GrowingGreen

Nate Lemons, ECE, Jason Trainor, CSE, Matthew Sargeant, EE, and Austin Hiller, CSE

***Abstract*— Our senior design product, the GrowingGreen System, is an indoor greenhouse designed to grow fresh produce inside homes and apartments with little to no user experience needed. The fully automated enclosure will properly regulate all necessary growing conditions in-order to deliver high-quality and healthy vegetation for user consumption. The GrowingGreen System is built to be as energy efficient as possible and to only consume power when needed in order to cut down on environmental impact and lesson user power costs. By working in conjunction with the real world and only reacting when current environmental variables become harmful to plant growth the system will consume less energy and be able to deliver edible vegetation, even in regions without environments conducive to growing.**

I. Introduction

THE GrowingGreen System is a fully automated, energy efficient, in-house grow site with focus on supplying the grower edible vegetation with minimal effort. Our goal is to increase the availability and desire of home growing by simplifying the process through the automation of manual processes, lessening of power consumption, and use of a user console with alerts to keep growers engaged and on schedule. By growing in-house, users will decrease their environmental impact by reducing their carbon footprint and pesticidal use on plants.

A. Significance

Proximity to fresh produce is something taken for granted every day. Fresh produce is not a commodity that is readily available to everybody in the world, let alone everyone in the United States. This problem is known as the grocery gap across America and stems from low income and rural communities. In fact, low income communities have 25% fewer supermarkets, which means a 25% decrease in fresh produce. Low income neighborhoods have half as many supermarkets as the wealthiest neighborhoods, and have four times as many smaller grocery stores, which often do not stock fresh produce. This is a big health issue as well, because access to a supermarket with fresh produce is strictly correlated to healthy diet habits. In a case study in Baltimore you can see how low-income communities and food deserts correlate when viewing Figure 1 and Figure 2 below. Both low income communities and food deserts are represented in dark red.

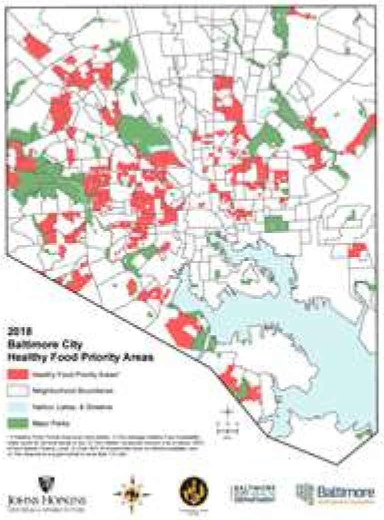


Figure 1: Baltimore City Food Map

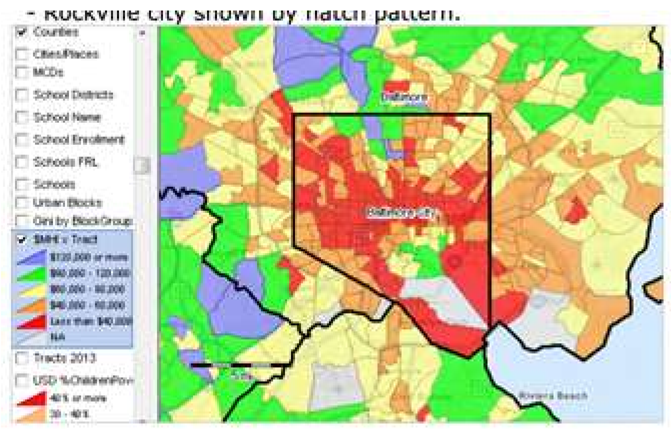


Figure 2: Baltimore City Wealth Map

In order to counter the fact that produce is not readily available everywhere, it is transported from out of the region to satisfy the demand. In fact, on average a typical meal in America will travel over 1,500 miles from farm to plate. This is a cause for concern for many reasons impacting the environment. In fact, 10 Kilocalories of fossil fuels are consumed per 1 Kilocalorie of energy we consume from food due to long distance, large-scale transportation dependency on fossil fuels [8]. Contributing to this high carbon footprint is the demand for fresh produce in climates that do not permit local growing; this causes faster, less CO2 efficient means of transport to be used to trek in produce from out of state or even from a different continent. This demand for fresh produce through transportation requires the produce to be picked unripe for transport and then chemically treated to 'ripen'. The large cost of transportation and food waste due to transport can account for over 15% of produce costs and have an even greater toll on the environment [9]. The transportation of produce has been a solution to the fact that many places around the world do not have climates or space conducive to growing locally as well as lack of knowledge and time often prevents people from growing at home, which would alleviate some of the environmental impact of our current food system.

B. Context and Existing Products

Our proposed problem is that there is currently a large tax on the environment due to our current food system. Effects of this problem have and continue to become increasingly worse as time passes. We aim to bring the growth of produce closer to the need as this will cut down immensely on the carbon footprint of transportation. There are currently few methods being used to solve the problem at a local level. This includes a recent product of the name FarmBot and a more conventional method of greenhouses. The similar problems to both of these methods is that they both require outdoor space and have large impacts from the outside environment. The conventional greenhouse also requires a large level of knowledge and time to be able to successfully produce vegetation. However, the greenhouse can manage growing in unconducive regions, unlike the FarmBot which cannot, but this comes with a very high energy cost. Our solution aims to allow growing in all regions without the high energy cost or knowledge needed to grow.

C. Societal Impacts

Throughout design and production, a common constituency was kept in mind, apartment and homeowners who do not have the suitable regions or enough time to produce their own produce. This would have a vast impact on not only homeowner's but also the environment as well. The GrowingGreen System will be able to cut down on produce cost and lessen the impact on the environment that the current industry has. The constituency we choose caused choices to be made with the goal of keeping overhead as well as running costs down. Though we do not believe any of the negative impacts to be that substantial, it is still worth mentioning that there would be a slight impact on the transportation industry. Also, the worst outcome for an owner in case of some malfunction would be plants that did not grow, meaning they would have to restart the growth cycle.

