Con-Sea-Erge
Automatic Fish Feeder and Monitor

Final Report

Team 25
ISU Accounting
Advisor: Dr. Fila
Client: ISU Campus Organizations Accounting

Devin Milligan/Firmware
Ethan Peterson/Firmware
Drake Dodson/Backend
Ryan Hickok/Backend
Josh Van Drie/Frontend
Brian Tran/Frontend
Hunter Northern/Hardware
sdmay23-25@iastate.edu
https://sdmay23-25.sd.ece.iastate.edu

Table of Contents

Introduction

Our focus for this project is the problem of having to feed your fish when you are away from home or on vacation. Our clients work in the ISU Campus Organizations Accounting office, where they keep multiple separate fish tanks. This raises the issue of who will feed the fish on vacations, weekends, or other times out of the office. So we have set out to solve this problem with a device that will allow users to remotely feed their aquatic friends through a mobile application connected to an IoT-enabled fish-feeding device.

Product

For our solution, we created a mobile web application that gives you control over a WiFi-enabled smart fish-feeding device that will allow you to feed your fish remotely for up to a month. The application allows you to set different schedules for your fish to be fed. As well it allows you to get continuous updates on the water quality in your fish tank, with the water temperature and pH being monitored. This solution allows our clients peace of mind knowing that their fish can be looked after in or out of the office.

Evolution of Design

By the end of 491, we had completed our project plan and began implementing key portions of the project. The base of our frontend application was up and running and able to send data to the backend, which was functional enough to receive and store data. On the firmware side of things, we were able to send and receive data to and from the backend, and the temperature and pH sensors and WiFi chip were fully implemented and functioning properly. From a hardware standpoint, we were in the early stages of planning the enclosure and had a couple of different design ideas that were in the modeling stage of development.

Since the end of 491, we have been able to complete the project and check off each of our initial requirements. In doing this, we continued frontend development as we had in the fall, and we shifted gears with the backend to not only store text but also store images and deploy the application to the web. In terms of firmware, we continued testing the temperature and pH sensors as well as implementing the motor and scheduling features of the device. The development of the scheduling functionality began with getting the motor to turn at appropriate times and eventually included retrieving scheduling information for each individual day so that the fish could be fed manually on some days. Lastly, the enclosure design is the portion of the project that changed the most since the end of 491. We ran into numerous issues with the initial designs that eventually led to their downfall and required us to start from scratch and keep the 3D printing process in mind while creating models. After numerous iterations and prints, we were finally able to come up with a design that functions properly. Overall, our design process

shifted considerably since 491, and many of the changes helped lead to our success in wrapping up the project this semester.

Requirements

The requirements layout the fine details that, when satisfied, will complete this project. These are set as benchmarks of what is needed in each section to be marked as completed. The product should do no less than what is listed here for a successful product.

Hardware Requirements

These requirements ensure that the components selected to be on the circuit board are necessary and complete the functionalities required by the product. Each functionality required here must be matched with a component that will fulfill it, and this outlines what will be needed to be on the circuit board.

Sensing Technologies

- ➔ pH
  - ◆ Ability to detect the pH to an accuracy of 0.1pH
- ➔ Temperature
  - ◆ Ability to detect the temperature to an accuracy of 0.1℉

Device Enclosure

- ➔ Food Input
  - ◆ Food is easy to place into the enclosure
  - ◆ Enclosure is able to hold at least 1 month worth of Betta fish food
- ➔ Food Output
  - ◆ Customizable amount of food is able to be dispensed
  - ◆ Food is able to be dispensed to the inside of the tank from the external enclosure
- ➔ Mounting
  - ◆ Device will be mounted to the tank externally
  - ◆ Mounting apparatus must not require any physical alterations to the tank

Device Power Supply

- ➔ Wired Power
  - ◆ 5V Input

LED Indicator

- ➔ RGB LED
- ➔ LED Warning Indications
    - ◆ YELLOW: Not Connected to internet
    - ◆ RED: Feeding motor jammed
    - ◆ BLUE: Normal Operation
    - ◆ GREEN: Doomsday Mode

Frontend/Application Requirements

These requirements pertain to the mobile application to be made for user interaction with the IoT fish feeder. These ensure that the application will be able to meet necessary functionalities, while being enjoyable and easy to use for the user. The application is the main interaction point with the whole system for the user, therefore these requirements affect the outcome of this product to a high degree.

User Experience

- ➔ User is able to login and manage tanks
- ➔ Users are able to switch between tanks with confirmation
- ➔ Tanks are labeled and able to be configured
- ➔ Notifications
    - ◆ Feeding confirmation
    - ◆ pH level warning
    - ◆ Temperature level warning

User Interface

- ➔ Feed command is able to be sent to the device
- ➔ Users are able to view last feeding date and time
- ➔ Users are able to set a weekly feeding schedule

Backend Connectivity

- ➔ Feeding timestamp is retrieved and displayed
- ➔ Feeding schedule is able to be manipulated

Device Monitoring

- ➔ Users are able to view PH level of each assigned tank
- ➔ Users are ability to view temperature of each assigned tank

Device Actions

- ➔ Users are able to dispense a desired amount of food at any time
- ➔ Additional fish/devices are able to be added
- ➔ Temp warning levels are set and displayed
- ➔ pH warning levels are set and displayed
- ➔ Manual feeding is available

Hardware

- ➔ Application will be available on iOS devices

Backend Requirements

The backend requirements were chosen in order to fully support the functionality of the frontend application and the embedded device. These requirements ensure that the backend will be able to communicate with the two other modules and store the information necessary for clients and sensor monitoring.

