4.3 Proposed Design

4.3.1 Overview

Provide a high-level description of your current design. This description should be understandable to non-engineers (i.e., the general public). Describe key components or sub-systems and how they contribute to the overall design. You may wish to include a basic block diagram, infographic, or other visual to help communicate the overall design.

Our proposed design consists of a wirelessly connectable device that is remotely accessible from anywhere with an internet connection through the use of an iPhone application. The main use of this device is to allow consistent monitoring of the temperature and pH levels of a fish tank, while also being able to set a feeding schedule in which the device will act out keeping the fish happy and fed at all times without any direct interaction with the fish tank.

There are multiple different components of this design that allow us to get the stated functionality. In order for the phone application to be able to connect to the tank device, we will be implementing a cloud-based backend service that will run out-of-sight to the user but will coordinate the connection between the phone application and the physical device on the fish tank itself. The phone application will communicate with the backend service through a cellular/Wi-Fi internet connection, which acts as a middleman between the feeding device and the phone app, sending any requests from the application to the physical device. This path also works in the reverse direction, allowing the monitoring information or device warning to be sent from the hardware device, over a Wi-Fi connection, to the phone application.

Context Diagrams

Diagram 1

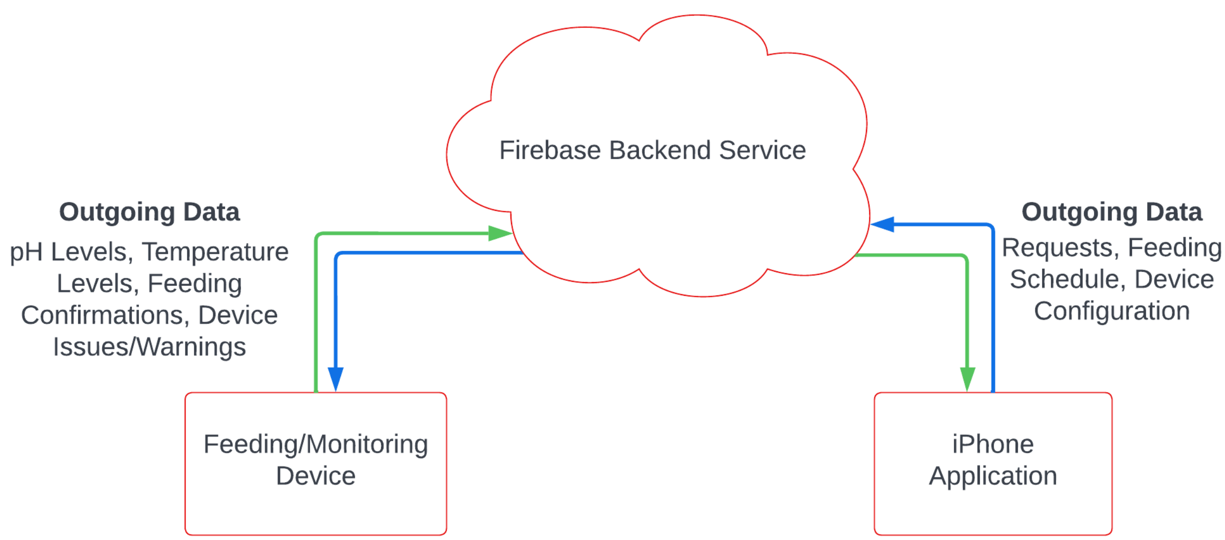
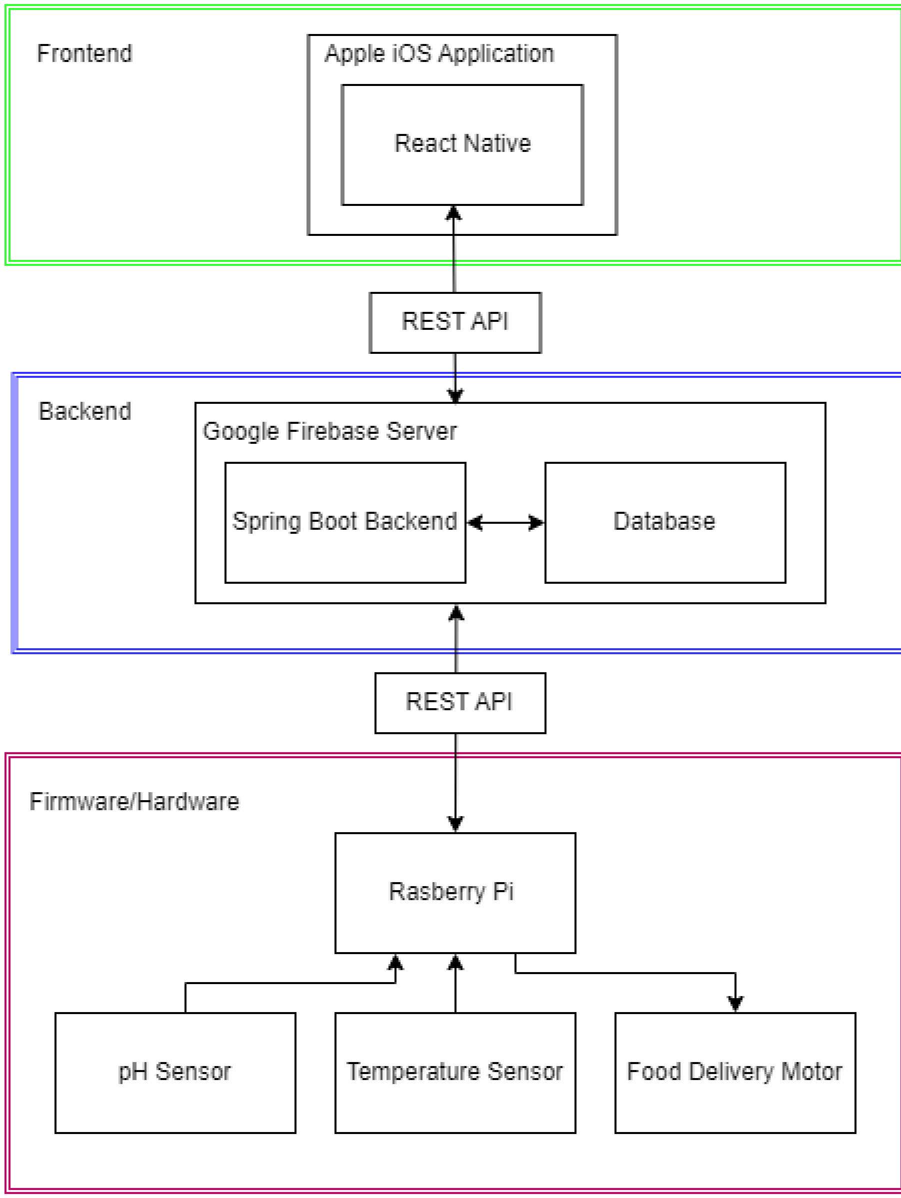


