Nurture: The Automated Plant Monitor System

Design Document

Sddec24-16

Client/Advisor

Ahmed Maruf

Team Members/Roles

Cameron Jones - Computer

Blake Hardy - Computer

Cayden Kelly - Electrical

Chase O'Connell - Electrical

Holden Brown - Software

Tejal Devshetwar - Software

Team Email

sddec24-16@iastate.edu

Team Website

https://sddec24-16.sd.ece.iastate.edu

Revised: 12/8/2024, V4

Executive Summary

Problem and Importance

Amateur gardeners often struggle to remember when to water and fertilize their plants and may worry about the status of their plants while they are out of the house. Our device seeks to rectify this issue by automating watering and fertilization and providing updates on soil quality. By addressing these issues we seek to allow even the most inexperienced and busy to enjoy the hobby of gardening.

Development Standards & Practices Used

Communications: RS485, UART, IEEE 802.11

Water resistance: IPx5

Development style: Waterfall / Agile Hybrid

Summary of Designs

Our design comprises three main components: the device, the database, and the app.

The device:

The device is composed of an Arduino MKR 1010 wifi, which acts as a microcontroller reading data from the two sensors, the NPK sensor and the PAR sensor and reporting that data to the server. The Arduino also controls a pump and solenoid valve that dispense water or fertilizer when a certain threshold amount is reported by the sensors.

The server:

Our server has two parts: Glitch and MongoDB. The Glitch server runs Mongoose and Express, which handle HTTP requests and schema interpretation. The Glitch server interprets the request and then performs it with MongoDB, our database. The server is set up to handle bad data gracefully, improving reliability.

The app:

We used React Native with Expo Go due to its compatibility with iOS and Android platforms. React Native was also chosen because of its seamless integration with the backend; both frameworks use JavaScript, which makes handling JSON objects much easier. The app lets the logged-in user create plants and display a 24-hour graph of data from any of its sensors: moisture, nitrogen, PAR, temperature, phosphorus, pH, potassium, and conductivity. The app allows users to manage their plants by editing, setting thresholds, deleting, and monitoring away from home.

Summary of Requirements

- A developed microcontroller system linked with a range of IoT sensors designed to assess essential soil nutrients like Nitrogen, Phosphorus, and Potassium.
- Ability to transmit collected sensor data to a central IoT platform to be analyzed by advanced algorithms to ascertain the plants' precise needs.
- Automatic watering and fertilizing systems based on data analysis.
- A developed user-friendly app that provides live updates on plant soil conditions, allowing users to take manual action when necessary.

How well do we Meet the Requirements

Requirements have all been met. The device is fully able to communicate with each individual component and can autonomously water and fertilize according to an algorithm. Further, the device is capable of reading and reporting to the user nearly all values that may be of concern when growing plants. However, some user needs may need to be addressed for example, the device still can only be used indoors due to being wall-powered and using a standard non-water resistant AC adaptor for power.

Next steps

To rectify the issue of it being unable to be used outside, a battery component should be added, or a more water-resistant plug should replace our current AC adaptor.

Applicable Courses from Iowa State University Curriculum

Com S 309 - Mobile App Development

Com S 319 - Usage of MongoDB to store data from users

Cpre 288 - Embedded Systems

Cpre 489 - Networking

EE 230 - Circuits 2

New Skills/Knowledge Acquired that were not Taught in Courses

Micropython library

React Native framework

Component selection skills

IoT systems

PCB Fabrication

Table of Contents

List of Figures/Tables/Symbols/Definitions

IoT (Internet of Things): Connected network of devices and hardware that facilitates communication between the devices and the cloud.

Raspberry Pi Pico W: A microcontroller that utilizes Micropython

Micropython: A variant of the Python programming language for use in embedded systems.

NPK Sensor: Soil sensor for nitrogen, phosphorus, and potassium, the three most important nutrients in plant care.

I2C: Inter-integrated circuit communication protocol.

UART: Universal asynchronous receive and transmit communication protocol.

Modbus/RS485: Communication protocol widely used in industrial automation.

Relay: Electromechanical switch.

UX: User experience.

UI: User interface.

MongoDB - A document database for user storage.

Express - Web application framework for Node.js that facilitates backend communication and request interpretation between the database and user.

React Native - Software framework to create frontend apps for Android or IOS.

PAR (Photosynthetically Active Radiation) Sensor: Measures light wavelengths that trigger photosynthesis.

Arduino Modbus - A library that allows Arduino devices to interact with Modbus devices via a UART port.

Arduino MKR 1010 WiFi-A microcontroller that utilizes Arduino Modbus.

Tables

Figures

1 Introduction

1.1 Problem Statement

Plants are a part of the daily lives of many people, from large-scale farmers to hobbyist gardeners. However, all these people encounter the problems and difficulties associated with growing plants: taking time to apply water and fertilizer, uncertainty about when to apply either and in what quantity, etc. Additionally, those who have more knowledge and experience with plant care still have to spend their time collecting data and monitoring the plants manually. Our device, “Nurture,” exists to alleviate these issues.

“Nurture” is a device that, when planted in the soil, automatically takes nutrient and moisture readings, which will then be tracked on a mobile app. Through the use of advanced algorithms, “Nurture” will know when to water and fertilize the plant without any human input. Ultimately, the device exists to help streamline the plant growing process by removing the time-consuming aspects of plant growing and preventing any health issues the plant may experience.

1.2 Intended Users

This product will be useful to anyone who wishes to grow plants. However, “Nurture” is mainly targeted toward hobbyist gardeners. This is because of two reasons. Our device is being designed with durability and cost-effectiveness first, and absolute accuracy second. A hobbyist will likely not be looking for scientific accuracy but will be looking for something relatively inexpensive. Making the device appealing to hobbyists but less appealing to others who grow plants professionally, such as farmers and scientists. Additionally, our device's features are centered around convenience with the functionality to dispense water and fertilizer automatically. The device appeals to those who have limited knowledge of plant care and who are looking to minimize effort to care for plants, namely hobbyists.

2 Requirements, Constraints, And Standards

2.1 Requirements & Constraints

2.1.1 Requirements

Physical requirements:

- Complete setup, water/fertilizer disbursement system, and sensor/microcontroller setup.
- The total device should be able to sit next to the pot with all sensors inside the pot placed next to the plant. The device should also be able to fit with pots that are larger than 3 inches across.

UI Requirements:

- On the app, the user must be able to access sensor readings for individual plants in both graphical and numeric formats. i.e. graphs detailing a sensor reading vs time, as well as current readings of particular sensors.
- Users should be able to create an account with persistent data about their account and plants.
- User data will be stored online in a database and should be accessible from the user's phone so long as they have access to the internet.

User Experiential Requirements:

- The device should be able to be turned on and forgotten for long periods not having to be refilled often.
- Sensor readings for all devices should be updated at least once a day.
- The app should be reliable and fast, meaning it should not crash unexpectedly and it shouldn't freeze up.

2.1.2 Constraints

Size:

- The water reservoir should be able to store at least one week's worth of water, roughly 600ml.

Power:

- Power is delivered through a 12V AC adapter to the device next to the pot.

Cost:

- The total material cost of the device should not exceed $500.

2.2 Engineering Standards

- 802.11ac WiFi Standard: Most devices communicating via WiFi today utilize this standard. Our project includes the WiFi module on the Arduino MKR 1010 WiFi, which communicates with the server. The server communicates with the user's phone while the phone runs the app.
- IP55 or better dust and water resistance rating on the device enclosure.
- RoHS Compliance: Relates to the restriction of hazardous substances in electronics. The custom PCB in our design uses a lead-free HASL finish and other RoHS-compliant components.
- ASME B18.2.8 Metric Clearance Hole Sizes: Integrated into the design for the PCB and enclosure for the use of metric screws.

3 Project Plan

3.1 Project Management/Tracking Procedures

The management style we chose for our project is a hybrid waterfall/agile project management strategy. The waterfall approach was used regarding hardware design, which was costly enough to eat up our project budget, making it difficult to reverse design decisions. This quality necessitated a protracted analysis and planning period and prevented us from returning to previous design steps, making a waterfall-based strategy seemingly the best option. The agile approach was used in software development, which can be continuously tested, retooled, and redeployed.

Informal project communication occurred via Discord due to its versatility, allowing for easy VOIP communication and image sharing to quickly communicate ideas. Formal software progress was stored on the provided team git lab repository, and hardware progress was shared via the team discord.

3.2 Task Decomposition

- Task 1: Complete the user interface design and implement it in React Native
    1. Design UI in Figma for the app and determine software to implement frontend and backend
    2. Get a software development environment setup for React Native
    3. Develop the UI in React Native
    4. Test UI
- Task 2: Set up MongoDB backend and complete a round trip through the Arduino and app.

1. Create a model of the database structure
2. Become familiar with MongoDB and how to use their hosting services
3. Develop database schema
4. Connect the app to the MongoDB Atlas database with the Express & Mongoose server running on Glitch
5. Deploy Express & Mongoose on a server so the database can be accessed without a local host
6. Get the Arduino to store data on the MongoDB Atlas using the Express & Mongoose server
7. Complete round trip and test functionality
8. Research needs of other plant types.
9. Install and link new sensors.
10. Update database and backend code.

- Task 3: Implement the necessary hardware for the device to work as intended. Once functionality has been established, create a PCB that will handle all design and functional requirements.
    1. Sensor Selection
    2. Work on receiving valid sensor data
    3. Actuator Selection
    4. Work to control water/fertilizer release with actuators
    5. Breadboarding circuit to incorporate sensor and actuator power.
    6. Ensure sensor data can be read and formatted in a form that can be useful. Initial software to command actuators.
    7. Connect the backend to Arduino to get sensor data and complete the round trip.
    8. Integrate the circuit within a waterproof enclosure.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Task 1 Criteria:

- App mockup is finished
- UI works as intended

Task 2 Criteria:

- Database structure complete.
- Database successfully deployed on Glitch server
- The app can update the MongoDB database through the server hosted on Glitch

Task 3 Criteria:

- Receive accurate, understandable data from sensors
- Send/receive parseable RESTful requests to the server
- Utilize actuators to release water/fertilizer in a controlled manner
- Custom PCB is created
- Actuators are activated by threshold values received by the user via the app
- The final working circuit is created

- Circuit secured within a waterproof enclosure

3.4 Project Timeline/Schedule

Task 1: Complete the User Interface Design and Implement it in React Native

February:
Week 1-4: Design UI in Figma for app and determine software to implement frontend and backend.

