Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Greenhouse

Submitted to

Ned Utzig
(978)895-5339
ned@utzig.com

by

Team 13
EcoHome

Team Members

Olivia Dorencz odorencz@bu.edu
Hok Yin Shum hyshum@bu.edu
Laura Morgan lamorgan@bu.edu
Kiahna Tucker kiahnat@bu.edu
Qian Zhang zhang1@bu.edu

Submitted: Apr 11, 2019

EcoHome

Table of Contents

Executive Summary

Our team aims to create a solution that will allow users to store their plants outdoors during the winter without damaging the plants. The final deliverable we have devised is a greenhouse that is able to withstand cold temperatures, rain, snow, and wind all while maintaining an internal environment that allows plants to grow and thrive. To do this, we are equipping the greenhouse with various sensors to monitor the internal temperature, light, soil moisture, and humidity. A heating system will also be included in the greenhouse to keep the temperature at an acceptable level. All of the sensor data will then be sent to a web application that will allow the user to view the real-time sensor readings. The web application will also allow the user to adjust the temperature within the greenhouse. If any of the sensor readings falls above or below the suitable levels, the user will be notified. In order to quickly access the plants inside the greenhouse, an easy to open door will be built into the structure to allow the user to water plants and provide

any other maintenance. The greenhouse is intended to make storing and growing plants in the winter easier for anyone that lives in a harsh winter climate.

1 Introduction

Cold New England winters present a unique challenge for home gardeners. Many plants cannot survive the cold, wind, and snow if left to grow outside. Yet bringing many plants inside reduces their ability to receive natural light, stunting their growth. While large scale greenhouses can provide an alternative solution, they are often far too expensive for a casual home gardener to consider. Over the course of this project, we have developed a small scale greenhouse with heating technology which allows the user to grow plants outdoors all year round at a significantly lower cost. It also gives the user the ability to monitor various conditions within the greenhouse in order to ensure that their plants have optimal growing conditions.

Through our web application, the user is able to view current and past data retrieved from the greenhouse. The measurements provided are temperature inside and outside the greenhouse in degrees Fahrenheit, light level in lux, air humidity in percent relative humidity, soil moisture in percent relative humidity, and to-date power consumption in kWh. The user can measure the soil moisture of up to three different plants.

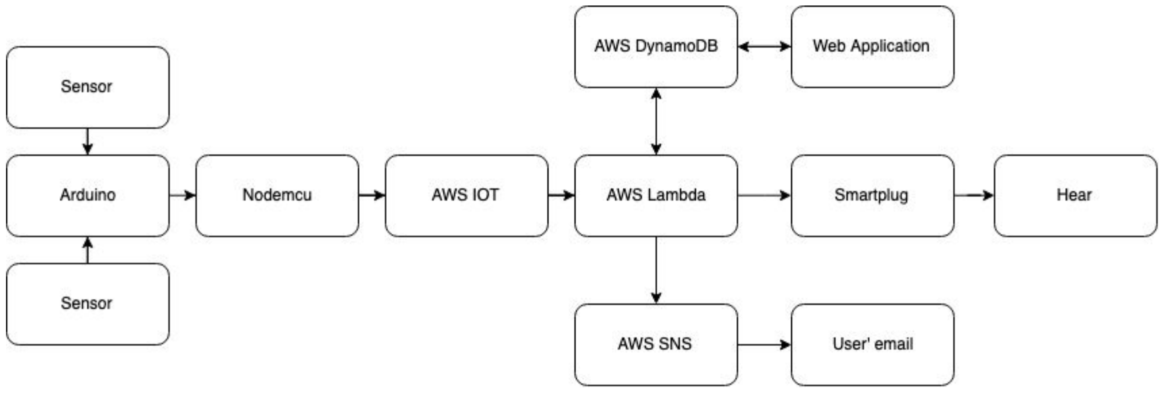
A heater is activated when the temperature inside the greenhouse drops below a certain threshold, which can be specified by the user via the web application. This event will also trigger an email notification to the user with all current measurements inside the greenhouse. The user also has the option to receive periodic emails with the measurements inside the greenhouse and has the ability to specify how frequent these notifications are sent.

The user should take care not to pour water on the central electronics system when watering plants. A hook mounted on the PVC pipe allows the central electronics system to be moved to any of the four supporting beams of the structure, so the user can customize where the electronics are housed and move it away from regions they frequently access. Water getting on the prongs of the three soil moisture sensors does not present a safety or quality concern.

Section two of this manual describes the overall system, including depictions of the UI of the web application and sketches of the final design of the greenhouse. It also provides information on how to first set up your system. Section three describes the system's normal and abnormal operation. Section four describes the technical components of the greenhouse including the development of the backend server and how data is transferred. Section five contains an itemized cost breakdown for production of the greenhouse.