D. System Requirements and Specifications

The requirements of our product can be seen detailed in Table 1 below. The GG System will be fully automated between planting and harvesting of produce. This involves controlling the output luminosity levels, maintaining the inter-related values of humidity and temperature, and providing proper irrigation everyday. With this automation we expect to harvest ~20 oz of microgreens every 2 weeks and successfully grow year round. The final project will consume less than 200 W on average through automation of systems and will be smaller than the framing of a typical window, the interior dimensions will be 2'x1'x'3' and the exterior will be 27"x15"x39". We will also collect data from all sensors and output controls and develop trends with the data for user analysis. Both our final product and our prototype accomplish these goals in some fashion. The function of how each goal is met will be laid out in the rest of this paper.

Feature	**Specification**
Reduce power by 3x the standard	<200W consumption
Functional all year	R-value >5, 12 oz microgreens/wk
Simplify growing process	Environmental Automation
Data available to user	Logs and trends updated every 15 min
Must fit against a typical window	39x27 in.

Table 1: Requirements and Specifications

II. DESIGN

A. Overview

Our solution is to build an automated indoor greenhouse. The environment of the greenhouse is maintained using a microcontroller wired with sensors and control of various elements able to affect the conditioned space in a quantifiable way. Enacting output controls will allow the device to reduce the power needed to grow vegetables, providing a financial benefit as well as being more sustainable. Some discarded options were incorporating a soil moisture sensor, a valve for irrigation, dampers to control air flow, and watering via hydroponics, refer to Appendix A for further information.

The control board is powered via a 5.2 V DC bus and provides 3.3 V DC to power the light and float sensors, while also providing 5 V DC to the temperature/humidity sensor. These sensors input data into the control board to be analyzed by the control code. This code will make calls to a directory to verify which conditions are or are not being met. Output commands will be sent to the environmental subsystems. Environmentals consist of lighting, water pump, heating cable, and fans. The lights are powered with a 4.5/3.5/2.5 V DC bus while the pump, heater, and fans are powered using a 12 V DC bus. As these controls adjust the environment of the shelter new data will be sent to the control board to continue maintaining the desired environment.

The first specification set for this project was to reduce power compared to other options. To do this we decided to vary the lighting output as even with LEDs, lighting consumes a lot of power. The requirement to function year round meant that heating would have to be maintained during a cold season, necessitating the need for insulation to maintain the temperature with a low power heating cable to promote the desired production level. To simplify the growing process we installed an array of sensors to monitor environmental levels and inform a board mounted microcontroller of the subsystems to turn on/off. In order to record the data our system was monitoring, we built out logging functionality. We used a raspberry pi as a logging file system and a serial connection to our PCB. Our pcb used Tx/Rx transmission to send the data to our raspberry pi to get logged. This data was then used on our website and plotting features. The enclosure was built to contain the growing area within most window frames, exempting casement windows, while using typical planting trays (10”x20”), this resulted in a 27”x15”x39” structure, with interior dimensions of 24”x12x16.5" per tier.

B. Irrigation System

Our irrigation system is a tray-in-tray bottom-watering system, in which the microgreen’s root system absorbs water through the holes at the bottom of the tray they are planted in [2]. In order to implement this design, we used a float sensor [15], two 12V submersible water pumps [16], and food grade vinyl tubing [17]. We have a reservoir mounted onto the side of the unit that contains one pump for each tier and a float sensor aligned with the intake of the pump to ensure that the pump is only running when there is adequate water in the reservoir. The float sensor is wired to our output control circuit, while the pumps are powered by the 12 V DC bus and controlled with a TIP120 NPN BJT switch [10] between the pumps and ground. The pumps are triggered to begin a watering cycle once per day in the morning, and in the event that the reservoir does not have enough water to complete the watering cycle, the float sensor will trigger the system to turn the pumps off to prevent it from burning out. The user will also be notified that when the reservoir needs to be refilled via a website hosted on the Raspberry Pi, as detailed in Appendix H. The pumps are programmed to turn on for 3 seconds, which we calculated was equal to roughly a half cup of water which is enough to keep the soil moist until the next watering cycle the following day. Our reservoir currently holds enough water for roughly 12 watering cycles, which allows the system to operate completely independent of its user for almost the entirety of a microgreen’s growth cycle before the user would have to refill it. Our pumps send the water from the reservoir through food grade vinyl tubing into the enclosure, where the tubing is mounted in the bottom tray below the edge of the growing tray, to ensure that no water splashes any soil onto the plants.

Through trial and error, we were able to verify that our irrigation system design works by checking that no water was left in the bottom tray by the end of daylight hours and that the soil was still moist hours after that. This test tells us that we were not overwatering and that the plants were getting plenty of water.

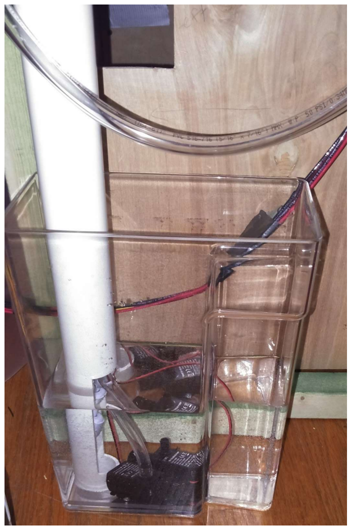


Figure 3: View of front of reservoir showing both pumps with tubing passed through PVC piping into the enclosure.

C. Lighting System

Lighting is controlled using 2 15-in LED strips per tier outputting in the blue and red spectrum. The lights are controlled using three transistor switches. When insolation is too low, a signal will be sent to operate the desired bus. See Appendix G for a full schematic of the lighting power circuit.

D. Air Flow System

Our air flow system is designed to regulate temperature and humidity by cycling the air out of the enclosure when the environment is too warm and/or humid which maintains the health of the microgreens. For each tier, we have two 12V 3” Square Axial Fans, one for air intake, and one for air outtake as shown below in Figure 5. The fans are powered by the 12 V DC bus and controlled with a TIP120[10] NPN BJT switch between the fans and ground, as detailed in Appendix G. These fans are controlled in a feedback loop which includes the temperature/humidity sensor and the heating cables. When our heat threshold of 70°F is exceeded, the outtake fan is triggered to turn on, while the intake fan acts as a louvre to allow air into the environment without disturbing the microgreens [2].

