A Smart Aquatic Control Platform

Jingwei Dai
Department of Computer Science
Earlham College
Richmond IN, USA
adai13@earlham.edu

May, 2024

Abstract

The marine aquarium industry has been exploring IoT (Internet of Things) technologies to automate maintenance, yet existing solutions often rely on proprietary Wi-Fi or Bluetooth implementations, hindering the development of reliable hardware. Moreover, these solutions operate within isolated ecosystems, posing integration challenges with established Smart Home platforms like Apples HomeKit.

This project explores the latest advancements in IoT connectivity standards, particularly focusing on Thread and its open-source counterpart, OpenThread. Our objective is to establish a Platform leveraging OpenThread that empowers third-party IoT manufacturers to easily build Thread-enabled products for specific applications within large-scale artificial environments, such as aviaries and marine aquariums. In addition, we developed a hardware prototype to showcase distinct advantages of a connected aquatic control system over traditional aquarium setups. Overall, the Platform means to highlight an integrated experience where a focus on ease of access and onboarding, both through the adoption of Thread and HomeKit, enhances the user experience and drives the development of animal-care technologies.

Keywords

SwiftUI, HomeKit, Beaglebone Black, Thread, OpenThread

1 Introduction

The current smart home landscape is dominated by three major platforms: Amazon Alexa, Apple HomeKit, and Google Home, all backed by their respective smart assistants. These define the quintessential frontend user experience, because they are what users directly interact with to control their smart homes. Throughout the 2010s, choosing which platform to stick with was an important decision. The platforms were not interoperable: Alexa, Siri, and Google Assistant spoke totally different languages under the hood, and most smart home products could only be connected to one platform at a time. Over the last few years, tech giants have come together to remedy the situation (Al-Qaseemi et al., 2016). The result is Matter, an open-source interoperability standard that vows to unite the world of IoT (Hill, 2022). All three platforms have signed on to fully integrate the Matter application layer in the next few years, which means they will eventually sport the same backend with the only difference being UI/UX. When the transition completes, smart home devices conforming to Matter will work under all platforms without extra tinkering. The eventual engineering product of this project will be fully compatible the Apple HomeKit platform, which already runs on a backend that fully supports Matter (Apple Inc., 2022).

The platforms, including those that have transitioned to Matter, support multiple connection protocols through which devices talk to the platform backend and each other. As of 2023, there are half a dozen popular protocols in the IoT industry (Pradeep et al., 2016). Major IoT manufacturers Philips and Aqara use ZigBee as the protocol for their entire line of smart home devices (Signify, 2023). ZigBee devices do not talk to the platforms directly. They can only function when a hub is connected to the internet router to act as a relay for communication between themselves and the platform backend, but hubs from some manufacturers are not compatible with competitors' devices. For example, Aqara's ZigBee implementation veers off the standard by an enough amount that third party devices cannot connect to Aqara's hubs.

However, standardization through Matter may solve this fragmentation of technology in the IoT industry. Matter recommends Thread, a relatively new protocol that consolidates the advantages of the most popular connection protocols. With the inclusion of a Thread radio in the iPhone 15 (Apple Inc., 2023), we are seeing major effort from industry leaders to promote and facilitate the adoption of this technology.

By employing Thread as the foundational technology, we differentiate ourselves from comparable platforms in

the market. Thread provides a robust infrastructure with low-latency, low-bandwidth, and low-power connectivity, establishing a mesh network among connected devices. This enables swift responses to user inputs and ensures platform scalability without compromising performance.