2 System Overview and Installation

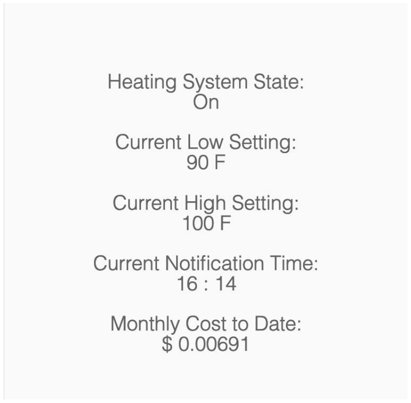
2.1 Overview block diagram



- Arduino collects all sensor data and transfer to Nodemcu.
- Nodemcu sends the data to AWS IOT.
- AWS IOT sends the data to Lambda.
- Lambda stores the data into AWS DynamoDB, and present the data on Web Application
- Lambda triggers the Simple Notification service if the temperature is too low or too high
- Lambda turns on the heater is the temperature is too low, or turns off the heater if the temperature is higher the desired lowest temperature

2.2 User interface.

Monitoring of the system:
There are two ways to monitor the system. The first way, is to view the states on the sidebar of the web application. The settings and state are clearly stated on the side bar.



Heating System State:
On

Current Low Setting:
90 F

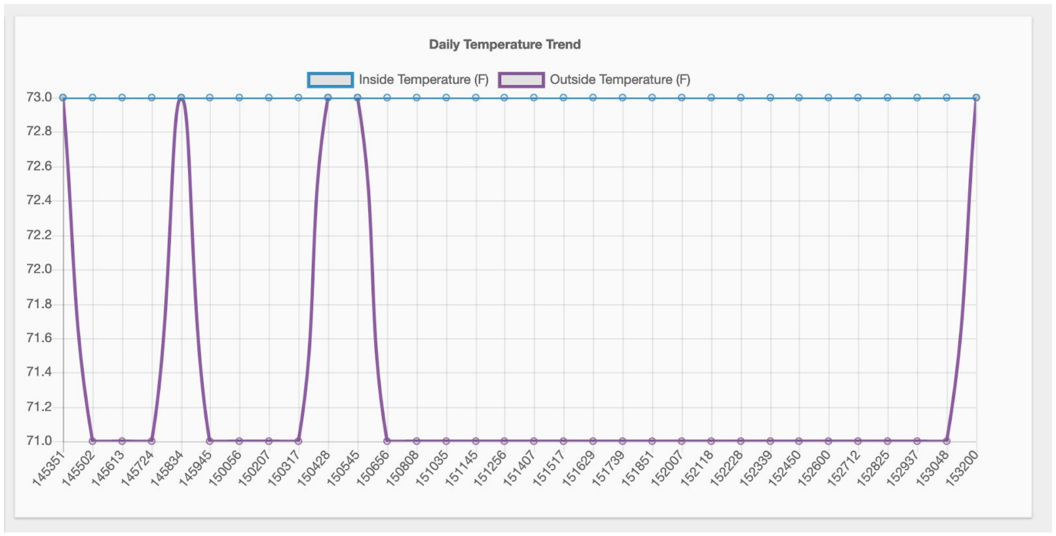
Current High Setting:
100 F

Current Notification Time:
16 : 14

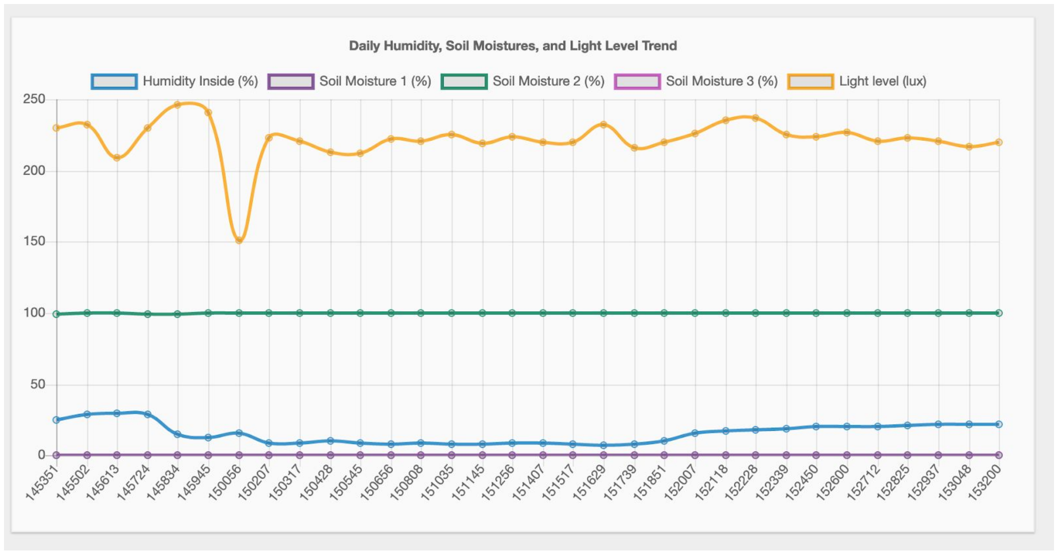
Monthly Cost to Date:
$ 0.00691

The second way is to monitor the system through the graphs. The graphs on the main part of the web application is shown for visualization. The first graph demonstrates the trend of inside and outside temperature. The second graph demonstrates the light level, humidity, and soil moisture. Users are select which specific set of data to be demonstrated for clear and focused view. The last graph is to provide the year's monthly cost of operation of the greenhouse so that the user can keep track of its cost change throughout the year. The horizontal axis is the timestamp in hour,minute, second as string.

