UNIVERSIDADE FEDERAL DO ABC

ENGENHARIA DE INSTRUMENTAÇÃO, AUTOMAÇÃO E ROBÓTICA

RENAN TADEU BALDINI DE BRITO

THYAGO NETTO BALTAZAR

SISTEMA DE CONTROLE E MONITORAMENTO DE AQUÁRIO MARINHO

SANTO ANDRÉ - SP

2020

RENAN TADEU BALDINI DE BRITO

THYAGO NETTO BALTAZAR

SISTEMA DE CONTROLE E MONITORAMENTO DE AQUÁRIO MARINHO

Monografia apresentada ao curso de Graduação em Engenharia de Instrumentação, Automação e Robótica como parte dos requisitos para obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Filipe Ieda Fazanaro

SANTO ANDRÉ - SP

2020

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus, que nos concedeu os dons necessários para chegarmos até aqui, aos nossos familiares e amigos que nos apoiaram e ajudaram, aos nossos professores, em especial nosso orientador, e a todos que estiveram envolvidos e empenhados em nos ajudar direta ou indiretamente.

“All we have to decide is what to do with the time that is given us.”

J.R.R. Tolkien

RESUMO

O crescimento do mercado brasileiro de aquários marinhos somado à complexidade e elevado custo de manutenção da vida neste ambiente implica em diversos problemas para os aquaristas nacionais. Diversos parâmetros como temperatura, densidade, pH, nível e luminosidade precisam ser controlados regularmente, pois alterações abruptas podem causar desequilíbrio e consequente perda de animais e/ou desenvolvimento de doenças contagiosas que condenam todo o tanque. Se faz necessário, portanto, a aferição periódica de tais parâmetros e eventual correção, tarefa que se torna bastante complexa dada a quantidade de processos, equipamentos e kits químicos necessários, consumindo tempo, atenção e, algumas vezes, dinheiro. Neste sentido, a proposta deste projeto é a construção de um sistema unificado para controle e monitoramento dos parâmetros de um aquário marinho somado ao desenvolvimento de um aplicativo para smartphone que informe a situação atual de cada parâmetro ao aquarista, reduzindo o tempo gasto para manutenção do aquário, facilitando a mão de obra do aquarista e assegurando um ambiente mais estável.

Palavras-chave: controle, monitoramento, aquário marinho, parâmetros, ESP32, aplicativo.

ABSTRACT

The increasing on the Brazilian market of marine aquariums added to the complexity and high maintenance cost of life on this environment implies in diverse problems to the national aquarists. Diverse parameters as temperature, density, pH, level and lightness must be regularly controlled, because abrupt changes can cause instability and consequent loss of animals and/or the development of contagious diseases that condemn the entire tank. It's necessary, therefore, the periodic admeasurement of those parameters and eventual correction, task that becomes too much complex, due to the quantity of processes, equipments and chemical kits needed, spending time, attention and, sometimes, money. On this way, the purpose of this project is the building of an unified system to the control and monitoring of the marine aquarium parameters, added to the development of an app to smartphone that informs the actual situation of each parameter to the aquarist, reducing the spent time to the maintenance of the aquarium, making the work easier to be done by the aquarist and ensuring a more stable environment.

Key-words: control, monitoring, marine aquarium, parameters, ESP32, app.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização

O aquarismo marinho tem conquistado espaço no mercado nacional nos últimos anos e adquirido um número cada vez maior de apreciadores. O colorido dos corais, que misturam tons vivos e efeito neon à incidência de luz azul; peixes dos mais diversos tamanhos, formas e comportamentos; crustáceos e equinodermos e ainda uma infinidade de outras formas de vida, para todos os gostos e apreciações, oferecem um hobby vasto e fascinante que atrai a atenção de todos, particularmente das crianças, e estimula o aumento do número de aquaristas amadores.

Historicamente, indícios apontam que as primeiras formas de aquários surgiram ainda no Egito antigo, sendo que é apenas a partir do século XX que o aquarismo começou a se tornar mais popular, principalmente em razão do transporte aéreo que possibilitou o transporte dos espécimes de um lugar para o outro[1,2]. No entanto, esta atividade não está apenas ligada ao lazer, mas também ao estoque de alimentos, ao tratamento terapêutico e ao estímulo à educação ambiental. Estudos neste sentido tem alcançado conclusões animadoras como segue:

> Através do diálogo com os docentes da educação básica, verificamos que os simulacros de ecossistemas, como aquários e/ou terrários são excelentes ferramentas didáticas a serem exploradas no ensino dos conteúdos científicos, por professores de Ciências e Biologia, uma vez que, despertam o interesse, a curiosidade e a motivação dos estudantes para aprender, mediante o contato com o vivo e com a simulação de ambientes reais. Além disso, essas ferramentas de ensino permitem ao aluno ampliar conhecimentos prévios, bem como, observar, analisar, discutir e interpretar processos e fenômenos biológicos. [3]

> Vale ressaltar que muitas terapias foram realizadas próximas ao aquário e que com isso os pacientes desempenharam com mais entusiasmo as atividades propostas. As fisioterapeutas atribuem este fato à proximidade com os peixes, tornando a terapia mais enriquecida e interessante.
>
> Justamente por ser um local em que se atende uma população onde há atrasos no desenvolvimento sensórios motores, déficits cognitivos e dificuldades de interação social, o aquário mostrou-se um excelente instrumento para o desenvolvimento de atividades que buscam estimular esses déficits, fugindo do tratamento convencional. [4]

Seja puramente pelo lazer, pelo estudo, pelo ensino, pela terapia ou por outros tantos benefícios que o aquarismo pode oferecer, a demanda por aquários tem aumentado não só em quantidade, mas em qualidade, muitos espécimes que não se adaptavam ao cativeiro hoje já

estão disponíveis em qualquer loja graças ao intenso estudo comportamental e adequação dos parâmetros do cativeiro às condições naturais. Aquários podem ser divididos segundo os espécimes que se pretende criar nele, sendo assim, temos aquários de água doce, aquários marinhos, aquários plantados, aquários de recifes e ainda, subdivisões destes com maior ou menor diversidade. Cada um destes ambientes exige cuidados especiais e parâmetros muito bem regulados para a manutenção e desenvolvimento da vida em seu interior. O aquário marinho, neste sentido é, provavelmente, o mais sensível e mais complexo, pois exige um controle rígido de múltiplos parâmetros, como densidade da água, pH, kH, temperatura, níveis de magnésio, cálcio, amônia, nitrito e nitrato, quantidade de luz incidente, sistema de filtragem de impurezas e ainda outros que tornam a tarefa difícil, especialmente para amadores, e extremamente trabalhosa mesmo para os mais experientes.

Neste cenário, a automatização de sistemas de medição e controle de parâmetros se torna fundamental para o sucesso do aquarista e de seu objetivo, como a regulagem da temperatura por aquecedores e coolers que já conta com uma automação de componentes, porém desprovida de um controle integrado. Faz-se necessário, portanto, uma automação mais efetiva e mais abrangente que possibilite não só um melhor controle de cada parâmetro, mas que também centralize e integre este controle, tornando a manutenção do ambiente mais fácil e mais prática para o aquarista.

1.2. O Aquário Marinho

Os aquários marinhos podem ser classificados de acordo com o seu volume em macro ou nano, no entanto, esta classificação não é bem definida entre aquaristas e a faixa que determina, em litros, se um aquário é nano ou macro varia muito. Desta forma, o uso mais comum e mais assertivo de se classificar o aquário é por meio da litragem e do tipo de espécime criado, por exemplo, um aquário de 70 litros “reef only”, designa um tanque com capacidade de 70 litros e que não possui peixes, apenas corais, rochas, pequenos crustáceos e algum outro espécime de interesse do aquarista, é possível também a classificação como “fish only” que designa um aquário sem corais, apenas com peixes, crustáceos, rochas, etc.

Inicialmente ao se montar um aquário marinho deve-se preparar a água adicionando sais sintéticos específicos (contendo sódio, potássio, magnésio, cálcio, etc.) em água. Estes sais podem ser encontrados em lojas de aquário e possibilitam que o próprio aquarista prepare a

água da forma que mais lhe parecer pertinente ao estado do aquário, ajustando densidade, temperatura e qualidade da água que alguns preferem utilizar água comum da torneira com produtos que retiram cloro, cloramina e etc., outros preferem utilizar água deionizada e há ainda relatos de aquaristas que buscam água diretamente do mar para realizar suas trocas. Os fabricantes determinam entre 30 e 35g de sal por litro, para se obter a densidade ideal de 1,024 kg/L, a qual é possível medir com densímetro ou refratômetro. É de suma importância para a manutenção da vida em um aquário marinho que as chamadas TPAs (Trocas Parciais de Água) ocorram em intervalos não superiores à 30 dias, sendo estas trocas de 20% do volume do aquário em caso de intervalos de 30 dias e entre 10% - 15% do volume em caso de intervalos de 15 dias. As TPAs são responsáveis por garantir a renovação da água, que ajuda a restaurar os parâmetros necessários como Magnésio, Cálcio, pH e kH e equilibra o ambiente, reduzindo elevações de parâmetros indesejados.

Além disso o aquário deve conter o substrato, sendo este composto por areia, aragonita (conchas moídas), dolomita, halimeda, entre outros compostos. Assim sendo, o substrato poderá acumular resíduos de matéria orgânica, os quais podem gerar aumento na concentração de amônia e nitritos. Para um aquário já maturado (que já passou pelo período de ciclagem e alcançou o equilíbrio na biologia marinha), as TPAs somadas à manutenção dos parâmetros "principais", como pH, densidade e temperatura são suficientes para processar amônia e nitritos, porém um aquário ainda em período de ciclagem não possui a biologia necessária para este processamento, se fazendo necessária a adição de aceleradores biológicos. Um ornamento que está presente em todos os aquários marinhos e contribui para o processamento de amônia e nitrito é a chamada "rocha viva", rochas porosas formadas por processo de sedimentação biológica, que fornecem em seu interior o abrigo perfeito para uma grande variedade de bactérias que processam amônia em nitrito, nitrito em nitratos e nitratos em nitrogênio livre.

A temperatura do aquário tipicamente deve estar entre 24°C e 27°C, pois acima de 27°C ocorrerá a mortalidade dos corais e, abaixo de 24°C, a mortalidade dos peixes. Porém estes valores podem variar de acordo com as espécies utilizadas. O aquecimento da água é realizado por meio de termostatos com potência aproximada de 1 Watt por litro de água, já o resfriamento da água geralmente é realizado por meio de ventilação forçada. Contudo, a salinidade do aquário em conjunto com a humidade produz condições impróprias para os ventiladores, diminuindo significativamente o tempo de vida útil dos mesmos.

Outra variável de grande relevância é o pH que, geralmente, deve ficar na faixa de 8.0 a 8.4, sendo o ideal 8.2. Porém estes valores podem variar de acordo com a escolha dos espécimes. Normalmente, o equilíbrio químico dentro do aquário tenderá, ao longo do tempo, para um pH inferior, acidificando a água. Existem diversas causas para a diminuição do pH, dentre as quais podem-se citar: aumento de matéria orgânica, aumento de amônia, tipo de substrato utilizado, entre outros. Contudo, um aquário corretamente montado dificilmente apresentará problemas como acidez na água, uma vez que seus elementos constituintes, tais como o sal sintético usado para confeccionar a água marinha, bem como, o substrato e a biologia (especialmente a microbiologia) formada, conduzirão ao equilíbrio químico alcalino tamponado. Existe ainda a possibilidade de adição de tamponadores sintéticos que auxiliam na estabilidade do ph do aquário, úteis, particularmente, para aquários que estão em período de ciclagem.

A iluminação do aquário além de fornecer uma boa estética também cumpre funções biológicas, sendo essencial principalmente para o desenvolvimento dos corais e regulando o ciclo biológico dos animais. Por esse motivo, comumente faz-se o uso de fontes luminosas branca e azul, as quais podem trabalhar em ciclos que tem por objetivo simular dia e noite e evitar mudanças abruptas na luminosidade, neste sentido, a luz azul é acionada uma hora antes da luz branca e desativada uma hora depois. Existem ainda, luminárias muito mais complexas, com outras cores e que simulam muitas outras características climáticas automaticamente, a luminária branca/azul é a mais básica existente, porém a mais amplamente utilizada, principalmente em território nacional, por seu custo reduzido e funcionalidade satisfatória.

O processo de filtragem da água ocorre no chamado sump, um compartimento que fica localizado na parte posterior, em aquários pequenos, ou na parte inferior dentro de móveis preparados, para aquários grandes. Neste compartimento, a água passa por um processo de filtragem com várias etapas que podem ser resumidas em filtragem por borbulhamento através do equipamento chamado Skimmer, filtragem mecânica através de perlon, filtragem química por meio de carvão ativo e filtragem biológica por meio de cerâmica própria. Muitos aquaristas utilizam ainda outros equipamentos no sump com o objetivo de melhorar a qualidade da água, um bom exemplo é o esterilizador UV.

Como é possível observar, a manutenção de um aquário marinho, ainda que pequeno, resulta em tarefa bastante difícil e trabalhosa, exigindo muito tempo e dedicação. Desta forma, a automação se mostra bastante propícia, tendo em vista que busca, dentre outras coisas,

simplificar e melhorar a operação e manutenção de um sistema de modo a facilitar e reduzir as tarefas decorrentes.

1.3. A Automação

O objetivo principal deste projeto é proporcionar uma experiência prática em automação para favorecer o aprendizado e a consolidação dos conceitos abordados durante a graduação. Resumidamente, a palavra automação pode ser compreendida como o ato de transformar processos ou mecanismos naturais em automáticos, ou seja, que não necessitem de interferência humana direta para serem executados.

A automação pode ser definida ainda como um conceito e um conjunto de técnicas por meio das quais se constroem sistemas ativos capazes de atuar com uma eficiência ótima pelo uso de informações recebidas do meio sobre o qual atuam, calculando, através de realimentação, as melhores ações a serem tomadas [7].

