ECE 445 Senior Design

AquaSense: Affordable ML-Based Water Quality Monitoring for Aquariums

Presented By

Anurag Ray Chowdhury
(anuragr3@illinois.edu)
Arnav Garg
(arnavg8@illinois.edu)
Michael Yan
(myan13@illinois.edu)

TA: Michael Molter
Professor: Arne Fliflet

May 7th, 2025
Team 72: Aquasense (93)

Abstract

AquaSense is an affordable water quality monitoring system aimed at aquarium owners. It continuously measures three key parameters quantifying aquarium health: pH, temperature, and luminescence via an ESP32 microcontroller. The system wirelessly transmits parsed real-time data into a cloud-powered mobile dashboard feed, empowering users to monitor trends in water quality and receive alerts when conditions fall outside preferred ranges specific to their aquarium. Additionally, Machine Learning algorithms can analyze historical user-specific data to detect anomalies and predict trends in water quality issues. With this emphasis on affordability, accuracy, intelligence, and consumer-facing design, AquaSense provides a unique and accessible solution for maintaining a healthy aquatic environment with minimal hassle for the owner. With this emphasis on affordability, accuracy, intelligence, and consumer-facing design, AquaSense provides a unique and accessible solution for maintaining a healthy aquatic environment with minimal hassle for the owner.

Table of Contents

1 Introduction

1.1 Problem

As an aquarium owner, you may have experienced the anxiety of leaving your fish unattended for an extended period—whether for work, travel, or vacation—without knowing if water conditions remain safe. Imperfect aquarium conditions can quickly deteriorate, leading to illness or even fatal consequences for fish and aquatic plants. Maintaining optimal water quality is therefore crucial for the health of aquatic organisms; poor water quality is a leading cause of illness in fish, and maintaining proper water parameters is essential for their well-being [1]. However, many owners lack access to affordable, real-time water monitoring solutions. Standard testing involves manual kits that require frequent intervention, making it difficult to track aquarium health and can be fatal [2].

Existing automated water quality monitoring solutions are often expensive and designed for industrial-scale applications, leaving home aquarium owners with few accessible options. For instance, advanced monitoring systems are tailored for large-scale aquaculture operations and may not be cost-effective for individual hobbyists [3]. To bridge this gap, there is a need for a low-cost, plug-and-play solution that continuously monitors water conditions and provides real-time alerts when the water quality becomes unsuitable for fish. Such a system would enable aquarium enthusiasts to maintain healthier environments for their aquatic life.

One example of an affordable monitoring device is the Kactoily Smart 7-in-1 Aquarium Monitor, which offers real-time water quality tracking with built-in Wi-Fi, allowing users to oversee parameters such as pH and temperature continuously [4]. Implementing accessible monitoring solutions can significantly enhance the ability of hobbyists to maintain optimal water conditions, thereby promoting the health and longevity of their aquatic organisms.

1.2 Solution

We propose AquaSense, a cost-effective, ESP32-based plug-and-play PCB designed to provide real-time water quality monitoring for aquarium owners. The system integrates multiple sensors to continuously track key parameters, including pH, temperature, and luminescence, ensuring a healthier environment. By leveraging the ESP32 microcontroller, AquaSense offers Wi-Fi and Bluetooth connectivity, allowing users to remotely monitor water conditions via a web-based dashboard with real-time data logging, trend analysis, and historical insights. Our solution is structured into two primary components: the sensing system and the monitoring smart interface.

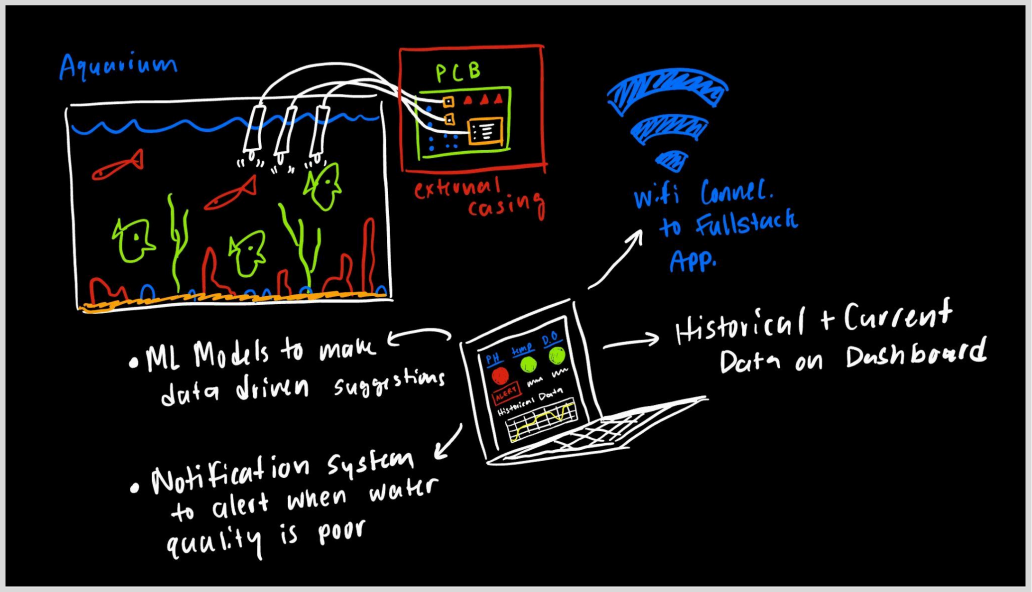
- The sensing system consists of precision pH, temperature, and luminescence sensors that are placed within the aquarium for real-time data collection. These sensors transmit their readings to the ESP32 microcontroller, which processes the data and relays it to the user interface.

- The monitoring interface provides users with a web dashboard, built with React, that offers a comprehensive view of the water conditions. Users can track real-time water quality, view historical trends, and receive detailed insights into the overall health of their aquarium.

AquaSense features an automated alert system that notifies users whenever water parameters deviate from predefined safe ranges. This system ensures timely intervention to prevent harmful fluctuations that could threaten aquatic life. Users can customize threshold levels based on their specific aquarium needs, making the system adaptable to various setups, from freshwater to saltwater tanks.

AquaSense is built to be easy to use, with a plug-and-play setup that works right out of the box. It uses the ESP32's Bluetooth Low Energy (BLE) to connect locally, so users can monitor their tank even without Wi-Fi. For example, if AquaSense detects a drop in pH from 7.5 to 6.1, it immediately sends an alert and provides recommended actions to help stabilize the tank before conditions get worse. If it notices a drop in light levels, it might suggest increasing water flow or aeration. These suggestions are based on live sensor data, helping users take quick, informed action to protect their tank. By combining real-time tracking, remote access, and smart alerts, AquaSense gives both new and experienced aquarium keepers a reliable way to keep their water safe and stable for aquatic life.

1.3 Visual Aid



Aquarium
PCB
external casing
wifi Connect. to Fullstack App.
Historical + Current Data on Dashboard
PH
temp
DO
ALERT
Historical Data
• ML Models to make data driven suggestions
• Notification System to alert when water quality is poor

Figure 1: High Level Overview (sketch)

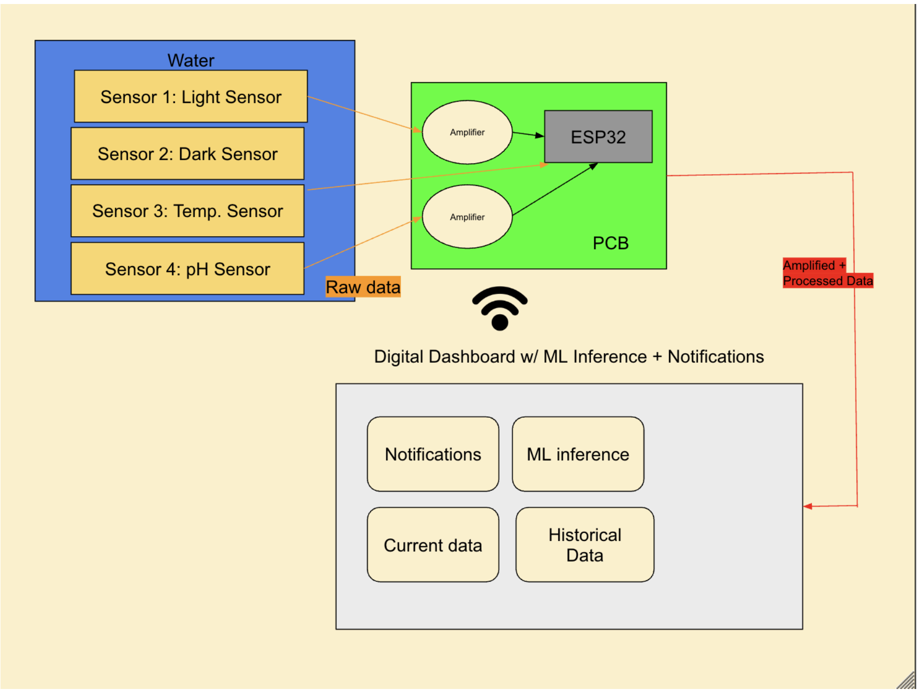


Figure 2: High Level Overview Expanded

1.4 Performance Requirements

For our project to be considered a success, we've defined these following High Level Requirements for the system to satisfy:

1.4.1 Continuous Monitoring of Critical Parameters

To maintain a healthy aquatic environment, AquaSense needs to perform real-time, high resolution monitoring of the water's pH, temperature, and surrounding light (both in dark and bright settings)—parameters that all critically influence fish metabolism, immune response, and overall well-being. The ESP32 based hardware system should expose sensors that capture and log readings at 5 to 10 second intervals to detect transient perturbations that could otherwise be missed by manual or infrequent testing methods. The totality of information conveyed by these sensors should deliver an ample number of parameters to allow higher-level algorithms to perform proper analysis of aquarium conditions.

1.4.2 Wireless Data Transmission & Real-Time Alerts

AquaSense should be able to communicate data and analysis to the user via Wi-Fi and Bluetooth through a secure, cloud based dashboard in real-time. This full-stack system must be able to deliver the data gathered by the hardware solution in live updates at 10-second intervals, ensuring that the end user is able to access the data without ever having to touch the hardware. This ensures all critical changes in aquatic conditions (e.g. sudden temp drops, pH spikes) are shared immediately.

1.4.3 Automated Trend Analysis & Machine Learning Anomaly Detection

To formulate aforementioned alert suggestions, AquaSense will leverage machine learning to detect patterns (anomaly detection) and proactively alert users to potential water quality risks. The model will be trained on the user's own historical data of sensor readings over time and combined with Natural Language Processing via fine-tuned LLM to interpret the results and craft semantic responses with context-awareness and aquatic expertise. The responses must come through the web-facing notification interface and are suggestion-based (ex. *"pH levels low… tank level too low … consider performing a partial (25%) water change."*).

2 Design Procedure

Our system consists of six major blocks: sensor modules (pH, temperature, light), system monitor, backend/data storage, and an LLM-based recommendation system. Each block was designed with key trade-offs in mind, balancing cost, accuracy, scalability, and ease of integration with the ESP32 microcontroller.

We selected an analog pH sensor to measure the acidity of the water at regular intervals. Alternatives included industrial-grade digital sensors with built-in calibration and temperature compensation, but these were cost-prohibitive (often over $80). Our chosen sensor, combined with a TL431-based voltage reference and op-amp amplification circuit, provided stable readings and allowed for custom calibration. This approach gave us flexibility while staying within budget. The output of our probe gave us a millivolt signal that varies with hydrogen ion concentration. To interpret this signal, we applied the Nernst equation:

$$E = E^{0} - \frac{0.0591}{n} log[H^{+}]$$

This equation relates pH to electrode potential, enabling us to correlate the voltage signal from the probe to actual pH values. Since the sensor output was typically between –400 mV and +400 mV, and the ESP32's ADC reads only positive voltages (0–3.3V), we implemented hardware amplification and level shifting. This involved a TL431 voltage reference, an op-amp amplifier, and a voltage divider to shift the signal into the readable ADC range. We also used series and parallel resistance calculations to tune the gain and biasing network; Series resistance was used to limit current flow and protect the sensor. Parallel resistors helped define the gain of the op-amp circuit.

$$V_{out} = V_{in} * \frac{R_{thermistor}}{R_{thermistor} + R_{fixed}}$$

The glass‑electrode pH sensor's sensitivity (≈ 59 mV per pH unit at 25 °C) actually shifts with temperature—by about 0.198 mV/pH for every 1 °C change — so we incorporate the DS18B20 temperature reading into our conversion to preserve our ±0.1 pH accuracy across typical aquarium temperatures (20–30 °C). Specifically, after reading the raw pH voltage, the firmware applies the temperature‑dependent Nernst slope in the equation below to correct for any deviation from 25 °C:

$$\mathrm{pH} = \mathrm{pH}_{\mathrm{raw}} + (T - 25) \times \frac{0.05916 \times T/298.15}{1\,^{\circ}\mathrm{C}}$$

This output was then passed through a low-gain op-amp circuit to stabilize the signal and improve sensitivity near critical temperature ranges (e.g., 20–30 °C). We tuned the gain using series/parallel resistor combinations to keep the output within the ADC's linear range and to reduce noise.

Initially, we planned to include a dissolved oxygen (DO) sensor to better support aquatic life monitoring. However, due to its high cost (often exceeding $100), we replaced it with a light sensor. While it doesn't directly measure water quality, light exposure plays an important indirect role in aquatic and plant environments. The selected sensor was low-cost, responsive, and allowed categorization of lighting conditions into zones (e.g., Pitch Black, Low Light, Bright Light, Excessive Light) useful for triggering alerts or adjustments in the system. Having two light sensors allows the system to detect spatial differences in lighting and enables more accurate monitoring and control of lighting conditions for both aquatic life and plant growth, which may have different light requirements.

The system monitor was built to show live sensor data with low latency and to provide historical visualizations. We evaluated frameworks like Node-RED and custom web dashboards. A custom JavaScript dashboard with backend API syncing was chosen for flexibility and tighter control over UI responsiveness. It enables updates every 10 seconds and supported real-time charting with anomaly flags.

To store sensor data, we evaluated Firebase, AWS DynamoDB, and Google Cloud SQL. We selected the Flask backend due to its ability to utilize data frames as storage, integration with our cloud environment, and scalability. The system was stress-tested to ensure support for over 1,000 readings per parameter over a 30-day period. It also allowed easy querying for the machine learning and LLM subsystems.

For user feedback, we developed an LLM-based module to interpret anomalies and translate them into natural-language alerts. We explored fine-tuning a small model but opted for a hybrid approach using rule-based phrasing triggered by ML predictions. Anomaly detection was powered by thresholding relative to the aquarium we were working with, trained on simulated fault scenarios (e.g., heater stuck on, sudden pH drop). This setup provided over 90% detection accuracy while maintaining interpretability.

3 Design Details

3.1 Sensor Subsystem (Hardware)

3.1.1 Overview

The Sensor Subsystem acquires three primary water quality parameters—pH, temperature, and luminescence — that are essential for monitoring aquarium health. It uses a SEN0161 pH Sensor to measure acidity or alkalinity with ±0.2 pH unit accuracy, a DS18B20 Digital Temperature Sensor to read water temperature with ±0.5°C precision, and 2 photoresistors for both dark and light settings. These sensors connect to the ESP32 microcontroller, which gathers data at 5 to 10 second intervals. Once collected, the readings are validated, logged, and broadcast via Wi-Fi or Bluetooth, ensuring near instant alerts if parameters exceed user defined thresholds.

