AzulTank - A Fish Tank Helper

Gabriel Besana, Rafael Nieves, Jazz Olario, Christian Rosado Arroyo

University of Central Florida School of Electrical Engineering and Computer Science, Orlando, FL

Abstract - **The AzulTank is a smart aquarium project aimed at revolutionizing fish care by automating critical aspects like feeding, water quality monitoring, and environmental adjustments. By integrating advanced sensors and a user-friendly mobile app, the AzulTank continuously monitors temperature, pH levels, and water turbidity, ensuring an optimal aquatic environment. Automated systems, such as a food dispenser and a pH tablet dispenser, minimize human intervention while maximizing fish health. This paper outlines the development, design, and engineering of AzulTank, emphasizing its practical applications and the innovative solutions employed.**

Index Terms — smart aquarium, automation, water quality, sensors, remote monitoring.

I. INTRODUCTION

Maintaining a healthy environment for pet fish can be challenging for many owners, especially when balancing personal and professional responsibilities. Traditional fish tanks require consistent monitoring and management of parameters like water temperature, pH levels, and feeding schedules. The AzulTank addresses these issues by incorporating advanced technology into a user-friendly system that automates essential tasks, providing a stress-free experience for fish owners.

The AzulTank aims to simplify fishkeeping by integrating sensors, automated dispensers, and a mobile app for real-time monitoring and control. Our system includes a temperature sensor, pH dispenser, and turbidity sensor, all working together to ensure a stable and healthy habitat for aquatic life. Additionally, the project explores innovative features like smart lighting and live video streaming, offering advanced functionality for novice and experienced aquarists alike.

This paper provides a comprehensive overview of the AzulTank's design and implementation. We explore the motivation behind the project, outline the system components, and discuss both the hardware and software concepts that form the foundation of the smart aquarium. Descriptions of the sensors, actuators, and communication protocols are provided, along with the challenges faced during development and the engineering solutions employed. Through this project, we aim to demonstrate how technology can transform a traditional hobby into a more efficient, sustainable, and enjoyable experience for fish owners of all experience levels.

II. SYSTEM COMPONENTS

The AzulTank system is a comprehensive assembly of interconnected hardware and software components, meticulously chosen to automate essential aquarium maintenance tasks while providing real-time monitoring and control. Each component plays a crucial role in ensuring the well-being of aquatic life and enhancing the convenience of fishkeeping for owners.

A. ESP32 Microcontroller (ESP32-WROOM-32E)

The ESP32 microcontroller is the heart of the AzulTank system. It handles all the data processing, sensor readings, and control commands required to automate the aquarium. This powerful, low-energy module features integrated Wi-Fi and Bluetooth capabilities, allowing for seamless wireless communication with the mobile application. Its dual-core processor ensures that multiple tasks, such as monitoring sensors and controlling actuators, are handled efficiently. The microcontroller also manages data storage through a microSD card slot for logging sensor readings, which aids in analyzing long-term trends in water quality.

B. Temperature Sensor (DS18B20)

The DS18B20 is a digital temperature sensor designed to provide highly accurate and consistent readings in aquatic environments. Fully waterproof and submersible, this sensor is ideal for continuous monitoring of the water temperature. Maintaining a stable temperature is critical for fish health, as even small fluctuations can lead to stress or disease. The DS18B20 can measure temperatures ranging from -55°C to +125°C with a high level of precision, ensuring the aquarium environment remains within a suitable range for the specific species of fish being kept.

C. Gravity: Industrial pH Sensor Pro

The pH sensor monitors the acidity or alkalinity of the aquarium water, an essential parameter for maintaining a healthy environment for the fish. The sensor provides real-time data to the ESP32 microcontroller, which analyzes the readings and determines whether any corrective action is needed. Accurate pH monitoring helps prevent stress and health issues in aquatic life, as even slight variations can have significant effects on fish and plants.