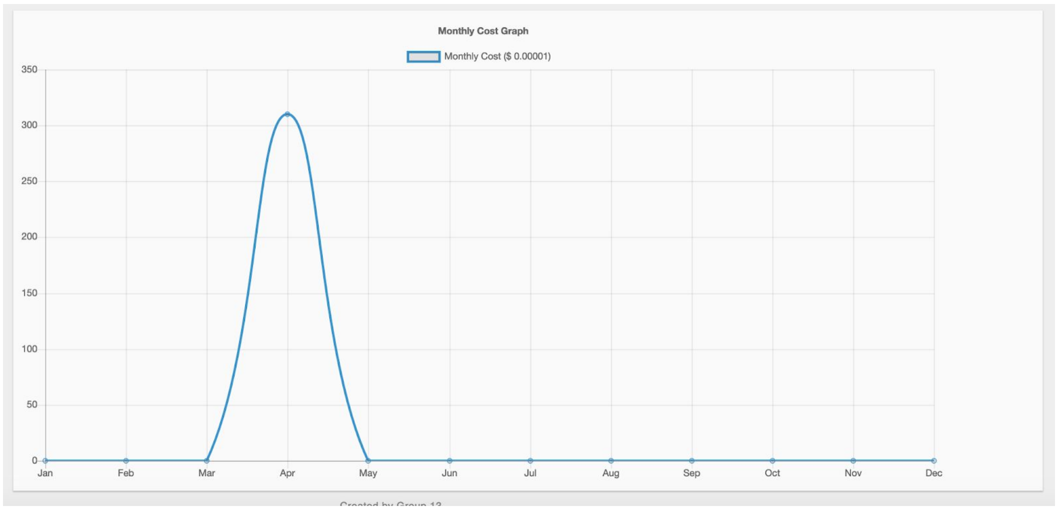
1st Graph: Daily Temperature Trend



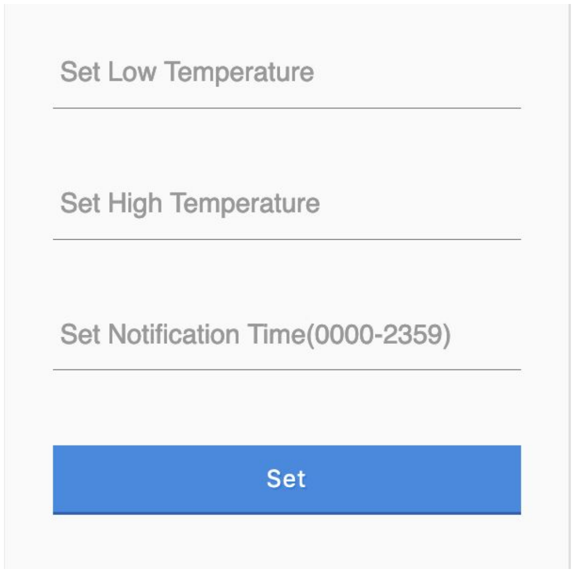
2nd Graph: Lux, Humidity, and Soil Moisture Daily Trend



3rd Graph: Monthly Cost Trend



Control:
Set the low temperature in F: input numbers ranging from 0 - 100
Set the high temperature in F: input numbers ranging from 0 - 100
Set notification time in Hour:Minute: input the time as numbers only from 0000 to 2359



2.3 Physical description.

Provide a sketch of your project hardware (accurate and to scale, in 3-D or as a series of planar views), or photograph.

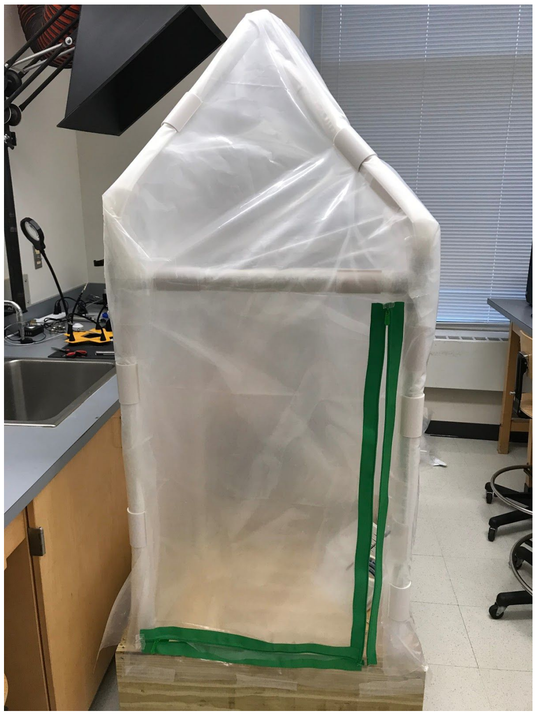
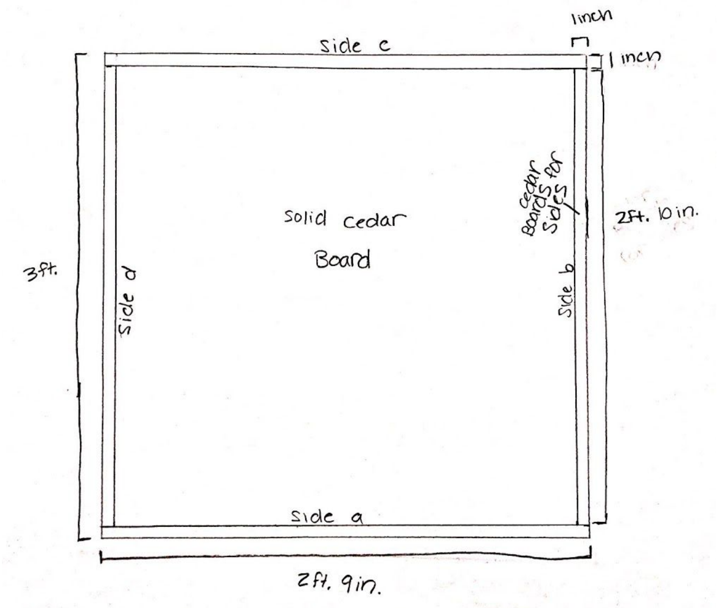


Photo of the completed structure.



