UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FISHMING – SISTEMA PARA CONTROLE E MONITORAMENTO DE PEIXES DOMÉSTICOS POR APLICATIVO MOBILE

PHILIP STEFAN HAERTEL

BLUMENAU
2017

PHILIP STEFAN HAERTEL

FISHMING – SISTEMA PARA CONTROLE E MONITORAMENTO DE PEIXES DOMÉSTICOS POR APLICATIVO MOBILE

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer – Mestre - Orientador

BLUMENAU
2017

FISHMING – SISTEMA PARA CONTROLE E MONITORAMENTO DE PEIXES DOMÉSTICOS POR APLICATIVO MOBILE

Por

PHILIP STEFAN HAERTEL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer – Mestre - Orientador, FURB
Membro:	Prof. Gabriele Jennrich Bambineti, Especialista – FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB

Blumenau, 30 de junho de 2017

Dedico este trabalho a meus familiares e amigos que sempre me apoiaram e incentivaram para a conclusão do meu curso de graduação.

AGRADECIMENTOS

A Deus por minha vida, família e amigos.

À minha família, que sempre me apoiou em todos os momentos.

À minha namorada Karolyne Tottene, pelo apoio e parceria nessa fase de nossas vidas.

Ao meu orientador, Miguel Alexandre Wisintainer pelo apoio no desenvolvimento do trabalho.

Aos meus colegas de trabalho Alex Sandro da Silva e Juliano Piccoli, que me auxiliaram durante o meu curso de graduação.

Aos meus amigos e colegas de curso Camila, Daniel, Filipe, Jean, Juliano e Luís que tornaram esses quatro anos de estudos mais agradáveis e divertidos.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“Dinheiro é só papel, não vai para a sepultura. No fim é só herdeiro brigando por suas indústrias. ”

Carlos Eduardo Taddeo

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema para controle e monitoramento de peixes domésticos por meio de aplicativo mobile. O sistema, batizado de Fishming, tem como objetivo realizar a captura de informações da água e do ambiente por meio de sensores e também controlá-lo por meio de atuadores. Foi construído utilizando como componentes principais um sensor de temperatura a prova d'água, um sensor de pH, um sensor de fluxo de água, um sensor de temperatura e umidade, um sensor de luminosidade, um módulo relê e um módulo baseado no chip ESP8266, com Wi-Fi integrado. O dispositivo conecta-se a rede Wi-Fi e envia regularmente os valores dos sensores para o serviço ThingSpeak, responsável pelo armazenamento dos dados. A visualização dos dados e o controle do ambiente são feitos por um aplicativo mobile. Para codificar o hardware foi utilizado o framework Sming com a linguagem de programação C++ e para codificar o aplicativo mobile o framework Ionic com a linguagem Typescript. Após o término do desenvolvimento, foram realizados testes com todos os sensores e constatou-se que os dados estão muito próximos da realidade. Por fim, o trabalho proposto atingiu os objetivos listados, mostrando-se como um sistema relevante para o controle e monitoramento de peixes domésticos.

Palavras-chave: Criação de peixes domésticos. Internet das coisas. ESP8266. Sensores. Atuadores. Smartphones. Sming. Ionic. ThingSpeak.

ABSTRACT

This work presents the development of a system for controlling and monitoring domestic fishes through a mobile application. The system, called Fishming, has as an objective to capture information of water and environment through sensors and also control it through actuators. It was constructed using a waterproof temperature sensor, a pH sensor, a water flow sensor, a temperature and humidity sensor, a light sensor, a relay module and a ESP8266 based module. The device connects to the Wi-Fi network and regularly sends the sensor values to the ThingSpeak service for storing the data. Data visualization and control of the environment are done by a mobile application. To code the hardware was used the Sming framework with the C++ programming language and to code the mobile application Ionic framework with Typescript language. After the end of the development, tests were performed with all sensors and it was verified that the data are very close to the reality. Finally, the proposed work reached the objectives listed, showing itself as a relevant system for the control and monitoring of domestic fishes.

Key-words: Home fishes creation. Internet of Things. ESP8266. Sensors. Actuators. Smartphones. Sming. Ionic. ThingSpeak.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP – Access Point

BSSID - Basic Service Set Identifier

EEPROM - Electrically-Erasable Programmable Read-Only Memory

HTTP - Hypertext Transfer Protocol

IDE - Integrated Development Environment

IOT - Internet of Things

IP - Internet Protocol

JSON - Notação de Objetos JavaScript

L/h – Litros por hora

NO - Normally Open

RF – Requisito Funcional

RNF – Requisito Não Funcional

SDK - Kit de Desenvolvimento de Software

SoC - System on Chip

SSID - Service Set Identifier

UDP - User Datagram Protocol

UML - Unfield Modeling Language

UV - Ultravioleta

SUMÁRIO

1 INTRODUÇÃO

Atualmente, existe uma crescente revolução tecnológica em que vários objetos domésticos estão ligados a rede mundial de computadores. A cada dia surgem novos objetos conectados a internet ou a outros dispositivos, como computadores e smartphones. Desde itens mais comuns como lâmpadas conectadas com o celular, até geladeiras, ar condicionados e TVs inteligentes.

Um termo muito comentado hoje em dia é "Internet das Coisas", que foi proposto por Kevin Ashton do MIT em 1999 (ASHTON, 2014). Segundo Santaella (2013) Internet das coisas ou Internet of Things (IOT) corresponde à fase atual da internet em que os objetos se relacionam com objetos humanos e animais os quais passam a ser objetos portadores de dispositivos computacionais capazes de conexão e comunicação. Em 2014, o número de desenvolvedores para IOT era de 300 mil. Estima-se que esse número irá aumentar drasticamente até 2020, ano em que será necessário cerca de 4.5 milhões de desenvolvedores para IOT. (ASAY, 2014)

Segundo Muratori e Dal Bó (2011), automatização residencial é o conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação. Levando-se em conta essa busca por automação residencial, surgiram vários projetos que auxiliam no controle e criação de animais domésticos, como alimentadores automáticos e sistemas para monitoramento controlados pelo celular.

Diante do exposto, com o objetivo de realizar uma automatização de tarefas e um monitoramento para peixes domésticos, este trabalho apresenta o desenvolvimento de um sistema em que o usuário consiga controlar e monitorar um aquário ou lago, por meio de aplicativo móvel.

1.1 OBJETIVOS

Este trabalho tem como objetivo desenvolver um sistema para controle e monitoramento de peixes domésticos por meio de aplicativo móvel.

Os objetivos específicos são:

a) desenvolver um hardware para controlar e monitorar os peixes utilizando o módulo ESP8266;
b) desenvolver um aplicativo para a plataforma Android, a ser utilizado pelos usuários para realizar a interação com o hardware.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O primeiro capítulo apresenta a introdução e os objetivos do trabalho. O segundo capítulo apresenta a fundamentação teórica sobre os temas abordados no trabalho, como: animais domésticos e criação de peixes, internet das coisas, ESP8266, Smart Config e também sensores e atuadores. O terceiro capítulo mostra os principais pontos do desenvolvimento do trabalho como: requisitos, especificação, implementação e resultados obtidos. Por último, o quarto capítulo relata as conclusões e sugere extensões para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 oferece uma visão geral sobre animais domésticos e a criação de peixes. A seção 2.2 expõe brevemente sobre o termo Internet das Coisas. A seção 2.3 aborda o módulo ESP8266 e justifica sua importância para neste trabalho. A seção 2.4 faz uma introdução sobre a tecnologia Smart Config. A seção 2.5 apresenta os conceitos de sensores e atuadores. Por fim, a seção 2.6 traz um breve estudo de três trabalhos correlatos.

2.1 ANIMAIS DOMÉSTICOS E CRIAÇÃO DE PEIXES

A preocupação com a criação de animais domésticos vem tendo uma importância crescente. Atualmente as pessoas não estão satisfeitas apenas em tratar seus animais de estimação como animais de estimação, mas querem dar condições humanas a eles. A relação entre os animais domésticos e seus donos sempre esteve presente, mas está se tornando mais comum devido ao crescimento do comércio que envolve produtos para animais de estimação. (CARVALHO; PESSANHA, 2012).

Uma das atividades que está em grande crescente no Brasil, é a criação de peixes domésticos. Alguns comerciantes que trabalham com venda de aquários personalizados estão crescendo cerca de 20% ao ano, com aumento de 50% somente em 2013 (GLOBO, 2014). Por conta da demanda na atividade de criação de peixes ornamentais, em 2017 foi criado um centro para estudar as necessidades desse tipo de produção. (GLOBO, 2017).

Com a grande expansão do mercado, também surgem necessidades de soluções de monitoramento para facilitar a criação e auxiliar nos desafios diários para manter em harmonia o ambiente do peixe. Existem desde soluções completas até soluções mais simples, dependendo das necessidades do cliente (ROMANI, 2015). É comum encontrar soluções comerciais ou acadêmicas com o intuito de realizar alimentação automática para os animais domésticos (DESSBESELL, 2014).

2.2 INTERNET DAS COISAS

Internet das coisas é uma rede crescente de objetos do cotidiano que podem compartilhar informações e realizar tarefas (SAS, 2016). O principal conceito é a conectividade entre esses dispositivos com a internet, desde eletrodomésticos até outros equipamentos. Assim, com o mundo cada vez mais conectado, não apenas as pessoas se comunicam, mas também os objetos se "comunicam" uns com os outros. Ou seja, o termo Internet das coisas é a conexão de todos estes objetos para otimizar a vida das pessoas e das organizações (SANTAELLA et al., 2013).

Até pouco tempo atrás, não era comum acessar internet na TV, porém hoje em dia são poucas as TVs que não são conectadas com a internet. Esse é o futuro de todos os dispositivos, onde todos estarão conectados a internet, e será possível desde configurar a cafeteira pelo celular ou até mesmo configurar as máquinas de uma empresa por meio de um computador.

A automatização residencial está em constante crescente devido sua popularização e diminuição dos preços da tecnologia. Esse crescimento também ocorre pois existe uma nova geração de pessoas mais exigente na automatização de tarefas residenciais, e buscam isso ao adquirir seu primeiro imóvel. (MURATORI; DAL BÓ, 2011). Existem vários módulos disponíveis no mercado com conexão a internet que podem ser utilizados com Internet das Coisas e neste trabalho foi utilizado o módulo ESP8266.

2.3 MÓDULO ESP8266

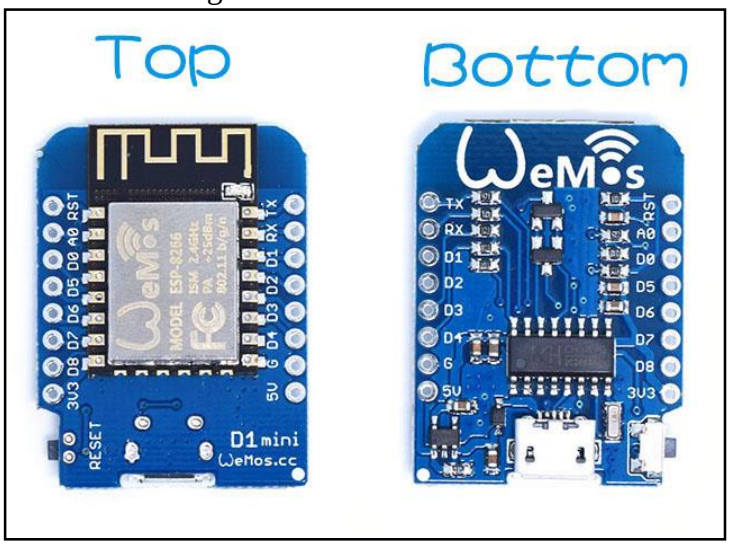
Microcontrolador é um circuito integrado capaz de executar programas (KOLBAN, 2015). Um dos mais conhecidos é o Arduino, por conta de sua arquitetura aberta que o permite ser utilizado em inúmeros tipos de projetos. Porém, assim como o Arduino, muitos microcontroladores não possuem acesso à internet sem a necessidade de um hardware complementar (KOLBAN, 2015).

O ESP8266 é um System on Chip (SoC) de baixo consumo de energia e alto desempenho com suporte a conexão wireless que está de acordo com o padrão IEEE802.11bgn (ESPRESSIF SYSTEMS, 2013, p. 4). Mesmo lançado inicialmente com quase nenhuma documentação em inglês, o produto fez um grande sucesso devido ao seu baixo preço, menos de 10 dólares. O que chama a atenção neste módulo são 2 principais fatores, o seu tamanho e seu preço. O módulo tem tamanho semelhante a uma moeda, e atualmente é vendido em sites chineses por cerca de 2 dólares (ALIEXPRESS, 2016).

No final de 2014 a Espressif lançou um Kit de Desenvolvimento de Software (SDK) que permitiu que o módulo fosse programado diretamente, eliminando a necessidade de um microcontrolador (BENCHOFF, 2014).

O módulo D1 Mini da WEMOS é um facilitador para prototipação pois também possui uma entrada de 5V, um botão para realizar o reset, uma memória flash de 4MB e uma conexão USB para enviar o código. É possível visualizá-lo na Figura 1.

Figura 1 - WEMOS D1 mini

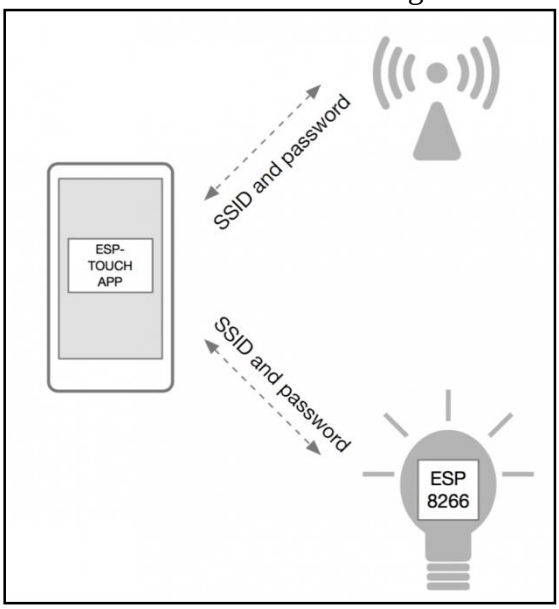


Fonte: Zonamaker (2017).

2.4 SMART CONFIG

Smart Config é uma tecnologia para auxiliar usuários a conectar um dispositivo a uma rede Wi-Fi através de uma simples configuração pelo smartphone. O chip ESP8266 implementa o protocolo ESP-TOUCH que realiza essa configuração (ESPRESSIF SYSTEMS, 2017). A Figura 2 demonstra o funcionamento dessa tecnologia.

Figura 2 - Funcionamento da tecnologia Smart Config



Fonte: Espressif Systems (2017).

Inicialmente o ESP8266 não está conectado à rede (mas consegue monitorar o tráfego), por isso o aplicativo não consegue enviar informações diretamente para o dispositivo. Em vez disso, o aplicativo envia pacotes User Datagram Protocol (UDP) para o Access Point (AP) e mesmo que ele não esteja interessado no conteúdo, basta que os pacotes sejam visíveis na rede.

O ESP8266 consegue monitorar o tráfego, mas não consegue descriptografá-lo. Porém a criptografia Wi-Fi afeta o tamanho dos pacotes de forma consistente, ou seja, adiciona X bytes adicionais ao tamanho de cada pacote, onde X sempre é constante. Com base nisso, o aplicativo codifica o Service Set Identifier (SSID) e a senha no comprimento de uma sequência de pacotes UDP, então o chip pode ver o tamanho dos pacotes criptografados e realiza a leitura das informações pelo tamanho dos pacotes (HAWKINS, 2017).