Sistemas automáticos ou automatizados se difundiram a partir da revolução industrial, com o advento das máquinas. Está estreitamente conectado a trabalhos mecânicos e repetitivos, fornecendo eficiência, aumento de produtividade e maior qualidade de produto e de vida. A automação, como conceito, envolve automatização, porém é mais abrangente, tomando características gerenciais de controle de processos e informações através, principalmente, do uso de softwares. Desta forma, um sistema de automação é composto por quatro partes básicas, o processo em si, unidades de sensoriamento, um controlador e atuadores. Sensores são componentes sensíveis a algum estímulo físico como temperatura, pressão, luz, entre outros. Estes componentes são integrados a transdutores, que realizam a conversão de sinal analógico, recebido em decorrência do estímulo físico, para digital. O controlador é a unidade que recebe este sinal digital e calcula a ação a ser aplicada no sistema, acionando assim os atuadores, que convertem o comando enviado por sinal digital para um sinal digital que representa uma ação física no sistema.

Desta forma, a automação traduz muitos aspectos positivos em sua utilização, dos quais podemos citar:

- Trata-se de um processo de evolução tecnológica irreversível;

- Valorização do ser humano em sua liberação na execução de tarefas entediantes e repetitivas, ou mesmo em situações de trabalhos insalubres e de riscos;
- Aumento da qualidade de vida de toda uma sociedade, promovendo seu conforto e maior integração;
- Maior enriquecimento pelo menor custo do produto (pela baixa manutenção, ou pela rapidez e precisão na execução de tarefas) ou pelo aumento de produtividade (num curto período de tempo);
- Uma questão de sobrevivência e forte apelo de marketing dentro de um mercado altamente competitivo;
- Criação de empregos diretos e indiretos, além de novos empregos relacionados com a manutenção, desenvolvimento e supervisão de sistemas;
- Busca pela qualidade do produto e a satisfação do cliente [7].

Evoluindo no tema, no tocante ao controle, muitos avanços foram alcançados através de novas tecnologias que buscam a maior eficiência no menor espaço físico, resultando nos componentes conhecidos como microcontroladores, que podem ser definidos como um pequeno computador dentro de um chip. Eles contêm um processador, memória RAM, suas entradas podem ler sinais analógicos e digitais e suas saídas podem ser, também, analógicas ou digitais. Os avanços mais recentes a este respeito resultaram na plataforma Arduino. Originalmente desenvolvido para auxiliar o desenvolvimento de estudantes, tornou-se um produto muito bem sucedido devido sua facilidade de uso, durabilidade e sua natureza aberta, o que o torna uma ótima opção para qualquer um que queira realizar projetos eletrônicos. A partir do sucesso do Arduino, novas plataformas surgiram oferecendo diferentes opções e vantagens, dentre estas o ESP32, utilizado no presente projeto [8].

1.4. Objetivos

Este trabalho tem como principal objetivo implementar os conhecimentos adquiridos durante o período da graduação através da elaboração de um protótipo de testes para um sistema de controle e monitoramento de um aquário marinho e do desenvolvimento de sensores e dos controles necessários para a manutenção dos parâmetros desejados.

Além da aplicação de componentes eletrônicos e dos conceitos de sinais, controle e conversões, este trabalho deve fornecer um aprofundamento nos conceitos envolvendo Arduino e programação, bem como um aprendizado com relação à construção de um aplicativo para celular que seja integrado e possibilite ao usuário acompanhar o status real dos parâmetros e executar algumas intervenções.

1.5. Justificativas

Aquários marinhos são ambientes sensíveis e que necessitam de um cuidadoso monitoramento de seus parâmetros, além de que, as soluções atualmente existentes no mercado são, em alguns casos, pouco eficientes e quase sempre nada integradas, proporcionando, portanto, um cenário perfeito para a implementação de automação e controle.

Estes fatores somados à acessibilidade permitida por smartphones e tecnologias emergentes, tornam a construção de um aplicativo, que permita o acesso à estas informações obtidas em tempo real do aquário, uma oportunidade para expandir a integração e acessibilidade do projeto rumo à uma automação já no ritmo da indústria 4.0.

1.6. Organização do trabalho

Para melhor compreensão a respeito desta monografia, ela está estruturada da seguinte forma: no capítulo 2, é possível encontrar a abordagem metodológica de construção e testes do protótipo, tanto em termos de Hardware como Software. No capítulo 3, são abordados os resultados obtidos nas versões finais do protótipo e também um link indicando um vídeo explicativo sobre o funcionamento final do protótipo.

Por fim, no capítulo 4, se encontram as conclusões às quais a construção do protótipo e elaboração desta monografia nos levaram, no capítulo 5, constam as bibliografias utilizadas para fundamentação teórica e no capítulo 6 os anexos contendo os códigos utilizados.

2. METODOLOGIA

Para o desenvolvimento deste trabalho, foi escolhida a plataforma ESP32, principalmente por possuir módulo wi-fi embutido e uma quantidade mais confortável de entradas analógicas.

Inicialmente estudava-se a possibilidade do uso de um Arduino Mega como plataforma para o desenvolvimento do projeto. Porém, através de pesquisas, optou-se pelo uso do ESP32 do fabricante Espressif Systems.

O ESP32 possui o microprocessador dual core Tensilica Xtensa 32-bit LX6, que trabalha com um clock típico de 160MHz, enquanto que o Arduino trabalha com clock de 16MHZ, o que implica em um processamento dez vezes maior que o Arduino. Além disso o ESP32 possui (sem a necessidade de módulos externos) Bluetooth e Wi-fi, sendo de grande interesse para o desenvolvimento de projetos na área de automação. Outros diferenciais são os 512KB de memória RAM, contra apenas 2KB do Arduino; e os 18 conversores analógico digital de 12 bits contra 6 conversores de 8 bits do Arduino, proporcionando assim maior precisão nas leituras analógicas. Destacam-se ainda, o baixo consumo de energia, o tamanho reduzido do circuito e o baixo custo (chegando a ser inferior a alguns modelos do Arduino).

O diagrama de blocos completo do projeto dá uma ideia geral das funcionalidades que são detalhadas nas próximas sessões e pode ser visualizado na Figura 1.

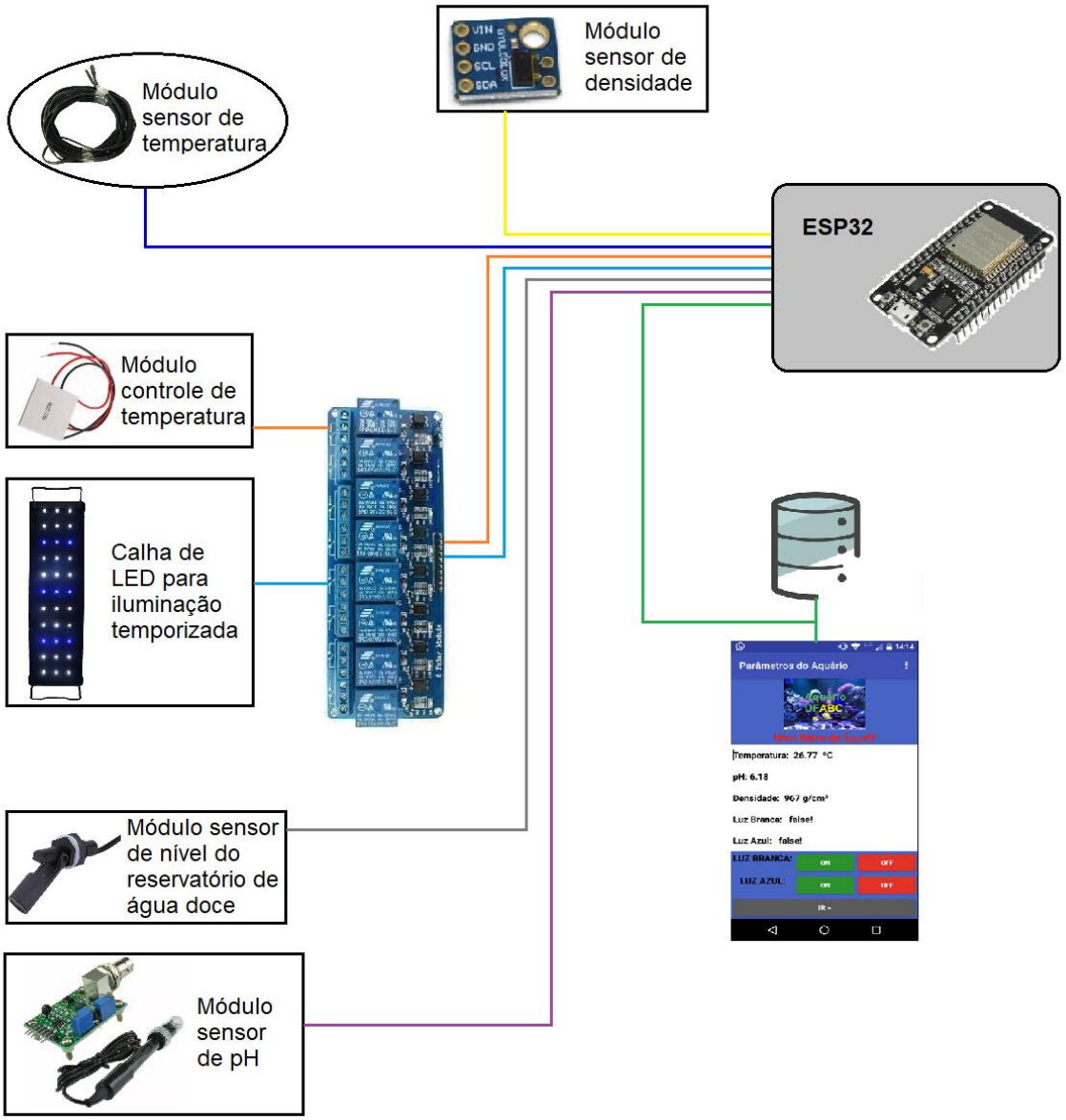

Figura 1: Diagrama de blocos do projeto.

2.1. Módulo Sensor de densidade

A medição de densidade apresentou um desafio maior por não existir no mercado um sensor próprio para este fim que não fosse componente laboratorial, o que encarece significativamente o equipamento, além de exigir cuidados e calibrações frequentes que o tornam inviável para a proposta do projeto. Desta forma, foi necessária a elaboração de um dispositivo capaz de mensurar a densidade da água e traduzir esta medida em sinal digital.

Para tanto, foi elaborado um sistema utilizando um densímetro flutuante dentro de um tubo PVC conectado diretamente ao aquário de forma que fosse possível obter uma coluna

d’água no mesmo nível de altura do aquário e com a amostra fidedigna. À extremidade superior do densímetro foi acoplado um componente plástico resistente, porém o mais leve possível, com o diâmetro muito próximo ao diâmetro interno do tubo PVC e este tubo foi selado com um “cap” de mesmo material ao qual foi acoplado um sensor ultrassônico. A Figura 2 mostra o esquema de montagem do sistema.

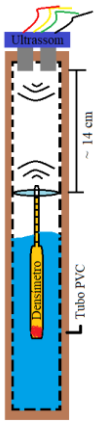

Figura 2: Esquema de montagem do sensor de densidade.

O funcionamento deste módulo se baseia no densímetro flutuante, que altera a altura de sua coluna graduada com relação à superfície da água de acordo com a densidade desta. Sendo assim, sabendo que o nível da água do tubo PVC seguirá o nível de água do aquário, que deve ser constante, é possível utilizar esta variação de altura para obter o valor de densidade. Com a fixação do componente plástico ao densímetro, o sensor ultrassônico é capaz de detectar a variação de altura e enviar ao ESP32, que traduz a leitura em centímetros para um valor de densidade correspondente por meio do código programado. As Figuras 3 a 5 apresentam o protótipo dos testes e uma estimativa de sua posição de montagem.

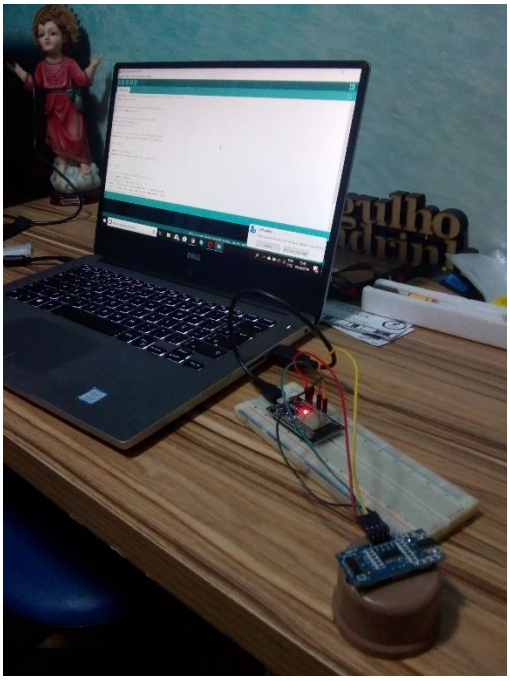

Figura 3: Protótipo de testes.

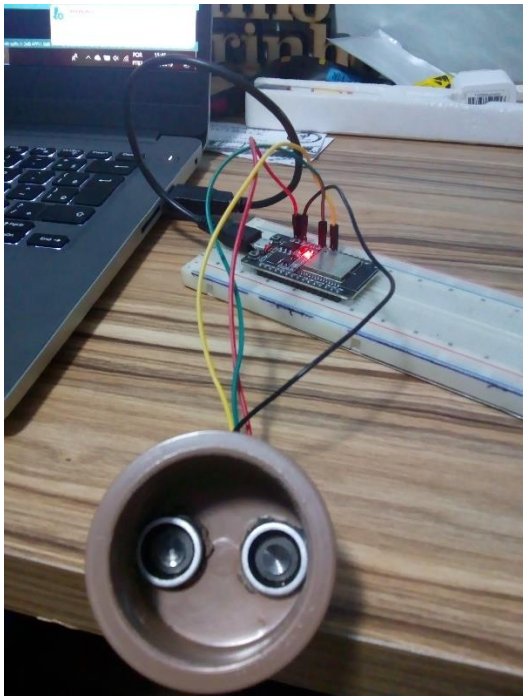

Figura 4: Disposição do sensor ultrassônico no protótipo.