Frontend Connectivity/API

- ➔ Connection to the frontend application is present, and relevant data is able to be accessed via backend databases
- ➔ Login verification between backend and frontend is seamless

Hardware Connectivity/API

- ➔ Connection to firmware device is present, and data sent by the device is able to be received by and stored in the backend
- ➔ Real-time updates to the database are able to be made based on data sent from the device

Backend Processes

- ➔ Feeding schedule is accurately followed so that feedings are triggered at appropriate times each desired day
- ➔ Login information, connected devices, previous feedings, scheduling information, and monitoring levels are stored in the database
- ➔ Data is able to be passed between the frontend and firmware via the backend, including feeding signals and temperature and ph readings

Firmware Requirements

The firmware on the device needs to work continuously and be able to handle any circumstance that the feeder is put in to ensure that the fish continue to get fed. These requirements pertain to the physical functionality of the product, such as the motor dispensing food and the water attribute sensing. This ensures that the device will be able to perform the functions required by it.

Wireless/Backend Connectivity

- ➔ Firmware is able to connect wirelessly to the backend
- ➔ WiFi connection is present through the use of a WiFi-enabled chipset
- ➔ Monitoring information is able to be pushed to the backend
- ➔ Data is able to be read from the backend so that feedings are able to be signaled at appropriate times

Feeder Control

- ➔ Device is able to dispense a customizable amount of food using values configured in the frontend and stored the backend

Temperature Monitoring

- ➔ Temperature levels of each tank are able to be monitored by the device and passed to the backend for storage

pH Monitoring

- ➔ pH levels of each tank are able to be monitored by the device and passed to the backend for storage

Implementation Details

The implementation of our product concerns different modules that work together. These modules are the Backend, Frontend/Application, Embedded Device, and Enclosure. Together they form our product as a whole. *Figure 1.1.* shows the connection of information between the device, frontend, and backend service. Information flows between these modules such as, pH data, temperature data, user logins, and feeding data.

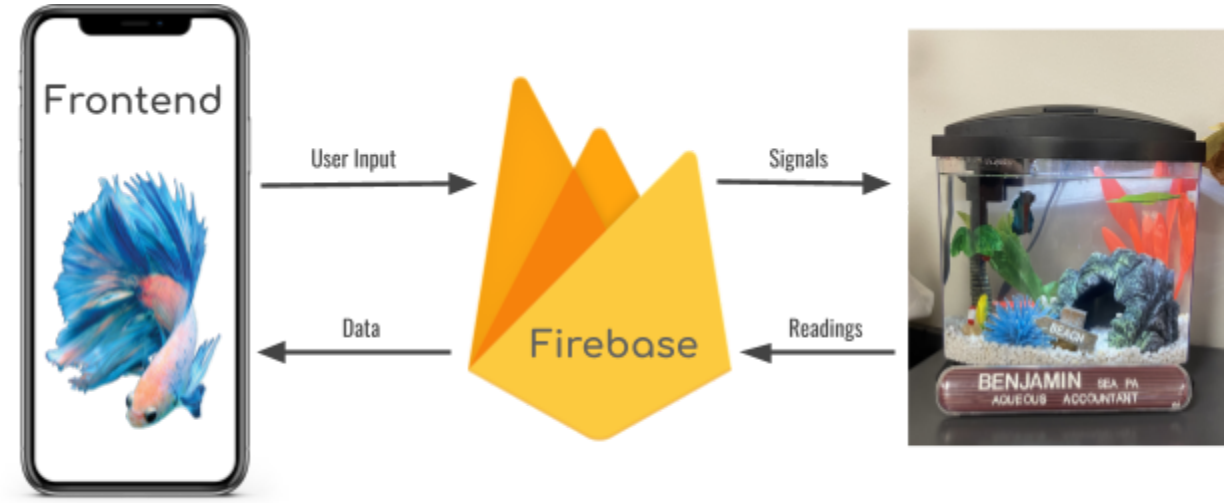


Figure 1.1.

Enclosure

The enclosure is a mechanically moving assembly that is needed to house the electronics of the project, as well as to evenly and consistently dispense fish food when necessary. The device enclosure is a 3D-printed assembly that is composed of 11 different components. Each of the components is printed out of PLA material. The whole enclosure, when assembled, is a side-unloading dispenser, which uses an auger-style spiral to empty out the top-loaded hopper, as shown in *Figure 1.2.*

Functionality

The contents to be dispensed are loaded at the top of the device where the hopper is located. The hopper was designed to fit at least a month's worth of fish food within the design. The food is then directed toward the spiral dispenser where it will be funneled to the slide to fall into the tank.

