Nurture: The Automated Plant Monitor System

DESIGN DOCUMENT

Sddec24-16

Client/Advisor

Ahmed Maruf

Team Members/Roles

Cameron Jones - Hardware

Blake Hardy - Hardware

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: 4/27/2024, V2

Executive Summary

Development Standards & Practices Used

Communications: RS485, I2C, UART, IEEE 802.11

Power: ANSI c84.1

Water resistance: IPx5

Development style: Waterfall

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.
- App notifications and customized care suggestions to maintain optimal plant health.

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 was 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: 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.

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.

Tables

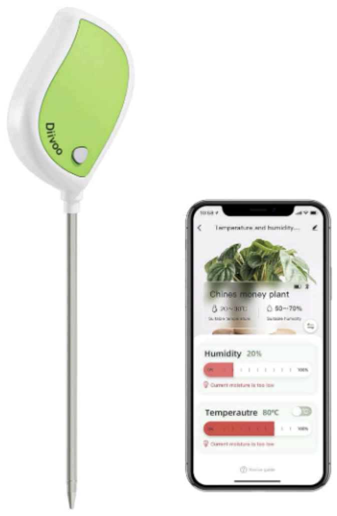


Figure 1: Diivo Smart Soil Moisture Meter Hardware and App

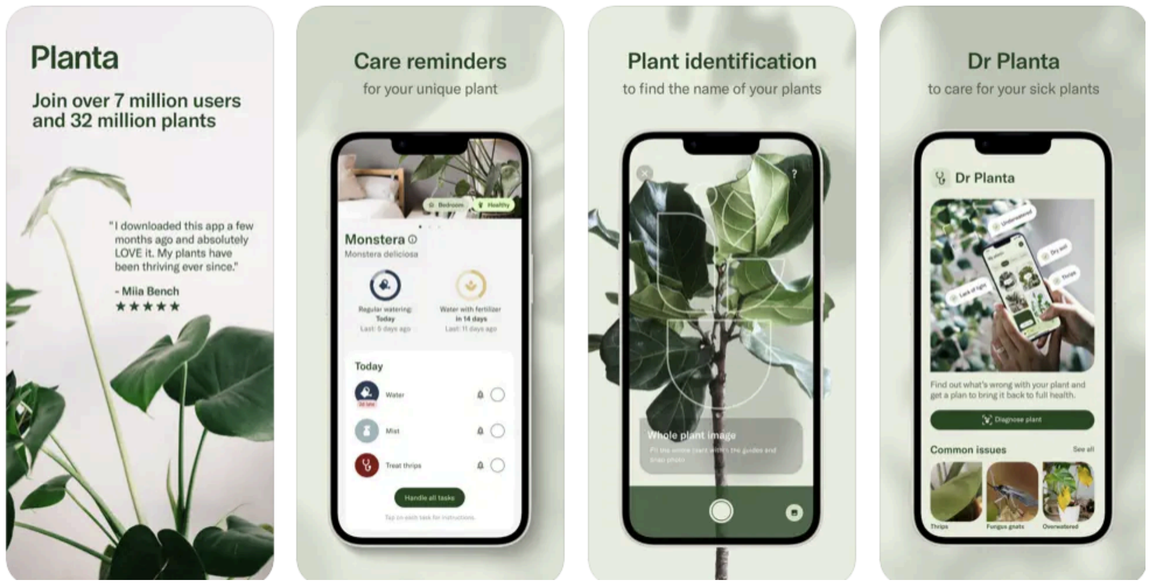

Planta
Join over 7 million users and 32 million plants
"I downloaded this app a few months ago and absolutely LOVE it. My plants have been thriving ever since."
- Miia Bench
★★★★★
Care reminders
for your unique plant
Monstera
Monstera deliciosa
Today
Water
Mist
Treat thrips
Handle all tasks
Plant identification
to find the name of your plants
Whole plant image
Dr Planta
to care for your sick plants
Dr Planta
Find out what's wrong with your plant and get a plan to bring it back to full health.
Diagnose plant
Common issues
See all
Thrips
Fungus gnats
Overwatered

