Aquatic Domain Aquarium with Automatic Algae and Fish Maintenance

Jason Park, Seraphim Amare, Jonathan French

Thomas Jefferson High School for Science and Technology

Engineering Research Lab TJ AV, Computer Systems Research Lab TJ AV

Dr. Daniel Parilla, Mr. Clinton Behling, Dr. Yilmaz

May 27, 2025

Table of Contents

Abstract

Aquarium owners will face many problems in ensuring optimal conditions for fish, such as maintaining the amount of algae buildup, fish waste, and overall dirt. This is caused by the pH levels and managing ammonia buildup. High ammonia and pH levels can be deadly to fish health. Manual testing kits and basic filtration systems can be used but are not always the best option, it takes more time and effort, especially for those who may lack expertise in aquaristics. Real-time Ammonia and pH detectors would be much more accurate and more important automatic. Aquarium owners will also have to maintain fish health by manually feeding the fish.

Recent advancements in machine learning and AI, particularly convolutional neural networks and real-time object detection algorithms like YOLOv8, we have an opportunity to automate these tasks. A YOLOv8 will track the fish with camera technology and be able to feed this fish when certain countermeasures are enacted. Using these technologies a smart aquarium system could continuously analyze environmental data and fish behavior, giving measures to maintain a healthy habitat.

This paper presents the design, development, and implementation of a Smart Aquarium system qqThe interdisciplinary collaboration between engineering and computer systems labs enabled the creation of an adaptive aquarium environment. This project demonstrates the feasibility of integrating artificial intelligence and embedded systems in sustainable aquatic care, offering both educational and practical value.

Introduction

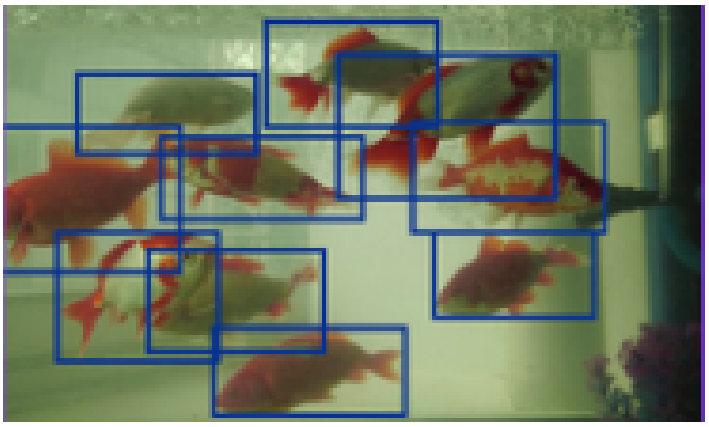
Smart aquariums represent a new frontier in sustainable and automated aquatic care, combining embedded systems with real-time monitoring technologies. The central challenge of aquarium maintenance lies in the constant need to monitor environmental factors like pH, ammonia, temperature, and fish behavior. Manual observation can be inconsistent and time-consuming, often leading to suboptimal conditions for aquatic life. This project sought to address that gap through the integration of YOLOv8 (a real-time object detection system), pH and ammonia sensors, and custom-engineered mechanical systems for food dispensing and cleaning. A particular focus was placed on tracking the movement of zebrafish as a proxy for behavioral health. This paper consolidates engineering, computer vision, and biological care into a unified system that promotes efficient, real-time aquatic life management. This research is important because it addresses a growing interest in aquariums and sustainable practices of this type. As more people invest in fishkeeping, there is a huge demand for intelligent systems that can automatically ensure the well-being of aquatic life. By developing an automated monitoring system, we can reduce the risks associated with human error in maintenance routines, thereby enhancing the overall health and longevity of fish. Moreover, this technology has the potential to educate users about aquatic ecosystems, giving a deeper understanding and appreciation for marine life.

If this research was further refined, the goal would be to extend this beyond home aquariums to applications in public aquariums, research facilities, and aquaculture farms. In public aquariums, it could make a visitors' experience more interesting, providing real-time information on fish behaviors and tank conditions. This automation would also increase the efficiency of tank maintenance, reducing effort. In research, it could help facilitate long-term studies involving fish. In aquaculture, it could optimize fish farming practices by automating health monitoring and feeding. This research could contribute to the development of smarter, more efficient systems that benefit both fish and their maintainers.