Sketch of a planar, aerial view of the bottom box of the greenhouse structure.

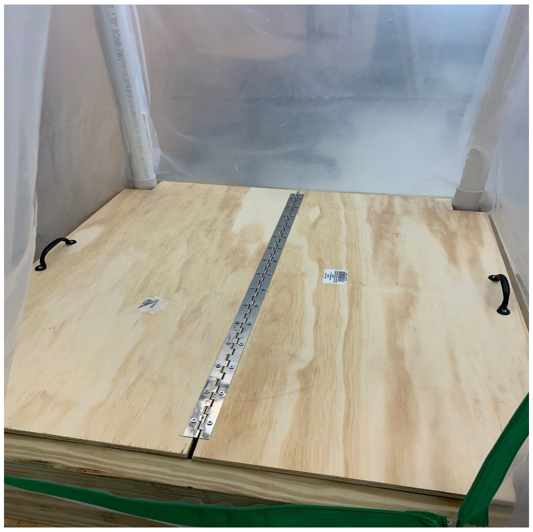


Photo of the subfloor within the greenhouse. The subfloor is composed of two 1/2in x 15in x 32in pieces of BC plywood joined together by a 1-1/16in x 30in nickel continuous hinge. Four 2.5in x 2.5in squares were cut from the outermost corners of the wooden panels to fit around the PVC frame.

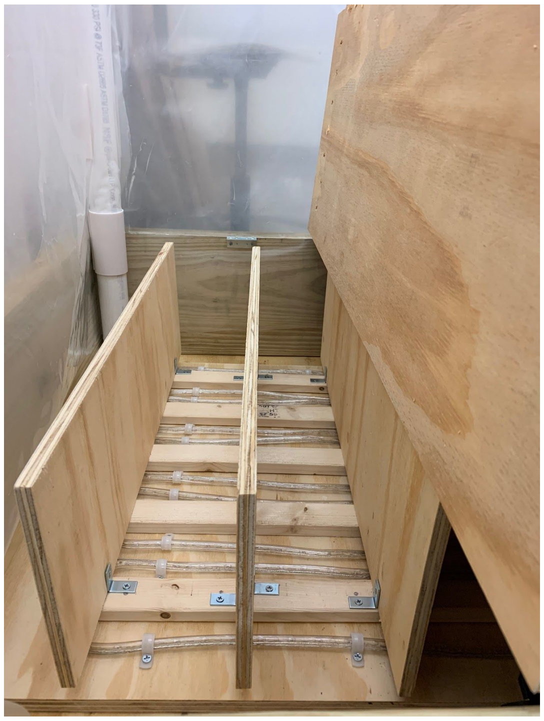
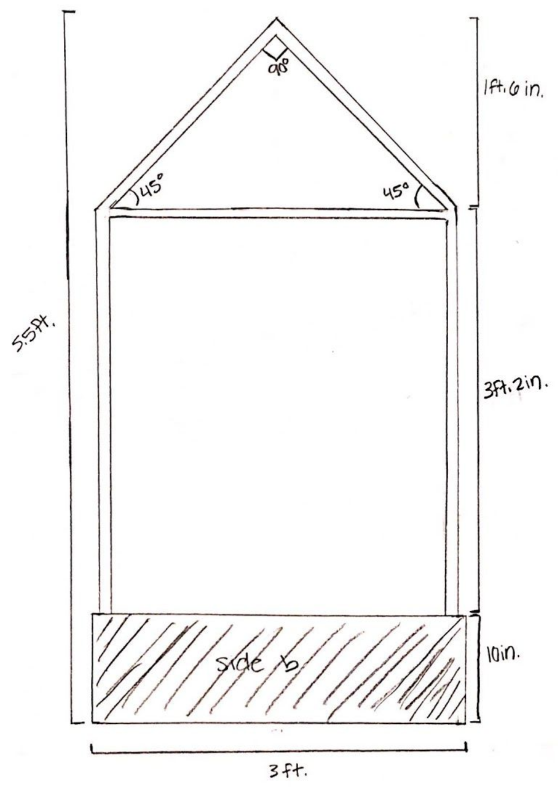


Photo of the wooden subfloor supports and heating pad. There are four, 1/2in x 8in x 2ft wooden supports in total, each placed five inches apart. One-inch corner brackets, fastened to the furring strips of the heating element, are used to keep the supports upright. In the event the incandescent rope light needs maintenance, the supports can be easily removed by lifting them upwards.



Sketch of a planar view of one side of the greenhouse structure. This sketch shows the PVC pipe frame placed in the bottom box. All sides of the frame follow this same design.