March:
Week 1: Set up software development environment setup for React Native.
Week 2-4: Develop the UI in React Native.

April:
Week 1-4: Develop the UI in React Native.

May:
Week 1-2: Develop the UI in React Native.
Week 3-4: Test UI.

August:
Week 2-3: Fix Node.js package conflicts.
Week 3-4: Fix app bugs with user management and updates.

September:
Week 1: Revamp user management to reliably refresh user data.
Week 2-4: Implement single press and long press actions for plant cards and plant detail sensor cards to allow more user control in an intuitive way.

October:
Week 1-3: Add backend functionality to handle single and long press settings.
Week 4: Update database columns/backend code.

December:
Week 1: Implement a screen to change the name and type of your plant.

Task 2: Set up MongoDB, Test it, and Complete a Round Trip through Arduino and the App

March:

Week 1: Get familiar with MongoDB and how to host the database.
Week 2-3: Develop backend.
Week 4: Host the MongoDB database on a server and have a set of RESTful endpoints.

April:
Week 1-3: Deploy the backend with hosting services and make breakpoints.
Week 4: Work on connecting the mobile app to the backend and Pi.

May:
Week 1-3: Connect the mobile app to the backend and Pi.
Week 1-2: Complete round trip and test functionality.

August:
Week 1-4: Research needs of other plant types.

September:
Week 1-2: Research needs of other plant types.
Week 2-4: Install and link new sensors.

October:
Week 1: Install and link new sensors.
Week 1: Update database columns/backend code.

Task 3: Implement the necessary hardware for the device to work as intended. Once functionality has been established, create a PCB that will handle all design and functional requirements.

February:
Week 3-4: Sensor functionality.

March:
Week 1-2: Sensor functionality.
Week 2-4: Actuator Selection and Individual Testing
Week 4: Breadboarding circuit to incorporate sensor and actuator power.

April:
Week 1-4: Breadboarding circuit to incorporate sensor and actuator power.
Week 1: Actuator Selection and Individual Testing

Week 2-4: Ensure moisture sensor data can be read and formatted in a form that can be useful. Initial software to command actuators.

May:
Week 1-2: Continue with ensuring moisture sensor functionality.

August:
Week 2-4: Review documentation and plan initial setup and goals for the second semester.

September:
Week 1-3: Setting up RS485 functionality with microcontroller.
Week 2-4: PCB design planning and research. Compiling initial list of on-board components.

October:
Week 1-2: Continue working on NPK sensor functionality through firmware development.
Week 2-4: Explore microcontroller alternatives as necessary. CompleteRS485 integration with TTL converter.

November:
Week 3: Software to read sensors completed. Data was successfully sent to the server.
Week 3-4: PCB design finished, tested, and integrated.

December:
Week 1: Finalize enclosure with all required components.

3.5 Risks And Risk Management/Mitigation

Risks:	**Risk Factor**
● Watering system interfering with electronics	0.5
● Selected sensors do not work with our system	0.8
● Database data is lost	0.5

Mitigation Strategies:

- Utilize waterproof enclosures to keep electronics and the watering system separated.
    - Pump system to allow for water reservoirs further away from electronics.

- Research sensors very thoroughly before buying to avoid financial loss.
    - Give preference to sensors that have datasheets.
    - Compare datasheets: voltages, frequencies, communication protocol, etc.
    - When in doubt, ask other team members or project advisors.
- Perform regular data backups.

Risks that occurred:

- Some selected sensors did not integrate well into our system. The Adafruit moisture and temperature sensor we used had no water resistance and exposed electronics. We ultimately had to select a different NPK sensor with moisture and temperature as additional metrics.

3.6 Personnel Effort Requirements

Task	**Subtask**	**Estimated hours**
Task 1: UI Design and React Native Implementation	Design UI in Figma	12
Task 1: UI Design and React Native Implementation	Determine software for frontend and backend	2
Task 1: UI Design and React Native Implementation	Setup React Native development environment	1
Task 1: UI Design and React Native Implementation	Develop UI in React Native	30
Task 1: UI Design and React Native Implementation	Test UI	5
Task 2: MongoDB Express Backend and Microcontroller Integration	Model database structure on paper	1

Task 2: MongoDB Express Backend and Microcontroller Integration	Learn MongoDB and Express	2
Task 2: MongoDB Express Backend and Microcontroller Integration	Develop schema for user data	3
Task 2: MongoDB Express Backend and Microcontroller Integration	Host MongoDB and Express on the cloud	1
Task 2: MongoDB Express Backend and Microcontroller Integration	Connect the app to the backend and Arduino	4
Task 2: MongoDB Express Backend and Microcontroller Integration	Research plant type needs	2
Task 2: MongoDB Express Backend and Microcontroller Integration	Complete round trip and test functionality	5
Task 2: MongoDB Express Backend and Microcontroller Integration	Install and link new sensors	10
Task 2: MongoDB Express Backend and Microcontroller Integration	Update database and backend code	10
Task 2: MongoDB Express Backend and Microcontroller Integration	Test extended functionality	2
Task 3: Hardware Implementation and PCB Design	Select sensors	10

Task 3: Hardware Implementation and PCB Design	Validate sensor data	5
Task 3: Hardware Implementation and PCB Design	Control water/fertilizer release with actuators	10
Task 3: Hardware Implementation and PCB Design	Breadboard sensor and actuator circuit	2
Task 3: Hardware Implementation and PCB Design	Format sensor data for the backend	3
Task 3: Hardware Implementation and PCB Design	Connect hardware with the backend	25
Task 3: Hardware Implementation and PCB Design	Custom PCB Created	20
Task 3: Hardware Implementation and PCB Design	Circuit secured within a waterproof enclosure	10

Table 1: Personal Effort Requirements

3.7 Actual Effort Requirements

Task	**Subtask**	**Actual Hours**	**Mismatch**

Task 1: UI Design and React Native Implementation	Design UI in Figma	12	
Task 1: UI Design and React Native Implementation	Determine software for frontend and backend	10	
Task 1: UI Design and React Native Implementation	Setup React Native development environment	2	
Task 1: UI Design and React Native Implementation	Develop UI in React Native	30	
Task 1: UI Design and React Native Implementation	Test UI	5	
Task 2: MongoDB Express Backend and Microcontroller Integration	Model database structure on paper	1	
Task 2: MongoDB Express Backend and Microcontroller Integration	Learn MongoDB and Express	2	
Task 2: MongoDB Express Backend and Microcontroller Integration	Develop schema for user data	3	
Task 2: MongoDB Express Backend and Microcontroller Integration	Host MongoDB and Express on the cloud	1	
Task 2: MongoDB Express Backend and Microcontroller Integration	Connect the app to the backend and Arduino	12	8
Task 2: MongoDB Express Backend and Microcontroller Integration	Research plant type needs	5	

Task 2: MongoDB Express Backend and Microcontroller Integration	Complete round trip and test functionality	30	28: Significantly longer integration time
Task 2: MongoDB Express Backend and Microcontroller Integration	Install and link new sensors	10	
Task 2: MongoDB Express Backend and Microcontroller Integration	Update database and backend code	10	
Task 2: MongoDB Express Backend and Microcontroller Integration	Test extended functionality	2	
Task 3: Hardware Implementation and PCB Design	Select sensors	15	5: Replaced all initial sensors.
Task 3: Hardware Implementation and PCB Design	Validate sensor data	5	
Task 3: Hardware Implementation and PCB Design	Control water/fertilizer release with actuators	5	5: Simpler than expected
Task 3: Hardware Implementation and PCB Design	Breadboard sensor and actuator circuit	7	5
Task 3: Hardware Implementation and PCB Design	Format sensor data for the backend	3	
Task 3: Hardware Implementation and PCB Design	Connect hardware with the backend	25	
Task 3: Hardware Implementation and PCB Design	Custom PCB Created	20	30: Additional revisions

Task 3: Hardware Implementation and PCB Design	Circuit secured within a waterproof enclosure	10	

3.8 Other Resource Requirements

Prototyping Components:

- **Arduino MKR 1010:** The central microcontroller for sensor data processing and actuator control.
- **Soil and Plant:** Essential for real-world testing of soil sensors.
- **NPK Sensor:** For measuring soil composition, including nitrogen, phosphorus, potassium, temperature, moisture, and pH.
- **PAR sensor:** To monitor ambient light levels affecting plant growth.
- **Actuators:** Solenoid valves or similar mechanisms for water and liquid fertilizer dispensing.

Hardware Assembly and Enclosure:

- **Enclosures:** Cases to house the electronics with modifications for sensor and actuator mounting.
- **Relays and Wiring:** For interfacing actuators with the microcontroller.

Connectivity and Control:

- **Power Supplies:** Adequate for powering the Arduino, sensors, and actuators.
- **PCB:** Custom board for neatly organizing and connecting electronic components for power distribution.

Supplementary Materials:

- **Tubing and Fittings:** For constructing the water and fertilizer dispensing system.
- **Fasteners and Mounting Hardware:** For securing components within the enclosure.

Testing Supplies:

- **Testing Equipment:** Tools like multimeters and oscilloscopes for circuit testing.
- **Consumables:** Solder, wire, and other materials for assembly and maintenance.