Diagram 2



4.3.2 Detailed Design and Visual(s)

Provide a detailed, technical description of your design, aided by visualizations. This description should be understandable to peer engineers. In other words, it should be clearly written and sufficiently detail such that another senior design team can look through it and implement it.

The description should include a high-level overview written for peer engineers. This should list all sub-systems or components, their role in the whole system, and how they will be integrated or interconnected. A visual should accompany this description. Typically, a detailed block diagram will suffice, but other visual forms can be acceptable.

The description should also include more specific descriptions of sub-systems and components (e.g., their internal operations). Once again, a good rule of thumb is: could another engineer with similar expertise build *the component/sub-system based on your description? Use visualizations to support your descriptions. Different visual types may be relevant to different types of projects, components, or subsystems. You may include, but are not limited to: block diagrams, circuit diagrams, sketches/pictures of physical components and their operation, wireframes, etc.*

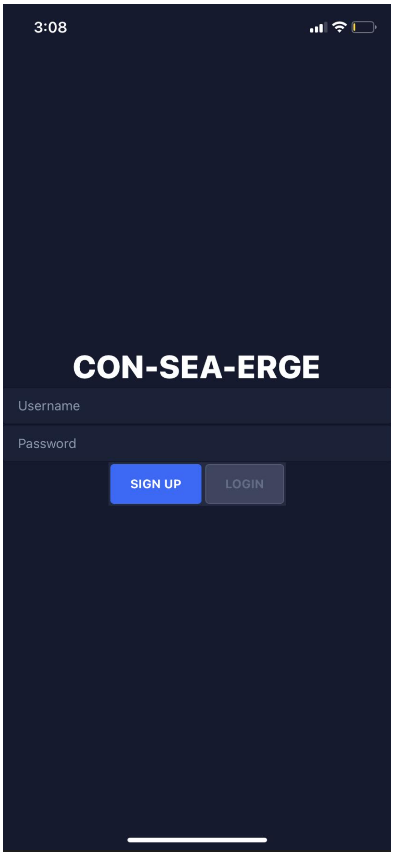
Frontend

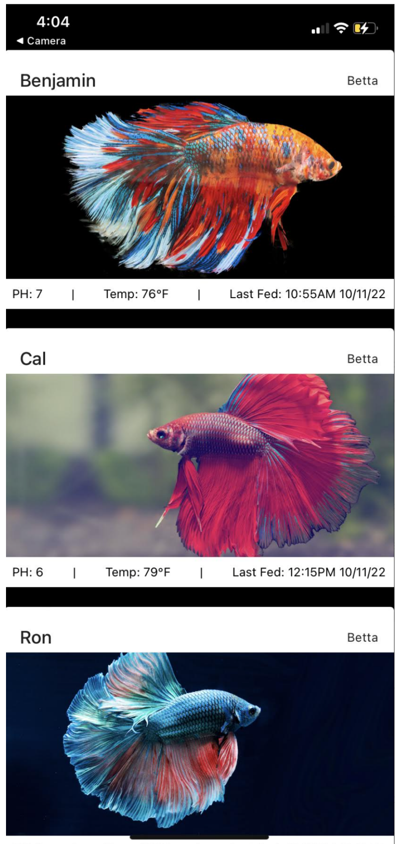
The frontend of our application will be built using the react-native language along with the Expo framework to create and initialize the application. React-native is a supplementary version of React that allows for cross platform application development. Based in javascript, it makes it easy to develop good-looking user interfaces that can be ported to both ios and android, as well as websites. Expo is the supporting framework for react-native that provides a large set of tools and makes initialization of the application simple and easy to start.

Our application will be broken down into three main screens that will provide all of the information the user will need. On opening the application, the user will be prompted with a login screen to ensure that only our clients will have access to view their fishes' information. Once logged in, the main screen will show a list of cards of all of the users fish. These cards contain the name, species, and picture of the fish, and will provide quick information to display the current PH, temperature, and last fed time for each. This screen will allow the user to monitor all of the fishes at once to make sure there are no issues with the tank.

Clicking on a fish card will open a new screen dedicated to that specific fish. This page will display the PH and Temperature circular data components that are color coded on a scale from green to red to show whether or not the tanks are in good conditions for the fish. This page will also allow the user to manually feed the fish at any given time, adjust the feeding schedule, or edit the fishes' information.