D. Gravity: Turbidity Sensor

The turbidity sensor monitors the clarity of the aquarium water by measuring the amount of suspended particles. High turbidity levels indicate poor water quality, which can lead to health issues for fish and affect the overall ecosystem. This sensor sends real-time data to the microcontroller, which can trigger alerts or actions, such as activating a water change notification or adjusting the lighting system to signal an issue. By proactively detecting water quality problems, the AzulTank ensures that necessary measures are taken to maintain a clean and safe environment.

E. Automated Stepper Motor-Driven pH Tablet Dispenser

The automated pH tablet dispenser, driven by a stepper motor, is responsible for making precise pH adjustments when needed. When the pH sensor detects a deviation from the desired range, the microcontroller activates the dispenser to release a pH-balancing tablet into the water. This automated system ensures the pH level remains stable without requiring manual intervention, contributing to a healthier aquatic ecosystem.

F. Automated Servo Motor-Driven Food Dispenser

The automated food dispenser utilizes a servo motor to dispense measured portions of fish food at scheduled intervals. Precise feeding is crucial, as overfeeding can pollute the water and underfeeding can harm the fish. The dispenser is designed to be adjustable, allowing users to set specific feeding times and quantities through the mobile app. The servo motor ensures accurate and repeatable movements, delivering consistent amounts of food to the fish and minimizing the risk of food waste.

G. NeoPixel RGB LED Strip

The NeoPixel Digital RGB LED strip provides dynamic and customizable lighting for the aquarium. It simulates natural light cycles, which can improve the well-being of the fish and enhance the aesthetics of the tank. Additionally, the LEDs can be programmed to change color based on water quality metrics, serving as a visual cue for the owner. For example, green light indicates optimal conditions, while red or yellow light alerts the owner to potential issues. The lighting system is energy-efficient and fully controllable via the mobile app.

H. Custom PCB

The custom PCB in the AzulTank system serves as the central hub for all hardware integration, meticulously designed to streamline connections and ensure efficient operation. This includes several crucial components and sub-boards to manage the various functionalities required for the smart aquarium. Our custom PCBs feature a voltage regulator, which provides stable and consistent power to all connected elements, preventing voltage drops or fluctuations that could disrupt the system.

Embedded within the PCB is the microcontroller unit (MCU) board, which houses the ESP32 module. This microcontroller acts as the brain of the AzulTank, processing data from the sensors, executing control commands for the actuators, and facilitating wireless communication with the mobile application. The PCB also includes a dedicated board for the turbidity sensor, ensuring that measurements of water clarity are accurately transmitted and processed. Signal routing and power connections for the turbidity sensor are optimized to minimize interference and maintain reliability.

Additionally, the custom PCB incorporates a specialized area for the pH sensor. This section is designed with proper signal conditioning to ensure the precision and stability of pH readings, which are critical for the health of the aquatic environment. The integration of the pH sensor board with the automated pH tablet dispenser allows for real-time adjustments to the water's acidity levels, maintaining optimal conditions for the fish. Overall, the custom PCB not only reduces the complexity of the wiring but also enhances the system’s robustness and maintainability.

I. Power Supply Unit

The AzulTank is powered by a stable DC power supply that is connected to a wall outlet. The power unit is designed to deliver sufficient energy to run all components, including the microcontroller, sensors, actuators, and LED strip, while keeping overall power consumption below 50 Watts. Voltage regulation and surge protection are included to ensure safe and efficient operation.

J. Mobile Application

The mobile app serves as the user interface for the AzulTank, providing an intuitive and modern design for monitoring and controlling the aquarium. Users can view real-time sensor data, adjust settings, and receive notifications about the tank's condition. The app also allows users to schedule feeding times, set pH adjustment parameters, and customize the LED lighting settings providing a comprehensive and remote aquarium management experience.

III. SYSTEM CONCEPT

AzulTank's operation can be understood in a cynical framework, with three important stages: Monitoring, Decision-Making, and Execution.