Figure 2: Planta Mobile App Advertisement

Figure 3: Sinbeda Plant Care System

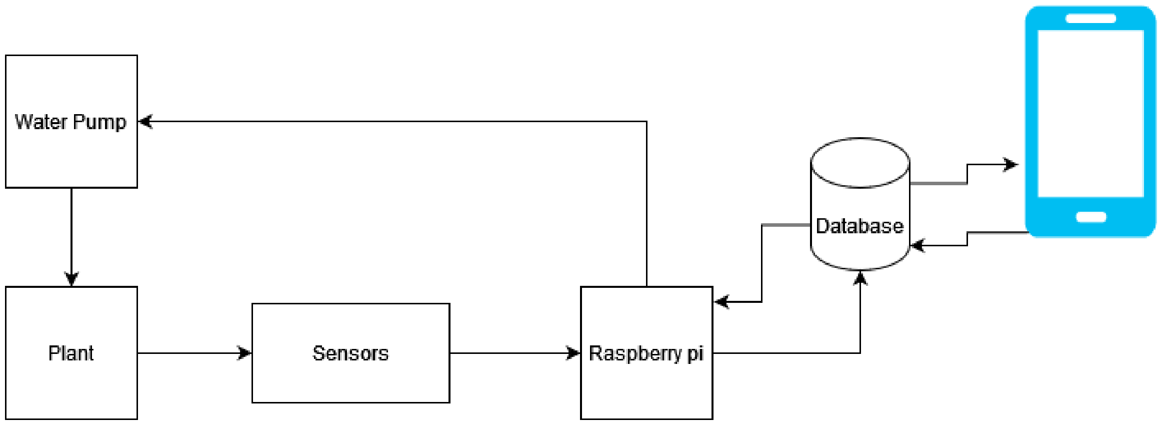


Figure 4: Block diagram of overall system

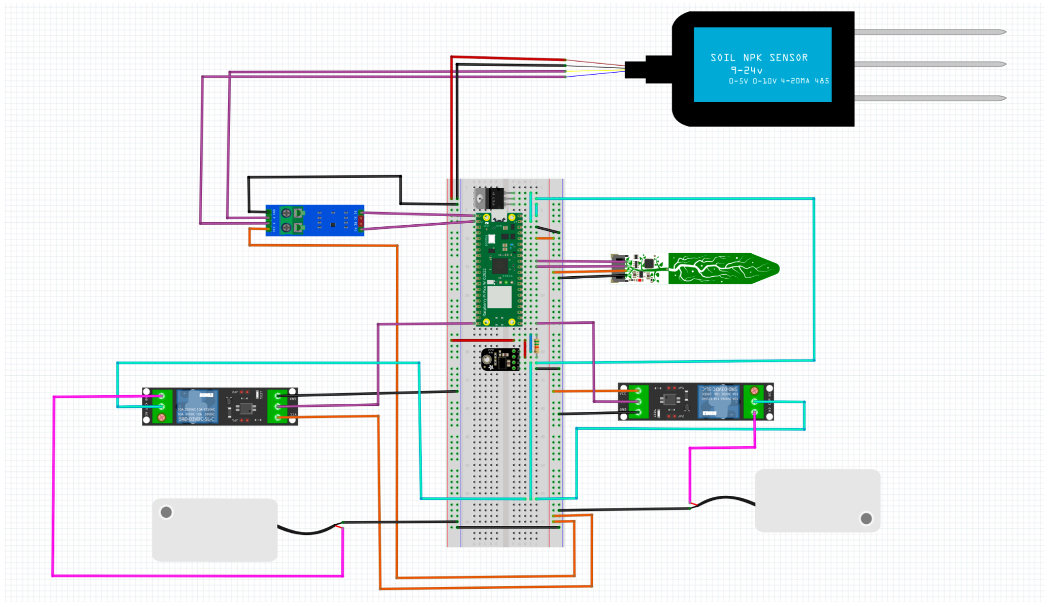


Figure 5: Circuit diagram of the sensor suite, pumps, microcontroller, and other necessary peripheral converters and power delivery devices.

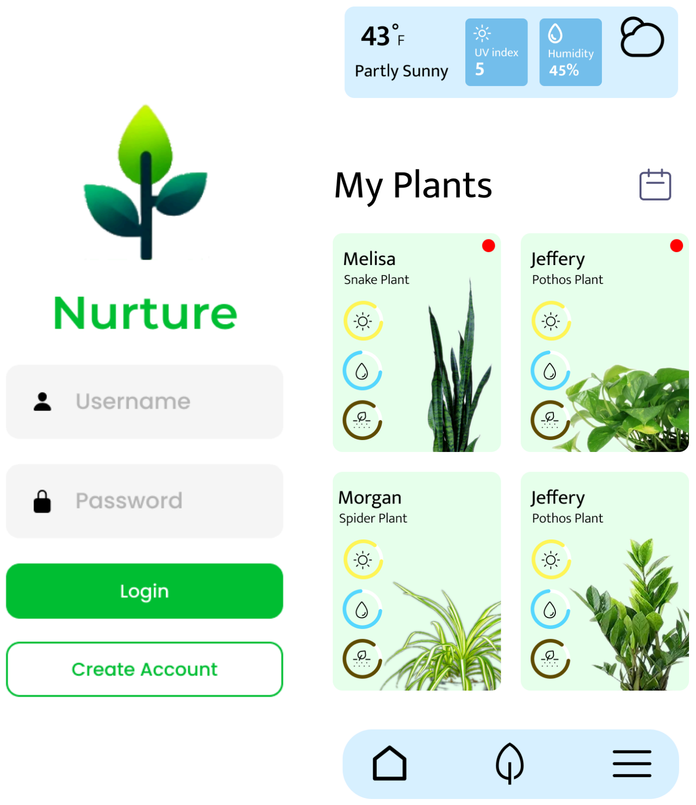
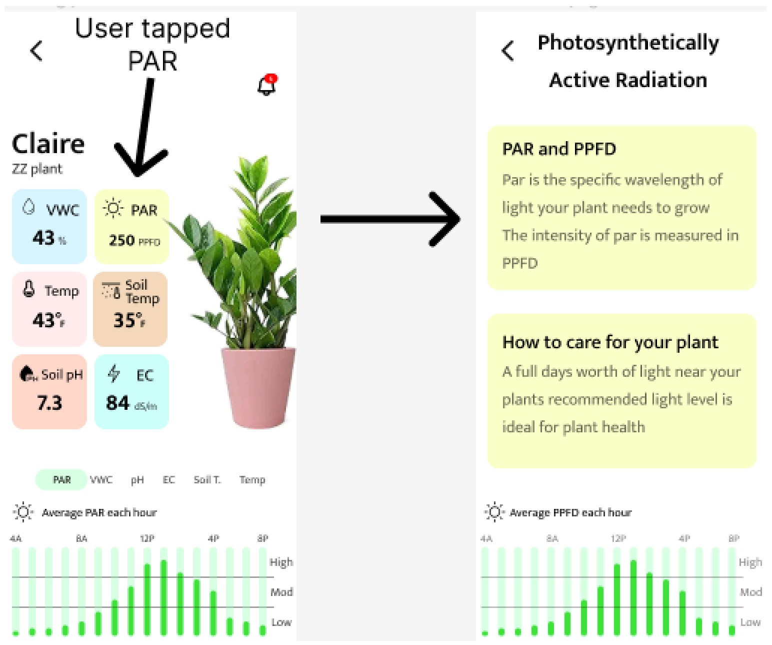


