Automated Indoor Nursery

Nicholas Leon, Mariana Lozano,

Austen Ordos, and Hamzah Ullah

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The overall goal of this project is to create a system that will be able to provide the necessary functionality to keep plants alive while customers go away for a certain period of time. The system contains both hardware and software components that work together to allow these plants to grow and stay alive with their specific care needs while being able to check the levels of humidity, light, temperature, and water from our website application. The use of various sensors allows for the communication of data between the hardware and software components.**

***Index Terms* — Automation, Garden Pod, Microcontroller, Temperature and Humidity Sensor, Web Stack, Database.**

I. Introduction

For a long time, people have been planting gardens indoor and outdoor. They grow to value their plants and take great care of them to ensure they stay alive and are well taken care of with enough sunlight, soil, and water. The only downside to having and growing these plants is who is going to take care of them while they are away on vacation or if something comes up and they need to leave them unattended. The Automated Indoor Nursery we created will help the customer during these times. Even though the concept of indoor gardening is pretty straightforward, the ultimate goal of this project is to be able to grow the plants you want inside without having to worry about them dying while you are gone and ensuring peace of mind for the customer. For plants to be able to survive indoors they require enough sunlight and hydration along with other components like humidity and temperature levels. The purpose of our device is to provide the necessary care and nutrition these plants need in order to survive.

Our device provides functionality to these unattended plants by certain specifications such as operating for a length of time, the needed resources to be adaptable to different types of plants, or in our case herbs, and being easy to use for the customer. Within the enclosure, we have decided to go with a drip irrigation system to water the plants and decrease the amount of excess water. A camera sits in the top of the enclosure to provide visual feedback to the customer while they are away. The LED lights that sit mounted on the corrugated sheet provide “sunlight” for the plants and function on their own to turn on and off for a certain amount of time. The use of a mobile web application allows the user to control how much water and sunlight is being given to the plant or they can just set up a schedule and leave it alone.

II. Project Goals

Our objective and motivation for this project is to be able to provide ease of mind to customers so they can be worry free while they are away. Many homeowners have some type of plant whether its flowers or a garden growing herb. Even though plants are self-sustaining they still require some kind of need to survive outside of their native environments. For plants to survive indoors they require enough hydration and sunlight. With the use of our product many pros that come with gardening will be ensured as well as ease. Gardening is said to reduce stress, boost moods, and keep a person entertained and busy. Going away may cause some stress for those that have devoted time to their garden, so with our product they can continue to be stress free and can still keep up with their beautifully growing garden while away with the feedback given to them by the camera.

III. System Design

The enclosure is made up of several different parts that make it all come together. The ultimate goal for our system is modularity. The system we are designing needs to be easy to use and easily accessible. Our overall system must not weigh more than 30 pounds and needs to be less than 10 cubic feet. The lighter we can make it the better. We decided to use PVC to make the basic structure of our enclosure in the shape of a cube. The PVC we chose is not only light because it's only ½” thick but also sturdy. It is covered in vinyl which can be easily punctured without the risk of breaking the entire enclosure and is able to keep the water in and provide the “closed” system to our enclosure. We then created a second level within the cube to hold the four pots above the drain pan so they could hang. For our specific product our four pots will contain two basil and two parsleys. These were chosen by their specifications and needs. The bottom of the enclosure acts as a drawer for the drain pan to be able to slide in and out easily for when the customer has to put freshwater in.

FIGURE 1. ENCLOSURE DESIGN (EXCLUDING VINYL COVER)

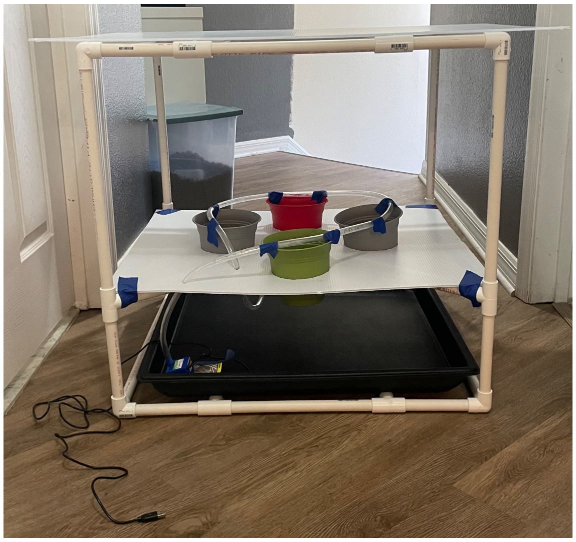
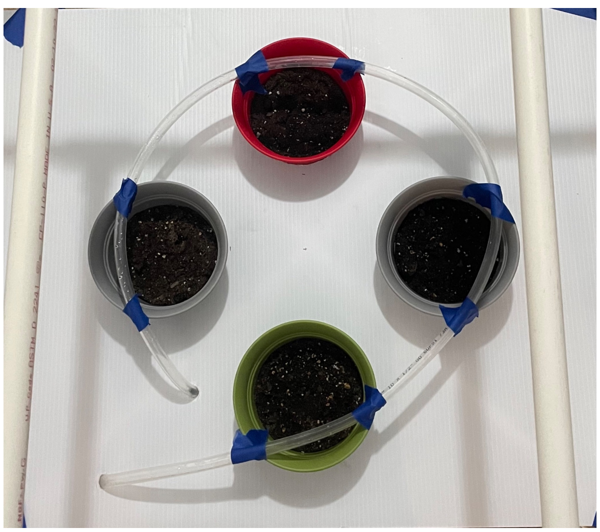