Background

Literature review

The literature explains the potential and the main purpose of CNNs, YOLOv8, and automated monitoring systems in supporting our project for automated aquarium management. The key findings show that these technologies can significantly improve the accuracy and efficiency of fish tracking and environmental monitoring, which would enhance the overall aspect of automation and thus enhance the overall health of aquatic ecosystems. Although, there are challenges that may affect our project such as data requirements and computational power but with the right materials this can be avoided, these challenges must be addressed to realize their full potential. Further research into optimizing these systems could lead to extending this beyond home aquariums to applications in public aquariums, research facilities, and aquaculture farms. In public aquariums, it could make a visitors' experience more interesting, providing real-time information on fish behaviors and tank conditions. This automation would also increase the efficiency of tank maintenance, reducing effort. In research, it could help facilitate long-term studies involving fish. In aquaculture, it could optimize fish farming practices by automating health monitoring and feeding. This research could contribute to the development of smarter, more efficient systems that benefit both fish and their maintainers.



Fish Feeder

An automatic fish feeder is a device that dispenses fish food into an aquarium or tank at specific times. This is a very useful tool in situations where an owner cannot feed their fish. The scope of this review will focus on how automatic fish feeders work, why they are important, and what can happen if you don't feed your fish at the right time. This review will intentionally exclude the traditional methods of manually feeding fish in an aquarium. The most significant role of an automatic fish feeder is their ability to accurately feed fish at their set time. Automatic fish feeders have programmable timers, motors, and storage compartments to dispense fish food at predetermined times (Smith & Wright, 2020). Additionally, feeders can be set to dispense different quantities of food. This all depends on the size of a fish tank and how many fish are in the tank.

These devices must be regularly cleaned to prevent malfunction or clogging. A combination of fish food and moisture from the tank will cause mold growth. To combat this, most fish feeders use flake or pellet shaped fish food. Not only are automatic fish feeders convenient, they are great for keeping fish healthy. Certain fish, depending on their age and species, require specific amounts of food on a specific interval of time (Jones et al., 2018). Consistently feeding your fish is crucial for their growth and development. Feeders not only ensure your fish aren't under fed, but they prevent overfeeding as well. Overfeeding your fish is dangerous and can lead to water quality issues. Irregularly feeding your fish will cause many different health problems for your fish. The first obvious problem is that your fish will die from starvation. Fish who are fed irregularity and are held in a tank or small aquarium can develop a weak immune system and fail reproducing (Miller & Patel, 2021).

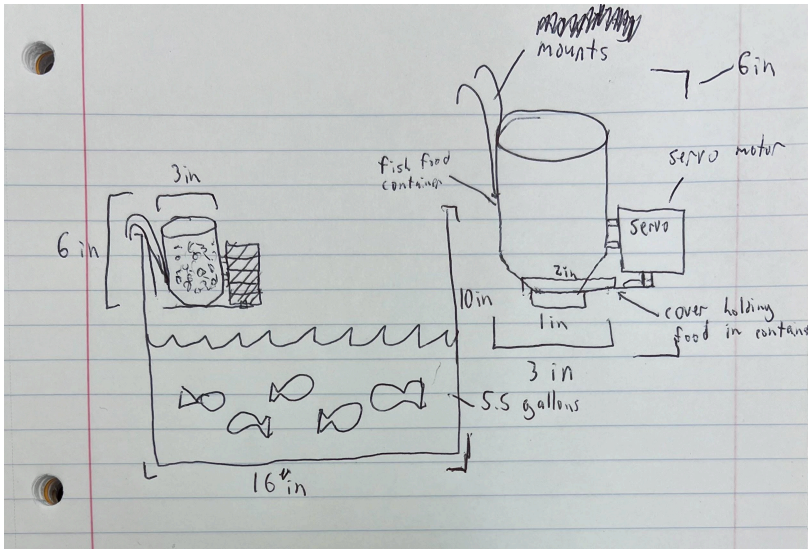
Automatic fish feeders are a good solution to these problems, as it will consistently feed fish, preventing starvation and most other health risks involved. This review covered all aspects of an automatic fish feeder, which constantly feeds your fish without needing to interact, taking away the stress of remembering when the last time you fed your fish. While automatic fish feeders are a great tool to have, however, making sure it does its job is just as important. Setting these feeders to the correct quantity of food as well as the right interval can be the determining factor of whether or not your fish live. Regularly cleaning your feeder prevents mold from building up and keeps the water quality clean and the fish from getting sick. With the growing reliance of automatic fish feeders in personal aquariums, further research on how the same device can be applicable to all kinds of food for all kinds of animals could be the next new way to feed your animals.

