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.
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 use of the lotus blossom technique we were able to expand our thoughts on what should be considered when selecting sensors. Five options we considered for this design decision were as follows:

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 itself, a server, and an app. The device reads in temperature, moisture, and nutrient data from the soil it is plugged into, then sends this data to a server to be stored. A user can then request this data through the use of the app where information will be shown graphically and numerically. The user can then request for the soil to be watered or receive nutrients via liquid fertilizer. This request is then passed through the server to the device where the device activates motors to draw either water or fertilizer from two reservoirs. Automatic fertilizer dispersal and watering is also available through the server based on nutrient and moisture levels sent in by the device.

4.3.2 Detailed Design and Visuals

High-Level:

Figure 4.1 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 aspect of the project. Data and control will flow between the user’s device and Raspberry Pi through this server.

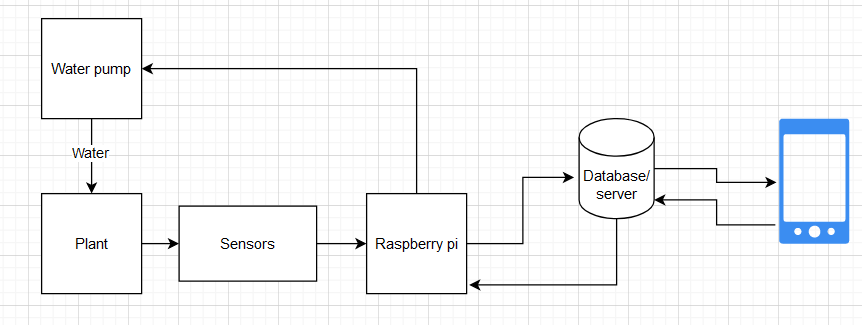


Figure 4.1: High-Level Block Diagram

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 I2C converter, relays for liquid pumps, and a buck converter to supply the necessary power to the microcontroller.