2.5 SENSORES E ATUADORES

Um sensor permite a transformação de um sinal, ou outra variável física, para um sinal que pode ser utilizado de forma mais eficiente pelo sistema que implementa o sensor. Sensores podem ser classificados como analógicos ou discretos. Um sensor analógico produz um sinal analógico cujo valor continuo varia de um modo análogo com a variável a ser medido. Um sensor que é discreto produz uma saída que só pode ter certos valores, binário ou digital. Um dispositivo binário produz um ou dois valores, por exemplo on e off. Um dispositivo digital produz um sinal de saída digital como um conjunto de bits paralelos ou uma série de impulsos que podem ser quantificados. Sensores digitais estão se tornando mais comuns devido à sua compatibilidade com sistemas de computação e sua facilidade de utilização (GROOVER, 2008).

Um atuador converte o sinal de comando do controlador para uma mudança de um parâmetro físico. Esta mudança é geralmente uma alteração mecânica, tal como mudança de posição ou uma alteração na velocidade. Muitos atuadores estão equipados com amplificadores, para converter sinais de baixo nível para nível em sinais fortes o suficiente para acionar o atuador. Três tipos de atuadores podem sem definidos: elétrico, hidráulico e pneumático. Os atuadores elétricos incluem motores elétricos de todos os tipos e motores de passo. Os atuadores hidráulicos incluem uma ampla variedade de dispositivos cilíndricos comprimindo fluídos hidráulicos para alcançar operação. Os atuadores pneumáticos incluem uma variedade de pistão e dispositivos cilíndricos que comprimem ar ou outros gases para alcançar mudanças físicas nas variáveis. (GROOVER, 2008).

2.6 TRABALHOS CORRELATOS

A seguir estão relacionados três trabalhos correlatos ao proposto. O item 2.6.1 detalha o Seneye. O item 2.6.2 detalha o Insight, desenvolvido pela Puratek. O item 2.6.3 detalha o Apex AquaController, desenvolvido pela Neptune.

2.6.1 Seneye

Seneye é um dispositivo para monitoramento do aquário que disponibiliza ao usuário informações referentes ao mesmo, como temperatura da água, pH da água e luminosidade. O Seneye permite que os detentores de peixes possam entender o que acontece dentro de seu aquário ou lago, monitorando parâmetros nocivos que não são detectáveis ao olho humano (SENEYE, 2016). Possui três versões, Seneye Home, Seneye Pound e Seneye Reef.

O dispositivo é compacto e tem apenas uma saída USB para conectar ao computador, conforme pode ser visto na Figura 3. Para utilizá-lo, é necessário colocar o dispositivo dentro da água do aquário e realizar a conexão com o computador por meio de uma porta USB, para então acessar o sistema e visualizar a dashboard com as informações, ilustrada na

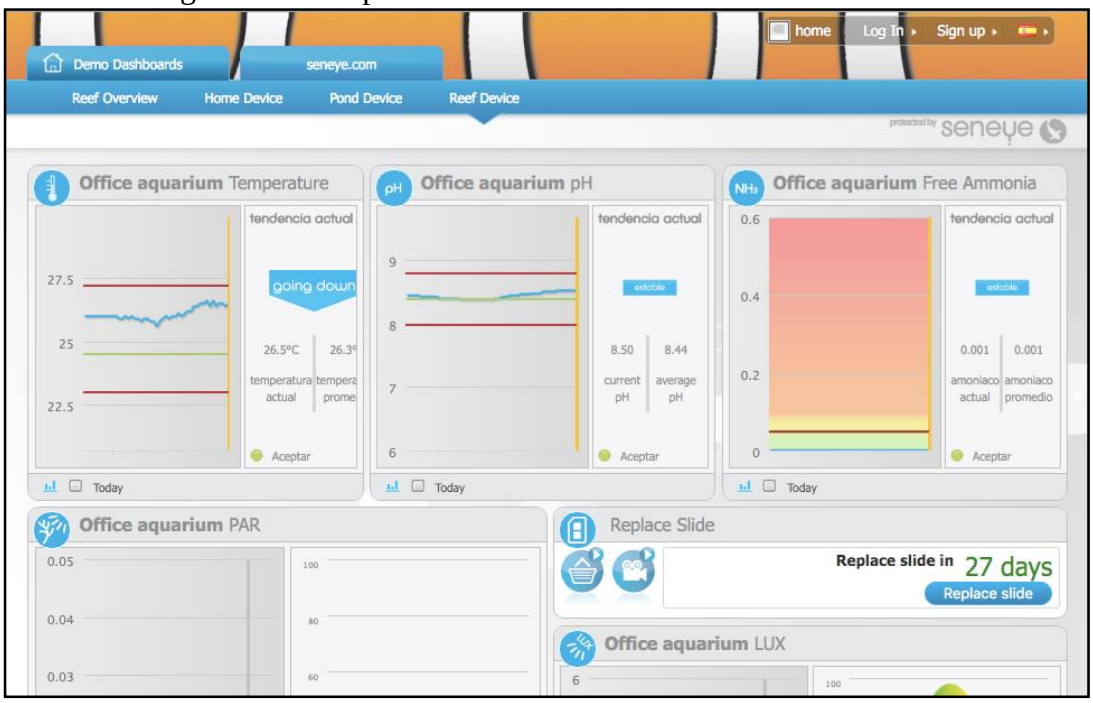
Figura 4.

Figura 3 - Dispositivo Seneye Reef

Fonte: Seneye (2016).

Através do sistema acessado pelo computador, o usuário pode descobrir informações referentes ao aquário, como temperatura, pH e luminosidade. Também é possível configurar alertas por e-mail e no próprio dispositivo para os sensores, como alta temperatura, por exemplo. O sistema também armazena os logs para visualizar as informações antigas referentes ao aquário, com limite de 30 dias.

Figura 4 - Exemplo de acesso a *dashboard* de monitoramento



Fonte: Seneye (2016).

Embora o dispositivo seja bem completo, ele peca por funcionar apenas conectado a um computador e tem um cabo de apenas 2.5 metros, o que obriga o usuário a colocar um computador perto do aquário, e não é compatível com Mac ou Linux. Também é necessário deixar o computador ligado em todo o tempo, o que pode ocasionar um consumo excessivo de energia. Há um aplicativo para smartphone, porém ainda é necessário a conexão com o computador. O preço é de 59 euros pela versão Home, 89 euros pela versão Pound, e 99 euros pela versão Reef (HILL, 2011).

2.6.2 Insight 24/7 Monitor/Controller

Desenvolvido pela Puratek, sua função é oferecer ao usuário um sistema para monitoramento e controle de aquários. Ele permite que o usuário automatize serviços simples como controle de luz e alimentação automática, até tarefas mais complexas, como mudanças químicas para alterar a qualidade da água (PURATEK, 2016). O dispositivo faz a leitura da temperatura da água, do pH da água, da luminosidade e da temperatura externa. Possui quatro versões, Insight Essential, Insight Choice, Insight Supreme e Insight Ultra.

Para utilizá-lo, é necessário conectar o dispositivo a uma tomada e conectar todos os sensores a uma central, nela é possível configurar todas as funções disponíveis. Para utilizar o computador para visualizar as informações, é possível realizar a conexão com o computador por meio de uma porta USB. A Figura 5 exibe os itens presentes na versão Essential.

Figura 5 - Itens presentes na versão Insight Essential

Fonte: Puratek (2016).

O Insight 24/7 Monitor/Controller é um dispositivo muito completo e profissional, porém com um alto custo. Sua versão mais básica que possui apenas monitoramento, chamada de Essential, custa 539 dólares. As outras versões, já com a possibilidade de controle, custam 849 dólares pela versão Choice, 899 dólares pela versão Supreme, e $999 pela versão Ultra (PURATEK, 2016). Para visualizar os dados no computador é necessário realizar a conexão por USB, o que pode gerar um incomodo.

2.6.3 Apex AquaController

O Apex AquaController é desenvolvido pela Neptune, é um dispositivo para monitoramento e controle do aquário. Com ele, é possível monitorar informações como temperatura, pH. Ele possui conexão Ethernet para conectá-lo a internet e é possível configurar alertas de notificação. Possui duas versões, Apex e Apex Gold (NEPTUNE SYSTEMS, 2016).

Para utilizá-lo, é necessário conectar o dispositivo a uma tomada e conectar todos os sensores a central, ilustrada na Figura 6. Depois é necessário conectar o cabo ethernet para realizar a conexão com a Internet, e assim acessar o Apex Fusion, conforme pode ser visto na Figura 7, para visualizar e configurar as informações referentes ao aquário.

Figura 6 - Central do Dispositivo Apex

Fonte: Neptune Systems (2016).

Assim como o Insight 24/7 Monitor/Controller, o Apex também é muito completo e profissional, e também muito caro. Sua versão mais básica sai por 549 dólares, e a versão Gold por 799 dólares (BULK REEF SUPPLY, 2016). Para visualizar os dados no computador é necessário realizar a conexão por cabo Ethernet, o que pode gerar um incômodo.

Figura 7 - Exemplo de interface do Apex Fusion

Fonte: Neptune Systems (2016).

3 DESENVOLVIMENTO

Este capítulo aborda os aspectos referentes à construção do sistema. Na seção 3.1 encontram-se os requisitos funcionais e não funcionais. Na seção 3.2 são apresentados o diagrama de arquitetura, diagrama de casos de uso e diagramas de atividades, especificando e detalhando o funcionamento do aplicativo. A seção 3.3 aborda os detalhes da implementação, como a construção do hardware e os principais trechos de código-fonte. Por fim, a seção 3.4 faz uma análise dos resultados.

3.1 REQUISITOS

Os requisitos funcionais e não funcionais da plataforma estão descritos conforme a lista abaixo:

a) o sistema deve permitir ao usuário o cadastro dos peixes (Requisito Funcional (RF));
b) o sistema deve permitir ao usuário a manutenção do cadastro dos peixes (RF);
c) o sistema deve permitir ao usuário o cadastro do hardware no aplicativo por meio de rede Wi-Fi (RF);
d) o sistema deve permitir ao usuário a consulta de temperatura atual da água (RF);
e) o sistema deve permitir ao usuário a consulta do histórico de temperatura da água através de gráficos (RF);
f) o sistema deve permitir ao usuário a consulta de pH da água (RF);
g) o sistema deve permitir ao usuário a consulta do histórico de pH da água através de gráficos (RF);
h) o sistema deve permitir ao usuário a consulta de temperatura externa (RF);
i) o sistema deve permitir ao usuário a consulta do histórico de temperatura externa através de gráficos (RF);
j) o sistema deve permitir ao usuário a consulta de umidade externa (RF);
k) o sistema deve permitir ao usuário a consulta do histórico de umidade externa através de gráficos (RF);
l) o sistema deve permitir ao usuário a consulta do funcionamento da lâmpada ultravioleta (RF);
m) o sistema deve permitir ao usuário a consulta do fluxo de água da bomba (RF);
n) o sistema deve permitir ao usuário a consulta do histórico do fluxo de água através de gráficos (RF);
o) o sistema deve permitir ao usuário configurar alertas para temperatura, pH e lâmpada ultravioleta desligada (RF);

p) o hardware será desenvolvido com o módulo ESP8266 (Requisito Não Funcional (RNF));
q) o hardware deverá ser alimentado com uma bateria carregada por energia solar (RNF);
r) o hardware utilizará a tecnologia Wi-Fi para enviar os dados para o servidor (RNF);
s) o hardware deverá utilizar a linguagem de programação C++ utilizando o *framework* Sming (RNF);
t) o aplicativo *mobile* deverá utilizar a linguagem de programação Typescript utilizando o *framework* Ionic (RNF);
u) o aplicativo *mobile* será acessível a smartphones com sistema operacional Android 4.1 ou superior (RNF).

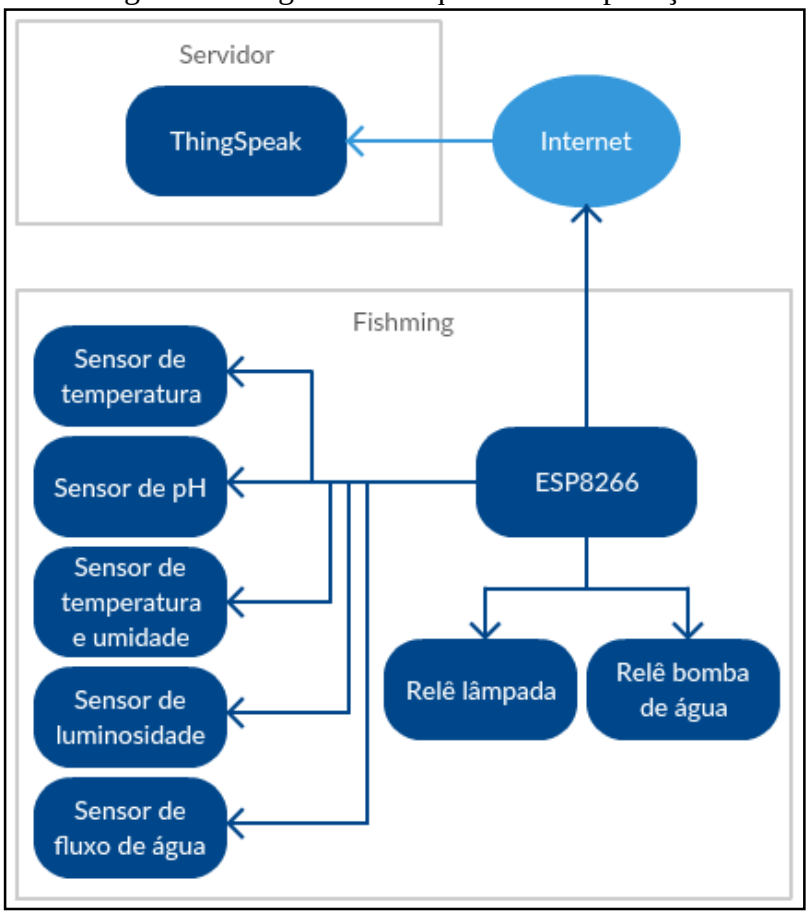
3.2 ESPECIFICAÇÃO

A especificação foi realizada utilizando a ferramenta online Creately seguindo a metodologia Unfield Modeling Language (UML). Para a especificação foram elaborados o diagrama de arquitetura da aplicação, os diagramas de casos de uso e o diagrama de atividades.

3.2.1 Diagrama de arquitetura da aplicação

A Figura 8 exibe as principais características da arquitetura da aplicação.

Figura 8 - Diagrama de arquitetura da aplicação



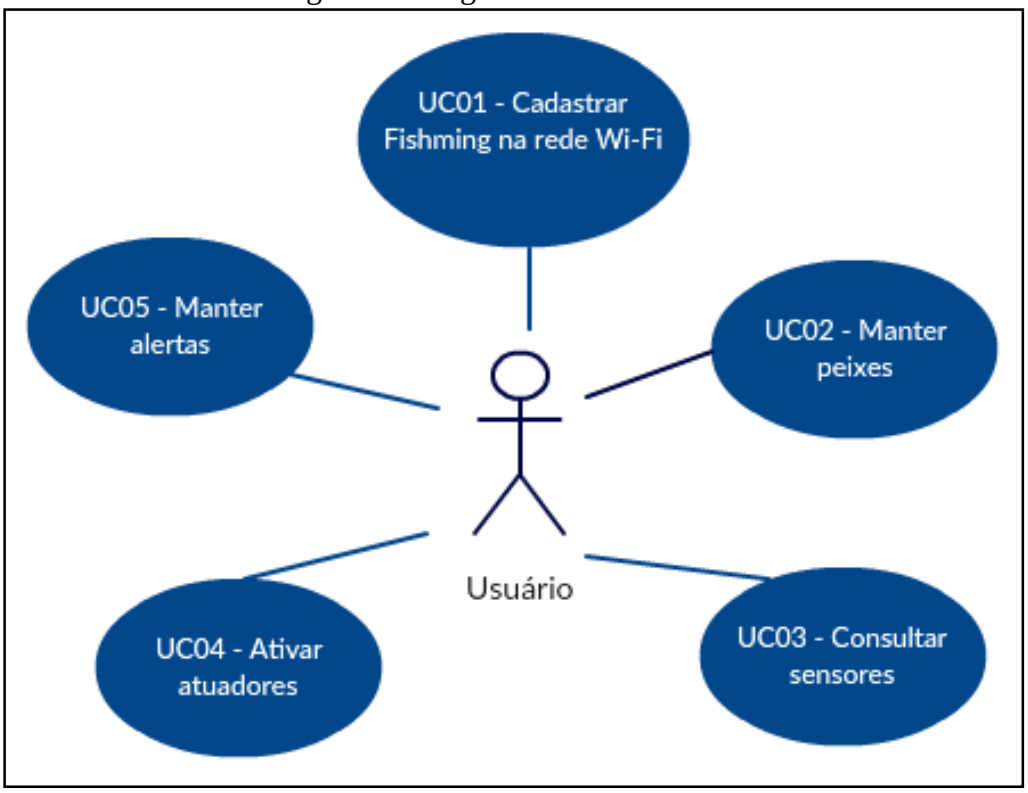
Fonte: elaborado pelo autor.