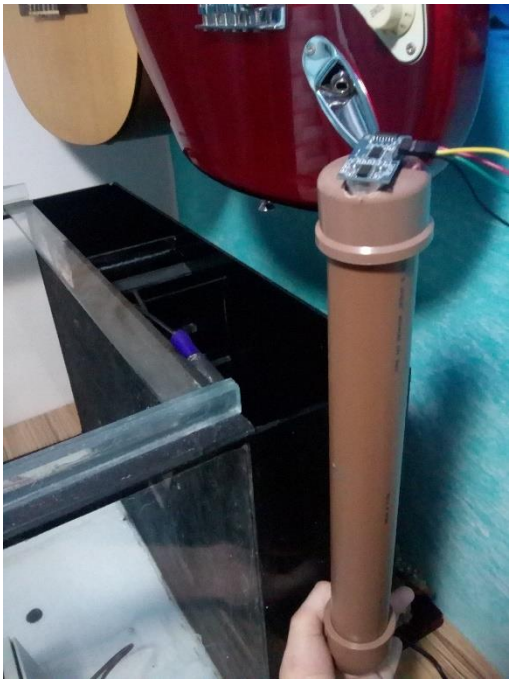

Figura 5: Posição estimada do Sistema.

Após os testes do módulo, os resultados se mostraram muito instáveis e imprecisos, levando-nos a adotar outro tipo de sensor de distância, o sensor infravermelho GY UL53LOX, que se mostrou muito mais preciso que o sensor ultrassônico, no entanto, o conjunto densímetro com componente plástico se mostrou uma dificuldade para a finalização confiável do módulo, pois perpetuou a instabilidade das medidas, mostrando ser necessária uma reavaliação do módulo, que foi fisicamente reformulado para driblar alguns eventos que causaram instabilidade como, por exemplo, a sucção gerada entre a água e a superfície plástica adaptada ao densímetro. Também foi alterado o código para que realizasse a linearização das medidas a partir de dois valores de referências aferidos experimentalmente.

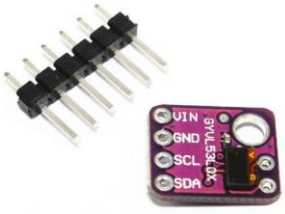

Figura 6: Sensor Infravermelho de distância, GY UL53LOX.

Desta forma, a partir de uma prévia e cuidadosa calibração da posição do sensor e de duas medidas de referência, foi possível aferir a densidade de maneira satisfatória. Os códigos utilizados neste módulo se encontram, na íntegra, em Anexo 1 e 2.

2.2. Módulo Sensor e Controle de temperatura.

Para a medição da temperatura, foi utilizado um sensor NTC de 10 KΩ, que é um resistor termicamente sensível com Coeficiente de Temperatura Negativo, ou seja, sua resistência decresce com o aumento da temperatura.

Para o Arduino, já existem inúmeras bibliotecas que fazem a leitura da resistência de sensores NTC e calculam a temperatura baseado nas características do componente específico e em fórmulas matemáticas que relacionam Resistência e Temperatura, como a equação de Steinhart-Hart para o termistor. No entanto, devido ao fato de utilizarmos a placa ESP32 e esta ter diferenças de funcionalidades internas com o Arduino, não foi possível utilizar nenhuma das bibliotecas existentes, tivemos de criar um novo programa utilizando estas bibliotecas como base de instruções afim de construir um código que realizasse a medida.

Inicialmente, estudamos a equação de Steinhart-Hart e realizamos medidas de temperatura, com um termômetro de fluidos, e resistência, com o termistor NTC e um multímetro, a fim de obter os dados necessários (pontos de operação) para calcular os coeficientes da equação para o termistor utilizado. As medidas e o passo a passo dos cálculos são apresentados abaixo.

Para obtermos os coeficientes da equação de Steinhart-Hart, é necessária a resolução da equação matricial abaixo.

$$\begin{bmatrix} 1 & \ln R_1 & \ln^3 R_1 \\ 1 & \ln R_2 & \ln^3 R_2 \\ 1 & \ln R_3 & \ln^3 R_3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \frac{1}{T_1} \\ \frac{1}{T_2} \\ \frac{1}{T_3} \end{bmatrix}$$

Tabela 1: Pontos de operação para o cálculo dos coeficientes de Steinhart-Hart

T(ºC)	R (KΩ)
9,2	19,11
17,8	13,42
59,2	3,30
T (K)	**R (Ω)**
282,35	1911
290,95	1342
332,35	3300

De posse dos dados da Tabela 1, iniciamos os cálculos obtendo L1, L2 e L3; Y1, Y2 e Y3. Sendo:

$$L_1 = \ln R_1, \quad L_2 = \ln R_2, \quad L_3 = \ln R_3$$
$$Y_1 = \frac{1}{T_1}, \quad Y_2 = \frac{1}{T_2}, \quad Y_3 = \frac{1}{T_3}$$

Desta forma obtemos os valores da Tabela 2.

Tabela 2: Valores para cálculo dos coeficientes de Steinhart–Hart.

L1 = 9,8580	Y1 = 3,5417x10^{-3}
L2 = 9,5045	Y2 = 3,4370x10^{-3}
L3 = 8,1017	Y3 = 3,0089x10^{-3}

Para o cálculo dos coeficientes, precisamos ainda obter os valores de γ2 e γ3, provenientes das equações abaixo.

$$\gamma_2 = \frac{Y_2 - Y_1}{L_2 - L_1}, \quad \gamma_3 = \frac{Y_3 - Y_1}{L_3 - L_1}$$

Tabela 3: Valores de γ2 e γ3.

γ2	γ3
2,9618x10^{-4}	3,0336x10^{-4}

Por fim, substituindo os dados na fórmula a seguir, obtemos os coeficientes.

$$C = \left(\frac{\gamma_3 - \gamma_2}{L_3 - L_2}\right)(L_1 + L_2 + L_3)^{-1}$$
$$B = \gamma_2 - C\left(L_1^2 + L_1 L_2 + L_2^2\right)$$
$$A = Y_1 - \left(B + L_1^2 C\right) L_1$$

Obtendo os valores da Tabela 4.

Tabela 4: Valores das constantes de Steinhart-Hart.

C	B	A
-1,86357303x10^{-7}	3,48586482x10^{-4}	2,8386514x10^{-4}

Em posse dos coeficientes, é possível montar a Equação de Steinhart-Hart que relaciona a resistência e a temperatura para o termistor utilizado, substituindo-os na forma geral que segue abaixo.

$$\frac{1}{T} = A + B\ln R + C(\ln R)^3,$$

O código desenvolvido realiza a medida da resistência do termistor, considerando um resistor fixo de referência, calcula o logaritmo deste valor e obtém a temperatura, a partir da fórmula anterior.

Figura 7: Sensor NTC encapsulado para utilização em aquário.

Na sequência, o código trabalha com um módulo de relés, duas bombas d'água e pastilha peltier com dissipadores para realizar o controle da temperatura a partir de limites escolhidos pelo usuário.

Figura 8: Mini bomba d'água submersa.

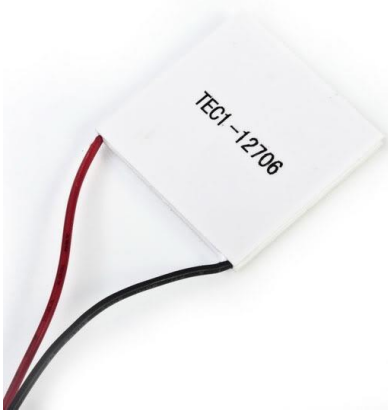

Figura 9: Pastilha Peltier.

Quando o sistema detecta que a água do aquário está abaixo do limite de temperatura escolhido pelo usuário, a pastilha peltier é acionada juntamente com a mini bomba d'água que faz a água do aquário circular por uma mangueira entre as ranhuras do dissipador fixado na extremidade que esquenta da pastilha, fazendo assim com que a água aqueça e retorne ao aquário. O procedimento é o mesmo quando a temperatura do aquário está acima do limite de temperatura, com a diferença de que a mini bomba acionada é aquela que faz com que a água circule entre as ranhuras do dissipador fixado ao lado que esfria da pastilha.

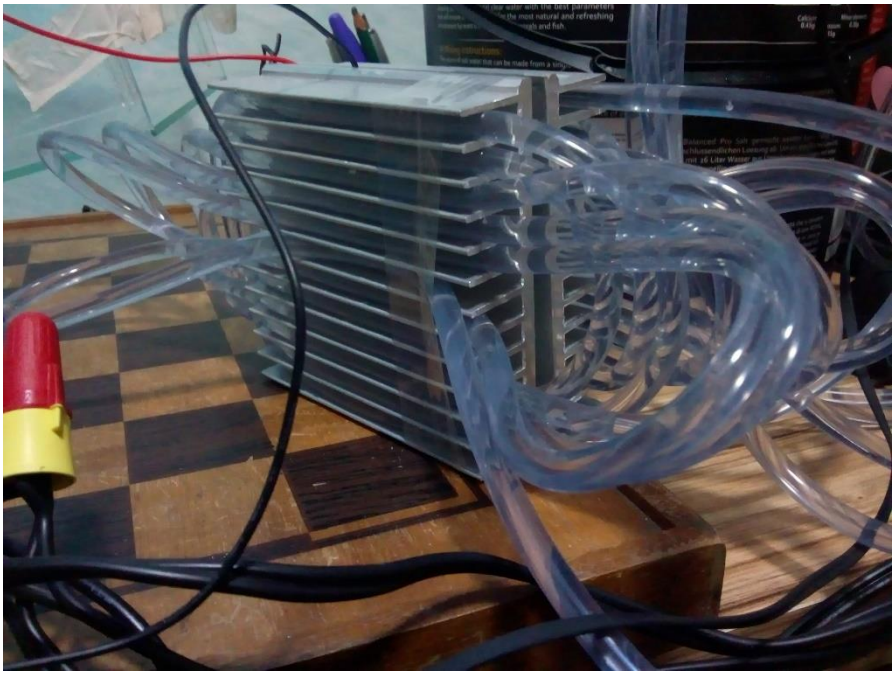

Figura 10: Conjunto dos dissipadores de alumínio envolvendo a pastilha peltier e com a mangueira para passagem de água.

O código deste módulo consta no anexo 3.

2.3. Módulo Controle de Nível do reservatório de água doce.

Neste módulo, utilizamos um sensor de nível para Arduino, do tipo boia horizontal magnética, que aciona um LED vermelho quando o nível de água do reservatório de reposição de água doce está baixo.

Figura 11: Sensor de nível do tipo boia horizontal magnética.

O código deste módulo pode ser visto em anexo 4.

2.4. Módulo RTC e funções temporizadas

Inicialmente, cogitou-se o uso de um módulo para sistemas embarcados denominado RTC (Real Time Clock), no qual seria necessário que o usuário configurasse a data e horário local, para que este módulo, a partir daí, contasse o tempo. Porém, após um dado período, o usuário teria que configurar o tempo novamente, pois haveria uma defasagem com relação ao horário verdadeiro.

Conforme mencionado anteriormente, o ESP 32 possui “wi-fi”, sem a necessidade de um módulo separado. Devido a esta facilidade, optou-se pelo uso do protocolo NTP (Network Time Protocol), que atualiza a data e horário local, via servidor web, mantendo-se sempre atualizado e excluindo o uso de um módulo RTC.

2.5. Módulo sensor de pH

A leitura do pH é realizada através do sensor PH-4502C da “DiyMore”. Trata-se de um sensor analógico, com tensão de alimentação de 5V e corrente de trabalho de 5 a 10 mA (variando com o pH do meio). O intervalo de detecção abrange toda a escala de pH, ou seja, passível de medir variações de pH que vão de 0 a 14. O eletrodo de leitura (transdutor de pH) é acoplado, através de um conector BNC, a um módulo com um circuito próprio. Tal circuito tem como funções, amplificar e linearizar a saída do sensor, pois o pH não segue uma escala linear, mas sim logarítmica. Além disso, o módulo possui ainda, a função de leitura de temperatura ambiente (podendo variar de 0 a 80 °C), porém esta função não é utilizada neste circuito. O tempo de vida útil do eletrodo indicado pelo fabricante é de três anos; após isto, o mesmo deverá ser substituído.

Figura 12: Sensor de pH utilizado (PH-4502C).

De forma simplificada, pH significa potencial hidrogeniônico (quantidade de prótons H+), possibilitando indicar a neutralidade, acidez ou mesmo a alcalinidade de uma solução líquida. Deste modo, a calibração do sensor de pH exige o uso de pelo menos três soluções aquosa de pH bem definidos, sendo uma de pH ácido, outra de pH neutro e uma de pH alcalino. Porém, através de pesquisas, encontrou-se uma segunda maneira de efetuar de forma confiável, a calibração. Ao alimentar o módulo do sensor com 5V, mas sem acoplar transdutor de pH, mediu-se a tensão na saída do módulo como sendo igual a 5V. Tal tensão corresponderia a um pH de 14. Em seguida, ao colocar-se um curto-circuito no conector BNC, mede-se novamente a tensão na saída do módulo, sendo de 2,5V, o que corresponderia a um pH de 7. Para se obter as respectivas tensões, é necessário ajustar o "trimpot" (R23). Após esta primeira calibração, faz-se necessário a adequação do sinal analógico da saída do módulo medição de pH (que varia de 0 a 5V), para a entrada analógica do ESP32 (GPIO34), uma vez que, conforme mencionado anteriormente, trabalha com tensões de entrada de 0 a 3,3V. Devido a baixíssima corrente exigida na entrada do ESP32 (na ordem de micro Ampere), considerou-se que o mesmo possui impedância de entrada "infinita", optando-se pelo simples uso de um divisor de tensão na saída do módulo de medição de pH, usando um trimpot de 10Kohms. Assim, com a saída do sensor de pH (transdutor + modulo) em 5V, ajustou-se a desejável tensão de 3,3V na entrada analógica do ESP32.

