GrowingGreen

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

***Abstract*—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 environment 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, reducing 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. 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 [11]. Both low income communities and food deserts are represented in dark red.

In order to counter the fact that produce is not readily available everywhere, it is transported from out of region to satisfy the demand. On average a typical meal in America will travel over 1,500 miles from farm to plate [8]. This is a cause for concern due to many reasons including a multitude of environmental concerns. In fact, 10 Kilocalories of fossil fuels are consumed per 1 Kilocalorie of energy we consume from food due to the dependency long distance, large-scale transportation has on fossil fuels [8]. Attributing 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.

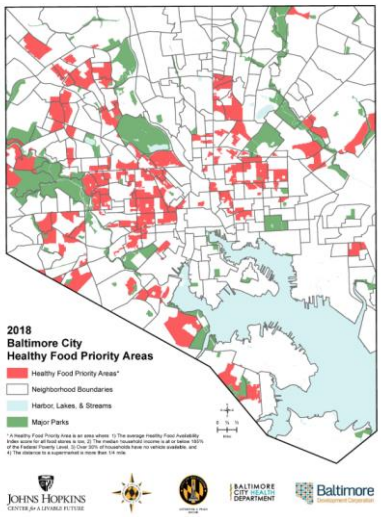


Figure 1: Baltimore City Grocery Gaps [12]

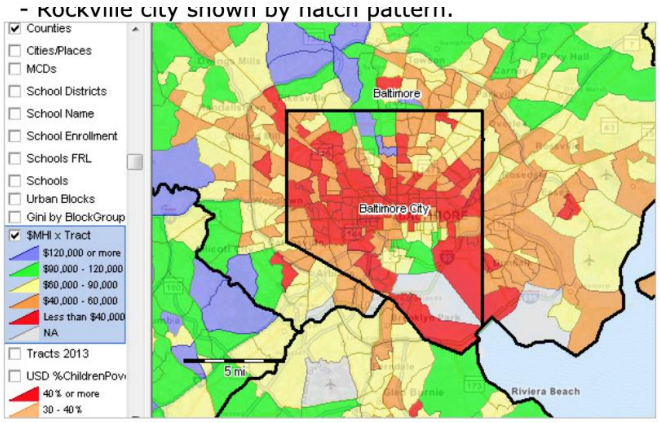


Figure 2: Baltimore City Household Income Map [12]

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

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, which automates the growing process in an outdoor flower bed, and a more conventional method of greenhouses [14]. 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.

D. Requirements Analysis 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.

Requirement	Specification	Value
Fully Automated	Automation of manual processes	Only user input is start and finish
	Light Control	Supplements light when needed
	Temperature	Maintains proper temp
	Humidity	Maintains proper humidity
	Irrigation	Waters daily
Reduce Power Consumption	Based on 300W average	Uses 200W or less
Fits Window	Typical window frame	32" x 50"
Data Collection	Sensor data	24/7 access to data & trends
	Output control data	
	Plotting	
Functional All Year	Not dependable	365 days a year
Produce Product Every Grow Cycle	Edible and nutrient rich	2 week grow cycle 20 ounces of product

Table 1: Requirements and Specifications

II. Design

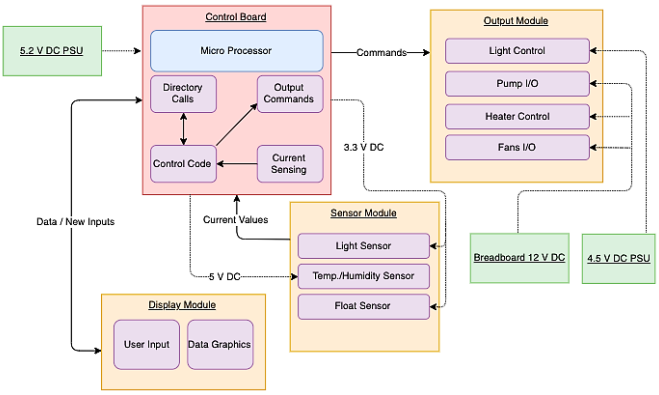


Figure 3: MDR Block Diagram

A. Overview

Our solution is to build an automated indoor greenhouse for microgreens. The purpose of choosing microgreens is the speed of growth allowing for multiple grow cycles in a short time and will provide the most benefit during testing. 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.