2.4 Installation, setup, and support

Installation of webapp on local machine

1. Clone the repository on to the local machine.
2. Add .env file that contains the API keys(content of .env is provided by our team) .
3. Install Python 3 (Python 2 won't work for this project)
4. Install Flask
5. Install Boto3 python client made by AWS
6. Install AWS client
7. Create new access key for this boto3 client on AWS
8. Run "aws configure" to add your generated access keys to your system

Command to run the the server:

1. Go to the webapp directory
2. "./run.sh" is the only command to run the server.
3. Route for the webapp is localhost:5000/login/dashboard

Configuring Wemo Smart Plug with your WLAN:

1. Download "Wemo" App from App Store onto your phone (available for both iOS and Android)
2. Connect your phone to your local 2.4GHz network. NOTE: we recommend disabling auto-reconnect to all other wifi networks or entering Airplane mode to ensure your device is paired to the correct network
3. Open Wemo App and select "Add a Wemo" in the settings page
4. Follow instructions on phone to add Wemo device. NOTE: if this is your first time installing Wemo or your SSID changes, you must hold down the power button while plugging your Wemo in to reset to factory mode. Wemo is ready for setup when light is flashing white and orange.
5. If remote access is not automatically enabled, select "Enable remote access" under settings
6. Plug heater into Wemo

Physical Structure

1. Open the door of the greenhouse to place the plants inside
    a. Pull vertical zipper upwards and horizontal zipper to the left to open the door
    b. Place plants on the wooden subfloor
2. After the plants are placed in the greenhouse, close the zipper door
    a. Pull vertical zipper downwards and the horizontal zipper to the right to close the door

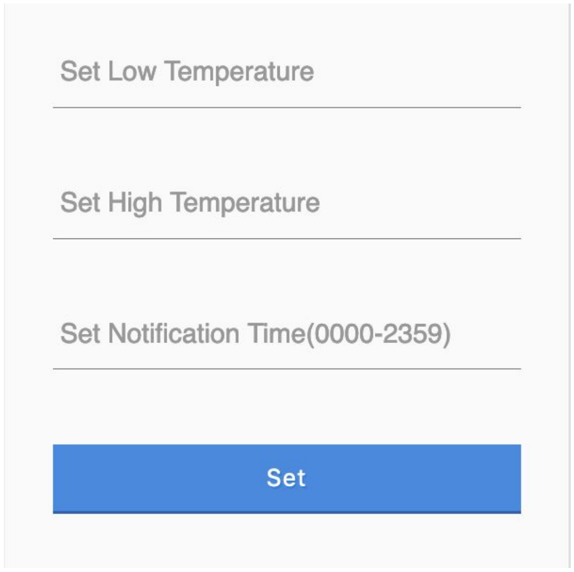
3 Operation of the Project

3.1 *Operating Mode 1: Normal Operation*

For the initial installation, the user needs to power the greenhouse, the Nodemcu will automatically connect to his/her wifi. The user needs to download "Wemo" App from App Store onto the phone (available for both iOS and Android), connect the phone to local 2.4GHz network, and follow instructions on phone to add Wemo device. When Wemo is connected to the internet, simply plug heater into Wemo.

After greenhouse is set up initially, our client does not need to do additional work as long as there is not abnormal situation. The sensor dynamically reads the temperature and sends the data package to Lambda, the central control system. All the functionalities run automatically when the greenhouse is powered. Although our client does not need to actively do additional work, he/she could customize the desired lowest, highest temperature, and the time that he/she wants to receive the regular update on the web.

At the locally hosted web application, the user could customize the the desired lowest, highest temperature, and the time of receiving the regular update.