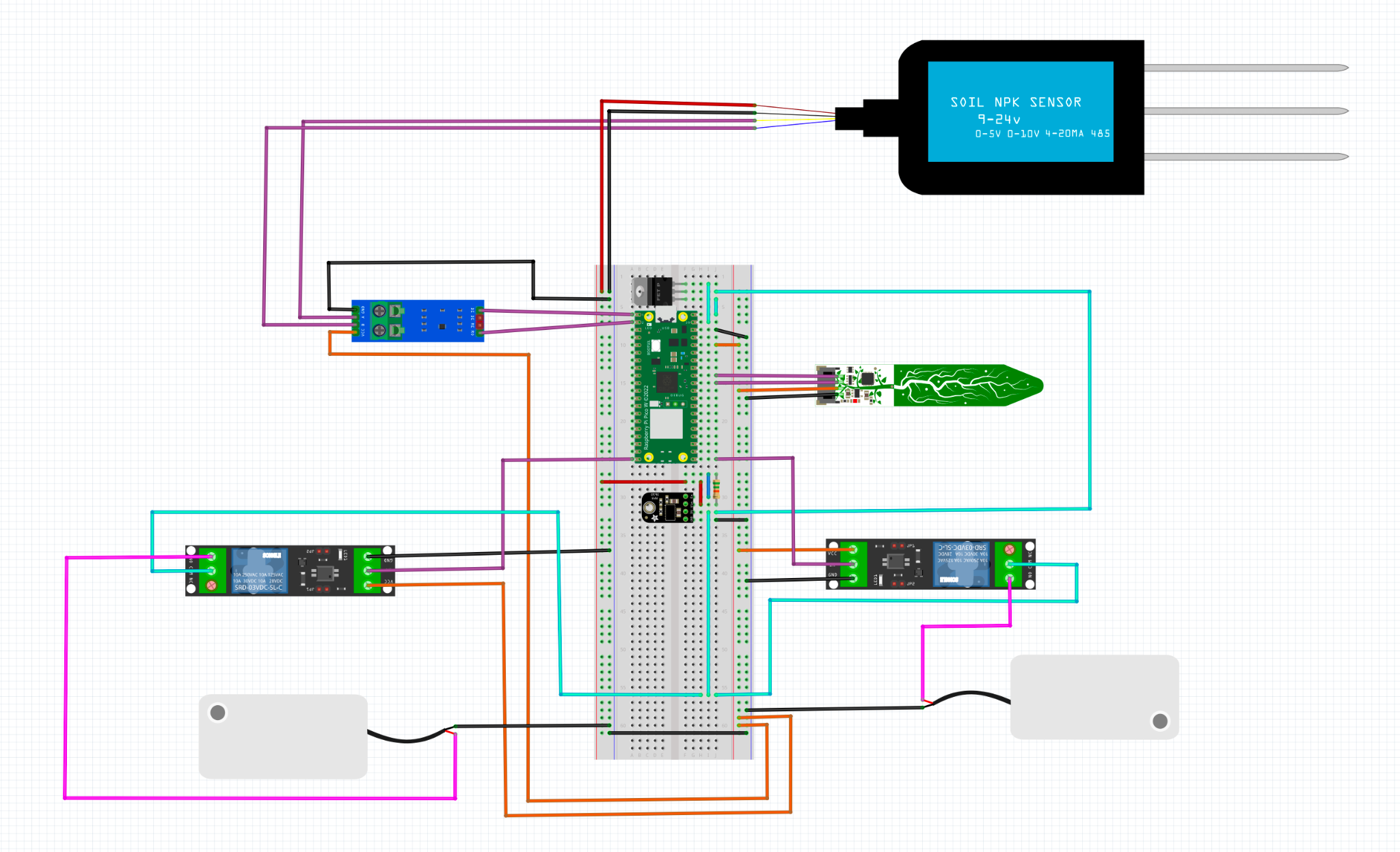


Figure 4.2: Overall Hardware Diagram

Software:

The user app contains profiles for each of the user’s plants. Tapping on a profile provides additional information on sensor data and plant health. Humidity, UV index, and ambient temperature are obtained through API and update based on location. Information on the key factors monitored by the device and how they affect plant health is also available to educate users.

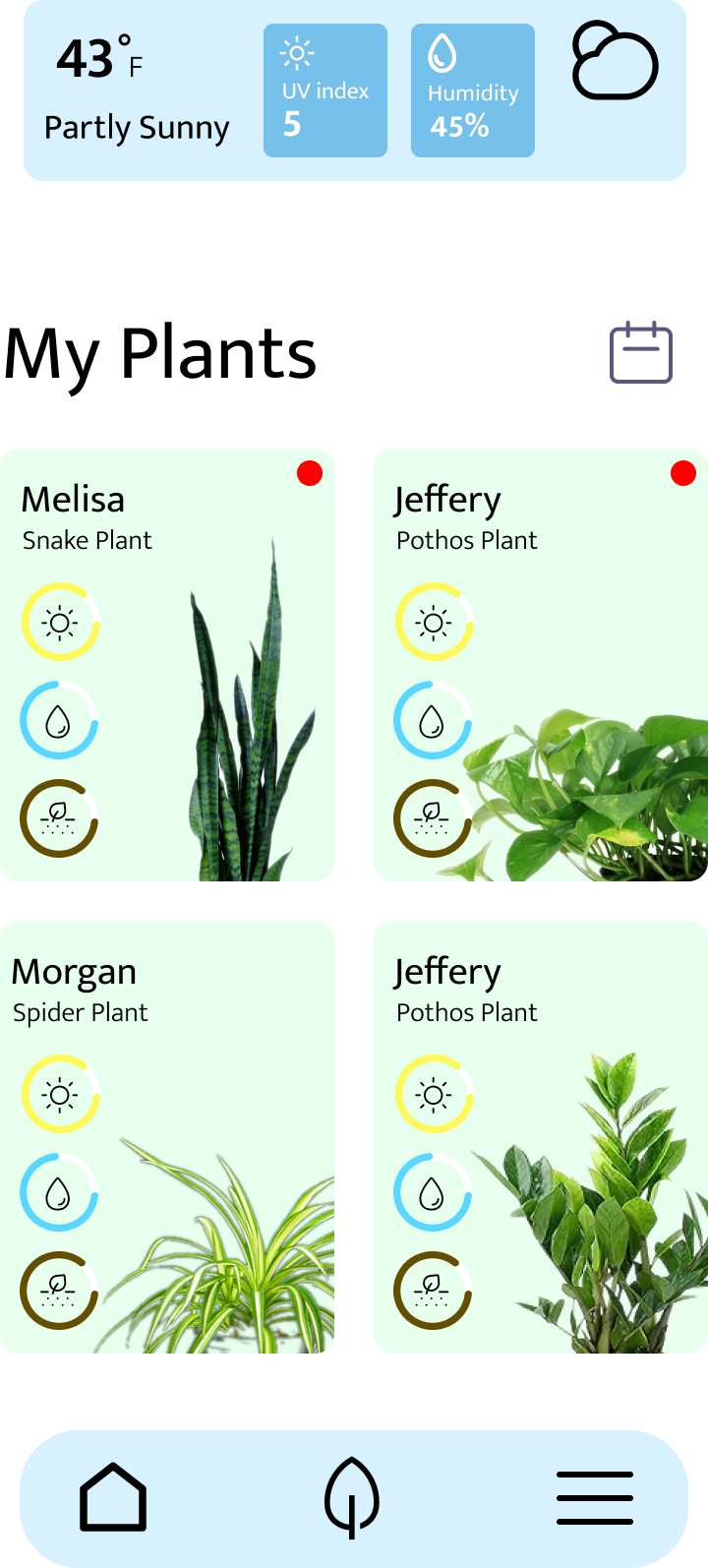
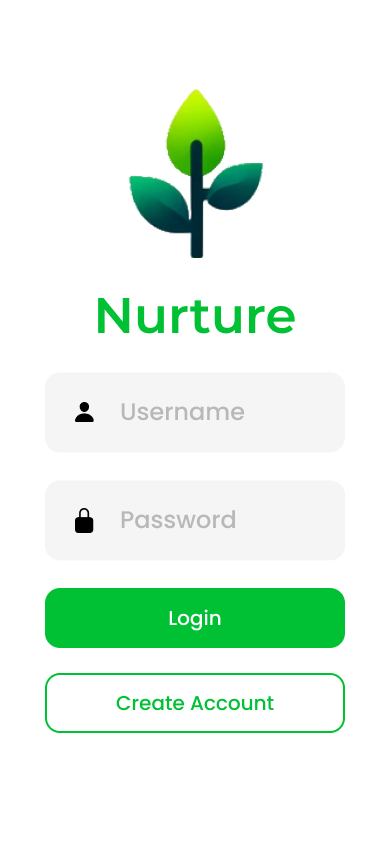