As shown in our Block Diagram above, the control board is powered via a 5.2 V DC USB power supply 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 then be sent to the environmental subsystems. Environmental consist of lighting, water pump, heating cable, and fans. The lights are powered off a 4.5 V DC PSU while the pump, heater, and fans are powered using the supply from a powered breadboard set to 12 V DC. 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 display module allows the user to select the type of plant being grown to provide the correct parameters needed to grow the plant, while also collecting data from the control board and outputting it into an accessible format for analysis.

B. Irrigation

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], a 12V submersible water pump [16], and roughly 3 feet of food grade vinyl tubing [17]. We have a reservoir that contains our pump and float sensor, both of which are controlled by the Raspberry Pi [18]. The float sensor is wired to a GPIO pin on the Pi and ground, while the pump is powered by the 12 V DC supply of the breadboard, and controlled with the same switching circuit as the lights, heating cable, and fans, a S9018 BJT with the base connected to the control board. The Pi controls when the pump is triggered to begin a watering cycle and turns the pump off when the float sensor is triggered as "empty." The pump is programmed by the Raspberry Pi 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 to complete roughly eight watering cycles, which allows the system to operate completely independent of its user for more than half of the microgreen's grow cycle before the user would have to refill it. For our final implementation of the irrigation system, we plan on having a reservoir that can hold enough water to complete at minimum one whole watering cycle. Our pump sends the water from the reservoir through food grade vinyl tubing to our prototype environment, where the tubing is angled into 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.

C. Lighting

Lighting is controlled with a 15-in LED strip outputting in the blue and red spectrum. The bar is capable of cycling between red, blue, and red-blue lighting by biasing a switch referenced to ground [10]. It is currently set to ground, leaving the output in the red-blue configuration for the time being. In the future work may be done to allow for spectrum control as well as controlling the lumens output. The lights are controlled using a S2018 BJT with the base being connected to an output pin of the control board [5]. When insolation is too low, a signal will be sent to bias the BJT into saturation mode, in effect closing the circuit powering the lights at 4.5 V DC.

D. Air Flow

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. We are currently using one fan for air intake, and one for air outtake, both of which are 12V 3" Square Axial Fans [19]. The fans are wired to GPIO pins and powered by the 12 V DC supply of the breadboard and controlled with the same switching circuit as the lights, pump, and heating cable, a S9018 BJT with the base connected to the control board. The Pi controls these fans 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 by the Pi and we observed the temperature sensor's readings fall back to the desired threshold before the Pi triggered the fans to turn off.

E. Heating

The temperature of the enclosure is maintained using a freeze stop insulated heating cable [4]. The cable is powered off the 12 V DC supply of the breadboard and controlled with the same switching circuit as the lights, pump, and fans, a S9018 BJT with the base connected to the control board [5]. The cable is able to output at 5 W/ft, it is approximately 16-in currently and outputting at 6.25 W. The cable has been able to maintain a temperature of 65 F during the prototype phase.

F. Code

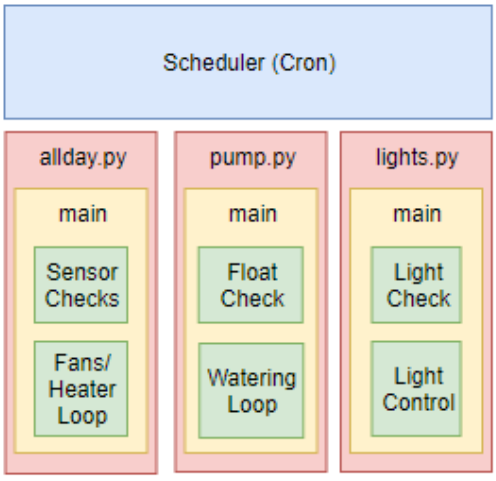


Figure 4: Code hierarchy as commanded by the Raspberry Pi

Currently we have a python driven code that is responsible for automating outputs to regulate the enclosed environment to specific setpoints. It is important to know that any code regarding sensors will react every ten minutes, this will allow for a lessened power consumption as well as a self-correcting environment. The program package runs daily with different python codes being called at various times throughout the 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 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.