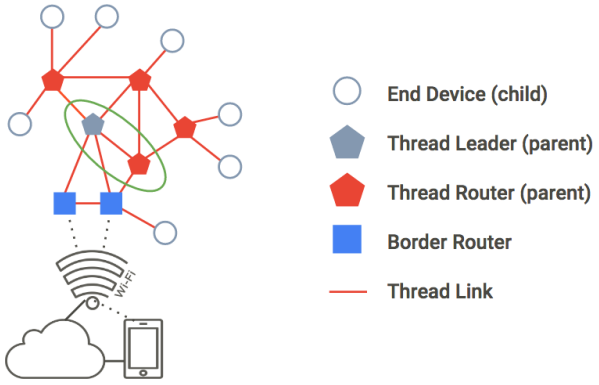

Figure 1: Thread Topology (Bumgardner, 2024)

Additionally, Thread devices boasts of a fallback mechanism called Border Routers assumable by all non-battery powered devices. Thread intelligently selects devices to serve as Border Routers, each enabling bidirectional connectivity between the mesh network and the internet. This approach significantly reduces the risk of system-wide shutdown caused by a single device failure. This contrasts with traditional low-latency smart home platforms that rely on a dedicated hub, which introduces a single point of failure capable of crippling the entire system.

Our vision for the Platform extends to serving systems of all sizes, ranging from home reef tanks to industrial coral farms and public aquariums. The integration of remote-controllable devices that establish a scalable mesh network among themselves greatly facilitates the management of these diverse systems, while Thread's fallback mechanism protects artificial ecosystems where stability is of the utmost importance.

In the aquarium industry, a frequent frustration among users of smart devices is the necessity to navigate through the manufacturer's app for even basic tasks. We offer a solution by exposing commonly-used device settings through HomeKit and by giving users the ability to customize which settings are most important to them. This enables users to, for instance, instruct Siri with a single command to activate the spectrum that enhances the color of corals right before the guests' arrival, rather than adjusting various sliders manually. HomeKit is integrated into all iOS devices, ensuring seamless accessibility.

2 Project Goal

In this project phase, our aim is to demonstrate the enhanced user experience and effectiveness in aquarium maintenance through an integrated aquatic control system. We've developed an iOS app and a hardware demo utilizing OpenThread's IoT Debian OS. Leveraging the Beaglebone's sensor capabilities, we showcase the potential of a smart home platform within the aquarium industry.

3 Components

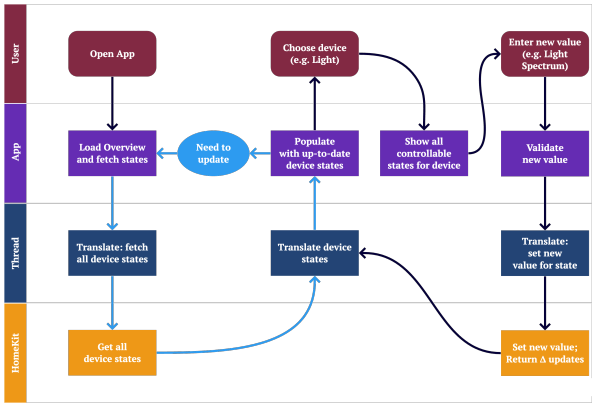

Figure 2: Data Architecture Diagram

The Platform comprises three interdependent components that complement one another. Our demo highlights two components with which users directly interact: Software, represented by the first two rows in Figure 2, and Hardware. The third component, Backend, comprises of an API that allows third-party hardware manufacturers to build firmware that integrates their hardware into the Platform. This API will be uploaded to GitLab (Dai, 2024) at a later date.

3.1 Software

Through our software, users may inspect and customize the behavior of all connected devices. We created an iOS app in SwiftUI that showcases one of the most important device categories in a reef aquarium: lighting. Screenshotted in Figure 3 is our demo app through which granular control over each group of diodes in a reef light is achieved. In this demo, we're simulating the Ecotech Radion XR30 G6 Pro reef light. For this purpose, we are primarily addressing the following 90 diodes: 4 395nm (ultraviolet), 2 405nm, 4 415nm, 12 430nm, 32 450nm, 32 465nm (blue), 2 510nm (green), and 2 645nm (red).