As detailed in Appendix E, we tested the function of these fans by observing our data logs to ensure that if the temperature threshold was exceeded, the air flow system would be efficient in circulating air to bring the environment back to the desired threshold. Although this was a rare occurrence due to lower than average ambient room temperatures in these winter months, we tested the air flow system’s function by reducing the temperature threshold and observing the outputs of the temperature sensor that followed. This verified that our design was functional because we first observed that the air flow system was turned on when the temperature threshold was exceeded. The fans were then triggered on and we observed the temperature sensor’s readings fall back to the desired threshold before the fans were triggered to turn off.

E. *Heating System*

The temperature of the enclosure is maintained using a freeze stop insulated heating cable [4]. The cable is powered off the 12 V DC bus and controlled with the switching circuit shown in Appendix G, a TIP120[10] NPN BJT acting as a switch between the component and ground. The cable is able to output at 5 W/ft, it is approximately 16-in currently and outputting at 6.25 W per tier.

F. *Control Code*

The final PCB code was done with C programming and is responsible for automating outputs to regulate the enclosed environment to specific setpoints. An important change between the prototype code and the final code was the change from checking the sensors every ten minutes to checking this constantly. Originally the idea was to have the program only check so often to avoid overcorrections, however it was found more efficient to implement a threshold-based functionality. The program package runs daily with different outputs calling for different code implementations based on time of day. The simplest command code is used to give the plant a specific amount of water through turning on the pump for an allotted amount of time which will be equated to a specific volume of water. The next code is responsible for controlling the temperature and humidity throughout the entire day. This code takes in readings from temperature and humidity sensors and compares the values read in against an ideal for each. If the threshold ranges are not met, then the code will call outputs to alert the state again until it can work its way back to the threshold range. A similar idea is used for the light control code, which runs starting at 6 a.m. every day for a set number of hours (default 10 hours). This code will take in light readings from both the front and back of the enclosure and calculate a general light reading. This light reading can then be used to compare against a threshold and set the lights on or off depending on whether or not the plants needs more light delivered to them. These three codes work together along with data log calls, to provide control commands to output components with the goal of influencing the environment to reach the ideals that are currently stored in a plant directory csv file. Another important aspect of the code is how expandable it is due to structs being used rather than defining pins specifically. This implementation allows for more enclosures to be added or removed with ease, simply by adding a line to the main with the enclosure name and plant of choice.

III. The Product

A. *Product Overview*

Our project, as seen in the product sketch in the below figure (Figure 4), is a fully automated greenhouse that uses light dependent resistors in order to determine the correct amount of artificial light to produce as well as regulates other environmental variables to specific setpoints. The greenhouse has two fans, two lights, one heater, one pump, one temperature and humidity sensor, one float sensor, and two light dependent resistors per tier of the enclosure. These input components communicate back to the PCB in order to determine the correct output levels for the various output devices. This relates heavily to our block diagram (Figure 5) as we have a sensor module that sends data to our control unit which then sends the correct output signals to the output module, there is also a power module that is responsible for giving the correct power levels to the entire enclosure. The correct components can also be found inside the appropriate blocks on the diagram. Figure 6 represents a single tier of our system with all of our inputs and outputs documented to visualize what each tier holds.