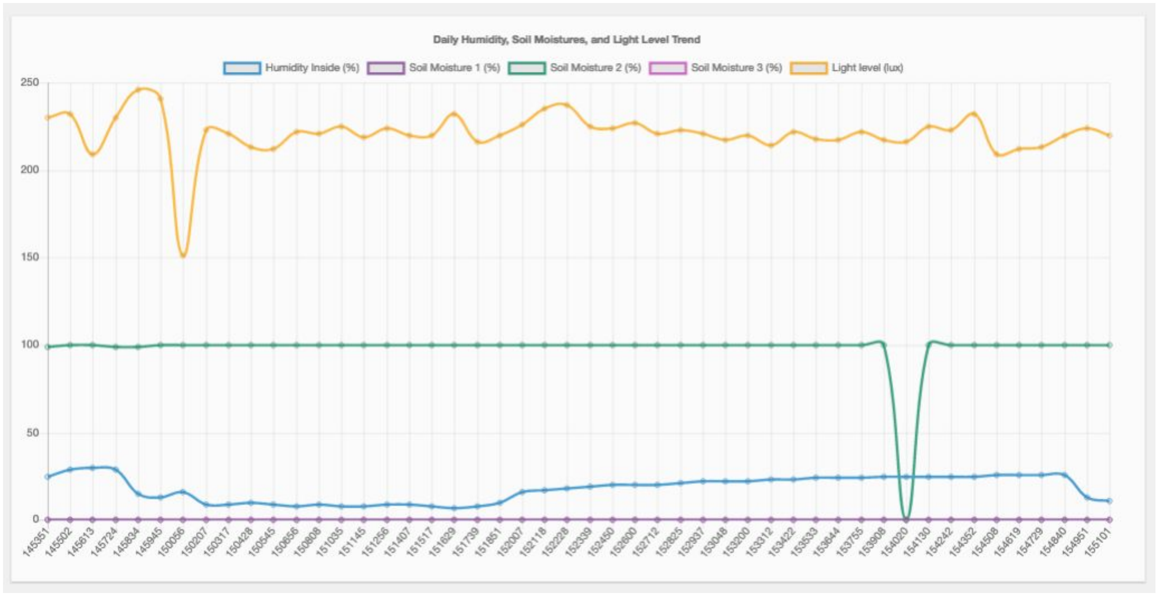


The temperature is in fahrenheit. When the temperature inside the greenhouse is lower than the lowest temperature or higher than the highest temperature, our client receives the alarm message. Also, if the temperature inside the greenhouse is lower than the lowest temperature, the heater is turned on.

The notification time is simply hour and minute. Our user receives the regular update, which includes the temperature inside and outside the greenhouse, humidity, soil moistures, light level and the power consumption of every months.

3.2 *Operating Mode 2: Abnormal Operations*

We have observed extremely rare but still possible inaccurate readings. We have taken multiple steps to remedy these incorrect readings, but would like to provide the user with a response mechanism if they are to occur. If any readings severely change, but then return to their normal state upon the next cycle, we would recommend that the user ignore these "severe" readings. A possible plot is attached. We have implemented software and hardware controls to prevent these inaccurate readings and have not seen them occur since, but with any software device there is always a non-zero chance for error. If severe changes occur and then persist, they are likely accurate and you should inspect your greenhouse for damage.



An extreme and persistent sensor reading (>1000) likely indicates that your sensor has become disconnected or even damaged. Please check your circuitry to ensure that all components are correctly wired. A correctly wired component that continues to provide incorrect readings is likely damaged and should be replaced.

If you fail to receive readings over a period of time greater than 5 minutes, it is likely your greenhouse has experienced connection issues. Please check that your wifi network is strong and is able to connect to your greenhouse. If you have chosen to use your own AWS settings, please also check that all public and private keys are correct.

3.3 *Safety Issues*

The main safety concerns with this project largely relate to damage to the electronics system. Water does present a concern in regards to this. To account for this, the electronics system is housed in an elevated box that can be moved to any of the four supporting PVC pipes. This allows the user to move the electronics system to where they feel it is least likely to be harmed from water damage. This is a decision the user must make as different plants require different watering techniques and reach different heights. Furthermore, individuals may interact with the greenhouse in different ways, so this customizable quality is important to provide the best safety for all potential users.

Installing the product does present a few minor safety concerns. The product rolls but only in one direction. It should be rolled slowly and carefully to ensure no harm to the user or the device itself. The user should also take care when plugging in all electronics . As the device uses a standard US 120V AC outlet, it is possible to cause harm by having any body parts directly in contact with the prongs of any plugs while plugging in the various parts of the device directly to either the wall outlet or the outlets on the extension cord.

4 Technical Background

Arduino and NodeMCU Embedded System:

Two microcontrollers are used to read current measurements and send these measurements to the DynamoDB database over a wifi connection. The first microcontroller is an Arduino Genuino Uno. All sensors are directly connected to the Arduino using digital and analog connections depending on the sensors requirements. The Arduino packs all of these readings into a single string which it sends to the NodeMCU over a Serial UART connection every 5 seconds.

Upon setup, the NodeMCU connects to the provided wifi network. It then attempts to connect to AWS IoT Core using MQTT. If either of these connections fails, the board will attempt to reconnect until successful. The NodeMCU receives the string from the Arduino, parses it, and then sends the measurements to AWS IoT Core over the user's wifi connection. The Arduino does not at any point read data sent by the NodeMCU, though the NodeMCU does print some information over a Serial connection for debugging purposes. If the NodeMCU disconnects from AWS or the user's wifi at any point, it will attempt to reconnect and continue operation. To detect and avoid of sending empty data package, an constant variable is sent with all other sensor data, and its value is always one.

AWS:

The central system is written in NodeJS and is located on Lambda supported by AWS, , where the code can run without provisioning or managing servers. The central system integrates all virtual funcalities except backend of the web application, and most of them are supported by AWS as well. Application built in Lambd is responsive to events and new information. For our system, the application is responding to events triggered by AWS IOT. We linked AWS IOT to our greenhouse by registering Nodemcu in IOT. The communication between Nodemcu and IOT is through MQTT, the publish-subscribe-based messaging protocol.

When AWS IOT is triggered by the MQTT request, it transfers the JSON package to AWS Lambda and triggers the central system. The central system in Lambda firsts check if the value of constant variable is equal to one. If its value is zero, this data package is invalid and rejected. If its value is one, this data package is valid and accepted. Next, the central system reads the current date and time and adds the timestamp into the data package, so every set of sensor data includes the timestamp. Wth the timestamp, the central system adds the data package inside the database. The database linked to Lambda is DynamoDB, the fully managed proprietary NoSQL database service that offered by Amazon as part of the AWS portfolio. DynamoDB does not only store the sensor data package, but also the parameters. Some of the parameters are only for running the system and stored internally, while some others are set by our client, including the lowest temperature in the greenhouse that our customers desired. Lambda reads the lowest temperature parameter stored in DynamoDB, and compares the temperature inside the greenhouse. If the lowest desired temperature is lower than temperature inside the greenhouse, Lambda sends an alarm to our client's email address via Simple Notification Service (SNS). SNS is notification service supported by Amazon and our client's email address is pre-subscribed to SNS. Lambda customizes the message and triggers the SNS, which sends the message to our client's email address.