Figure 6: Login card of app (left) and general homescreen of app (right).



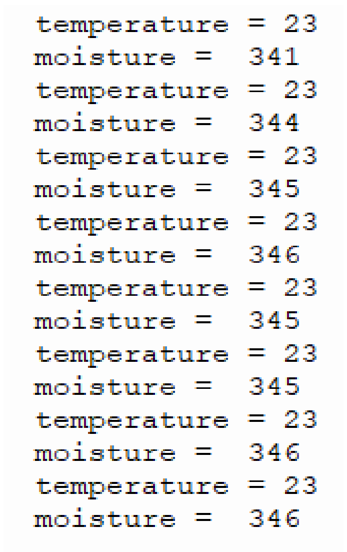
PAR and PPFD

Par is the specific wavelength of light your plant needs to grow The intensity of par is measured in PPFD

How to care for your plant

A full days worth of light near your plants recommended light level is ideal for plant health

Figure 7: Plant information card (left) and nutrient description card (right).



```
temperature = 23
moisture =  341
temperature = 23
moisture =  344
temperature = 23
moisture =  345
temperature = 23
moisture =  346
temperature = 23
moisture =  345
temperature = 23
moisture =  345
temperature = 23
moisture =  346
temperature = 23
moisture =  346
```

Figure 8: Raw Data Sample from the Temp. and Moisture Sensor

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. First, 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 automatically dispense water and fertilizer. 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:

- Sensors + microcontroller package ideally should fit in the palm of the user's hand.
- Complete setup, including water/fertilizer disbursement system and sensor/microcontroller setup.
- Must fit within most soil pot sizes.

Power Requirements:

- Power is delivered via a 12v wall adapter. Other power solutions such as solar or batteries are under consideration.

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 individual sensors.
- Additionally, users should have individual profiles that can be logged into, which are connected to a list of various plants belonging to them.

User Experiential Requirements:

- The device should be able to be turned on and forgotten for long periods of time not having to be recharged or refilled often.
- Sensor readings for all devices should be updated at least once a day.
- The app should be stable and be able to quickly and easily communicate with the server.

2.1.2 Constraints

Size:

The water reservoir must be 1+ gallon(s) to accommodate multiple days of watering.

Power:

Power is delivered via a 12v wall adapter. Other power solutions such as solar or batteries are under consideration.

Cost:

The total material cost of the device should not exceed $300.

2.2 Engineering Standards

- 802.11ac WiFi Standard: Most devices that communicate via wifi today utilize this standard. In our project, this will include the wifi module on the Raspberry Pi, which communicates with the server, the server itself, which will communicate with the user on their phone, and the phone which will contain our app.
- IP55 or better dust and water resistance rating on device enclosure.
- OWASP: ruleset pertaining to security of open web applications to protect against common vulnerabilities

3 Project Plan

3.1 Project Management/Tracking Procedures

The management style we have chosen for our project is a hybrid waterfall/agile project management strategy. The waterfall approach is used in regard to hardware design which is 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 is used in software development which is able to be continuously tested, retooled and redeployed.

Informal project communication will occur via discord due to its versatility, allowing for easy VOIP communication and image sharing allowing for quick communication of ideas. Formal progress software will be stored on the provided team git lab repository, and hardware progress will be 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
    5. Deploy on the app store
- Task 2: Set up MongoDB backend and complete a round trip through Pi MongoDB and app.
    1. Create a paper model of the database structure
    2. Get familiar with MongoDB and how to use their hosting services
    3. Develop database schema
    4. Connect the app to the MongoDB Atlas server with Express
    5. Deploy Express on a server so the database can be accessed without a local host
    6. Get the Pi to store data on the MongoDB Atlas using the Express server
    7. Complete round trip and test functionality
- 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 Pi to get sensor data and complete the round trip.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Task 1 Milestones:

- App mockup is finished
- UI works as intended

Task 2 Milestones:

- Database structure complete on paper
- Database successfully deployed on Atlas server
- The app is able to update the MongoDB database hosted on Atlas

Task 3 milestones:

- 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 remotely by the user via App
- Final working prototype circuit is created

3.4 Project Timeline/Schedule

Target 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.
Week 3: Deploy on the app store.

Target 2: Set up MongoDB, Test it, and Complete a Round Trip through Pi MongoDB 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 backend with hosting services and make break points.
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.

Target 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 sensor data can be read and formatted in a form that can be useful. Initial software to command actuators.
Week 4: Connect the backend to Pi to get sensor data and complete round trip.

