UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
MESTRADO EM TECNOLOGIAS COMPUTACIONAIS PARA O AGRONEGÓCIO

IZAIAS BATISTA DOS SANTOS

AUTOMAÇÃO DE PROCESSOS DO AGRONEGÓCIO AUXILIADA PELA INTERNET DAS COISAS (IOT): UMA PROPOSTA DE IMPLEMENTAÇÃO DE UM GATEWAY DE IOT PARA SIMPLIFICAR A AUTOMATIZAÇÃO DA AQUICULTURA

DISSERTAÇÃO

MEDIANEIRA
2020

IZAIAS BATISTA DOS SANTOS

AUTOMAÇÃO DE PROCESSOS DO AGRONEGÓCIO AUXILIADA PELA INTERNET DAS COISAS (IOT): UMA PROPOSTA DE IMPLEMENTAÇÃO DE UM GATEWAY DE IOT PARA SIMPLIFICAR A AUTOMATIZAÇÃO DA AQUICULTURA

TRABALHO DE DIPLOMAÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio, da Universidade Tecnológica Federal do Paraná - UTFPR, Campus Medianeira, como requisito parcial para obtenção do título de “Mestre em Tecnologias Computacionais para o Agronegócio” – Área de Concentração: Tecnologias Computacionais Aplicadas à Produção Agrícola.

Orientador: Prof. Dr. André Sandmann
Co-Orientador: Prof. Dr. Bruno Estevão de Souza

MEDIANEIRA
2020

Dados Internacionais de Catalogação na Publicação

S237a

Santos, Izaias Batista dos

Automação de processos do agronegócio auxiliada pela internet das coisas (IoT): uma proposta de implementação de um gateway de IoT para simplificar a automatização da aquicultura / Izaias Batista dos Santos – 2020.

111 f.: il.

Orientador: André Sandmann

Coorientador: Bruno Estevão de Souza

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio. Medianeira, 2020.

Inclui bibliografias.

1. Arduino – (Controlador programável). 2. Algoritmos computacionais. 3. Peixes. 4. Ciência da Computação – Dissertações. I. Sandmann, André, orient. II. Souza, Bruno Estevão de, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio. IV. Título.

CDD: 004

Biblioteca Câmpus Medianeira

Marci Lucia Nicodem Fischborn CRB 9/1219

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação
Mestrado em Tecnologias Computacionais para o Agronegócio

TERMO DE APROVAÇÃO

AUTOMAÇÃO DE PROCESSOS DO AGRONEGÓCIO AUXILIADA PELA INTERNET DAS COISAS (IOT): UMA PROPOSTA DE IMPLEMENTAÇÃO DE UM GATEWAY DE IOT PARA SIMPLIFICAR A AUTOMATIZAÇÃO DA AQUICULTURA

Por

Izaias Batista dos Santos

Esta dissertação foi apresentada no dia 06 de março de 2020 como requisito parcial para a obtenção do título de Mestre em Tecnologias Computacionais para o Agronegócio, da Universidade Tecnológica Federal do Paraná, Campus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Profª. Dr. André Sandmann UTFPR – Câmpus Medianeira (Orientador)	Profª. Dr. Bruno Estevão de Souza IFPR – Câmpus Foz do Iguaçu (Coorientador)
Prof. Dr. Pedro Luiz de Paula Filho UTFPR – Câmpus Medianeira	Profª. Dr. Arcângelo Augusto Signor IFPR – Câmpus Foz do Iguaçu

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso.

A Deus e a seus espiritos por me guiarem,
à minha família, aos meus professores e aos meus amigos...

AGRADECIMENTOS

Ao Prof. Dr. Orientador, André Sandmann e ao Prof. Dr. Coorientador Bruno Estevão de Souza, que contribuíram com importantes orientações em todas as etapas deste trabalho.

À minha tia Francisca e ao meu tio Valdemar por sempre terem me apoiado e incentivado a estudar, ao meu pai Arlindo e à minha mãe Joceli por terem me apoiado em momentos importantes e a todos da minha família, pela confiança e motivação.

Aos professores e colegas de curso, pois juntos trilhamos uma etapa importante de nossas vidas.