Login	Fish Cards	Fish Components

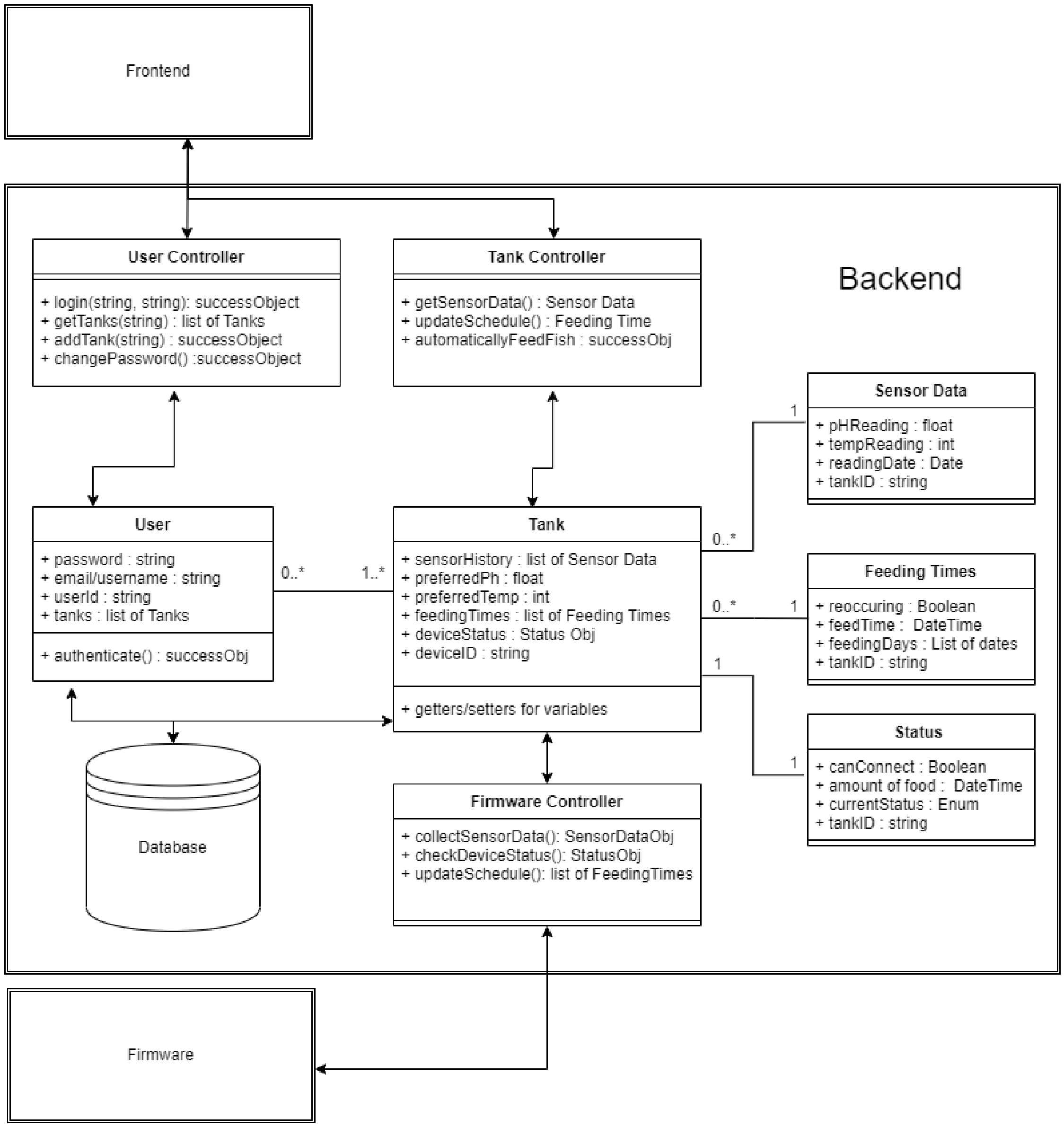




Backend

The Backend will be written in Java SpringBoot and will be deployed to Google Firebase. There will be two main class components with a User class and Tank class. The User class will handle all of the user data such as login and each user will have the ability to add and connect to new tanks. The tank objects will contain information pertaining to each feeder device which includes the history of the sensor data, a list of feeding times, and the status of the tank. Data from the User and Tank classes will be stored within a database

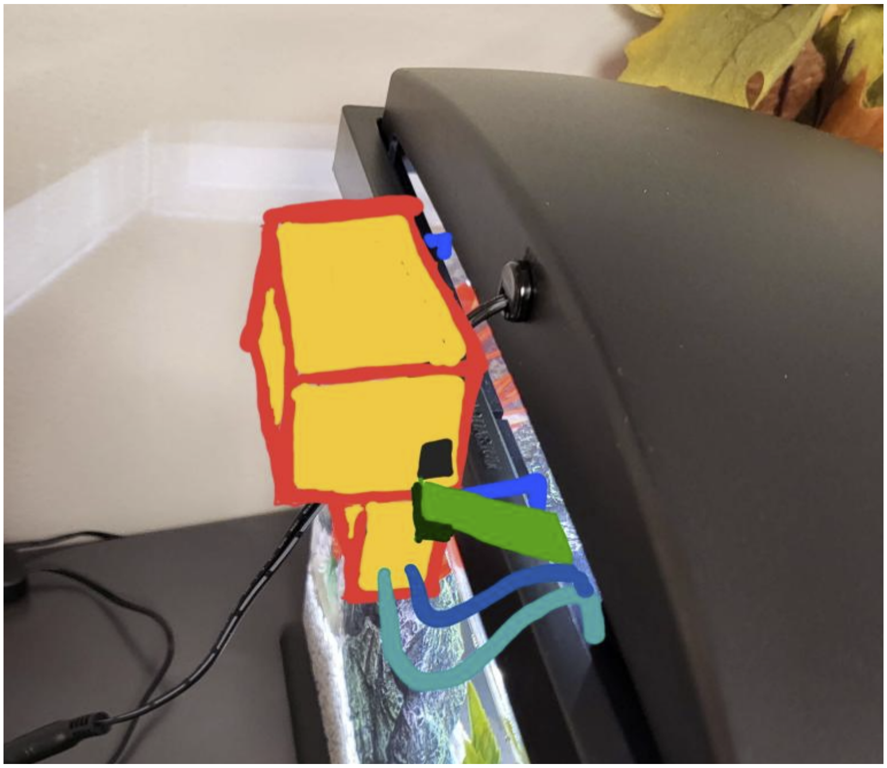
The system will have 3 front-facing controllers. Two for the frontend to interact with and one for the firmware to interact with. Each controller will have a form of authentication for the messages that get sent to them in order to prevent requests from being made that shouldn't be.