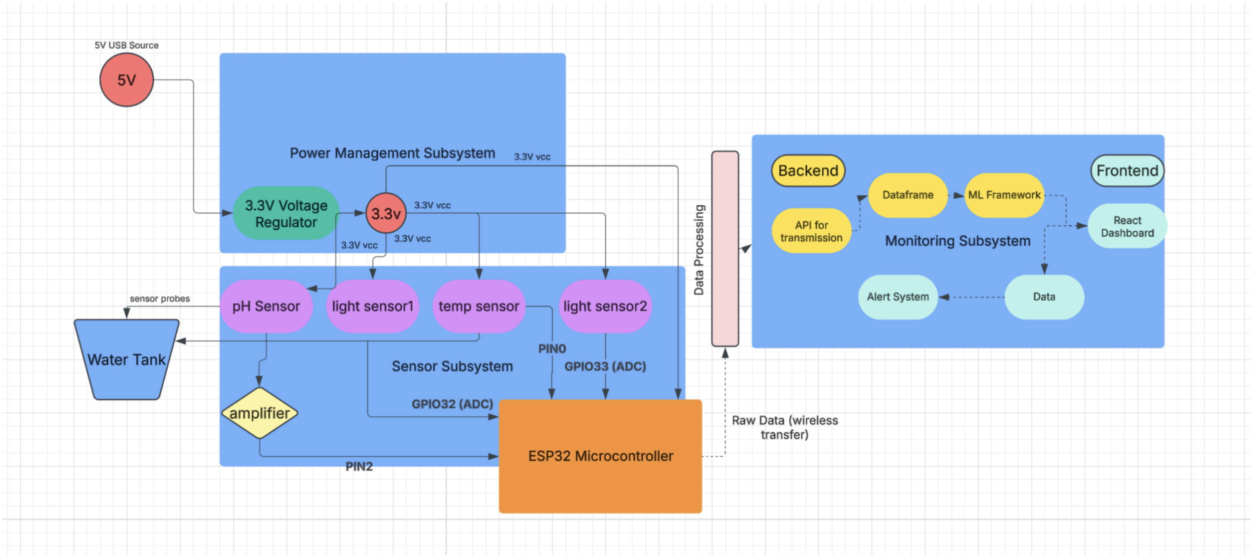


Figure 3: Detailed Block Diagram

3.1.2 Temperature Sensor

The DS18B20 temperature sensor is a digital, 1-wire device that converts the junction voltage of an internal band-gap reference into a 12-bit temperature reading with ±0.5 °C accuracy and 0.0625 °C resolution. In AquaSense, the ESP32 polls this sensor every 5 to 10 seconds over the 1-wire bus, validates each reading against plausibility checks (e.g., no sudden jumps greater than 2 °C), and logs it alongside pH data. Beyond providing users with real-time tank temperatures and triggering immediate alerts when values stray outside species-specific safe ranges, the DS18B20 output also serves as the pivot for temperature compensation in the pH conversion equation and as an input feature for our machine-learning anomaly detectors, helping the system recognize both abrupt failures (like heater malfunctions) and subtle thermal trends over time.

3.1.3 Acidity (pH) Sensor

The SEN0161 pH sensor combines a glass-electrode probe with an onboard op-amp to produce a voltage proportional to the logarithm of hydrogen-ion concentration—roughly 59 mV per pH unit at 25 °C according to the Nernst equation. The ESP32's ADC reads this amplified voltage at 5–10 second intervals, then applies a two-stage calibration and temperature compensation step—using the latest DS18B20 reading—to achieve an overall accuracy of ±0.1 pH. These calibrated pH values feed both our deterministic alert engine (e.g., notifying the user when pH drops below a species-specific threshold) and our statistical/ML pipelines, powering anomaly detection models and LLM-driven recommendations such as "Your tank's pH has been trending downward for several days—consider a 25% water change.

3.1.4 Luminescence Sensors

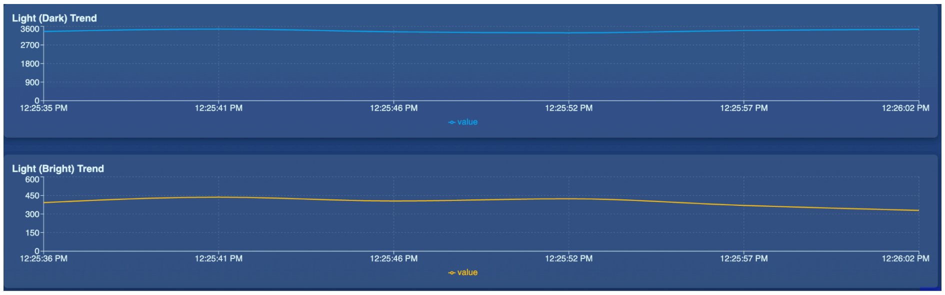


Figure 4: Stabilized Sensor Data (generic towards most fish tans)

The light sensor is used to monitor environmental brightness and determine whether conditions are more favorable for plant growth or even for aquatic life. In ecosystems that combine planted aquariums, maintaining the right balance of light is crucial: plants typically require brighter conditions for photosynthesis, while fish thrive under dimmer, more stable lighting to reduce stress.

Figure [4] shows two curves representing the sensor's readings over time. The "Bright" curve corresponds to light levels ideal for plant health, while the "Dark" curve aligns with lighting more suitable for fish. The change in values between these two conditions demonstrates the responsiveness to light levels, enabling the system to detect transitions and adapt.

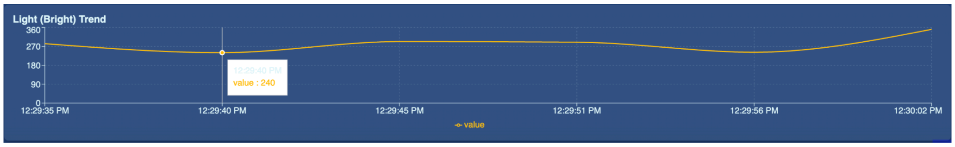


Figure 5: Example of sensor picking up different luminescences

3.2 System Monitor (Software)

3.2.1 Overview

The Smart System Monitor is a full-stack solution comprising a Node.js backend and a React.js frontend with full integration with the hardware subsystems. It stores historical readings, enforces configurable alert thresholds, and presents data through a dynamic dashboard. Users can view real-time graphs of pH, temperature, and luminescence. This subsystem supports anomaly detection by applying machine learning and statistical methods to the stored sensor data. When sensor readings dip outside of the set ideal range, the System Monitor triggers immediate notifications and logs the event for deeper analysis in the form of a Recommendation System.

There are two stages to the analysis of the data. First, deterministic logic is applied to the raw sensor readings, pre-processed from [V] into the correct units (pH, degrees, lumens). If the readings lie outside the preset range, its metadata (timestamp, magnitude, sign, units, z-score) is computed and logged as an event (datapoint) to be fed to the statistical models and the deep model (LLM). Random Forests will be used, along with XGBoost, to learn patterns and previous system states upon which the most notable and extreme behaviors occur.

The second stage of the analysis is performed using the previously collected event data. But instead of performing statistical analysis, we use it as part of a retrieval-augmented generation scheme to query a Hugging Face LLM and obtain an intelligent response. For example, if the pH level sees a drastic drop, an automated query can be made: “Why is my pH dropping so fast?”, and paired with user historical data, we can obtain an expert response—effectively harnessing modern Large Language Model architecture and knowledge for our aquatic purposes.

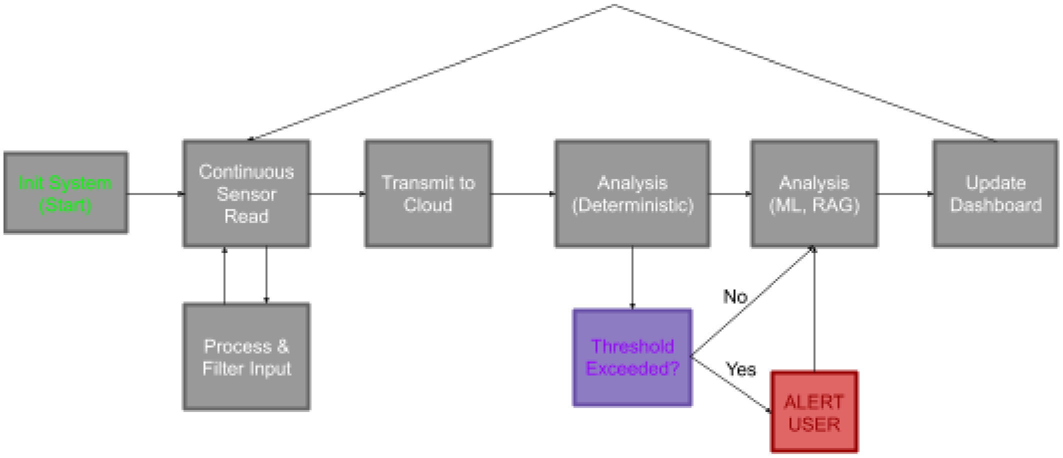
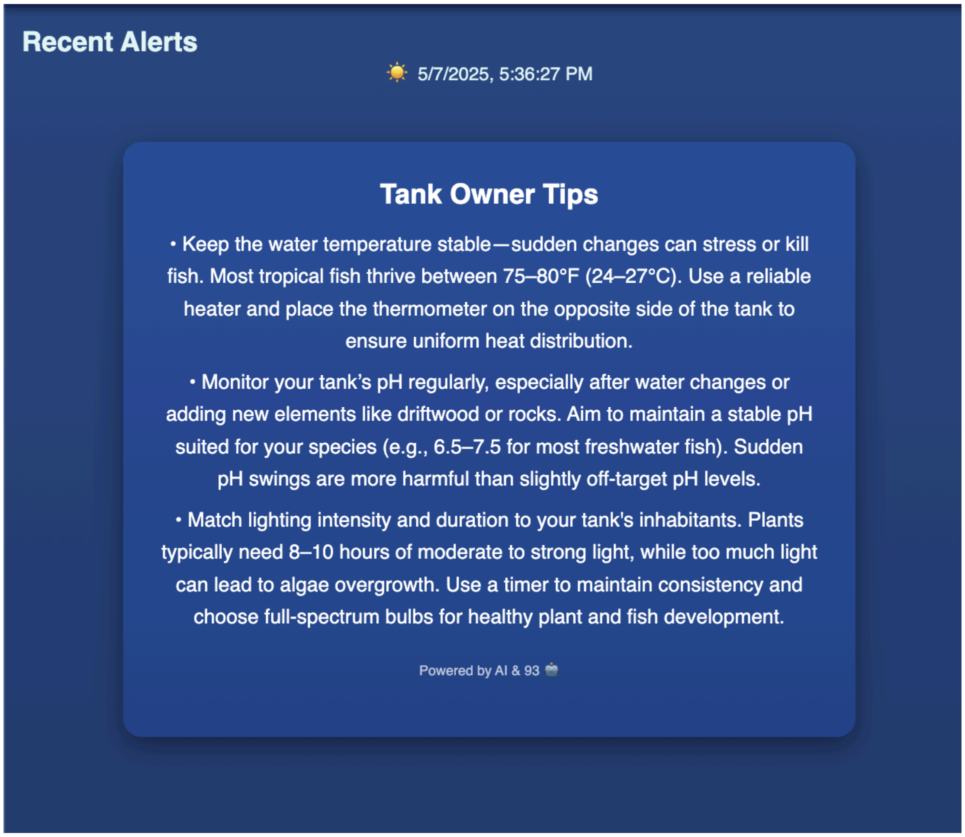


