Boston University

Electrical & Computer Engineering

EC464 Greenhouse Prototype Test Report II

by

Team # 13

GreenHouse

Olivia Dorencz odorencz@bu.edu

Laura Morgan lamorgan@bu.edu

Hok Yin Shum hyshum@bu.edu

Kiahna Tucker kiahnat@bu.edu

Qian Zhang zhang1@bu.edu

Submitted: February 28, 2019

Table of Contents

Summarization of Equipment and Setup

Our prototype is partitioned into two, major categories: hardware and software. The hardware division consists of a heating system and the structure of the greenhouse, while the software section is composed of a web application, Arduino and sensors, as well as a series of cloud-based services provided by AWS (i.e. Amazon IoT, Amazon DynamoDB, and Amazon Lambda).

The underlying framework of the greenhouse was setup to provide a strong depiction of the finished project. A cedar box, constructed using a BC plywood base and four 2” x 10” x 3’ weather-treated, cedar lumber walls, served as the foundation of the greenhouse. Four wheels are attached to the underside of the plywood, enabling forward and backward translation. Within this wooden semi-enclosure, a heating pad is found in its center. The platform of the pad is a 0.75” x 2’ x 2’ BC plywood base with five 1” x 1.5” x 1.5’, pine furring strips placed 2.5” apart. An 18’ incandescent rope light is looped between the furring strips, held in place with 0.5” cable clamps. Additionally, one and a half inch PVC pipe was used to create the main frame of the structure.

Our prototype was intended to test the ability to accurately obtain measurements from our various sensors, confirm that we were able to store the sensor data to the online database via nodemcu, and visualize data in a web application. The nodemcu connects to the laptop computer with a USB 2.0 cable for power.

In the testing, we used two DHT11 temperature and humidity sensors, and two QLOUNI soil moisture sensors with the Flying Fish MH Sensor Series module. We used two temperature sensors as our final product will be able to measure temperature both inside and outside of the greenhouse. We tested the two soil moisture sensors by applying wet paper towels to the sensors and observing the increase in detected moisture.

We also used a NodeMCU ESP8266 Wifi board to demonstrate transmitting data to Amazon DynamoDB. One constraint of this board is its limited amount of analog inputs. NodeMCU has only one analog input, whereas our final product will require multiple analog inputs as the soil moisture sensors require an analog input connection. We do not want our customer to be limited to only detecting the soil moisture in one plant, but to have the option to obtain readings from multiple plants. In the testing, we connected the two soil moisture sensors to the single analog input through a multiplexer and set a delay so the soil moisture data could be collected sequentially. However, for the future, we decide to connect all the sensors to the Arduino UNO board which we used in our preliminary prototyping last semester.

The Arduino UNO has six analog inputs, so we are not limited by the same constraints that the NodeMCU has. In our final product, all of our sensors will be connected directly to the

Arduino UNO. The TX and RX pins of the NodeMCU and Arduino will be connected allowing for serial communication between the boards. Two digital pins will also be connected to indicate whether each board is in a read or write state. The sensor data gathered from the Arduino is packed into a string and sent to the NodeMCU. The NodeMCU decodes this string and sends the information to DynamoDB using the MQTT protocol. Our testing verified that communication between the Arduino and the NodeMCU is an effective method of utilizing the additional Analog pins on the Arduino while retaining the wifi functionality of the NodeMCU. However, for the purposes of our second prototype testing, this system was tested separately and only the standalone NodeMCU with multiplexed analog inputs was included in our test plan due to time constraints.

When the NodeMCU is powered, the sensor collects data every 10 seconds, and transfers them to Amazon IoT Core. In Amazon Iot Core, two actions are triggered simultaneously. The first action is inserting the data package into a DynamoDB table in the format of JSON. The second action is when the temperature inside the greenhouse is higher than 50 degrees, an alert is sent to the customer's mailbox. The code to achieve this functionality is stored in Amazon Lambda.

The web application is implemented using Flask as the backend. We demonstrated that the web application can establish a database connection to DynamoDB. On the front end, the web application displays graphs of temperature both inside and outside of the greenhouse on a linear plot. It also displays air humidity within the greenhouse and soil moisture. A sidebar allows the user to manually switch the heater on and off as well as set the threshold

temperature at which the heater will automatically turn on.