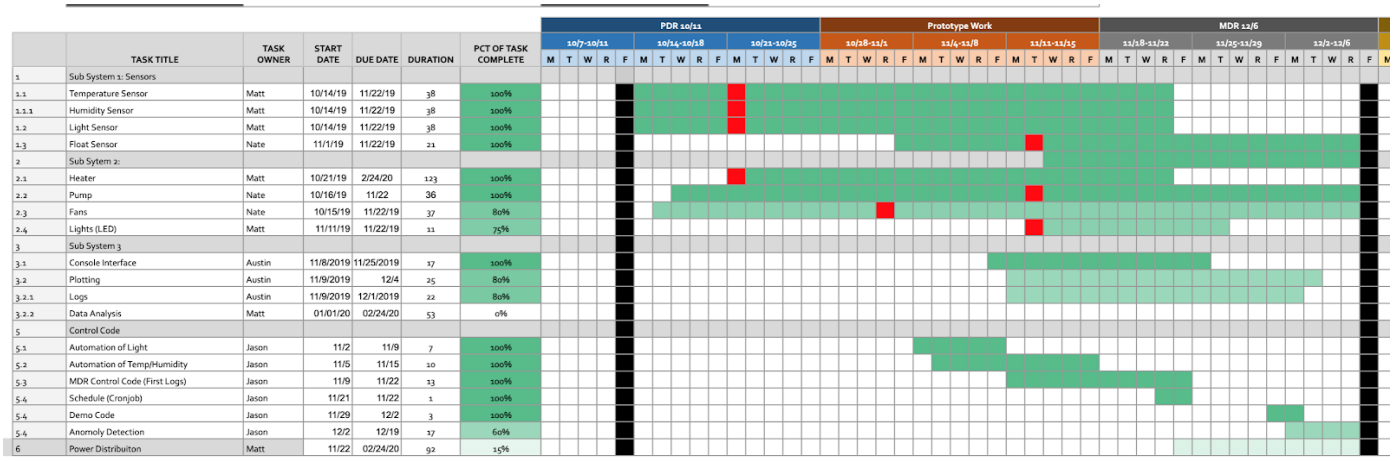
III. Project Management

MDR Deliverable	Completed	Notes
Prototyped enclosure with working sensors and output components	✓	Modified plastic container as prototyped enclosure
Control and sensor feedback controlled through raspberry pi	✓	Sensor and output control successful
Automation of manual Processes	✓	Fully automated
Successful grow cycle	✓	One successful 14 day grow cycle completed
Data available for manipulation and plotting	✓	Logs available on raspberry pi