Software and Development:

- **Development Environments:** For programming the Arduino and MongoDB Express backend.
- **Mobile Development Framework:** Such as React Native for app development connected to the hardware.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Effect
Public health, safety, and welfare	Focused on the system's core purpose of reliable plant monitoring and automated care, which directly impacts user success in gardening.	This device streamlines the gardening process. Helping individuals access fresh vegetables ultimately acting as a benefit to public health.
Global, cultural, and social	Emphasized the bridge between technology and traditional gardening, acknowledging both aspects rather than viewing them as conflicting.	Some may feel a device like this is taking the “human touch” away from gardening.
Environmental	Highlighted both the resource optimization benefits and the reality of electronic component impact, providing a balanced view.	While stimulating plant growth will help with global co2 emissions, the individual parts still require manufacturing and shipping, each which have a negative impact on the environment.
Economic	Repositioned the economic impact to focus on the project's target market and value proposition rather than viewing it as an unnecessary purchase.	This device helps perform a service that most people are capable of performing on their own. This means that consumers are buying something they may not strictly need by buying this product.

Table 2: Broader Design Context

4.1.2 Prior Work/Solutions

Multiple products involving soil data collection paired with a mobile app exist on the market currently. Three such examples are Diivo, Planta, and Sinbeda.

- Diivo Smart Soil Moisture Meter [1]
    - Device connects to a mobile app via Bluetooth.
    - Small form factor, fits in the palm of one's hand.
    - Simple setup: Insert the device into the soil and press a button.
    - No soil nutrient monitoring.
    - No automatic watering or fertilizing.
    - Cost: ~$15
    - The system is shown in Figure 1.
- Planta [2]
    - Camera usage for plant identification and light measuring.
    - Plant illness identification.
    - Community section of mobile app for social media posts.
    - No external hardware is required other than a phone.
    - No automated care.
    - No way to measure soil nutrients.
    - Cost: Free/In-app purchases
    - App store information is shown in Figure 2.
- Sinbeda [3]
    - Measures soil moisture, temperature, light intensity, and soil nutrients.
    - Database of 6000+ plants.
    - Battery button cell for power.
    - The app provides users with tailored plant care advice.
    - Bluetooth connection rather than WiFi.
    - No automatic watering or fertilizing.

Figure 1: Diivo Smart Soil Moisture Meter Hardware and App

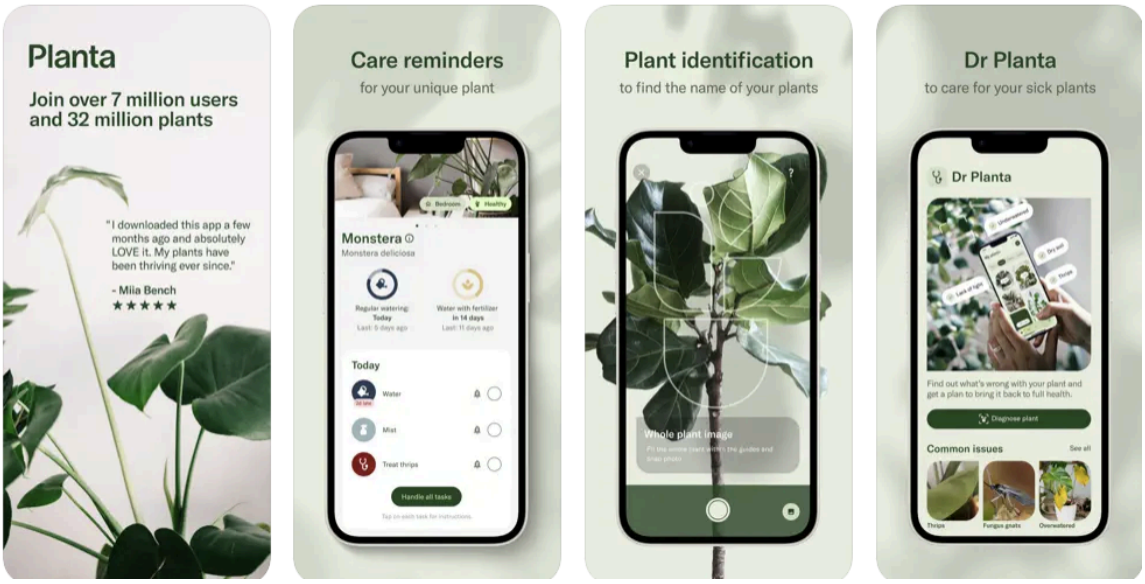

Figure 2: Planta Mobile App Advertisement

Figure 3: Sinbeda Plant Care System

4.1.3 Technical Complexity

Hardware:

1. The design centers around a microcontroller that can interface and format data from a variety of sensors, each of which communicates through Modbus protocol.
2. Designing the overall hardware system requires thorough component selection, comparison, and testing.
3. PCB Design to optimize performance, cost, and form factor requires an understanding of many electrical engineering principles. Additionally, component selection, schematic creation, and cross-checking datasheets play a role in this aspect of the project.

Software:

1. The mobile app's front end requires pages for login, overview, individual plant cards, data visualization through graphs, and measurement explanations.
2. The system's backend must communicate with the server and access user data.
3. Low-level embedded programming is necessary for receiving and formatting data from sensors and activating actuators based on commands sent from higher levels in the project.

4.2 Design Exploration

4.2.1 Design Decisions

1) **Sensor selection:** Choosing the correct type of sensor is key for ensuring the success of the project, as our project is built around sensor data. Choosing the wrong sensor could result in inaccurate data being collected, unnecessary data being collected, or no data being collected at all. Initially, we started with two sensors with limited capabilities: the initial NPK sensor and the moisture/temperature sensor. While these sensors read relevant data concerning plants, they ultimately did not read a wide enough variety of values and were exchanged for the more advanced new NPK sensor and the PAR sensor, which read a much wider variety of values relevant to plant growth.

2) **Server type:** The server is the hub for processing and storing data. It manages the data sent to and from devices and provides the app with the needed sensor data. A reliable server ensures smooth communication and uninterrupted data flow, which are critical for the app's success. The Glitch server was chosen because it is cost-effective, user-friendly, and meets our need for managing plant sensor data using Express and Mongoose with MongoDB for storage.

3) **App platform:** The decision of whether to make the app for Android, Apple, or both presents a trade-off. If our team only focused on one platform, there would be more time to finish other aspects of the project; however, this decision would also ignore a section of the user base. Because of this, carefully considering which platforms to develop to appeal to a wide audience and save time was important.

4) **Microcontroller:** Initially, we chose a Raspberry Pi Pico due to its low cost, relatively high performance, and extensive features. However, as time went on it became clear that it was not capable of supporting our needs without significant time investment for a custom implementation of existing protocols. The decision was made to move to an Arduino platform that was more expensive and had fewer features on paper but was far better suited to our needs. Requiring only a few hours to do what the Pi failed to achieve in several months, thanks to Arduino's vastly superior library support. The Pi's extra features proved to be largely irrelevant.

4.2.2 Ideation

Through the lotus blossom technique, we expanded our focus on what should be considered when selecting sensors. We considered five options for this design decision:

1) Overall sensors related to plant care: moisture, temperature, NPK, pH, etc.
2) Selecting sensors best suited for a specific base-case test plant we select.

3) Sensors that can be calibrated according to soil contents.
4) Basing sensor selection around the most essential nutrients generally needed by plants: nitrogen, phosphorus, and potassium.
5) Higher accuracy sensors as opposed to more cost-effective sensors to meet the needs of our key demographic.

4.2.3 Decision-Making and Trade-Off

Our team focused on the pros and cons of each aspect of sensor selection to make our decision. Selecting sensors generally applicable to plants would allow us to accommodate a wide variety of plants but may not be the best at monitoring any specific plant's health closely. Focusing on selecting a base case causes too narrow of a selection for an application that is meant to be general. Because of this, we determined that sensors for general care would be best. Regarding general care, nutrient sensors and sensors that can be easily calibrated would be ideal with limited trade-offs.

While using many sensors would improve data collection, this would conflict with our budgetary constraints. With a focus on a general audience of users, budget is a key factor in hardware selection. Our team ultimately decided that a limited number of mid-range cost sensors would be best.

4.3 Final Design

4.3.1 Overview

Our design features three main components: the device, servers, and the app. The device reads the soil's temperature, moisture, and nutrient data and the ambient PAR light value and then stores it within the database. A user can then view this sensor data numerically and graphically on the app. After the specified threshold value is crossed, the microcontroller operates the actuators to dispense the needed liquid from the reservoirs.

4.3.2 Detailed Design and Visual(s)

Hardware:

High-Level:

Figure 4 displays the conceptual flow of information and control within our system. The Arduino and server act as the bridge between the hardware and software aspects of the project. Data and control will flow between the user's device and Arduino through this server.