Figure 4.3: Nurture App Login and Home Pages

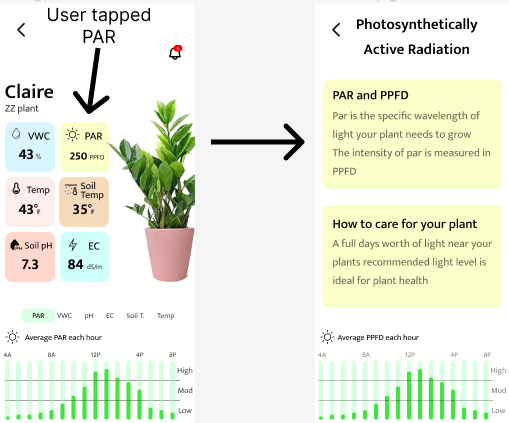


Figure 4.4: Nurture App Plant Info Card

4.3.3 Functionality

The user’s role would be relatively simple: after purchasing the device and downloading the app, they would then have to insert a 12V battery in order to power the device. Following this, the user would place the device in the soil they intend to monitor. Next they would open the app and 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 until the battery of the device runs low. 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. Through the use of relatively cheap sensors and actuators, we are able to drive the per unit price down. Although there is a trade off with sensor precision due to the cheapness of the sensors, our primary audience of hobbyist gardeners will likely prefer the cheapness of the product over absolute precision. Additionally due to the product's 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 their 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 which communicates via I2C as well as an NPK sensor which 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 is also using MySql database to house data in our server. This technology is scalable, efficient and easy to use so there are few drawbacks.

4.5 Design Analysis

Currently we have a working backend and app developed and reading correct values from the sensors onto the microcontroller the only step left is to connect these three components. Ultimately though without having a complete prototype at this time, we cannot say whether or not our proposed solution in 4.3 works or not simply just that individual sub components of the device work. That said, it is quite 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.