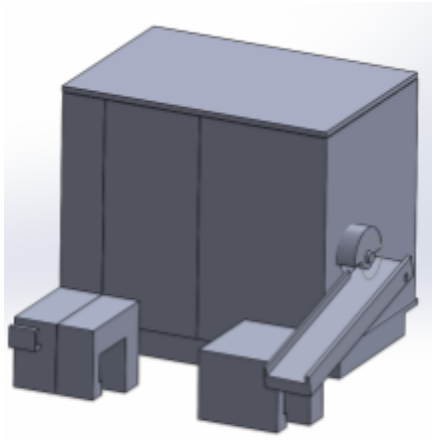


Figure 1.2.

The circuit board and motor that operate the enclosure are stored in the bottom of the device, held by a base secured with screws. The motor then interacts with the gears housed in the bottom compartment, where its power is redirected to turn the spiral dispenser.

The PH and temperature probes exit the device through a hole on the left side of the device. These cables are routed through the cable clip on the front of the left tank arm.

The tank arms on the enclosure are designed to fit the type of tank that the clients are using. They attach to the back of the tank with no effect on the tank lid's placement.

The main portion of the enclosure measures 10x7x8cm.

Main Components

Figure ii. shows the main components that make up the functionalities of the enclosure.

Shown in *Figure 1.3.*, part a., is the left side of the main body of the enclosure. This part is where the internal cables exit the device, as well as where holes are placed so the LED indicator is visible from the top of the device. Part b. is the middle of the main body that holds the motor and gears, as well as the slot for the end of the Spiral Dispenser to enter. Part c. of the enclosure is where the contents of the hopper are dispensed and where the food slide attaches to. These three sections of the main body of the enclosure are glued together to make a seamless print. They are printed in three sections to make the print quicker and more accurate without print supports.

Part d. is the Food Slide, where the food is directed into the tank after it is dispensed.

Part e. is the Spiral Dispenser which empties out the hopper of its contents by using an auger-like style to funnel the contents towards the output hole.

Part g. are the motor and spiral gears. These attach to the motor and the Spiral dispenser to redirect the power of the motor in the direction necessary to turn the Spiral Dispenser.

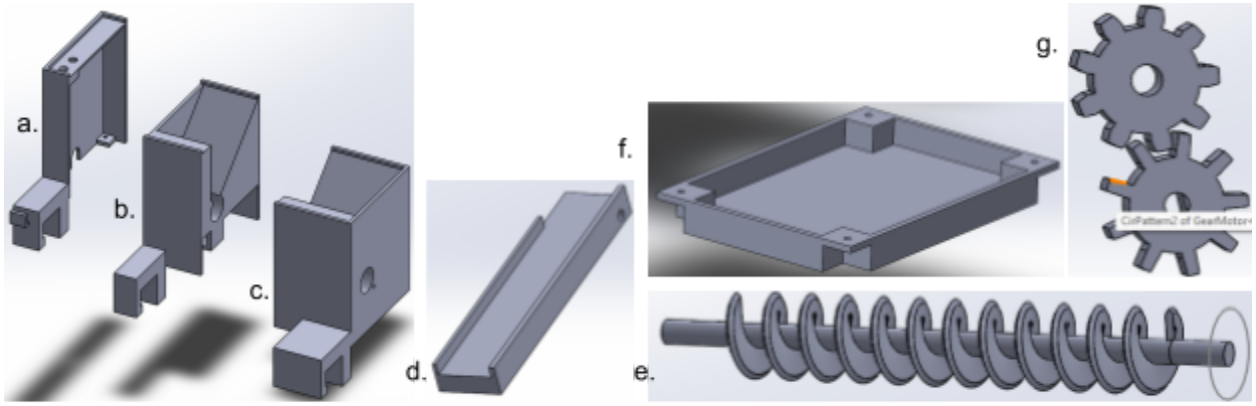


Figure 1.3.

Hardware Components

Components for this project were selected in order to fulfill the requirements that our product needed to fulfill. The circuit board needed the functionalities of WiFi, Bluetooth, pH sensing, and temperature sensing. These components communicate with each other in order to act as one complete system to perform the necessary tasks.

Main MCU/Bluetooth/Wifi

ESP32-C3-DevkitM-1ESP32, shown in *Figure 2.1.,* is the development board that was selected to be the main MCU for the project. This module accomplishes Bluetooth functionality, as well as WiFi.

Figure 2.1.

Temperature Sensing

A K-Type Thermocouple was used as the probe for the temperature sensor. The MAX31858, *Figure 2.2.*, chip allows us to take readings from the thermocouple and translate them into Celsius or Fahrenheit temperatures.

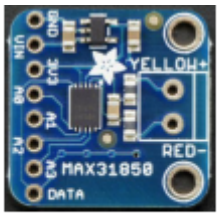


Figure 2.2.

pH Sensing

A long body pH electrode probe is used to be inserted into the tank's water for pH readings. The EZO-PH, *Figure 2.3.*, development board allows us to interpret the readings from the pH probe into pH levels.

Figure 2.3.

Components Connection

The components are connected with different protocols to communicate. *Figure 2.4.* shows what protocols the components communicate with, as well as which other module they are communicating to.