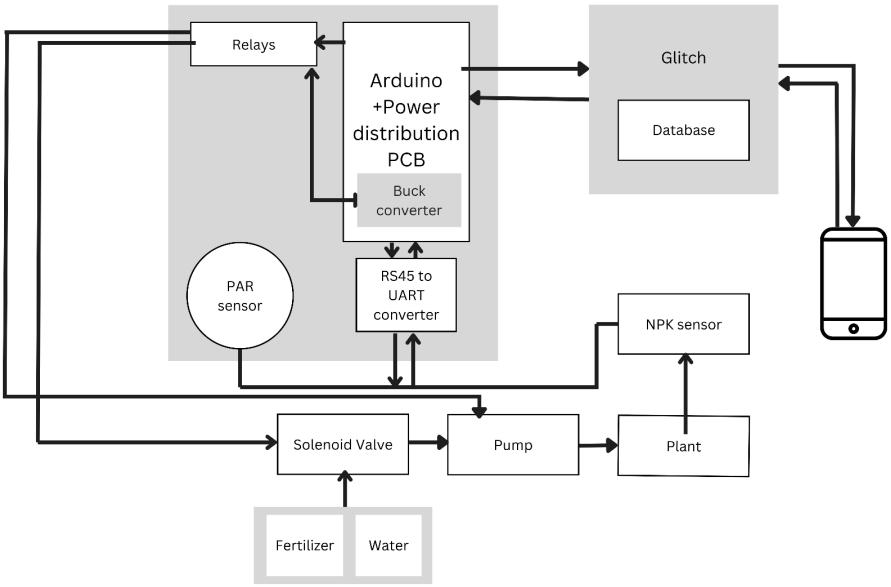

Figure 4: Block diagram of the overall system

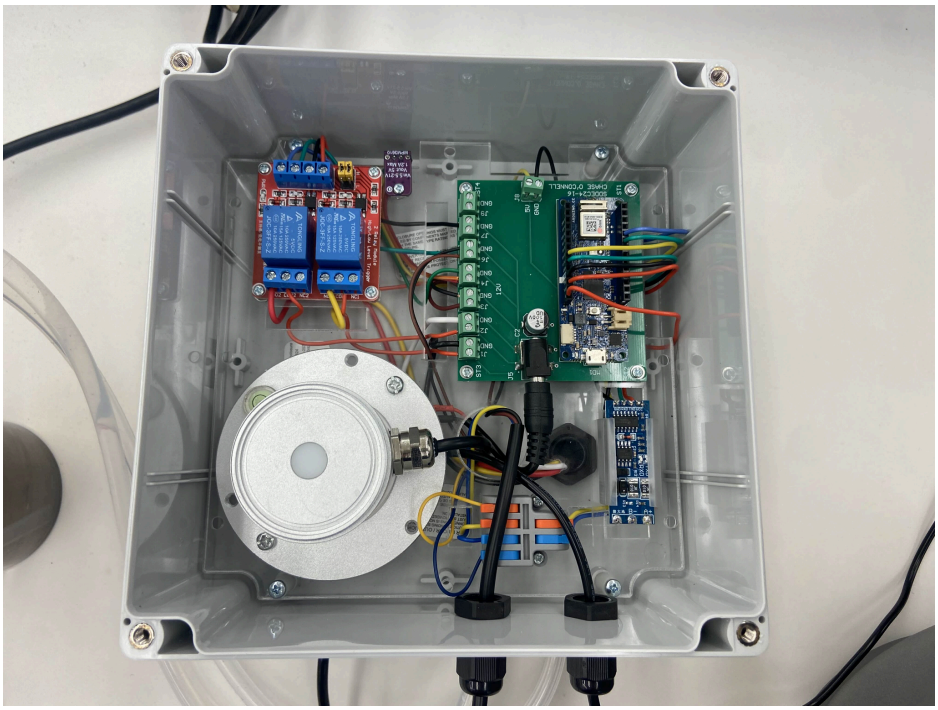

Figure 5: Picture of the electrical setup

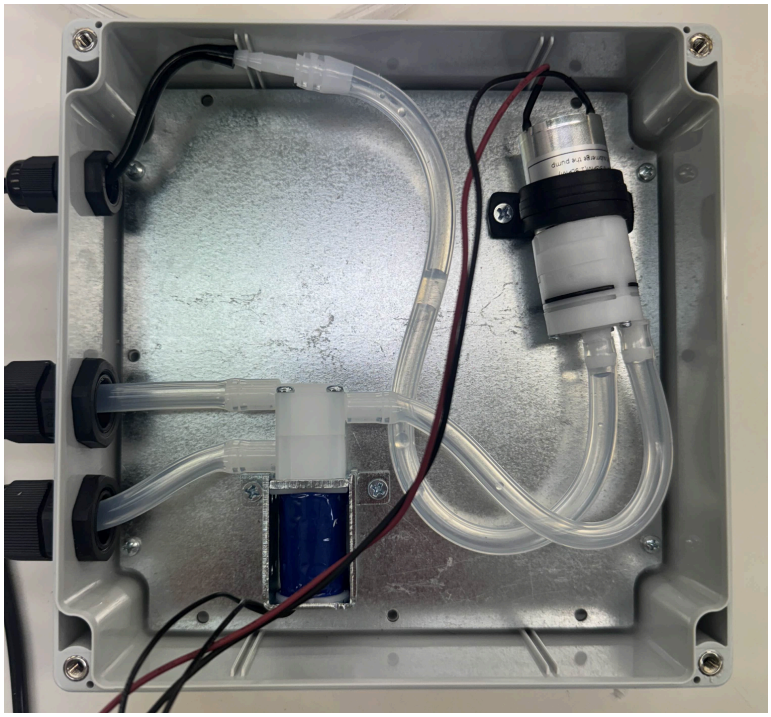

Figure 6: Picture of the pump solenoid valve setup

Our device incorporates an Arduino Maker wifi 1010 as the main microcontroller and method for low-level data handling. Peripherals connected to the Arduino include an NPK sensor, PAR sensor, RS485 to UART converter, relays for liquid pumps, a solenoid valve, and a buck converter to supply the necessary power to the microcontroller.

Sensor reading: Both the NPK sensor and the PAR sensor communicate with rs485 Modbus and are daisy-chained in the same Modbus network. As the Arduino does not natively support RS485, a Modbus RS485 to UART converter is used as a bridge between the two devices, allowing communication. To facilitate this communication, the libraries ArduinoModbus and ArduinoRs485 are used to both interpret and write Modbus data frames that have been converted into UART data. Both sensors are powered by the 12V port embedded in the PCB.

Pump Control: As the code reading the sensor data is run, moisture and nutrient values will be tested to see if they pass a threshold determined by the user on the app, and if so, they will activate the pump system, adding either water or fertilizer. To do this, certain GPIO pins on the Arduino are set high, activating one of two relays: the pump relay and the solenoid valve relay. If fertilizer is called for, then the 12-volt signal passes through the relay to activate both the pump and the solenoid valve, allowing for fertilizer to pass

through. If water is called for, the 12-volt signal only travels to the pump, allowing water to pass through.

Communication with the server: In order to send data to the server, the Arduino uses the WiFiNINA library, which allows the Arduino to establish a connection to the server and deliver HTTP requests through printing to the “WiFiCLIENT” struct, something defined within the WiFiNINA library.

Software:

The software system consists of a mobile application, app for short, a server, and a database. The mobile application is with React Native, Expo, and Expo Go, all of which are Javascript frameworks, and together, they allow using an iPhone as your emulator, and since our software team didn't have Mac’s, this was essential for development. The server consists of Express and Mongoose, Node.js for short, and the server interprets our user data and sends it for storage in the database. The server stores the schema for our data, so that's why it's able to interpret it. Our database, MongoDB, doesn't understand our schema; it only stores what it receives. The server and the app are both Javascript frameworks, allowing seamless communication through Javascript Object Notation (JSON). Server app communication can be tricky because both the app and the server need to understand one another, but since ours both speak JSON, this problem is nullified.

The Mobile Application Features:

The app ensures secure access through its login functionality, as depicted in **Figure 7**. After logging in, users are greeted by the **home screen**, a central hub providing access to key actions such as viewing plant profiles, refreshing data, and navigating to the settings screen. The simplicity of this design supports users with varying levels of technical proficiency.

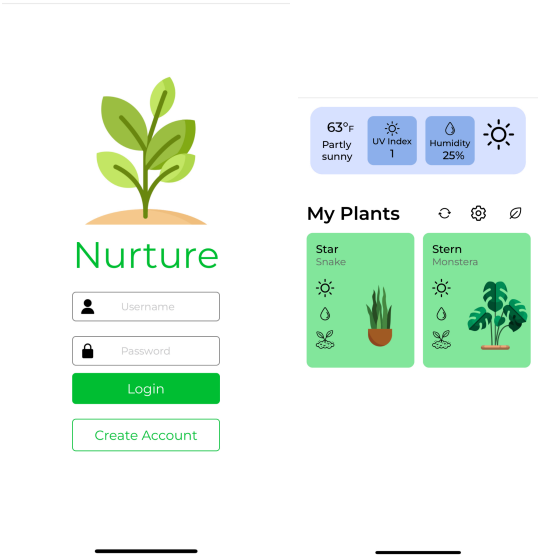

Figure 7: The left panel shows the login card, while the right panel displays the home screen, where users can manage their plants or navigate to other app functionalities.

The **home screen** serves as where users can access the app's core functionalities. It is designed for simplicity and efficiency, ensuring that all actions are available. A refresh button is placed on the home screen, allowing users to sync the app with the latest data from the server, allowing users access to the most up-to-date information. The Settings button provides access to the account management screen (Figure 10), where users can update their username and password or log out of their account securely. Ambient condition data, such as humidity, UV index, and temperature, are displayed directly on the home screen. This integration with external APIs ensures that users are informed about the environmental context of their plants, complementing the sensor data provided in the Plant Detail screen.

Editing or deleting plants is made intuitively through a **long press** gesture on any plant card, as shown in Figure 8 (right). This action opens a popup menu where users can select the desired action, reducing the need for excessive buttons and maintaining a clean interface.

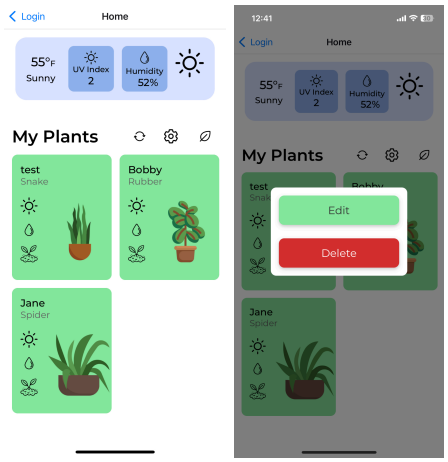

Figure 8: The left panel shows the layout of the home screen, while the right panel demonstrates the result of a long press on a plant card, opening a popup for editing or deleting plant profiles.

Users can add new plant profiles by tapping the **Create Plant** button, which navigates to the **Create Plant screen** (Figure 9). Here, users can input details such as the plant's name and species.