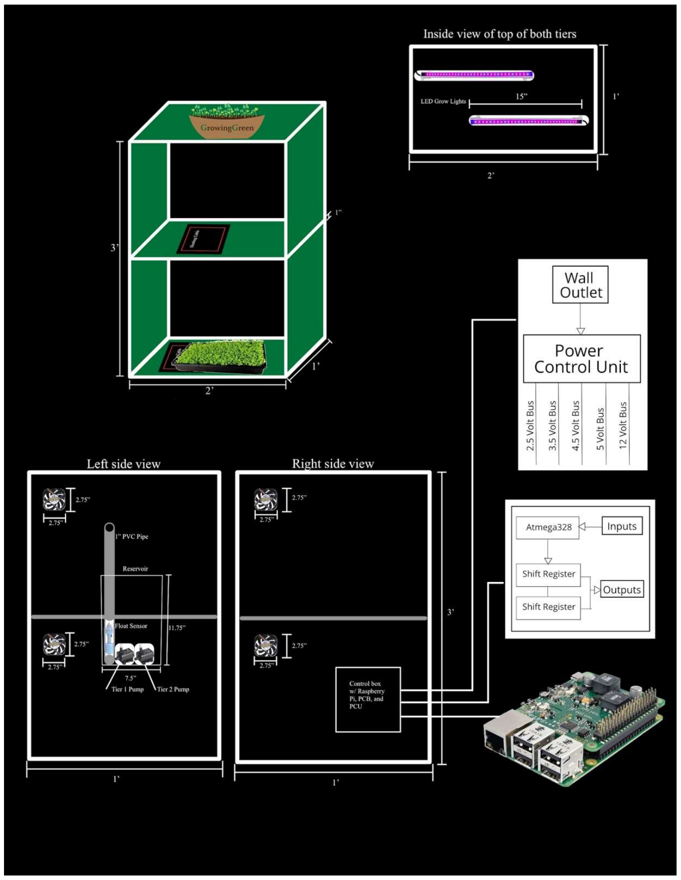


Figure 4: Product Sketch

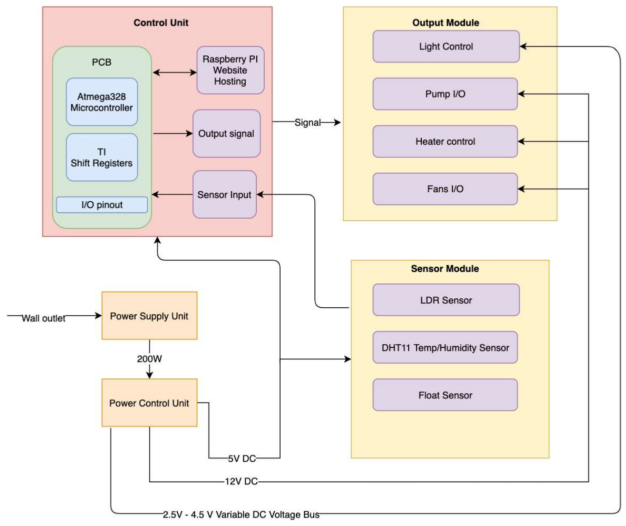


Figure 5: Block Diagram

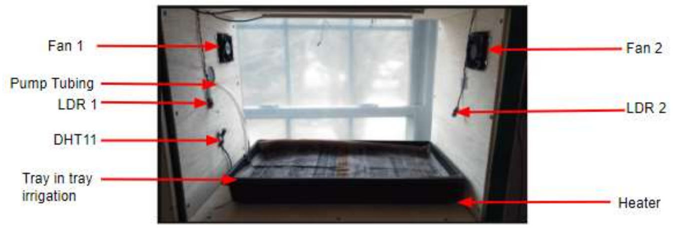


Figure 6: View of the top tier of the enclosure

B. *Electronic Hardware Component*

The design and fabrication of our electronic hardware component evolved throughout the design of our system. Our MDR prototype was functional through the use of a raspberry pi, which gave us processing power through onboard microprocessors and I/O control through GPIO pins. We needed to migrate our design onto an electronic hardware component by the time of CDR, so we decided to keep our design simple and variablistic. We decided to choose the Atmega328U microcontroller to provide analog and digital I/O, and enough processing power to meet our needs. We also knew this would be low power to meet our power requirements. We also knew we would need extra I/O pins to adequately meet our I/O demand, so we needed to add 2 TI shift registers to integrate with our Atmega328U. Lastly, we wanted to add enough female headers to support the integration to all of our output components and data inputs. We decided to use an academic license for Altium Designer to design our a printed circuit board to meet these needs. Figure 6 shows the final design of our printed circuit board. The female headers P2 and P3 correspond to the headers supporting I/O functionality, while the headers P1 correspond to the pins needed to program the Atmega328. In order to power each IC and Microcontroller, we provide 5V and GND to 2 pins to provide adequate voltage for every device. We were able to manufacture the PCB with JLCPCB and ordered 5 different PCBs to make sure we had enough for testing and failure. We hand soldered our PCB and tested with programs that signal an LED on and off at each I/O pin. After each pin was tested and our Atmega328 signature was read by our programmer, we knew that we had successfully created a PCB that we can use in our design.

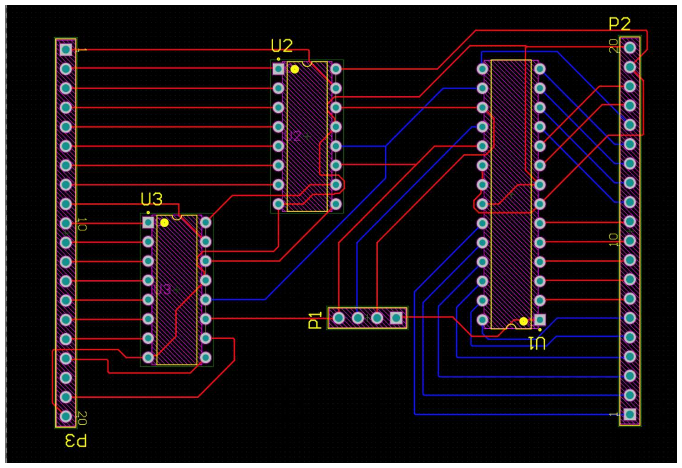


Figure 7: PCB Schematic

C. *Product Functionality*

During CDR we had a fully functioning greenhouse with a slight hiccup. Shortly before the presentation our PCB received a short leaving it ineffective. We had a working video of the enclosure, on a PCB, to show the reviewers that it was working and that it was just an unforeseen issue. However, all other components were working as expected: code functionality checked out, all output and input devices were responsive, everything was powered using our power network, and the raspberry was successfully hosting a website to track live data of the greenhouse.

D. *Product Performance*

To ensure the first specification was met, a 15V 13.4A PSU [15] was used to provide a base DC voltage limited to 200W as well as implementing variable lighting output dependent on monitored insolation. To ensure year-round growth temperature would have to be maintained at ~70 degrees Fahrenheit while near a window. To do this during winter, heating and insulation was needed, a heating cable per tier was used to heat the enclosure, however no insulation had been installed at the time of campus closure and data was unable to be collected. If one cable per tier was insufficient, there was room in the budget for 2 cables per tier. The Atmega328U was used to control automation through sensor input and subsystem control using a shift register for each tier to organize controls. Our logging and plotting was functional as we were collecting data and manipulating upon it. Our Atmega328 was successfully sending the data to our raspberry pi file system. Then custom programs were manipulating the data into relevant charts and graphs for a strong user interface. The frame of the enclosure was built to the needed specifications with the desired grow space provided. The next step in construction would have been developing a method to mount the enclosure to the window without the use of the top of a piece of furniture.

IV. Conclusion

We were able to meet most of the deliverables we set for CDR, including having a completely built enclosure with two tiers, a fully autonomous greenhouse with additional sensors and controls to support the second tier, and functional serial TX/RX data transfer for communication of data to the user. However, we were unable to complete multiple grow cycles by the time of CDR due to unexpected challenges with populating and integrating our PCB. Because of the issues we faced with our PCB, our system still partially relied on a breadboard circuit along with many circuits soldered onto perfboards. However, the system still functioned as planned and would have only required some rewiring to integrate the PCB once properly populated.

The bulk of our work to meet the specifications for FPR would have come with fixing the issues we were having with populating and integrating our PCB, but we had narrowed the cause of the problem down to the programmer we were using to populate the PCB. In addition, we were actively working on providing continuous delivery of sensor data and output control to the user via a website hosted on a Raspberry Pi, as detailed in Appendix H. The rest of the work required to finish our product would have been mostly cosmetic, as we were planning on organizing all hardware, circuitry, and extra wiring into a control box so that it was securely stored and out of harm's way.

It is very unfortunate that SDP was stopped early due to Covid-19 as our team was really invested in the project and were looking forward to finalizing the project into a finished product.

Acknowledgment

We would like to thank our Project Advisor, Professor Kundu, for all his help and guidance on our project thus far. We would also like to thank Professor Anderson, Professor Krishna, Professor Hollot, and Professor Soules for their useful feedback and evaluations during presentations and benchside meetings. Also, we would like to thank Jason D. Lanier, Angela Madeiras, and Geoffy Njue of the Stockbridge School of Agriculture here at UMass for all of their valuable advice and help thus far.