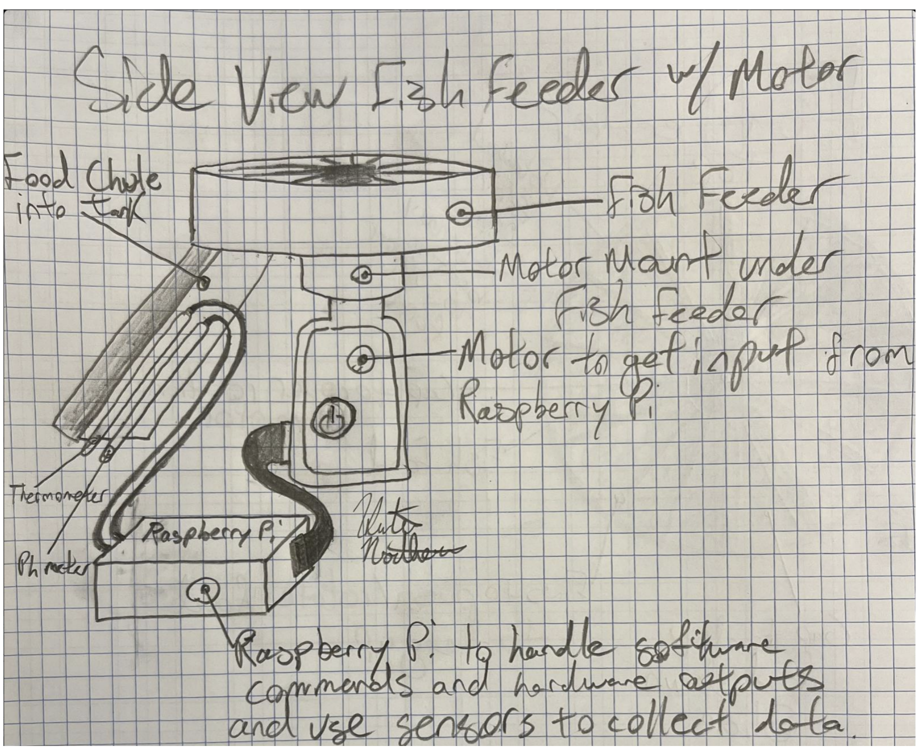
Hardware/Enclosure

The enclosure is based on our drawings and will be designed in Solid Works / CAD to then be 3-D printed to fit our components such as the probes, firmware, and food. We are basing our designs off of 2 different enclosures (A and B shown below). Enclosure A will use a mechanized corkscrew method to push food out of the enclosure down the chute into the fish tank allowing the number of rotations to be controlled and putting out a controlled amount of food while still holding and enclosing all of the components of our design. Enclosure B will use a sectional rotator to move a controlled amount of food around which then drops down the chute into the fish tank to feed the fish and keep all the components consolidated.