Aos amigos do IFPR, pelas colaborações e experiências que compartilhamos diariamente e por contribuírem para minha participação no programa.

Aos amigos militares que conheci no Exército Brasileiro por terem me incentivado e apoiado a estudar e pelas experiências profissionais e de vida compartilhadas.

Aos amigos que trabalharam comigo nas empresas Eits Prognus Group, Itaipu Binacional e FPTI pelas experiências profissionais compartilhadas.

Aos professores de cursos anteriores, por contribuírem para que eu pudesse atingir o nível atual.

Aos demais amigos e colegas, pela força em relação para esta jornada.

“O senhor Deus é a minha fortaleza
e o meu refúgio, minha cidadela.”

Salmos 18

RESUMO

SANTOS, Izaias Batista dos. **Automação de processos do agronegócio auxiliada pela internet das coisas (IoT): Uma proposta de implementação de um *gateway* de IoT para simplificar a automatização da aquicultura.** 2020. Dissertação (Mestre em Tecnologias Computacionais para o Agronegócio) - Universidade Tecnológica Federal do Paraná.

Com o aumento da população mundial exigirá maior eficiência na produção de alimentos e uma forma de aumentar este quesito é usar a tecnologia para automatizar e otimizar os processos de produção. Neste trabalho é apresentada uma proposta de facilitação da automatização dos processos da aquicultura por meio da tecnologia da informação seguindo o conceito de Internet das Coisas, (IoT) que faz parte da definição do termo plataformas emergentes, tal termo denota um conjunto de tecnologias que surgiram a partir de 2010 quando então ganhou popularidade e se tornou importante, tanto na área da tecnologia da informação (TI), como a aplicação da IoT na automação pode facilitar esta atividade, para profissionais do agronegócio sem prévia experiência tecnológica em automação, sendo voltado para profissionais que desejam automatizar os processos da aquicultura. A relevância deste trabalho evidencia o entender que a automação, na produção de peixes e no manuseio dos mesmos, visa a eliminação de trabalhos desgastantes e ou desnecessários das atividades que geram estresses quando executadas de forma errônea e ajuda a criar demanda por mão de obra especializada que geraria oportunidades de empregos nas quais os trabalhadores são mais valorizados. Fora feita revisão bibliográfica sobre: Internet das Coisas (IoT), automação, eletrônica, arduino e algoritmos para monitoramento de dados, tendo como objetivos: avaliar conceitos, coletar melhores práticas, mensurar os resultados apresentados nas bibliografias encontradas e apresentar os testes realizados em campo. Foram utilizados métodos quantitativos de instrumentação, por meio de sensores para coleta dos dados, e experimento, com a aplicação dos componentes implementados em campo. O *gateway* foi implementado e implantado e os dados de leitura dos sensores e atuadores foram coletados e registrados.

Palavras-Chave: Arduino. Peixe. Sensor. Atuador.

ABSTRACT

SANTOS, Izaias Batista dos. **Automation of agrobusiness processes assisted by the internet of things (IoT): A proposal for implementation of an iot gateway to simplify the automation of aquaculture.** 2020. Dissertation (Master Degree in Computational Technologies for Agrobusiness) - Universidade Tecnológica Federal do Paraná.

With the increase in the world population, it will require greater efficiency in food production and one way to increase this aspect is to use technology to automate and optimize production processes. This work presents a proposal to facilitate the automation of aquaculture processes through information technology following the concept of Internet of Things (IoT), which is part of the definition of the term emerging platforms, this term denotes a set of technologies that have emerged from 2010 when it then gained popularity and became important, both in the area of information technology (IT), as the application of IoT in automation can facilitate this activity, for agribusiness professionals without previous technological experience in automation, being focused on professionals who want to automate aquaculture processes. The relevance of this work shows the understanding that automation, in fish production and handling, aims at eliminating stressful and unnecessary work from activities that generate stresses when performed erroneously and helps to create demand for specialized labor that would generate job opportunities in which workers are most valued. A bibliographic review was made on: Internet of Things (IoT), automation, electronics, arduino and data monitoring algorithms, with the purpose of: evaluating concepts, collecting best practices, measuring the results presented in the bibliographies found and presenting the tests performed in the field. Quantitative methods of instrumentation were used, using sensors for data collection, and experiment, with the application of the components implemented in the field. The gateway was implemented and deployed and the reading data from the sensors and actuators were collected and recorded.