Como pode-se observar o Fishming é composto por um ESP8266 conectado a alguns periféricos, entre eles cinco sensores: sensor de temperatura, sensor de pH, sensor de temperatura e umidade, sensor de luminosidade e sensor de fluxo de água. Também está conectado a dois atuadores: relê da lâmpada e relê da bomba de água. O ESP8266 é o encarregado de enviar os dados ao servidor ThingSpeak através de conexão Wi-Fi por meio de protocolo Hypertext Transfer Protocol (HTTP).

3.2.2 Diagrama de casos de uso

A Figura 9 exibe os casos de uso do aplicativo implementado neste trabalho.

Figura 9 - Diagrama de casos de uso

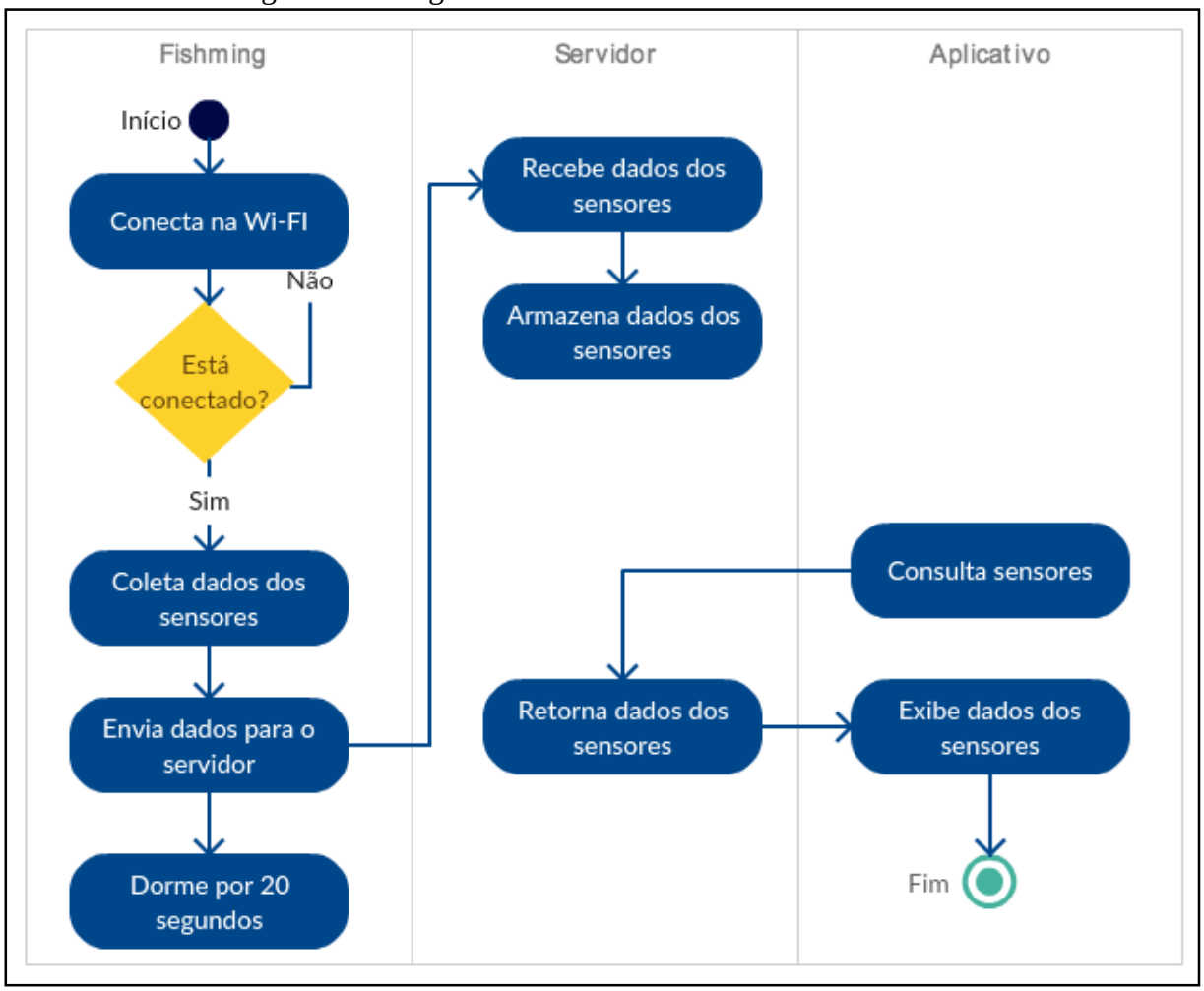


Fonte: elaborado pelo autor.

3.2.3 Diagrama de atividades

Para facilitar a compreensão das atividades realizadas pela aplicação, foram elaborados dois diagramas de atividade. A seguir, na Figura 10, é exibido o diagrama de atividades da parte de leitura dos sensores.

Figura 10 - Diagrama de atividade da leitura dos sensores



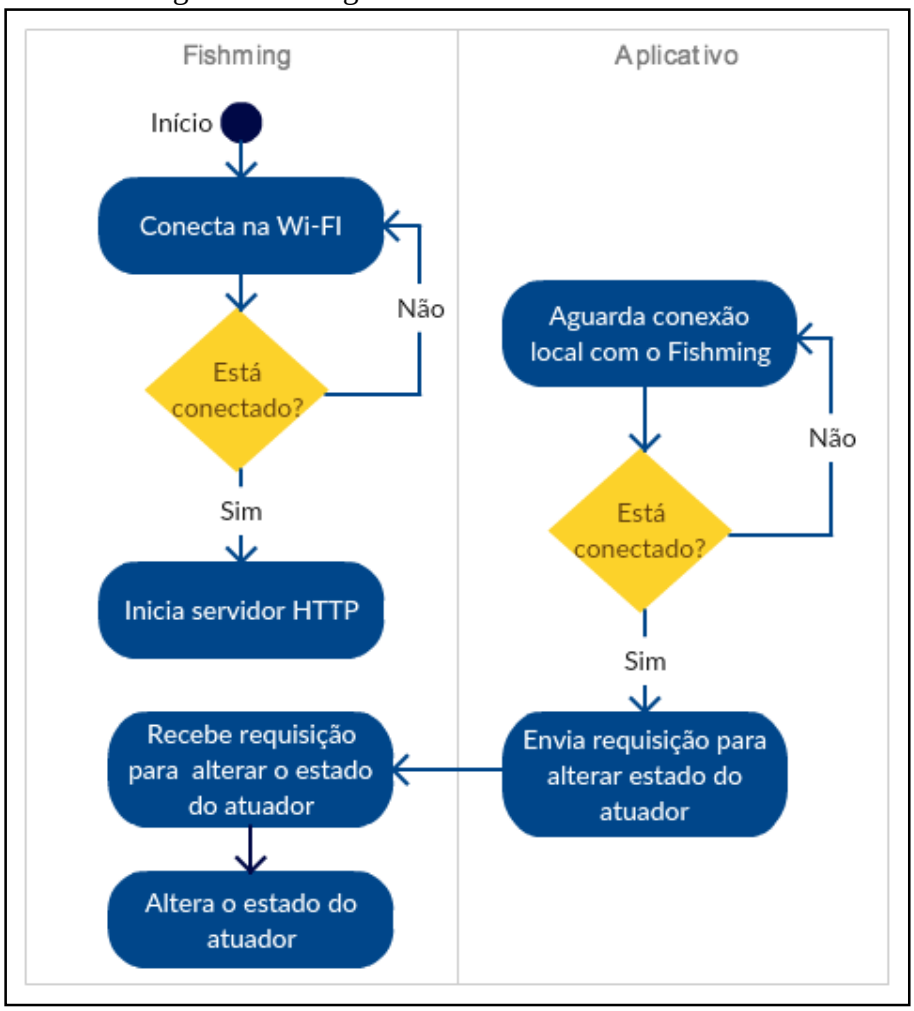
Fonte: elaborado pelo autor.

Como pode-se observar, o processo de leitura dos sensores é dividido em 3 partes: Fishming (hardware), servidor (ThingSpeak) e aplicativo. As partes se dividem da seguinte forma:

a) a primeira etapa corresponde ao hardware instalado, com acesso a internet por meio de Wi-Fi;
b) a segunda etapa corresponde ao servidor ThingSpeak, o qual recebe e armazena os dados obtidos pelos sensores;
c) a terceira corresponde ao aplicativo *mobile*, onde é possível consultar estes dados armazenados.

Na Figura 11 está demonstrado o funcionamento da parte dos atuadores, onde o usuário pode ligar ou desligar o atuador por meio do aplicativo *mobile*.

Figura 11 - Diagrama de atividade dos atuadores



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

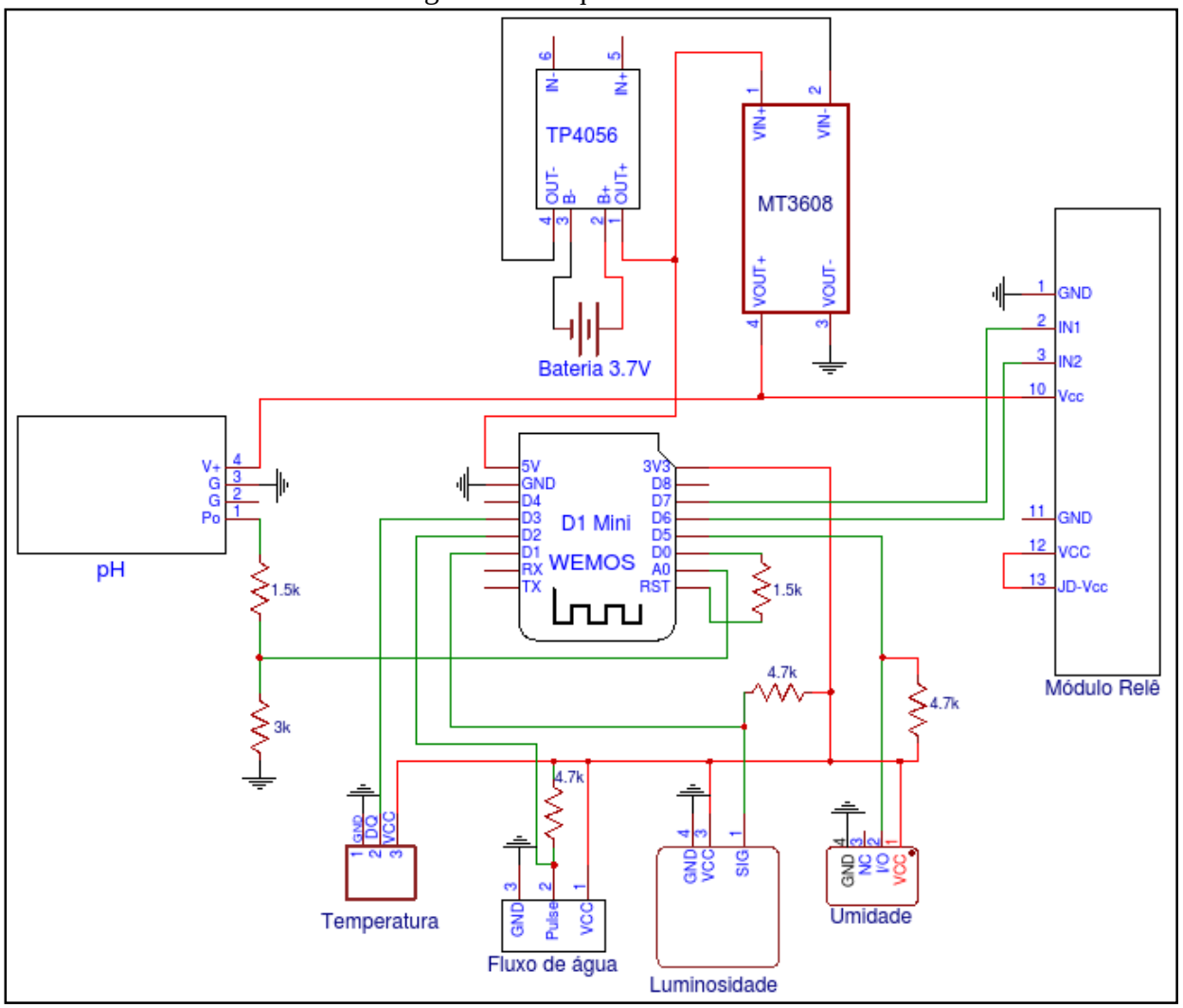
Nesta seção são mostrados os detalhes da construção do hardware, as técnicas e ferramentas utilizadas e a operacionalidade da implementação. A seção 3.3.1 apresenta detalhes sobre a construção do hardware. A seção 3.3.2 apresenta as técnicas e ferramentas utilizadas. A seção 3.3.3 aborda os principais trechos de código-fonte. Por fim, a seção 3.3.4 descreve a operacionalidade da implementação.

3.3.1 Construção do hardware

Para a construção do hardware, foi utilizado um Wemos D1 mini, que é baseado no módulo ESP-8266, um sensor de temperatura (DS18B20), um sensor de pH, um sensor de fluxo de água (FS400A), um sensor de luminosidade, um sensor de temperatura e umidade (DHT11), e um módulo com dois relês. Também foi necessário utilizar uma placa ilhada, resistores, fios, conectores, um módulo para carregamento da bateria (TP4056), e um módulo conversor DC-DC *step up* (MT3608). O custo das peças e periféricos foi disponibilizado no Apêndice A. Na

Figura 12 é exibido o esquema elétrico completo do hardware, que foi montado utilizando a ferramenta EasyEDA.

Figura 12 - Esquema elétrico



Fonte: elaborado pelo autor.

Conforme pode-se observar no esquema elétrico, o D1 mini é alimentado diretamente pelas baterias em sua porta 5V, que permite uma tensão entre 3.5V até 5V. São utilizadas três baterias de 3.7 volts com 2800mAh cada, e para o seu carregamento, é utilizado o módulo TP4056, que é alimentado com 5V pela placa de energia solar. O sensor de pH e o módulo relê precisam de 5 volts, e para gerar essa tensão é utilizado o módulo MT3608, que é um conversor DC-DC *step-up*[1] Entre as portas D0 e RST do módulo, existe um resistor de 1.5K para o funcionamento da função *deep sleep*[2].