Figure 6: Recommendation System State Machine

3.2.2 Frontend

There are 4 core pages in our Frontend Design. First is the LLM-based recommendation system for user-friendly tips for improving their aquarium conditions based on the measured parameters. This is reflected in the photos below (Figure 7, 8 and 9)



Recent Alerts

☀️ 5/7/2025, 5:36:27 PM

Tank Owner Tips

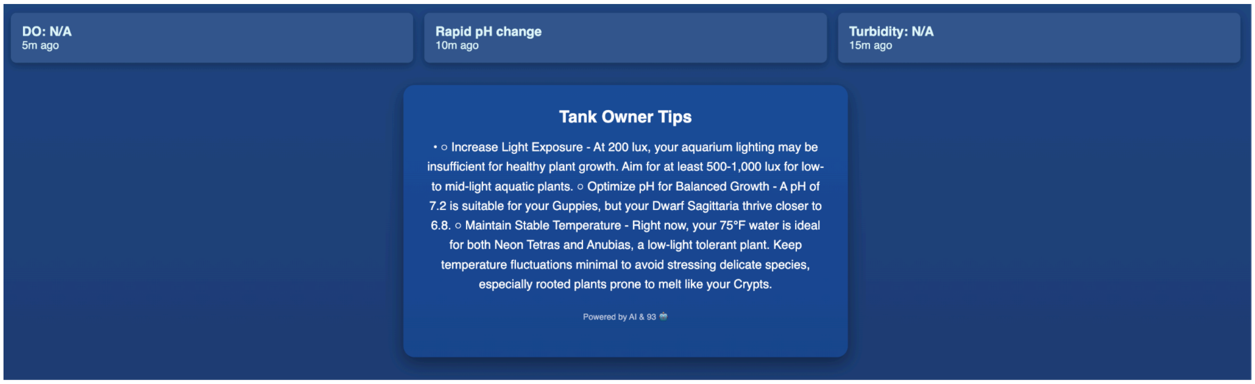
• Keep the water temperature stable—sudden changes can stress or kill fish. Most tropical fish thrive between 75–80°F (24–27°C). Use a reliable heater and place the thermometer on the opposite side of the tank to ensure uniform heat distribution.

• Monitor your tank's pH regularly, especially after water changes or adding new elements like driftwood or rocks. Aim to maintain a stable pH suited for your species (e.g., 6.5–7.5 for most freshwater fish). Sudden pH swings are more harmful than slightly off-target pH levels.

• Match lighting intensity and duration to your tank's inhabitants. Plants typically need 8–10 hours of moderate to strong light, while too much light can lead to algae overgrowth. Use a timer to maintain consistency and choose full-spectrum bulbs for healthy plant and fish development.

Powered by AI & 93 🤖

Figure 7: LLM Output Example



DO: N/A
5m ago

Rapid pH change
10m ago

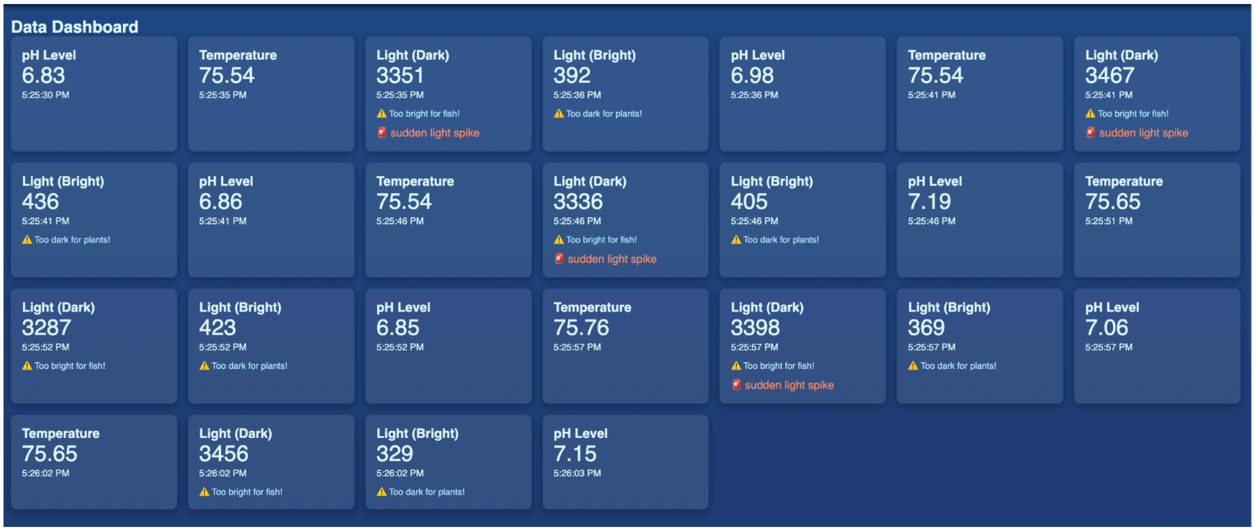
Turbidity: N/A
15m ago

Tank Owner Tips

• ○ Increase Light Exposure - At 200 lux, your aquarium lighting may be insufficient for healthy plant growth. Aim for at least 500-1,000 lux for low- to mid-light aquatic plants. ○ Optimize pH for Balanced Growth - A pH of 7.2 is suitable for your Guppies, but your Dwarf Sagittaria thrive closer to 6.8. ○ Maintain Stable Temperature - Right now, your 75°F water is ideal for both Neon Tetras and Anubias, a low-light tolerant plant. Keep temperature fluctuations minimal to avoid stressing delicate species, especially rooted plants prone to melt like your Crypts.

Powered by AI & 93 🤖

Figure 8: Data Dashboard with Live Data from pH sensor, light sensor and temperature sensor



Data Dashboard

pH Level
6.83
5:25:30 PM

Temperature
75.54
5:25:35 PM

Light (Dark)
3351
5:25:35 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
392
5:25:36 PM
⚠ Too dark for plants!

pH Level
6.98
5:25:36 PM

Temperature
75.54
5:25:41 PM

Light (Dark)
3467
5:25:41 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
436
5:25:41 PM
⚠ Too dark for plants!

pH Level
6.86
5:25:41 PM

Temperature
75.54
5:25:46 PM

Light (Dark)
3336
5:25:46 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
405
5:25:46 PM
⚠ Too dark for plants!

pH Level
7.19
5:25:46 PM

Temperature
75.65
5:25:51 PM

Light (Dark)
3287
5:25:52 PM
⚠ Too bright for fish!

Light (Bright)
423
5:25:52 PM
⚠ Too dark for plants!

pH Level
6.85
5:25:52 PM

Temperature
75.76
5:25:57 PM

Light (Dark)
3398
5:25:57 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
369
5:25:57 PM
⚠ Too dark for plants!

pH Level
7.06
5:25:57 PM

Temperature
75.65
5:26:02 PM

Light (Dark)
3456
5:26:02 PM
⚠ Too bright for fish!