Tapping on a plant card opens the **Plant Detail screen** (Figure 11), where users can access sensor readings, interactive graphs, and threshold settings. This allows users to get specific plant data with minimal effort.

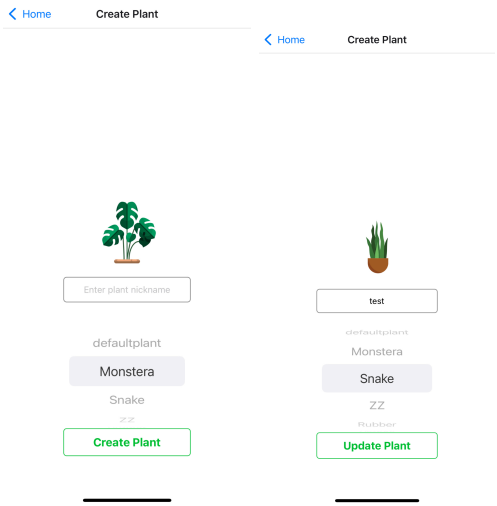

Figure 9: The Create Plant screen has functionality for both creating and editing a plant's name and species.

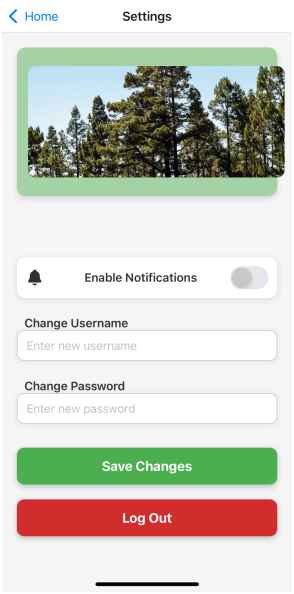

Figure 10: The settings screen has the functionality to log out or change your username and password.

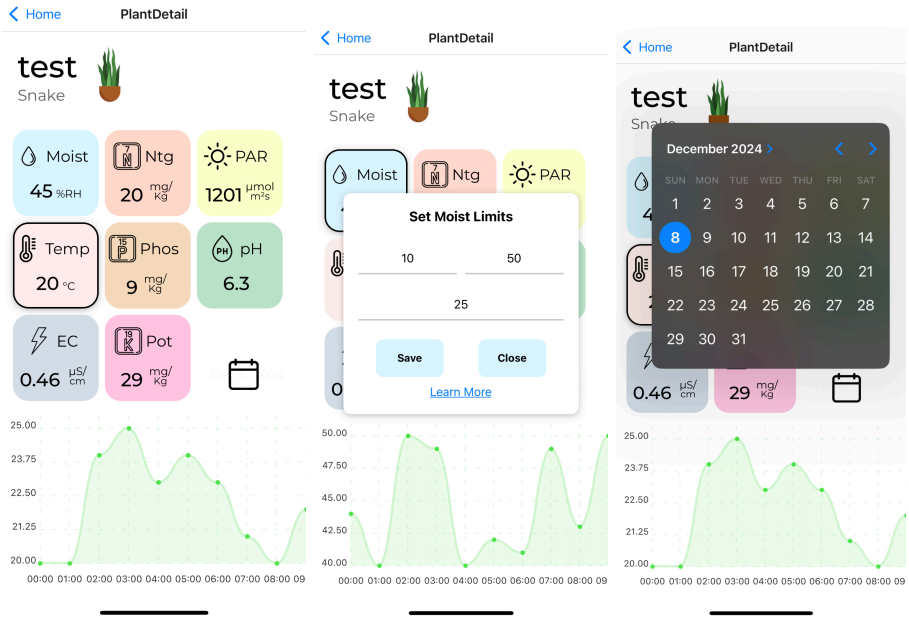

Figure 11: The Plant Detail Screen features cards displaying the most recent data from a sensor, a graph of sensor data, and a calendar button. Each of these cards has a single press and long press functionality.

Long Press: opens a menu for inputting threshold values min max and hold as is seen within the middle image.

Single Press: changes the sensor that the graph displays at the bottom result can be seen when comparing the left image to the middle image.

Calendar Button: The calendar button opens a calendar popup for selecting a date. Selecting a date will display sensor data from that day.

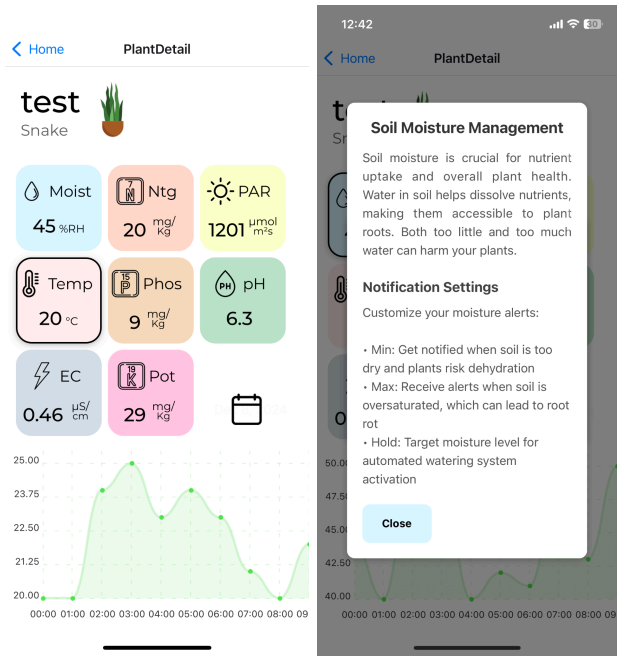

Figure 12: Plant sensor screen (left) and the result of a long press on the sensor detail cards (Temp) nutrient description card (right).

The Plant Detail screen gives users an in-depth view of their plant's sensor data, offering real-time readings, interactive graphs, and customizable settings to enhance plant care. The screen is designed to prioritize usability, ensuring that users can access and interact with data efficiently.

Sensor data is displayed on individual cards, each representing a different measurement, such as NPK levels, soil moisture, PAR, and temperature (Figure 11). These cards present the most recent sensor readings, while the graph displays them over time.

Customizing sensor thresholds is made easy with a long press on any sensor card. This action opens a menu where users can set minimum, maximum, and hold values for each parameter (Figure 11 middle). These thresholds allow users to tailor alerts to their plants' specific needs.

The Plant Detail screen also integrates educational content to enhance user understanding. A long press on a sensor card can display additional information, such as nutrient descriptions, explaining how specific parameters impact plant health (Figure 12, right). This feature transforms the app into a learning tool, equipping users with actionable knowledge for optimal plant care.

4.3.3 Functionality

The user's role would be relatively simple: after purchasing the device and downloading the app, they would have to plug the device's main power adapter into a wall outlet. The user would then place the device adjacent to the plant they intend to monitor and then probe the soil around the plant with the prongs of the NPK sensor. Next, they would open the app, create an account and a profile for the plant they are monitoring, sync their device with their account, and complete the setup stage. The user would then be free to forget the device, periodically checking sensor values and nutrient and water levels. The goal of our system is to allow for a hands-off approach from the user.

4.3.4 Areas of Concern and Development

One major concern in development was keeping the overall price per unit of the device low, due to this being a hobbyist device, it was accepted that if the device were too costly, it would drive away the target demographic. However, it was also accepted if the components of the device were too cheap then the usefulness of the device would be sacrificed. To deal with both of these issues, middle-range components were chosen, for example, sensors that lacked scientific precision yet were still reliable and offered a variety of values they could read.

Another key issue was getting the microcontroller to read in Modbus data. On the initially chosen microcontroller, the Raspberry Pi Pico communicating with the Modbus sensors required a lot of effort as there were no readily available libraries for interpreting for creating Modbus data frames. To rectify this problem, the microcontroller was swapped out with the currently used Arduino, which did have a library for working with Modbus data frames.

Our design fits user needs well. We can drive the per-unit price down using reasonably priced sensors and actuators. Although there is a trade-off with sensor precision due to the relatively low cost of the sensors, our primary audience of hobbyist gardeners will likely prefer the lower costs of the product over absolute precision. Additionally, due to the nature of eliminating a time-consuming part of gardening, our product appeals to non-professionals who likely do not want to invest as much time into managing plants as a professional farmer.

4.4 Technology Considerations

Our group has selected two sensors: an NPK sensor and a PAR sensor, both of which communicate via rs485 Modbus. However, the microcontroller cannot receive Rs485 data, and thus, a bridge is required between the microcontroller and the sensors. As this is the

only NPK sensor within our price range, we found this to be an acceptable trade-off. Our group also uses the MongoDB database to house data on our Glitch server. This technology is scalable, efficient, and easy to use, so there are few drawbacks.

4.5 Design Analysis

Successfully developed a working backend integrated with the frontend app that reads user and sensor data from the database. The database was deployed and operated correctly on a Glitch server with storage on MongoDB. All components were successfully connected, resulting in a prototype validating our proposed solution in 4.3. This prototype demonstrated not only that the individual sub-components of the device functioned as intended but also that they worked as intended together. Our design effectively used libraries such as the Python requests library and Volley, both known to perform well in this type of setting.

5 Testing

5.1 Unit Testing

- Interpretation of Sensor Data: To ensure that correct data is being collected, the code for reading sensor data will be run, and the resulting values will be interpreted on the computer that uploaded the code to the Arduino IDE's console.
- Backend Functionality: Sending "post" and "put" commands to the server to send and update the database. Get commands to the server via Postman to ensure the database correctly receives and interprets these commands.
- Frontend Functionality: To ensure that the local structure of the front end is working, the user will menu through each screen and button to ensure none are broken.

5.2 Interface Testing

Two significant interfaces within this system were tested. The communication between the Arduino and the server, the server and the mobile app

Arduino to Server:

To test proper functionality between these two components, the Arduino sends "POST" commands to the server via the Request library. The server's contents were then analyzed through a "get request" via Postman or the app.