FIGURE 2. TOP VIEW OF POTS AND DRIP IRRIGATION SYSTEM



IV. HARDWARE DESIGN

This section will discuss the hardware aspect of our system. This includes our microcontroller and the sensors that go along with it such as the Wi-Fi module, camera module, humidity, temperature sensor, etc.

A. Printed Circuit Board

Our printed circuit board was designed with the intent that it has the capacity to work on its own with no need for an external printed circuit board to go attached to it.

B. Sensors

In order to be able to calculate how many external factors there are present in our enclosure sensors are used. The DHT11 sensor we used incorporates various sensors in one module. These sensors are for temperature and humidity.

There is a light sensor that detects the amount of visible light and UV rays via an index from 0 - 300. Zero signaling nothing, and 300 signaling the maximum calculable input.

A pH sensor was going to be inserted in the main water reservoir that would allow the pH level in the water to be detectable. The scale reads from 0 to 14. Any number below 7 (neutral) reads as acidic and the lower the number the more acidic it is. Any rating above 7 becomes alkaline. In the end we were not able to incorporate this sensor due to the fact we were never able to properly calibrate it to get accurate readings.

There is also the implementation of a liquid level sensor which detects the presence of liquid where the sensor's probe is inserted. The serial port will display a 1 (True) for when a liquid is detected and a 0 (False) when no liquid is detected.

C. Microcontroller

The chosen MCU was the ATmega2560 which is a low power 8 bit microcontroller.

D. Camera

A camera was placed in order to be able to capture images or display a live view of the enclosure. The Arducam Mini 2MP camera module was chosen due to compatibility since most of the equipment runs on Arduino hardware and software.

E. Wi-Fi-Module

A Wi-Fi module is used in the system to allow internet access as a medium for manipulation of data. With the help of the Wi-Fi module, we can access the daily pictures or a live view of the enclosure. It is also capable of relaying live sensor data as well. This is crucial in case the user forgets needs to change any levels in the system.

F. Pumps

There is a water pump in our design. It pumps water up the drip irrigation system from the water reservoir. It is controlled by an individual relay that is programmed to switch open and close according to when the pump needs to operate.

G. LED Lights

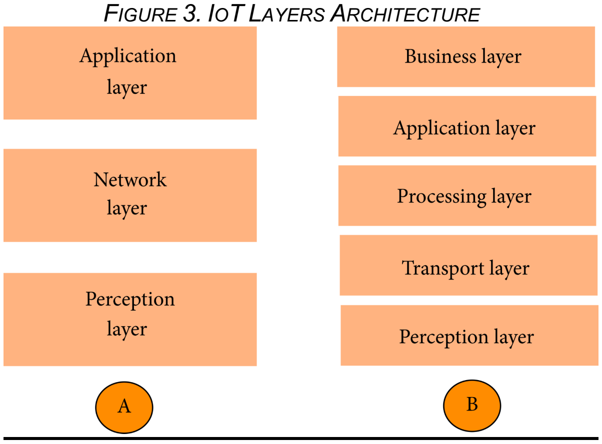
The lights chosen were two grow lights that were programmed each through a relay to turn on and off. An individual external power source was used for both blocks of lights. These will be controlled digitally with signals commanding the lights to be on and off for certain hours through the programmed relays. All plants will receive the same amount of light because of their specifications.

V. Software Design

This section will cover the overall software design of the system by splitting the software into the following sections: IoT Architecture, MERN Stack, Database, web server, and website design. These sections will give a better overview for the software aspect of our system. The use of software was needed to help the system communicate with the website and printed circuit board. Then the database would update the information with the sensors used.