May:
Week 1: Ensure sensor data can be read and formatted in a form that can be useful. Initial software to command actuators.
Week 1-2: Connect the backend to Pi to get sensor data and complete round trip.

August:
Week 2-4: PCB Design.
Week 2-4: Ensure sensor data can be read and formatted in a form that can be useful. Final software to command actuators.

September:
Week 1: Ensure sensor data can be read and formatted in a form that can be useful. Final software to command actuators.
Week 1-3: PCB Design.
Week 4: Hardware Testing.

October:
Week 1-4: Hardware Testing.

November:
Week 1: Hardware Testing.

Target 4: Additional database and app 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.
Week 2-4: Update database columns/backend code.

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

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 advisor.
- Perform regular data backups.

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 1: UI Design and React Native Implementation	Deploy on app store	5
Task 2: MongoDB Express Backend and Pi Integration	Model database structure on paper	1
Task 2: MongoDB Express Backend and Pi	Learn MongoDB and Express	2

Integration		
Task 2: MongoDB Express Backend and Pi Integration	Develop schema for user data	3
Task 2: MongoDB Express Backend and Pi Integration	Host MongoDB and Express on the cloud	1
Task 2: MongoDB Express Backend and Pi Integration	Connect the app to the backend and Pi	4
Task 2: MongoDB Express Backend and Pi Integration	Complete round trip and test 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 backend	3
Task 3: Hardware Implementation and PCB Design	Connect hardware with backend	25
Task 4: Database and App Functionality Extension	Research plant type needs	5
Task 4: Database and App Functionality	Install and link new sensors	10

Extension		
Task 4: Database and App Functionality Extension	Update database and backend code	10
Task 4: Database and App Functionality Extension	Test extended functionality	2

Table 1: Personal Effort Requirements

3.7 Other Resource Requirements

Prototyping Components:

- **Raspberry Pi Pico:** 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.
- **Light Sensor:** To monitor ambient light levels affecting plant growth.
- **Humidity and Temperature Sensor:** For tracking the air conditions around the plant environment.
- **Actuators:** Solenoid valves or similar mechanisms for water and liquid fertilizer dispensing.

Hardware Assembly and Enclosure:

- **Enclosure:** A case to house the electronics with modifications for sensor and actuator mounting.
- **Relays and Wiring:** For interfacing actuators with the Raspberry Pi Pico.

Connectivity and Control:

- **Power Supplies:** Adequate for powering the Raspberry Pi Pico, sensors, and actuators.
- **PCB:** Custom board for neatly organizing and connecting electronic components.

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 a multimeter and potentially an oscilloscope for circuit testing.
- **Consumables:** Solder, wire, and other materials for assembly and maintenance.

Software and Development:

- **Development Environment Subscriptions:** For programming the Raspberry Pi Pico 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**	**Examples**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices
Environmenta l	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials

Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

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 mobile app via Bluetooth.
  - Small form factor, fits in the palm of one's hand.
  - Simple setup: Insert device into soil and press a button.
  - No soil nutrient monitoring.
  - No automatic watering or fertilizing.
  - Cost: ~$15
  - System 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 required other than a phone.
  - No automated care.
  - No way to measure soil nutrients.
  - Cost: Free/In-app purchases
  - App store information shown in Figure 2.
- Sinbeda [3]
  - Measures soil moisture, temperature, light intensity, and soil nutrients.
  - Database of 6000+ plants.
  - Battery button cell for power.
  - App provides users with tailored plant care advice.
  - Bluetooth connection rather than WiFi.
  - No automatic watering or fertilizing.