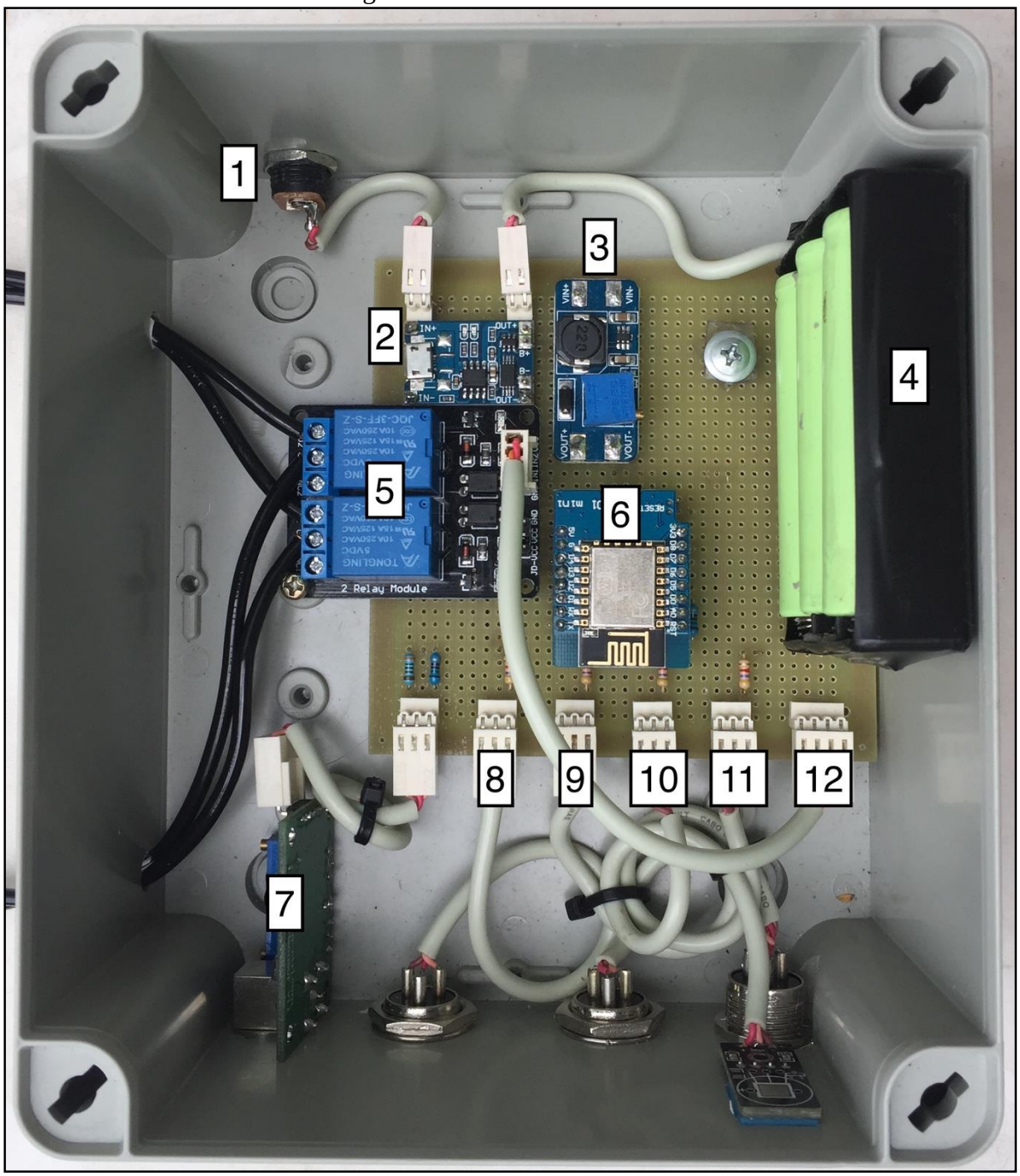
[1] Um conversor DC-DC *step-up* é utilizado para aumentar a tensão de sua entrada para sua saída.
[2] *Deep sleep* é um modo de operação de baixo custo energético.

A conexão dos sensores é feita com o módulo D1 mini. O sensor de pH é alimentado com 5V e para realizar a sua leitura utiliza-se a porta analógica (A0) do módulo, porém utilizando 2 resistores para montar um circuito *voltage divider*[3], que diminuiu a tensão de 5V para 3.3V (tensão máxima da porta analógica). Todos os outros sensores são alimentados com a saída 3.3V do módulo. O sensor de temperatura da água utiliza a porta D3. O sensor de fluxo de água está ligado na porta D2 do módulo. O sensor de luminosidade está ligado na porta D1 e o sensor de umidade está ligado na porta D5. Os sensores de fluxo de água, luminosidade e umidade são ligados em um resistor *pull-up*[4] de 4.7k cada.

A conexão dos atuadores também é feita com o módulo D1 mini. Um módulo relê com 2 canais é responsável pela ativação dos atuadores, e este é alimento com 5V. O módulo possui uma entrada digital para cada relê que estão conectadas as saídas D7 e D6 do ESP8266 e são utilizados para a bomba de água e a lâmpada ultravioleta (UV), respectivamente. Como a bomba de água e a lâmpada UV ficam constantemente ligadas, elas são ligadas a saída *Normally Open* (NO) de cada relê. Na Figura 13 pode-se observar o hardware depois de montado.

[3] *Voltage divider* é um circuito simples que transforma uma certa tensão em uma menor. Usando apenas duas resistores em série e uma tensão de entrada, é possível criar uma tensão de saída que é uma fração da entrada (SPARKFUN, 2017a).
[4] O resistor *pull-up* é utilizado para manter a tensão de um circuito em nível lógico alto. Ou seja, o respectivo circuito é ligado a tensão positiva através de um resistor para que represente nível lógico alto enquanto não receber tensão de outro lugar.

Figura 13 - Hardware montado



Fonte: elaborado pelo autor.

O hardware é melhor detalhado a seguir:

a) o item 1 da Figura 13 é um conector P4 para a entrada de alimentação;

b) o item 2 é o módulo TP4056, responsável pelo carregamento das baterias de lítio. Sua tensão de entrada é de 5V, sua a tensão saída é de 4.2V, e a capacidade máxima de carga é de 1A;

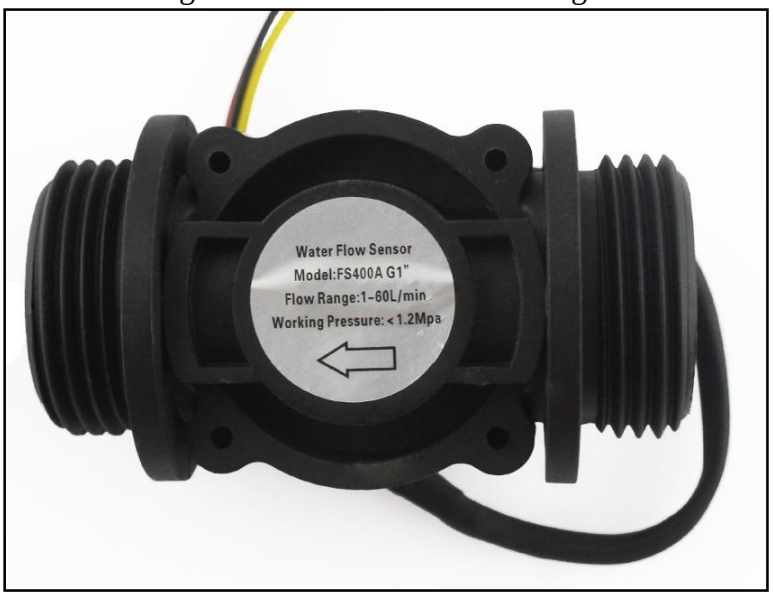
c) o item 3 é o módulo MT3608. Ele é responsável por aumentar a tensão da entrada de 3.7V para 5V, para realizar a alimentação do sensor de pH e do módulo relê;

d) o item 4 é o suporte para as 3 baterias de 3.7 volts de tamanho 18650 com 2800mAh cada que ficam em paralelo, totalizando 8400mAh;
e) o item 5 é o módulo relê de 2 canais, responsável pela ativação dos atuadores. É alimentado com 5V diretamente da saída do item 3;
f) o item 6 o WEMOS D1 mini, que é responsável por controlar os sensores e atuadores;
g) o item 7 é um módulo sensor de pH. Ele é responsável por realizar a leitura do nível de pH e é conectado na saída analógica do D1 mini;
h) o item 8 é o conector para o sensor de temperatura DS18B20. É responsável por coletar informações sobre a temperatura da água, e é conectado ao D1 mini através de conexão *one-wire*[5];
i) o item 9 é o conector para o sensor de fluxo de água FS400A. É responsável por realizar a leitura do fluxo de água através de pulsos;
j) o item 10 é o conector para o sensor de temperatura e umidade DHT11. É responsável por coletar informações sobre a temperatura e a umidade externa e é conectado ao D1 mini através de conexão *one-wire*;
k) o item 11 é o conector para o sensor de luminosidade. É responsável por verificar o funcionamento da lâmpada UV;
l) o item 12 é o conector para o módulo relê (item 5).

Foram utilizados alguns sensores externos para captura de informações da água e do ambiente. Na Figura 14 é possível observar o sensor FS400A, responsável pela leitura do fluxo de água.

[5] One-wire é um protocolo serial que utiliza apenas um fio de dados para comunicação.

Figura 14 - Sensor de Fluxo de água



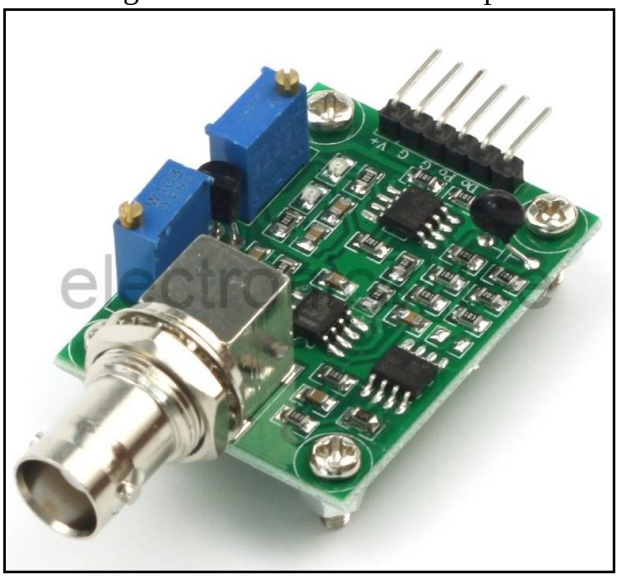
Fonte: Haoyu (2017).

O sensor de fluxo de água consiste em uma válvula de plástico, um rotor de água e um sensor de efeito hall. Quando a água passa pelo rotor, ele gira. Sua velocidade muda com diferentes taxas de fluxo. O sensor de efeito hall é responsável pela geração de pulsos correspondentes, que são enviados para o D1 mini. Com base nesses pulsos é realizado um cálculo para obter o valor em litros por hora. Na Figura 15 pode-se observar a sonda de pH e na Figura 16 o módulo sensor de pH.

Figura 15 - Sonda de pH

Fonte: elaborado pelo autor.

Figura 16 - Módulo sensor de pH



Fonte: elaborado pelo autor.

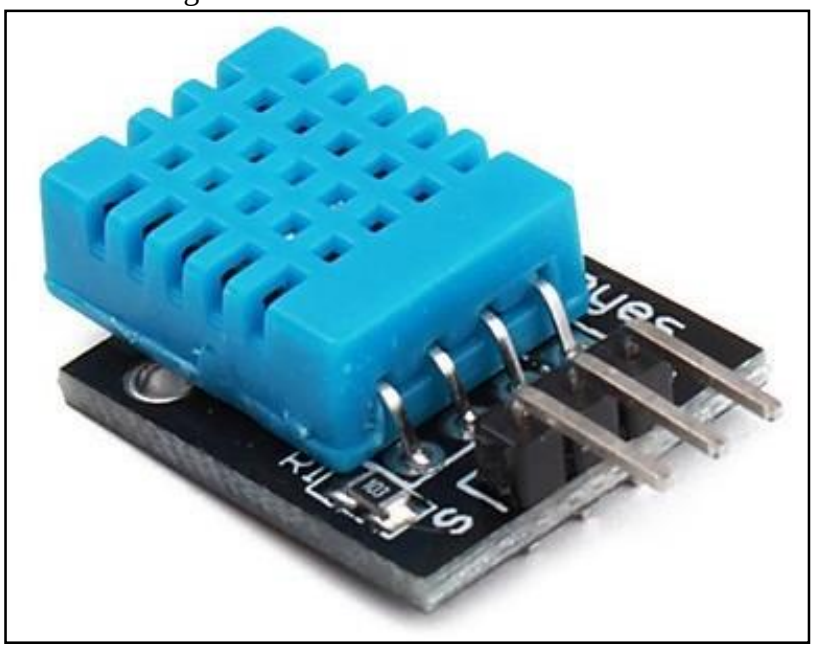
A sonda de pH fica conectado ao módulo sensor de pH por meio de um conector BNC. O módulo é alimentado com 5V e possui uma saída analógica que informa o valor de pH. Para ser conectado ao D1 mini, foi necessário utilizar um circuito *voltage-divider*, que diminuiu os 5V gerados para 3.3V com a utilização de resistores de 1.5kOhm e 3kOhms. Na Figura 17 é possível observar o módulo DS18B20, responsável pela leitura da temperatura da água.

Figura 17 - Sensor de temperatura DS18B20

Fonte: elaborado pelo autor.

O DS18B20 é um sensor de temperatura alimentado com 3.3V que envia os dados para o D1 mini através de uma conexão *one-wire*. Ele suporta temperaturas entre -55ºC e 125ºC com 0,5ºC de margem de erro entre -10ºC e 85ºC e pode ser utilizado em baixo da água (SPARKFUN, 2017b). Foi adicionado um conector mike de 3 vias para facilitar a montagem e desmontagem do protótipo. A Figura 18 exibe os detalhes do módulo DHT11, responsável pela leitura da umidade do ambiente.

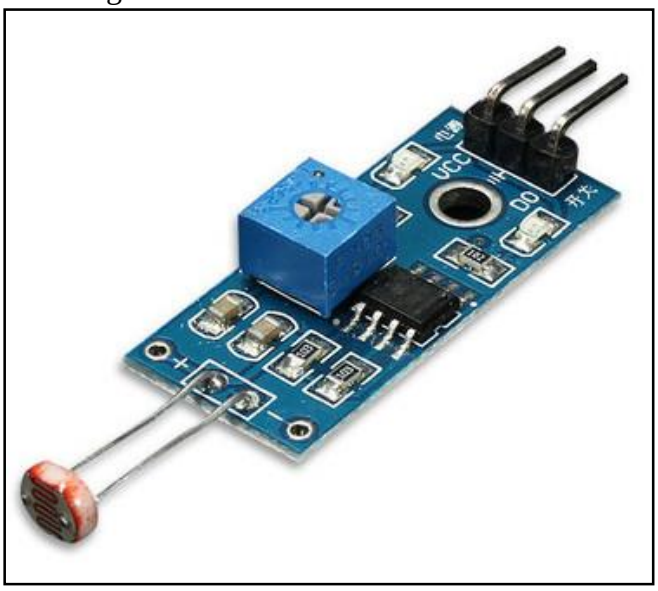
Figura 18 - Sensor de umidade DHT11



Fonte: Robotistan (2017).

O DHT11 é um sensor de temperatura e umidade que envia os dados para o D1 mini através de uma conexão *one-wire*, e é alimentado com 3.3V. Ele suporta realizar leituras de umidade entre 20% e 90%, com 5% de margem de erro. O sensor de temperatura realiza leituras entre 0ºC e 50ºC, com margem de erro de 2ºC (ROBOTISTAN, 2017). A Figura 19 exibe o sensor de luminosidade, responsável pela verificação de funcionamento da lâmpada UV.