A. IoT Architecture

Figure 3. IoT Layers Architecture



Perception Layer:

The perception layer of our project is how we collect information about our plant and surrounding environment. The perception layer consists of our hardware that is used to collect temperature, humidity, light, and growth rate. Referencing the software diagram, the information from the sensor would be sent to microcontroller memory. From there, we would format the data before being sent to the user interface to proceed with an action or display data for the user.

Network Layer:

The network layer of our project is responsible for transmitting the information between itself and the user interface. In our case, the user interface would depend on a server built on Express.js and a NoSQL database. Using a document model, MongoDB, we store relevant sensor data as an object to build a history of the plant's growth and relevant sensor data alongside. Storing and retrieving that data would be done using a querying language called GraphQL. This information is accessible in the user interface once they query the database and retrieve the relevant data.

Referencing the software diagram in figure 4, this represents the in-between our middleware of the project where all persistent data resides. For our server to retrieve the data, the hardware interface is reliant on a Wi-Fi module. The module allocates the formatted data from MCU memory and transmits it to an application programming interface (API) endpoint where the middleware servers can then retrieve the data and store it in the MongoDB database. Finally, our front-end, or user interface would make a request to the database and either send, receive, or remove information as it determines its next course of action based on the chain of request from MCU to the user interface.

Application Layer:

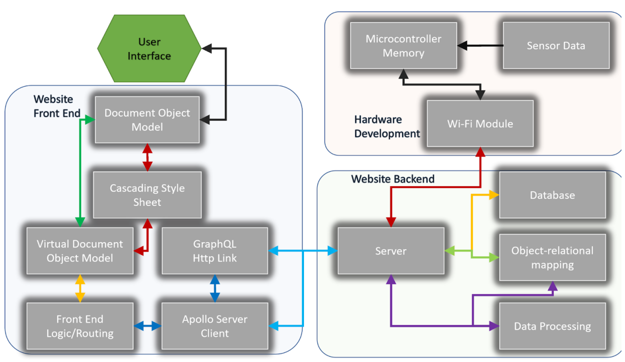
The application layer of our project represents the user's interaction with our automated nursery. Utilizing the data retrieved from the network layer, allocated by the perception layer, the user can interact and perform actions and make decisions based on sensor data from the perception layer. For example, if the user is able to see that the humidity in the automated nursery is out of range for the optimal environment, the user can perform actions to correct the issue.

Referencing the software diagram in figure 4, our front-end for our web application represents the application layer and the main point of entry in any IoT layer between the user and the device. Our previous example of humidity out of optimal range, this would be determined by the perception layer's humidity sensor. It would relay the information through the network layer and store it in the database. Once the user retrieves an alert or checks their plant within the application layer, they can perform the required actions; Increase humidity in the application layer's web application, the network layer performing the request to the Wi-Fi module and MCU memory. The MCU would then perform the requested action. All while this is happening, the perception layer is continuing to monitor the progress in the perception layer, and the cycle repeats again.

B. MERN Stack

A stack represents a series of technologies used when building an application component. For our project, our stack represents the technologies we will use to build the web application necessary for the user to interact with the hardware integration and automate the nursery. The acronym, 'MERN' represents 4 major components of a web application: database integration, server integration, front-end framework, and back-end framework.

FIGURE 4. SOFTWARE DIAGRAM



(M)ongoDB - Database Integration:

For our database integration, we are using MongoDB. MongoDB is a NoSQL database using a document model system. The document model benefits from being faster than the relational model, a non-rigid schema model and favors a hierarchical node structure, as well as scalability to add more fields quickly. This benefits our projects in several ways. First, having a non-rigid schema allows us to quickly add and remove data throughout the prototyping process while retaining the information from the previous interactions. A relational database would require us to clear the entire schema model if any changes are made, especially when considering joins between one another. Second, horizontal scaling allows us to add more components if necessary to increase user functionality without having to remake the database and lose valuable data. Since proof of concept of our project requires long term preparation as the plants grow, replicating and retaining old data will be crucial in determining success. Finally, the hierarchical node structure benefits allow us to create users which will be the head of every node structure, while it propagates all the way down to the initial sensor data as we process it in the database.

Pairing MongoDB with the Mongoose package, allows us to quickly build schemas in Javascript notation without directly writing any NoSQL script. Furthermore, the integration of the querying done through GraphQL, allows us to define queries directly in the front end without running the risk of NoSQL injection attacks while explicitly retrieving the data we need in a single JSON response.