3.1.1 Diode Configuration

Each slider in Figure 3 represents the intensity setting for a group of diodes emitting the same wavelength of light. For example, at intensity 39%, all ultraviolet diodes would be set to 39% power. We then convert all wavelengths into their approximate RGB counterparts

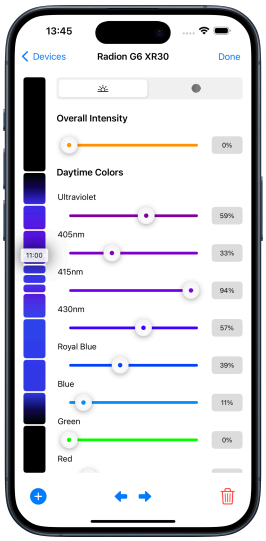

Figure 3: Diode Configuration

and mix all the RGBs based on their intensity and diode count.

Wavelength to RGB

While the human eye typically discerns wavelengths between approximately 400nm to 720nm, our use case incorporates extremities such as 395nm UVA, which is crucial for coral health and growth. To accommodate these variations, we define the permissible range as 380nm to 780nm.

We then map the wavelengths to their corresponding RGB values by employing a multistep formula then scaling the results so that they fall within $[0, 255]$. For example, $\lambda = 395nm$ would yield an RGB value of (92, 0, 214). The complete mapping and conversion algorithm is hosted on GitLab (Dai, 2024).

Timeline

On the left side of the screen, we feature a Timeline displaying breakpoints corresponding to each instance when a Diode Configuration is set. For example, Figure 3 shows the diode configuration for the 09:30 breakpoint. The Timeline provides a dynamic visualization of the expected lighting effects throughout the day. With the gradient, users can quickly discern when the light activates or deactivates, as well as when the aquarium is expected to maximize its actinic, fluorescent glory.

Combining RGBs

To combine colors, we employ a simple algorithm that also takes into account the alpha, alternatively called weight, of each diode group. This means the 4 UVA/395nm diodes do not affect the combined color as much as the 32 blue/465nm diodes.

For example, to combine ultraviolet and blue, we start by converting 395nm and 465nm to their RGB values: (92, 0, 214) and (0, 178, 255). Assuming they both have an alpha/weight of 1.0, the following calculations give us a single combined color to showcase for this breakpoint on the Timeline.

$$r = (92 + 0)/2 = 46$$
$$g = (0 + 178)/2 = 89$$
$$b = (214 + 255)/2 = 234$$

3.1.2 Spectrum Graph

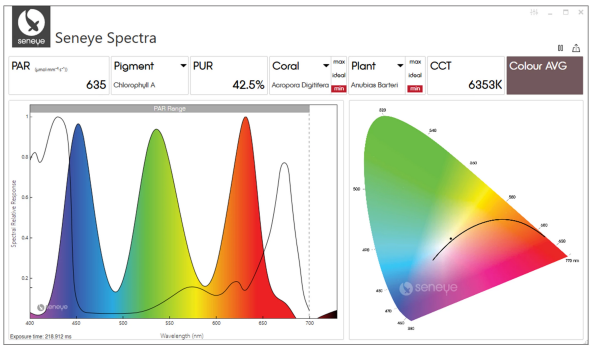

Figure 4: Seneye Spectra software on Windows

Professional spectrometer software typically utilizes spectrum graphs, as illustrated in Figure 4 (Seneye, 2024), to aid users in visualizing spectra within particular areas of an aquarium. Such hardware is often necessary due to the varying wavelength distributions resulting from different spatial configurations of the same multi-fixture light source. Efforts are frequently made to arrange lights to achieve the most uniform spread, as this enables aquarists to maximize the horizontal space available for coral species of similar lighting needs.

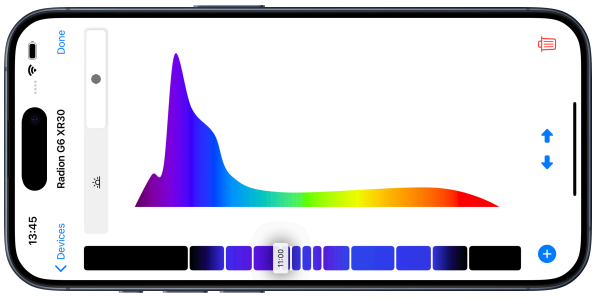

Figure 5: Spectrum Graph

The Spectrum Graph can be generated for any diode configurations. We congregate each diode group and plot their intensities on a 2D plane where the x-axis denotes wavelength and the y-axis denotes the weighted intensity value (see Figure 5).