Server to App & App to Server:

The connection between the mobile app and the server was tested by sending post requests from the app to the server to create users and plants. The app-to-server connection will then be validated by using Postman to perform a get request on the users, showing the users where it is easy to tell if the changes done in the app are present on the database. To test if the server can send data to the mobile app, Postman will be used to create a user with plant and plant data. Then, on the app, the created user will be logged in, which sends a get request to the server, and the server responds with a user object that is then stored locally. The plant created on the app should be displayed, and when clicked, plant data should be shown in a graph.

5.3 Integration Testing

A critical integration path in the design is the integration of the microcontroller and the server. Without these two components communicating, no data may be viewed on the mobile app, which would defeat the device's purpose. This was tested by sending restful commands through the Arduino to the server. The server was then queried through Postman or on the app itself to see if the data had been properly received and interpreted in the database.

5.4 System Testing

To test the entire system, “full loop” tests were performed. Starting with the sensors sending data to the Arduino which then interpreted the sensor data and sent it to the server. The server should then collect that data and store it in the database. A user should then be able to create a new account or log in and view that data on their screen. The user should then send a command to the Arduino to set a threshold value for watering or fertilizing, resulting in the soil being watered to stay above the threshold. This system should allow the testing of both the interconnectivity but also the local capabilities of each component.

5.5 Regression Testing

Each time a new component is added, unit tests are run to ensure that that component is working properly. More thorough testing is unnecessary as most components in the system do not affect the functionality of other components other than the ability to pass along data. For example, if a sensor is replaced, it will not affect how effectively the Arduino can transmit data, but it may act as a bottleneck toward data being transmitted if it does not work.

5.6 Acceptance Testing

A “full loop” test had to be completed to verify that all functional requirements were met. The sensors needed to pick up data from the soil, and then the data picked up by the sensors needed to be interpreted by the microcontroller and sent up to the server, which then will receive and store the data. A user on the mobile app needed to make an account, also stored on the server, and request the soil data, which was then to be displayed on the app. The user needed to then send a command to the Arduino to set a moisture or nutrient threshold. The Arduino then needed to activate its GPIO and activate the corresponding pump when that threshold value was crossed. This will satisfy the functional requirements.

Most nonfunctional requirements will be solved in the part selection process such as the device size being reasonable next to 3+ inch. diameter pots and the price of the device not exceeding $500.

5.7 User testing

As the device was completed very recently, true user testing was rendered impossible; however, a valid user test would be to grow a plant with the device to see if it fulfills its purpose, that is to automate plant growth

5.8 Results

These tests have been crucial in assessing the functionality, reliability, and user interface of our system. This revised summary reflects the functionalities and outcomes based on the latest system specifications.

Unit Testing:

Unit testing confirmed that the NPK and PAR sensors function correctly, showing data from the moisture and temperature sensors. A sample of this raw data can be seen in Figure 8. The backend tests demonstrated that the MongoDB database effectively handles data storage and retrieval. Frontend tests confirmed that the app's user interface components function as designed. Tests with the communication between the Arduino and the server reveal that we can exchange data effectively from the Arduino to the server.


```
{"nitrogen": 0,"phosphorus": 0,"potassium": 0,"moisture": 0,"temperature": 257,"ph": 30,"salt": 0,"par": 0,"ec": 0}
115

HTTP/1.1 201 Created
Date: Sat, 16 Nov 2024 22:07:55 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 1012
Connection: close
x-powered-by: Express
etag: W/"3f4-5+H+8KvZxgkajN7Sj+LzD3U2N+M"
```


Figure 13: Raw Data Sample from the Temp. and Moisture Sensor

Interface Testing:

The microcontroller successfully retrieves moisture nutrient and light data and successfully communicates with the server

The mobile app effectively communicates with the MongoDB database to fetch and update data.

Integration Testing:

Integration testing highlighted a lack of direct communication between the microcontroller and the server; however, the system compensates with effective indirect data handling through the Python code on the Glitch server. This setup allows for timely updates and interactions via the mobile app.

Integration testing revealed all separate elements of the system working correctly simultaneously

System Testing:

The system performs a “full loop” in adequate time with no real delay. With the omission of communication from the server to the device.

Regression Testing:

Regression tests confirmed that updates or changes to the system components did not negatively impact overall functionality. The tests showed that each component could operate independently.

Acceptance Testing:

Multiple “full loop” tests were completed successfully indicating a functionality is in alignment with stated requirements, again with the omission of communication from the server to the device.

6 Implementation

Our final design aligns well with our requirements as each individual component is capable of communicating with the other effectively as well as performing their individual functions. This includes the device itself being able to accurately read soil and light data

and then transmit and store that data in the database via HTTP request. That data is able to be graphically displayed in the app.

Some initially planned systems were not able to be implemented such as the use of websockets for moment-to-moment communication, due to time constraints. Along with this, the functionality to manually command the device to begin watering was not completed due to later decisions to focus on the autonomous aspect of the project. Additionally, the functionality to send messages from the app to the device directly was never completed, meaning that while there is a working watering system based on thresholds, these values must be flashed into the memory of the Arduino.

6.1 Design Analysis

Most facets of the project work well; the data collected is accurate and communication between each section is fast and responsive. One aspect that does not work well is the system used to power the device. Due to the device being powered by wall power, only its capability to be used outside is hampered. What could have been done differently to rectify this would be to switch the device to a battery powered setup with an onboard battery held on the plexiglass base supporting the device. Another aspect to change would be to complete the section of code allowing for the device to receive HTTP requests from the app as this would allow the user additional customization of their experience with the device.

7 Professional Responsibility

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Area of responsibility	**Definition**	**NSPE**	**IEEE version**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Asks engineers to continually maintain technical competence meaning that
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Asks engineers to avoid conflicts of interest and unlawful

			professional actions. Each of which could count for a number of actions against ones employer or client
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	IEEE asks its members to be realistic when stating claims
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	IEEE asks its members to make the health of the public and environment paramount. And to make designs as ethical as possible
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	IEEE asks its members to avoid unnecessary damage of all others property
Sustainability	Protect the environment and natural resources locally and globally.		According to code one the protection of the environment should be top priority among protection of public health.
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the	According to code four all manner of unlawful business should be avoided

		profession	

Table 3: Areas of Professional Responsibility

Work competence differences:
Does not explicitly request this however prohibits engineers from doing anything they know will cause harm and or will be negatively affecting other engineers.
Financial Responsibility differences:
Does not ask to be faithful to employers but asks that conflicts of interest should be avoided and unlawful career moves should be avoided as well
Communication honesty differences:
IEEE asks its members to avoid stating unrealistic claims. However, it does not ask its members explicitly to speak objectively outside of the times when subjective claims would harm others/
Property Ownership differences:
While NSPE asks its members to only consider the property of those they work for IEEE asks you to avoid damaging all types of property
Sustainability differences:
NSPE and IEEE largely ask the same thing of its members to ensure that the environment is protected.

Social Responsibility differences: IEEE asks its members to avoid unlawful business practices however does not ask its members directly to act with honor nor with the endeavor to uphold the reputation of the profession you are currently in all though it could be argued that the point of such a code of ethics is implicitly to do just that.

7.2 Project Specific Professional Responsibility Areas

Area of responsibility	**Definition**	**NSPE**	**Relevance**	**Performance**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	This is fairly relevant in terms of the difficulty of different parts of this project; some parts are more difficult than others, thus potentially necessitating people to step	(HIGH) Each member completes their task punctually and delivers high-quality work.

			outside of their normal areas of competency.	
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	This topic is very relevant to our project. The sensors and other components we could buy range in price between tens of dollars and thousands of dollars if no attention was paid to these costs we could end up wasting the entire budget of the project, jeopardizing our ability to finish it.	(HIGH) By using cheap hardware and Glitch servers, our costs have stayed low.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	This is very relevant to the project. Every team member must be honest about their work not to make it appear more valuable to the team than those who are contributing more.	(HIGH) When promises to do things are made, they are usually done, and if not, not without good reason. Everyone communicates their standing effectively.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	This is not particularly relevant. All physical components are low power	(HIGH) All parts are safe, and we take care of electrical components

			devices more likely to just cease working if they got wet or malfunctioned. The main risk is the fertilizer reservoir; however, even this risk can be mitigated by using the proper fertilizer.	around water so that no one's health is put at risk.
Property Ownership	Respect the property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	This topic is relevant in the fact that we have many components on loan from the ETG it is our responsibility to take good care of them	(HIGH) All parts have been handled carefully thus far.
Sustainability	Protect the environment and natural resources locally and globally.		This topic is minorly relevant while currently, the impact on the environment is minor. The environmental impact will be considered if the device ever hits mass production.	(LOW) Currently, it is a prototype so we are not incorporating sustainability in our product. Once scaled, sustainability will become a priority.
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the	This has low relevance to the current project. The effect of the device is currently very minor as it is a	(HIGH) No criminal or otherwise unethical actions have taken place during this

		honor, reputation, and usefulness of the profession.	prototype, and as well the type of unethical actions possible during the course of this project are limited by the small scope of the project.	project.

Table 4: Project Specific Professional Responsibility

7.3 Most Applicable Professional Responsibility Area

The most applicable professional responsibility for this project is primarily financial. Because our project's target audience is hobbyist gardeners, ensuring that our device would be in an affordable price range for the user is essential. By selecting hardware (sensors, actuators, microcontrollers) and a relatively inexpensive server, we can best meet the needs of our users. The key factor to balance this financial responsibility with is the limited accuracy and precision that often comes with cheaper components.

8 Conclusions

8.1 Summary of Progress

Requirements are fully completed with one exception. There is a full setup with a microcontroller and a water/fertilizer disbursement system. The device can interface with pots starting at three inches in diameter. The app features all relevant components, including graphical representations of data stored in the database. However there is no ability to set thresholds via the app. Data is capable of being updated regularly and neither the microcontroller itself nor the app crashes with any frequency.