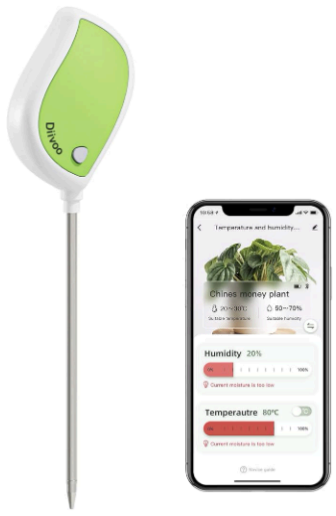


Figure 1: Diivo Smart Soil Moisture Meter Hardware and App

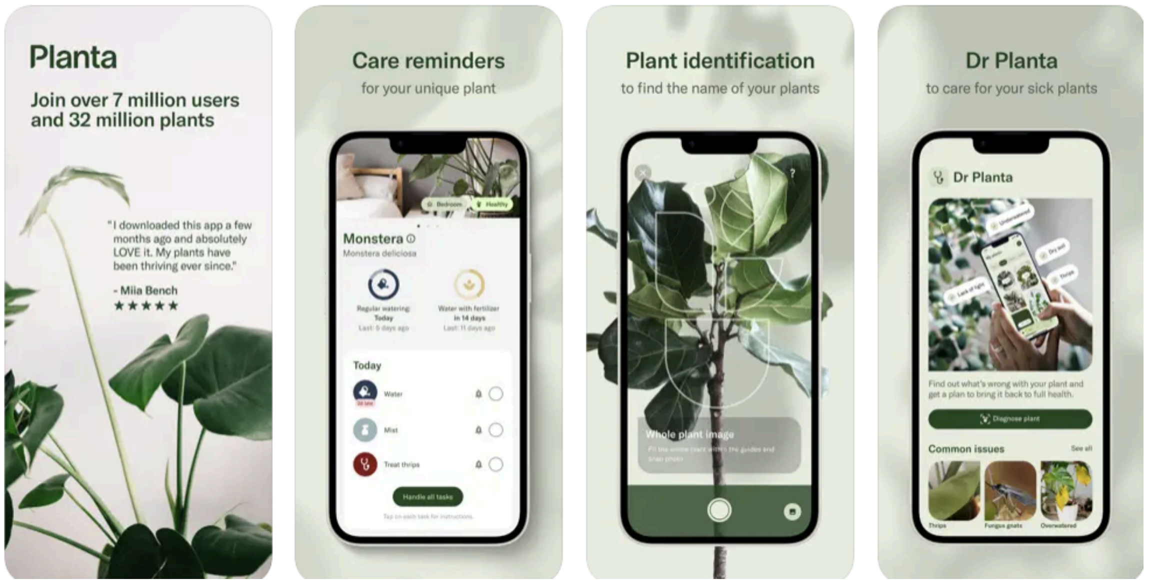

Planta
Join over 7 million users and 32 million plants
"I downloaded this app a few months ago and absolutely LOVE it. My plants have been thriving ever since."
- Miia Bench
★★★★★
Care reminders
for your unique plant
Monstera
Monstera deliciosa
Today
Water
Mist
Treat thrips
Handle all tasks
Plant identification
to find the name of your plants
Whole plant image
Dr Planta
to care for your sick plants
Find out what's wrong with your plant and get a plan to bring it back to full health.
Diagnose plant
Common issues
See all
Thrips
Fungus gnats
Overwatered

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 is able to interface and format data from a variety of sensors and communication protocols, including UART, I2C, and RS485.
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 frontend requires pages for login, overview, individual plant cards, data visualization through graphs, and measurement explanations.
2. The system's backend needs to communicate with the server and access user data.
3. Low-level embedded programming is necessary for receiving and formatting data from sensors and activation of 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.
2) Server type: The server is the main information hub where the data sent by the device is stored and where the app draws its data. Choosing a reliable server would ensure smooth communications without dataflow disruptions, a key aspect of success within our project. The server must also be cost-effective while meeting the computational requirements of the plant health algorithms.
3) App platform: The decision of whether to make the app for Android, Apple, or both presents a trade-off. If our team only focuses on one platform, there will 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 is important.

4.2.2 Ideation

Through the lotus blossom technique, we expanded our essentials 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 our selection on a base case might cause us to narrow our selection too much for an application that is supposed to be general, so we determined that sensors for general care would be best. In terms of 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 cheaper sensors and a limited number of sensors would be best.

4.3 Proposed 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 then stores it within the database. A user can then view sensor data graphs in the app. The user can request watering or liquid fertilizer on the app based on sensor data. Dispensing to your plant is communicated to the microcontroller through a websocket for real-time results. After the user sends the request, the microcontroller interprets it and operates the actuators to dispense the needed liquid from the reservoirs. Automatic fertilizer and water dispersal are also available based on user-defined nutrient and moisture levels.

4.3.2 Detailed Design and Visual(s)

High-Level:

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

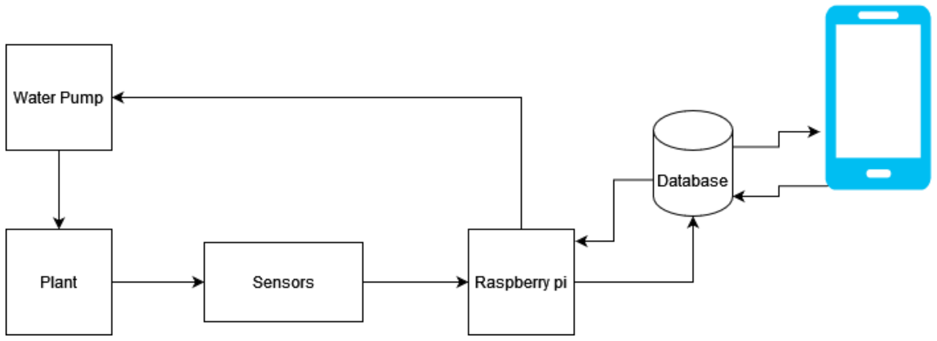


Figure 4: Block diagram of overall system

Hardware:

Our device incorporates the use of a Raspberry Pi Pico as the main microcontroller and method for low-level data handling. Peripherals connected to the Pico include an NPK sensor, soil moisture/temperature sensor, RS485 to UART converter, relays for liquid pumps, and a buck converter to supply the necessary power to the microcontroller.