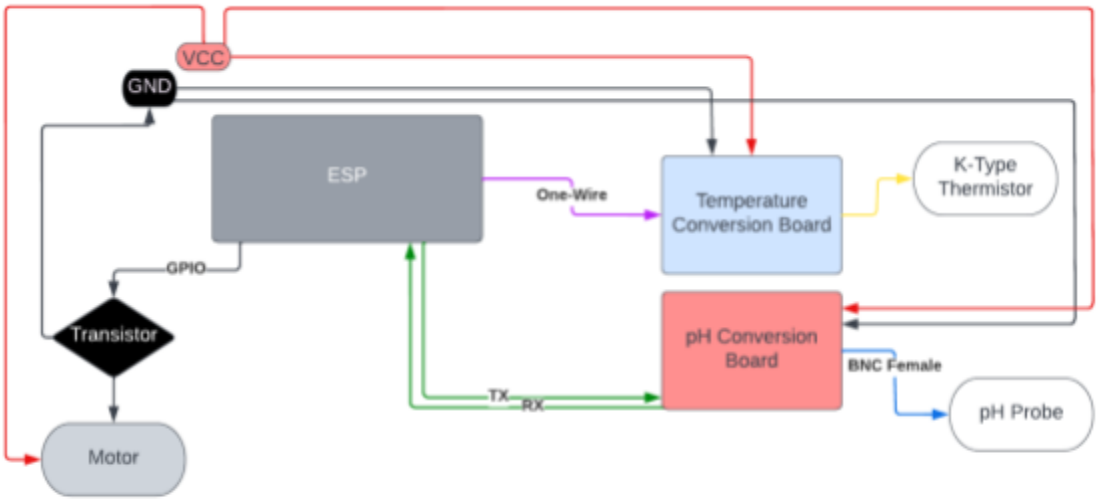


Figure 2.4.

The protocols utilized are UART, One-Wire, and GPIO.
Figure 2.5. is the schematic of the pin-to-pin connection on the circuit board shown in *Figure 2.6.*

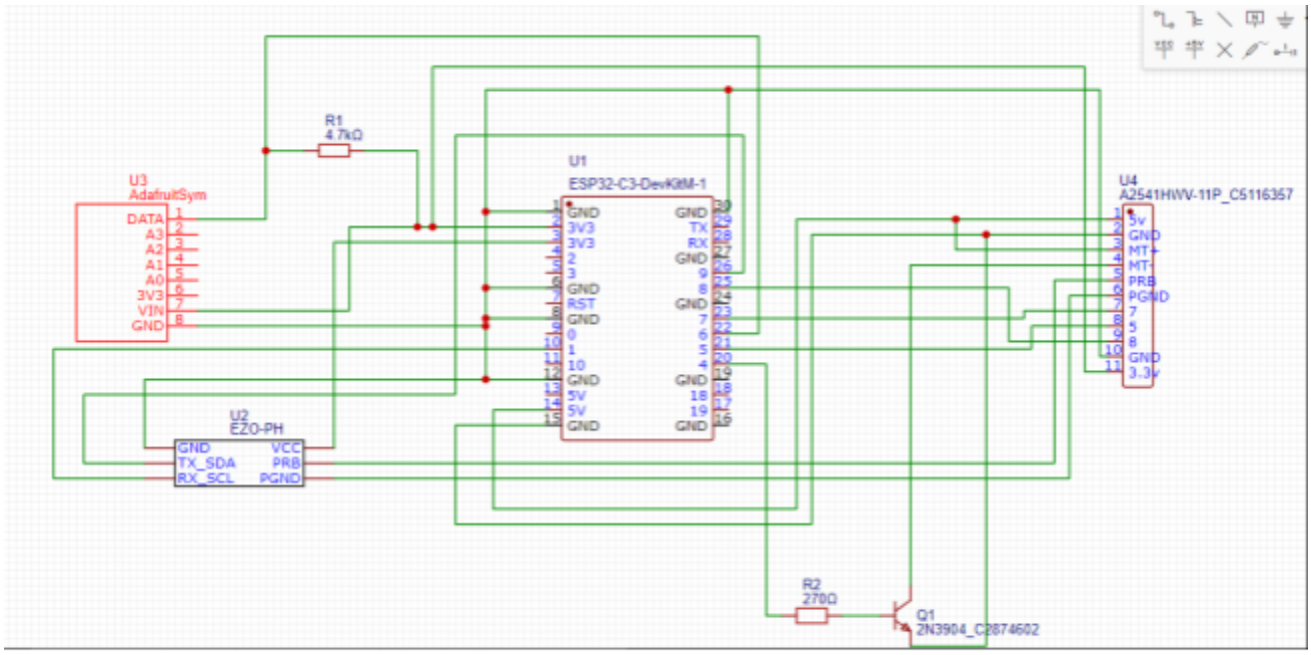


Figure 2.5.