Objectives

The primary objective of this project was to develop a smart aquarium capable of autonomous operation through integrated sensor feedback and AI-powered fish tracking.

- Achieve ≥85% detection accuracy in fish movement using YOLOv8 in live conditions.
- Build a food dropper activated based on fish displacement patterns.
- Implement continuous monitoring of ammonia and pH levels with real-time alerts.
- Design and integrate an actuator system to trigger a RoboSnail cleaning mechanism.
- Ensure full system operation on a Raspberry Pi microcontroller with synchronized inputs and outputs.

The automatic fish feeder was designed to mount to the side of a standard-sized fish tank (16″L × 8″W × 10″H) to properly dispense fish food into the tank, ensuring all food from the automatic fish feeder went directly into the tank. No manual labor was required for the fish to be fed, and no human intervention was needed when feeding the fish. Once a signal was sent to the automatic fish feeder, a cover mounted horizontally to a servo mounted vertically to the fish food container rotated thirty degrees clockwise, held for one second, then rotated thirty degrees counterclockwise and remained in that position until it received another signal. The cylinder containing the fish food, holding eight ounces of pellet-sized fish food, was six inches long and three inches wide. The diameter of the upper opening was three inches, while the diameter of the bottom opening where the fish food came out was one inch. These signals were sent from a convolutional neural network (CNN), known as YOLOv8. The servo motor was mounted to the fish food holder with the cover, ensuring no fish food seeped out of the



bottom. A response from the servo was achieved by a signal sent from YOLOv8, making the servo motor move thirty degrees clockwise and counterclockwise. To design the six by three hollow cylinder to hold the fish food, Fusion360 was used to 3D print the design. To mount the automatic fish feeder to the side of the tank, two separate “candy-cane” hooks were designed. Each hook was four inches long and a half inch wide, with an eighth-inch hole one inch from the bottom of each hook to be screwed into the cylinder. The cylinder also had an eighth-inch hole. Opposite to those two holes, there was an additional set of holes for the servo motor to screw into. The servo motor was ordered from Amazon (https://www.amazon.com/ServoMotorMicro). Using a bandsaw, a two by two-inch piece of eighth-inch plywood was cut to cover the bottom hole of the cylinder full of fish food. This piece of wood was mounted horizontally to the servo motor. The only components that were 3D-printed were the cylinder holding the fish food and the two hooks to mount the automatic fish feeder to the tank.

Methodology

This work was divided into three principle systems: combining the sensors, training the AI model and doing mechanical engineering. An object detection model based on YOLOv8 was developed with Roboflow and put to work on a Raspberry Pi. The zebrafish were tracked by analyzing all the frames to measure how often they were displaced. Ammonia and pH were supervised by a molecular pH indicator and a sensor. The color readings were run through OpenCV routines that spotlighted important thresholds.

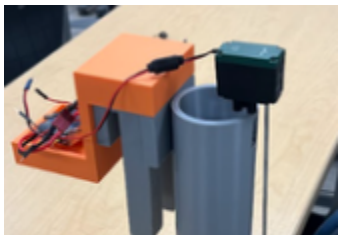
The entire food dropper structure was built in Fusion360 and printed in the Engineering Lab. Only when certain motion thresholds were achieved was food successfully released using a servo motor controlled by Arduino. The RoboSnail actuator was built to fit the commercial RoboSnail cleaner and adjust the torque and positioning to improve its reliability as a button-pusher.

Tests for the full system took place in stages: first with dummy fish and finally with real live animals, so that accuracy could be checked and actuation adjusted. Data obtained from all measurements was kept to monitor performance and plan future improvements.

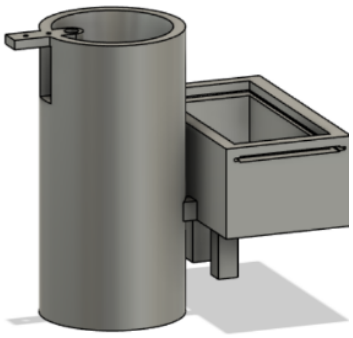
We created the Smart Aquarium by putting together modules that include sensors, computer vision and prototyping. On a Raspberry Pi microcontroller, I set up the system for data gathering, processing and sending out command signals.