References

[1] (). How to water microgreens in 3 easy steps, a super cool hack. Available: https://www.greensguru.com/how-to-water-microgreens-in-3-easy-steps-a-super-cool-hack/.
[2] (Aug 18, 2017). Growing Microgreens 101. Available: https://www.bootstrapfarmer.com/blogs/microgreens/how-to-grow-microgreens-101.
[3] (). &nbsp;Automated Microgreen Bottom Watering System – DIY. Available: https://www.7thgenerationdesign.com/automated-microgreen- bottom-watering-system-diy/.
[4] OEMHeaters, 'FreezeStop Low VOltage Self Regulating-Heat Tape', https://oemheaters.com/images/ProductDocuments/Heaters/Freezstop%20FLV.pdf
[5] Jiangsu Changjiang Electronics Technology Co., LTD, 'Plastic-Encapsulate Transistors', S9018 datasheet, June 2011 http://vakits.com/sites/default/files/S9018%20Transistor.pdf
[6] Department of Planning. (2019). Food Environment Maps. [online] Available at: https://planning.baltimorecity.gov/baltimore-food-policy-initiative/food-environment [Accessed 19 Dec. 2019].
[7] Proximityone.com. (2019). Maryland State GIS Project. [online] Available at: http://proximityone.com/dataresources/guide/k12_md_gis_project.htm [Accessed 19 Dec. 2019].
[8] "How Far Does Your Food Travel to Get to Your Plate?," CUESA, 05-Feb-2018. [Online]. Available: https://cuesa.org/learn/how-far-does-your-food-travel-get-your-plate. [Accessed: 19-Dec-2019].
[9] "Transport Costs," The Geography of Transport Systems, 17-Aug-2019. [Online]. Available: https://transportgeography.org/?page_id=5268. [Accessed: 19-Dec-2019].
[10] Full Spectrum led Strip, TBTeek Grow Light Strip Light with Auto ON & Off Function, 3/9/12H Timer, 5 Dimmable Levels and 3 Switch Modes for Indoor Plants, Red/Blue Spectrum
[11] The Grocery Gap. (2010) The Food Trust Organization. Available at: http://thefoodtrust.org/uploads/media_items/grocerygap.ogiginal.pdf [Accessed 01-Oct-2019]
[12] Baltimore City Food Deserts Map (2015) The Baltimore Sun. Available at: http://baltimoresun.com/maryland/baltimore-city/bal-bmorefoodmap-graphic-20150610-htmlstory.html [Accessed 01-Oct-2019]
[13] Maryland State Communities and K-12 Schools GIS Project. (2015) Available at: http://proximityone.com/dataresources/guide/k12_md_gis_project.htm [Accessed 01-Oct-2019]
[14] "Open-Source CNC Farming," FarmBot. [Online]. Available: https://farm.bot/. [Accessed 08-Oct-2019]
[15] Anndason 6 Pieces Plastic PP Float Switch Fish Tank Liquid Water Level Sensor,Model: DP5200.
[16] Decdeal Submersible Water Pump DC 12V 5W Ultra-Quiet Pump for Pond, Aquarium, 280L/H Lift 300cm
[17] Learn To Brew LLC Food Grade Vinyl Tubing - 10 feet 5/16 ID - 7/16 OD
[18] D. Ibrahim, Raspberry Pi 3. 2018.
[19] AVC 707015mm DE07015B12U 12V 0.7A 4Wire 7cm cooling Fan

Appendix

A. Design Alternatives

Much of the discarded technology was related to irrigation. The two main factors that lead us to our irrigation system design were the overall health of the plant, and the ease of automation of the system. Specifically, for microgreens, irrigation can be very difficult, as one has to make sure that the soil stays damp enough between watering cycles without overwatering, while also keeping the plant and its leaves clean to ensure its health [3]. This makes conventional top-watering methods much more difficult to implement as one would have to be certain that no soil is contaminating the microgreens at any point in the growing cycle. Although this may be realistic for someone to do manually, it was not reasonable for us to implement as an automated irrigation system. Two other irrigation designs were originally looked at, hydroponics and gravity fed watering. Hydroponics was discarded to increase accessibility for the customer. Gravity fed originally incorporated the use of a valve to control the water flow; this would require the water reservoir to be above the enclosure, making it more difficult to change out and also increasing the chance a water leak could cause damage to electronic equipment located below the reservoir. The soil moisture sensor was to be used in determining when to water the plant, however scheduled watering provides adequate water supply to plants. We originally were planning on using a louvre for air intake and fan for air outtake to design our air flow system. However, we decided to discard the louvre because the fan used for air outtake was not powerful enough to open the louvre on the opposite side of the enclosure. We ended up deciding to use an intake fan to replace the louvre, which allows for the amount of air intake we originally desired.

B. Technical Standards

Due to the cancellation of final product reviews, our functioning product at the time of CDR did not meet any technical IEEE standards as it was still under final developments.

C. Testing Methods

In order to test our system as a whole, we first needed to test each part separately. We first had to test that our sensors were collecting adequate and correct data. We tested out humidity/temp sensor by placing it in a container with a working thermometer and humidity reader. We then collected values and made sure our sensor was reporting with +/- 5%. We then tested our LDR's by getting readings of resistance across the LDR in low, medium, and high light conditions. We were then able to correlate resistance values with light conditions which we use in the control of our program. We lastly tested our float sensor by placing it in a full tub of water, and empty one and found at exactly what amount of water the float sensor will trigger an empty container. We then needed to test our output controls starting with the fans. To test these, we first plugged them into 12V voltage buses to make sure they have adequate airflow. We then tested them connected to a switch controlled by the Raspberry Pi. We would trigger an on signal from the Raspberry Pi, and then check to see if the fans were on or off. We then did the same experiment for the LED. We connected the LED to a 4.5V bus and made sure it was functioning. Then we added a switch to the Raspberry Pi and sent an on/off signal to the LED to make sure it was turning off through a condition set on the Raspberry Pi. We followed the same steps for the water pump and the heating cable, both connected to 12V buses and a switch. Once we knew that our sensors were collecting proper data and we could control our outputs with the Raspberry Pi, we needed to test everything together in our main logic flow. In order to do this, we needed correct logging functionality because we could check our system working with the logs.