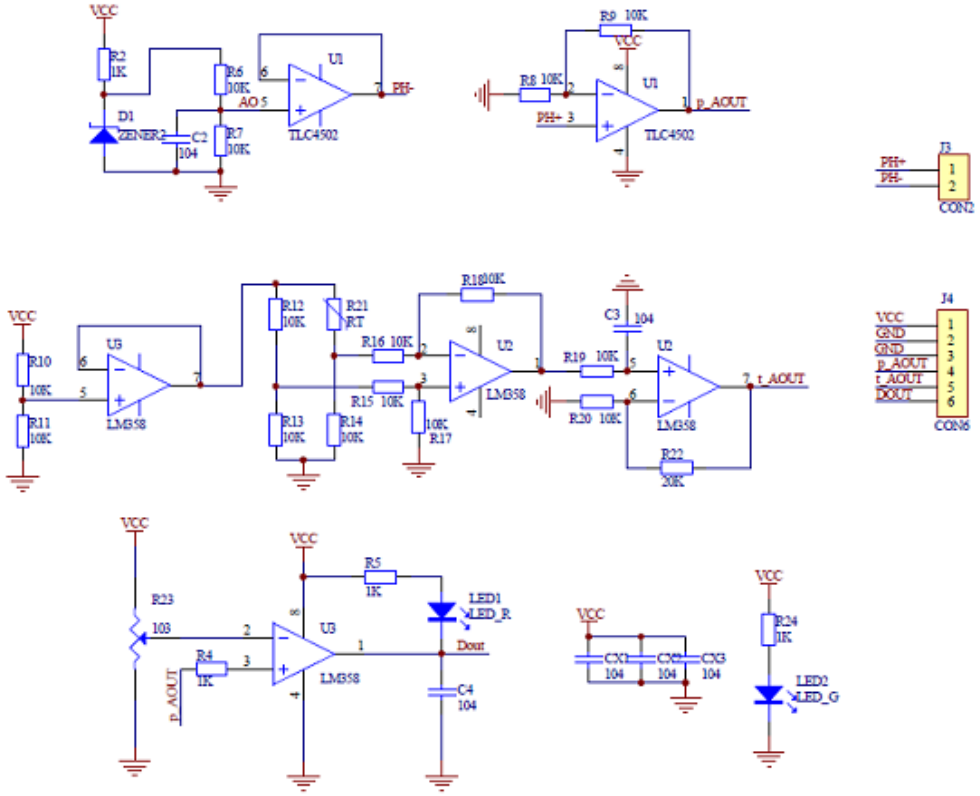

Figura 13: Esquema elétrico do circuito do módulo PH-4502C.

Ao se efetuar as primeiras medições do sensor de pH, observou-se oscilações significativas nos valores obtidos, mesmo considerando uma incerteza de ±0,2 no valor de pH, conforme descreve o fabricante. Optou-se então pela utilização de filtros digitais, de modo a minimizar as oscilações nas leituras. O filtro utilizado é o de cálculo da média gaussiana, o qual tem seu modelo matemático da função “densidade de probabilidade” brevemente apresentado a seguir.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad -\infty < x < \infty$$

O modelo probabilístico gaussiano, consiste na aproximação de “n” leituras, em um período de amostragem “T”, para um histograma com os valores “X” das leituras do sensor de pH realizados pelo ESP32, os quais são divididos em pequenos intervalos de valores; aplicados a uma biblioteca de média gaussiana, a qual realiza a os cálculos de média μ e do desvio padrão

σ para a curva. Então o valor da média é retornado e apresentado no display LCD. É possível verificar uma versão do programa usada para testes na leitura do pH no anexo 5.

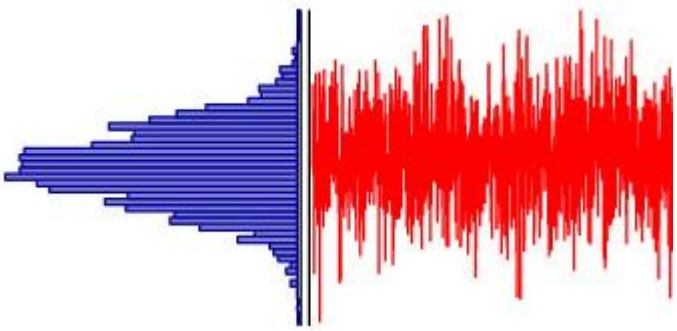

Figura 14: Distribuição de um sinal ruidoso em um gráfico de histograma.

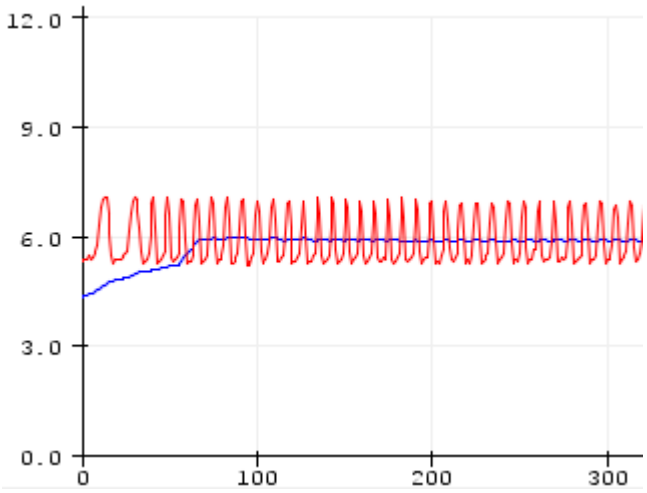

Figura 15: Gráfico das medidas de pH em função do tempo.

Conforme é possível observar na figura anterior, temos, para fins de comparação, duas curvas de pH sendo plotadas no mesmo gráfico. A curva de cor vermelho representa a leitura do pH sem aplicação da média gaussiana, enquanto que a curva de cor azul representa a leitura do pH com a aplicação da media gaussiana. É possível notar aqui, uma nítida diferença entre as leituras de ambas as curvas, havendo uma maior estabilidade no caso em que se utiliza a média gaussiana. Porém, também é necessário considerar que a aplicação deste filtro ocasiona um atraso no sinal de saída (leitura realizada), com relação ao sinal real na entrada do ESP32. O atraso em questão é diretamente dependente da quantidade de amostragens por cálculo da média.

A quantidade de amostragens, por sua vez, considerando o teorema de Nyquist, também conhecido como “teorema da amostragem” (onde em um processamento digital de sinais a frequência de amostragem do sinal deve ser, no mínimo, o dobro da maior frequência que compõe o sinal original, para assim manter suas características), foi estabelecida em um número de 100 leituras para cada valor de média, ocasionando um atraso estimado de 60 segundos. Levando em conta o gradiente de pH ao longo do tempo em aquários marinhos, torna-se insignificante tal atraso na leitura deste paramento.

2.6. O esquemático desenvolvido

Existe diversos softwares de desenho e simulação de circuitos eletrônicos, porém alguns ainda não possuem o ESP32 em sua biblioteca de componentes. Pensando nisso, optou-se pelo software “Fritzing”, que além de ser gratuito, possui uma vasta biblioteca de componentes eletrônicos, tais como sensores e transdutores que permite que o desenvolvedor crie PCBs(Printed Circuit Boards) e também simule programas e circuitos.

A representação do diagrama esquemático do circuito desenvolvido não pode ser simulada pois ocorreram incompatibilidades com a IDE do Arduino e o Fritzing. Ainda assim, o desenho do circuito foi de grande utilidade para a criação de um protótipo.

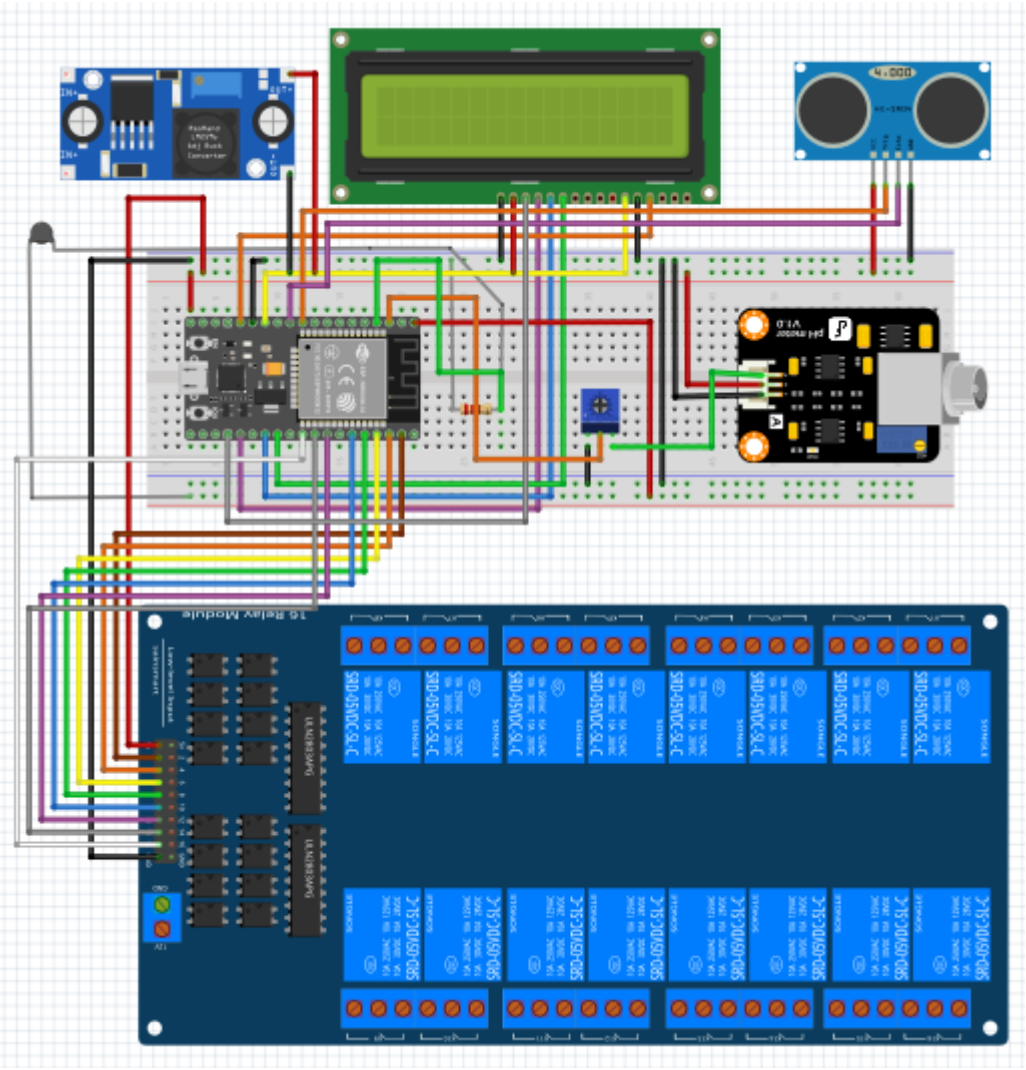

Figura 16: Representação do circuito no Fritzing.

O protótipo desenvolvido utiliza uma placa de acrílico onde foram fixados os módulos e sensores, facilitando os testes executados durante a etapa de programação.

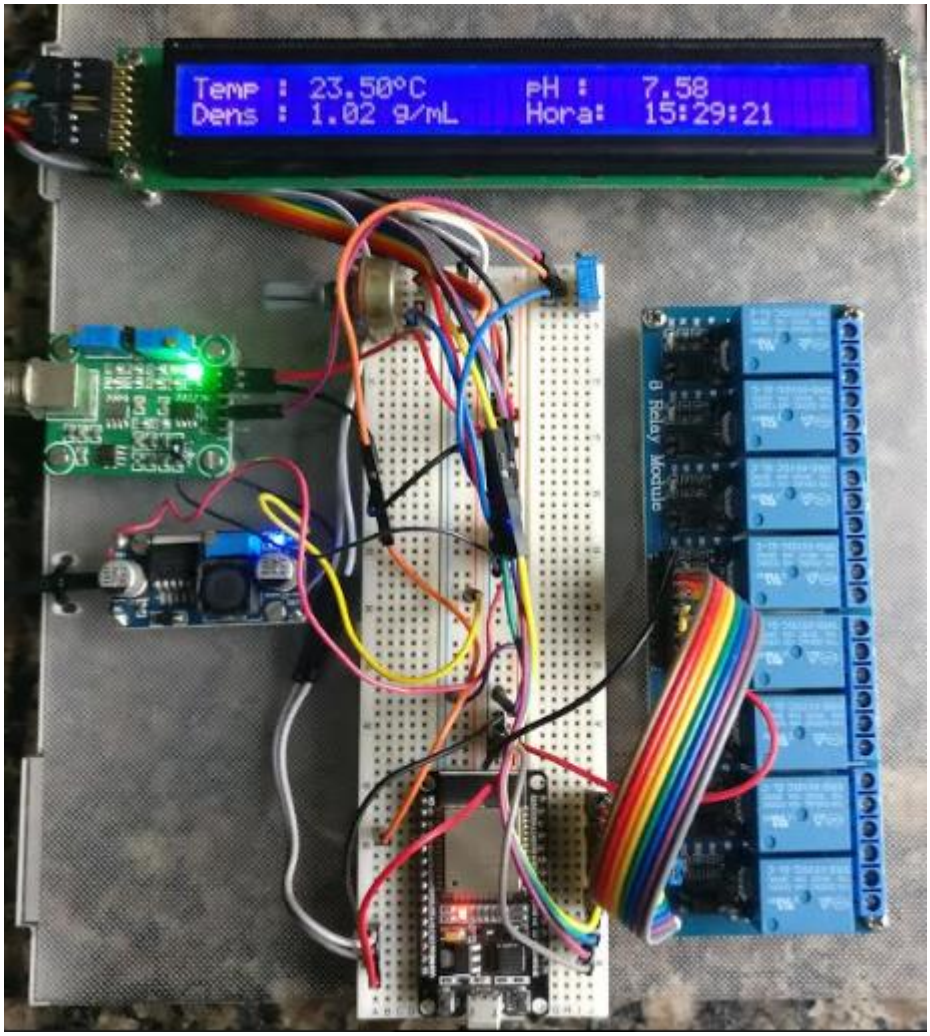

Figura 17: Protótipo desenvolvido para testes de software e hardware.