Using the YOLOv8 model, the team created a system for recognizing zebrafish movement. We created a dataset of images of zebrafish, labeled them all using Roboflow and added variations to the dataset to deal with false positives. In order to speed up inference, the YOLOv8 model was deployed on a Raspberry Pi using an AI hat. First, images were divided into a grid, bounding boxes were found around the fish and the coordinates (x and y) of each fish's center location were determined. Changes in where individuals were placed were used to lead to conclusions about their emotions such as hunger or stress.

Automated Food Dropper



The food dropper system was CAD-modeled in Fusion360 and underwent multiple design iterations. The first design was too large, as half of it was submerged in our fish tank. Each component was far too big. It also dispensed too much fish food. Our new version had a cylindrical container with an internal rotating lid connected to a servo motor. The servo was programmed via Arduino to spin when prompted by the YOLOv8 system. The motor was mounted on an extruded platform, connected to a battery encased to prevent water exposure. Heat-shrink tubing was used on all soldered wires to waterproof electrical components. A tunnel was extruded inside the cylinder instead of on the outside. We also had to lower the mounting hooks from the original design to be towards the middle of the cylinder that holds the fish food. This ensures no part of the food dropper is in close contact with the water.



Fish Detection & Food Dropper Code

```
# Import necessary libraries
from inference import InferencePipeline
from inference.core.interfaces.stream.sinks import render_boxes
import supervision as sv
import numpy as np
import math

# Confidence threshold to filter out low-confidence detections
CONFIDENCE_THRESHOLD = .99  # Lowered to detect more potential fish

# Track previous positions and distances
previous_centers = []
previous_movement = float('inf')
feed = False
```

```python
def calculate_distance(p1, p2):
    """Calculate Euclidean distance between two points."""
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

# Function to process predictions and print fish locations
def process_predictions(predictions, frame):
    global previous_centers, previous_movement, feed

    detections = sv.Detections.from_inference(predictions)

    if len(detections.xyxy) == 0:
        print("No fish detected in this frame.")
        previous_centers = []
        return render_boxes(predictions, frame)

    current_centers = []
    fish_count = 1
    total_movement = 0

    dets_sorted = sorted(list(zip(detections.confidence,
detections.xyxy)))
    print(dets_sorted)









    for (x_min, y_min, x_max, y_max), confidence in
zip(detections.xyxy, detections.confidence):
        if confidence < CONFIDENCE_THRESHOLD:
            continue
```

```
            width = x_max - x_min
            height = y_max - y_min
            x_center = x_min + width / 2
            y_center = y_min + height / 2
            current_centers.append((x_center, y_center))

            print(f"Fish {fish_count}: Center=({x_center:.2f},
{y_center:.2f}), Width={width:.2f}, Height={height:.2f}")
            fish_count += 1

        # Only compare if we have previous centers with same number of
detections
        if len(current_centers) == len(previous_centers):
            for curr, prev in zip(current_centers, previous_centers):
                total_movement += calculate_distance(curr, prev)

            print(f"Total Movement: {total_movement:.2f}")
            if total_movement < previous_movement:
                feed = True
                print("🐟 Movement decreased. Feed = True")
            else:
                feed = False
                print("➖ Movement did not decrease. Feed = False")

            previous_movement = total_movement
        else:
            print("⚠️ Number of fish changed. Skipping movement
comparison.")
            feed = False
            previous_movement = float('inf')

        previous_centers = current_centers
        return render_boxes(predictions, frame)

# Initialize the inference pipeline
pipeline = InferencePipeline.init(
    model_id="zebrafish_in_aquarium/4",
    video_reference=0,
    on_prediction=process_predictions,
```

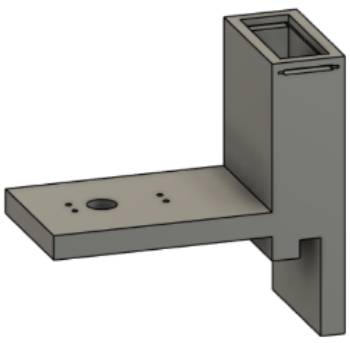
```
    api_key="4VbrdF990dieMBW6yHSR"
)

# Start the real-time detection
pipeline.start()
pipeline.join()
```

The food dropper code initializes a camera to look at the aquarium tank and detects fish using a YOLOv8 model detecting the 5 zebrafish it is most confident about and putting boxes around them to track its displacement. It then starts setting baseline for fish movement in short intervals and compares if movement has decreased and would actuate the food dropper.