In the *Monitoring* stage, this involves using the system's smart sensor to continuously gather data on important water parameters such as the temperature, pH level and turbidity. These readings reflect the current state and create an up-to-date profile of the water quality and overall environment of the aquarium.

In the *Decision-Making* stage, the system compares the collected sensor data to the user-defined thresholds and pre-set configurations. This phase assesses and determines if the current water conditions are optimal or if corrective actions are required. For example, if the water temperature and pH level drifts outside the acceptable range or if the turbidity levels signal an issue with the water clarity, the system determines an appropriate response.

Once Decision is made, the system enters the *Execution* phase. Here, AzulTank activates the relevant devices to carry out the necessary corrective or scheduled actions as well as sends notification to the user through the app to keep them informed of the adjustments made by the system or actions that may need to be taken by the user. Overall, ensures that the aquarium remains stable and that any deviation from optimal conditions is handled immediately.

A. System Hardware Concept

With more understanding of the system concept the hardware concept for AzulTank can now be detailed. This revolves around the integration of hardware components to allow for a full automation smart aquarium.

The AzulTank's hardware system can be organized into three main categories: (1) environment monitoring with the sensors, which track parameters such as temperature, pH, and water quality, (2) data processing and component management through the microcontroller unit (MCU) board, which interprets sensor inputs and coordinates motor controller system responses, and (3) actuators for automated functions like feeding, pH adjustment, and lighting to maintain optimal conditions. These components work together to enable real-time monitoring and response, ensuring a stable and controlled environment. An overview of this system is presented in Figure 1 of the hardware block diagram and is separated by color sections.

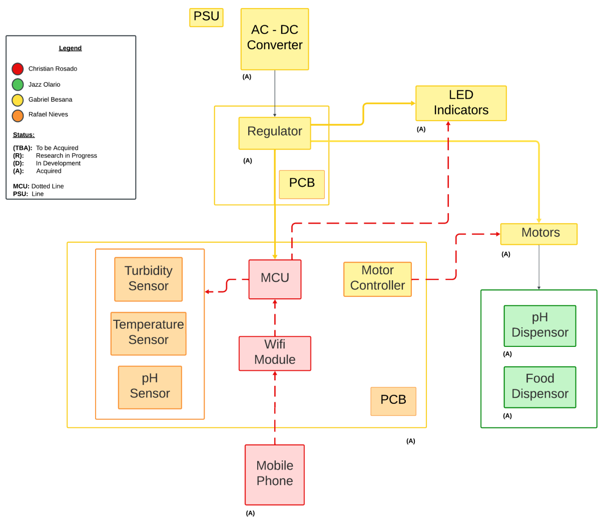


Figure 1: AzulTank Hardware block diagram of the major system components

B. System Software Concept

The mobile application is an essential component of the AzulTank's system which enables seamless user interaction and control over the aquarium's operations. This serves as the primary interface of receiving data, and sending commands which keeps the users involved with their aquarium's status.

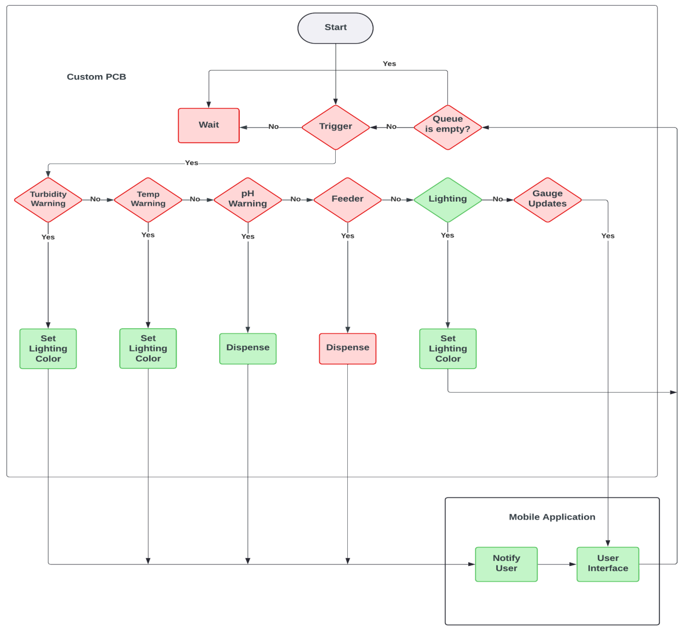


Figure 2: AzulTank Software diagram

Core Functions:

(1) *Real-Time Data Display and User Interface* Displays the monitoring systems which includes the temperature, pH levels and turbidity.
(2) *Controls and Custom Commands:* Allows the user to send requests to trigger any specific actions within the system. These commands are processed by the microcontroller to execute the necessary operation.
(3) *Dispenser System:* Dispenses can be scheduled or manually triggered, ensuring that aquatic life is cared for even when users are not physically present.
(4) *Customizable Threshold Setting*s*:* Allows the user to set and adjust thresholds for temperature, pH level and turbidity to ensure that the aquarium's environment can be tailored to meet the needs of different types of aquatic life.
(5) *Notification system:* Identifies any change in the tank's environment that requires attention. The mobile app sends the notifications to ensure that the user is alerted to the potential issues and allows them to take any action if needed.

IV. HARDWARE DETAIL

For the system hardware concept, the technical features of our major hardware components of the system are explained more in depth in this section . Due to mentions throughout the paper, the microcontroller won't be included in this section. This will in turn detail the surrounding hardware components found within the Figure 1 hardware block diagram.

A. Power Supply Unit

(1) To power our system, we have chosen a switching regulator to step down the wall outlet power from 120V to 12V, ensuring efficient and reliable operation for our components. For this purpose, we selected the TPS54528DDAR, a 5-A synchronous buck converter designed for high-performance applications. This regulator operates within an input voltage range of 4.5V to 18V, making it highly versatile and capable of handling a wide variety of power sources. Its output is optimized for precision and stability, making it an excellent fit for our system's energy-sensitive requirements. The TPS54528DDAR also boasts a high efficiency across various load conditions, thanks to its advanced design. It features an adjustable switching frequency of up to 2 MHz, enabling the use of smaller external components. This is particularly beneficial for projects like ours, where compact layouts are critical due to space constraints. Additionally, the regulator maintains excellent thermal performance, ensuring that it can operate in demanding environments without overheating or degrading performance over time.

B. Sensors

(1) The temperature sensor that we have decided to use on the AzulTank system is a DS18B20 waterproof digital sensor, chosen for its high accuracy and suitability for submersion in aquatic environments. With a precision of ±0.5°C, it provides real-time data crucial for maintaining stable water temperature.
This continuous monitoring is especially beneficial for aquatic health, as temperature fluctuations can significantly impact the well-being of fish and other tank inhabitants. The sensor is directly connected to the ESP32 microcontroller, where temperature readings are processed and sent to our mobile application prompting the user to take corrective action if the water falls out of the desired range. This is to ensure the tank's temperature remains within a safe range.
(2) The second sensor of our system is our pH sensor. We have chosen the Gravity: Analog pH Sensor v2, and it monitors the water's acidity or alkalinity, a key parameter for maintaining a healthy ecosystem within the aquarium. Accurate to ±0.1 pH, this sensor includes built-in temperature compensation, which allows it to deliver reliable readings despite minor water temperature changes.
The pH sensor provides analog output to the custom PCB, where the data is transmitted to the ESP32 controller. By continuously assessing pH levels, this can allow our system to send notifications to our mobile application to trigger pH-balancing mechanisms if needed, maintaining optimal conditions for aquatic life and reducing the need for frequent manual testing.
(3) Finally the last sensor of our system is the AzulTank's turbidity sensor, the Gravity: Analog Turbidity Sensor. This sensor measures water clarity with a range from 0 to 1000 NTU (Nephelometric Turbidity Units) and an accuracy of ±5 NTU. High sensitivity enables it to detect even slight changes in water cloudiness due to particles such as algae, waste, or other debris.
These real-time measurements are essential for tank maintenance, as increasing turbidity often signals deteriorating water quality. The turbidity sensor's data is transmitted via the PCB to the ESP32, where it can trigger LED indicators and mobile app notifications to alert users. We chose this specific model due to its high sensitivity to allow for more of a proactive monitoring that allows users to address water clarity issues promptly, supporting a healthy and balanced aquarium environment.