Keywords: Arduino. Fish. Sensor. Actuator.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
ET	Eletronic Tongue
GWT	Google Web Toolkit
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
LCD	Liquid Cristal Display
JEE	Java Enterprise Edition
JPA	Java Persistence API
JSF	Java Server Faces
LAN	Local Area Network
PIB	Produto Interno Bruto
RAS	Recirculation Aquaculture System
RFID	Radio-Frequency Identification
RPC	Remote Procedure Call
ROI	Return on Investment
RTC	Real Time Clock
TI	Tecnologia da Informação
TSS	Totais de Sólidos em Suspensão
WAN	Wide Area Network
WSN	Wireless Sensor Networks

SUMÁRIO

1 INTRODUÇÃO

A aceleração contínua das inovações tecnológicas demanda cada vez mais eficiência na produção, pois a competitividade aumenta em virtude das facilidades de conexão entre as fronteiras, que em um rol de vantagens, tem uma que é a facilitação do comércio de produtos.

Entre os produtos que tem o comércio facilitado estão os agrícolas e quando se trata do comércio internacional, os produtores que conseguem reduzir os custos de produção e aumentar a qualidade ganham vantagem competitiva no setor de “commodities” que, de acordo com Sarquis (2011), são fundamentais para assegurar saldos comerciais e amenizar restrições de poupança externa do Brasil, podendo ser fortalecidas com maior agregação de valor nos segmentos industriais derivados da agropecuária entre outros setores.

Uma forma de reduzir os custos de produção e aumentar a qualidade é utilizar a Tecnologia da Informação (TI) a favor desta demanda, e um termo da TI que pode ser usado é o de Internet das Coisas (*Internet of Things* - IoT).

A IoT faz parte do conceito de plataformas emergentes que consiste em um termo que denota um conjunto de tecnologias que surgiram a partir de 2010. O termo ganhou popularidade e se tornou importante também na área de TI.

O termo IoT é um conceito que consiste na capacidade de conexão com qualquer tipo de dispositivo a fim de permitir com isto um alto nível de interoperabilidade, que é a capacidade de tornar flexível a integração de sistemas desenvolvidos em plataformas ou linguagens heterogêneas.

Conforme Ji et al. (2015), a IoT é um conceito emergente apresentado nos últimos anos. É uma rede de objetos habilitados para Internet, bem como os serviços de rede que interagem com esses objetos.

Alguns exemplos que podem auxiliar no entendimento da IoT são: relógios inteligentes, óculos inteligentes como o google glass, tênis com sensores e IP que passam informações de corrida em tempo real as quais podem ser analisadas por sistemas de computador.

Na área de negócios a IoT oferece diversos benefícios dentro de uma série

de áreas das quais é possível destacar; comércio, eventos, agricultura, saúde, energia, automação e transporte. Na área de agricultura é possível utilizar a IoT para conectar sensores de detecção de umidade a fim de obter mais precisão no controle da irrigação, câmeras IP para análise de imagens visando identificar a presença de anomalias tais como: insetos, doenças, etc.

Para o cenário ideal, no ramo do agronegócio, deveria ser possível gerir as atividades do campo de forma mais fácil e precisa, otimizando a utilização dos recursos e evitando desperdícios no intuito de torná-la mais eficiente e efetiva sendo um exemplo de recurso a água que segundo Karim (2017) devido às mudanças climáticas o suprimento de água tem se tornado escasso e em virtude disso existe a necessidade urgente de irrigar eficientemente a fim de otimizar a utilização da água.

Todavia alguns desperdícios que ocorrem no campo, quando não há sistemas inteligentes de suporte à decisão são, por exemplo: excesso de irrigação, quando já existe previsão de chuva, exagero na aplicação de produtos químicos como adubos e agrotóxicos quando não há necessidade, etc. Para Karim (2017) a implementação de técnicas inteligentes de irrigação que melhoram a eficiência do uso da água ajudará os agricultores a tornar suas atividades mais rentáveis e, ao mesmo tempo, melhorar a sustentabilidade da agricultura.