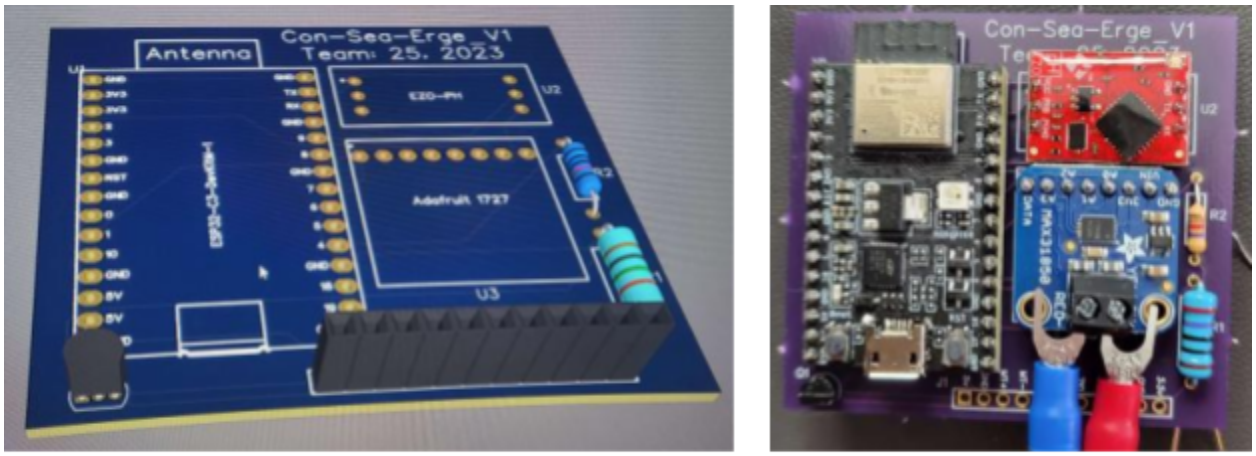


Figure 2.6.

On the circuit board, there is an un-populated header, listed on the schematic as J4. This header is for connecting the PH and temperature probes, as well as the DC motor. Also, included are pins for IO on the ESP, and 3.3v power was provided in case additional sensors were added later.

Embedded Firmware

Overview

Our firmware is responsible for connecting the chip to the ISU Wifi, connecting to our real-time database, reading and updating the Temperature and pH metrics of each tank as well as rotating the motor to feed the fish according to a user-inputted schedule. To accomplish this, we used helper classes and useful libraries. Our code in the firmware was implemented on the Arduino IDE as it is very useful for similar embedded projects and included many powerful libraries for us to use. Our firmware implementation can be described with the following diagram(figure vii.).

Helper Classes

We currently have two helper classes, one for our pH and one for our Temperature as well as header files for both of them. The pH helper class has an initialization function as well as a getpH function. The Temperature helper class has an initialization function and a getTemperature function. These classes help reduce clutter from our main class.

Libraries

To support all of our functionality on the firmware side of things we are using ArduinoJson, Firebase ESP32 Client, MAX31850 DallasTemp, OneWire, and Time libraries. These libraries are

used to send and receive data from our Firebase database to our esp32, MAX31850 DallasTemp and OneWire are used for the temperature sensor, and the time library is used to the scheduling functionality.

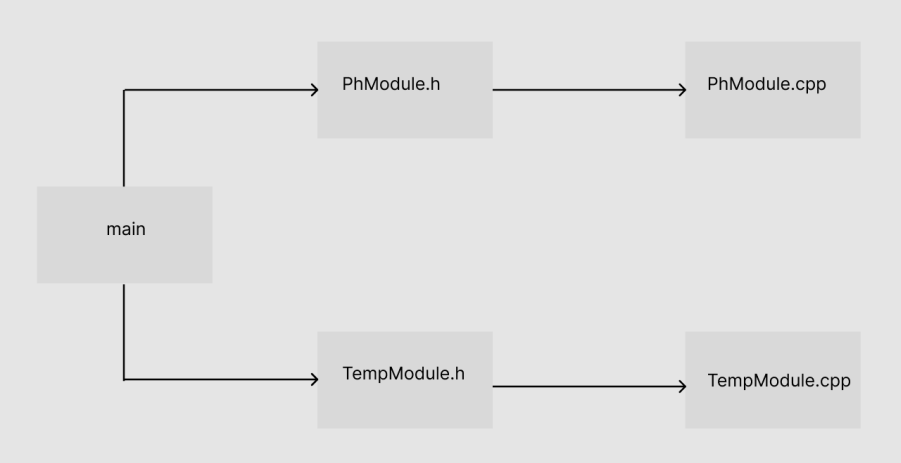


Figure 2.7.

Frontend/Application

Overview

The frontend consists of a website con-sea-erge.web.app designed with mobile usage in mind. The web application consists of four main pages: Login/Signup, Home, Tank Details, and Settings. The app is written in react native which supports web, ios, and android. The app was originally developed to be a standalone iPhone application and later converted to a web application in order to avoid a $100 a year fee to the Apple developer program. Therefore, the app is supported on web browsers, however it was designed with mobile screen sizes in mind.

Login / Sign Up

Login allows users to sign in with their email address and password, or they can sign up for an account on the sign up page. New users automatically gain access to the two physical devices.