Enclosure A:



Enclosure B:



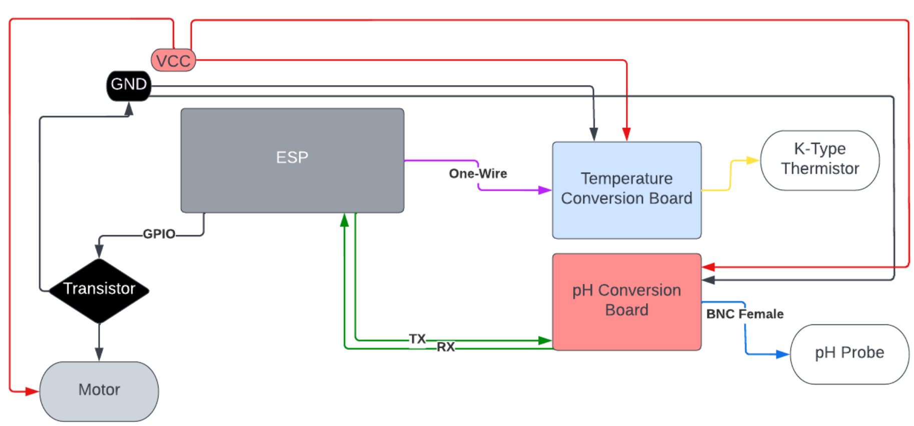
Hardware/Chip-Down

The PCB hardware that we are using in this project is based on a custom chip-down design. This allows our team to select very specific components for the functionalities that we require to be included in the hardware of the device itself.

We will be including the components listed below:

- pH Probe
  - Consumer-Grade pH probe
  - This type of probe was chosen because of its price compared to higher-end grade probes, as well as the accuracy of 0.01 pH is above our requirements.
- pH Conversion Board
  - Module: EZO-pH
  - This conversion board allows us to simply take readings from the pH probeand translate these readings into meaningful pH values quickly.
- Temperature Probe
  - K-Type thermocouple
  - K-Type thermocouples are less prone to oxidation and acidity than their counterpart of J-Type, which is important for a water application
- Temperature Conversion Board
  - Module: Adafruit-1727
  - This conversion board allows us to simply take readings from the thermocouple and translate these readings into meaningful temperatures quickly.
- Wi-Fi/BLE Enabled MCU
  - Module: ESP32-C3-MINI-1
  - This component will enable the device to have an internet connection through Wi-Fi while as well supplying the ability to connect to a phone over the BLE service.
- DC Motor
  - Compact, DC motor geared for torque
  - We do not need a quick motor as the dispensing of food has to be relatively slow in order to dispense the correct amount of food. As well, a motor geared for torque over speed will jam less frequently on foot pellets than compared to a speedier motor.

These hardware components will be connected in accordance with the following diagram:



This device will also feature and LED for warnings and direct information about device operation.

(This is not pictured in the diagram above)

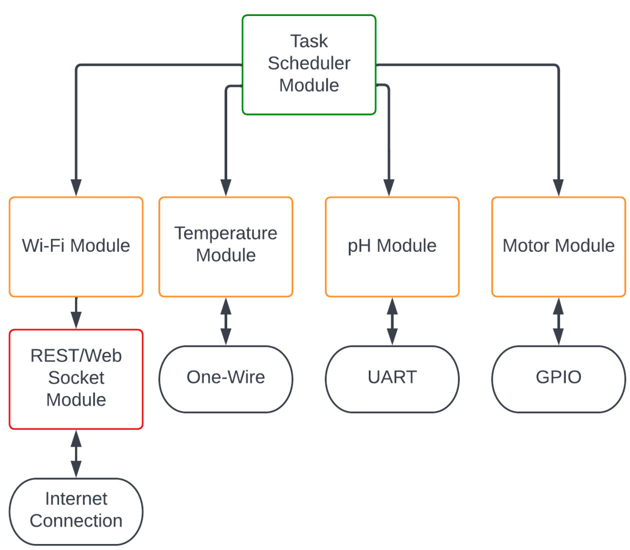
Firmware

The firmware will include a modular design to allow for easy portability of the code and allow for simple changes down the road, if necessary.