(E)xpress.js - Server Integration:

Express.js allows us to quickly create a server environment with powerful tools we can integrate within the middleware provided by Express.js. Using it in conjunction with Apollo Server Express, we can use the unique querying capabilities of GraphQL to create both a path for accessing the database, and processing data within its resolvers, before sending it back to the user's front-end environment. Express.js middleware capabilities allow us to bring packages in our environment that will help us meet specifications of our project, such as increasing header size for large data types, bundling technologies that benefit mobile environments.

(R)eact - Front End Framework:

React is a front-end framework that leverages the benefits of having dynamic content. Since the automatic nursery data will constantly be changing , using React and utilizing its state-based architecture will be pivotal in providing an experience to the user that will minimize page refresh while updating with the latest information from the server and MCU. The component-based system of React will allow us to reuse components across our application, maintaining uniformity between pages with predictable behavior. In conjunction with the Redux state management system , we can keep track of state throughout the application, even with components that may require data from a child element.

(N)ode.js - Frontend Framework:

Node.js benefits from being an asynchronous technology that while not in use, will not constantly poll for new data. Upon receiving an event that will trigger Node, the network will respond and complete its promise-based request. For our project, retrieving data when necessary is pivotal in extracting relevant data, while preventing unnecessary information from hitting the Node queue, preventing relevant information being processed and used. Node in conjunction with our Express server will allow data to sequentially pass through the database which is then being used by our front end React application.

C. Database

Models and GraphQL:

We are using a NoSQL database with a GraphQL querying language to access the database. Using MongoDB, we can quickly get the database up and running by using built-in functions provided by the Mongoose library. This in conjunction with GraphQL allows for strict typing of data types and values before the data can successfully be passed into the database. GraphQL allows us to also request the data we are specifically requesting as opposed to the whole table based on the use case. For example, if our user has just a single plant, it can request the user's single plant from a single device, if necessary, without returning 'Plant 2' with null, empty or undefined values. This allows only relevant data to be parsed in the front or backend, speeding up the request response cycle for the user.

For the database, 4 models are created. The user which will be required to be logged in to access the database schemas. There is a structure in place for 'Authorization' to contain the signed JWT or JSON Web Token to be in place before access add information to the web application which will populate the website.

Figure 5. GraphQL query referencing all models


```
{
  "data": {
    "user": {
      "_id": "636b8ed559c59d6e2a46ac44",
      "email": "a@gmail.com",
      "password": null,
      "devices": [
        {
          "_id": "636b8f7959c59d6e2a46ac47",
          "name": "Basil and Parsley",
          "uuid": "123456",
          "plants": [
            {
              "_id": "636b90bf59c59d6e2a46ac56",
              "humidity": 55,
              "name": "Basil",
              "pH": 6.2,
              "position": 1,
              "temperature": 72,
              "history": [
                {
                  "_id": "636b90d459c59d6e2a46ac5d",
                  "createdAt": [
                    "1667993812651",
                    "1667993856786"
                  ],
                  "humidity": [
                    50,
                    56
                  ],
                  "pH": [
                    6,
                    6.9
                  ],
                  "temperature": [
                    70,
                    72
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  }
}
```


Figure 5 is an example of all the populated models with their respective data. The models themselves are the User, Devices, Plants, PlantHistory. The user will have a one-to-many relationship to the amount of unique devices, where each device corresponds to an individual Wi-Fi module unique id, along with a corresponding name representing the automated indoor nursery. Each device will house a number of plants with the database storing them as an array. The one-to-many relationship is limited to up to four plants per device. Each plant will have the 'ideal' conditions for the plants entered by the user. Each plant will contain a one-to-one relationship to its history, allowing us to keep track of the data as the conditions change within the enclosure. The history contains a generated timestamp along with the data which would come into the sensor.

D. *Website Design*

Responsive Design:

Responsive design is the concept that a web application user interface has been designed in a method that would allow a person on a mobile device to have an equally enjoyable experience by modulating display content in a usable way. Our project which highlights the freedom for the user to care for their plants while they are away, requires us to have a responsive first design for one application.

By using cascading style sheets (CSS), we can optimize critical components on the page to render within a usable format for the user. One approach is to use a CSS framework that follows a philosophy of mobile first design, so as we build out components, they are assigned classes that automatically consider screen sizes and resolution. This would result in a web design that would automatically scale up or down the size of the components based on the screen size. In an automated indoor nursery, where the user is expected to take care of their plants while they are away, it is expected that they may not have access to a desktop or laptop, so designing mobile first maximizes the user's ability to navigate the application effectively and quickly.

Strategies generally used to optimize for mobile experience are to use larger buttons, enlarge text size, simply or remove any unnecessary menus, easy contact, and simple forms. Since our application can benefit from all the listen optimizations, we plan to design for mobile responsiveness and an optimized mobile experience by leveraging a cascading style sheet framework that optimizes components based on screen size, as well as creating custom media queries to optimize screen sizes with larger market shares.