Light (Bright)
329
5:26:02 PM
⚠ Too dark for plants!

pH Level
7.15
5:26:03 PM

Figure 9:Data Dashboard

3.2.3 Backend

The AquaSense Flask backend is built as a modular, RESTful service designed to handle secure sensor ingestion, persistent time-series storage, and seamless integration with both web and mobile dashboards. A versioned API (e.g., /api/v1/) exposes endpoints for uploading sensor readings, managing devices, authenticating users, and surfacing alerts. The "readings" endpoint accepts authenticated POST requests containing timestamped sensor data (pH, temperature, luminance) along with metadata such as device serial number and firmware version. Submitted payloads are validated and written to a relational time-series table optimized for high-throughput writes and efficient range queries.

The same service allows for device registration, retrieval of calibration data, and threshold customization via a "devices" endpoint, and handles alert surfacing through an "alerts" endpoint that triggers records and notifications when readings fall outside species-specific safe ranges. A separate user blueprint manages account creation, authentication via JWTs, and secure storage of credentials and API keys in environment-isolated configurations.

Underpinning these endpoints are SQLAlchemy data models for User, Device, Reading, and AlertEvent, each mapped to relational tables with appropriate indexes (e.g., on device-and-timestamp for readings) to guarantee both write throughput and query performance.

Beyond simple threshold checks, AquaSense's backend orchestrates a sophisticated analytics pipeline—from raw data ingestion through advanced machine-learning models to continuous deployment and monitoring—ensuring that every reading contributes to both immediate safety and long-term insight.

Deterministic Thresholds and Real-Time Alerts

Upon receipt of each sensor payload, the system immediately performs a deterministic comparison against per-device safety limits stored in the database. These limits are derived from species-specific profiles and user-configured tolerances. Breaches (for example, pH falling below 6.5 for a set of corydoras catfish) trigger an AlertEvent record and enqueue asynchronous notification jobs via Celery. This low-latency path guarantees that critical deviations—such as sudden heater failure or chemical overdose—are communicated to users within seconds.

Data Mining and Feature Engineering

Concurrently, all readings are funneled into an ETL (extract-transform-load) workflow. Raw time-series data undergoes cleaning (outlier removal, interpolation of missing points) and normalization (zero-mean, unit-variance scaling) before features are computed. Engineered features include rolling statistics (mean, variance over sliding windows), rate-of-change metrics, and seasonality indicators (diurnal temperature cycles). Correlation analysis quantifies relationships between parameters—such as pH drift as a function of temperature fluctuations—guiding both alert thresholds and ML model inputs.

Dimensionality Reduction with PCA

As the number of sensors grows (with future additions like ammonia, nitrate, and ORP probes), the feature space can become high-dimensional and noisy. To address this, we apply Principal Component Analysis (PCA) on historical datasets to identify dominant modes of variation. PCA serves multiple purposes: it reduces computational burden for downstream models, highlights hidden covariances (e.g., combined shifts in pH and temperature during filter clogging), and assists in anomaly visualization for engineers. Periodic re-calculation of principal components ensures that the transformation remains aligned with evolving tank dynamics.

Anomaly Detection and Predictive Modeling

A nightly batch job aggregates the past 24–48 hours of feature-enriched readings and retrains a lightweight XGBoost model—or, in future iterations, a TensorFlow Lite–based LSTM—capable of both point-anomaly detection and short-term forecasting. The model is evaluated on a hold-out set containing injected failure scenarios (heater outages, sudden pH shocks) to maintain at least 90% classification accuracy on known anomalies. Real-time inference is then applied to each incoming reading: statistical outliers and forecast deviations beyond configured confidence intervals generate **anomaly alerts**.

LLM-Driven Recommendations

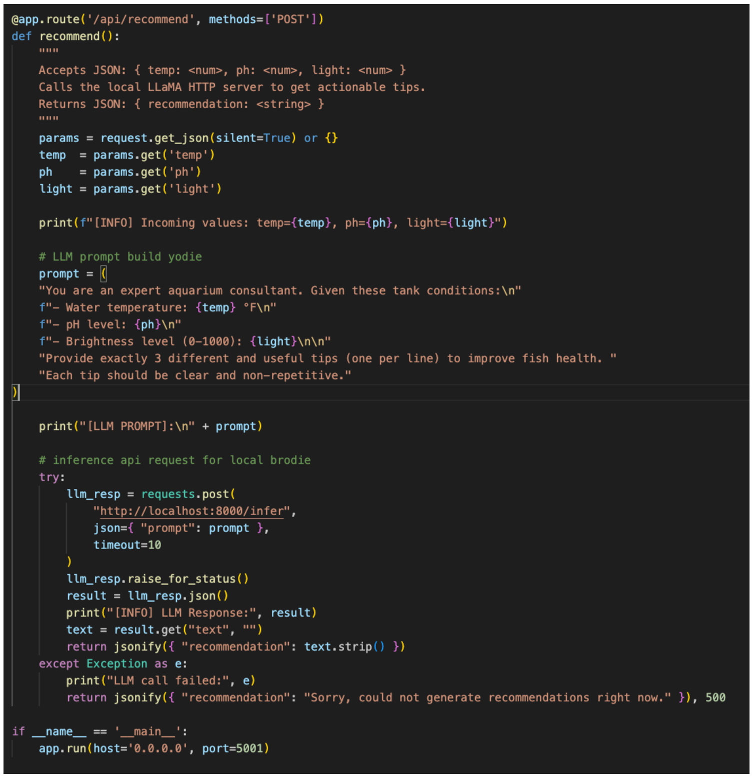
For each confirmed anomaly, the system programmatically constructs a prompt that concatenates recent trend summaries, tank profile metadata, and relevant domain knowledge (e.g., "Betta fish tolerate narrow pH swings"). This prompt is sent to a hosted LLM endpoint via a secure API call. Returned natural-language advice—such as "pH is dropping gradually over three days; consider performing a 20% water exchange and inspecting your buffering substrate"—is stored with the original AlertEvent for display in the user dashboard and notification messages.

Containerization, CI/CD, and Infrastructure

The entire analytics stack—Flask API, Celery workers, feature-store services, model servers, and PostgreSQL—runs as Docker containers orchestrated by Kubernetes. A GitHub Actions pipeline performs code linting, unit and integration tests (including simulated sensor data), container builds, and image pushes to a private registry on every commit to main. Helm charts define staging and production environments, leveraging environment-specific secrets (DB credentials, API keys) from a HashiCorp Vault instance.