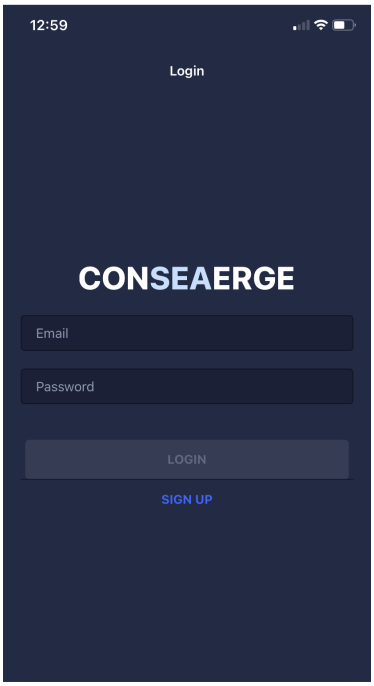


Figure 3.1

Home Page / Fish Cards

The home page displays fish cards, which contain quick information about the tank as well as a user uploaded photo of the tank. Information displayed on the fish cards include the fish's name, species, ph level, temperature, and last fed date or time. Clicking on a fish card will open the detailed info page for that tank.

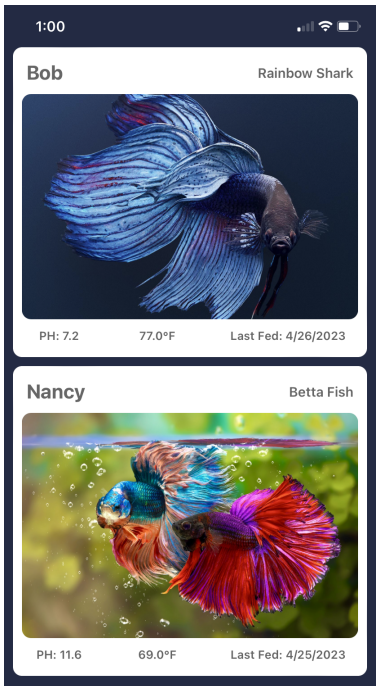


Figure 3.2

Detailed Tank Info

The detailed information page displays all the information about that specific tank. PH level and temperature are displayed at the top in a circular component that will change color based on the ideal temperature and ph levels for small tank fish, specifically bettas. This page also contains the feeding scheduler. There are three feed options per day, breakfast, lunch and dinner. These times are fully customizable to any time of the day and can also be individually disabled. The weekly scheduler allows you to schedule days of the week to feed the fish. This will feed the fish at the given times selected in the feeding scheduler for only the days that are selected. The bottom of the page displays the last fed date and time, along with the temperature and ph level accurate to two decimal places.

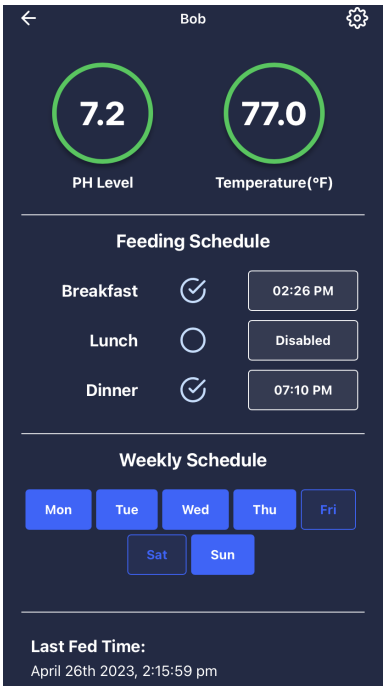


Figure 3.3

Settings

Each tank has their own settings page. This page displays the tank id of each tank and allows the user to edit the name of the fish and its species. The species dropdown contains a preselected list of the twenty five most common fish species for small tanks in the event that the user purchases a new fish. The settings page also contains the manual feed button which allows the user to feed the fish at any given time without having to set a schedule. In addition, the user has the option to change their tank image by uploading a photo and confirming the change before it is submitted.

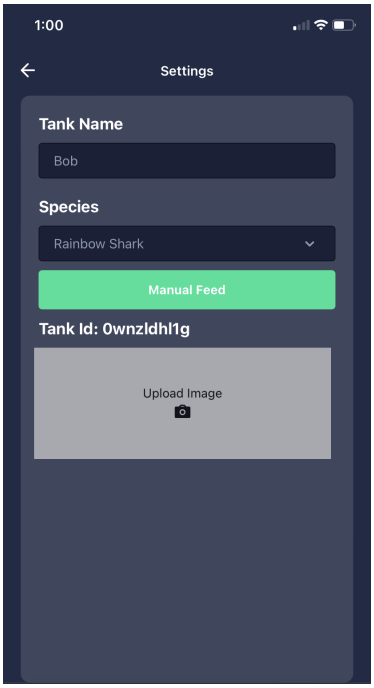


Figure 3.4

Backend

Overview

The Backend is built using Google Firebase SaaS. The application utilizes many of the features that Firebase provides and the backend is connected from the frontend to the firmware device to provide both of them with access to these services. For our specific clients use-case we fit well within the data usage for the free-tier of Firebase so we shouldn't have to pay anything for using the service. However, if we were to scale the app that is a possibility and Firebase makes it really easy to scale these services if we so choose.

User Authentication

Google Firebase simplifies the authentication process for the application. We did not have to worry about things like email services and storing encrypted passwords as Firebase handles all of that for us. In the frontend application there is a connection to the backend that is open and sends information to Firebase when a user creates an account. We then create an entry that stores the new users UID in the Realtime Database to retrieve specific information. The Authentication service also has features to delete accounts or send out password reset emails.