RoboSnail Actuation Mechanism

To automate algae cleaning, we ordered a RoboSnail. A RoboSnail is an automatic cleaning robot with wall detection and can clean the walls of the tank when the ON button is pressed. The way the RoboSnail works is that it has two sides, one on the outside of the tank and one on the inside of the tank and the two are connected by magnets. The part on the outside has wheels and will move around the whole tank while the inside part is dragged by the magnet and will clean all the algae and dirt off the walls of the tank. The only problem with ordering this product is that there is no way to customize it. In other words, we couldn't directly link RoboSnail to the main YOLOv8 system. The only way to turn on the RoboSnail was to engineer a servo-driven actuator that was built to press the button of a commercial RoboSnail. Initial designs used lateral servo placement but failed to provide enough torque or alignment. The final version used a 20kg torque servo mounted above the RoboSnail. A custom-fitted shaft was designed to transmit rotational force from the servo to a pressing disc. Wedges were inserted on either side of the mount to stabilize the alignment. Successful button pressing was confirmed through



multiple test runs, completing the autonomous cleaning cycle. CAD schematics were refined to accommodate shaft tolerance and servo-disc pressure alignment.

Sensor Integration

Ammonia levels were monitored using a Seachem Ammonia Alert sensor, which changes color depending on ammonia concentration. A YOLOv8 model trained on the alert's color spectrum (green, yellow, red) identified critical states. Similarly, a pH probe from Atlas Scientific was calibrated using standard buffers and connected to the Raspberry Pi. Custom Python code was written to read analog pH levels via serial interface and activate responses based on thresholds. Real-time readings were used to simulate environmental responses such as alerts or actuation.

```
import cv2
import numpy as np
import time

# Define the region of interest (ROI) where the ammonia alert
is located
ROI = (100, 100, 50, 50)

# Define HSV color ranges
COLOR_RANGES = {
    "yellow": ([75, 5, 160], [110, 40, 200]),          # Safe
    "light_green": ([60, 30, 150], [85, 100, 255]),   # Danger
    "light_blue": ([90, 50, 150], [105, 150, 255]),   # Danger
    "dark_blue": ([105, 100, 50], [125, 255, 150]),   # Danger
}

def detect_color(hsv_roi):
    for color, (lower, upper) in COLOR_RANGES.items():
        lower_np = np.array(lower)
        upper_np = np.array(upper)
        mask = cv2.inRange(hsv_roi, lower_np, upper_np)
        if cv2.countNonZero(mask) > 100:
            return color
    return "unknown"

# Webcam start
cap = cv2.VideoCapture(0)
```

```
    last_print_time = time.time()

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        x, y, w, h = ROI
        roi = frame[y:y+h, x:x+w]
        hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

        # Calculate average HSV value
        avg_hsv = cv2.mean(hsv_roi)[:3]
        avg_hsv_int = tuple(map(int, avg_hsv))

        # Print HSV values every second
        if time.time() - last_print_time >= 1:
            print(f"Average HSV in ROI: H={avg_hsv_int[0]},
S={avg_hsv_int[1]}, V={avg_hsv_int[2]}")
            last_print_time = time.time()

        # Detect color
        detected_color = detect_color(hsv_roi)
        if detected_color == "yellow":
            label = "SAFE: Yellow"
            color = (0, 255, 255)
        elif detected_color in COLOR_RANGES:  # any known danger
color
            label = f"DANGER: {detected_color.replace('_', '
').title()}"
            color = (0, 0, 255)
        else:
            label = "DANGER: Unknown"
            color = (0, 0, 255)

        # Draw rectangle and label
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.putText(frame, label, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)
```

```
            # Draw small preview of average HSV color
            avg_color_bgr = cv2.cvtColor(np.uint8([[avg_hsv_int]]),
cv2.COLOR_HSV2BGR)[0][0]
            cv2.rectangle(frame, (10, 10), (60, 60), tuple(int(c) for
c in avg_color_bgr), -1)
            cv2.putText(frame, "HSV", (10, 75),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

            # Display frame
            cv2.imshow("Ammonia Color Detection", frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    cap.release()
    cv2.destroyAllWindows()
```

The code would be looking at an Ammonia Alert attached to the side of a tank, and puts a box around its region of interest and assures the color within it is yellow, as yellow mean ammonia levels are in a safe range, otherwise its sends a warning message that's it not yellow and would send out the robosnail.