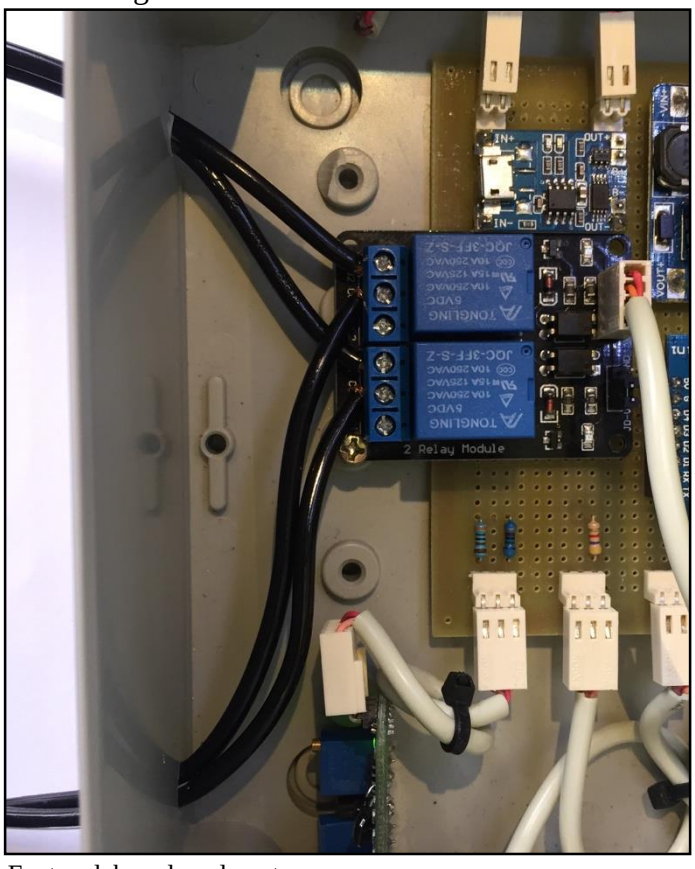
Figura 19 - Módulo sensor de luz LDR



Fonte: Filipeflop (2017).

O módulo sensor de luz LDR é um sensor que converte o dado analógico de um sensor LDR para um sinal digital que indica se possui ou não luz. Possui um potenciômetro para ajustar a luminosidade necessária para ligar o sinal. É alimentado com 3.3V e sua leitura é feita através da leitura digital da porta pelo D1 mini, quando em nível baixo indica que a luz está ligada. Na Figura 20 pode-se ver o módulo relê responsável pela ação dos atuadores.

Figura 20 - Módulo relê com 2 canais



Fonte: elaborado pelo autor.

O módulo relê com 2 canais é alimentado com 5V e possui 2 relês de 10 amperes cada. Possui duas entradas digitais para ativar ou desativar o relê, responsáveis por controlar a bomba de água e a lâmpada UV. Essas duas entradas são ativadas com uma tensão entre 2V e 5, por isso são ativadas diretamente pelo D1 mini.

3.3.2 Técnicas e ferramentas utilizadas

No desenvolvimento, o sistema é subdivido em 2 partes: aplicativo móvel e hardware. Para o desenvolvimento do aplicativo foi utilizado o editor de código fonte Visual Studio Code e a linguagem de programação utilizada foi o TypeScript. Para o desenvolvimento do hardware utilizou-se a Integrated Development Environment (IDE) Eclipse e a linguagem utilizada foi C++. O código fonte do aplicativo e do hardware foram gerenciados pelo sistema de controle de versão git, por meio do site Bitbucket.

Para o desenvolvimento da camada *front-end*, ou seja, o aplicativo onde o usuário realiza toda a interação com o hardware, foi utilizado o framework NPM para o gerenciamento de dependências. O Ionic foi utilizado para construir a aplicação para dispositivos com sistema operacional Android, empacotando os elementos do *front-end* e formando um arquivo instalável. O *framework* AngularJS foi utilizado para aumentar a abstração do código Typescript, utilizando diretivas para facilitar a integração entre o modelo e as páginas HTML. O AngularJS foi utilizado também para realizar chamadas HTTP para o servidor ThingSpeak, utilizando o padrão Notação de Objetos JavaScript (JSON). Foram adicionados alguns plug-ins no projeto: Chart.js – para geração de gráficos, cordovaEsptouch – para utilização da tecnologia Smart Config, e WifiWizard – utilizado para coletar dados sobre a rede Wi-Fi atual.

Para o hardware, foi utilizado o framework Sming para realizar o desenvolvimento nativo para ESP8266 com a linguagem C++. Para leitura dos sensores, foram utilizadas as seguintes bibliotecas, já incluídas no Sming: DS18S20 e DHT. Para utilização de objetos JSON foi utilizado a biblioteca ArduinoJSON. Para utilização da memória EEPROM, foi utilizada a biblioteca EEPPROM do Arduino. A compilação de código para execução no ESP8266 e a transferência do código para o hardware foi realizada sempre a partir um MacOS rodando a versão 10.12.4, utilizando a ferramenta esptool. A conexão entre o computador e o D1 mini foi feita através de cabo USB.

3.3.3 Código fonte da aplicação

Nesta seção serão apresentados os principais trechos de código da aplicação, divididos entre hardware e aplicativo móvel.

3.3.3.1 Hardware

O hardware inicia o seu funcionamento realizando a leitura das configurações salvas na EEPROM e após isto é realizado a tentativa de conexão com a rede Wi-Fi. Para isso, é chamado o método `start` da classe `WifiControllerClass`, conforme pode-se observar no Quadro 1. O primeiro item que se verifica é se já possui as credenciais da rede Wi-Fi salvas (linha 14) e caso possua, realiza a conexão por meio do método `connect` (linha 19). Caso não possua as credenciais da rede salvas, inicia a configuração SmartConfig, e fica aguardando a conexão do aplicativo (linha 26).

Quadro 1 – Configuração da conexão Wi-Fi

```
14 void WifiControllerClass::start() {
15     if (Settings.networkInfo.ssid && Settings.networkInfo.password) {
16
17         WifiStation.enable(true);
18         WifiStation.setIP(Settings.networkInfo.ip);
19         connect(Settings.networkInfo.ssid, Settings.networkInfo.password);
20
21     } else {
22         LoggerManager.log("WifiController", "Configuration file is empty, starting smart config" );
23
24         WifiAccessPoint.enable(false);
25         WifiStation.enable(true);
26         WifiStation.smartConfigStart(SCT_EspTouch, SmartConfigDelegate(&WifiControllerClass::smartConfigCallback, this));
27     }
28 }
```

Fonte: elaborado pelo autor.

Após realizar conexão com sucesso, conforme pode-se ver no Quadro 2, cria-se o servidor HTTP, responsável pelo recebimento de algumas chamadas locais feitas direto pelo aplicativo, que são: ligar ou desligar os atuadores e definir a lista de alertas do sistema. A chamada para o serviço `keepAlive` é feita a cada 30 segundos quando o usuário está com o aplicativo aberto, e é utilizada para evitar que o hardware entre no modo `deepSleep`.

Quadro 2 - Criação do servidor HTTP

```
13 void WebServerClass::start() {
14     LoggerManager.log("WebServer", "Starting HTTP Server on IP: " + WifiStation.getIP().toString());
15
16     server.listen(80);
17     server.addPath("/keepAlive", HttpPathDelegate(&WebServerClass::onKeepAliveRequest, this));
18     server.addPath("/actuator", HttpPathDelegate(&WebServerClass::onActuatorRequest, this));
19     server.addPath("/alerts", HttpPathDelegate(&WebServerClass::onAlertsRequest, this));
20 }
```

Fonte: elaborado pelo autor.

Também após realizar a conexão com a rede Wi-Fi, inicia-se a captura dos sensores. Todos as classes que capturam informações dos sensores estendem a implementação da classe `Sensor`, que pode ser vista no Quadro 3. A classe possui como propriedades `id` – identificador do sensor e `pin` – pino utilizado no D1 mini para a leitura. Todos os sensores precisam implementar o método `getValue`, que retorna o valor do sensor no exato momento.

Quadro 3 - Classe `Sensor`

```
7  class Sensor {
8
9  public:
10
11     virtual ~Sensor();
12     virtual float getValue() = 0;
13
14     // getters
15     int getPin() { return _pin; };
16     int getId() { return _id; };
17
18 protected:
19
20     Sensor(int id, int pin);
21
22     int _id;
23     int _pin;
24
25 };
```

Fonte: elaborado pelo autor.

No Quadro 4 é possível observar o funcionamento da classe `WaterFlowSensor`. No construtor da classe, é adicionado um gatilho para ser disparado quando o pino estiver com valor alto (linha 9). A cada vez que esse gatilho é disparado, incrementa-se a variável `flowFrequency` (linha 25). Na linha 12 é adicionado uma *callback* para ser chamada a cada 1 segundo, que faz o cálculo do fluxo de água em litros por hora. O método `getValue` (linha 19), apenas retorna o valor que é calculado pelo método `read`.

Quadro 4 - Implementação do sensor de fluxo de água

```
4  WaterFlowSensor::WaterFlowSensor(int id, int pin) : Sensor(id, pin) {
5      flowFrequency = 0;
6      value = 0;
7
8      LoggerManager.log("WaterFlowSensor", "starting on pin: " + String(_pin));
9      attachInterrupt(_pin, InterruptDelegate(&WaterFlowSensor::flow, this), RISING);
10
11     Timer* timer = new Timer();
12     timer->initializeMs(1000, TimerDelegate(&WaterFlowSensor::read, this)).start();
13 }
14
19 float WaterFlowSensor::getValue() {
20     LoggerManager.log("WaterFlowSensor", "is " + String(value) + " L/h");
21     return value;
22 }
23
24 void WaterFlowSensor::flow() {
25     flowFrequency++;
26 }
27
28 void WaterFlowSensor::read() {
29     // (Pulse frequency x 60 min) / 4.8Q = flowrate in L/hour
30     value = (flowFrequency * 60 / 4.8);
31     flowFrequency = 0;
32 }
```

Fonte: elaborado pelo autor.

No trecho de código do Quadro 5 pode-se observar a implementação da classe `PHSensor`. No construtor é adicionado uma *callback* para realizar a captura do pH a cada 200ms (linha 12). Toda vez que esse método é chamado, ele calcula o valor atual da porta analógica e adiciona na lista de amostras (entre a linha 19 e linha 26). A implementação do método `getValue` converte o valor analógico em um valor da escala pH (linha 37).

Quadro 5 - Implementação do sensor de pH

```
3  #define inputMin 784
4  #define inputMax 552
5  #define phMin 4
6  #define phMax 10
7
8  PHSensor::PHSensor(int id, int pin) : Sensor(id, pin) {
9      LoggerManager.log("PHSensor", "starting on pin " + String(_pin));
10
11     Timer* readTimer = new Timer();
12     readTimer->initializeMs(200, TimerDelegate(&PHSensor::read, this)).start();
13 }
14
19 void PHSensor::read() {
20     int total = 0;
21     int readSamples = 1000;
22     for (int i = 0; i < readSamples; i++) {
23         total += analogRead(_pin);
24     }
25
26     samples.add(total / readSamples);
27 }
29
30 float PHSensor::getValue() {
31     int total = 0;
32     for (int i = 0; i < samples.size(); i++) {
33         total += samples.get(i);
34     }
35
36     int analogInput = total / samples.size();
37     float ph = ((float) (analogInput - inputMin) * (phMax - phMin) / (float) (inputMax - inputMin) + phMin);
38
39     LoggerManager.log("PHSensor", "is " + String(ph));
40
41     samples.clear();
42     return ph;
43 }
```

Fonte: elaborado pelo autor.

O Quadro 6 mostra a implementação principal da classe `WaterFlowSensor`, responsável pela captura da temperatura da água por meio do sensor DS18B20. O método `getValue` retorna o valor em graus celsius que é capturado através da biblioteca `DS18S20` (linha 23).

Quadro 6 – Implementação do sensor de temperatura da água

```
16 float WaterTemperatureSensor::getValue() {
17     if (ds18s20.MeasureStatus()) {
18         LoggerManager.error("WaterTemperatureSensor", "measure is in progress");
19         return 0;
20     }
21
22     if (ds18s20.GetSensorsCount() == 1) {
23         float temperature = ds18s20.GetCelsius(0);
24         LoggerManager.log("WaterTemperatureSensor", "is " + String(temperature) + "ºC");
25
26         ds18s20.StartMeasure();
27         return temperature;
28     } else {
29         LoggerManager.error("WaterTemperatureSensor", "sensors count is 0");
30     }
31     return 0;
32 }
```

Fonte: elaborado pelo autor.

O Quadro 7 exibe a implementação da classe `HumiditySensor`, responsável pela captura da umidade do ambiente através do sensor DHT11. A captura é feita pela biblioteca `DHT`, e o método `getValue` apenas retorna o valor capturado.

Quadro 7 – Implementação do sensor de umidade

```
14 float HumiditySensor::getValue() {
15     float humidity = dht.readHumidity();
16     if (isnan(humidity)) {
17         LoggerManager.error("HumiditySensor", "Failed to read from DHT (humidity)");
18         return 0;
19     }
20
21     LoggerManager.log("HumiditySensor", "is " + String(humidity) + "%");
22     return humidity;
23 }
```

Fonte: elaborado pelo autor.

No Quadro 8 pode-se ver a implementação da classe `ExternalTempertureSensor`, que faz a captura da temperatura atual do ambiente com o sensor DHT11. A captura funciona do mesmo modo que o sensor de umidade, utilizando a biblioteca `DHT`.

Quadro 8 – Implementação do sensor de temperatura externa

```
14 float ExternalTemperatureSensor::getValue() {
15     float temperature = dht2.readTemperature();
16     if (isnan(temperature)) {
17         LoggerManager.error("ExternalTemperatureSensor", "Failed to read from DHT (temperature)");
18         return 0;
19     }
20
21     LoggerManager.log("ExternalTemperatureSensor", "is " + String(temperature) + "ºC");
22     return temperature;
23 }
```

Fonte: elaborado pelo autor.

No trecho de código do Quadro 9, pode-se observar a implementação da classe `LightSensor`, responsável pela verificação do funcionamento da lâmpada UV. Na linha 13 é feito a captura do valor da porta digital, quando esse valor está em nível lógico baixo, indica que a lâmpada está ligada. Esse valor é retornado pelo método `getValue`.

Quadro 9 - Implementação do sensor da lâmpada UV

```
12 float LightSensor::getValue() {
13     int disabled = digitalRead(_pin);
14     float value = !disabled;
15
16     LoggerManager.log("LightSensor", "is " + String(value));
17     return value;
18 }
```

Fonte: elaborado pelo autor.

O Quadro 10 exibe o corpo do método `readSensors`, que fica na classe principal `FishmingClass`. Após verificar se está conectado na rede Wi-Fi, o método coleta o valor atual de cada sensor através do método `getValue` de cada sensor (linha 73) e adiciona em um `HashMap`. Os valores são enviados para o método `check` da classe `AlertController` (linha 76), e para o método `send` da classe `ThingSpeak` (linha 77). Os métodos fazem a verificação dos

alertas e envio dos valores para o servidor, respectivamente. Caso não esteja conectado na rede Wi-Fi, realiza a chamada do mesmo método após 5 segundos.

Quadro 10 - Leitura dos sensores

```
67 void FishmingClass::readSensors() {
68     if (WifiStation.isConnected()) {
69         HashMap<int, float> *map = new HashMap<int, float>();
70 
71         for (int i = 0; i < sensors.size(); i++) {
72             Sensor *sensor = sensors[i];
73             (*map)[sensor->getId()] = sensor->getValue();
74         }
75 
76         AlertController.check(map);
77         ThingSpeak.send(map);
78     } else {
79         timer.initializeMs(5000, TimerDelegate(&FishmingClass::readSensors, this)).startOnce();
80     }
81 }
```

Fonte: elaborado pelo autor.

O Quadro 11 exibe a implementação do método `check` da classe `AlertController`. Na linha 24 percorre-se cada um dos alertas cadastrados e depois verifica se o alerta é disparado com o valor atual do sensor (linha 27). Se o alerta foi disparado, é chamado a *callback* `onAlertFiredCallback`, que faz o envio da notificação push para o aplicativo mobile por meio do serviço Google Firebase Notifications.

Quadro 11 – Método de verificação dos alertas

```
22 void AlertControllerClass::check(HashMap<int, float>* values) {
23     alertFired = false;
24     for (int i = 0; i < Settings.alerts.size(); i++) {
25         Alert *alert = Settings.alerts[i];
26         float value = values->valueAt(values->indexOf(alert->field));
27         int fired = alert->isFired(value);
28         if (fired > 0) {
29             if (onAlertFiredCallback) {
30                 onAlertFiredCallback(alert->field, value, fired);
31                 alertFired = true;
32             }
33         }
34     }
35 }
```

Fonte: elaborado pelo autor.