É apropriado considerar, que o exagero na aplicação de agrotóxicos pode causar danos à saúde, tornando a aplicação precisamente controlada por tecnologia da informação relevante, haja visto que o ser humano pode cometer falhas e que num sistema bem elaborado auxiliaria em acertos significativos.

No presente trabalho o conceito de IoT é aplicado ao ramo do agronegócio com uma abordagem voltada para o complexo da aquicultura que de acordo com Kotha; Gupta (2017), quando a IoT é usada na criação de sistemas inteligentes para a aquicultura, o agricultor pode monitorar facilmente as informações em relação à água, isto é, pH da água, sua temperatura e outros detalhes. Usando tal sistema o agricultor prevê e amplia o crescimento dos peixes.

O foco, em tecnologias computacionais que podem ser utilizadas para implementar a automatização dos processos da aquicultura no universo de tecnologias computacionais para o agronegócio, fundamentam essa pesquisa. O alvo, produtores do ramo do agronegócio, precisamente os aquicultores. Tendo em

vista que estes são os principais beneficiados pela redução do custo de produção para o setor e para os que possuem maior amplitude de conhecimento a respeito das demandas do complexo da aquicultura.

É importante realizar estudos voltados para a aquicultura tendo em vista que a população mundial vem aumentando e segundo FAO (2018), este aumento também aumenta a quantidade de alimentos exigida para sustentá-la e alguns dos alimentos que vem sendo cada vez mais consumidos são os da produção aquícola.

Os estudos bibliográficos foram realizados na Universidade Tecnológica Federal do Paraná – Campus Medianeira e os experimentos práticos foram realizados no Instituto Federal do Paraná – Campus Foz do Iguaçu. Os instrumentos usados para coletar dados foram sensores eletrônicos e os dados, por estes coletados, foram analisados por meio de algoritmos computacionais programados conforme as restrições da área de aquicultura e da espécie de cultivo selecionada e por meio de observação.

Este trabalho é uma proposta para melhorar as dificuldades de monitoramento da aquicultura, tendo em vista que a automação dos processos é complexa e cara. O diferencial desta proposta de automação está na utilização do conceito de IoT que proporciona um controle mais flexível e menos tolerante a falhas ou erros.

O trabalho está organizado da seguinte forma: Inicialmente é feito uma breve apresentação no resumo e na sequência é feito uma introdução ao tema na qual há também a apresentação da importância, logo após o trabalho é justificado e os objetivos são apresentados, feito isto se inicia a abordagem dos temas estudados na revisão de literatura onde é possível visualizar os assuntos que estão sendo abordados territorialmente e globalmente a respeito da pesquisa deste trabalho, depois os materiais e métodos usados para a pesquisa, são apresentados e por último os resultados alcançados são relacionados e comentados.

1.1 JUSTIFICATIVA

É relevante evidenciar que a automação visa a eliminação dos trabalhos penosos, das atividades redundantes que geram desgaste, físico, psíquico e não-rentável, quando executadas por pessoas despreparadas que demandam necessidade por mão de obra especializada, gerando empregos e valorização

humana, haja vista que os trabalhadores podem trabalhar em funções como na construção, venda, operação e manutenção das máquinas ao invés de executar as atividades que causariam grande desgaste físico e mental.

A aplicação da IoT na automação pode facilitar o trabalho de automatização para profissionais do agronegócio sem prévia experiência em tecnologias de automação e neste trabalho os estudos são mais precisamente voltados para profissionais que desejam automatizar os processos da aquicultura visando amenizar as dificuldades para realizar a automatização desta área.

1.2 OBJETIVOS

Elaborar uma proposta de facilitação da automatização dos processos do complexo da aquicultura por meio da confecção de um *gateway* de IoT com aplicações que amenizem as dificuldades existentes na automação e no trabalho de monitoramento dos processos da aquicultura.

1.2.1 Objetivo geral

Implementar um *gateway* de IoT para simplificar a automatização da aquicultura de tal forma que uma pessoa com poucos conhecimentos sobre tecnologia da informação ou automação consiga automatizar os processos da aquicultura com poucas dificuldades.