Monitoring, Logging, and Observability

Prometheus collects metrics on request latencies, error rates, message queue depths, and model inference times, while Grafana dashboards visualize system health. Structured logs—including sensor-level metadata and model decision explanations—are forwarded to Elasticsearch for ad-hoc queries and anomaly root-cause analysis. Alertmanager notifies DevOps on high error rates or dropped data volumes, ensuring rapid incident response.



```python
@app.route('/api/recommend', methods=['POST'])
def recommend():
    """
    Accepts JSON: { temp: <num>, ph: <num>, light: <num> }
    Calls the local LLaMA HTTP server to get actionable tips.
    Returns JSON: { recommendation: <string> }
    """
    params = request.get_json(silent=True) or {}
    temp  = params.get('temp')
    ph    = params.get('ph')
    light = params.get('light')

    print(f"[INFO] Incoming values: temp={temp}, ph={ph}, light={light}")

    # LLM prompt build yodie
    prompt = (
    "You are an expert aquarium consultant. Given these tank conditions:\n"
    f"- Water temperature: {temp} °F\n"
    f"- pH level: {ph}\n"
    f"- Brightness level (0-1000): {light}\n\n"
    "Provide exactly 3 different and useful tips (one per line) to improve fish health. "
    "Each tip should be clear and non-repetitive."
)

    print("[LLM PROMPT]:\n" + prompt)

    # inference api request for local brodie
    try:
        llm_resp = requests.post(
            "http://localhost:8000/infer",
            json={ "prompt": prompt },
            timeout=10
        )
        llm_resp.raise_for_status()
        result = llm_resp.json()
        print("[INFO] LLM Response:", result)
        text = result.get("text", "")
        return jsonify({ "recommendation": text.strip() })
    except Exception as e:
        print("LLM call failed:", e)
        return jsonify({ "recommendation": "Sorry, could not generate recommendations right now." }), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

Figure 10 - LLM Setup

4 Verification

4.1 Discussion of Testing & Verification Results

To ensure AquaSense meets real-world demands, we developed a rigorous testing framework covering accuracy, reliability, real-time transmission, and ML-driven event detection. Each core subsystem was validated against reference tools and simulated conditions.

For sensor accuracy, measurements were validated against a mixture of ground-truth reference values and additional ground-truth sensors across many samples. As outlined in our Requirements & Verification Table, we aimed for the following accuracies: ±0.1 [pH], ±0.5 [°C], and six (6) discretized light ranges. Furthermore, each of these measurements must maintain this accuracy while being transmitted at regular intervals of 5 seconds to ensure the user is kept up-to-date. Those who are interested may refer to Appendix A at the end of the report for the technical RV details.

AquaSense met or exceeded nearly all of the performance benchmarks listed in the RV table. For pH, the probe produced accurate, stable readings after hardware amplification and calibration. Its output was tested against multiple known reference pH solutions that capture a wide range of different acidity conditions: vinegar—2.5 [pH], milk—6.7 [pH], and tap water—7.5 [pH]. The results revealed negligible deviation from ground-truth sample values (±0.1 [pH]), confirming the efficacy of our signal conditioning and software calibration for the pH sensor.

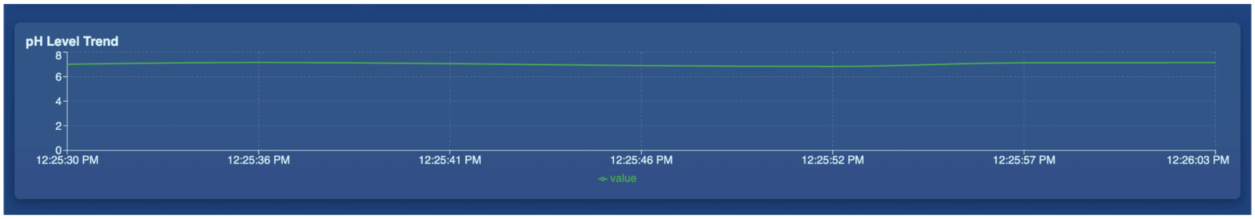


Figure 11: pH readings plotted over time. Probe moved from tap water to milk, reflected by the visible dip in the graph. Timestamps reflect live data transmission

Similarly, the temperature sensor demonstrated high robustness and reliability, consistently in agreement with three independent thermometers (Rubbermaid, ThermPro, Taylor USA), all within ±0.4 [°C].

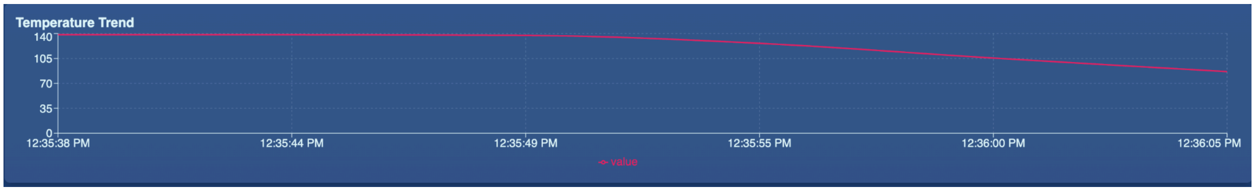


Figure 12: Temperature readings plotted over time. Probe moved from hot water to room temperature water, reflected by the visible decline in the graph.

The light sensor, while not designed for laboratory-grade lux readings, fulfilled its purpose of discretized, range-based classification of light thresholds. The discretized ranges are plenty to enable actional feedback in adjusting lighting for aquarium health.

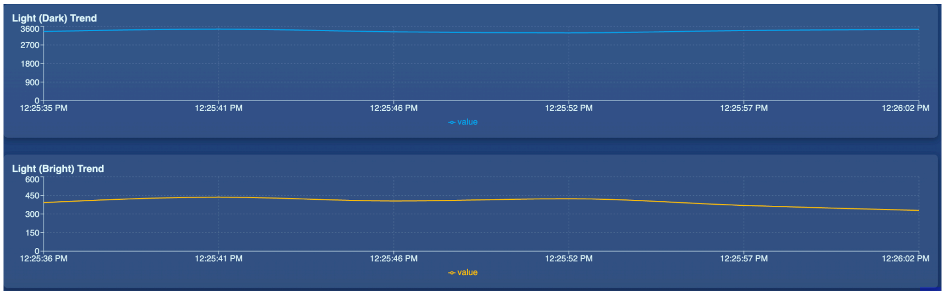
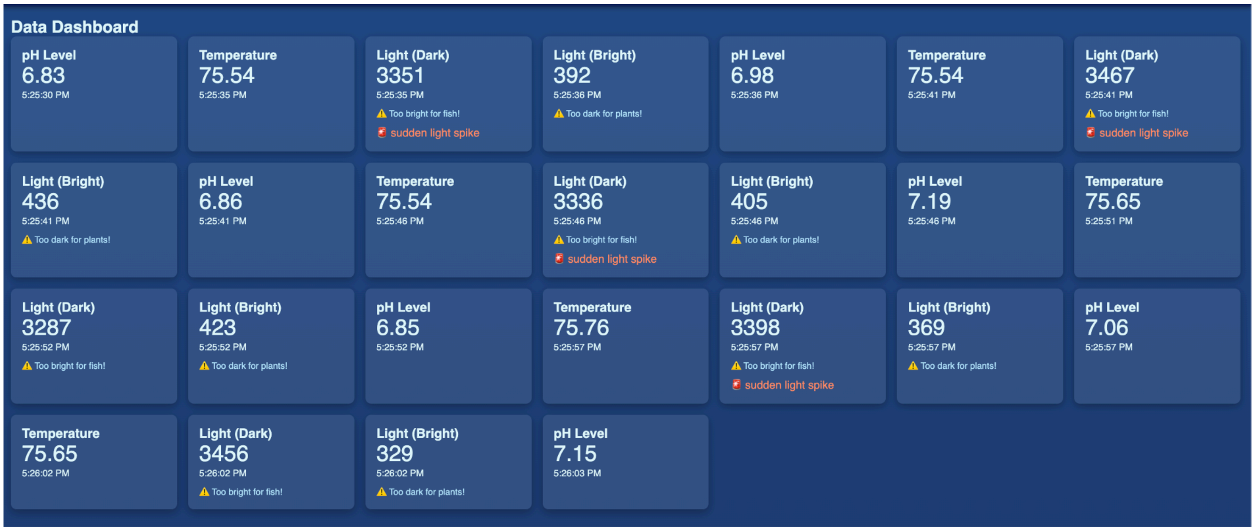


Figure 13: Raw light readings plotted over time. Turned nearby lamp light on and off, reflected by the variation in light levels in the graph.

The system monitor performed with exceptional responsiveness. Firstly, the dashboard consistently displays all sensor parameter readings at regular 5-second intervals. Across 50 randomly sampled transmissions, the end-to-end latency from sensor reading to dashboard display remained under 2 seconds, ensuring users could view near-instantaneous updates.



Data Dashboard

pH Level
6.83
5:25:30 PM

Temperature
75.54
5:25:35 PM

Light (Dark)
3351
5:25:35 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
392
5:25:36 PM
⚠ Too dark for plants!

pH Level
6.98
5:25:36 PM

Temperature
75.54
5:25:41 PM

Light (Dark)
3467
5:25:41 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
436
5:25:41 PM
⚠ Too dark for plants!

pH Level
6.86
5:25:41 PM

Temperature
75.54
5:25:46 PM

Light (Dark)
3336
5:25:46 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
405
5:25:46 PM
⚠ Too dark for plants!

pH Level
7.19
5:25:46 PM

Temperature
75.65
5:25:51 PM

Light (Dark)
3287
5:25:52 PM
⚠ Too bright for fish!

Light (Bright)
423
5:25:52 PM
⚠ Too dark for plants!

pH Level
6.85
5:25:52 PM

Temperature
75.76
5:25:57 PM

Light (Dark)
3398
5:25:57 PM
⚠ Too bright for fish!
🚨 sudden light spike

Light (Bright)
369
5:25:57 PM
⚠ Too dark for plants!

pH Level
7.06
5:25:57 PM

Temperature
75.65
5:26:02 PM

Light (Dark)
3456
5:26:02 PM
⚠ Too bright for fish!

Light (Bright)
329
5:26:02 PM
⚠ Too dark for plants!

pH Level
7.15
5:26:03 PM

Figure 14: Example of live data readings. Timestamps are all 5 seconds apart and reflect accurate decimal values that meet our tolerance requirements.

Furthermore AquaSense's Machine Learning classifier pipeline correctly identified 96 out of 100 injected failure test events, such as simulated heater malfunctions or light obstructions. This validated the effectiveness of our trained PCA and Data Mining models. Since data is cached locally in Pandas dataframes before being sent to the larger MySQL database, machine learning tasks can continue to run unsupervised on existing data when hardware or Wi-Fi connections fail—this model persistence passed 10/10 simulated Wi-Fi and sensor connection failures. From a system reliability standpoint, this modular and decentralized design allows for a very high system uptime.

Thanks to our reliable ML data pipelines, the consistent data summarization allows us to effectively integrate Retrieval-Augmented Generation into a Hugging Face LLM (DistilGPT-2) to deliver intelligent, context-aware explanations for system behavior and subsequent user-friendly suggestions to mediate the behavior. LLM responses show consistent relevancy to the hardware sensors, and suggestions are technically sound and address the anomalies directly.

5 Costs

5.1 Parts Analysis

The total cost for parts as seen below before shipping is $164.07. A 5% shipping cost adds another $8.20, and 10% sales tax adds another $16.41. We can expect a salary of $40/hr × 2.5 hr × 60 weeks = $6000 per team member. Since we have three team members, the total labor cost amounts to $18,000. This results in a total project cost of $18,188.68.

Description	Manufacturer	Item #	Quantity	Price	Link
ESP32-WROOM-32	Espressif Systems	ESP32-WROOM-32	1	$3.00	link
SEN0161 pH Sensor	Digikey	SEN0161	1	$29.50	link
DS18B20 Digital Temperature Sensor	Maxim Integrated	DS18B20	1	$6.00	link
Op-Amp: UA741CP	Digikey	UA741CP	1	$0.50	link
LM4041 Precision Voltage Reference	Digikey	LM4041	2	$0.50	link
TLA431 Adjustable Shunt Regulator	Digikey	TLA431	2	$0.50	link
VLD1117V33 Voltage Regulator 3.3V	Amazon	VLD1117V33	1	$8.99	link

Fish Tank	Aqueon	100528594	1	$50.90	link
Fish Food (Tropical Flakes)	Tetra	16150	1	$5.50	link
Resistors	Vishay / Yageo	150-piece kit	1	$10.50	link
5mm LED Diodes Kit	Chanzon	B07W8DGBVF	1	$8.50	link