After our MDR, we knew that we had functioning outputs and functioning inputs, so we needed to migrate the design to a PCB and test complete functionality. With our PCB under design and fabrication, we built our PCB on a breadboard for initial testing, shown on Figure 8. We needed to build a testing center with our new microcontroller, so we decided to use an AVR programmer to download code onto the Atmega328. For this, we built a header for our programmer to download onto, and a breadboard to hold our parts. We tested our output controls through the signalling of our Atmega I/O pins and the signalling of our shift register I/O pins. We tested the signal to a BJT, which would turn a LED on and off. Once we were able to control an LED, we knew that we needed to build a functional main control code and treat the LED's as our outputs. Once we were able to build this to test our outputs, we needed to make sure that our Atmega328 was collecting data from our DHT11 and LDRs. Without a console, we decided to interface an LCD screen in order to see data collected by the Atmega, which is shown in Figure 9. Once we knew that our sensors were integrated to our microcontroller and our outputs could be controlled, we knew that we could continue. The last test we needed was to confirm data was being sent to the raspberry pi. We tested this using a serial listener on the raspberry pi and sent controlled variables from the Atmega328. Once we confirmed this was working, we needed to integrate the design to our PCB, which was fully tested before on the breadboard.

Because we tested everything on our own breadboard, the integration to PCB was easier than expected. We just used the connection test to confirm strong solders, and then began using some testing methods to control out I/O components. Then we needed to integrate out power buses, which were tested from our PCU, and connect everything together. Isolated testing made debugging much easier, for as we connected devices, we knew they worked separately.

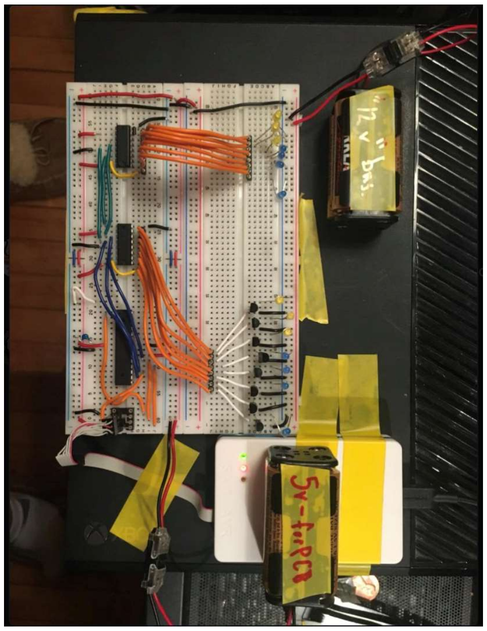
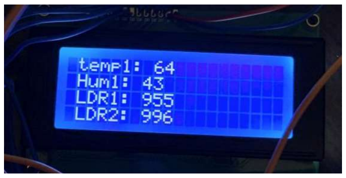


Figure 8: Breadboard Circuit



```
temp1: 64
Hum1: 43
LDR1: 955
LDR2: 996
```

Figure 9: LCD integration for data integrity

D. Team Organization

Our team has shown great chemistry and comradery when working together to complete this project. Our project manager is Austin, and everyone has taken leads on separate parts of the project, which are listed below. Though we have separate leads, we have all worked together to complete tasks. Jason and Austin worked together to send a signal from the raspberry pi to switch voltage buses that Matt created. Austin must work together with Jason to implement logging inside the main control code that Jason wrote. Our prototype is installed at Matt's home in order to obtain sufficient sunlight, and Jason has needed to implement new design features remotely which means Jason and Matt had to work closely together to make the proper changes on the prototype itself. Nate did most of the plant research and product research, so he worked together with everyone to make sure the right parts and conditions were set. We have a communication server set up in order to work together efficiently, even when we are not together in person. Communication can break down, but the way we are constantly able to come together in person to solve issues is in person.

After MDR our team came together to work on issues as a group more so as our product was coming together and needed parts to integrate. Matt worked on getting our power buses controlled to specific power requirements from a 250W PSU. Jason worked on the integration of our python code to C, and then built the testing systems on the Atmega328. Austin focused on building the PCB, so he worked closely with Jason as the breadboard controlled the design of the PCB. Austin also worked on integrating the sensors with the Atmega328 and data transmission to the raspberry pi. Matt, Jason, and Nate built the box itself, while Nate designed the irrigation and air flow systems. We all came together to integrate parts and test the systems built out together. Nate worked on soldering breakout boards to handle the BJTs and MOFSETS used for switching mechanisms, while Matt helped with the fine-tuned soldering. Nate built the website that will be hosted by the raspberry pi and worked on data collection as well. All in all, our team came together in the end to build a great product that was driven through teamwork, as we all had to work together to integrate our final parts. It is a bummer we cannot present together as a team.

Team Member	Responsibilities
Austin Hiller (Manager)	PCB, Sensor integration, Data, serial transmission
Nate Lemons	Irrigation, airflow, breakout board soldering, website
Matt Sargeant	Power buses, power management, final soldering
Jason Trainor	Control code, testing systems, Sensor integration, lead developer

Table 2: Team Member Roles and Responsibilities

E. Beyond the Classroom

For this project to be successful we had to learn a lot in relation to growing plants. For example, we learned that temperature is the most critical parameter to control microgreen growth, and that the watering of microgreens has to be done much more carefully than most plants to ensure that the plants are not contaminated in the process. We can attribute the majority of our gained knowledge to research done on the growth of microgreens specifically, but we also learned a lot thanks to communication with various professors from the Stockbridge School of Agriculture here at UMass.

F. Data

In order to show that our design is working, we needed to implement a way to track temperature, humidity, and resistance across our LDR's. We also needed to track the states of our outputs, i.e. fan, light, pump, in order to correlate our output states to our greenhouse conditions. We chose to implement a logging feature in our code that stores logs on our Raspberry Pi. Every 10 minutes our logs are updated with the current conditions of the greenhouse and the states of our outputs are logged as well. The logs are stored on our system and accessed when we want to display and plot our data or check on system functionality.

We use our logs to create plots on our own computers in order to represent our data in a visual format. Figure 9 shows our plots after 3.5 days and shows really strong data in regard to the conditions we set. The temperature plot is significant for we set a 60-degree Fahrenheit condition, and we were able to maintain that condition for almost 4 days. Our light readings show really strong data as well, for when the reading is at 10k ohms, it is nighttime, and we are able to track that through data. When the reading is below 1k ohms, it is day time and we do not need to trigger any of our own LED's, so we can determine that by placing the box on a window sill, the sun can produce enough light to grow our plants.