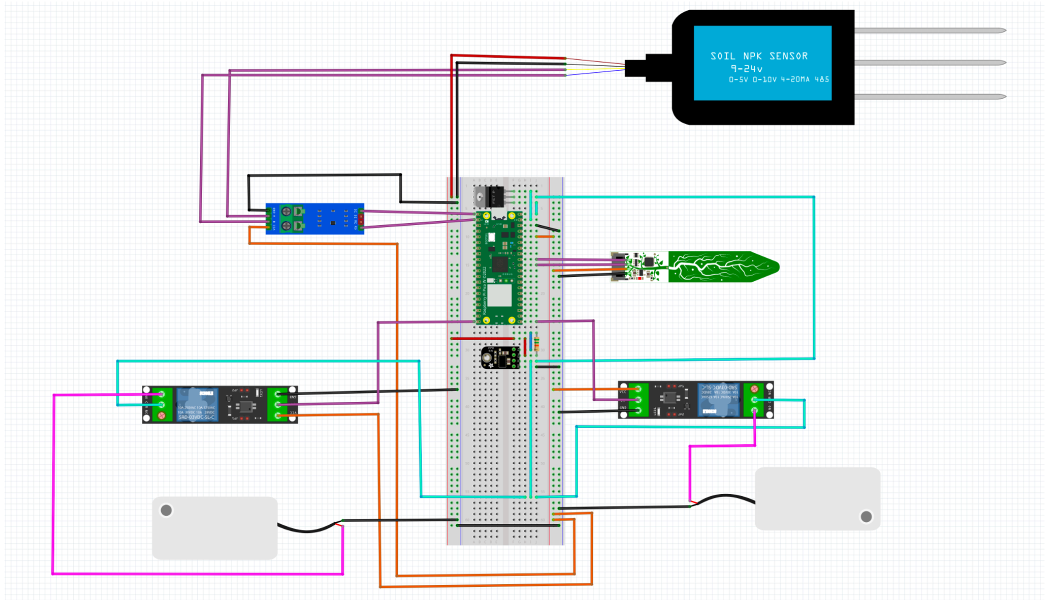


Figure 5: Circuit diagram of the sensor suite, pumps, microcontroller, and other necessary peripheral converters and power delivery devices.

Software:
The user app contains their local profile data. Tapping on a profile provides additional information on sensor data and plant health. Humidity, UV index, and ambient temperature are obtained through API and updated based on location. Information on the key factors monitored by the device and how they affect plant health are also available to educate users.

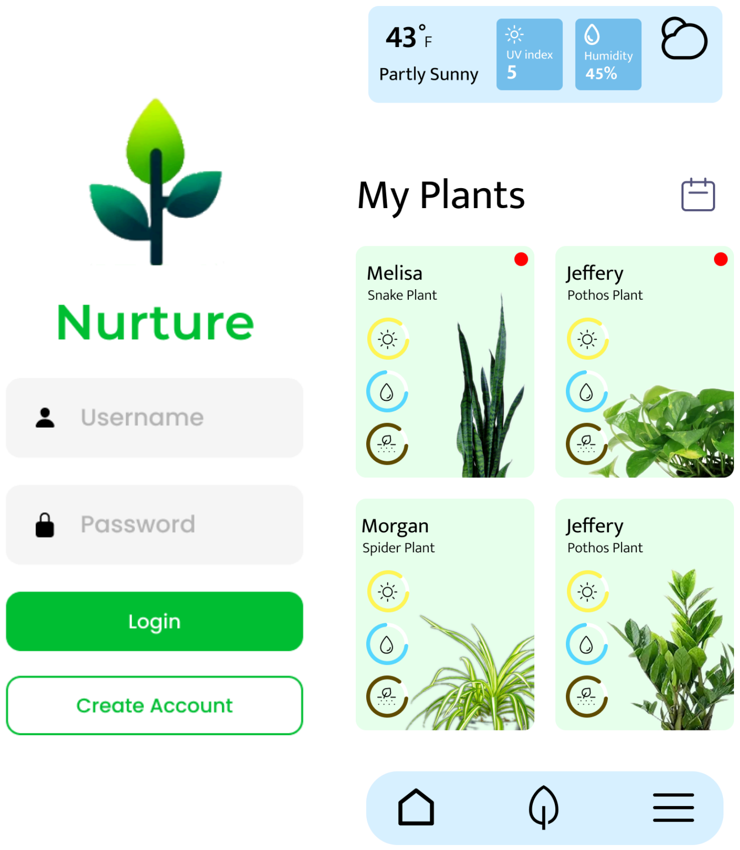
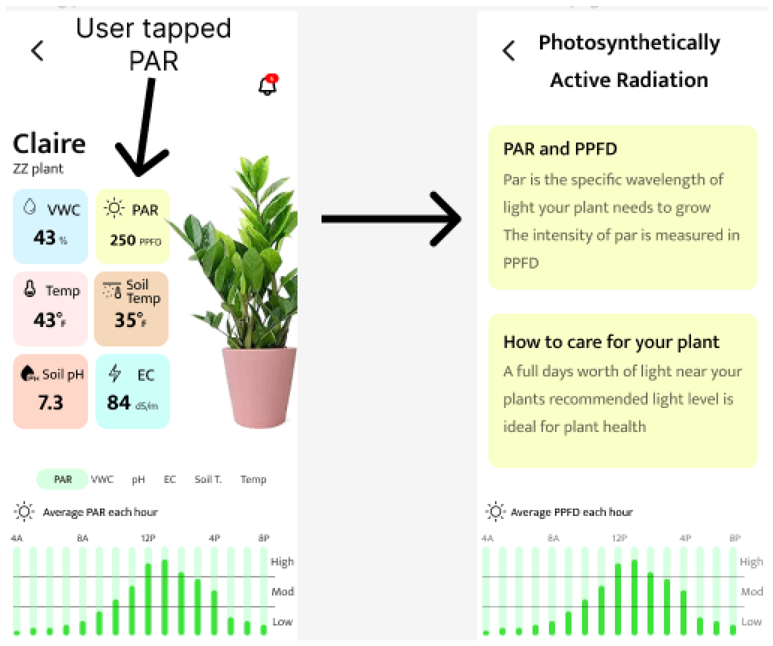


Figure 6: Login card of app (left) and general homescreen of app (right).