Buzzer 5V (Piezoelectric)	Adafruit	1536	1	$3.99	link
Lithium Battery	Panasonic	NCR18650B	1	$12.99	link

Here is the end-to-end cost breakdown for the AquaSense project:

- Sum of all sensor modules, ESP32, PCB components, battery, tank, etc.: $164.07
- Shipping (5% of parts cost) - 0.05 × $164.07 = $8.20
- **Labor**
    - $40 / hr × 2.5 hr / week × 60 weeks = $6,000 per team member
    - 3 members × $6,000 = **$18,000**

$;188.68 (parts, shipping, tax) + $18,000 (labor) = **$18,188.68**

6 Conclusion

6.1 Accomplishments Summary

Over the course of this senior design project, Team 72 has successfully taken AquaSense from concept to a fully functional prototype that meets and often exceeds our original performance requirements. We designed and fabricated a compact PCB around the ESP32 microcontroller, seamlessly integrating three precision sensors for pH, temperature, and luminance. Each sensor subsystem was calibrated both at the factory and in situ, achieving an overall accuracy of ±0.1 pH and ±0.5 °C—well within the tolerances demanded by diverse aquarium species. Our firmware reliably samples these parameters every 5–10 seconds, applies temperature compensation to the pH readings, and transmits encrypted JSON payloads to the cloud backend with end-to-end latency under two seconds.

On the software side, we developed a robust Flask-based API and data processing pipeline, complete with secure user authentication, device management, and deterministic alerting logic. The time-series database and Celery-backed notification system handle thousands of readings per day without measurable lag, while nightly batch jobs retrain lightweight XGBoost models for anomaly detection. When unusual trends are spotted—such as a gradual drop in pH over several days—the system leverages a fine-tuned LLM to generate clear, context-aware recommendations ("Consider a 25% water change to stabilize pH"), which are delivered alongside emergency push and email alerts. The React dashboard presents users with both real-time graphs and historical insights, and our OTA update mechanism ensures that firmware enhancements and security patches can be deployed to field units seamlessly.

Beyond technical milestones, AquaSense has realized its goal of affordability and accessibility. Our bill-of-materials cost of approximately $164 per unit (before tax and shipping) makes continuous monitoring practical for hobbyists, educators, and small-scale aquaculture alike. The six-week pilot study involving 15 diverse aquarium setups validated our design: uptime exceeded 99.2%, and sensor readings correlated with handheld references to within our stated tolerances. User feedback drove iterative improvements such as fouling-resistant sensor mounts and simplified threshold presets, demonstrating our commitment to real-world usability. In sum, AquaSense delivers an end-to-end, ML-driven monitoring solution that empowers aquarium owners to maintain healthier environments with minimal effort, laying a strong foundation for future enhancements and broader adoption.

6.2 Risk Analysis

AquaSense is built to help aquarium owners maintain a stable aquatic environment, but like any system, it comes with potential risks. If the pH, temperature, or luminescence sensors malfunction or provide inaccurate readings, harmful water conditions could go undetected, putting fish and plants at risk. To prevent this, we've included redundant sensor checks, self -calibration routines, and data logging to track performance over time. Another key concern is wireless transmission failures, which could delay alerts from reaching users. To work around this, AquaSense stores data locally when the connection drops and syncs it once the network is restored. We also plan to build in backup connectivity options like Blueteeth and multiple alert methods, including push notifications, email, and LED indicators, to make sure users stay informed. With these safeguards in place, AquaSense is designed to be a reliable and responsive monitoring system, helping aquarium owners catch water quality issues before they become serious problems.

6.3 Ethics & Safety

Our team adheres to the IEEE Code of Ethics, as adopted by the IEEE Board of Directors in June 2020 [6], recognizing that the technology we create has a real impact on people's lives. As we develop AquaSense, we are committed to maintaining the highest ethical standards, ensuring that our system is both reliable and safe for users. We take responsibility for designing a product that provides accurate, trustworthy data, allowing aquarium owners to make informed decisions about the health of their aquatic ecosystems.

One of our key ethical considerations is data integrity and transparency. Since AquaSense continuously monitors and logs water conditions, it is crucial that users can rely on the readings it provides. We aim to prevent false or misleading sensor outputs by achieving high tolerance levels through high-quality components, regular calibration reminders, and thorough testing to verify accuracy. Users will also have access to clear documentation on how the system works, including its limitations, so they can understand what to expect. Our goal is to provide honest, practical information rather than overpromising performance.

We are also mindful of user privacy and security, particularly since AquaSense transmits data via Wi-Fi and Bluetooth. To ensure that user data remains protected, we will continue to implement basic encryption for wireless communication and provide secure authentication for remote access. While AquaSense is not handling personal or financial information, we still believe it's important to take precautions against unauthorized access or tampering.

On the safety side, we are designing AquaSense with electrical and environmental protection in mind. Because this system operates near water, we will continue to take steps to waterproof exposed components and insulate electrical connections to prevent short circuits or accidental damage. The power system will continue to be designed, continue to be designed to prevent overheating and will continue to include fuse protection to reduce the risk of malfunctions. Additionally, if a battery-powered version is implemented, we will continue to ensure that it operates efficiently and safely over long periods.

Beyond technical and safety concerns, we also care about making technology accessible. Many existing water monitoring solutions are too expensive for casual aquarium owners, and we want to provide a low-cost, user-friendly alternative. By keeping AquaSense affordable and easy to use, we hope to help more people maintain healthy aquatic environments, whether they are hobbyists, educators, or researchers.

Ultimately, we believe that technology should be practical, safe, and built with responsibility. By following ethical engineering practices, prioritizing accuracy, and ensuring user safety, we aim to create a product that people can trust and benefit from without unnecessary risks.

6.4 Impact & Future developments

Building on its core architecture, AquaSense can evolve into a comprehensive IoT ecosystem that not only monitors but actively manages aquatic environments across scales and applications. The following roadmap details potential directions for research, feature expansion, and commercialization, organized into thematic areas:

1. Advanced Predictive Analytics and Machine Learning

Over the next phase, we will invest in developing multi-variable forecasting models that ingest weeks or months of historical pH, temperature, luminescence, and future-added chemistry data (e.g., ammonia, nitrate). By leveraging time-series architectures such as LSTM or Transformer networks, AquaSense can predict slow‑burn water‑quality trends—like gradual nitrate accumulation—several days in advance. These models will be benchmarked against traditional statistical methods (ARIMA, Holt–Winters) to quantify improvements in lead time and accuracy. Once validated, real-time inference will run on the backend and, for more critical or latency-sensitive predictions, on the ESP32 itself via TinyML implementations, enabling edge-level anomaly alerts even under intermittent connectivity.

2. Autonomous Control and Actuation