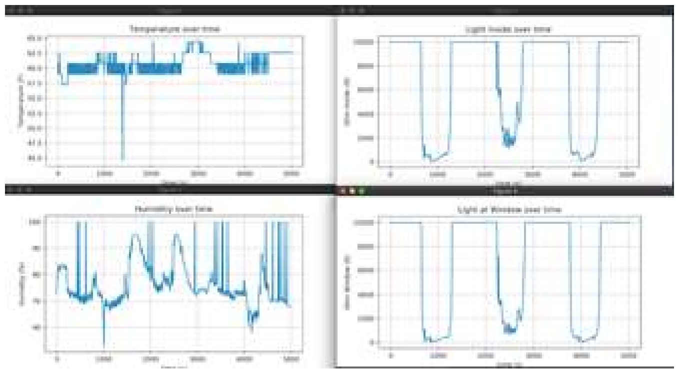


Figure 10: Data plots for temperature, humidity, and LDR values

We decided to set our temperature control to 70 degrees after 4 days to see if our heating system will be strong enough to heat the greenhouse sufficiently. In Figure 11, you can see the jump in temperature, but we were only able to reach an average of 67 degrees. This proves that we can change the conditions in the box with our variable controls, but we will need to add in more heating unit to provide sufficient heat. We are impressed with this result because it is now wintertime and we are using a container with negligible insulation. Another key data trend to notice is the increase in humidity (Figure 12), after we set the new temperature condition. Because the 70-degree conditional was never met, our system was in constant heating mode, which is heaters on, and fans off. This should theoretically increase humidity, which we were able to measure in Figure 12. This is a strong data trend line because we believe humidity will be the hardest variable to control and we were able to control and measure a change.

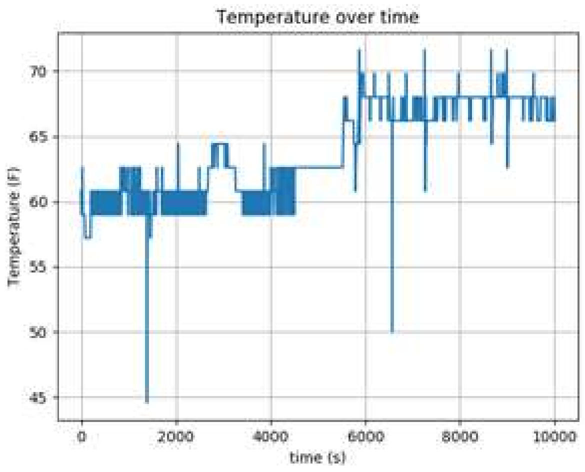


Figure 11: Temperature plot over time

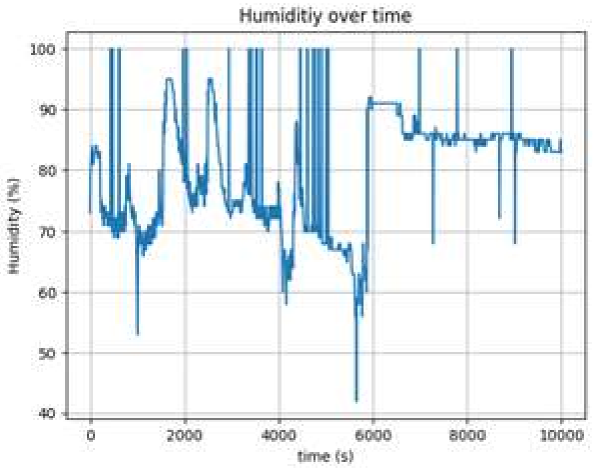


Figure 12: Humidity plot over time

The last strong set of data we collected is Figure 13, our light resistance over time. This plot shows light resistance over 12 days. When the plot is at 10k resistance it is nighttime, and when it is low, it is daytime and transitional time. During this time, our light system got stuck in an always on state, which is represented by the middle of the plot, where when it is nighttime, the ohm readings stayed around 1k. This conditional mistake was actually able to show us that our single LED is able to produce around 1k of light resistance, which is our ideal value for growing microgreens. During this time our plants got too much sunlight, but the data is able to prove to us that we can simulate enough sunlight to grow our plants.

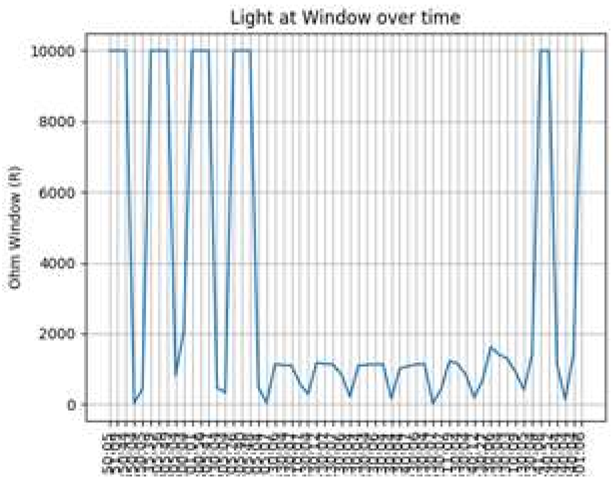


Figure 13: LDR value over time

This data shows that we were able to optimize and alter environmental data in a small plastic container. This also shows that we are able to collect, log, and plot environmental conditions of our greenhouse that we can translate to our final greenhouse. After MDR, we wanted to focus our experiments and data in controlling temperature in our greenhouse structure, as we were increasing the size of the container by 3, and not changing the size of our heating element. We believed that the heating element would be able to control temperature as the new structure had an optimized cooling system and stronger insulating elements. We decided to conduct the same experiments to control heating and cooling of our new greenhouse. We conducted the following experiments outlined in the graphs below.

- Maintain temperature at 60 degrees, starting at a higher temperature so our cooling system will be strained and utilized. Figure 14
- Maintain temperature at 65 degrees, starting at a cooler temperature so our heating system will be strained and utilized. Figure 15
- Maintain temperature at 70 degrees, starting at a cooler temperature so our heating system will be strained and utilized. Figure 16

These experiments range between 60 and 70 degrees Fahrenheit, which is the optimal temperatures that most microgreens grow at. These experiments show strong data that our system is able to control and maintain temperatures throughout the conditions we will be setting. This also tested out heating and cooling systems as we started at different temperatures than those that will be maintained.