C. *Actuators*

(1) The motor we have chosen that was utilized for our pH dispenser was the MIKROE ROHS 28BYJ-48 Stepper Motor. This was essential to the AzulTank's automated pH balancing system and was selected for its precision and reliable performance. The 28BYJ-48 motor fits the exacting requirements of pH dispensing in the aquarium environment. Featuring a unipolar design with five or six leads, it operates at a 5V nominal DC input. This model has a step angle of 5.625°/64 in half-step mode and provides a resolution of 4096 steps per resolution ensuring fine-grained positioning accuracy. With a holding torque 0.34 kg·cm it enables fine control over the pH dispenser's tablet dispensing process, and helps to facilitate accurate adjustments to the tank's pH levels. This high level of control enables the motor to deliver incremental rotations, which are critical for precise dosing of the pH tablets. Coupled with the custom PCB, this stepper motor supports reliable, automated pH regulation, and prevents any human error when manually trying to fix the water's pH levels.

(2) The servo motor utilized in the AzulTank system that we chose was the TowerPro SG90, a small, lightweight motor known for its precision and reliability in applications requiring controlled movement. This motor is integral to the automated feeding mechanism, where precise adjustments are essential for maintaining appropriate feeding conditions within the aquarium. Operating at a voltage range of 4.8V to 7.2V, the SG90 provides a maximum torque of 1.8 kg-cm and can rotate within a 180° range, allowing for controlled, incremental movements. With a typical speed of 0.12 seconds per 60°, it is fast enough to dispense fish food without causing disturbances in the tank. The motor's potentiometer-based feedback system enables the ESP32 microcontroller to achieve precise positioning, crucial for exact dosing of the fish food. Connected to the custom PCB, the SG90 motor integrates seamlessly with other components, allowing the AzulTank to automate feeding schedules accurately, minimizing manual intervention and maintaining optimal feeding conditions for aquatic life.

V. SOFTWARE DETAIL

One of the key successes of this project is the ability to integrate data from various sensors, analyze it in real-time and perform actions when it is necessary based on user-defined parameters.

One thing to highlight of the system's software design is its flexibility in managing multiple operational scenarios, from automatic responses to user-triggered actions. This allows the system to adapt and adjust its behavior to maintain water quality and control feeding schedules. The user application is also important for interaction, notification and custom configurations.

A. System Firmware

The system firmware of this project serves as the operational core of the project which is running on the ESP32 microcontroller. It is implemented in C++ and organized into several modules which resulted in 1,397 lines of code.
The microcontroller itself is responsible for coordinating data collection, processing, and device management in order to maintain optimal conditions for the aquarium. Overall, the system firmware is the backbone of AzulTank that enables the automation tasks and ensures that the system can operate autonomously and manually while keeping the user informed and in control using the mobile application.

```
azul.ino
setup.ino
    ➔ connectWiFI() ;
    ➔ io.connect();
    ➔ mqtt.subscribe();
loop.ino
    ➔ MQTT_connect();
    ➔ while ((subscription =
      mqtt.readSubscription(100))) {
          if(colorFeed) {
            handleColorData(controlData);
            handleOnOff();
          }else if(scheduleFeed){
            handleScheduleData(scheduleData);
            printAllSchedules();
          }else if(servoFeed) {
           handleServoMovement();
          }else if(stepperFeed){
            handleActivateStepper();
          }elseif (tempSensor.requestTemp();) {
          }else if(analogRead(pHSensorPin);{
      }else if(analogRead(turbiditySensorPin);
     }
    ➔ checkSensorValues();
    ➔ publishSensorData();
```

Figure 3: .ino files and function calls used to control all the hardware used in this project.