1.2.2 Objetivos específicos

a) Desenvolver o *gateway* de IoT;

b) Implantar o *gateway* desenvolvido e mensurar a sua eficiência;

c) Otimizar os processos de automação da aquicultura;

d) Avaliar os custos para realização da atividade de automatização dos processos da aquicultura.

2 REVISÃO DE LITERATURA

Neste capítulo é apresentado a revisão de literatura sobre a aplicação da IoT no agronegócio de forma geral e depois sobre a aplicação no complexo da aquicultura. Na sequência as tecnologias para implementação do conceito de IoT são listadas e detalhadas.

2.1 APLICAÇÃO DA IOT NO AGRONEGÓCIO

Neste tópico é feita uma abordagem da aplicação da IoT no agronegócio e na sequência alguns exemplos de atividades de determinados complexos da produção rural nos quais a IoT pode ser aplicada são listados.

A aplicação do conceito de IoT não é obtida com a simples automação, contudo, esta ajuda a implementar a IoT. O conceito de internet das coisas é mais abrangente, quando automatizamos, fazemos com que uma atividade que exigia um esforço manual seja executada por mecanismos tecnológicos, contudo em muitos casos a interação humana permanece, como por exemplo, dirigindo uma empilhadeira ou operando um controle remoto. O conceito de IoT é mais avançado e neste a interação humana é menos demandada ou seja as máquinas interagem umas com as outras e fazem isto com o auxilio da internet.

> A plataforma de IoT deve fornecer suporte para pesquisadores e desenvolvedores trabalharem no projeto em soluções de protótipos para vários cenários dos domínios de pesquisa da agricultura de precisão, maricultura e monitoramento ecológico. A plataforma deve permitir a rápida criação de bancos de ensaio e protótipos de novas modelagem e funções preditivas. (POPOVIC et al., 2017, p. 256).

De acordo com Ray (2018), um sistema IoT é baseado em dispositivos que fornecem atividades de sensoriamento, atuação, controle e monitoramento. Na IoT os dispositivos podem trocar dados com outros dispositivos conectados e aplicações, ou coletar dados de outros dispositivos e processar os dados localmente ou enviar os dados para a central de servidores ou back-ends de aplicativos baseados em nuvem para processar os dados ou executar algumas tarefas localmente e outras dentro da infraestrutura de IoT baseada em restrições temporais e espaciais (memória, capacidades de processamento, latências de comunicação e de velocidades e prazos).

Em grande parte dos trabalhos analisados os autores elaboraram os

mecanismos de automação utilizando hardware de computadores em conjunto com microcontroladores como o arduino e diversos componentes eletrônicos como resistores, sensores, atuadores, transistores, protoboard, *gateways*, etc.

Um *gateway* de IoT é uma suíte de componentes eletrônicos e softwares que funciona como uma interface mediadora entre sensores, computadores, homens e internet por isso a camada de interface é também conhecida como camada de *gateway*. A camada de *gateway* de acordo com Kotha; Gupta (2018) contém as informações de redes como LAN ou WAN, etc. Ela realiza transformações de dados e torna os dados brutos recebidos adequados para serviços em nuvem e estabelece o caminho para a comunicação de ponta a ponta.

No meio rural são vários os complexos que podem ter parte de seus processos automatizados seguindo os princípios da IoT sendo alguns deles por exemplo: avicultura, suinocultura, aquicultura, hidroponia, aquaponia, irrigação, pecuária, greenhouses, máquinas pesadas, agricultura de precisão, controle de ervas daninhas, controle de qualidade de alimentos, monitoramento de produtos armazenados, etc.

Em um aviário, por exemplo, as tarefas de processos que podem ser automatizadas são: controle da temperatura ambiente, alimentação das aves incluindo alimentadores e bebedouros, ventilação, aquecimento, controle dos gases por exaustão, etc.

De acordo com Camargo (2019), o conforto térmico no interior de instalações de criação de frangos de corte é essencial na obtenção de bons resultados nesta atividade de produção. A avaliação das condições termodinâmicas adequadas requer a medição e controle, geralmente implicando em custos e manutenção especializada. No trabalho do mesmo foi implementado um sistema automatizado cujo objetivo foi monitorar a distribuição de temperatura, umidade relativa e velocidade do ar utilizando sistema: de baixo custo, de código aberto e de fácil uso, com hardware Arduíno e software Scilab para aquisição de dados em tempo real.

