to leak out somehow, this would pose a threat to the local environment. This may lead to the toxification of the environment making it uninhabitable for the local plant and animal life. For example if the chemicals were to be leaked into a local water source, the aquatic life may perish due to the increased toxicity. Even if no wildlife is in any immediate danger, the chemicals within the battery can still potentially harm a human operator.

Our system also would house aquatic life and plants as produce. The organisms housed inside our system may be considered foreign to the local surrounding environment. If these organisms were to somehow escape containment, they could interfere with the local species. The long lasting impacts could be harmful to the environment by causing an unbalance in the local ecosystem. For example if a foreign

fish were to be introduced into the local water system, it may interfere with the local food chain by either consuming the local fishes to extinction or competing with them for food sources.

6.2 Food Security

Aquaponics allows people who have traditionally not had access to large amounts of fresh vegetables and fish to get these resources with a minimum of disruption to the environment and landscape around them. The best example of the benefits aquaponics can provide in the first world is to look at the modern problem of food deserts. These are areas of inner cities where there is very little access to fresh produce, people living there must walk over a mile to get to a traditional supermarket, and do not have access to a car. In these areas, aquaponics can be deployed as small, even rooftop sized systems that will provide nearby residents with a valuable source of nutrition that does not already exist in the community. The other benefit of our system in particular is that the aquaponics system includes substantial green-energy infrastructure. This can help subsidize the cost of running not only the aquaponics system, but also of other energy needs, as it is generally sized for the worst case which will not happen very often. In the third world the target audience and problem are a little more abstract. Africa is expected to experience most of the population growth over the next several decades, but is already using most of its arable land (Cassman et al.). Even with significant efficiency improvements, it will require a massive undertaking to produce enough food to feed the expected population from local sources. This is where aquaponics can help with the problem.

Although over time the system will pay for itself, it still has a fairly significant upfront cost that could drive away potential users. One possible way to help alleviate this problem, would be to sell these systems through a Non-Profit, that could potentially front the high upfront cost in the form of a low interest loan, and then be paid back using the profits the system generates.

This system is designed to be extremely profitable for the end user. Almost all of the cost comes in the form of the design, after that upkeep should be fairly simple and low cost. The highest cost will be the cost of the food for the fish, but this price will be offset by the drop in the user's grocery bill since the price of the fish food is cheaper than the price of fish or of vegetables that the fish food produces.

The rarer and bigger parts of the upkeep come from the solar array. The batteries have an operational lifetime of around 5 years that can often be stretched out to 10 years, the solar panels have a similar lifetime. This is a costly enterprise but, again, the reduced price of the grocery bill should more than offset the cost. What we have found is that aquaponics may not be the most economically efficient means of growing food but it is the most space efficient, making it critical for places that are suffering a food crisis by optimizing their growing potential.

7.0 Bibliography

Martin K. van Ittersum, Lenny G. J. van Bussel, Joost Wolf, Patricio Grassini, Justin van Wart, Nicolas Guilpart, Lieven Claessens, Hugo de Groot, Keith Wiebe, Daniel Mason-D'Croz, Haishun Yang, Hendrik Boogaard, Pepijn A. J. van Oort, Marloes P. van Loon, Kazuki Saito, Ochieng Adimo, Samuel Adjei-Nsiah, Alhassane Agali, Abdullahi Bala, Regis Chikowo, Kayuki Kaizzi, Mamoutou Kouressy, Joachim H. J. R. Makoi, Korodjouma Ouattara, Kindie Tesfaye, and Kenneth G. Cassman
Can sub-Saharan Africa feed itself?
PNAS 2016 113: 14964-14969. <http://www.pnas.org/content/113/52/14964.full>

Figure 1: *Arable land percent world.png* by Roke~commonswiki licensed under CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)] <https://upload.wikimedia.org/wikipedia/commons/8/85/Arable_land_percent_world.png>

"BU-201a: Absorbent Glass Mat (AGM)." *Absorbent Glass Mat (AGM) Battery Information - Battery University*. N.p., n.d. Web. 5 Mar. 2017.