(1) *azul.ino*: is responsible for initializing the global variables, libraries and data structures used for this project. It sets up Wi-Fi and MQTT communication to connect with Adafruit IO for remote monitoring and control. The code also configures and connects various data pins to hardware components where it assigns the

appropriate GPIO pins for temperature, pH and turbidity sensors as well as the servo and stepper motor.

(2) *setup.ino*: performs a series of initialization tasks to ensure that the system is ready to run smoothly. This includes, setting up the Wi-Fi, connecting to Adafruit IO and setting up MQTT subscriptions to manage data communication with remote servers. Additionally, it initializes hardware components used for this project.

(3) *loop.ino:* is responsible for managing continuous operation of the by project by coordinating tasks and calling functions from other *.ino* files if needed. This includes keeping the MCU connected to the backend, Adafruit IO, managing schedules, activating the servo motor and reading user-defined sensor limits. The loop contains the following calls:

(4) MQTT_connect(); Ensures a connection to the Adafruit IO MQTT server. If disconnected, it attempts reconnection.

(5) mqtt.readSubscription(100) {} is a loop that listens for incoming MQTT messages. It handles various backend feeds, used to trigger specific application actions.

(6) handleScheduleData() updates schedule by adding or deleting schedules based on what it is received from the App.

(7) tempSensor.requestTemperatures(); reads the temperature sensor's value and stores it in a variable.

(8) analogRead(pHSensorPin); reads the analog value from the pH sensor and stores it in a variable.

(9) analogRead(turbiditySensorPin); reads the analog value from the turbidity sensor and stores it in a variable.

(10) checkSensorValues(); checks the current sensor readings against the user-defined threshold values such as temperatureMin, temperatureMax, etc.

(11) publishSensorData(); reads the variables that hold the sensor values and publishes them to the AdaFruit IO backend service in a JSON string format. Its purpose is to be retrieved by the React software and displayed on the UI.

(12) handleServoMovement(); manages the timed movements of the servo based on its current state. Its purpose is to return the servo motor to its home position.

B. Mobile Application Software

The mobile application software consists of 2,497 lines of code and is developed in JavaScript using React Native. The codebase is organized into multiple `.jsx` files, each dedicated to specific features and UI components, with functions to manage various aspects of the application. This modular structure allows for a clean separation of logic and design, where each file focuses on distinct UI screens, component styling, or sensor interactions, enhancing the app's responsiveness and user experience.

The main functionality is organized within index.jsx. This file contains the primary function, App(), which serves as the entry point for the mobile application. The function initializes the core components and structure of the app, managing the rendering of UI elements such as gauges to display sensor data and buttons to control various features. The index.jsx file contains the following function calls:

(1) The startLongPolling() function manages ongoing data retrieval, calling fetchSensorData() every 10 seconds. Its purpose is to reduce server load while providing updates only when requested.

(2) The fetchSensorData() function is responsible for retrieving real-time sensor data from the backend's endpoint and updating the state variables for temperature, pH, and turbidity used to update the UI gauges.

(3) The handleActivateServo() function sends a command to the backend, which is then processed by the system's firmware to activate the servo motor and dispense the fish food a user-specified number of times.

(4) handleActivateStepper() function sends a command to the backend, which is then processed by the system's firmware and dispenses a pH tablet.

The functionality of the remaining buttons, such as those for history, settings, schedules, and LED settings, is managed by other .jsx files.

(5) The toggleLED function handles LED state changes. It sends a command to ESP to toggle the LED between "ON" and "OFF."

```
 startLongPolling();
     ➔  fetchSensorData();
 handleActiveServo();
 handleActivateStepper();
toggleLED();
```

Figure 4: Function calls under index.jsx

The history.jsx file contains the react component that displays sensor and dispenser data fetched from Adafruit IO. The following function calls occur within it:

```
fetchHistoryData();
```

Figure 5: Function call under history.jsx

(1) fetchHistoryData(); is an asynchronous function and fetches the history data from the backend service in the form of JSON string.