2.7. O aplicativo

O desenvolvimento do aplicativo apresentou um vasto campo de possibilidades, porém, ao mesmo tempo, grandes desafios, uma vez que seria necessário a integração de diferentes ferramentas e linguagens de programação.

Inicialmente a ideia era criar um site que informasse ao usuário os parâmetros do aquário e permitisse que o mesmo alterasse o estado da luminária. Tal site funcionaria como um aplicativo.

Para o desenvolvimento do site foram pesquisadas diferentes soluções, visando uma alternativa que representasse uma solução simples, com qualidade e sem custos. Na internet é possível encontrar diversos servidores para hospedagem de sites, os chamados “hosting services” gratuitos. Geralmente esse tipo de serviço é oferecido por tempo limitado, ou

apresenta um baixo desempenho (o servidor fica frequentemente fora do ar), além de manter constantemente anúncios e propagandas dos mais variados assuntos, não inspirando portanto uma solução plausível.

Surgiu então uma segunda possibilidade, a criação de um site que ficasse hospedado no próprio computador. Então, optou-se pelo uso de um pacote de softwares gratuitos denominados pela sigla WAMP(Windows Apache MySQL e PHP). O WAMP é frequentemente usado por desenvolvedores web e possui também versões para Linux (LAMP) e Mac (MAMP). De modo simplificado, o Apache é um software que atua como servidor, sendo, portanto, capaz de hospedar sites no próprio computador. O MySQL funciona como um gerenciador de banco de dados do servidor. E o PHP é uma linguagem de programação comumente utilizada no desenvolvimento de sites, sendo capaz de interligar o banco de dados ao servidor.

Devido a dificuldades na programação e integração entre o site e o hardware desenvolvido, logo descartou-se esta possibilidade, uma vez que o entendimento necessário da ferramenta demandaria uma grande quantidade de tempo e não havia garantia de sucesso.

Então optou-se pelo desenvolvimento de um aplicativo. Existem diversas plataformas gratuitas de desenvolvimento. Foi escolhido inicialmente o software gratuito Android Studio, sendo uma das mais conhecidas ferramentas, dispondo, portanto, bastante literatura. De fato, mostrou-se uma ferramenta bastante dinâmica e capaz de oferecer uma solução interessante. Contudo, o software apresentou lentidão, acarretando em travamento do computador ao rodar o emulador, dificuldades na simulação do aplicativo e falha ao não identificar um pacote de ferramentas denominado SDK (Software Development Kit). Devido a dificuldades em sanar frequentes dúvidas em relação a programação e aos demais problemas apresentados, resolveu-se estudar o uso de outras ferramentas. Logo surgiu a ideia de buscar uma solução capaz de armazenar os parâmetros do aquário em nuvem.

O armazenamento em nuvem dispensa a necessidade de um armazenamento local, ficando os dados guardados e disponíveis em um servidor online. Bastando, para acessar esses arquivos armazenados, que o usuário disponha de uma conexão estável com a internet. Como o hardware desenvolvido utiliza o ESP32, que conforme visto, possui um sistema wi-fi integrado, facilitando tal conexão, torna-se possível o envio dos parâmetros dos sensores do aquário para a nuvem, diminuindo portanto o uso de memória no hardware e dando ao usuário a possibilidade do acesso remoto das informações. Mas para que isso seja possível é necessário

estabelecer um protocolo de comunicação entre o hardware e a nuvem. Embora ainda não haja um protocolo definido como padrão para tecnologias IoT, o protocolo que vem sendo amplamente adotado para tal aplicação é o chamado MQTT (Message Queuing Telemetry Transport). Isso se deve à facilidade na comunicação entre máquinas, que constantemente pode ser encontrada em literatura como M2M(Machine-to-Machine).

O protocolo de comunicação MQTT foi desenvolvido pela IBM ainda nos anos 90. Tal protocolo permite a um sistema a troca de mensagens de modo assíncrono, desacoplando o emissor e o receptor das mensagens, tanto no espaço quanto no tempo, por esse motivo é ideal para ambientes de rede instáveis.

As mensagens trocadas via MQTT são organizadas por tópicos, permitindo ao desenvolvedor de aplicativos, maior flexibilidade ao especificar quais clientes podem interagir com determinadas mensagens. Deste modo os sensores poderiam suas leituras em um banco de dados “backend”, que por sua vez poderia, por exemplo, ser acessado por um aplicativo destinado a um usuário sem privilégios, apenas para leitura dos parâmetros. Também é possível criar um usuário com privilégios para alteração de configurações, conforme as necessidades do projeto. Contudo, apesar das vantagens do uso deste protocolo, considerando os demais aspectos necessários para o desenvolvimento de um aplicativo, procurou-se uma ferramenta que oferecesse maior facilidade e dinamismo na hora de programar.

Existem diversos serviços gratuitos de armazenamento em nuvem. Depois de estudar algumas opções, foi escolhido a plataforma da Google de nome Firebase.

O Firebase é uma ferramenta digital para auxiliar o desenvolvimento e analise de aplicativos web e mobile. É possível utilizá-lo nas principais plataformas de desenvolvimento de aplicativos. Existe uma infinidade de funções do Firebase, que incluem desde segurança até divulgação do aplicativo. Com ele é possível criar um banco de dados na nuvem, onde poderão ser salvos dados das leituras dos sensores. As funcionalidades mais avançadas, as quais envolvem diversas ferramentas de análise de aplicativos, são geralmente cobradas. Porém o Firebase também fornece um pacote de serviços gratuitos que, apesar de algumas limitações, é satisfatório para o desenvolvimento do presente projeto.

Para a utilização do Firebase inicialmente foi necessário criar uma conta Google (“aquario.ufabc@gmail.com”) e acessar o console através do pacote denominado “Plano Spark”. Então criou-se o projeto com nome “Aquario_UFABC”, que por sua vez gerou uma

chave de acesso e um endereço “host” para escrita e leitura de dados. Foi utilizado o “realtime database” que utiliza um protocolo próprio de comunicação para troca de dados.

O banco de dados em tempo real permite o armazenamento e sincronismo dos dados entre usuários e dispositivos em tempo real com um banco de dados NoSQL hospedado na nuvem.

Os dados atualizados são sincronizados em todos os dispositivos conectados em segundos, no caso do projeto desenvolvido observou-se um atraso de aproximadamente 8 segundos. Além disso, os dados permanecem disponíveis caso o aplicativo fique offline, sendo, neste caso, atualizado imediatamente na reconexão com a rede.

A comunicação entre o “Firebase” e o “Esp 32” inicia-se através da instalação de uma biblioteca específica denominada “FirebaseESP32.h”, observada no início do código desenvolvido. Em seguida, define-se as variáveis de “host” (endereço) e “Senha_Fire” (chave de acesso), conforme a figura:

```
/*Cria duas variaveis, para o host e para a senha do firebase*/
#define Host "https://aquario-ufabc.firebaseio.com/"//host
#define Senha_Fire "                                        "//chave do firebase
```

Figura 18: Linhas de código de configuração de ‘host’ e senha para o hardware acessar a nuvem.

A troca de dados então ocorre por meio dos comandos “Firebase.set”, usado para enviar; e “Firebase.get” para receber os respectivos valores das variáveis do banco de dados em nuvem. As variáveis podem ser criadas tanto na programação do ESP 32 como também na própria plataforma “realtime database”.

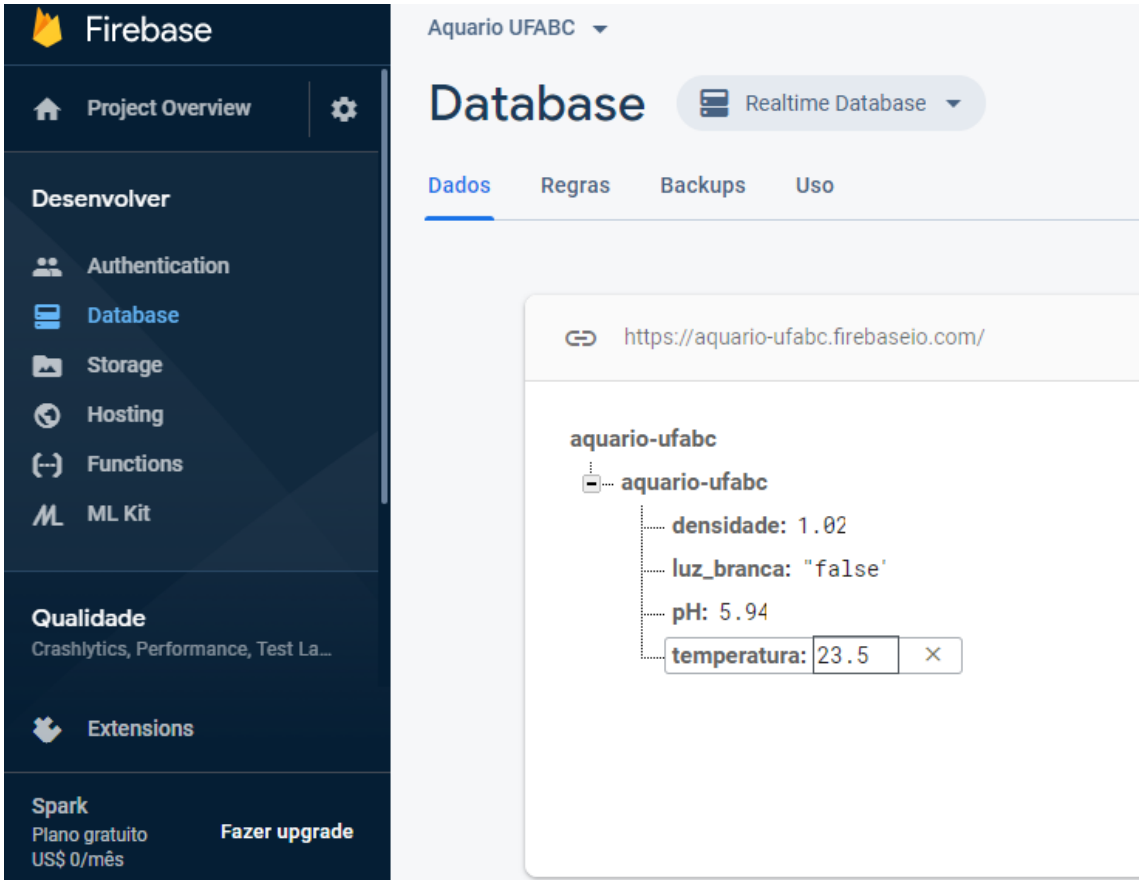

Figura 19: Parâmetros do aquário no Realtime Database do Firebase.

Uma vez estabelecida a comunicação entre o hardware e a nuvem, buscou-se o desenvolvimento do aplicativo.

A plataforma escolhida para o desenvolvimento do aplicativo foi o Kodular. Esta ferramenta online de desenvolvimento de aplicações para Android, permite a programação de maneira simples e intuitiva, através de blocos lógicos. O Kodular apresenta uma grande variedade de funções, permitindo a criação de inúmeras aplicações, tornando-o uma ferramenta bastante didática e interessante. Além disso, o Kodular possui funcionalidades específicas que permite o fluxo de dados com o Firebase, atendendo as características propostas pelo presente projeto.

O desenvolvimento do aplicativo pode ser dividido em duas principais etapas: a criação da interface gráfica com o usuário através de “Screnns” (telas) e a programação em blocos. Na figura seguinte é possível observar a plataforma Kodular. À esquerda, ao entrar na aba “Google” é possível selecionar a ferramenta “Firebase Database”, onde deve-se configurar o endereço e a senha de acesso ao banco de dados do Firebase, estabelecendo assim portanto, a comunicação entre o aplicativo e a nuvem.

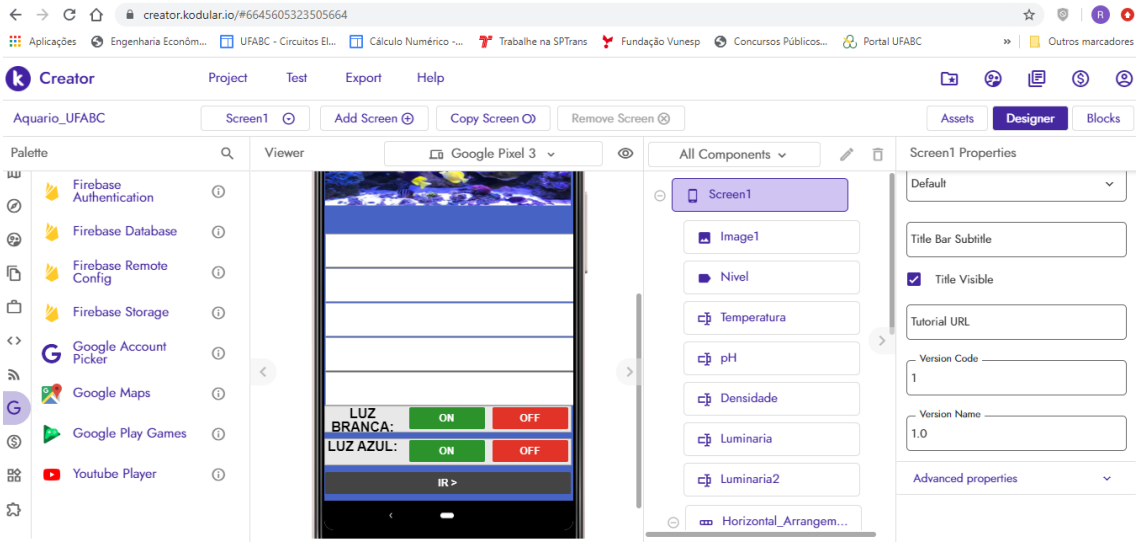

Figura 20: Plataforma Kodular de desenvolvimento de aplicativos.