Simultaneously when SNS is triggered, Lambda triggers the heater. The heater is powered by Smartplug that could be triggered by receiving HTTP request. By sending HTTP request, Lambda turns on and turns off the heater remotely. If the temperature inside the greenhouse is lower than the lowest desired temperature and the heater is currently off, Lambda sends a HTTP request to turn on the heater. Similarly, If the temperature inside the greenhouse is higher than the lowest desired temperature and the heater is currently on, Lambda sends a HTTP request to turn off the heater. Lambda also records how long the heater is on and stores the time in DynamoDB by months. By multiplying the time by the power of the heater, the power consumption is estimated. By multiplying the power consumption by the electricity rates, the electricity cost is estimated.

We also provide the service of updating our the client the status of the greenhouse at a specific time. This time is input in the web application and stored in DynamoDB. Matching the specific time to the timestamp of data package is difficult. The central system stores and updates the latest timestamp in a item. Whenever Lambda receive the new data package, it compares the latest timestamp and the new timestamp to the specific time. If the latest timestamp is before the specific time and the new timestamp is after the specific time, Lambda customize a message which includes all the sensor data and power consumption and sends to client by triggering SNS.

Web Application Backend:

The backend of the web application is built with Flask written in Python. There are many reasons for using Flask. First, the software engineers on this team are fairly familiar with Python, so Flask seemed like an obvious extension of that. Secondly, Flask is a lightweight framework and accelerates the development and production as developers can focus on the creation of the functions. Thirdly, it has Object-Relational Mapping support and clear and defined Model-View-Controller organization which assists in rapid software development. Last but not least, there are many available Python modules available so choosing Flask allows us to utilize existing Python modules.

The Flask backend communicates with the system through the use of BOTO3 Python AWS Client for simplicity. Flask uses BOTO3 to interact with the DynamoDB

NoSQL service of AWS. Using BOTO3 greatly simplifies the process of development and also accelerates development. BOTO3 retrieves lists of readings of data from DynamoDB as JSON packets. With the JSON module, reading of JSON data is easy to decode.

Control settings are the only mechanism by which the user directly communicates with the database. These controls are handled by a NoSQL write in the DynamoDB database. The low temperature setting, high temperature setting, and the weekly notification time can all be set by the user from the web application. These new values are sent to both the AWS Lambda Function and the AWS DynamoDB database.

Web Application Frontend:

The rendering of the dynamic dashboard is done with the Flask template. The only other module used in the front end is the Chart.JS framework. The Chart.JS is used to render the graphs displayed to the user. Data are retrieved from the database, transformed into an array of values in the backend from JSON objects. Then, those arrays are sent to the front end as a part of the response parameters. With the Flask templating engine, the templated variables are now provided with the necessary data for rendering from the provided data. The templated variables on the side bars are also provided and replaced to show the current state of the greenhouse. With the templated fields, the graphs are rendered.

Hosting:

The client has informed us that hosting the web application on one of his existing domains or hosting locally are both acceptable options. We have chosen to host locally to reduce unnecessary cost. The project is easy to run on localhost. A run script has been written to simplify the process to start the server. Running the server should be a fairly simple task. Though the user does not have to configure AWS to run on their own account, it is strongly recommended they do so. Currently DynamoDB and Amazon Lambda are being hosted on a student account, which is likely to close after graduation. The user should set up their own AWS account to continue to receive information after May.

5 Cost Breakdown

Consider your EC464 prototype to be the *alpha* version. The next unit made, according to your engineering specifications and design, would be the *beta* version. Later a manufacturing version or release-version would be made.

What would be the cost of your ***beta*** unit when it is created? This should assume market costs, i.e. no donations, no picking through the customer's parts closet.

You can edit the table below to describe the project expenses for the beta version. It is not necessary to provide every detail about parts, labor, and services in your cost breakdown. Decide upon a level of aggregation of investment and group costs accordingly.

Project Costs for Production of Beta Version (Next Unit After Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
Arduino Genuino Uno	1	Microcontroller to read sensor readings	$22.00	$22.00
NodeMCU (ESP8266)	1	Microcontroller to send sensor readings over wifi to AWS	$8.39	$8.39
DHT11	2	Temperature and humidity sensor	$2.40	$4.80
NSL19M51	1	Photoresistor	$1.13	$1.13
QLOUNI Soil Moisture Sensor (5 pk)	1	Soil moisture sensor	$7.99	$7.99
Wemo Mini Smart Plug	1	Smart plug to activate heater	$16.89	$16.89
Wires	25ft	Wires to connect all components	$0.29	$7.25
PVC Pipes	4 - 10'	PVC pipes were used to construct the Greenhouse frame	$5.13	$20.52
PVC Joints	2 - 90° 4 - 45° 4 - 4 way	Joints used to hold PVC pipe frame together	$2.13 $1.26 $3.81	$4.26 $5.04 $15.25