Realtime Database Storage

The Realtime Database is where all of the information for the user specific data and the tank specific data is stored. Each user has their Within each tank, we store temperature and pH readings along with scheduling information that allows users to manually feed at any time or set a feeding schedule that can be toggled on or off depending on the desired day of the week. Values representing the last feeding date/time and connection status are also saved to the Realtime Database.

Figure 4.1

Cloud Storage

We utilized Firebase's Cloud Storage application to send and receive pictures for the users tank photos. Right now the application only stores two pictures for the two tanks but the application has the ability to store more if needed.

Deployment Pipeline

All of the web application code is able to be deployed to Firebases Hosting service. All we have to do is compile the code and push the code to Firebase and it will update the web application code onto the URL con-sea-erge.web.app.

Engineering Standards and Constraints

Engineering standards and constraints are important for a project as they describe industry standards that are implemented into your product. These standards ensure that all functionalities involved with them are implemented safely and correctly to mitigate further issues that could occur with improper implementation.

IEEE 802.11

This standard is for use with WiFi-enabled devices to ensure that they follow the correct protocols and specifications of a proper WiFi connection. This is a large aspect of our project due to our IoT device connecting to the backend service using only a WiFi connection. Therefore, we have to confirm that this connection follows all standards to ensure that this connection will stay up indefinitely without issues.

User Secrets

It is important that we take user data, such as names and passwords, and store them in ways that make it difficult for that information to be seen and accessed by others. Any kind of sensitive user information should not be sent in plain text form over a wired or wireless connection. Firebase security keeps our sensitive data secure for this product's application.

Code Reviews and Merge Requests

The team followed an Agile approach to doing code reviews and merge requests. We made sure that all code that was pushed to the code base was first pushed to a separate branch and made into a merge request. This way we were able to review each other's code to get a better understanding of what changes were made.

Testing

Frontend

Most of the tests that were done on the frontend side were to verify that things looked the correct way and that data was being retrieved correctly from the backend database.

Hands on Client testing

In order to get a better understanding of how the application would be used by our clients we allowed them to have hands-on experience with the application. We monitored their responses to certain features of the application and made improvements based on their feedback.

Database Testing

The most important part of the frontend application is the ability to directly interact with and update the information on the backend. Due to this, we did a lot of testing to verify that the correct information was being properly stored in the database when it was updated from the frontend application.

Backend

Out of the box Firebase is fairly feature complete, and there wasn't much testing needed to verify that its services were working. Regardless, we did manually testing through the application to verify that everything was working as expected.

Firebase Authentication

This testing required us to create new accounts on the frontend application and then check that the Authentication section was updated. We also needed to check and make sure that the new user's UID was properly stored in the Realtime DB. This was easy to visualize through the Firebase service

Firebase Realtime Database - Frontend to Backend

For this test we wanted to see if the frontend was correctly hooked up to the backend. In order to do this, we sent data that would create a new tank on the backend with a particular structure, and we checked Firebase's RDB to verify that the exact structure was updated. After that, we only updated a specific value, like tank name, and verified that the correct entry was updated.

Firebase Realtime Database - Firmware to Backend

For the firmware to backend connection we tested that the firmware was able to pass its updated readings to the backend every 15 seconds. We would view what the reading was through the firmware and then verify that the appropriate field (pH or temperature) was updated in the Realtime database.

Firebase Realtime Database - Full round trip

In order to verify that data was flowing correctly from the frontend to the backend we checked to see if things like pH and temperature were getting updated while the app was running and the firmware was collecting results. We then tested to see if when a field was updated from the frontend that the firmware was correctly updated. We did this through manual feeding as there is a boolean that gets updated by the frontend. The firmware then checks this value, and if it is set to true, then the firmware does a manual feed. The firmware then updates that value to false. We were then able to see the Realtime Database update these values as we went through the process and the manual feeding was able to be completed.

Firmware

Data Transfer testing

From the firmware side of things, we tested that our code could both receive and send data to the backend. We tested that our pH and Temperature readings were getting updated in the backend as they should be. Then to test that we are receiving values correctly, we implemented a schedule using the values in the backend to move the motor at certain times. After we tested these and found no problems, we moved on.

WiFi Testing

To test the connectivity of the board to Wifi, we put it in multiple different scenarios: Turning on the board normally with WiFi on, Turning on the board normally without the WiFi then turning the WiFi on after, having the board on and with the WiFi on then turning the WiFi off and back on. After experimenting with all these scenarios, our code was able to handle these different scenarios, being able to reconnect to the WiFi if it was shut off and connect to the WiFi if it wasn't initially on.

Enclosure

The enclosure tests performed were centered around the continued functionality of the dispensing function. These tests had the purpose of ensuring that the device would not jam up during dispensing of the contents of the hopper, as this is critical to this product's functionality.

Food Jamming

The enclosure was left running continuously, stopping and starting at a two-second interval. This interval was set to simulate a real-world short feeding. During operation, a full canister of

Betta fish food was quickly dumped into the hopper of the device and left to operate until the hopper was completely empty of food.

This test was iterated many times with the final enclosure design, and each time the device successfully output 100% of the hopper's contents.