Figure 5: MDR Deliverables

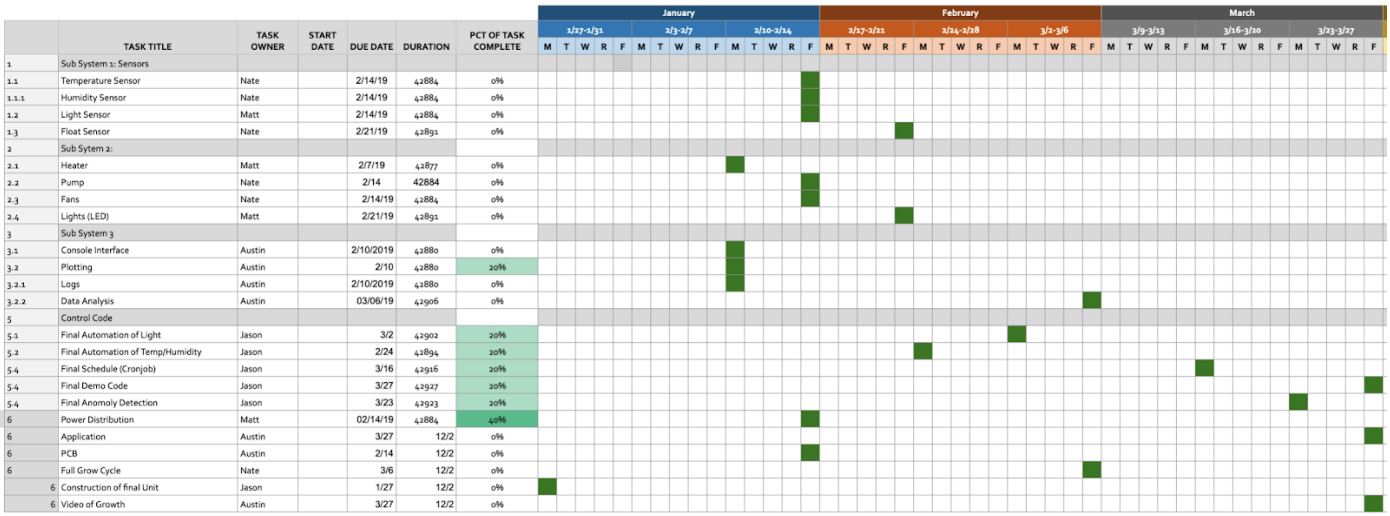
As outlined above in Figure 5, we made significant progress in our design and were able to complete and automate a functional prototype, and therefore successfully completed our MDR deliverables. As a team we were able to meet our goals, which will only help us in our transition to a final product with PCB control.

We were able to stick to our goals which is fully outlined and documented in our MDR Gannt chart which is shown below in Figure 6. In order to stick to our timeline and produce our final product we will stick to the schedule outlined in our future plans Gannt chart shown below in Figure 7.



	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Sub System 1: Sensors					
1.1	Temperature Sensor	Matt	10/14/19	11/22/19	38	100%
1.1.1	Humidity Sensor	Matt	10/14/19	11/22/19	38	100%
1.2	Light Sensor	Matt	10/14/19	11/22/19	38	100%
1.3	Float Sensor	Nate	11/1/19	11/22/19	21	100%
2	Sub Sytem 2:					
2.1	Heater	Matt	10/21/19	2/24/20	123	100%
2.2	Pump	Nate	10/16/19	11/22	36	100%
2.3	Fans	Nate	10/15/19	11/22/19	37	80%
2.4	Lights (LED)	Matt	11/11/19	11/22/19	11	75%
3	Sub System 3					
3.1	Console Interface	Austin	11/8/2019	11/25/2019	17	100%
3.2	Plotting	Austin	11/9/2019	12/4	25	80%
3.2.1	Logs	Austin	11/9/2019	12/1/2019	22	80%
3.2.2	Data Analysis	Matt	01/01/20	02/24/20	53	0%
5	Control Code					
5.1	Automation of Light	Jason	11/2	11/9	7	100%
5.2	Automation of Temp/Humidity	Jason	11/5	11/15	10	100%
5.3	MDR Control Code (First Logs)	Jason	11/9	11/22	13	100%
5.4	Schedule (Cronjob)	Jason	11/21	11/22	1	100%
5.4	Demo Code	Jason	11/29	12/2	3	100%
5.4	Anomoly Detection	Jason	12/2	12/19	17	60%
6	Power Distribuiton	Matt	11/22	02/24/20	92	15%