Description of Measurements

Our first step in testing consisted of verifying the measurements taken from the various sensors by displaying them on the Serial Monitor of the Arduino IDE, verifying the accuracy in the AWS database, and applying conditions to change the measurements read from the sensor. When the nodemcu was initially powered on, the following data sheet was presented in the shadow state, the page where the data package sent from nodemcu is presented:

```
{
  "reported": {
    "time": 1,
    "Temperature_inside": 25,
    "Temperature_outside": 23,
    "Humidity_inside": 6,
    "Humidity_outside": 5,
    "Soil Moisture 1": 0,
    "Soil Moisture 2": 0
  }
}
```

Same data readings were found in the serial monitors, which proved the successful connection between nodemcu and Amazon IOT. The DHT11 sensors read an average value of the room temperature of 24°C with humidity of 5.5%. Since the two sensors were running in the same circumstance, there is not significant difference between two readings. Compared to the prototype last semester, this semester we connected two soil moisture sensors to the NodeMCU and set the delay to overcome the limitation of analog inputs, so we set three testing conditions to demonstrate their capability and accuracy. First the sensor was placed in dry air, resulting in a reading of 0% soil moisture. Secondly, soil moisture sensor 1 was wrapped by a half-wet tissue, which resulted in a water content reading between around 50%. And finally, soil moisture sensor 2 was wrapped by a wet tissue, resulting in a reading between 98% and 100%. All the data reading in the serial monitor and Amazon IOT matched. For testing the notification, we used the heating gun to increase the temperature of the inside DHT11 sensor to 100 degrees. When the reading reaches 50 degrees, an alert message was received in the subscribed mailbox, which was: “Warning: the current temperature in your greenhouse is lower than 30 degree”. In most circumstances, the user will be more concerned about the temperature in the greenhouse being too low. However, for the sake of testing in the controlled environment of the senior design lab, it was significantly more feasible to demonstrate a notification when the temperature is too high rather than when it is too low.

We also kept tracking if the data package is stored in Amazon DynamoDB. When the data package is received in Amazon IOT, an action is triggered and the data package is sent immediately to Amazon DynamoDB, where it is stored. We found the data readings in the serial monitors, Amazon IOT and DynamoDB databases all matched, and the time delay of the transaction was negligible.

In order to test the incandescent rope light's ability to generate heat, a small-scale version of the greenhouse was created and placed outdoors. The miniature greenhouse had a 1" x 2' x 2' BC plywood base, a square, PVC frame, and was wrapped in polyethylene. With the aid of 0.5" cable clamps, the incandescent rope light was secured directly to the base, the cable laid creating even horizontal lines. Three temperature sensors—located just above the rope light, at the top of the interior of the greenhouse, and outside of the greenhouse—tracked the critical temperature changes. An ESP Huzzah32 board collected the data from the sensors and stored them in a local database (generated using MongoDB on a personal computer). With outdoor temperatures ranging from 32°F to 50°F, it took approximately 30 to 47 minutes for the air just above the rope light to reach a relatively steady temperature of 100°F to 102°F. At the 40 minute mark, the temperature sensor placed at the top of the greenhouse's interior began to stabilize, recording values within the range of 71°F to 80°F. Given the optimal temperature for photosynthesis for most plants is around 75°F to 77°F, the heat generated by the heating pad is sufficient to create a suitable growing environment.

Conclusions based on Test Data

We successfully demonstrated that we could read accurate data from all of our sensors, communicate with the NodeMCU over Wifi, store the data in the database, and display data in a web application. So far, we have achieved of sending data from sensor to database, and visualize the data on web. However, the desired temperature within in greenhouse is supposed to be controlled by the user, so we still need to determine how to control the heater within the greenhouse by transferring the desired range from the web to the heater. Our initial plan for this was to achieve this using an electronic relay. We demonstrated the basic capability of using the relay last semester, but have yet to actually wire it to our heater. In order to wire our heater to the relay, we would need to physically cut open the power cable of the heater, correctly wire it to the relay, and reseal the cable using electric tape or heat wrap. The risks of doing this incorrectly include completely destroying all of our electronics as well as potential harm to ourselves as our heater will be plugged directly into the wall socket.