Gree, Adam De. "Materials Used in Solar Panels." *AZoM.com*. N.p., 07 May 2015. Web. 5 Mar. 2017.

"What is Aquaponics." The Aquaponic Source. The Aquaponics Source, 2017. Web. 12 June 2017. <https://www.theaquaponicsource.com/what-is-aquaponics/>.

"MPPT vs PWM Solar Controllers." Enerdrive Pty Ltd. Enerdrive, 26 Aug. 2015. Web. 8 June 2017. <http://www.enerdrive.com.au/mppt-vs-pwm-solar-controllers/>.

"USDA Defines Food Deserts." American Nutrition Association. American Nutrition Association, n.d. Web. 13 June 2017.

8.0 Appendices

8.1 Aquaponics Sizing

Aquaculture sizing- In order to get palatable fish you need a fish tank of at least 55 gallons. At 55 gallons you can only put one tilapia/two gallons of water. Once you reach 250 gallons you can put in a tilapia/ gallon of water. The amount of fish is the main determinate in the size of the grow beds so I have selected to go with a 55 gallon system for a better ratio of produce space and production to aquaculture space and production. A tank of 270 gallons was then selected because of its easy to find size based of its relatively standard dimensions of 4ft x 6ft x 18in.

Grow bed sizing- The system requires roughly the same volume of grow bed as you have of aquaculture tank volume once you have hit the one fish per gallon mark. 6 gallons of water roughly equals 1 cubic foot giving us 45 cubic feet of grow bed. At 1.5 ft deep this gives us 6 square feet of grow space. 3ft x 2ft x 1.5ft.

Pump sizing- The system needs to cycle the volume of the aquaculture tank every hour through the grow beds. This is done in a 45 minute period though as 15 minutes of every hour are allotted to letting the grow bed drain and dry a bit. With a 55-gallon tank this gives us a roughly 75 gallon per hour pump requirement. The system is running on 12VDC power so a 12VDC pump is needed. Pumps require head specifications and I have a 5 foot head. This requires about 2 psi of pressure in order to raise the water to that height. From this we find that we need a pump that is 75 gph with 2 psi.

Lighting- If we are using high efficiency LED lighting then we need 25 Watts per square foot of grow space. With the 25 square feet of grow space this gives us 150 W of LED lighting required.

Power Requirement- Powering the system has two different requirements. A peak hours requirement that must be supplied from the solar panels with extra that can be used to charge the batteries for the evenings. The first requirement is

Lights + Water Pump + Fan + Sensors and Other Components = 1400W

And the second is

Water Pump + Fan + Sensors and Other Components = 400W

In total it comes to about 22 kWh/day.

8.2 Budget

We initially requested $5790 worth of expenses in order to make this product operational in its original form. We received $1500 from the school. We were able to get solar panels donated by

Sun Power. We then changed the battery from AGM to flooded lead acid as it was a cheaper option. It was also cheaper to get batteries that ran at 48V rather than 12V and then convert the power back down. We will also remove the use of the DO2 sensor and change our lights out for a cheaper option.

From here we took donations from teammates and our advisor to close the gap in the budget and complete the project.

Line Item Budget:

Power	
- Solar Panels (Donated by Sun Power)	(1400)
- Solar Panel Mounts	(300)
- Battery	(1200)
- Charge Controller	(100)
- Wire	(100)
Total	(3100)
Control Systems	
- Solar Thermal Heater	(In Possession)
- Fan	(80)
- Aerator	(20)
- Temperature Sensors	(50)
- Dissolved Oxygen Sensor (Removed)	(400)
- Raspberry Pi	(40)
- Fish feeder	(40)
- Alarm	(15)
- Float valve	(10)
Total	(655)
Lights	
- Lights	(100)
- Shipping	(50)
Total	(700)
Structure and Aquaponic Table	
- Glass for Greenhouse	(In Possession)
- PVC Pipe	(50)