Figure 6: Gannt Chart for Fall Semester



	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Sub System 1: Sensors					
1.1	Temperature Sensor	Nate		2/14/19	42884	0%
1.1.1	Humidity Sensor	Nate		2/14/19	42884	0%
1.2	Light Sensor	Matt		2/14/19	42884	0%
1.3	Float Sensor	Nate		2/21/19	42891	0%
2	Sub Sytem 2:					
2.1	Heater	Matt		2/7/19	42877	0%
2.2	Pump	Nate		2/14	42884	0%
2.3	Fans	Nate		2/14/19	42884	0%
2.4	Lights (LED)	Matt		2/21/19	42891	0%
3	Sub System 3					
3.1	Console Interface	Austin		2/10/2019	42880	0%
3.2	Plotting	Austin		2/10	42880	20%
3.2.1	Logs	Austin		2/10/2019	42880	0%
3.2.2	Data Analysis	Austin		03/06/19	42906	0%
5	Control Code					
5.1	Final Automation of Light	Jason		3/2	42902	20%
5.2	Final Automation of Temp/Humidity	Jason		2/24	42894	20%
5.4	Final Schedule (Cronjob)	Jason		3/16	42916	20%
5.4	Final Demo Code	Jason		3/27	42927	20%
5.4	Final Anomoly Detection	Jason		3/23	42923	20%
6	Power Distribution	Matt		02/14/19	42884	40%
6	Application	Austin		3/27	12/2	0%
6	PCB	Austin		2/14	12/2	0%
6	Full Grow Cycle	Nate		3/6	12/2	0%
6	Construction of final Unit	Jason		1/27	12/2	0%
6	Video of Growth	Austin		3/27	12/2	0%

Figure 7: Gannt Chart for Spring Semester

IV. Conclusion

We are excited to have a fully operational and functional MDR prototype at this time in our design process. We worked together as a functioning team to bring our design to life. We all took specific leads on the design process and worked together integrating the systems together.

Our MDR prototype consists of a single tiered greenhouse with functioning automated control through a continuous feedback. The entire grow and life cycle of our plant is automated, so the user does not need to intervene on any process. We also integrated logging and plotting features to ensure our greenhouse is correctly automating our outputs.

We plan to move our greenhouse to a 2-tiered greenhouse we assemble ourselves. We plan to integrate insulated plastic sides and create the greenhouse to elegantly house our PCB, sensors, and outputs. We believe the integration to a PCB will be a difficult step for us because we are currently fully operational from a development board with built in programming libraries to increase functionality. We will need to develop our control code in C, which is a transfer from python, and sync our control code onto a microprocessor.

We are currently in the development stage of our application and are researching the idea of running our system through the cloud for data storage and to reduce issues when we develop system updates. If we move our storage and updates to the cloud, our application will also be able to access system data from anywhere, rather than from in range of Bluetooth, which was our first proposed design. This design process and research should conclude before February so we can integrate this design quickly.

Lastly, we plan to integrate an LED dimming feature so we can accurately produce the correct sunlight, whether natural or not, to both tiers of our system. This will include a specific power regulation system that we will need to build so all of our sensors and outputs are receiving the correct power needed. We are 80% done with this system and are in our final testing phase.

Next semester we expect to face many challenges, but we believe that after what we learned this semester, we will be able to handle them as a team and produce a robust and interesting final design.

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. 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 (Light Dependent Resistor) by taking 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. Below are 2 examples of our logging working, which in turn shows our system working as a whole.

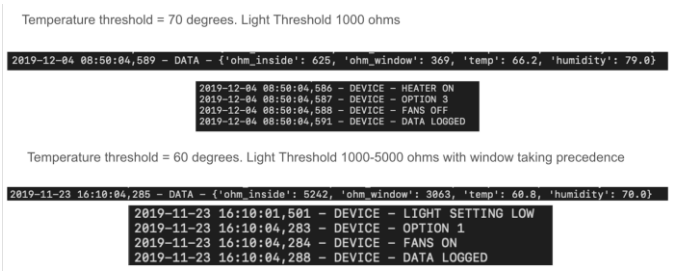


Figure 8: Threshold Testing Results

Above, Figure 8 shows two different scenarios and the outputs working correctly in comparison to the threshold values set. We would also make sure that the devices are actually on and that the logging is not reporting erroneous data. Our testing methods were simple when testing each unit and made simple through our logging feature. We are constantly testing for our greenhouse is always on, and we can make sure our greenhouse is functioning through our logs.