Outro método chamado após a leitura dos sensores, é o método `send` da classe `ThingSpeak`, que é exibido no Quadro 12. Na linha 19 é criado o objeto JSON e nesse objeto é adicionado a chave de escrita fornecida pelo ThingSpeak (linha 20) e o código do fuso horário local (linha 21). Na linha 23 percorre-se o mapa de sensores para preencher o JSON com o valor atual de cada sensor. O objeto JSON é adicionado na requisição HTTP e a mesma é enviado para o servidor.

Quadro 12 – Método de envios dos sensores para o servidor ThingSpeak

```
void ThingSpeakClass::send(HashMap<int, float>* values) {
    DynamicJsonBuffer jsonBuffer;
    JsonObject& root = jsonBuffer.createObject();
    root["api_key"] = THING_SPEAK_KEY;
    root["timezone"] = "America/Sao_Paulo";

    for (int i = 0; i < values->count(); i++) {
        int id = values->keyAt(i);
        float value = values->valueAt(i);
        root["field" + String(id)] = value;
    }

    String postBody;
    root.printTo(postBody);

    httpClient.setPostBody(postBody);
    httpClient.setRequestContentType("application/json");
    httpClient.downloadString(THING_SPEAK_URL, HttpClientCompletedDelegate(&ThingSpeakClass::onDataSent, this));
}
```

Fonte: elaborado pelo autor.

O Quadro 13 exibe o método `thingSpeakSendCallback` que é chamado após o envio dos valores dos sensores para o servidor ThinkSpeak. O método inicia verificando se o servidor WebServer está ativo ou algum alerta foi disparado. Caso sim, chama o método `readSensors` após 40 segundos e inicia novamente o ciclo de leitura dos sensores. Caso não, chama o método `deepSleep` e coloca o hardware para “dormir” por 40 segundos.

Quadro 13 – Método de *callback* após o envio

```
void FishmingClass::thingSpeakSendCallback(bool successful, DateTime dateTime) {
    if (WebServer.isActive() || AlertController.fired()) {
        int sleepSeconds = 40;
        timer.initializeMs(sleepSeconds * 1000, TimerDelegate(&FishmingClass::readSensors, this)).startOnce();
        Serial.println("----> Calling readSensors function in " + String(sleepSeconds) + " seconds");
    } else {
        int sleepSeconds = 40;
        Serial.println("----> Sleeping for " + String(sleepSeconds) + " seconds");
        System.deepSleep(sleepSeconds * 1000);
    }
}
```

Fonte: elaborado pelo autor.

3.3.3.2 Aplicativo mobile

Após a inicialização do aplicativo, é verificado se as configurações de rede já estão salvas. A classe `SmartConfigService` é responsável por esta configuração. No Quadro 14 é possível observar a implementação do método `isConfigured` desta classe, que retorna um valor booleano indicando se o Fishming já está configurado. Esse valor é verificado a partir da variável `fishmingIP` que é coletada no armazenamento local. Caso a variável esteja não-nula, indica que o aplicativo já está configurado.

Quadro 14 - Método de verificação da configuração do aplicativo

```
35    isConfigured(): Promise<boolean> {
36        return new Promise<boolean>((resolve, reject) => {
37            this.storage.get('fishmingIP').then((fishmingIP) => {
38                console.log('FishmingIP is: ' + fishmingIP);
39                this.fishmingIP = fishmingIP;
40                resolve(fishmingIP != null);
41            });
42        });
43    }
```

Fonte: elaborado pelo autor.

Caso seja a primeira inicialização do aplicativo (retorno falso do método `isConfigured`), é feito a configuração do hardware pelo aplicativo na rede Wi-Fi local. Para que isso seja realizado, é feito a captura das informações da rede Wi-Fi conectada por meio do plugin `WifiWizard`, conforme pode-se observar no Quadro 15. Nas linhas 17 e 23 são coletados o Service Set Identifier (SSID) e o Basic Service Set Identifier (BSSID) da rede local, respectivamente.

Quadro 15 - Coleta do SSID e BSSID da rede Wi-Fi conectada

```
17    WifiWizard.getCurrentSSID((ssid: string) => {
18        ssid = ssid.substring(1, ssid.length - 1);
19        console.log('WifiWizard currentSSID:' + ssid)
20        this.currentSSID = ssid;
21    });
22
23    WifiWizard.getCurrentBSSID((bssid: string) => {
24        console.log('WifiWizard currentBSSID:' + bssid);
25        this.currentBSSID = bssid;
26    });
```

Fonte: elaborado pelo autor.

Após a coleta dos dados, é realizado a configuração por meio da tecnologia SmartConfig. É utilizado o plugin `esptouch` para iniciar a configuração, conforme pode ser visto no Quadro 16. Após a conexão com sucesso, é salvo o Internet Protocol (IP) do Fishming no armazenamento local (linha 49).

Quadro 16 - Método que inicia a configuração do SmartConfig

```
45  start(password: string): Promise<boolean> {
46      return new Promise<boolean>((resolve, reject) => {
47          if (typeof esptouchPlugin !== 'undefined' && this.currentSSID != null && this.currentBSSID != null) {
48              esptouchPlugin.smartConfig(this.currentSSID, this.currentBSSID, password, 'NO', 1, (fishmingIP: string) => {
49                  this.storage.set('fishmingIP', fishmingIP);
50                  resolve(true);
51              }, function (error) {
52                  resolve(false);
53              });
54          }
55      });
56  }
```

Fonte: elaborado pelo autor.

A classe `Sensors` é responsável pelo gerenciamento dos sensores. No Quadro 17 pode-se observar o método `updateFields`, que utiliza o serviço `thingSpeakService` para fazer uma requisição HTTP para o servidor ThingSpeak para coletar os dados dos sensores. Em cada atualização, salva-se a data atual para que a próxima requisição pegue apenas os dados deste instante em diante (linha 50). Na linha 51, todos os dados são salvos no armazenamento local.

Quadro 17 - Método que atualiza os sensores com o ThingSpeak

```
38  updateFields(lastEntryDate?: Date) {
39      this.thingSpeakService.getFields(lastEntryDate).subscribe((feed: ThingSpeakFeed) => {
40          if (feed.lastEntry.date != null) {
41              if (this.feed == null) {
42                  this.feed = feed;
43              } else {
44                  for (let field of this.feed.fields) {
45                      let newEntries: Array<ThingSpeakEntry> = feed.getField(field.id).entries;
46                      field.entries = field.entries.concat(newEntries);
47                  }
48              }
49
50              this.feed.lastEntry = feed.lastEntry;
51              this.storage.set('feed', JSON.stringify(this.feed))
52          }
53      });
54  }
```

Fonte: elaborado pelo autor.

A classe `FishmingService` é responsável pela comunicação entre o aplicativo e o Fishming pela rede local. Ela é utilizada para definir o estado de um atuador e definir a lista de alertas. O Quadro 18 exibe os principais métodos da classe, entre eles o método `keepAlive` que é utilizado pelo aplicativo para manter o Fishming com o servidor HTTP ativo aguardando um comando, caso o usuário fique com o aplicativo aberto sem tomar uma ação. O método `setActuactorEnabled` é utilizado para definir o estado de um atuador e o método `saveAlerts` recebe a lista de alertas que deve ser salva e envia para o Fishming.

Quadro 18 - Principais métodos da classe FishmingService

```
61    keepAlive(): Observable<string> {
62        return this.http.get(this.getFishmingURL() + '/keepAlive')
63            .map((res:Response) => res.toString())
64    }
65
66    setActuatorEnabled(item: any): Observable<string> {
67        let headers: Headers = new Headers({ 'Content-Type': 'application/json' });
68        let body = JSON.stringify(item);
69        return this.http.post(this.getFishmingURL() + '/actuator', body, headers)
70            .map((res: Response) => res.toString());
71    }
72
73    saveAlerts(alerts: Array<any>): Observable<string> {
74        let headers: Headers = new Headers({ 'Content-Type': 'application/json' });
75        let body = JSON.stringify(alerts);
76        return this.http.post(this.getFishmingURL() + '/alerts', body, headers)
77            .map((res: Response) => res.toString());
78    }
```

Fonte: elaborado pelo autor.

3.3.4 Operacionalidade da implementação

Nesta seção é abordada a operacionalidade do sistema Fishming desenvolvido neste trabalho. É demonstrado a configuração inicial do aplicativo e algumas telas disponíveis no aplicativo.

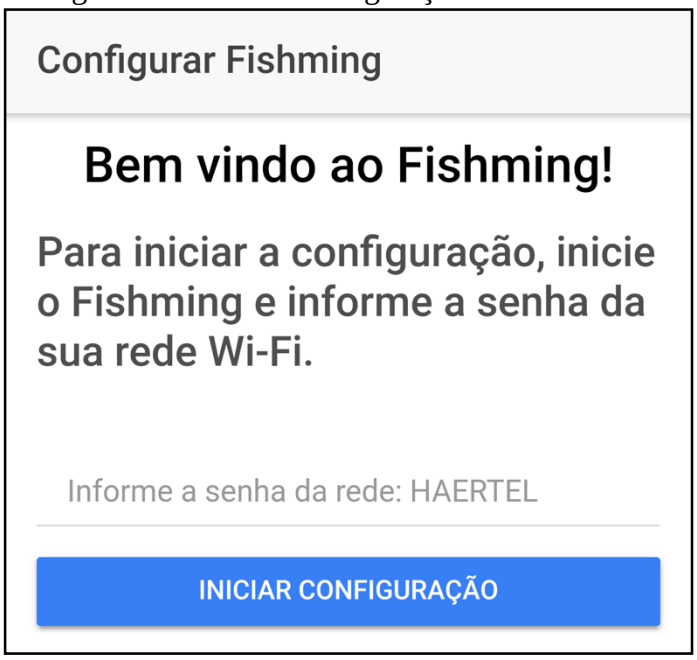
Para a execução do Fishming cada sensor ou atuador é conectado a central através de sua referente entrada, e então liga-se o aparelho. O Fishming irá iniciar e aguardar a conexão com a rede Wi-Fi, que é configurada pelo aplicativo Fishming Mobile. Na Figura 21 pode-se observar o Fishming instalado e com todos os sensores conectados.

Figura 21 –Fishming instalado com todos os sensores conectados

Fonte: elaborado pelo autor.

Depois de deixar o Fishming aguardando a conexão, inicia-se o aplicativo. No primeiro acesso o usuário depara-se com a tela de configuração da rede local, exibida na Figura 22. É solicitado que o mesmo inicie o Fishming e informe a senha da rede Wi-Fi conectada atualmente, para iniciar a configuração.

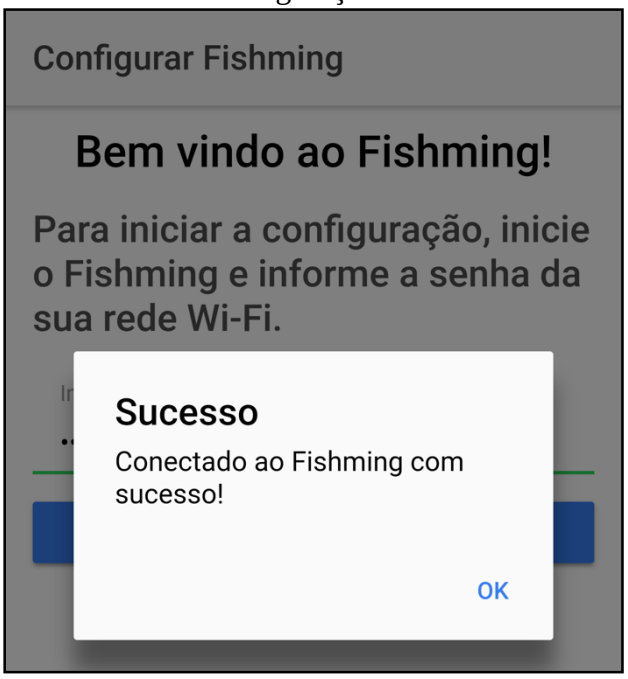
Figura 22 - Tela de configuração da rede Wi-Fi



Fonte: elaborado pelo autor.

Após informar a senha da rede e clicar no botão iniciar, o aplicativo inicia a configuração do SmartConfig. Depois de conectado com sucesso, é exibido uma mensagem de sucesso e o usuário é redirecionado para a tela inicial do aplicativo, conforme pode-se ver na Figura 23.

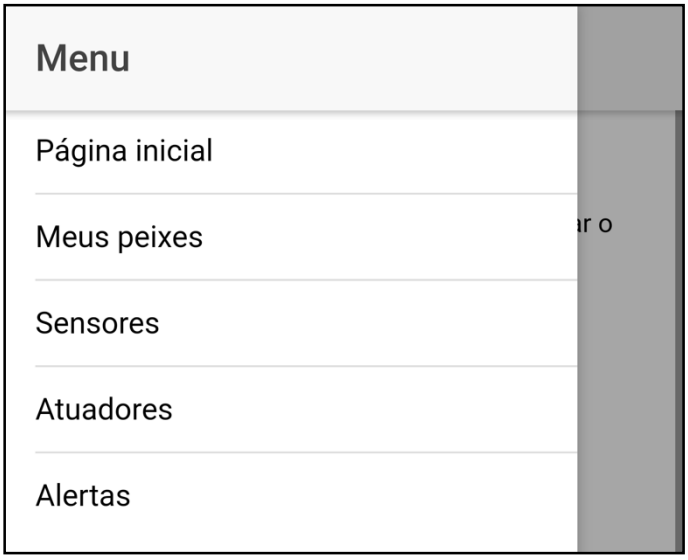
Figura 23 - Tela de configuração realizada com sucesso



Fonte: elaborado pelo autor.

Após realizar a configuração do Fishming, o usuário é redirecionado para o a tela principal do sistema que conta com um menu para navegação entre os recursos. Conforme pode-se observar na Figura 24, as opções são: Página Inicial, Meus Peixes, Sensores, Atuadores e Alertas.

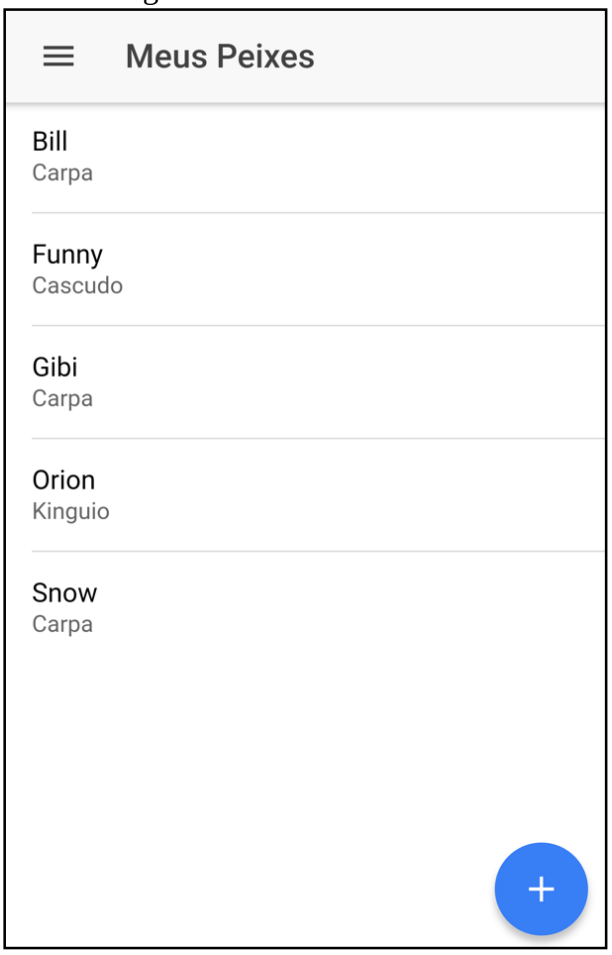
Figura 24 - Tela inicial com menu aberto



Fonte: elaborado pelo autor.

Ao acessar a opção `Meus Peixes` o usuário poderá criar e editar as informações sobre os seus peixes, conforme pode-se observar na Figura 25. Durante o cadastro do peixe, é necessário informar o nome e a espécie do peixe. Os dados são salvos sempre no armazenamento do smartphone.