8.2 Value Provided

The device suits the intended users' needs well and fixes the problem we set out to address. By creating a reasonably priced device, a reliable system to read relevant soil and light data and automate watering and fertilizing, we provide those who do not have the time or energy to perform plant care the option to still have plants.

8.3 Next steps

While this project does successfully fill the need, there could be some aspects improved upon by future teams. For one, by making the device battery powered, the environments the device can operate in would be greatly expanded by allowing the device to be used outside. By implementing websockets, the moment-to-moment updates on soil data could be achieved, allowing for more accurate data displayed on the app. Finally, implementing a way to customize watering or fertilizing thresholds on the app or place built in thresholds within the code.

9 References

[1] Diivo, "Diivoo Smart Soil Moisture Meter for Indoor Plants, Bluetooth Plant Water Monitor and Soil Tester with Mobile Phone app for use in Plant Care, Great for Garden, Lawn, Farm," *amazon.com*. [Online]. Available: https://www.amazon.com/Diivoo-Moisture-Bluetooth-houseplant-bedrooms/dp/B0BQYJTB8W. [Accessed April 16, 2024]

[2] Planta, "Planta: Complete Plant Care" *apps.apple.com*. [Online]. Available: https://apps.apple.com/us/app/planta-complete-plant-care/id1410126781. [Accessed April 16, 2024]

[3] Sinbeda, "Soil Moisture Meter 4 in 1 for HHCC, Plant Water Monitor, Automatically detects Moisture/Temperature/Light/Fertility, Can Connect to Mobile Phone via Bluetooth, Plants Sensor for Indoor (Green - 1pcs)," *amazon.com*. [Online]. Available: https://www.amazon.com/Automatically-Temperature-Fertility-Bluetooth-Hygrometer/dp/B0BG5KP2WV?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&smid=AB0JJOLR5E90L&th=1. [Accessed April 16, 2024]

10 Appendices

APPENDIX 1 – OPERATION MANUAL

App Setup:

1. The Nurture app shall be downloaded from the Apple App store.
2. Once downloaded, open the app and follow the onscreen instructions to create a username and password for your account.

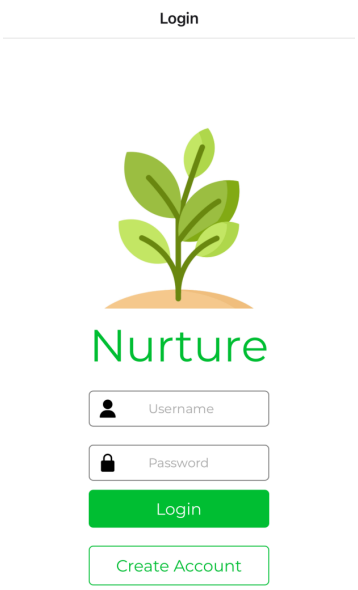

Figure 14: The image above shows the opening page of the Nurture app. Select "Create Account" if this is your first time in the app, or enter your username and password to login to your existing account.

3. Select the leaf icon next to the "My Plants" text on the app to add a plant.

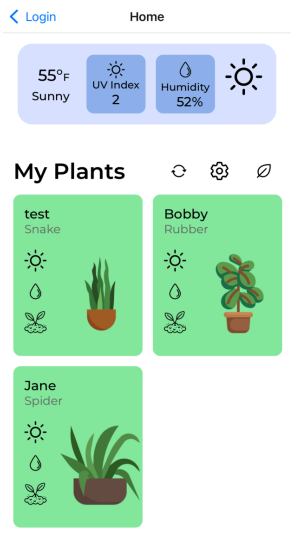

Figure 15: The above image shows the homepage you should be met with after logging in.

4. Follow the onscreen instructions to create a plant.
    a. Create a plant name.
    b. Select the type of plant

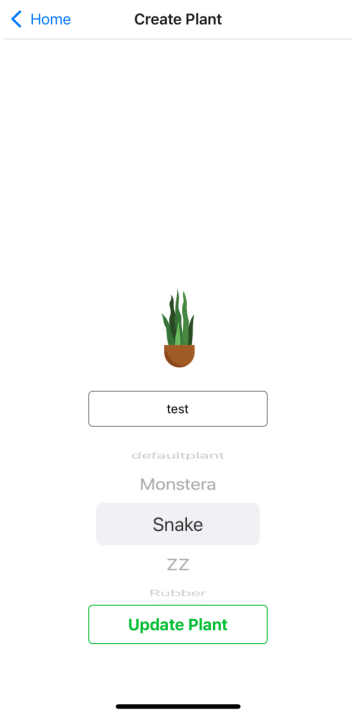

Figure 16

5. Select a plant to view the data being collected, past data, and learn more about what the information means that is being collected.

Hardware Setup:

Initial Setup

(Only necessary when a new wifi network is used)

1. The clear enclosure lid shall be unscrewed and removed.
2. The Arduino shall be plugged into a computer with a Micro USB cable.

Note: The AC power adapter **CANNOT** be plugged into wall power when the Arduino is plugged into a computer. **MAJOR** system damage will occur.

Note: The screws on the corner of the PCB may have to be loosened in order to insert the Micro USB cable, depending on the Micro USB cable used.

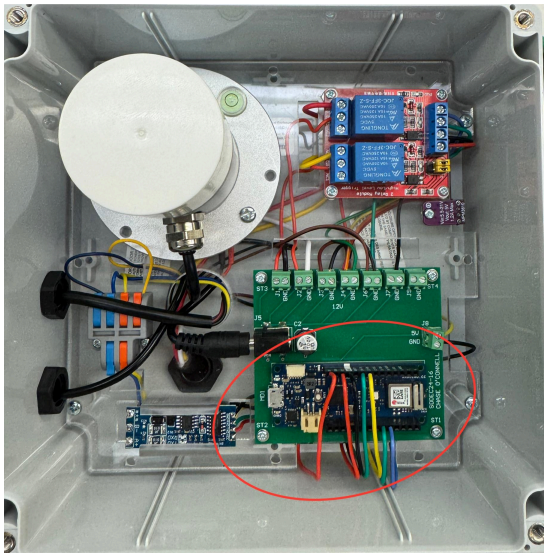

Figure 17: The above image shows the Arduino within the red oval and the micro usb port on the left side of the Arduino.

3. The Arduino IDE shall be downloaded to the computer that the Arduino is plugged into.
    a. The Arduino IDE can be downloaded using the following link: https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE
4. Download the files stored in the folder at the following link: https://drive.google.com/drive/folders/1teVMCk71ZaysSmZv4upRW1huuVRRQbTG?usp=sharing
5. Open all three files in the Arduino IDE.
6. Edit the “secrets.h”
    a. Change the line “#define SECRET_SSID "" “ to include your Wifi network name within the quotes.
    b. Change the line “#define SECRET_PASS "" “ to include your Wifi network password within the quotes.
7. Save the file, compile, and upload the three files to the Arduino.
8. Replace the clear enclosure lid and tighten the screws.

General Setup

1. The hardware enclosure and plant shall be placed in an indoor, or well protected environment where the AC wall adapter will not encounter any moisture.
2. The plant pot shall be placed within three feet of the hardware enclosure in order for the NPK sensor and black hosing to reach the plant pot.
3. The hardware enclosure shall have the clear lid be considered the top of the enclosure and be oriented in such a way that the PAR sensor will not be shadowed by any surrounding objects in order for the sensor to give accurate readings.

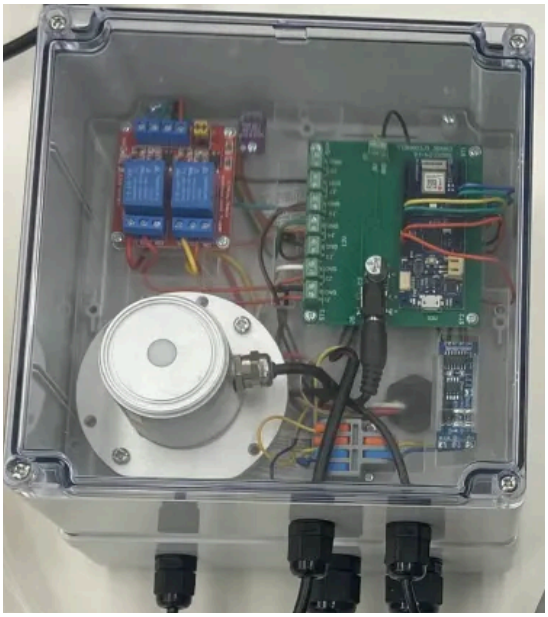

Figure 18: The PAR sensor is shown in the bottom left corner in the image above.

4. The blue drip irrigation stakes shall be pushed into the soil around the root zone of the plant. Each drip irrigation stake shall be connected together with the T-joints and the black 10mm tubing. One T-joint shall be connected to the black 10mm tubing from the hardware enclosure.
5. The NPK sensor (the black device exiting the hardware enclosure with 5 soil stakes) shall be placed near the root zone of the plant with the steel soil stakes pushed completely into the soil. This soil probe is completely water resistant and can also be buried completely in the soil within the root zone of the plant.
6. The two clear tubes shall be placed in two separate reservoirs. The reservoir with the tube exiting the enclosure closer to the corner of the enclosure shall be filled with water. The reservoir with the tube exiting the enclosure closer to the center of the enclosure shall be filled with your choice of plant fertilizer.
7. The AC wall plug shall be plugged into an appropriate wall outlet.

Figure 19: This image shows a viable setup with the Nurture hardware system. The fertilizer/water reservoir is shown on the left, the hardware enclosure in the center, and the pot with the watering system and NPK system on the right.