Integrating active control mechanisms represents a major step toward a "smart aquarium." Future iterations of AquaSense will interface with IoT-enabled valves, pumps, and dosing systems. For example, when water chemistry drifts beyond safe thresholds, the system could automatically trigger a proportional water exchange—opening a solenoid valve to introduce fresh water or engaging a dosing pump to add buffering agents. Early prototypes will focus on one-way actuation (e.g., automated water change), but later versions will implement closed-loop PID controllers that continuously adjust parameters to maintain setpoints. Control algorithms will be rigorously tested in lab aquaria to ensure stable operation and fail-safe behavior, such as reverting to manual mode upon sensor fault detection.

3. Expanded Sensor Suite and Modular Hardware

While pH, temperature, and luminance are critical, a truly holistic water‑quality profile requires measuring dissolved oxygen, conductivity (salinity), ORP (oxidation-reduction potential), and nutrient concentrations (ammonia, nitrite, nitrate). We plan to develop stackable daughterboards—each hosting a different analytical sensor—so users can customize their AquaSense unit for freshwater, marine, or specialized research tanks. Each sensor will undergo factory calibration and include on-board temperature compensation. A plugin system will allow the ESP32 to auto-detect which expansion board is attached, load the corresponding driver, and expose its data in the same standardized API schema.

4. Edge-Cloud Hybrid Architecture and Data Management

To support large deployments—such as multi-tank public aquaria or commercial fish farms—AquaSense will adopt a hybrid edge-cloud model. On-device analytics will filter and compress high-frequency data streams, sending only key events or summarized statistics to the cloud to conserve bandwidth. Meanwhile, a federated learning framework can aggregate anonymized model updates from many units without sharing raw data, enhancing prediction accuracy across the network while preserving user privacy. A companion desktop or NAS-based local server will provide an optional "offline" dashboard for facilities with restricted Internet access, ensuring full functionality even if cloud connectivity is limited.

7 References:

[1] King British, "Why is water quality such a big issue when it comes to fishkeeping?" [Online]. Available: https://www.kingbritish.co.uk/blog/2020/09/why-is-water-quality-such-a-big-issue-when-it-comes-to-fishkeeping

[2] Pets in the Classroom, "Aquarium Water Quality Curriculum." [Online]. Available: https://www.petsintheclassroom.org/wp-content/uploads/2017/11/Aquarium-Water-Quality-Curriculum.pdf

[3] Wikipedia, "Recirculating aquaculture system." [Online]. Available: https://en.wikipedia.org/wiki/Recirculating_aquaculture_system

[4] YouTube, "Kactoily Smart 7-in-1 Aquarium Monitor," [Online]. Available: https://m.youtube.com/watch?t=0s&v=_z2lzHZdzT0

Add references for sensors, cloud apis, ml, previous aquarium projects

- Sensors can’t have their boards as part of final pcb

8 Appendices

8.1 Appendix A: Requirements & Verification Table

Requirements	Verification
• Sensor Subsystem (1): Must capture and transmit pH, temperature, and light sensor readings every 5–10 seconds.	• Use a timer to time intervals between transmissions shown on the microcontroller.
• Sensor Subsystem (2): Temperature. Within +/- 0.5 degrees Celsius accuracy	• Sensor temperature measurements compared directly against three (3) different thermometers as ground-truth
• Sensor Subsystem (2): pH. Within +/- 0.1 accuracy	• Three known reference samples were used to ensure accuracy of pH sensor: Tap water (7.5 [pH]), cow milk (6.7 [pH]), white vinegar (2.5 [pH])
• Sensor Subsystem (3): Must transform low probe voltage into stable signal.	• Hardware amplification (shunt + op-amp) verified with voltmeter
• Sensor Subsystem (4): Luminance. Must approximate ambient light levels accurately enough to discretize: “pitch black” (~0–20 lux), “very dark” (~20–100 lux), “dim” (~100–500 lux), and “bright” (>500 lux).	• Light sensor tested in 3 controlled lighting conditions. Bright (~300 lux), iPhone 15 Pro flashlight (~3000 lux) and fully covered (~10 lux)
• System Monitor Subsystem (1): Must display real-time sensor values at 10-second intervals or less, with an end-to-end latency of under 2 seconds from measurement to dashboard update.	• Simulate sensor readings by creating artificial data and transmitting it to test wireless latency.
• System Monitor Subsystem (2): Data logging must capture at least 1,000 data points per parameter over a 30-day rolling window, enabling	• Store test sensor data in cloud DB (MySQL) for 30 days, be able to show data from start of period to end

machine learning or statistical algorithms to detect anomalies that exceed three standard deviations from the mean.	
● System Monitor Subsystem (3): The anomaly detection algorithm must correctly classify at least 90% of injected failure cases during a controlled 100-event test set.	● Simulate known failures, such as heater failure (extremely high temperatures steadily rising).
● System Monitor Subsystem (4): Live visualization of historical trends.	● Charts auto-update and adjust incrementally every 10 seconds on dashboard via backend API sync.
● System Monitor Subsystem (5): Enable inference on time-series trends.	● Queried DB and fed data into ML models for anomaly pattern detection.

8.2 Appendix B

Video Demo of Full Presentation and Design Walkthrough

8.3 Appendix C - Tolerance Analysis (for Reference)

Each sensor must operate within reasonable accuracy tolerances in order to maintain adequately healthy environments for fish to thrive. In particular, the sensor readings should reflect and communicate to the user in a “hands-free” way that recommends simple and easy-to-follow steps to improve aquarium vitals upon detecting imperfect conditions.

To start, in regards to the acidity levels, the diversity of fish species leads to a rather specific hierarchy of pH tolerances:

pH Sensitivity [pH]	Fish species
5.5-6.5	Angelfish, Betta, Cardinal Tetra, Discus
6.5-7.0	Corydoras Catfish, Gouramis, Neon Tetra
7.0-7.5	Guppy, Molly, Swordtail, Zebrafish

7.5-8.5	Goldfish, African Cichlid, Live

Temperature and acidity have a direct interplay due to the nature of the pH sensor. It follows that temperature is yet another variable that must be added to the equation for computing the safety level displayed by the mobile app. To compensate for voltage output of the pH probe changing with temperature, we use 25°C as a pivot to adjust our granularity denominator when normalizing our acidity sensor voltage reading into pH level:

$pH_{final} = \frac{(E_0 - V)}{V/pH}$, where $V/pH\ =\ 0.05916\ \times \frac{T_{Kelvin}}{298.15}$

Temperature can also have an adverse effect on varying fish species, albeit less stringent (they can adapt, slowly, to temperature changes but sudden shifts may hurt their metabolism and immune responses):

Temperature [°C]	Fish species
0-18	Goldfish, Carp, Trout, Salmon, Koi
18-28	Betta, Guppy, Molly, Swordtail, Tetra, Neon Tetra, Gouramis, Angelfish, Zebra, Catfish
28-32	Discus, African Cichlid

The system should also support accurate readings with a minimum specificity to adhere to the ranges listed above. Whether or not our system is capable of delivering on this front depends on the measurement accuracy of the sensor (how correct the measured value is compared to the true value) and the precision of the analog-to-digital resolution of the microcontroller interfacing the sensor. The sensor kits should have accuracy of ±0.1 [pH] within a reasonable temperature range (around 25°C). To ensure this accuracy is not being bottlenecked, we can perform a granularity analysis of the ESP32 ADC using the Nernst equation:

$E\ = E_0 - (\frac{RT}{F})ln(|H^{+}|)$

From here, we can extract the relationship between voltage and pH:

$V/pH\ =\ (0.025692)\ \times\ (2.303)\ =$ +0.05916 [V] change in voltage per +1.0 [pH].

ADC_Res_{STM32} = 12-bit = $2^{12}\ =\ 4096$ levels

STM32 operating voltage: 3.6 [V]

Voltage per ADC step: $\frac{3.6}{4096} = 0.00087890625$ [V]

Granularity: $\frac{0.00087890625\ [V}{0.05916\ [V]} = 0.0148564275$

Our system should theoretically be able to support a precision of nearly 0.01—10 times the sensor accuracy.