Longevity

The enclosure was left running continuously, stopping and starting at a two-second interval. This interval was set to simulate a real-world short feeding. The device was then left for a 24-hour period to ensure that it would continue to function under strenuous conditions.

The device was able to withstand 24 hours of continuous operation without jamming or an increase in friction in the turning of the Spiral Dispenser.

Related Products

Looking at related products helped give us an idea of how designs that were already being sold, functioned that were similar to our product idea. This enabled us to take away ideas that did not seem to work well or could use an improvement in design.

Petbank Automatic Fish Feeder

This fish feeder, as shown in *Figure i.,* is an automated fish-feeding device that was not IoT enabled. The schedule for the feedings is set right on the screen of the device and you are able to set up to four feedings a day. The pros to this device are that it is battery operated, so it does not require being plugged in, and that it has an adjustable clip for the edge of the tank.

Figure 5.1

What did we learn?

We decided that our device did not need to be battery powered, as this one is, due to all fish tanks needing to be close by to a few power outlets already where the filter and the tank are plugged in. Then the device does not need to be removed and charged.

The adjustable clip is a nice feature to have. This design was created for our product, but was ultimately decided to be unnecessary to implement on this iteration due to our audience having one kind of fish tank.

Appendix I

Operation Manual

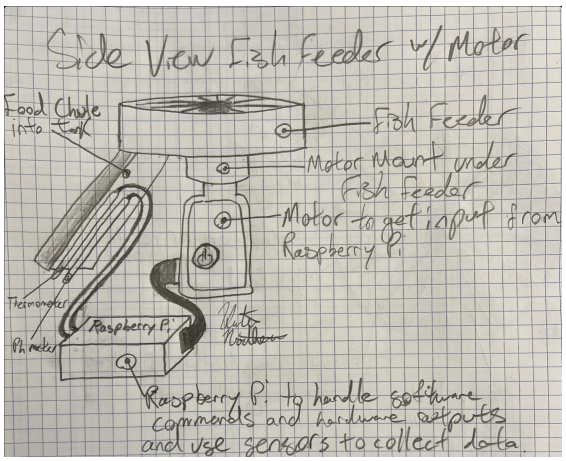
The operation manual is the guide to troubleshooting, assembling, and getting started with the automatic fish feeding system. This guide is located in a separate file *OperationManual_v1.pdf*.

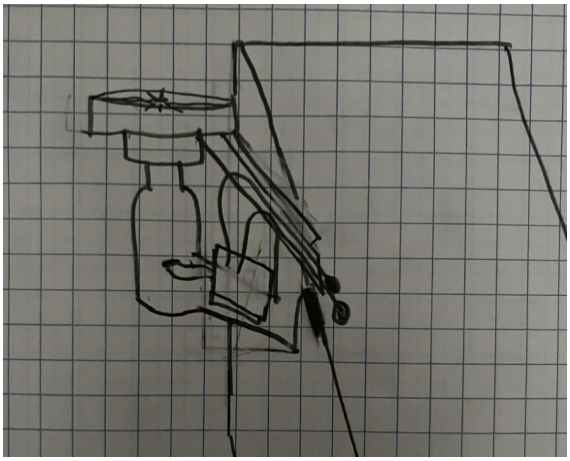
Appendix II

Alternate Designs

Alternate designs were created in case our original design would not function properly or the alternate design functioned better than the original design.

Alternate Enclosure Design

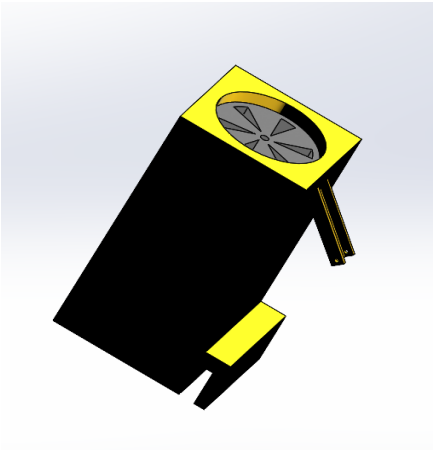




Figures 6.1 and 6.2

This was the initial idea and sketches for our alternate enclosure design. It included a limited number of fins and the motor being very large to hold the dispenser, and a free-standing chute with a very abnormal looking body. This original design did nothing to cover our requirements of being able to feed for a month, was overall not very pleasing to the eye for our clients, and did not protect our electrical components.

In order to combat these concerns we redesigned this enclosure to more fully encompass the inner components and have a more pleasing structure for our clients.




Figures 6.3 and 6.4

The next iteration of this enclosure as shown above, added a regular rectangular prism to hold the components and conceal them instead of the odd shape as shown before. Next, we designed a funnel to mount onto the top of the enclosure to feed more food into the chute to feed the fish up to the standards set by our client.

Alternate Components

For the main CPU on the circuit board we also considered using a Raspberry Pi to house the logic of the device. This would have allowed us more possibilities on the firmware for the functionality of how the device sends/updates data to the backend cloud service.

We did not choose to go this route mainly due to the price of the Raspberry Pi, the large form factor, and the excess computing power necessary for this project. Going with the ESP32 that we decided on, it makes our product more specialized and smaller.

Figure 5.2.