Progressive Web Application:

To meet our mobile first design of the web application, we leveraged the technology called progressive web applications. Progressive web applications (PWAs) are installable websites on mobile devices. They mimic an application through a mobile store service, without having to proceed through the Android or iOS system. PWAs allow our application to appear as a user-friendly design without the user having to navigate to a web browser. To meet the requirements for a PWA, there are a set of guidelines in place determined by Google that initializes it as a PWA. By doing a lighthouse either through the Google PWA website at www.web.dev, or directly in the chrome browser we can get a snapshot of what it takes to meet PWA guidelines along with other useful information such as search engine optimization, accessibility standards, performance, and coding best practices.

Some key components to highlight to meet a PWA is the site having a service worker. A service worker allows the site to be cached within the browser to load critical components quickly after the first load is complete. On mobile devices, this allows for the application to not only run quicker on slower networks by retrieving the cached data, but also allow offline use. During offline use, the application will determine if the user is back online and push all the cached actions to complete their respective actions. For our project, allowing the user to have cached database actions that will push up to the server once connection is restored is important to providing the end user peace of mind when caring for their plants when they may be in a remote region or with slow or spotty service.

VI. Cost Analysis

Figure 6. Cost Analysis

Item	Quantity	Price
Final PCB	1	50.00
Potting Mix	1	5.97
Planters	4	13.92
Miracle Grow Plant Food 2 lbs	1	10.97
Plant Seeds (Parsley and Basil)	6	10.14
PVC pipe + elbow fittings	35	43.81
3/4' to 3/8' adapter	1	10.64
Reservoir	1	38.56
Corrugated Sheet	2	15.16
Vinyl Tubing	1	16.19
Cat Water Fountain Pump	1	11.99
Vinyl (per yard)	3	45.10
Light Sensor	1	21.10
CQRobot Liquid Sensor	1	18.99
Coospider UV Green Killer	1	20.77
DHT11 Humidity and Temperature Sensor	1	free
Wifi Module	3	8.99
Grow Lights	2	32.00
Drip Irrigation System	1	21.84
Liquid Level Sensor	1	free
Camera	1	25.99
Barrel Plug (around 24 V)	1	8.89
Heroku Subscription	1	6.99
	Total:	438.01

VII. Conclusion

Our Automated Indoor Nursery has come a long way since the first discussions as a group. After all the obstacles we had to get through we are so proud of our finished product. The use of software and hardware was needed to bring this all together. Our system was made to provide relief to customers when they go away on business or vacation while also being usable year-round to help their plants grow. The team used their communication skills as well as their engineering skills throughout the two semesters of Senior Design to bring this product to life and create the Automated Indoor Nursery.

VIII. Acknowledgement

Our group greatly appreciates the faculty and the professors at UCF for their expertise and support through our project, and all the years studying engineering. Thank you to our reviewers for taking the time out of their day to be here, watch our presentations, and read our documentation.

IX. References

[1] “Progressive Web Apps (Pwas).” *MDN*, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.

[2] “Semantic UI User Interface Is the Language of the Web.” *Semantic UI*, https://semantic-ui.com/.

[3] Sethi, Pallavi, and Smruti R. Sarangi. “Internet of Things: Architectures, Protocols, and Applications.” *Journal of Electrical and Computer Engineering*, Hindawi, 26 Jan. 2017, https://www.hindawi.com/journals/jece/2017/9324035/#copyright.

[4] “What Is a Rest Api?” *Red Hat - We Make Open Source Technologies for the Enterprise*, https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[5] “What Is The Mern Stack? Introduction & Examples.” *MongoDB*, https://www.mongodb.com/mern-stack.

X. Team Member Autobiographies

Nicholas Leon is currently a Senior at the University of Central Florida working towards his BSEE. Following graduation Nicholas plans on working with John Deere as a Product Engineer II.

Mariana Lozano is currently a Senior at the University of Central Florida currently doing a study program with Ford Motor Company in Product Development. Upon completion of her BSEE and graduation, she hopes to continue working with Ford Motor Company and doing their Ford College Graduate Program.

Austen Ordos is a Senior at the University of Central Florida getting his BSEE. Following his internship with Northrop Grumman he plans on continuing to work with them as an Electronics Engineer.

Hamzah Ullah is a Senior at the University of Central Florida acquiring his Computer Science Bachelor's Degree. Hamzah is currently working full time with 2U | edX as a Software Engineer and plans on continuing with them following graduation.