Results

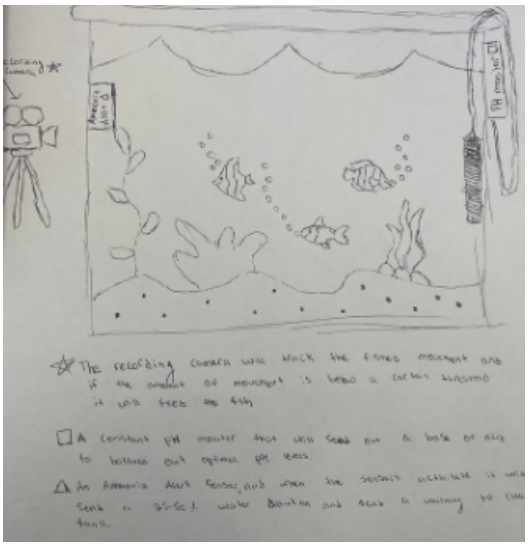
System testing was done in several phases, first using components and then ready system integration. Under the best conditions, YOLOv8 was able to detect in real-time with an accuracy of over 90% when placed optimally in front of a subject. Data from the movement was used to operate the food dropper and its design was improved several times with help from Fusion360. The mechanism at the feeder was very small, sealed so no moisture entered and released just the right amount each time it was triggered. Color detection in the ammonia output was used to set a threshold for alerts which worked correctly. The Raspberry Pi displayed live results of the pH measurements. We put together and tested the RoboSnail actuator; the arm moved to press the cleaning button when it was activated by the servo at the proper torque. All in all, the system achieved the results it was meant to achieve.

Discussion

In the aquatic experiment, the Smart Aquarium combined sensors and machine learning in real life. As a result, this approach could help care for fish proactively rather than reactively which made the system unlike other commercial ones. Although the YOLOv8 model did very well, its accuracy dropped when the light was weak or when fish were hiding behind tank objects. The system was good at monitoring, but including more accurate temperature and oxygen sensors might make analysis more complete. Making the actuator system work in the EV involved solving difficult problems because of high torque needs and waterproofing concerns. It took many efforts before we were able to make RoboSnail's movements successful. An important insight was that using modules allowed each section to be checked, upgraded and swapped out individually. Movements in stress patterns could be recognized by fish behavior in future improvements.

Conclusion

The program proved that real-time, AI-assisted management of an aquatic environment is workable. As a result of using computer vision, colorimetric sensors, actuators and embedded control systems, the team designed a working prototype that significantly supports both animal welfare and user needs. More work in this area could focus on storing data on the cloud, using AI for stress spotting in fish and including temperature control and automated injection of chemicals into their tanks. Using data from zoos could actively help improve aquatic systems or businesses like public aquariums and aquaculture farms.



Limitations

One major problem we faced was staying within the time specified for the project. Another of the main difficulties was that we could not use a Raspberry Pi or other microcontroller that would handle the sensor data and automate responses according to

programmed levels. As a result, we could not implement the full responsive control system. We also worked on a system that could automatically add water when ammonia levels rose, but it turned out to be hard to combine with the rest of our machine. A further problem was discovered with pH regulation — we could maintain accurate pH readings, but did not develop a smooth system for dispensing pH altering chemicals or peat moss.

Appendix

Appendix A: Materials List

A table listing all major components:

Component	Quantity	Description / Notes
Raspberry Pi 5	1	Microcontroller to handle YOLO and sensor data
Atlas pH Probe	1	pH sensor connected via serial to Pi
Seachem Ammonia Alert	1	Colorimetric ammonia sensor attached to tank
Servo Motor	2	One for food dropper, one for RoboSnail actuation
Zebrafish	5	Used to test fish movement and feeding detection
RoboSnail	1	Commercial magnetic algae scrubber

Appendix B: Smart Aquarium Timeline (August 2024 – May 2025)

This appendix provides a condensed log of major milestones and engineering progress for the Smart Aquarium project

---

August 27, 2024
Project officially begins. Decided to build a Smart Aquarium integrating YOLO object detection and environmental sensors. Initial focus was on researching key sensors (pH, ammonia, temperature) and how to use a Raspberry Pi for data processing and response.