O Kodular possui diversos blocos que permitem uma fácil comunicação entre o banco de dados do Firebase com o aplicativo. Para o desenvolvimento aqui apresentado, foram utilizados os blocos principais “Got Value”(valor obtido) o qual indica que uma solicitação “GetValue” foi bem-sucedida e “Data Changed” (dado alterado) o qual indica que os dados no Firebase foram alterados e inicia um evento com a “tag”(nome da variável) e o valor que foram atualizados.

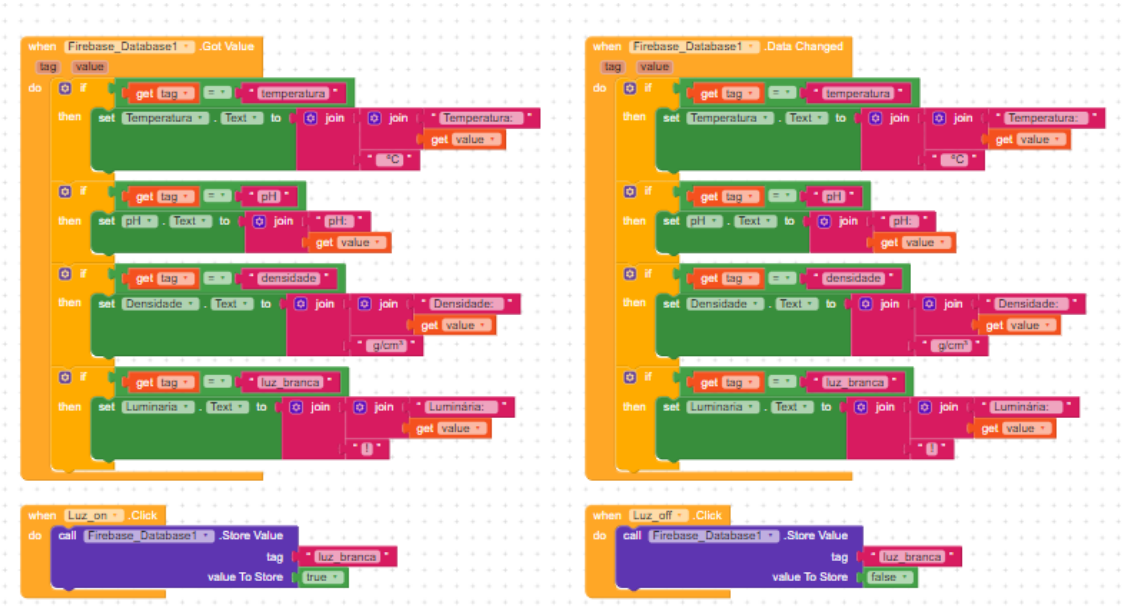

Figura 21: Programação em blocos no Kodular.

3. RESULTADOS

Os resultados obtidos neste projeto superaram as expectativas iniciais de quando concebemos a ideia. Os módulos construídos separadamente facilitaram bastante o trabalho inicial, que evoluiu rápido, e possibilitou testes, cálculos e correções mais pontuais e simples.

A partir do teste dos módulos individuais, foi possível verificar os resultados no plotter serial do Arduino, como por exemplo os resultados do módulo de densidade e temperatura que seguem nas figuras 20 e 21, e acompanhar cada variável do código para interpretar melhor eventuais erros e atuar mais assertivamente.

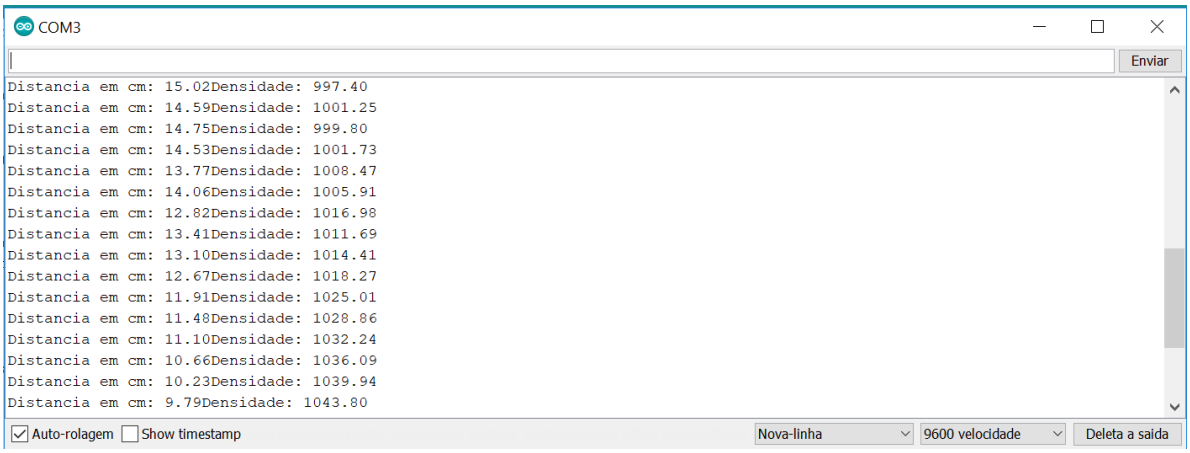

Figura 22: Resultados Experimentais - Densidade.

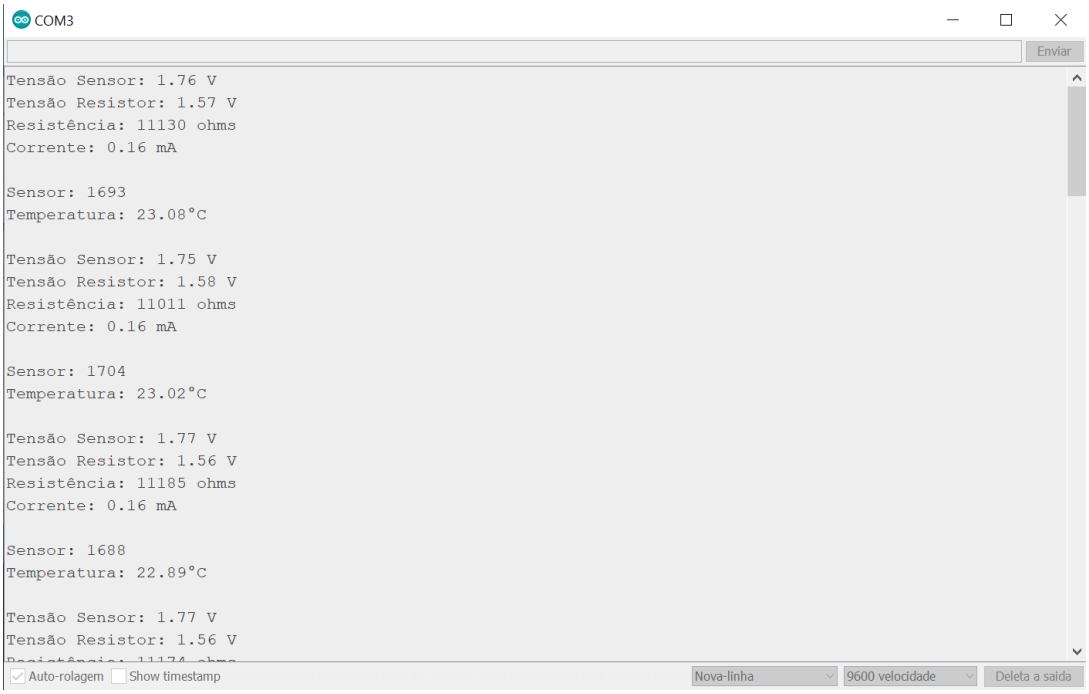

Figura 23: Resultados obtidos em uma leitura - Temperatura.

Na finalização do projeto, todos os módulos foram agregados em um só, proporcionando ao protótipo um controle único e centralizado. Nesta integração de módulos, surgiram algumas dificuldades funcionais, de conectividade e de compatibilidade, no entanto, foram sanadas mais facilmente devido à organização e familiaridade com o código.

O aplicativo desenvolvido, por sua vez, demonstrou funcionalidade plena, dentro do que era esperado e que foi programado, apresentando as medições de forma clara e tempo de resposta bastante curto, além um visual amigável, conexão direta com o ESP32 e possibilidade de melhorias futuras. Apesar do bom tempo de resposta do aplicativo, o protótipo completo integrado apresentou um atraso de aproximadamente 8 segundos com relação às atualizações dos parâmetros e, consequentemente, aos comandos implementados. É possível observar como ficaram as telas do aplicativo na figura 22, já na Figura 23, é possível observar os alertas para prazo de realização de trocas vencido.

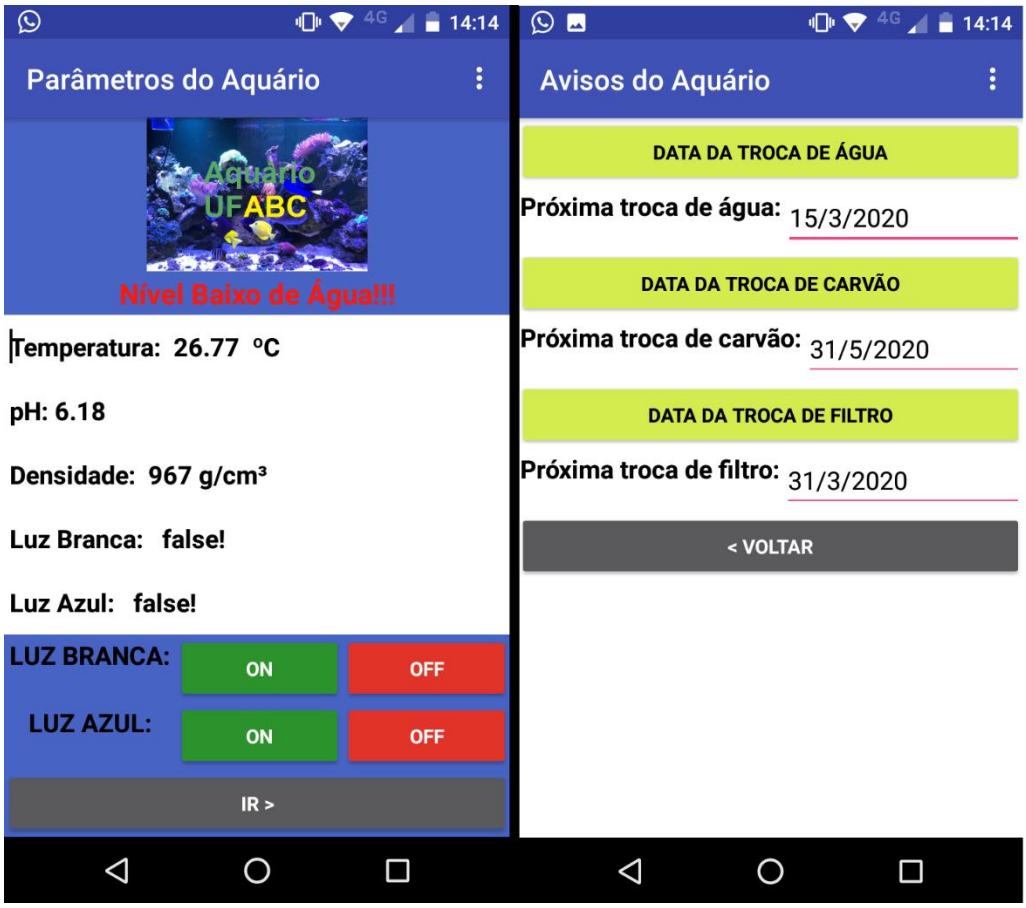

Figura 24: Aplicativo desenvolvido na plataforma Kodular.

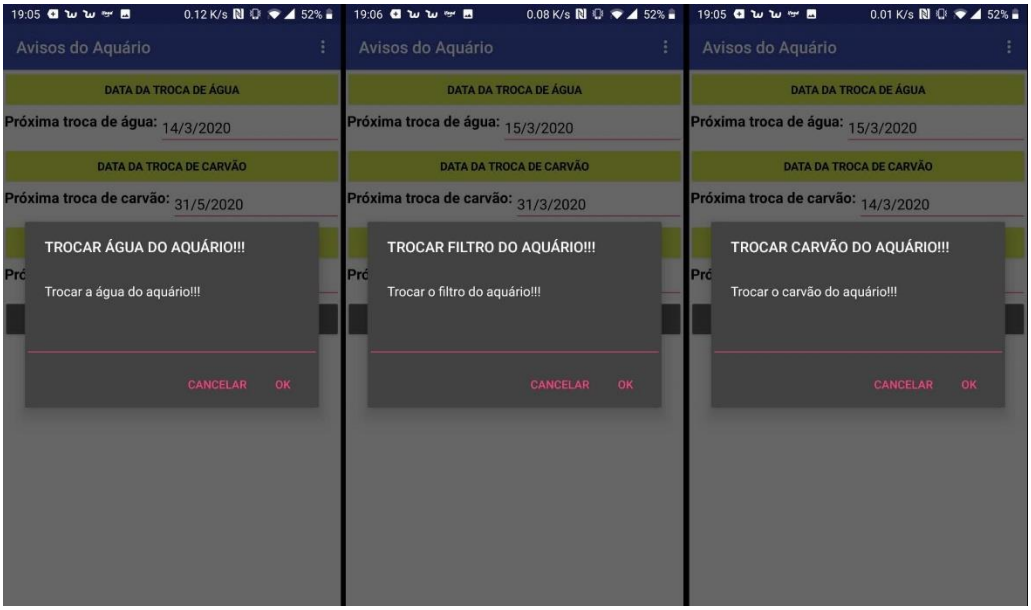

Figura 25: Alertas de vencimento de trocas.

O sistema de controle e monitoramento de aquário marinho, ainda que com um atraso de resposta de aproximadamente 8 segundos, foi capaz de aferir todos os parâmetros com precisão acima do esperado e atuar no controle dos parâmetros que nos propusemos a controlar, superando desta forma, as expectativas iniciais quando da elaboração da ideia.

O código final do projeto pode ser obtido no anexo 6, também estamos disponibilizando-o no GitHub elencado abaixo, bem como demais arquivos referentes ao projeto. Além disso, no link descrito abaixo é possível visualizar um vídeo demonstrativo da funcionalidade do protótipo.