The main modules in the firmware are listed below:

- pH Module
    - Requests and receives pH information from the pH conversion board
    - Communication Protocol: UART
- Temperature Module
    - Requests and receives temperature information from the temperature conversion board
    - Communication Protocol: One-Wire
- Motor Module
    - Controls the actuation of the motor
    - Communication protocol: GPIO
- Wi-Fi Module
    - Allows ability to connect to internet
    - Receives/sends requests to the backend service
- REST/Web Socket Module
    - Configures and creates requests to the backend service in the specified formats
- Task Scheduler Module
    - Main control module that handles the timing of the device application and the initiation of device actions

The class hierarchy is displayed in the diagram below:



4.3.3 Functionality

Describe how your design is intended to operate in its user and/or real-world context. What would a user do? How would the device/system/etc. respond? This description can be supplemented by a visual, such as a timeline, storyboard, or sketch.

Once our design is fully functional, the feeding mechanism will be able to feed fish in the tank on a specified schedule and report pH and temperature data from the tank to the app. The functional app will allow users to log in, view/set feeding schedules, and monitor pH and temperature levels of specific tanks.

A user would log into the mobile application and be able to view all the tanks that they monitor. They will be able to see the current pH and temperature with the current reading. Our device will take these measurements every 5 minutes then send that data to the backend and our backend will send that to the frontend to be displayed to the user. If the pH or temperature get to a dangerous level for the fish, a notification will be sent out to the user to notify them. A feeding schedule for each tank will be able to be set and viewed by the user. Whenever the schedule is updated, the backend will send that data to our device and it will save that to the non-volatile memory to be used from then on. If the internet connection disconnects on our device, the schedule will still be acted upon as it is still in the memory. Using the internal clock, we can do calculations to continue the schedule.

For the enclosure our product will have the functionality to consolidate all of our components and to push food into the fish tank in a controlled manner in regards to serving size and component structure.

4.3.4 Areas of Concern and Development

How well does/will the current design satisfy requirements and meet user needs?

Our current design encompasses all the requirements and user needs outlined by our clients. They emphasized some key aspects that we took into account such as it being a mobile application, having the ability to set and view a feeding schedule when they are out of office and being able to monitor both pH and temperature remotely. As long as there aren't any immovable blockers, our design completes the project's requirements.

Based on your current design, what are your primary concerns for delivering a product/system that addresses requirements and meets user and client needs?

Will our connection from frontend to backend and backend to firmware work correctly?

This was an initial concern for us going into this project. If we are not able to solve a problem, it would be detrimental to the success of our project. Over the course of implementing all the separate sections, we researched how to connect all three and it looks like we will be able to do it just fine using our experience in the field.

Will our development board be able to handle staying on schedule without WiFi connection? Will it stay on schedule after it reconnects?

Our clients made it clear that this should be a goal to have it stick to the schedule while being disconnected from the WiFi and also get back on schedule after reconnecting. Looking deeper into all the components on the Development board, we will be able to keep track of time using the internal clock and comparing that to the last time that the fish was fed. While this might be difficult to implement, we believe that we are capable of accomplishing this as the firmware team is very experienced in low-level programming.

Will our backend hinder the performance of the systems?

The main concern here is that the backend needs to be constantly active so that it can send out updates to the device from the mobile act. The backend works as a middle ground between the two front-facing devices and it is imperative that the backend doesn't hinder the processes of either the mobile app or the physical device. This means that if the backend is down you will be unable to view data from the firmware on the mobile app. We believe that we will be able to accomplish this by deploying the app onto a cloud service to make it constantly available for the users. Once this is done we can do testing to make sure that the data flow is fast enough and consistent enough to make the product a pleasant experience for the user.

How to roll out an application for iOS devices?

Currently we have thought of several ways to get the app to our clients. First thought would be to release the application to the App store but it is seemingly unlikely as the user base is quite small and limited with the App store having several restrictions to be published and required ongoing maintenance as well. Our second thought would be to directly install the application through a devtool but have to do extensive research to learn how to circumvent certain restrictions.

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

How fast would the client like changes to the device to go into effect?

If an iPhone application isn't possible, would a web application suffice? (More research and thought needed)

4.4 Technology Considerations

Describe the distinct technologies you are using in your design. Highlight the strengths, weaknesses, and trade-offs made in technology available. Discuss possible solutions and design alternatives.