The settings.jsx file also serves as a routing point to sensor settings, dispenser settings and LED settings files. The dispenserSettings.jsx file contains the component that manages the settings for the food dispenser allowing users to configure both scheduled and manual dispensing values. The following function calls occur:

```
loadValues();
saveValues();
sendToAdafruitIO();
```

Figure 6: Function calls under dispenserSettings.jsx

(1) loadValues();is the asynchronous method responsible for retrieving the current values of ‘scheduledValue’ and ‘manualValue’ variables from AsyncStorage.
(2) saveValues(); function saves the current ‘scheduleValue’ and ‘manualValue’ variable values to AsyncStorage and calls the function sendtoAdafruitIO();.
(3) sendToAdafruitIO(); is the function that sends the updated ‘scheduledValue’ and ‘manualValue’ variable contents to the AdaFruitIO service to then be retrieved and processed by the system firmware.
The sensor settings contain the react component that manages the interface for setting sensor limits, including temperature, pH, and turbidity. It uses useState hooks to store the minimum and maximum values for each sensor parameter.

```
loadValues();
saveValues();
publishToAdafruitIO();
```

Figure 7: Function calls under sensorSettings.jsx

(1) loadValues (); is an asynchronous function that fetches sensor limit values for temperature, turbidity, and pH from AsyncStorage used to send out notifications when values go above or below a set threshold.
(2) saveValues (); is a function triggered by the user after tapping a Save button. It saves the current sensor limit values to AsyncStorage and then calls publishToAdafruitIO() to send the values to the backend.
(3) publishToAdafruitIO (feedKey, data); is the function responsible for publishing the sensor limits. It converts the read data to JSON format and posts it on the backend service to be retrieved and executed by the system firmware.
The ledSettings.jsx is created for controlling the LED color and brightness through the mobile app. It uses Adafruit IO for storing and fetching LED control settings and communicates with the MCU via MQTT to send updates.

```
hexToRgb();
sendColorAndBrightnessToESP32();
```

Figure 8: Function calls under ledSetting.jsx

(1) hexToRgb(); converts a HEX color value to its RGB equivalent.
(2) sendColorAndBrightnessToESP32(); sends the color and brightness to MCU via MQTT.

```
fetchSchedules();
addSchedule();
sendScheduleToAdafruitIO();
```

Figure 9: Function calls under schedulePage.jsx

(1) fetchSchedules(); fetches all schedules from Adafruit IO.
(2) addSchedule(); Adds a new schedule if one is not already being added. It also checks for duplicates before sending the schedule to the MCU.
(3) sendScheduleToAdafruitIO(); send in the new or updated schedule to the MCU.
addSchedule.jsx & editSchedule.jsx is created to either create or edit schedules for the devices such as LEDs and feeders. This allows them to select a start time, days and choose a device.

```
onChange();
formatSelectedDays();
deleteSchedule(); → Only when editing schedule
```

Figure 10: Function calls under addSchedule.jsx & editSchedule.jsx

(1) onChange(); this allows the user to select the desired time.
(2) formatSelectedDays(); allows the user to select the repeat days of the schedule. If no days are selected, it simply returns “None.”
(3) deleteSchedule(); this is only when a schedule is already created and the user simply just wants to delete a single schedule.