PAR and PPFD

Par is the specific wavelength of light your plant needs to grow The intensity of par is measured in PPFD

How to care for your plant

A full days worth of light near your plants recommended light level is ideal for plant health

Figure 7: Plant information card (left) and nutrient description card (right).

4.3.3 Functionality

The user's role would be relatively simple: after purchasing the device and downloading the app, they would have to insert a 12V battery to power it. The user would then place the device in the soil they intend to monitor. Next, they would open the app, create 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 until the device's battery runs low and needs replacement. The goal of our system is to allow for a hands-off approach from the user.

4.3.4 Areas of Concern and Development

Our design fits user needs well. We can drive the per-unit price down using relatively cheap 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.

Due to our selection of relatively cheap sensors, we may encounter durability issues especially in a wet and somewhat exposed environment. If this product were to be scaled up in development, the underlying hardware of a Raspberry Pi may not be the best microcontroller for the application due to its cost.

Testing the durability of the sensors and actuators will come naturally with the testing of the product. The issue of finding a microcontroller that will work on a large economic scale will be somewhat more difficult and will likely take the design of a custom PCB. After developing a proof-of-concept version of the device, we will continue to explore this possibility.

4.4 Technology Considerations

Our group has selected a moisture/temperature sensor that communicates via I2C as well as an NPK sensor that communicates with Rs485. The I2C sensor is compatible with our microcontroller and thus causes no issues. However, the microcontroller cannot receive Rs485, and thus a bridge is required between the microcontroller and the sensor. 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 Atlas server. This technology is scalable, efficient, and easy to use, so there are few drawbacks.

4.5 Design Analysis

Currently there is a working backend with an app reading sensor data stored in the database. The database is running correctly on an Atlas server. The only step left is to connect these three components. Ultimately, we need a complete prototype to say whether or not our proposed solution in 4.3 works or not, simply that individual sub-components of the device work. That said, it is pretty likely that our current design will work as we will be using libraries known to work in this type of setting, such as the Python request library and Volley.

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 another computer over PuTTY to see if they are accurate.
- 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

There are four significant interfaces within this system. The communication between the Raspberry Pi Pico and the server, the server and the mobile app, and the mobile app and the Pico.

Pico to Server:

To test proper functionality between these two components, the Pico will send “put” commands to the server via the Request library. The server's contents will then be analyzed through a “get request” via Postman.

Server to App & App to Server:

The connection between the mobile app and the server will be 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 plants 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.

App to Pico:

To test if the mobile app and the pico can communicate the user will send a request on the app to send water to the soil for a selected amount of time, if the motor begins working or an led activates then we will know that the request has been received.

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 purpose of the device. This will be tested through the use of the Request library on the Pico sending Restful commands to the server. The server will then be queried through Postman to see if the data has been properly received and interpreted in the database.

5.4 System Testing

To test the entire system, a “full loop” must be completed starting with the sensors sending data to the Pico which then interprets the sensor data and sends it to the server. The server should then collect that data and store it in the database. A user should then be able to either create a new account or login and view that data on their screen. The user should then send a command to the Pico to begin watering for a set amount of time, resulting in the soil being watered for that amount of time. 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 Pico can transmit data, but it may act as a bottleneck toward data being transmitted if it does not work.

5.6 Acceptance Testing

To verify that all functional requirements are met, a “full loop” test must be completed. The sensors must pick up data from soil, then the data picked up by the sensors must be interpreted by the pico and sent up to the server, which then will receive and store the data. A user on the mobile app must then make an account, also stored on the server, and request the soil data, which will then be displayed on the app. The user must then send a command to the pico via a web socket to water the soil. The pico must then activate its

GPIO and activate the motor actuators for the amount of time specified by the user. This will satisfy the functional requirements.

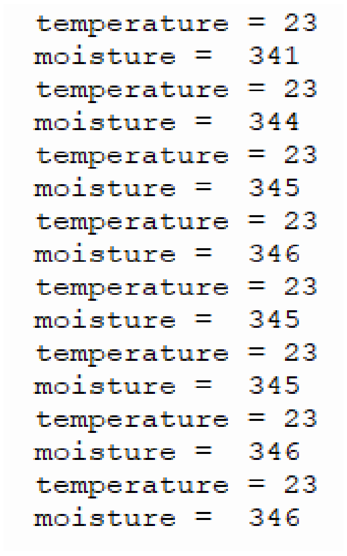
Most nonfunctional requirements will be solved in the part selection process such as the device being able to fit in the palm of the user's hand and the price of the device not exceeding $300. However, for the requirement that the device must be able to remain powered for up to three days the long-term test will be performed where the device will be performed where the device is kept powered for three days and periodic commands are made to it to ensure that the device remains powered and is continuing to transmit information.

5.7 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 moisture and temperature sensors function correctly, showing data from the moisture and temperature sensor. A sample of this raw data can be seen in Figure 8. Note that the NPK sensor data is not currently interpretable. 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.



```
temperature = 23
moisture =  341
temperature = 23
moisture =  344
temperature = 23
moisture =  345
temperature = 23
moisture =  346
temperature = 23
moisture =  345
temperature = 23
moisture =  345
temperature = 23
moisture =  346
temperature = 23
moisture =  346
```

Figure 8: Raw Data Sample from the Temp. and Moisture Sensor

Interface Testing:

The microcontroller successfully retrieves moisture and temperature data but does not connect to the database.