- Plastic fish tubs	(100)
- Caulking	(10)
- Sediment for fish tub	(40)
- Lightweight Expanded Clay Aggregate	(300)
- Plants	(50)
- Fish	(150)
- Plant Tub	(100)
- Wood	(300)
- Pumps	(50)
-Strong Ties	(50)
-Door	(100)
- Nuts and Bolts	(35)
Total	(785)
Grand Total	(5240)

8.3 Code Listing

```
aquaponics-scu/bin/main.py
==========================================
#!/usr/bin/python

import sys
from time import sleep
from pid.decorator import pidfile
import logging
from logging.handlers import RotatingFileHandler
import os
sys.path.append(os.path.abspath('..'))

import RPi.GPIO as GPIO

import SimpleConfig
import LightController
```

```
import PumpController
import TemperatureController

@pidfile()
def main():
    config =
SimpleConfig.SimpleConfig(['/home/pi/aquaponics-scu/config/aquaponics.
cfg'])
    logger = setup_logger(config)

    logger.info("Starting Aquaponics System")

    # Set GPIO mode
    GPIO.setmode(GPIO.BOARD)
    logger.debug("Setting GPIO to BOARD_MODE")

    pumps = PumpController.PumpController(config, logger)
    lights = LightController.LightController(config, logger)
    temp = TemperatureController.TemperatureController(config, logger)

    logger.info("Initialized Modules: Lights, Pumps, Temperature")
    logger.debug("Initializing Loop")

    count = 0
    while 1:
        try:
            pumps.runModule()
            sleep(1)
            lights.runModule()
            sleep(1)
            temp.runModule()
            logger.info("Completed loop: {} Sleeping 60 seconds before
next loop".format(count))
```

```
            count += 1
            sleep(60)
        except Exception as err:
            logger.error("encountered error while running: %s", err)

def setup_logger(config):
    logger = logging.getLogger("basiclogger")
    logger.setLevel(logging.INFO)

    path = config.get("logdir",
'/home/pi/aquaponics-scu/logs/aquaponics.log')
    rotating_handler = RotatingFileHandler(path, maxBytes=20000000,
backupCount=10)
    basic_formatter = logging.Formatter('%(asctime)s | %(levelname)s |
%(message)s')
    rotating_handler.setFormatter(basic_formatter)
    logger.addHandler(rotating_handler)

    return logger

if __name__ == '__main__':
    main()
```

aquaponics-scu/config/aquaponics.cfg

```
========================================
# Pump Configs
pump.on_interval.seconds=2700
pump.off_interval.seconds=900
pump.output_board_pin=7

# Light configs
lights.ontime=0730
lights.offtime=1930
```

```
lights.output_board_pin=11

# Temp Configs (in Fahrenheit)
temp.setpoint=70
temp.error.margin=2
temp.heat.output_board_pin=13
temp.cool.output_board_pin=15
temp.runpid=False
```

aquaponics-scu/lib/LightController.py

==========

```
import RPi.GPIO as GPIO
import datetime


class LightController(object):
    def __init__(self, config, logger):
        self._initialize(config, logger)

    def _initialize(self, config, logger):
        self.logger = logger
        self.onTime = config.getInt('lights.ontime', 0730)
        self.offTime = config.getInt('lights.offtime', 1930)
        self.outPin = config.getInt('lights.output_board_pin')

        if GPIO.getmode() is None:
            raise Exception("GPIO mode not initialized")

        try:
            GPIO.cleanup(self.outPin)
        except:
            pass  # need to do for reinit, so don't bother with
exceptions
```

```
        GPIO.setup(self.outPin, GPIO.OUT)
        GPIO.output(self.outPin, GPIO.LOW)
        self.lastState = 'off'

    def runModule(self):
        currTime = datetime.datetime.now()
        testTime = currTime.hour * 100 + currTime.minute

        self.logger.debug("last state of lights: %s on-time: %s
off-time: %s current-time: %s", self.lastState, self.onTime,
self.offTime, testTime)

        # FIXME: there is a shitty inherent assumption that lights
should be on during the day
        if self.lastState == 'off' and testTime > self.onTime and
testTime < self.offTime:
            GPIO.output(self.outPin, GPIO.HIGH)
            self.logger.debug("Turning Lights ON")
            self.lastState = 'on'
        elif self.lastState == 'on' and (testTime > self.offTime or
testTime < self.onTime):
            GPIO.output(self.outPin, GPIO.LOW)
            self.logger.debug("Turning Lights OFF")
            self.lastState = 'off'
```