GitHub: https://github.com/Thyago-Netto-Baltazar/SISTEMA-DE-CONTROLE-E-MONITORAMENTO-DE-AQUARIO-MARINHO.git

Link para o vídeo demonstrativo: https://youtu.be/yhSx_I89C_8

4. CONCLUSÕES

O projeto do controlador do aquário marinho representou um grande desafio, desde o início da proposta, tanto pela abrangência de diversos conteúdos, que vão desde estudos sobre biologia marinha até a programação de aplicativos voltados para IoT, quanto pela complexidade técnica dos diversos módulos de sensoriamento e controle.

Considerando o aspecto interdisciplinar do projeto aqui apresentado, procurou-se dar maior ênfase às características que melhor salientassem a proposta curricular do curso de Engenharia de Instrumentação, Automação e Robótica. Assim sendo, aspectos voltados a outras áreas foram abordados com menor profundidade.

Procurou-se cumprir com a proposta apresentada desde o início, visando ao máximo manter a originalidade, ainda que surgissem desafios, como de fato aconteceu.

Todas as etapas do desenvolvimento do projeto foram de grande aprendizado; não só pela gestão de tempo e demais recursos, mas pela superação destes desafios, buscando sempre encontrar soluções criativas. Mesmo as ideias que não foram adotadas no formato final do sistema desenvolvido, mas que em alguma etapa foram cogitadas, representaram aprendizados válidos, despertando curiosidade e interesse. Principalmente no que diz respeito a parte de programação de aplicativos, IoT, banco de dados e armazenamento em nuvem.

Como toda tecnologia, o aprimoramento é fundamental, buscando sempre o estado da arte. Nesse sentido, o projeto desenvolvido, abre horizonte para uma infinidade de aprimoramentos, os quais, se realizados, poderão resultar em um produto plausível ao mercado.

5. BIBLIOGRAFIA

[1] TREVISOL, Fabio; MORI, Marcelo Rosales; DA SILVA, Márcio José Dantas, -Sistema de controle e monitoração dos fatores determinísticos para o desenvolvimento de macrófitas aquáticas em aquários||, Trabalho de Conclusão de Curso de graduação, do Curso Superior de Tecnologia em Automação Industrial, Universidade Tecnológica Federal do Paraná, Curitiba, 2014.

[2] RIBEIRO, Felipe de Azevedo Silva; LIMA, Marco Tulio; FERNANDES, Carlos João Batista Kochenborger. (2010). Boletim ABLimno. *Panorama do mercado de organismos aquáticos ornamentais*[Online]. 38(2). Disponível em: ABLimno.

[3] ARAGÃO, Silva. (2011). A utilização de aquários e terrários como ferramenta de ensino: um olhar pelo viés da experimentação, Departamento de Ciências Biológicas, Universidade Estadual de Feira de Santana. [Online]. Disponível em: http://www2.uefs.br/semic/upload/2011/2011xv-025tha053-220.pdf

[4] GOMES, Alessandra et al. Aquário na sala de fisioterapia com crianças portadoras de necessidades especiais,|| Apresentado no Salão Internacional de Ensino, Pesquisa e Extensão, v. 9, n. 7, 2018.

[5] Como reduzir nitritos [Internet]. [acesso em 2019 Mar 30]. Disponível em: https://www.comidadecorais.com/como-reduzir-nitritos/

[6] O que é Rocha Viva? [Internet]. [acesso em 2019 Mar 30]. Disponível em: https://oaquashow.wordpress.com/reefs/o-que-e-rocha-viva/

[7] P. R. Da Silveira e W. E. Santos, Automação e controle discreto, 9ª Edição. São Paulo: Érica, 1998.

[8] S. Monk, Programação com Arduino: começando com sketches, 2ª Edição. Porto Alegre: Bookman, 2017.

[9] A. M. V. Cipelli e W. J. Sandrini, Teoria e Desenvolvimento de Projetos de Circuitos Eletrônicos, 12ª Edição. São Paulo: Érica, 1986.

[10] Figura sensor de pH [Internet]. [acesso em 28/06/2019]. Disponível em: https://www.diymore.cc/products/diymore-liquid-ph-value-detection-detect-sensor-module-monitoring-control-for-arduino-m

[11] Circuito do modulo de pH [Internet]. [acesso em 28/06/2019] . Disponível em: https://raspberrypi.stackexchange.com/questions/53421/how-i-connect-and-connect-this-ph-sensor-to-raspberry-pi-2

[12] Programa usado para desenhar o circuito [Internet]. [acesso em 19/06/2019]. Disponível em: http://fritzing.org/download/

[13]Sobre a média gaussiana: https://www.inf.ufsc.br/~andre.zibetti/probabilidade/normal.html (acesso em 10/01/2020)

6. ANEXOS

ANEXO 1 – Código para o módulo sensor de densidade com sensor ultrassônico

```
//Carrega a biblioteca do sensor ultrassonico
#include <Ultrasonic.h>
#include <LinkedList.h>
#include <Gaussian.h>
#include <GaussianAverage.h>

//Define os pinos para o trigger e echo
#define pino_trigger 27
#define pino_echo 25

//Inicializa o sensor nos pinos definidos acima
Ultrasonic ultrasonic(pino_trigger, pino_echo);
GaussianAverage Ultra_MediaGausseana(10);

void setup()
{
  Serial.begin(9600);
  Serial.println("Lendo dados do sensor...");
}

void loop()
{
  //Le as informacoes do sensor, em cm e pol
```

```
  float cmMsec, inMsec;
  long microsec = ultrasonic.timing();
  cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
  inMsec = ultrasonic.convert(microsec, Ultrasonic::IN);
  double media = cmMsec;
  Ultra_MediaGausseana += media;
  Ultra_MediaGausseana.process();
  cmMsec = Ultra_MediaGausseana.mean; //Cálculo da média gaussiana para resultados com menor flutuação.
  //Exibe informacoes no serial monitor
  Serial.print("Distancia em cm: ");
  Serial.print(cmMsec);
  Serial.print("Densidade: ");
  Serial.println(((14.64-cmMsec)*8.87)+1000); //Faz o cálculo da densidade a partir da distância, tendo como base a distância inicial, o fator multiplicativo que converte a distância informada pelo sensor em cm para a escala do densímetro em g/ml, somando 1000 para corresponder ao modo como é apresentada a densidade por aquaristas.
  delay(1000);
}
```

ANEXO 2 – Código para o módulo sensor de densidade com sensor infravermelho

```
#include "Adafruit_VL53L0X.h"
#include <Gaussian.h>
#include <GaussianAverage.h>

Adafruit_VL53L0X lox = Adafruit_VL53L0X();
//GaussianAverage Ultra_MediaGausseana(25);

//SDA PINO 21
//SCL PINO 22

void setup() {
  Serial.begin(9600);

  while (! Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}

void loop() {
```

```
  VL53L0X_RangingMeasurementData_t measure;
  Serial.print("Reading a measurement… ");
  lox.rangingTest(&measure, false);

    float reading = measure.RangeMilliMeter;
    float readingCM = (reading/10);
    //Ultra_MediaGausseana += reading;
    //Ultra_MediaGausseana.process();
    //float reading2 = Ultra_MediaGausseana.mean;
    Serial.print("Distance (cm): ");
    Serial.println(readingCM);
    Serial.print("Densidade: ");
    Serial.println((1000+(30*((readingCM-13.6)/(12.6-13.6)))));
      delay(100);
}
```

ANEXO 3 – Código para o módulo sensor e controle de temperatura

```
#include <LinkedList.h>
#include <Gaussian.h>
#include <GaussianAverage.h>

#define pinNTC 4
#define Rele_1_Resfria 23
#define Rele_2_Aquece 22
#define Rele_3_Peltier 21

GaussianAverage NTC_MediaGausseana(100);
GaussianAverage Temperatura_MediaGausseana(5);
void setup()
{
  pinMode(pinNTC ,INPUT);
  pinMode(Rele_1_Resfria, OUTPUT);
  digitalWrite(Rele_1_Resfria, HIGH);
  pinMode(Rele_2_Aquece, OUTPUT);
  digitalWrite(Rele_2_Aquece, HIGH);
  pinMode(Rele_3_Peltier, OUTPUT);
  digitalWrite(Rele_3_Peltier, HIGH);
    Serial.begin(9600);
}

void loop()
{
```

```
int sensorNTC = analogRead(pinNTC);

NTC_MediaGausseana += sensorNTC;

NTC_MediaGausseana.process();

int sensor = NTC_MediaGausseana.mean;

float tensaoResistor = 0.19 + (sensor*(3.33/4095));// 0.19 é o erro entre tensão física e calculada pelo ESP32, 3.33 o valor de tensão fornecido pelo ESP e 4095 a resolução.

float corrente = (tensaoResistor/9890.0)*1000;

float tensaoSensor = 3.33 - (tensaoResistor);

long resistencia = (tensaoSensor/corrente)*1000;

double Temp = log(resistencia); // Salvamos o Log(resistance) em Temp (de Temporário) para não precisar calcular 4 vezes depois.

Temp = 1 / (0.00028386514 + ((0.000348586482) * Temp) + (-0.000000186357303 * Temp * Temp * Temp));   // Agora Temp recebe o valor de temperatura calculado por Steinhart-Hart.

Temp = Temp - 273.15;  // Convertendo Kelvin em Celsius

double Temperatura = Temp;

Temperatura_MediaGausseana += Temperatura;//Realizando a média Gausseana para diminuir flutuações.

Temperatura_MediaGausseana.process();

Temp = Temperatura_MediaGausseana.mean;


Serial.print("Tensão Sensor: ");

Serial.print(tensaoSensor);

Serial.println(" V");

Serial.print("Tensão Resistor: ");

Serial.print(tensaoResistor);

Serial.println(" V");

Serial.print("Resistência: ");

Serial.print(resistencia);

Serial.println(" ohms");

Serial.print("Corrente: ");

Serial.print(corrente);
```

```
Serial.println(" mA");
Serial.println("");
Serial.print("Sensor: ");
Serial.print(sensor);

Serial.println("");

Serial.print("Temperatura: ");
Serial.print(Temp);
Serial.println("°C");
Serial.println("");

 if (Temp >= 28)
   {
    digitalWrite(Rele_2_Aquece, HIGH);//Resfriando a água
    digitalWrite(Rele_1_Resfria, LOW);
    digitalWrite(Rele_3_Peltier, LOW);
   }
 else if (Temp >= 25)
   {
    digitalWrite(Rele_1_Resfria, HIGH);//Desativa na faixa escolhida
    digitalWrite(Rele_3_Peltier, HIGH);
    digitalWrite(Rele_2_Aquece, HIGH);
   }
 else
   {
    digitalWrite(Rele_1_Resfria, HIGH);//Aqecendo a temperatura
    digitalWrite(Rele_2_Aquece, LOW);
    digitalWrite(Rele_3_Peltier, LOW);
   }
```

```
   delay(1000);
}
```

ANEXO 4 – Código para o módulo do sensor de nível do reservatório de reposição de água doce.

```
#define pin_sensor 32
void setup() {
Serial.begin(9600);
pinMode(pin_sensor, INPUT);
}

void loop() {
  int valor_s1 = digitalRead(pin_sensor);
  Serial.print(valor_s1);
    delay(1000);
}
```

ANEXO 5 – Código para o módulo do sensor de pH e obtenção do horário local via protocolo NTP.

```
//Bibliotecas usadas
#include <LinkedList.h>
#include <Gaussian.h>
#include <GaussianAverage.h>
#include <NTPClient.h> //Biblioteca do NTP.
#include <WiFi.h> //Biblioteca do WiFi.
#include <LiquidCrystal.h>  //Inclui biblioteca para display lcd

//-------- Configurações de Wi-fi-----------
const char* ssid = "nome da rede";
const char* password = "senha";

WiFiUDP udp;
NTPClient ntp(udp, "a.st1.ntp.br", -3 * 3600, 60000);//Cria um objeto "NTP" com as configurações.
//Definição da pinagem utilizada no display lcd
const int rs = 13, en = 12, d4 = 17, d5 =16, d6 = 2, d7 = 15;//Cria constantes do tipo inteiro
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);//Atribui ao objeto "lcd" constantes as quais definem seus pinos de ligação
//GPIOs usados
#define sensor_pH 34

//Define o número de amostras realizadas para o cálculo da média
GaussianAverage pH_MediaGausseana(100);
void setup() {
```

```
  //Define o sensor de pH como entrada
pinMode(sensor_pH ,INPUT);

// initialize serial communication at 115200 bits per second:
Serial.begin(115200);

WiFi.begin(ssid, password);
  delay(2000);               // Espera a conexão.
  ntp.begin();               // Inicia o NTP.
  ntp.forceUpdate();         // Força o Update.

lcd.begin(40,2); //Define o display com 40 colunas e 2 linhas
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:

  //Armazena na variável HORA, o horário atual.
  horario = ntp.getFormattedTime();

//Lê entrada analógica do sensor de pH
int sensor_pH_Value = analogRead(sensor_pH);
pH_MediaGausseana += sensor_pH_Value;
pH_MediaGausseana.process();

// Converte o sinal analógico na entrada do ESP32 (0 - 4095) para pH(0 -14) que varia com a tensão de entrada (0 - 3.3V):
float pH = pH_MediaGausseana.mean * (14 / 4095.0);
float pH_semfiltro = sensor_pH_Value * (14 / 4095.0);
```

```
//Imprime na serial uma tabela e/ou plota um gráfico com a leitura de cada um dos sensores e o respectivo horário da leitura

//Imprime horário

Serial.print(horario);

Serial.print(" ");

//Serial.print("PH: ");

Serial.print(pH);

Serial.print(" ");

//Serial.print("PH2: ");

Serial.println(pH_semfiltro);

delay(800);

}
```