To draw spectra correctly, we studied multiple interpolation methods and compared their results to popular renders of spectra. We found that we could draw the

most conventional spectra by using cubic spline interpolation to fill in intermediary data points. The Cubic Spline interpolation could be expressed mathematically as such:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

While the Spectrum Graph is not intended to replace dedicated spectrometers for experienced reefers, we believe it offers valuable insight. Providing an estimate of the spectrum output across the effective coverage area of the light source can assist reefers in assessing the efficacy of their diode configurations. The complete interpolation algorithm is hosted on GitLab (Dai, 2024).

3.2 Hardware

We use a Beaglebone Black to showcase components of our software platform that interact and respond to the environment. The Beaglebone runs on an IoT-flavored (GUI-less) Debian OS (OpenThread, 2024), with a photoresistor connected as per the Adafruit guide (Adafruit, 2024). Our iOS app regularly retrieves data from the photoresistor, leveraging its voltage readings to assess ambient light intensity and suggest optimal diode configurations. However, this application represents just one example of the myriad uses for environmental sensors within a reef aquarium.

Example: Temperature Sensor

Temperature sensors play a crucial role in regulating aquatic systems, particularly in maintaining consistency within a reef aquarium. It's essential for reefers to minimize temperature fluctuations, as stability is paramount for the health of the ecosystem. Common solutions include aquarium heaters/chillers or room AC units, but such hardware are prone to failure, especially as they are often under heavy use. To mitigate this risk, reefers often implement a failsafe mechanism by integrating an additional temperature sensor into the control system. This sensor monitors ambient temperature and prompts the system to deactivate the heater/chiller if a significant deviation is detected. This often requires multiple temperature sensors for multiple pieces of hardware, and it is easy to configure incorrectly.

In our integrated Platform, no extra hardware or wiring is necessary. Users can set up Rules that respond to environmental changes automatically. For instance, if there's a notable increase in room temperature, the system triggers an alert to notify the user. The Platform can then activate backup chillers or AC units while continuously monitoring water temperature to make informed adjustments. By primarily relying on software, users are spared from dealing with excessive wiring, tinkering, and maintenance.

3.2.1 Photoresistor

We follow the Adafruit tutorial to correctly wire the photoresistor (Figure 6).

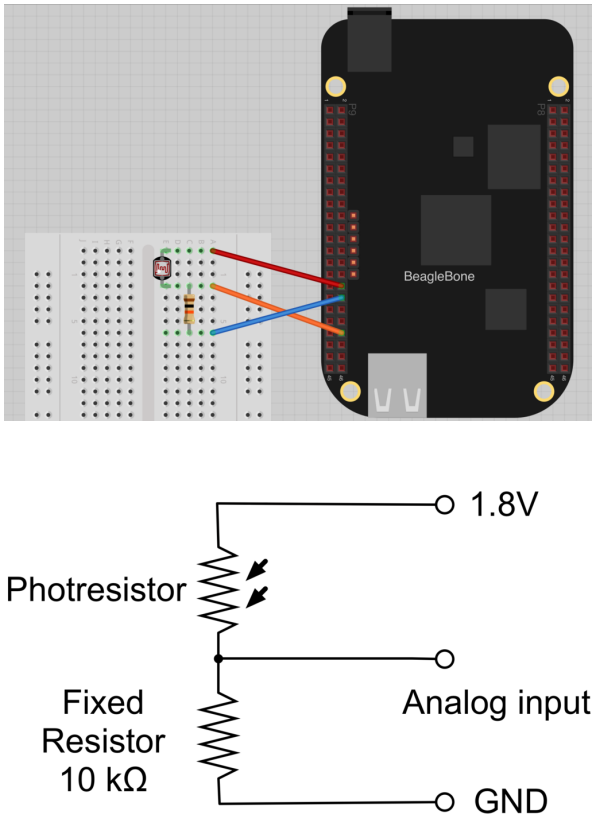

Figure 6: Beaglebone Wiring

Reading from the photoresistor is simple. The `Adafruit_BBIO` library enables access to the GPIO pins on the Beaglebone, acquiring sensor outputs that can be roughly converted to voltage using the following formula:

$$volts = reading * 1.800$$

The iOS app begins by guiding users to move the sensor inside and around the aquarium to establish a baseline reading of the system. It records this initial reading and can use it as a reference for future spectrum recommendations. Once initialized, whenever reading mode is engaged, the app fetches readings every few seconds and proposes adjustments to diode intensity. The app is also primed to take advantage of more advanced light sensors that provide more precise lumen or PAR data to enable spectrum customizations, should users opt to incorporate additional sensors.

3.2.2 Website

The Beaglebone hosts a server through Flask/Python. The server's homepage displays the current diode configuration of the device (Figure 7) and has three routes.

Radion G6 XR30

Time	Diode ID	Wavelength	Count	Description	Intensity	Alpha
21600	ab7532bb-6820-4ecf-8d07-4579e2291572	395	4	Ultraviolet	0.134	0.3
	20eb4ed8-3a02-4a0b-a65e-25f3c11db5e9	405	2	405nm	0.614	0.2
	56e33a28-c59b-48f5-a08f-d3ac6abbd72d	415	4	415nm	0.45	0.3
	66c7ec49-0404-4535-8815-d8a7f225f6a4	430	12	430nm	0.781	0.6
	167c41d0-ef31-4142-afab-3947d640b8ec	450	32	Royal Blue	0.872	0.9
	e2e6a663-5fe3-4ee7-a9a1-cbb58c90c175	465	32	Blue	0.387	0.8
	0cc5b236-c54d-459f-8bdf-fb5d5e3ec9e9	510	2	Green	0	0.3
	2bdc07f1-c46d-4a68-a596-48b2b1614c9b	645	2	Red	0.768	0.2
28800	a5a17d5b-5a5b-4b1d-b4a5-6b885c227b68	395	4	Ultraviolet	0.097	0.3
	dedb33ae-61d2-40b5-80cb-ec5450d99043	405	2	405nm	0.723	0.2
	9c9357b1-189e-4b18-a29e-977dc7d9c75f	415	4	415nm	0.817	0.3
	4ad244d5-8e54-4b29-b2c5-22eb8657ec79	430	12	430nm	0.252	0.6
	3fa2b8e7-68c2-4fc8-84b1-2f7a93b29552	450	32	Royal Blue	0.528	0.9
	5b0d5b3b-f5d7-41d2-a124-61ec4359d4b7	465	32	Blue	0.877	0.8
	b0faa13b-af7e-409d-82b1-e200986e58cb	510	2	Green	0.691	0.3
	10abce6a-5dd5-46fb-9b86-d868700bce06	645	2	Red	0.651	0.2
34200	4c9f3e5c-b915-4645-9942-8c5e617518b2	395	4	Ultraviolet	0.243	0.3
	835e6f7e-9c48-4c6c-a5a0-9cb1c97ab3b9	405	2	405nm	0.67	0.2
	efea2555-f6c0-4895-a084-2b19e31c1c1e	415	4	415nm	0.423	0.3
	98f2c94e-10a2-48cb-80d4-17b7e0dc091b	430	12	430nm	0.981	0.6
	100e5125-fcfb-44a2-bf29-142f9fe1b23e	450	32	Royal Blue	0.789	0.9
	de64f6c4-2bcb-42e9-8988-b14c1b6341f2	465	32	Blue	0.171	0.8
	baaa2085-15a5-4f0f-81c3-6499ef1085bb	510	2	Green	0.336	0.3
	64c3b7c0-52f7-438f-a273-3a4ba0ed6df4	645	2	Red	0.188	0.2

Figure 7: Beaglebone Server

The first two routes, `get_data` and `update_data`, allow the iOS app to read and update diode configurations. The third route, `get_reading`, shares the photoresistor's output with our software. After enabling Internet Sharing with the Beaglebone[1], we can access the server at `beaglebone.local:8080`. Future firmware versions to enable more functionalities should still follow the same three-route principle.