VI. BOARD DESIGN

With the creation of our boards for the AzulTank it presented some difficulties due to the complexity of balancing delicate devices all on one board. Due to needing a more compact design, it was important to design a balanced layout for the signal integrity, minimized interference of components and ensuring the proper and efficient power distribution from our voltage regulator to all our major components. Our main challenge with our board design was integrating both the pH and turbidity sensor as we had many errors with the layout of those two sensors on our first set of boards.. Having to adapt an older version of the pH board schematic and similarly trying to replicate on our custom PCB the functions of the boards that came with our sensors was a bit complex. It was very important for us to keep the precision and stability of both sensors when paired with all of our hardware components as they took a different voltage signal and there were a lot of adjustments for each individual layout before setting it up all on one main board. Finally, we decided on a separate voltage regulator PCB for a safety measure to prevent our more sensitive hardware components like the sensors from getting damaged if problems arise.

VII. CONCLUSION

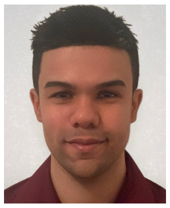
AzulTank enhances traditional aquariums with smart technology, ensuring the well-being of the fish. The selected sensors offer real-time analytics and alerts, and a user-friendly app design enables remote monitoring and control of aquarium features. Overall, AzulTank combines automation and convenience, providing a practical, cost-effective solution that supports aquatic care for all experience levels.

VIII. THE ENGINEERS

Gabriel Besana is a 24 year-old graduating Electrical Engineering student with career goals to pursue a path in power and renewable energy. Aspiration is to be part of a company in the push for clean and efficient energy.

Rafael Nieves is a 26 year-old graduating Computer Engineering student with career goals to pursue a path in firmware or software development aspiring to work in a large, innovative tech company.

Jazz Faye Olario is a 27 year-old graduating Computer Engineering student. Jazz' career goals are to pursue a path in firmware development within a large innovative tech environment.



Christian Rosado Arroyo, is a 29-year-old graduating Computer Engineering student, aiming for a career in embedded systems development with the aspiration of becoming an Imagineer at Walt Disney World.

IX. ACKNOWLEDGEMENT

We would like to acknowledge our advisors, Dr. Arthur Weeks, Dr. Chung Yong Chan, and Dr. Lei Wei for their continued guidance and assistance throughout this project. We also appreciate the support from the University of Central Florida for the resources necessary for our research.

X. REFERENCES

[1] Nowak, Maja. "Flutter vs. React Native in 2022." *Nomtek*, 1 Feb. 2024, www.nomtek.com/blog/flutter-vs-react-native.

[2]"Gravity__analog_ph_sensor_meter_kit_v2_sku_sen0161-V2." DFRobot, wiki.dfrobot.com/Gravity__Analog_pH_Sensor_Meter_Kit_V2_SKU_SEN0161-V2.

[3] "Turbidity_sensor_sku__SEN0189." DFRobot, wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189

[4]"Waterproof_ds18b20_digital_temperature_sensor__sku_dfr0198_." DFRobot, wiki.dfrobot.com/Waterproof_DS18B20_Digital_Temperature_Sensor__SKU_DFR0198_.

[5] Imperial College London. "*Servo Motor SG90 Data Sheet*", www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf. Accessed 8 July 2024.

[6] *Step motor 5V - mikroelektronika*. MIKROE. (n.d.). https://www.mikroe.com/step-motor-5v

[7]*TPS54528*. TPS54528 data sheet, product information and support | TI.com. (n.d.). https://www.ti.com/product/TPS54528?utm_source=google&utm_medium=cpc&utm_campaign=app-bsr-null-44700045336317437_prodfolderdynamic-cpc-pf-google-ww_en_int&utm_content=prodfolddynamic&ds_k=DYNAMIC%2BSEARCH%2BADS&DCM=yes&gad_source=1&gclid=Cj0KCQiAi_G5BhDXARIsAN5SX7qmvwglphHr-NJ9uWPqGcz_Uks8Si_Zd4ZSPrNUd9dw7dlFEmhKUEwaAlK3EALw_wcB&gclsrc=aw.ds

[8] Geeksforgeeks. "Microcontroller and Its Types." *GeeksforGeeks*, 20 Feb. 2022, www.geeksforgeeks.org/microcontroller-and-its-types/. Accessed 27 June 2024.