ANEXO 6 – Código Final do Projeto.

```
////////////////////////////////////////////////////////////////
//UNIVERSIDADE FEDERAL DO ABC                                 //
//TRABALHO DE GRADUAÇÃO                                       //
//ENGENHARIA DE INSTRUMENTAÇÃO AUTOMAÇÃO E ROBÓTICA           //
//ORIENTADOR: PROF. DR. FILIPE IEDA FAZANARO                  //
//ALUNOS:                                                     //
//THYAGO NETTO BALTAZAR          RA:11078011                  //
//RENAN TADEU BALDINI DE BRITO   RA:11070113                  //
//                                                            //
//AUTOMAÇÃO DE AQUÁRIO MARINHO                                //
////////////////////////////////////////////////////////////////

//Bibliotecas usadas
#include <LinkedList.h>
#include <Gaussian.h>
#include <GaussianAverage.h>
#include <NTPClient.h> //Biblioteca do NTP.
#include <WiFi.h> //Biblioteca do WiFi.
#include <LiquidCrystal.h>  //Inclui biblioteca para display lcd
#include <FirebaseESP32.h>
#include "Adafruit_VL53L0X.h"//Inclui biblioteca para sensor de distância Infravermelho


//-------- Configurações de Wi-fi-----------
//const char* ssid = "wi-fi-baldinis";
```

```
//const char* password = "10021993alan";

char* ssid = "CiaBaltazar";

char* password = "4815162342";

//-------- Configurações de relógio on-line----------

WiFiUDP udp;

NTPClient ntp(udp, "a.st1.ntp.br", -3 * 3600, 60000);//Cria um objeto "NTP" com as configurações.


//Definição da pinagem utili ózada no display lcd


//LiquidCrystal lcd(13,12,16,4,2,15); //(RS,EN,D4,D5,D6,D7)

const int rs = 13, en = 12, d4 = 17, d5 =16, d6 = 2, d7 = 15;//Cria constantes do tipo inteiro

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);//Atribui ao objeto "lcd" constantes as quais definem seus pinos de ligação


//Array simbolo grau

byte grau[8] ={ B00001100,

        B00010010,

        B00010010,

        B00001100,

        B00000000,

        B00000000,

        B00000000,

        B00000000,};


//GPIOs usados

#define sensor_pH 34

#define led 2 // Define o LED ao pino D4.

#define Rele_1 23 //Relé da luminária branca

#define Rele_2 32 //Relé da luminária azul

//NÃO USAR pinos 21 e 22, são usados pelo sensor de distância infravermelho
```

```
#define pinNTC 35 //
#define Rele_8_Resfria 5   //Relé que aciona a bomba do circuito de resfriamento de água
#define Rele_7_Aquece 18  //Relé que aciona a bomba do circuito de aquecimento de água
#define Rele_5_Peltier 19 //Relé que aciona a pastilha Peltier
#define Rele_4 4 //Relé de ajuste de nível
#define pin_sensor 14


//Variáveis globais
String horario;           // Variável que armazenara a hora
float temperatura;
float pH;
float densidade;
boolean luz_branca;
boolean luz_azul;
boolean valor_s1;
String horario_troca_agua;
//String data_troca_agua;

/*Cria duas variaveis, para o host e para a senha do firebase*/
#define Host "https://aquario-ufabc.firebaseio.com/"//seu host
#define Senha_Fire "IRqOpPXfIrRkwTYIH1UlOPehkCcYc72DAX7anFy1"//sua chave do firebase

//Define o número de amostras realizadas para o cálculo da média
GaussianAverage pH_MediaGausseana(100);
GaussianAverage NTC_MediaGausseana(100);
GaussianAverage Temperatura_MediaGausseana(5);

//Inicializa Sensor de distância infravermelho
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
```

```
void setup() {

//Define sensor de nivel como entrada
pinMode(pin_sensor, INPUT);

//Define o sensor de pH como entrada
pinMode(sensor_pH ,INPUT);

// initialize serial communication at 115200 bits per second:
Serial.begin(115200);

lcd.begin(40,2); //Define o display com 40 colunas e 2 linhas
//Executa rotina de inicialização
inicializacao();

//Inicializa conexão WIFI
WiFi.begin(ssid, password);
//Enquanto o status do WIFI for diferente de conectado, escreva na serial "Conectando ao WIFI..."
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Conectando ao WIFI..");
    lcd.clear(); //Limpa o display
    lcd.print("Conectando ao WIFI..");
  }

  //Escreve na serial que o WIFI está conectado
  Serial.println("WIFI Conectado");
  lcd.clear(); //Limpa o display
```

```
  lcd.print("WIFI Conectado!");
  delay(1000);
  lcd.clear(); //Limpa o display

  delay(200);
  ntp.begin();              // Inicia o NTP.
  ntp.forceUpdate();        // Força o Update.
  Firebase.begin(Host, Senha_Fire);//Conecta-se ao Firebase
  delay(200);

//Cria o caractere customizado com o simbolo do grau
lcd.createChar(0, grau);

//Cria condições para executar as leituras do sensor de distância infravermelho
  Serial.begin(9600);

  while (! Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));

  //Setup do módulo de temperatura
  pinMode(pinNTC ,INPUT);
  pinMode(Rele_8_Resfria, OUTPUT);
  digitalWrite(Rele_8_Resfria, HIGH);
```

```
  pinMode(Rele_7_Aquece, OUTPUT);
  digitalWrite(Rele_7_Aquece, HIGH);
  pinMode(Rele_5_Peltier, OUTPUT);
  digitalWrite(Rele_5_Peltier, HIGH);
    Serial.begin(115200);

  //Setup módulo Sensor de Nível
  Serial.begin(9600);
  pinMode(pin_sensor, INPUT);

  //Portas Luz OUTPUT
  pinMode(Rele_1, OUTPUT);
  pinMode(Rele_2, OUTPUT);

}//Fim do setup

//Funções
//Lê e retorna variáveis do tipo "string" enviadas pela comunicação serial
String leStringSerial(){
  String conteudo = "";
  char caractere;

  // Enquanto receber algo pela serial
  while(Serial.available() > 0) {
    // Lê byte da serial
    caractere = Serial.read();
    // Ignora caractere de quebra de linha
    if (caractere != '\n'){
      // Concatena valores
      conteudo.concat(caractere);
```

```
    }
    // Aguarda buffer serial ler próximo caractere
    delay(10);
  }

    //Testa função "leStringSerial"

 // Serial.print("Recebi: ");
 //Serial.println(conteudo);

  return conteudo;
}//Fim da função leStringSerial

//Função mensagem de inicialização do display lcd
void inicializacao(){
                lcd.clear(); //Limpa o display
                lcd.print("UFABC- TG ENGENHARIA IAR/2020");
                lcd.setCursor(0,2);
                lcd.print("AUTOMACAO DE AQUARIO MARINHO");
                delay(3000);
                lcd.clear(); //Limpa o display
                lcd.print("Professor Orientador");
                lcd.setCursor(0,2);
                lcd.print("Dr. Filipe Ieda Fazanaro");
                delay(3000);
                lcd.clear(); //Limpa o display
                lcd.print("Desenvolvido por...");
                delay(1000);
                lcd.clear(); //Limpa o display
                lcd.print("Thyago Netto Baltazar");
```

```
            lcd.setCursor(0,2);
            lcd.print("Renan Tadeu Baldini de Brito");
            delay(3000);
            lcd.clear(); //Limpa o display
        }// Fim da função de inicialização do lcd


// the loop routine runs over and over showing the voltage on A0

void loop() {

//Loop sensor de nivel
valor_s1 = digitalRead(pin_sensor);
 // Serial.print(valor_s1);
   delay(10);

//Armazena na variável HORA, o horário atual.
  horario = ntp.getFormattedTime();
 // Serial.println(horario);
 delay(10);

//Lê entrada analógica do sensor de pH
int sensor_pH_Value = analogRead(sensor_pH);
pH_MediaGausseana += sensor_pH_Value;
pH_MediaGausseana.process();

// Convert the analog reading (which goes from 0 - 4095) to a voltage (0 - 3.3V):
 pH = pH_MediaGausseana.mean * (14 / 4095.0);
float pH_semfiltro = sensor_pH_Value * (14 / 4095.0);
```

```
//variáveis criadas apenas como exemplo para mostrar parametros no display lcd
//float t = 0.00; // valor da temperatura
//float d = 0.00; // valor da densidade
//temperatura = 0.00; //valor da temperatura


int sensorNTC = analogRead(pinNTC);
   NTC_MediaGausseana += sensorNTC;
   NTC_MediaGausseana.process();
   int sensor = NTC_MediaGausseana.mean;
   float tensaoResistor = 0.19 + (sensor*(3.27/4095));// 0.19 é o erro entre tensão física e calculada pelo ESP32, 3.33 o valor de tensão fornecido pelo ESP e 4095 a resolução.
   float corrente = (tensaoResistor/9890.0)*1000;
   float tensaoSensor = 3.27 - (tensaoResistor);
   float resistencia = abs((tensaoSensor/corrente)*1000);
   float Temp = log(resistencia); // Salvamos o Log(resistance) em Temp (de Temporário) para não precisar calcular 4 vezes depois.
   Temp = 1 / (0.00028386514 + ((0.000348586482) * Temp) + (-0.000000186357303 * Temp * Temp * Temp));   // Agora Temp recebe o valor de temperatura calculado por Steinhart-Hart.
   Temp = Temp - 273.15;  // Convertendo Kelvin em Celsius
   float Temperatura = Temp;
   Temperatura_MediaGausseana += Temperatura;//Realizando a média Gausseana para diminuir flutuações.
   Temperatura_MediaGausseana.process();
   Temp = Temperatura_MediaGausseana.mean;
   temperatura = Temp;


//Controle de temperatura
    if (Temp >= 28)
      {
        digitalWrite(Rele_7_Aquece, HIGH);//Resfriando a água
        digitalWrite(Rele_8_Resfria, LOW);
        digitalWrite(Rele_5_Peltier, LOW);
```

```
      }
    else if (Temp >= 25)
      {
       digitalWrite(Rele_8_Resfria, HIGH);//Desativa na faixa escolhida
       digitalWrite(Rele_5_Peltier, HIGH);
       digitalWrite(Rele_7_Aquece, HIGH);
      }
    else
      {
        digitalWrite(Rele_8_Resfria, HIGH);//Aqecendo a temperatura
        digitalWrite(Rele_7_Aquece, LOW);
        digitalWrite(Rele_5_Peltier, LOW);
      }

  delay(1000);

//Leitura da desnsidade
  VL53L0X_RangingMeasurementData_t measure;
  Serial.print("Reading a measurement… ");
  lox.rangingTest(&measure, false);

   float reading = measure.RangeMilliMeter;
   float readingCM = (reading/10);
   //Ultra_MediaGausseana += reading;
   //Ultra_MediaGausseana.process();
   //float reading2 = Ultra_MediaGausseana.mean;
   densidade = ((1000+(30*((readingCM-13.6)/(12.6-13.6)))));
    delay(100);

//Mostrando valores dos parâmetros no display LCD
```

```
//Carregango valores nas variaveis para o firebase e display LCD
  Firebase.setFloat("aquario-ufabc/temperatura", temperatura);
  Firebase.setFloat("aquario-ufabc/pH", pH);
  Firebase.setFloat("aquario-ufabc/densidade", densidade);

//Firebase.setBool("aquario-ufabc/luz_branca",luz_branca ); //cria variável "luz_branca" no firebase
  luz_branca = Firebase.getBool("aquario-ufabc/luz_branca");//recebe do firebase o valor da variável "luz_branca"
  delay(50);
  digitalWrite(Rele_1, luz_branca );//iguala o estado do relé 1 ao estado da variável "luz_branca"
  //Serial.print("Luminária branca: ");
 // Serial.print(luz_branca);

  //if(luz_branca == 0){digitalWrite(Rele_1, 0 );}
  //else if(luz_branca == 1){digitalWrite(Rele_1, 1 );}

//Firebase.setBool("aquario-ufabc/luz_azul",luz_azul ); //cria variável "luz_azul" no firebase
  luz_azul = Firebase.getBool("aquario-ufabc/luz_azul");//recebe do firebase o valor da variável "luz_azul"
  delay(50);
  digitalWrite(Rele_2, luz_azul);//iguala o estado do relé 2 ao estado da variável  "luz_azul"
  //Serial.print("Luminária azul: ");
  //Serial.print(luz_azul);

  Firebase.setBool("aquario-ufabc/nivel",valor_s1 ); //envia para o firebase o valor da variável "valor_s1" para variável "nivel"(do firebase)
 // valor_s1 = Firebase.getBool("aquario-ufabc/nivel");//recebe do firebase o valor da variável "nivel"
  delay(50);
  digitalWrite(Rele_4, valor_s1);//iguala o estado do relé 4 ao estado da variável "valor_s1"
```

```
  //Data e Horário de Alarme de troca de água

  //Firebase.setString("aquario-ufabc/horario_troca_agua", horario_troca_agua);//envia para o firebase o valor da variável "horario_troca_agua"

 // Firebase.setString("aquario-ufabc/data_troca_agua", data_troca_agua);       //envia para o firebase o valor da variável 'horario_troca_agua"


//Escreve os valores dos parametros no display lcd

lcd.begin(40,2); //Define o display com 40 colunas e 2 linhas

lcd.clear(); //Limpa o display

lcd.setCursor(0,0);

lcd.print("Temp : ");

lcd.print(" ");

lcd.setCursor(7,0);

lcd.print(temperatura,2);

lcd.setCursor(12,0);


//Mostra o simbolo do grau formado pelo array

lcd.write((byte)0);

lcd.setCursor(13,0);

lcd.print("C");


//Mostra o simbolo do grau quadrado

//lcd.print((char)223);


lcd.setCursor(20,0);

lcd.print("pH : ");

lcd.print(" ");

lcd.setCursor(27,0);

lcd.print(pH,2);
```

```
lcd.setCursor(0,1);
lcd.print("Dens : ");
lcd.print(" ");
lcd.setCursor(7,1);
lcd.print(densidade,2);
lcd.setCursor(12,1);
lcd.print("g/mL");

lcd.setCursor(20,1);
lcd.print("Hora:");
lcd.print(" ");
lcd.setCursor(27,1);
lcd.print(horario);

}//Fim do loop
```