However, we have determined that using a smart plug may be a smart alternative. It would be significantly safer and smart plugs exist that can be used outside without damage to the plug or to the electronics plugged into the plug. They are also relatively low cost. Many have APIs that allow for them to be controlled through different applications. Rather than sending the signal to turn on the heater from our web server to our NodeMCU to our Arduino, we could send the signal to turn on the heater directly from our web server to the smart plug. This also reduces the number of times we will have to encode and unpack the signal to turn the heater on, allowing for a simpler programming model. We intend to utilize this in our next prototype and demonstrate the full datapath between the various hardware components in the greenhouse and our backend web application.

Currently, data is sent without a proper timestamp. Therefore, there isn’t a way to write an appropriate query to display the data chronologically. In our prototype testing we demonstrated use of the web application using a set of data that was fetched from the database and cached, rather than real time data. The functionality will be extended for our next testing to allow for chronological display of data.

Appendix

1. Sketch of the physical structure

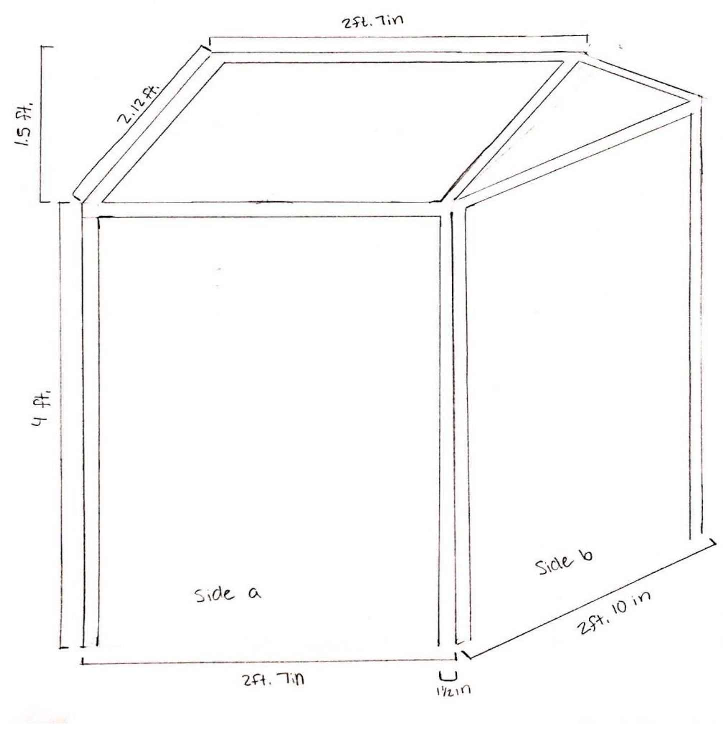

Figure 1: This design sketch is the three dimensional view of the PVC pipe frame of the greenhouse. The total height of the greenhouse frame will be 5 feet 6 inches tall. The PVC pipe width is 1.5 inches and will be joined together using pre-existing PVC pipe joints.

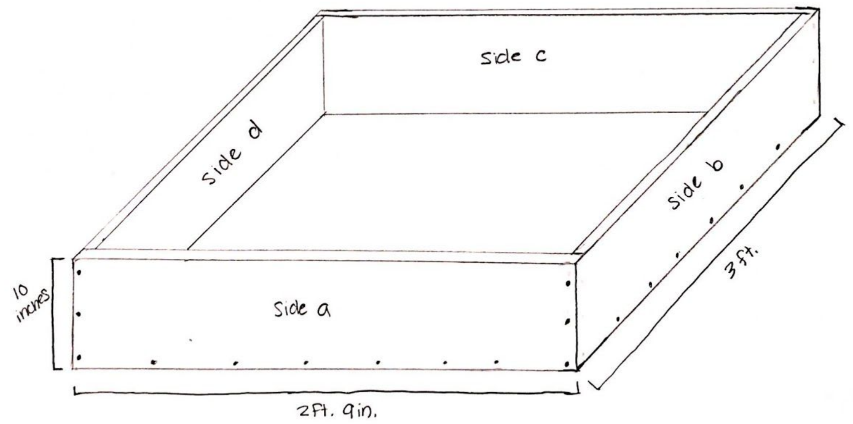

Figure 2: This design sketch is the three dimensional view of the box that is going to be the base of the greenhouse. The frame (depicted in Figure 1) will be mounted to this box. The box itself will be made using cedar lumber. The side boards will be 10 inches wide and 1 inch thick.