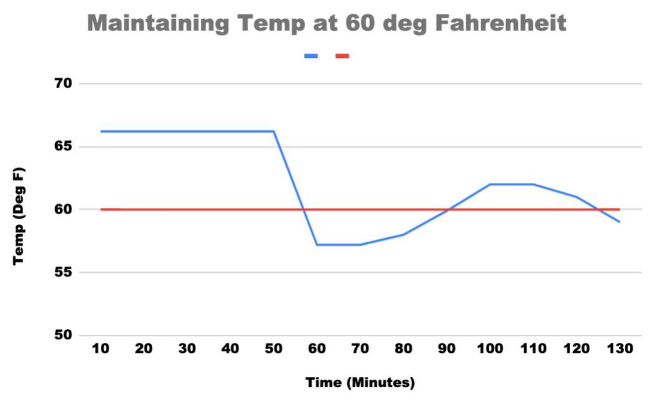


Figure 14: Maintaining temperature at 60 degrees Fahrenheit

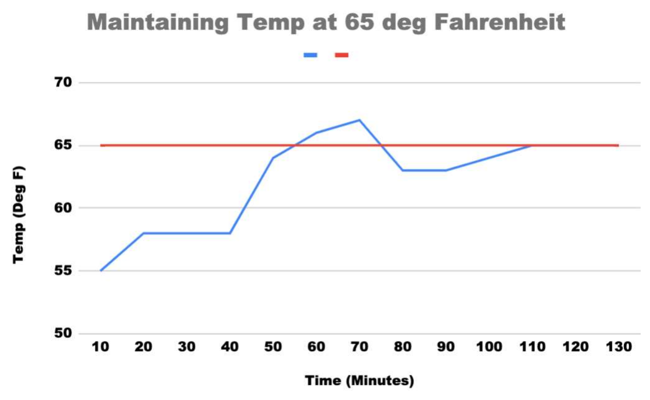


Figure 15: Maintaining temperature at 65 degrees Fahrenheit

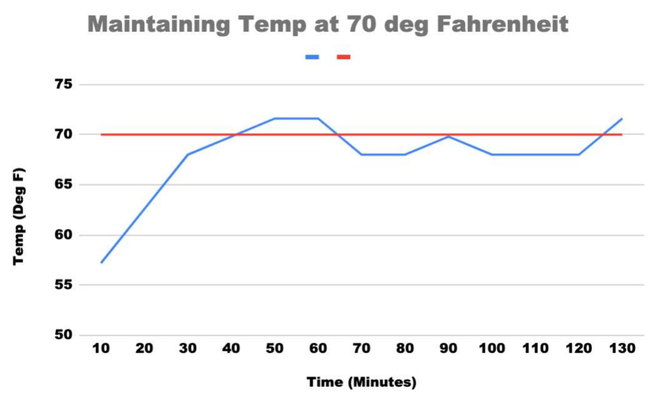


Figure 16: Maintaining temperature at 70 degrees Fahrenheit

G. Power Distribution Schematic

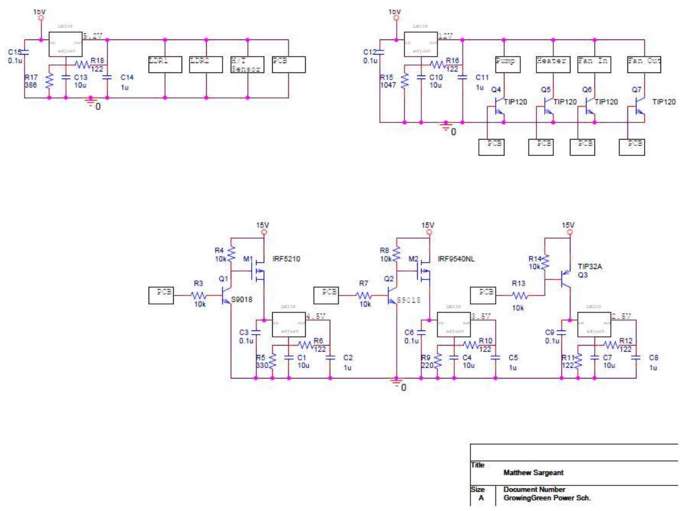
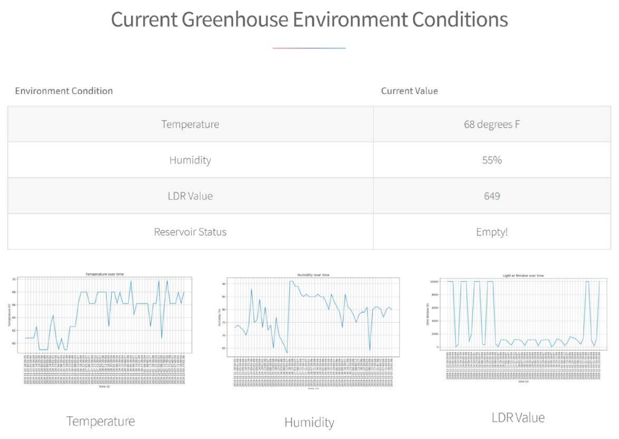


Figure 17: Power distribution schematic

H. Website Hosting and UI Design

In order to provide useful data and information about different components of our system to the user, we set up a website hosted on the Raspberry Pi to provide real time data from the greenhouse. As shown in Figure 18 below, the user interface included real time values for environmental conditions such temperature, humidity, LDR sensor output, and reservoir status. Because this data is gathered and stored on the Raspberry Pi in our system, the website also included plots for temperature, humidity, and LDR sensor data computed with real time values so the user can confirm that the system is operating as expected. This user interface would be incredibly useful because it also serves as a medium for our system to notify the user of important events that require user intervention such as when the reservoir requires a refill in order for the plants to continue to receive watering cycles. Overall, this website provided a useful and convenient way for a user to monitor the growth of their microgreens.



Current Greenhouse Environment Conditions

Environment Condition	Current Value
Temperature	68 degrees F
Humidity	55%
LDR Value	649
Reservoir Status	Empty!

Figure 18: Website user interface