Já no caso da suinocultura um mangueirão de porcos também tem vários itens que podem ser automatizados tais como: cortinas de blecaute, alimentadores, bebedouros, controle dos gases por exaustão, ventilação, etc.

Para Bokingkito Jr (2017), na gestão moderna da área de aquicultura, um controle remoto da qualidade da água e cultura intensiva controlada por computador

é a futura tendência para a área da aquicultura.

Na aquicultura é possível automatizar o monitoramento da temperatura da água, nível de pH, nível de oxigênio dissolvido (OD), nível da água, turbidez, controle de alimentador e de aeração, etc.

Segundo Benavent (2018), o monitoramento dos peixes e a inspeção na aquicultura exigem manipulação extremamente delicada do cultivo a fim de evitar danos ou prejuízos, contudo métodos de amostragem são geralmente invasivos, caros, demorados e trabalhosos. Sensores ópticos e sistemas de visão de máquina demonstraram serem métodos muito apropriados para um desenvolvimento mais rápido, mais barato e não invasivo para trabalhar com peixes vivos.

No contexto do plantio hidropônico as plantas crescem apenas com água e componentes químicos que são misturados nesta a fim de nutrir as necessidades das plantas. Itens que podem ser automatizados nesta área são: recirculação da água, irrigação, dosagem de nutrientes, controle do crescimento, monitoramento das pragas, etc.

Outra atividade que pode ter seus processos automatizados é a aquaponia, segundo Romli et al. (2018), a palavra aquaponia surge da junção da aquicultura com a hidroponia, os peixes e outros animais aquáticos produzem resíduos e o sistema de cultivo trata-os como nutrientes e absorve-os no canteiro hidropônico. Após passar pelos canteiros a água limpa retorna ao tanque. Isto é como o sistema de recirculação da aquicultura realmente trabalha.

A integração da aquicultura com a hidroponia por meio da aquaponia traz redução de custo para ambas haja vista que a água dos tanques dos peixes passa pelos canteiros hidropônicos a fim de nutrir as plantas e como já possui os resíduos originados dos peixes, não há necessidade de inclusão de nutrientes químicos adicionais depois a água passa por filtros de argila expandida e volta filtrada para os açudes. Alguns itens que podem ser automatizados na aquaponia são leitura automatizada e monitoramento de: crescimento das plantas, controle da luminosidade, temperatura da água, vazão da recirculação, pH, etc.

No complexo da pecuária pode ser usado balanças em campo para pesagem dos gados a fim de que esta atividade cause menos impacto físico e estresse nos animais, sensores RFID podem ser inclusos individualmente nos animais para controlar a quantidade do rebanho e a geolocalização em tempo real, o estado das cercas pode ser monitorado para que um alerta seja enviado caso algum

fio de arame seja arrebentado.

Nas greenhouses que no Brasil são, também, conhecidas como estufa inteligente é possível instalar sensores para medição de temperatura e umidade do ar e acionar atuadores a fim de manter as variáveis do ambiente interno das estufas em valores ideais, identificar por meio de câmeras e sistemas de analise de imagens a presença de pragas agilizado, assim, a aplicação das medidas necessárias. Vale lembrar que nestes ambientes controlados a presença de pragas como insetos é muito mitigada.

Um exemplo de IoT aplicada à greenhouse é o do trabalho de Wang (2018), que implementou um sistema de gerenciamento, monitoramento e controle do ambiente em estufa que foi desenvolvido baseado no Google Web Toolkit (GWT), usando método de chamada remoto do inglês *remote method call* (RPC) AJAX como método de comunicação entre o navegador e o servidor da web, no sistema foram realizadas funções como: configuração dos parâmetros de aquisição e controle, correspondência adaptativa do banco de dados entre *gateway* e servidor, diagnóstico adaptativo dos parâmetros de monitoramento, o notificador dos parâmetros de monitoramento, a geração adaptativa da interface e conseguinte.