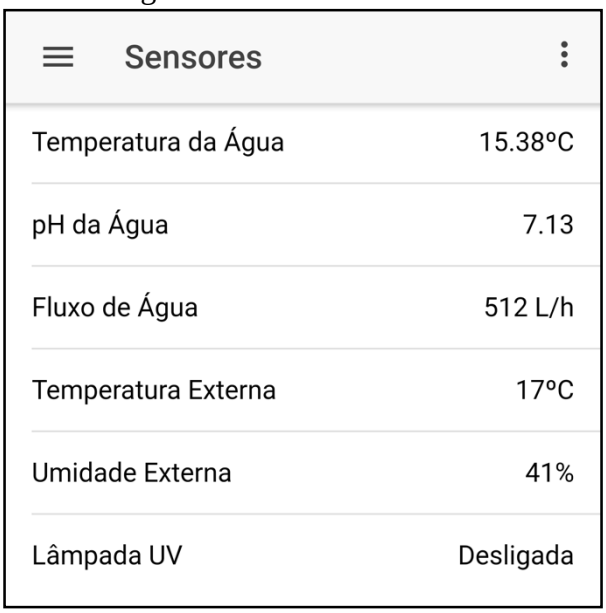
Figura 25 - Tela Meus Peixes



Fonte: elaborado pelo autor.

A Figura 26 exibe a tela de sensores do aplicativo, em que o usuário pode verificar o valor atual de cada sensor. A tela exibe a lista de sensores e ao selecionar um sensor o usuário é levado para a tela detalhada do mesmo.

Figura 26 - Tela dos sensores



Fonte: elaborado pelo autor.

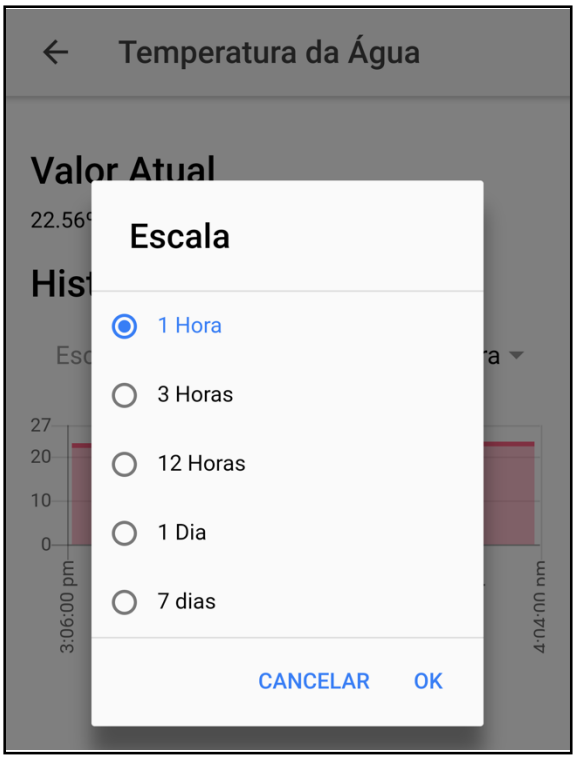
Na tela detalhada de cada sensor é possível visualizar o histórico através de um gráfico, conforme pode-se observar na Figura 27.

Figura 27 - Tela detalhada do sensor

Fonte: elaborado pelo autor.

A escala padrão do histórico é de 1 hora e esse valor pode ser alterado para: 3 horas, 12 horas, 1 dia e 7 dias, conforme pode-se observar na Figura 28.

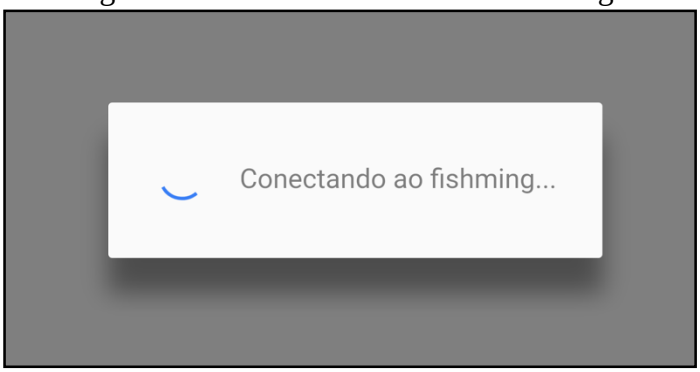
Figura 28 - Alterando a escala do histórico do sensor



Fonte: elaborado pelo autor.

As telas de atuadores e alertas dependem de conexão local com o Fishming, ou seja, o usuário precisa estar na mesma rede em que o aparelho está executando. Ao acessar uma destas opções, caso ainda não tenha se conectado, o aplicativo tenta realizar a conexão com o Fishming por meio do IP salvo na configuração inicial. O aplicativo tenta a conexão durante 60 segundos, e caso não consiga exibe uma mensagem de erro e o usuário é redirecionado para a tela inicial. Na Figura 29 pode-se observar o aplicativo tentando realizar a conexão com o Fishming.

Figura 29 - Tela "Conectando ao fishming"



Fonte: elaborado pelo autor.

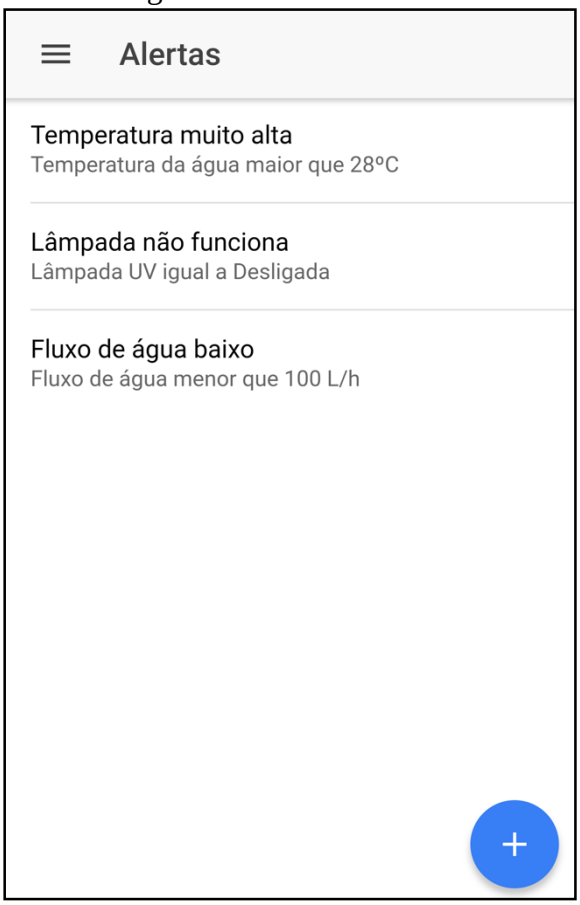
Caso o aplicativo conecte ao Fishming com sucesso, o usuário pode navegar nas telas de atuadores e alertas. A Figura 30 exibe a tela de atuadores, em que é possível ativar e desativar cada um dos atuadores.

Figura 30 - Tela de atuadores

Fonte: elaborado pelo autor.

Na tela de alertas pode-se configurar alertas para serem disparados de acordo com o valor atual de um sensor. A Figura 31 exibe a tela de alertas com alguns alertas criados e opção de criar novos alertas. Ao clicar sobre um alerta é possível editá-lo ou excluí-lo.

Figura 31 - Tela de alertas



Fonte: elaborado pelo autor.

Na criação do alerta, o aplicativo solicita o nome, o sensor que será monitorado, o tipo do alerta (maior que, menor que ou igual a) e o valor para gerar o alerta. Um exemplo de criação de alerta pode ser visto na Figura 32.

Figura 32 - Tela de criação do alerta

Fonte: elaborado pelo autor.

A Figura 33 mostra um exemplo de notificação enviada pelo Fishming. Um alerta foi configurado para notificar quando a temperatura da água está acima de 28ºC, então a notificação foi enviada quando a temperatura da água estava em 32.27ºC.

Figura 33 – Exemplo de notificação

Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

O desenvolvimento deste projeto permitiu o controle e o monitoramento de peixes através de aplicativo móvel. Para confirmar a veracidade dos dados coletados pelos sensores, foram realizados alguns testes. Para o teste com o sensor de fluxo de água, utilizou-se uma bacia e um cronômetro. Três testes foram realizados da seguinte maneira: foi colocado a bacia durante 30 segundos na saída da bomba para coletar a quantidade de litros de água, depois esse valor foi multiplicado por 120 para obter-se o valor em Litros por hora (L/h). Ao mesmo tempo foi visto o valor aferido pelo sensor, e os resultados podem ser vistos na Tabela 1.

Tabela 1 - Testes do sensor de fluxo de água

Litros de água (30 segundos)	**Fluxo de água real (L/h)**	**Fluxo de água aferido pelo sensor (L/h)**
3,9	468	462
3,5	420	425
3,5	420	425

Fonte: elaborado pelo autor.

Conforme pode-se observar na tabela acima, os valores reais e os valores aferidos pelo sensor foram bem parecidos. O que pode ter causado a pequena diferença é o cálculo manual do tempo e do valor em litros, então os resultados foram satisfatórios.

Para a conferência dos valores do sensor de pH, foi utilizado um medido de pH digital que pode ser visto na Figura 34. Esse medidor acompanha duas soluções com pH definido e uma chave para realizar a calibração. Também foi utilizado uma solução de amônia com pH de 10.3 para o teste.

Figura 34 - Medidor de pH digital

Fonte: Mercado Livre (2017).

Foram realizados três testes da seguinte forma: o medidor e o sensor foram mergulhados nas três soluções, que possuem pH 4.00, 6.86 e 10.3. A Tabela 2 exibe os resultados dos testes.

Tabela 2 - Testes do sensor de pH

Valor aferido com o medidor digital			**Valor aferido pelo sensor**		
pH 4.00	**pH 6.86**	**pH 10.3**	**pH 4.00**	**pH 6.86**	**pH 10.3**
4.0	6.9	10.3	4.00	6.87	10.28
4.0	6.9	10.3	3.97	6.87	10.28
4.0	6.9	10.3	3.97	6.87	10.31

Fonte: elaborado pelo autor.

Com base na tabela acima, pode-se afirmar que os resultados foram satisfatórios. O medidor possui precisão de apenas um digito, e o sensor conectado ao Fishming possui dois dígitos de precisão. Os sensores de temperatura e umidade externa foram comparados com dados do site de previsão do tempo Climatempo. A Tabela 3 apresenta o resultados dos testes, que foram realizados em três dias diferentes.

Tabela 3 - Testes do sensor temperatura e umidade externa

Valor do site Climatempo		**Valor aferido pelo sensor**	
Temperatura	**Umidade**	**Temperatura**	**Umidade**
23ºC	60%	24ºC	57%
22ºC	66º	24ºC	61%
19ºC	76%	20ºC	68%

Fonte: elaborado pelo autor.

Conforme os dados da tabela acima, os valores capturados pelos sensores ficaram bem próximos da realidade, então pode-se dizer que o resultado foi satisfatório. Para realizar a averiguação do sensor de temperatura da água foi utilizado um medidor de temperatura

analógico. Três testes foram realizados em dias diferentes e os resultados podem ser vistos na Tabela 4.

Tabela 4 - Testes do sensor de temperatura da água

Temperatura aferida com sensor analógico	Fluxo de água aferido pelo sensor
21ºC	21,56ºC
24ºC	23.32ºC
18ºC	18.63ºC

Fonte: elaborado pelo autor.

Considerando os dados da tabela acima é possível perceber que os resultados foram bem semelhantes. O que pode ter diferido um pouco é a precisão, que no sensor é de 2 dígitos.

Para o teste do sensor de luz, foi apenas ligado e desligado a lâmpada e verificado se o valor capturado pelo sensor estava correto. Em todos os testes os sensores ficaram dentro do esperado, portanto alcançou-se o objetivo de realizar a captura de dados do sensor o mais próximo da realidade.

A conexão do chip com a internet se mostrou bem estável. O chip foi colocado a uma distância de cerca de 10 metros do roteador Wi-Fi e não foram encontrados problemas. Houve um problema com a resolução de DNS para o servidor do ThingSpeak em alguns momentos, por isso foi alterado o código para utilizar o IP fixo para enviar os dados dos sensores.

Inicialmente o D1 mini foi alimentado utilizando a saída de 5 volts do módulo MT3608, porém a sua perda energética foi muito grande, e o módulo ficava poucas horas ligado na bateria. Então foi realizado uma alteração no esquema elétrico para alimentar o D1 mini diretamente pelas baterias. Também foi realizado um teste sem a utilização do sensor de pH. Os resultados dos teste de bateria são exigidos na Tabela 5.

Tabela 5 - Testes de duração da bateria

Situação	Duração da bateria (horas)
D1 mini alimentado pelo módulo MT3608	10
D1 mini alimentado diretamente pelas baterias	26
Projeto sem sensor de pH	48

Fonte: elaborado pelo autor.

Conforme pode-se ver na tabela acima, a duração de bateria aumentou muito sem a utilização do módulo sensor de pH. O principal problema é que o módulo de pH somente pode ser alimentado com 5V e para isso é necessário utilizar o módulo MT3608 para realizar a conversão de 3.3V para 5V. Para medir a a corrente elétrica do projeto foi utilizado um multímetro e os resultados podem ser vistos na Tabela 6.

Tabela 6 – Corrente elétrica

Modo de utilização	Corrente
Normal	90mAh
Deep sleep	16mAh

Fonte: elaborado pelo autor.

O gasto da bateria foi de aproximadamente 90mAh quando ligado no modo normal. No modo deep sleep a utilização foi de cerca de 16mAh, principalmente pelo módulo sensor de pH, utilizando em torno de 15mAh.

O Quadro 19 apresenta um comparativo entre as características mais relevantes dos trabalhos correlatos apresentados e as características do trabalho desenvolvido.

Quadro 19 - Comparativo entre os trabalhos citados

Características mais relevantes	Trabalhos correlatos			Haertel (2017)
	Seneye	Puratek Insight 24/7	Apex AquaController	Fishming
Medição de temperatura da água	Sim	Sim	Sim	Sim
Medição de pH da água	Sim	Sim	Sim	Sim
Medição de vazão de água da bomba	Não	Não	Não	Sim
Medição de luminosidade	Sim	Sim	Não	Não
Medição de temperatura externa	Não	Sim	Não	Sim
Medição de umidade externa	Não	Não	Não	Sim
Consulta do histórico através de gráficos	Sim	Sim	Sim	Sim
Possiblidade de criar alertas	Sim	Sim	Sim	Sim
Necessário conexão com computador	Sim	Não	Não	Não
Visualização através de smartphone	Sim	Sim	Sim	Sim
Conexão Wi-Fi	Não	Não	Não	Sim

Fonte: elaborado pelo autor.

Através das informações disponíveis no quadro acima, é possível perceber que o Apex AquaController e o Puratek Insight 24/7 são os mais completos, mas são extremamente caros. O Seneye é uma opção barata para realizar o monitoramento do aquário, porém sem nenhuma opção de controle do aquário e peca pela obrigatoriedade de conexão com o computador. No Brasil, existem poucas opções, o que se encontra são os modelos descritos acima, mas com um preço muito elevado. O Apex AquaController é possível encontrar por cerca de R$2300 na Empório Aquático (EMPÓRIO AQUÁTICO, 2016).

O trabalho proposto se mostra interessante por não ter a necessidade de utilizar um computador para visualizar os dados dos sensores, tudo é feito por meio de um aplicativo

mobile. Também é relevante por utilizar um módulo de baixo custo e, além disso, que ainda possui a tecnologia Wi-Fi, eliminando a necessidade de utilizar uma conexão cabeada.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um sistema para controle e monitoramento de peixes, batizado Fishming. Os principais objetivos eram desenvolver um hardware utilizando o módulo ESP8266 para capturar dados referentes ao ambiente e desenvolver um aplicativo para a plataforma Android para exibir os dados coletados pelo hardware. Conforme apresentado, foi concebido um protótipo e um aplicativo que demonstram a viabilidade do projeto.