aquaponics-scu/lib/PumpController.py

```
==========================================
import RPi.GPIO as GPIO
import time


class PumpController(object):
```

```
    def __init__(self, config, logger):
        self._initialize(config, logger)

    def _initialize(self, config, logger):
        self.logger = logger
        self.onCycle = config.getInt('pump.on_interval.seconds', 2700)
        self.offCycle = config.getInt('pump.off_interval.seconds',
900)
        self.outPin = config.getInt('pump.output_board_pin')

        self.totalCycle = self.onCycle + self.offCycle;

        if GPIO.getmode() is None:
            raise Exception("GPIO mode not initialized")
        try:
            GPIO.cleanup(self.outPin)
        except:
            pass  # need to do for reinit, so don't bother with
exceptions

        GPIO.setup(self.outPin, GPIO.OUT)
        GPIO.output(self.outPin, GPIO.LOW)

        self.lastChange = 0
        self.lastState = 'off'

    def runModule(self):
        currTime = time.time()

        self.logger.debug("Last Pump state: %s last change time: %s
current time: %s on-cycle: %s, off-cycle: %s", self.lastState,
self.lastChange, currTime, self.onCycle, self.offCycle)
```

```
        if self.lastState == 'on' and currTime - self.lastChange >
self.onCycle:
            GPIO.output(self.outPin, GPIO.LOW)
            self.logger.debug("Turning Pumps OFF")
            self.lastState = 'off'
            self.lastChange = currTime
        elif self.lastState == 'off' and currTime - self.lastChange >
self.offCycle:
            GPIO.output(self.outPin, GPIO.HIGH)
            self.logger.debug("Turning Pumps ON")
            self.lastState = 'on'
            self.lastChange = currTime
```

aquaponics-scu/lib/TemperatureController.py

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


class TemperatureController(object):
    def __init__(self, config, logger):
        # TODO: there has to be another way this is horrible
        os.system('modprobe w1-gpio')
        os.system('modprobe w1-therm')

        self._initialize(config, logger)

    def _initialize(self, config, logger):
        self.logger = logger
        self.setPoint = config.getInt('temp.setpoint', 70)
```

```
        self.errorMargin = config.getInt('temp.error.margin', 2)
        self.heatOutPin = config.getInt('temp.heat.output_board_pin')
        self.coolOutPin = config.getInt('temp.cool.output_board_pin')
        self.basedir = config.get('temp.sensor.basedir',
'/sys/bus/w1/devices/')
        self.runPID = config.getBool('temp.runpid', False)

        self.errorSum = 0
        self.maxErrorBound =
config.getInt('temp.max.integrator.value', 20)
        self.minErrorBound =
config.getInt('temp.min.integrator.value', -20)

        if GPIO.getmode() is None:
            raise Exception("GPIO mode not initialized")

        try:
            GPIO.cleanup(self.heatOutPin)
        except:
            pass  # need to do for reinit, so don't bother with
exceptions

        try:
            GPIO.cleanup(self.coolOutPin)
        except:
            pass  # need to do for reinit, so don't bother with
exceptions

        GPIO.setup(self.heatOutPin, GPIO.OUT)
        GPIO.output(self.heatOutPin, GPIO.LOW)
        GPIO.setup(self.coolOutPin, GPIO.OUT)
        GPIO.output(self.coolOutPin, GPIO.LOW)
```

```
    def runModule(self):
        waterTemp, airTemp = self.getTemp()
        self.logger.info("water temp: {} air temp:
{}".format(waterTemp, airTemp))

        water_error = self.setPoint - waterTemp
        air_error = self.setPoint - airTemp