September 12, 2024
Sketched the first prototype of the aquarium system. Identified core inputs and outputs:

pH level, ammonia alert, and fish activity. Explored how to track fish movement using YOLOv8 and activate actuators based on thresholds.

September 24, 2024
Deep dive into zebrafish care requirements, tank design, and environmental ranges. Finalized fish species (zebrafish), tank size (20 gallons), and target pH/temp values. Began planning automated feeding and cleaning systems.

October 22, 2024
Designed fish activity tracking system based on movement detection over short time windows. Explored displacement logic using center coordinates of fish bounding boxes from YOLO.

November 12, 2024
Finalized the plan to use a color-based Ammonia Alert sensor and YOLO color detection instead of numerical sensors due to cost and complexity. Also started sourcing a dropper and talking with the engineering lab for custom mounts.

December 16, 2024
Received core components: Raspberry Pi, UVC cams, Atlas Scientific pH probe. Began testing YOLOv8 inference pipeline with camera integration. Calibrated pH probe.

January 21, 2025
Connected pH probe to Raspberry Pi. Continued development of YOLO-based ammonia color detection. Ran experiments to ensure stable serial communication for real-time alerts.

February 25, 2025
Filled the tank, removed chlorine, and jumpstarted the nitrogen cycle using blood worms and Quick Start. The tank is now prepped for live fish.

March 4, 2025
Improved YOLO model to track displacement of zebrafish more precisely. Began coding logic to actuate the food dropper if movement dropped below baseline.

March 20, 2025
Implemented center-point and confidence threshold logic in YOLO detection to improve accuracy and reduce false positives. Movement comparison logic complete.

April 10, 2025
Reserved zebrafish at PetSmart, recalibrated pH probe, tested servo actuation using Arduino and Raspberry Pi. Worked with partners on servo integration for dropper and RoboSnail.

April 22, 2025
Bought and introduced zebrafish into the tank. Began real-world testing of ammonia alert color detection and refined thresholds. Used bloodworms and Aqua Start to stabilize the environment.

April 29, 2025
Final testing of fish feeding response using displacement-based thresholds. Observed reduced fish movement leading to successful servo-triggered food drops. Completed integration of ammonia and pH systems with visual feedback.

Works Cited

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

Henry, J. Q., Lesoway, M. P., & Perry, K. J. (2020). An automated aquatic rack system for rearing marine invertebrates. *BMC Biology, 18*(1). https://doi.org/10.1186/s12915-020-00772-w

Li, D., Xu, X., Li, Z., Wang, T., & Wang, C. (2020). Detection methods of ammonia nitrogen in water: A review. *TrAC Trends in Analytical Chemistry, 127*, 115890. https://doi.org/10.1016/j.trac.2020.115890

Vonau, W., & Guth, U. (2006). PH monitoring: A review. *Journal of Solid State Electrochemistry, 10*(9), 746–752. https://doi.org/10.1007/s10008-006-0120-4

Wu, T.-H., Wang, T.-W., & Liu, Y.-Q. (2021). Real-time vehicle and distance detection based on improved YOLO v5 network. *2021 3rd World Symposium on Artificial Intelligence (WSAI)*, 24–28. https://doi.org/10.1109/WSAI51899.2021.9486316

Smith, E., Wright, A., & Morsy, O. (2021). Evaluation of fish feeder manufactured from local raw materials. *Scientific Reports*, *11*(1). https://doi.org/10.1038/s41598-021-98383-0

Jones, B. (2022, August 26). *The pros and cons of using an auto feeder in your aquarium*. Natural Environment Aquatix. https://naturalenvironmentaquatix.com/blogs/the-fish-tank-blog/the-pros-and-cons-of-using-an-auto-feeder-in-your-aquarium

Miller, F., Patel, N., (2021). *Aquarium fish health: Overfeeding fish: A common problem and possible solutions*. (n.d.). https://www.liveaquaria.com/article/353/?aid=353