4 Conclusion

Through our software/hardware demo, we illustrate the journey of a hobbyist embarking on the setup of their aquarium. With the aid of an integrated smart system, the hobbyist navigates the initial setup phase with confidence, sidestepping potential pitfalls that could disrupt the delicate balance of their burgeoning aquatic ecosystem. Real-time sensor data empower users to make informed decisions, enabling them to make timely adjustments and optimize conditions for aquatic life.

As the system accumulates data over time, it evolves to be capable of autonomously managing various aspects of aquarium maintenance. This autonomy not only streamlines the caretaking process but also grants hobbyists the freedom to delve deeper into the mesmerizing intricacies of marine life.

Expanding this out to research to allow for more efficient reef studies in labs Automation allows for better biological studies Intricacies of marine life

[1]On macOS, enable Networking Sharing with the USB-connected Beaglebone, then run `sudo dhclient eth0` to enable internet (Solarian Programmer, 2018)

5 Future Work

Potential enhancements could involve expanding the range of hardware categories supported, including wavemakers, peristaltic dosers, etc.

To fully showcase the advantages of OpenThread over alternative connectivity standards, additional hardware are necessary. For example, by incorporating a broader array of sensors, we can better highlight the capabilities of OpenThread within our integrated aquatic control system.

We hope to expand the project scope to enable more research on marine life. Autocollection of environment data

6 Acknowledgements

I would like to thank the esteemed faculty of the Computer Science department at Earlham College and friends who have supported me. Specifically, I want to thank Dr. Charlie Peck for his hardware expertise and general guidance, Dr. David Barbella for his feedback on my ideas and research proposal, Porter Libby for his help in brainstorming effective implementations of those ideas, and Gary Coker for proofreading and thoughtful discussions when bouncing ideas around.

References

Adafruit. (2024). *Measuring light with a beaglebone black.* Retrieved 2023, from `https://learn.adafruit.com/measuring-light-with-a-beaglebone-black/wiring`

Al-Qaseemi, S. A., Almulhim, H. A., Almulhim, M. F., & Chaudhry, S. R. (2016). Iot architecture challenges and issues: Lack of standardization. In *2016 future technologies conference (ftc)* (p. 731-738). doi: 10.1109/FTC.2016.7821686

Apple Inc. (2022). *Matter support in ios 16.* Retrieved from `https://developer.apple.com/apple-home/matter/`

Apple Inc. (2023). *Apple unveils iphone 15 pro and iphone 15 pro max.* Retrieved from `https://www.apple.com/newsroom/2023/09/apple-unveils-iphone-15-pro-and-iphone-15-pro-max/`

Bumgardner, J. (2024). *Simulating a thread network with openthread.* Retrieved from `https://openthread.io/codelabs/openthread-simulation-posix`

Dai, J. (2024). *Aquarium control in swiftui.* Retrieved 2024, from `https://code.cs.earlham.edu/adai13/aquatic-control-system`

Hill, S. (2022). *Here's what the 'matter' smart home standard is all about.* Retrieved from `https://www.wired.com/story/what-is-matter`

OpenThread. (2024). *Openthread border router on beaglebone black.* Retrieved April, 2024, from `https://openthread.io/guides/border-router/beaglebone-black`

Pradeep, S., Kousalya, T., Suresh, K. A., & Edwin, J. (2016). Iot and its connectivity challenges in smart home. *International Research Journal of Engineering and Technology (IRJET)*, *3*, 1040–1043.

Seneye. (2024). *Seneye spectra.* Retrieved 2024, from `https://spectra.seneye.com`

Signify. (2023). *Zigbee 3.0 support in hue ecosystem.* Retrieved from `https://developers.meethue.com/zigbee-3-0-support-in-hue-ecosystem`

Solarian Programmer. (2018). *Beaglebone black sharing usb internet to macos.* Retrieved from `https://solarianprogrammer.com/2018/12/04/beaglebone-black-sharing-usb-internet-macos/`