C. 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.

Team Member	Lead
Austin	Display Unit, Data, PCB, Application,
Nate	Humidity Sensor, Temp Sensor, Float Sensor, Fans, Pump, Application
Jason	Construction, Control code
Matt	LDR Sensor, LED, Heater, Power Distribution

Table 2: Team Member Roles

D. Beyond the Classroom

A lot of information had to be acquired in relation to growing plants. For this project to be successful we had to learn about watering for example. Learning that temperature is the most critical parameter to control in microgreen growth was very important to create the proper environment in the enclosure.

E. 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.

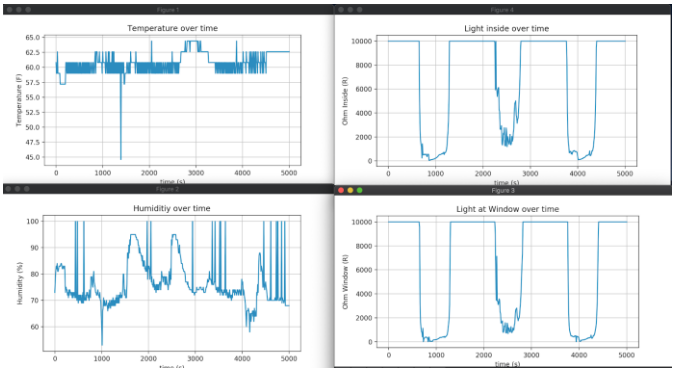


Figure 9: Sensor Data Over Time

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 10, 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 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 is shown in Figure 11. 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.

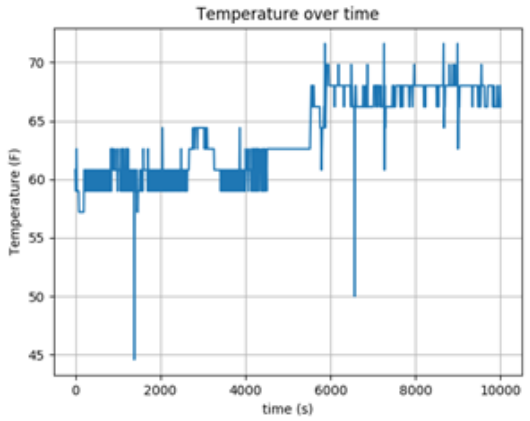


Figure 10: Temperature Sensor Data Over Time

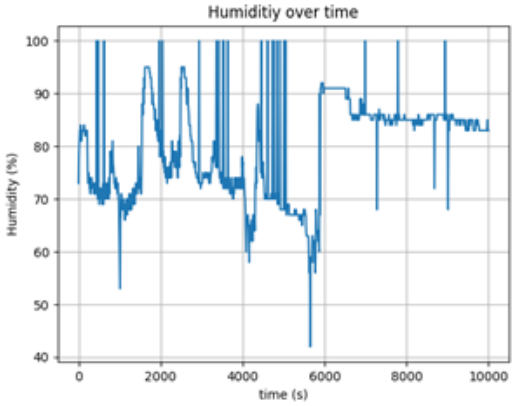


Figure 11: Humidity Sensor Data Over Time

The last strong set of data we collected is Figure 12, 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.

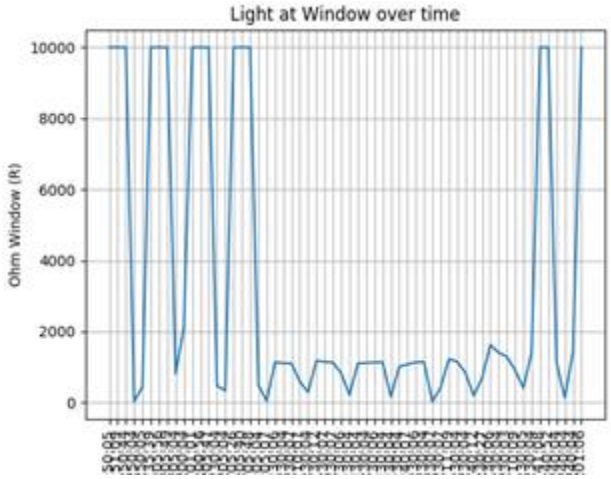


Figure 12: Light Received at Window Over Time

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] (). 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