	joint			
PVC clamps - pack of 10	4	Secures polyethylene plastic to PVC pipe frame	$10.45	$41.80
Polyethylene Plastic 12ft x 25ft	1	Covers the PVC pipe frame	$57.94	$57.94
Plywood 4x4 BC plywood	1	Used as the floor of the bottom box	$20.88	$20.88
Treated Lumber 2inx10inx12ft	1	Walls of the bottom box	$18.77	$18.77
Adhesive Zipper	1	Creates door of the greenhouse	$17.99	$17.99
Incandescent rope light	1	Main heating element for the greenhouse	$33.98	$33.98
Cable Clamps 1/2in (12 pk)	2	Secures rope light to heating pad platform	$1.28	$2.56
Wood Screws #8 x 1/2in Zinc Plated (100 pk)	1	Attach furring strips to heating pad platform	$3.92	$3.92
Plywood 1/2in x 2ft x 2ft BC plywood	1	Base of heating pad platform	$9.19	$9.19
Pine Furring Strip 1in x 2in x 8ft	1	Cut to make five 1.5ft long furring strips for heating pad platform	$1.42	$1.42
Wood Screws #8 x 1in Zinc Plated (100 pk)	1	Attach furring strips to heating pad platform	$4.76	$4.76
Plywood 1/2in x 2ft x 4ft BC plywood	2	Cut to make two, 30in x 32in panels for the subfloor and four, 7-3/4in x 2ft supports for the subfloor	$13.92	$27.84

Continuous Hinge 1-1/16in x 30in Nickel	1	Join panels of foldable subfloor	$7.47	$7.47
Door Handle 4-7/8in	2	Attached to subfloor, allowing either panel to be raised or the entire floor removed	$2.97	$5.94
Corner Brace 2in Zinc (4 pk)	1	Support for subfloor	$2.56	$2.56
Corner Brace 1in Zinc (20 pk)	1	Used to keep wooden subfloor supports in place	$7.48	$7.48
Wood Screws #6 x 1/2in Zinc Plated (100pk)	1	Used to fasten corner braces to wooden subfloor supports	$3.82	$3.82
Beta Version Total Cost				$381.84

6 Appendices

6.1 *Appendix A - Specifications*

Greenhouse Structure

1. Enclosure for more than one plant, the following plants as examples: Mandevilla, Poblano Pepper, Basil and Rosemary;
2. Walls should be transparent to allow light in.
3. Can be opened easily for maintenance and watering in 10 seconds by an individual.
4. Plants can be removed from greenhouse if desired, this should be done without anyone’s assistance.
5. Transportable by an adult with wheels
6. Should be able to withstand rain and snow without damage to the plants or electronics.
7. Automatically maintains a suitable range of temperature during New England winter and summer. Heating must be automatic, but any need for cooling can be accomplished by other means such as a vent that must be manually opened. The ideal temperature range can be set by the user via the web application:

   Examples:

   a. Mandevilla: 60-65º F at night, 68º F during the day.
   b. Poblano Pepper: 60-75º F.
   c. Basil: 50 - 75º F.
   d. Rosemary: 60 to 65 º F.
8. Has sensors measuring temperature (ºF) inside and outside the greenhouse, air humidity (%), soil moisture (%), and power consumption (Watts).
9. Sensor data is sent to web application over user’s WiFi network.
10. Works using one standard North American wall plug outlet (110-120V AC).

Software

1. If the temperature inside the greenhouse is lower than the desired lowest temperature set by user, the heater turns on automatically. An alarm message is sent to client
2. If the temperature inside the greenhouse is higher than the desired highest temperature set by user, an alarm message is sent to client
3. If the greenhouse is warming and the temperature inside the greenhouse becomes higher than the desired lowest temperature set by user, the heater turns off automatically
4. Everyday a message containing all sensor data and power consumption is sent to user, and the specific time is set by him/her
5. The greenhouse records the time that the heater is on, and estimate the electricity cost

Web Application

1. Displays all current sensor readings within the greenhouse
2. User can set the ideal temperature range, and the specific that he/she wants to receive the regular update
3. User can see the power consumption and estimated cost on the web

6.2 Appendix B – Team Information

6.3

Olivia Dorencz

Olivia is a computer engineering major graduating this semester. She loves computers, painting, and baking. She hoped this senior design project would give her more of a "green thumb", but she has killed two succulents since the beginning of the year. After graduation she will be working as a software engineer on the Ground Systems Team at United Launch Alliance in Denver, CO.

Hok Yin Shum

Hok Yin Shum is a computer engineering major student. He is obsessed with hong kong style homemade milk tea. He doesn't like having boba in his milk tea because they are too high in calories. He also doesn't like adding sugar either.

Laura Morgan

Laura is majoring in computer engineering and is graduating this semester. She enjoys reading, playing video games, and being outdoors. After graduation she will be working as a systems engineer at Unisys where she will join the networking and communications team in the operating system department.

Kiahna Tucker
Kiahna Tucker is a computer engineering major graduating this semester. After graduation she will continue her studies as a part-time, Boston University graduate student (pursuing her masters in computer engineering) while working as a software engineer at Google LLC.

Qian Zhang
Qian Zhang is a computer engineering major graduating next semester. He transferred from School of Business to College of Engineering at the year of junior since he wanted to learn some practical and challenging knowledge. After graduation he will apply for graduate school and major in Accounting and Finance.