Appendix 2 - ALTERNATIVE/INITIAL VERSION OF DESIGN

Hardware changes:

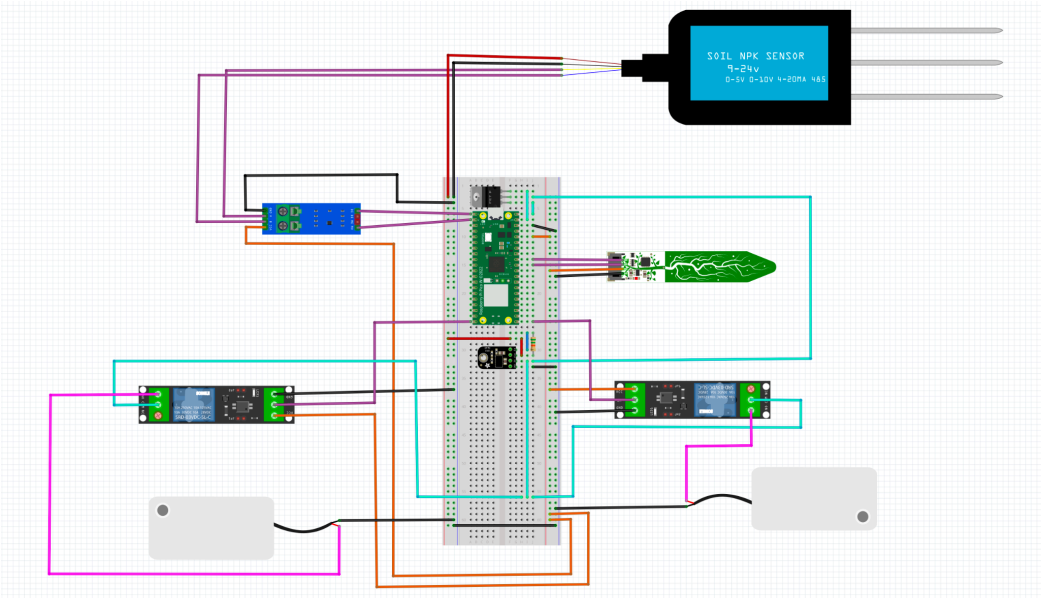

Figure 20: Circuit diagram of initial prototype

There are two main hardware versions of the device: the initial prototype and the final version. The final version largely kept the same design as the initial prototype; however, several key components were replaced with more powerful or versatile versions of those components.

Sensors: One example of this is that the initial prototype had an NPK sensor and a moisture/temperature sensor however, the NPK sensor was ultimately replaced with a newer NPK sensor which could survey more soil values than the original sensor. Among these new soil values were moisture and temperature data making the moisture/temperature sensor redundant. Along with this, a PAR sensor was added to make the readings performed by the device more varied

Pump system: Another component that was swapped out was the initial two pumps replaced with a single stronger pump set up in conjunction with a solenoid valve controlling which substance, water or fertilizer, was being pumped. This choice was made to consolidate space within the device and reduce the price of the device overall.

Microcontroller: Finally the raspberry pi pico was swapped out with an Arduino MKR 1010 WIFI due to communicating with rs485 Modbus being too temperamental.

Software changes:

Initially, our design incorporated Atlas as the primary tool for managing our database and handling requests. However, as the project progressed, we decided to switch to Glitch. This change was driven by its streamlined integration with our tools, its flexibility in accommodating our project's specific needs, and its simplified setup process, which allowed us to accelerate deployment.

APPENDIX 3 – OTHER CONSIDERATIONS

Overview of What we Learned

Throughout this year, we have learned a lot! Anything from defining a product to the ins and outs of the RS485 protocol, along with a whole lot in between, were discussed and learned about this year as we have traversed through this project. We have continued to learn to work effectively as a team, the necessity of good communication, and the neccessity of failure to bring about a better outcome. We have also learned about new protocols, database structures, the ins and outs of building apps, along with relearning old skills, like using CAD software.

Things to Know

The inner workings of the nutrient sensor that we selected measures the nutrient values of the soil (nitrogen, phosphorus, and potassium) in terms of their availability in the soil. This means that the nutrients must dissolved in water in order for their concentrations to be measured. This makes it directly relatable to plants since nutrients enter the plant through the process of diffusion, meaning the nutrients must be dissolved in water for them to be available to the plant. This has added some additional complexity to our fertilizing methods, since we must bring the soil moisture up to a consistent level in order to determine if nutrient levels are actually below our threshold and fertilizer needs to be applied.

APPENDIX 4 – CODE

https://git.ece.iastate.edu/sd/sddec24-16

Appendix 5 - Team

Team Members

- Cameron Jones
- Blake Hardy
- Chase O'Connell
- Cayden Kelley
- Tejal Devshetwar
- Holden Brown

Required Skill Sets for Your Project

Backend development: examples: SpringBoot/MySQL database/MongoDB

Embedded design/development Knowledge of how to connect sensors to the microcontroller and interpret the data sent in by those sensors.

Circuit design: Used for ensuring that the sensors are receiving the correct power and signals.

Skill Sets covered by the Team

Cameron Jones: Embedded design experience, backend development(Spring boot, mySQL)

Blake Hardy: Computer networking, embedded systems.

Chase O'Connell: PCB design experience, low-level programming, hardware design and testing.

Cayden Kelley: Circuit power requirements, hardware design and testing.

Tejal: Frontend app development.

Holden: Backend development.

Project Management Style Adopted by the team

Waterfall

Initial Project Management Roles

Cameron Jones - Hardware

Blake Hardy - Hardware

Chase O'Connell - Electrical

Cayden Kelley - Electrical

Tejal Devshetwar - Frontend

Holden Brown - Backend

Team Contract

Team Members:

1) Cameron Jones
2) Blake Hardy
3) Chase O'Connell
4) Cayden Kelley
5) Holden Brown
6) Tejal Devshetwar

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings: 4:20PM Mondays at SIC hybrid on discord channel.

2. Preferred method of communication: Discord

3. Decision-making policy (e.g., consensus, majority vote): Consensus by relevance / experience and background. Meetings for larger issues. For example, if the decision is about EE specific things, the EE people will need to reach an agreement for the decision.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): Google Doc in a shared folder; different person will do it each week. Links are posted in a Discord channel

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings: So expected attendance, on time. We expect those who are absent to catch themselves up and ask necessary questions. Virtual is acceptable as an alternative to in-person attendance.

2. Expected level of responsibility for fulfilling team assignments, timelines, and

deadlines: Setting deadlines as a team. Expected equal contributions over time. Can be some flexibility from week to week but on average have each person do the same amount of work through the course of the project. Specifically by major, each person is expected to contribute as much as the other team members in their major.

3. Expected level of communication with other team members: Before any major decisions, contact other team members. Provide updates at the weekly meetings. Discord will have channels based on majors each person is expected to make a short message instruction what other people could pick up on that they left off on or where they were stuck. This could also encompass issues that need to be worked on further.

4. Expected level of commitment to team decisions and tasks: High level of commitment to completing assigned tasks and working through major decisions as a team

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.): Equal leadership between all team members. Depending on who completes or is assigned certain tasks, that member can be considered the "leader" of that topic.

2. Strategies for supporting and guiding the work of all team members: The general channel should be used to guide the whole team in addition to the weekly meeting. Discord chat channels for each major should be used to communicate what tasks they are working on and what issues they have. Issues should be solved by both team members if one has hit a roadblock.

3. Strategies for recognizing the contributions of all team members: Members will track their own projects and contributions for the sake of recordkeeping.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

1. Chase O'Connell - Embedded hardware / PCB design in Altium. Firmware and embedded software development. Circuit design, component selection, and circuit testing.
2. Cameron Jones- Experience programming embedded systems. Backend design using spring boot. PLC programming experience and work with PCB design. Solidworks modeling. Willingness to learn.
3. Cayden Kelley - Experience designing, building, and testing circuits. I have an agricultural background and also have experience developing and documenting software requirements along with some C coding experience. Experience building structures and using hand and power tools.
4. Holden Brown - expertise in frontend design and UI planning. Decent at backend programming with spring boot backend and integration with frontend. Experience with Figma for UI design. Good at learning new skills and technologies and integrating them with programming.
5. Blake Hardy - microcontroller embedded systems, spring framework backend, 3d modeling/printing, limited fabrication + power tools,
6. Tejal Devshetwar- Experience with Frontend design using Android studio. Some familiarity with Figma. Experience with Canva as an alternative for UI/UX. Previous experience with Java and C in other classes.
2. Strategies for encouraging and supporting contributions and ideas from all team members:
Creating an inclusive environment for sharing ideas and what people worked on while keeping in mind effort put in rather than progress achieved. Ensuring each individual has a time to provide their updates during team meetings. Being considerate of others' working methods and finding a common ground when it comes to disagreements.
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their

opportunity or ability to contribute?) Be direct, if issues persist, elevate to the rest of the team or advisor to help with resolution.

Goal-Setting, Planning, and Execution

1. Team goals for this semester: Basic non-integrated functionality for individual components. Detailed research, design plans, and schematics.

2. Strategies for planning and assigning individual and teamwork: In each meeting, discuss what you have completed, then add to future goals. Tasks are assigned during weekly meetings.

3. Strategies for keeping on task: Through weekly meetings we will ensure that each person is keeping on task and will address issues as necessary. If you're falling behind or were assigned too much work, you can get help.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Discuss issues at the weekly meetings and create a plan of action for improvement.

2. What will your team do if the infractions continue?

If issues persist, have a team consensus on contacting the project supervisor and course instructors

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) _____Cameron Jones_____ DATE 1/29/24

2) _____Tejal Devshetwar_____ DATE 1/30/24

3) _____Chase O'Connell_____ DATE 1/29/24

4) _____Cayden Kelley_____ DATE 1/29/24

5) _____Blake Hardy_____ DATE 1/29/24

6) _____Holden Brown_____ DATE 1/29/24

Appendix 6 - Miscellaneous

Appendix A: Empathy Map, Personas, and Journey Map

https://www.figma.com/file/W1V47EijGF067I6gA1RoTp/Empathy%2C-Personas%2C-Journey-Maps-16?type=whiteboard&node-id=0-1&t=CI50MGZ7ltaNdWXC-0