SpringBoot

SpringBoot is a fairly popular backend framework and it is used in a lot of places to create microservices. The framework is open source which means we do not need to pay to use the framework and it has many resources available for us to learn more about it. One of the main reasons we went with this framework over others was due to our team having a higher exposure to the language of Java in our courses as well as some of us having professional experience using Java Spring Boot. Other alternatives could be something like Ruby on Rails or .Net, however, these frameworks are written in other languages that the team is less familiar with.

Firebase

Firebase is one of the leading backend frameworks nowadays, and it pairs great with our app because of its lengthy list of features and generous free package. Firebase's free tier allows for a large number of authentications, database hits, and cloud uploads/downloads, so the fact that our app will require a relatively small number of each of these items makes the free tier a great option. In addition, one of the features of Firebase is that there is no need to set up or connect to an external server, as Google handles all of the hosting on its own. One of the aspects of Firebase that has proven to be challenging so far has been getting it connected to our app since we are using React Native for the frontend. This is particularly challenging because Google has simple setup procedures for frameworks such as Flutter, but there is a bit more to work through when connecting to a React Native app.

React

React is heavily used throughout the world. There are plenty of libraries, documentation and support regarding the framework. By using React, we can use the various libraries that are widely available via open-source. Primary reason that we chose React is that our team is quite familiar and comfortable with React using JavaScript or TypeScript. Alternatives include using AWS Amplify to create the application or Flutter but these tools/environments are unfamiliar territory to the team.

Chip-Down Design/Firmware

One of the main reasons that we went with a "chip-down" design instead of moving forward with an off the shelf solution for the PCB, such as a Raspberry Pi, was that the off the shelf solutions were not readily available and in stock. Deciding to go with a more customizable solution allows our product to be specifically tailored to the purpose that we are designing it for, as well allows us to decrease the amount of unnecessary hardware in the device.

The selection of the ESP as the main MCU that will be running our design allowed us to easily know which language and IDE to be using since the ESP module code is written in C and uses an IDE tailored to that device. With a Raspberry Pi solution we would be able to use different programming languages, but being that C is one of the lowest-level full programming languages, this would allow our code to be as quick as possible allowing for the best performance and configurability.

The main trade-off with moving forward with a chip-down design is that it increases the project complexity. Raspberry Pi's are extremely easy to work with and have a lot of community support online if issues arise. That main benefit of the off the shelf solution allowed us to consider that as a viable option over the rest of the benefits of the chip-down design. After weighing the pros and cons of the two, it seemed going with a chip-down design would be better for the project as a whole.

Enclosure

The enclosure is only limited by Solid Works specifications and 3-D Printing capabilities but both technologies are heavily researched and time-tested so in terms of enclosure design we will be able to produce a product that meets our specifications.

4.5 Design Analysis

Discuss what you have done so far, i.e., what have you built, implemented, or tested? Did your proposed design from 4.3 work? Why or why not? Based on what has worked or not worked (e.g., what you have or haven't been able to build, what functioned as expected or not), what plans do you have for future design and implementation work? For example, are there implications for the overall feasibility of your design or have you just experienced build issues?

For the frontend, we have some simple components created such as components for visually representing pH levels and temperatures. We created a card component representing fish tanks and a login screen. Currently we have dummy data to show on the frontend while we work on connecting to the backend; waiting for the firmware to be able to provide data to the backend as well. We have not started testing components or APIs as of yet but plan to do so when we can run some test calls. We also plan to add visual unit and integration tests for our frontend components in the future. We also plan to implement a CI/CD pipeline to ensure our deployment is successful and that each consequent update would not break the application. For the backend, we have researched potential backend services and decided that we will use Google Firebase to host our backend. Our project is setup in Firebase, and we are currently working on integrating it to the frontend so that information can be passed and tested. The current issue that we are working through revolves around assigning a frontend type to the project so that connections can start to be made. As for the firmware, we have figured out how to send code to our Development board using the Arduino IDE using C as the programming language. We have been able to connect to the WiFi using the Wifi module. We are currently looking into handling REST api calls after that, using Postman to test that. Later, we will actually connect the board to Firebase once the Backend gets fully set up. As for the sensors, this week, one person of our firmware team will focus on that while the other person on the team will focus on the WiFi aspect. This week, we plan on working on getting the sensor code implemented where it will output both the temperature and pH level. Overall for the firmware aspect, we have not run into any issues as of now.