# Maybe Someday...
#        if self.runPID:
#            error = self.getPIDValue(error)

        self.logger.debug("water-error: %s air-error: %s error-margin:
%s", water_error, air_error, self.errorMargin)

        if (abs(water_error) > self.errorMargin and water_error > 0)
or (abs(water_error) < self.errorMargin and abs(air_error) >
self.errorMargin and air_error > 0):
            self.logger.debug("turning heat ON")
            GPIO.output(self.heatOutPin, GPIO.HIGH)
        else:
            self.logger.debug("turning heat OFF")
            GPIO.output(self.heatOutPin, GPIO.LOW)

        if (abs(air_error) > self.errorMargin and air_error < 0) or
(abs(air_error) < self.errorMargin and abs(water_error) >
self.errorMargin and water_error < 0):
            self.logger.debug("turning cooling ON")
            GPIO.output(self.coolOutPin, GPIO.HIGH)
        else:
            self.logger.debug("turning cooling OFF")
            GPIO.output(self.coolOutPin, GPIO.LOW)
```

```
    def getTemp(self, device_list=None):
        total_temp_water = 0
        total_temp_air = 0
        count_water = 0
        count_air = 0

        temp_devices = device_list or glob.glob(self.basedir + '28*')
        for device in temp_devices:
            self.logger.debug("getting temp from device %s", device)

            lines = self.getRawLines(device + "/w1_slave")
            while lines[0].find('YES') == -1:
                self.logger.debug('FAILED getting good data from %s
trying again in .2 seconds', device)
                time.sleep(0.2)
                lines = self.getRawLines(device)

            equals_pos = lines[1].find('t=')
            if equals_pos != -1:
                temp_string = lines[1][equals_pos + 2:]
                temp_c = float(temp_string) / 1000.0
                temp_f = temp_c * 9.0 / 5.0 + 32.0

                # FIXME horrible horrible hack
                if "28-000008a3d594" in device:
                    self.logger.debug("temp %s from device %s was
added to air total", temp_f, device)
                    total_temp_air += temp_f
                    count_air += 1
                else:
                    self.logger.debug("temp %s from device %s was
added to water total", temp_f, device)
                    total_temp_water += temp_f
```

```
                    count_water += 1
            else:
                self.logger.debug("Couldn't get good temp from device
%s", device)

        if count_water < 1:
            count_water = 1
        if count_air < 1:
            count_air = 1

        return total_temp_water / count_water, total_temp_air /
count_air

    def getRawLines(self, filename):
        f = open(filename, 'r')
        lines = f.readlines()
        f.close()
        return lines

    def getPIDValue(self, error):
        kp = 1
        ki = .2
        # Just PI for now
        # kd = .1

        self.errorSum = self.errorSum + error

        if self.errorSum > self.maxErrorBound:
            self.errorSum = self.maxErrorBound
        elif self.errorSum < self.minErrorBound:
            self.errorSum = self.minErrorBound

        return kp * error + ki * self.errorSum
```

aquaponics-scu/lib/SimpleConfig.py

```
=======================================
class SimpleConfig(object):

    def __init__(self, file_list):
        self._generate(file_list)

    def _generate(self, file_list):
        self.data = {}

        for fname in file_list:
            try:
                f = open(fname, 'r')
                for line in f:
                    line = line.strip()
                    if not '=' in line:
                        continue

                    pieces = line.split('=', 2)
                    key = pieces[0].strip()
                    value = pieces[1].strip()
                    if len(key) < 1:
                        continue

                    self.data[key] = value or ''
            except Exception:
                pass

    def get(self, key, default=None):
        return self.data.get(key, default)

    def getInt(self, key, default=None):
```

```
    value = self.data.get(key, default)
    try:
        return int(value)
    except ValueError:
        return default

def getFloat(self, key, default=None):
    value = self.data.get(key, default)
    try:
        return float(value)
    except ValueError:
        return default

def getBool(self, key, default=None):
    value = self.data.get(key, default)
    if not value:
        return default
    elif value == 'True':
        return True
    elif value == 'False':
        return False
    else:
        return default
```