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

A websocket running on the Glitch server facilitates communication between the Python code (not hosted on the microcontroller) and the mobile app.

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.

System Testing:

The moisture and temperature sensors provide reliable data, which is processed by the microcontroller.

While the microcontroller does not directly send data to the server, the system's design ensures that relevant data is accessible through the app via the server's database and the Python code on Glitch.

Users can send commands through the app using websockets, which are properly received by the Python code.

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:

Key functionalities like data retrieval from the moisture and temperature sensors and user interaction through the app are operational.

Nonfunctional requirements such as device size, and battery life were not met although the cost was met.

6 Implementation

Our initial prototype is missing some key features so our primary work will be to fix these before moving on to the creation of a more finalized prototype. We currently cannot receive data from the NPK sensor. There is no connectivity between the microcontroller server and mobile app and there has been no testing on the water/fertilizer dispersal system. Fortunately, none of these features are required to be finished before we can move on to another allowing all of these to be worked on in parallel.

- Work on the NPK will be accomplished by further research into how to receive UART information in a Raspberry Pi and further investigation into how the RS485 to UART converter operates. Before writing code to capture the converted UART signal from the NPK sensor.
- Python code has been created to communicate between the microcontroller and the server. Python successfully accesses the users within the database and creates a plant although this needs to be tested on the microcontroller. Further research needs to be done into how Python works with our database.
- Synchronous communication between the mobile app and the microcontroller uses a websocket. Currently, the websocket is running on a server and works properly to communicate with Python code and our frontend although the Python code needs to be deployed on the microcontroller and tested there. In addition, further research into the functionality of websockets needs to be done to improve functionality for multiple concurrent users.
- Testing the water/fertilizer dispersal system will take the purchase of a reservoir to hold water and will take some research into time functions to measure how long to keep the relay active that keeps the actuator running.

Following these objectives, a functional prototype will be completed leaving the development of something closer to production to be the last step. This will require an analysis of whether our current components are conducive to scaling; at this stage, some components may need to be replaced, setting us back to the functional prototype stage. When all components have been finalized, we will begin the PCB design containing all the electrical components, helping to reduce the space the device occupies and making it less unwieldy to hold. Finally, a container to hold the PCB and reservoirs will be designed. This will require research into waterproofing to prevent issues with the reservoirs.

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 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 profession	According to code four all manner of unlawful business should be avoided

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 outside of their normal areas of competency.	(HIGH) Each member completes their task punctually and delivers high-quality work.
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 Atlas servers, our costs have stayed low.
Communicatio n Honesty	Report work truthfully, without	Issue public statements only in an objective	This is very relevant to the project. Every	(HIGH) When promises to do things are

	deception, and understandable to stakeholders.	and truthful manner; Avoid deceptive acts.	team member must be honest about their work not to make it appear more valuable to the team than those who are contributing more.	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 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.	(HIGH) All parts are safe, and we take care of electrical components 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		This topic is	(LOW)

	environment and natural resources locally and globally.		minorly relevant while currently, the impact on the environment is minor. The environmental impact will be considered if the device ever hits mass production.	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 honor, reputation, and usefulness of the profession.	This has low relevance to the current project. The effect of the device is currently very minor as it is a 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.	(HIGH) No criminal or otherwise unethical actions have taken place during this 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 Closing Material

8.1 Discussion

Requirements are currently not fully complete; the device is in a prototype stage where it would be impossible to hold in the palm of one's hand. No power source component has been selected, and thus no work on the battery life requirement has been completed. However, all UI requirements have been completed, the frontend can view data stored on the server graphically, and user profiles can be logged in. Additionally, the device can read sensor data and has code to communicate sensor data to the server; said code hasn't been tested on the microcontroller, although it has been tested and observed to work. Thus, its not possible to say full connectivity of the system has been achieved, although we are very close. Finally, the overall price has remained under $300, so the cost requirement has still been fulfilled.

8.2 Conclusion

We nearly have all the needed parts to complete the project barring water and fertilizer reservoirs. We have a completely functional mobile app and backend. We lack code for reading data from the NPK sensor testing on the water/fertilizer disbursement system and connectivity between the microcontroller and the server. We aim to rectify these missing components by completing our initial prototype. Later, the prototype will implement the PCB and weatherproof casing, which is a nearly complete prototype. To do this, we will have regular meetings to move the project forward and distribute weekly tasks. What kept us from achieving these goals was largely down to time constraints and other responsibilities barring us from working on the project. However, these constraints can be dealt with through a more thoughtful action plan.

8.3 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]

8.4 Appendices

Appendix A: Empathy Map, Personas, and Journey Map

https://www.figma.com/file/W1V47EijGF067I6gA1R0Tp/Empathy%2C-Personas%2C-Journey-Maps-16?type=whiteboard&node-id=0-1&t=CI50MGZ7ltaNdWXC-0

9 Team

9.1 Team Members

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

9.2 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 how to interpret the data sent in by those sensors.

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

9.3 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.

9.4 Project Management Style Adopted by the team

Waterfall

9.5 Initial Project Management Roles

Cameron Jones - Hardware

Blake Hardy - Hardware

Chase O'Connell - Electrical

Cayden Kelley - Electrical

Tejal Devshetwar - Frontend

Holden Brown - Backend

9.6 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