As ferramentas utilizadas para a especificação e durante o desenvolvimento atenderam bem as necessidades identificadas. Em relação ao hardware e componentes utilizados, o chip Wemos D1 mini utilizado supriu as necessidades do projeto, apesar de algumas dificuldades com a utilização de alguns pinos digitais. Os sensores obtiveram resultados satisfatórios e supriram as necessidades do projeto.

Na parte de software, o framework Sming utilizado mostrou-se estável e completo, com todas as bibliotecas necessárias. Houve problemas de compilação quando tentou-se utilizar o sistema de arquivos presente no framework, por isso os dados foram gravados diretamente na EEPROM. Porém sua utilização foi importante, visto que existem poucos trabalhos utilizando o chip ESP8266 sem a IDE do Arduino. No desenvolvimento do aplicativo Android, o framework Ionic se demonstrou satisfatório e foram encontradas as bibliotecas necessárias para a implementação da tecnologia Smart Config.

O trabalho foi importante por implementar uma forma de automatização para facilitar a criação de peixes domésticos. Também contribuiu com a utilização do módulo ESP8266 e na sua programação sem a utilização do Arduino. Os resultados obtidos foram satisfatórios, então pode-se concluir que o trabalho alcançou os objetivos propostos. Um ponto que pode ser ressaltado é o seu preço, que ainda pode ser diminuído e deixar o projeto com um custo mais acessível.

4.1 EXTENSÕES

Para dar continuidade ao trabalho, sugerem-se as seguintes extensões:

a) criar de um servidor para guardar os dados coletados pelo sensor. Com isso o projeto não ficaria dependendo de um serviço de terceiros;
b) substituir o módulo Wemos D1 mini por um módulo ESP-12, para diminuir o custo do projeto. Esse módulo foi utilizado para ter mais facilidade no desenvolvimento (conexão USB) e na prototipagem, porém poderia ser substituído por um módulo com custo menor;

c) criar um esquema de autenticação para permitir que mais usuários controlem o mesmo ambiente;
d) criar uma interface web para utilização do sistema;
e) adicionar um display para verificar as informações diretamente do hardware, sem a necessidade de conexão com o aplicativo.

REFERÊNCIAS

ALIEXPRESS. **ESP8266 remote serial Port WIFI wireless module.** Disponível em: < https://www.aliexpress.com/item/New-version-ESP-12E-replace-ESP-12-ESP8266-remote-serial-Port-WIFI-wireless-module/32339917567.html> Acesso em 20 mar. 2016.

ASAY, Matt. **The Internet Of Things Will Need Millions Of Developers By 2020.** [S.l.], 2014. Disponível em: <http://readwrite.com/2014/06/27/internet-of-things-developers-jobs-opportunity/>. Acesso em 16 mar. 2016.

ASHTON, Kevin. Internet das Coisas, nova revolução da conectividade. **Inovação em Pauta**. Porto Alegre, n. 18, p. 6-9, 14 dez. 2014.

BENCHOFF, Brian. **An SDK for the ESP8266 wifi chip.** [S.l.], out 2014. Disponível em: <http://hackaday.com/2014/10/25/an-sdk-for-the-esp8266-wifi-chip/ >. Acesso em 16 mar. 2016.

BULK REEF SUPPLY. **Neptune System Apex Controllers.** [S.l], 2016. Disponível em: <http://www.bulkreefsupply.com/aquarium-monitors-controllers/neptune-systems-aquarium-controllers.html>. Acesso em 24 mar. 2016.

CARVALHO, Roberto L.; PESSANHA, Lavínia D. R. **Relação entre famílias, animais de estimação, afetividade e consumo: estudo realizado em bairros do Rio de Janeiro.** Santa Maria, set. 2012. Disponível em: <http://periodicos.ufsm.br/sociaisehumanas/article/download/6562/pdf>. Acesso em: 11 mai. 2016.

DESSBESELL, Elton et al. Desenvolvimento e construção de máquina para alimentação automática de pequenos animais. In: CONGRESSO BRASILEIRO DE ENGENHARIA AGRÍCOLA, 43., 2014, Campo Grande. **Anais...** Panambi: UNIJUÍ, 2014.

EMPÓRIO AQUÁTICO. **Neptune Systems Apex Controller.** São Paulo, 2016. Disponível em <http://www.emporioaquatico.com.br/neptune-systems-apex-controller.html>. Acesso em 25 mar. 2016.

ESPRESSIF SYSTEMS. **Espressif Smart Connectivity Platform: ESP8266.** [S.l], out. 2013. Disponível em: <https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf>. Acesso em: 20 mar. 2016.

_____. **ESP-Touch Overview.** [S.l], 2017. Disponível em: <https://espressif.com/en/products/software/esp-touch/overview>. Acesso em: 14 mai. 2017.

FILIPEFLOP. **Sensor de Luz LDR**. Florianópolis, 2017. Disponível em: <http://www.filipeflop.com/pd-10fd75-sensor-de-luz-ldr.html>. Acesso em: 07 mai. 2017.

GLOBO. **Aumento da procura por aquários beneficia fabricantes e lojistas.** [S.l], abr. 2014. Disponível em: <http://g1.globo.com/economia/pme/noticia/2014/04/aumento-da-procura-por-aquarios-beneficia-fabricantes-e-lojistas.html>. Acesso em: 10 mai. 2016.

_____. **MG concentra o principal polo de criação de peixe ornamental do Brasil.** São Francisco do Glória, jun. 2017. Disponível em: <http://g1.globo.com/economia/agronegocios/globo-rural/noticia/2017/06/mg-concentra-o-principal-polo-de-criacao-de-peixe-ornamental-do-brasil.html>. Acesso em: 19 jun. 2017.

GROOVER, Mikell P. **Automation, Production Systems, and Computer-Integrated Manufacturing**. 3. ed. Bethlehem, 2008. 840 p.

HAOYU. **G1" Water Flow Sensor.** Shenzhen, 2017. Disponível em: <https://learn.sparkfun.com/tutorials/voltage-dividers>. Acesso em: 06 mai. 2017.

HAWKINS, George. **CC3000 Smart Config - transmitting SSID and keyphrase.** [S.l.], out. 2013. Disponível em: <http://depletionregion.blogspot.com.br/2013/10/cc3000-smart-config-transmitting-ssid.html>. Acesso em: 14 mai. 2017.

HILL, Nathan. **Seneye aquarium monitor review.** [S.l], nov. 2011. Disponível em: <http://www.practicalfishkeeping.co.uk/content.php?sid=4491>. Acesso em 24 mar. 2016.

KOLBAN, Neil. **Kolban's Book on the ESP8266**. [S.l.], out. 2015. Disponível em: < https://vk.com/doc2272082_437091639>. Acesso em: 15 mar. 2016.

LIRA. Davi. **Internet das Coisas.** [S.l], dez. 2013. Disponível em: <http://porvir.org/wiki/internet-das-coisas/>. Acesso em: 18 mar. 2016.

MERCADO LIVRE. **Medidor Ph Digital Água Lcd Phmetro Atc + Bateria Sashê Case.** [S.l.], 2017. Disponível em: <http://produto.mercadolivre.com.br/MLB-667282293-medidor-ph-digital-agua-lcd-phmetro-atc-bateria-sash-case-_JM>. Acesso em: 23 mai. 2017.

MURATORI, J. R.; DAL BÓ, P. H. Automação residencial: histórico, definições e conceitos. **O Setor Eletrônico.** Ed. 62. P.70-77. Março, 2011.

NEPTUNE SYSTEMS. **Apex – Neptune Systems.** Morgan Hill, 2015. Disponível em: <https://www.neptunesystems.com/products/apex-controllers/apex-controller-system/>. Acesso em 24 mar. 2016.

PURATEK. **Aquarium monitor and controller system Puratek.** Pompano Beach, 2016. Disponível em: <http://puratek.com/products/insight-monitor-controller/insight-247-home>. Acesso em 24 mar. 2016.

ROBOTISTAN. **DHT11 Temperature and Humidity Modul.** Istanbul, 2017. Disponível em: <http://www.robotistan.com/dht11-temperature-and-humidity-modul>. Acesso em: 07 mai. 2017.

ROMANI, Bruno. **Projeto permite monitorar aquário pelo smartphone.** São Paulo, jan. 2015. Disponível em: <http://www1.folha.uol.com.br/tec/2015/01/1576118-projeto-permite-monitorar-aquario-pelo-smartphone.shtml>. Acesso em: 10 mai. 2016.

SANTAELLA, Lucia et al. Desvelando a Internet das Coisas. **Geminis**. São Carlos, ano 4, n. 2, p. 19-32, 2013.

SAS. **O que é a Internet das Coisas (IoT)**. [S.l.], 2016. Disponível em: <http://www.sas.com/pt_br/insights/big-data/internet-das-coisas.html>. Acesso em 16 mar. 2016.

SENEYE. **Aquarium monitor system – Fish tank water sensor – Seneye.** Norwich, 2016. Disponível em: <https://www.seneye.com/ >. Acesso em 24 mar. 2016.

SPARKFUN. **Voltage Dividers.** [S.l], 2017a. Disponível em: <https://learn.sparkfun.com/tutorials/voltage-dividers>. Acesso em: 17 abr. 2017.

_____. **Temperature Sensor - Waterproof (DS18B20).** [S.l], 2017b. Disponível em: <https://www.sparkfun.com/products/11050>. Acesso em: 07 mai. 2017.

ZONAMAKER. **Wemos D1 mini – Introdução e primeiros passos**. [S.l], 2017. Disponível em: < https://www.zonamaker.com.br/wemos-d1-mini-introducao-e-primeiros-passos>. Acesso em: 07 mai. 2017.

APÊNDICE A – Relação dos componentes utilizados

A Tabela 1 contém todos os componentes utilizados para elaboração do projeto, o seu valor unitário e o custo total do projeto. Os componentes com valor em dólar foram comprados nos sites Ali Express e Deal Extreme, e a taxa de câmbio utilizada foi de R$3,20 para 1 dólar. Os componentes com valor em real foram comprados no site da Proesi.

Tabela 7 - Relação dos componentes

Peça	Quantidade	Valor Unitário	Valor Total
Wemos D1 mini	1	US$ 2,60	R$ 8,32
Sensor de pH	1	US$ 12,83	R$ 41,06
Sonda de pH	1	US$ 6,04	R$ 19,33
Sensor de Temperatura DS18B20	1	US$ 1,09	R$ 3,49
Sensor de Fluxo de água FS400A	1	US$ 6,98	R$ 22,34
Sensor de Temperatura e Umidade DHT11	1	US$ 1,61	R$ 5,15
Módulo sensor de luz digital LDR	1	US$ 0,96	R$ 3,07
Módulo relê 2 canais	1	US$ 0,88	R$ 2,82
Módulo conversor DC-DC step-up MT3608	1	US$ 0,44	R$ 1,41
Módulo Carregar de baterias TP4056	1	US$ 0,50	R$ 1,60
Suporte para 3 baterias 18650	1	US$ 2,19	R$ 7,01
Bateria de lítio recarregável 18650 3.7V	3	US$ 1,72	R$ 16,51
Painel de energia solar 1.8W 5V	1	US$ 4,94	R$ 15,81
Placa de Circuito Impresso Ilhada 10X10cm	1	R$ 8,66	R$ 8,66
Resistor Carbono CR25 - 1/4W - 4K7	3	R$ 0,02	R$ 0,06
Resistor Precisão 1% 1/4W - 3K	2	R$ 0,12	R$ 0,24
Resistor Precisão 1% 1/4W - 1K5	1	R$ 0,12	R$ 0,12
Jack P4 para painel, 2,5mm	1	R$ 1,27	R$ 1,27
Conjunto de Conector Circular - 3 Vias - Macho e Fêmea	3	R$ 7,09	R$ 21,27
Conector 5051-2 - Fêmea 2,5mm - 2 Vias	2	R$ 0,10	R$ 0,20
Conector 5045-2 - Macho 2,5mm - 2 Vias	2	R$ 0,16	R$ 0,32
Conector 5051-3 - Fêmea 2,5mm - 3 Vias	5	R$ 0,09	R$ 0,45
Conector 5045-3 - Macho 2,5mm - 3 Vias	5	R$ 0,28	R$ 1,40
Conector 5051-4 - Fêmea 2,5mm - 4 Vias	3	R$ 0,14	R$ 0,42
Conector 5045-4 - Macho 2,5mm - 4 Vias	3	R$ 0,29	R$ 0,87
Terminal 5159T - 2,5mm	31	R$ 0,06	R$ 1,86
Cabo Manga 2X26 AWG - sem Malha	0,2	R$ 0,84	R$ 0,17
Cabo Manga 3X26 AWG - sem Malha	1	R$ 1,06	R$ 1,06
Cabo Manga 4X26 AWG - sem Malha	0,1	R$ 1,28	R$ 0,13
Fio Jumper 2X0,5mm	1	R$ 0,55	R$ 0,55
		TOTAL	**R$ 186,95**

Fonte: elaborado pelo autor.

ANEXO A – Configuração do ambiente de desenvolvimento

Para configurar o ambiente de desenvolvimento é necessário instalar e compilar alguns programas e bibliotecas. Os programas necessários são: Xcode command line tools, Homebrew e Eclipse. Para instalação, executa-se os comandos do Quadro 20

Quadro 20 - Instalação dos programas necessários

```
xcode-select --install

ruby -e "$(curl -fsSL

https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew install Caskroom/cask/eclipse-cpp
```

Fonte: elaborado pelo autor.

Para instalação das ferramentas necessárias para compilação, utiliza-se o gerenciador de pacotes Homebrew instalado anteriormente, conforme o Quadro 21:

Quadro 21 - Instalação das ferramentas necessárias para compilação

```
brew tap homebrew/dupes

brew install binutils coreutils automake wget gawk libtool gettext gperf
gnu-sed --with-default-names grep

export PATH="/usr/local/opt/gnu-sed/libexec/gnubin:$PATH"
```

Fonte: elaborado pelo autor.

É necessário instalar o Kit de desenvolvimento de software (SDK) do ESP. Para o sistema operacional MacOS, é possível encontrar uma versão pré-compilada, conforme o Quadro 22:

Quadro 22 - Instalação dos SDK do ESP8266

```
cd ~/
curl -L -O https://bintray.com/artifact/download/kireevco/generic/esp-
alt-sdk-v1.5.0.258-macos-x86_64.zip

sudo mkdir /opt/esp-open-sdk

sudo tar -zxf esp-alt-sdk-v1.5.0.258-macos-x86_64.zip -C /opt/esp-open-
sdk

sudo chmod -R 775 /opt/esp-open-sdk
```

Fonte: elaborado pelo autor.

No Quadro 23 é possível observar o download e compilação do framework Sming:

Quadro 23 – Download e instalação do framework Sming

```
cd /opt/
sudo git clone https://github.com/SmingHub/Sming.git


cd Sming/Sming
sudo make
```

Fonte: elaborado pelo autor.

O Quadro 24 exibe a a configuração das variáveis de ambiente necessárias:

Quadro 24 – Configuração das variáveis de ambiente

```
sudo /usr/libexec/PlistBuddy -c "Add :LSEnvironment:ESP_HOME string
'/opt/
esp-open-sdk'" /opt/homebrew-cask/Caskroom/eclipse-
cpp/4.5.1/Eclipse.app/
Contents/Info.plist

sudo /usr/libexec/PlistBuddy -c "Add :LSEnvironment:SMING_HOME string
'/opt/
sming/Sming'" /opt/homebrew-cask/Caskroom/eclipse-cpp/4.5.1/Eclipse.app/
Contents/Info.plist

sudo /System/Library/Frameworks/CoreServices.framework/Frameworks/
LaunchServices.framework/Support/lsregister -v -f /opt/homebrew-
cask/Caskroom/eclipse-cpp/4.5.1/Eclipse.app
```

Fonte: elaborado pelo autor.