As máquinas pesadas mais atuais já apresentam sistemas embarcados e IoT inclusos na estrutura. Os tratores autônomos, possuem sensores de distancia que possibilitam identificar antecipadamente o risco de colisão com outros veículos por enviar alertas, de determinados problemas, para o trator em tempo oportuno para que haja a frenagem já que são equipados para transmitir mapas geográficos das áreas que estão sendo trabalhadas de forma online.

De acordo com Foughali et al. (2017), a agricultura de precisão pode ser definida como a arte e a ciência do uso da tecnologia para melhorar a produção agrícola. Esta fornece informações pertinentes à agricultura, adequadamente relacionadas a fatores meteorológicos como: temperatura, umidade, sol e vento. Na agricultura de precisão é possível aplicar a IoT em: plantio inteligente do inglês *smart planting*, mapeamento geográfico, controle de pulverização, etc.

Um exemplo de aplicação da IoT à agricultura de precisão é o do trabalho de Castro et al. (2016), que implementou uma rede de sensores sem fio que é capaz de obter medições de umidade do solo de diferentes zonas de um cultivo de morangos e de acordo com os dados coletados determina o tempo de irrigação de uma área

em particular que deve ser irrigada utilizando o método de gotejamento.

No contexto da agricultura de precisão a implementação de técnicas inteligentes de irrigação que melhoram a eficiência e o uso da água ajudará os agricultores a tornar suas atividades mais rentáveis e, ao mesmo tempo, melhorar juntamente a sustentabilidade da agricultura e de acordo com Karim (2017) os sistemas de suporte à decisão no contexto das fazendas é inevitável e a supervisão em tempo real das condições climáticas é a única maneira de conhecer as necessidades de água de uma cultura.

Na irrigação pode ser utilizados sensores de umidade do solo para irrigar de forma eficiente, ou seja, quando realmente houver necessidade, sensores de chuva para não ligar os equipamentos de irrigação de forma desnecessária, controlar os parâmetros de qualidade da água, sistema de alerta para o controle do estresse hídrico de plantas usando a IoT, etc.

Utilizando a IoT na agricultura também é possível fazer a automação do controle de ervas daninhas e conforme Dankhara (2019) atualmente, os robôs inteligentes aprimoraram o sistema de controle de plantas daninhas perceptivas para fornecer tratamento por planta. No entanto, isto requer um classificador de planta daninha que pode separar as plantas e as ervas daninhas analisando dados de imagem usando técnicas de visão computacional, IoT e rotule-as em tempo real. Esses robôs inteligentes baseados em IoT usam modelos pré-treinados armazenados para a classificação de diferentes tipos de combinações de plantas e ervas daninhas que variam ao longo da estação e das colheitas.

A IoT também pode ser aplicada na avaliação da qualidade dos produtos das agroindústrias e um exemplo de utilização está no trabalho de MA (2018), que implementou um sistema de língua eletrônica (ET – *Eletronic Tongue*), sistema inteligente que imita o mecanismo de percepção humano do sabor, para detecção da qualidade do suco de laranja com base na internet das coisas. O sistema consiste em três peças: terminal de detecção portátil, sistema de comunicação sem fio e plataforma de serviço em nuvem.

Durante a detecção, o terminal de detecção portátil foi usado para obter dados de “impressões digitais” de vários tipos de sucos baseados no grande potencial de varredura de pulso e depois transmitiram esses dados para um serviço em nuvem através de um sistema de comunicação sem fio, bem como os métodos

de reconhecimento de padrões utilizado para analisar os dados. Por fim, os resultados foram comparados com o banco de dados interno de “impressão digital” da bebida na plataforma em nuvem, de modo a obter as informações de marca ou adulteração do suco. Neste estudo, o sistema ET desenvolvido foi usado para identificar a marca de suco de laranja e detecção de pureza. Por meio das interfaces para dispositivos móveis, desktop ou web os usuários podem monitorar em tempo real os parâmetros de qualidade dos sucos e a rastreabilidade do produto na sociedade.

A IoT pode ainda ser usada para avaliar em tempo real a qualidade dos produtos armazenados em depósitos como por exemplo o modelo proposto segundo Tervonen (2018), que implementou um sistema para monitoramento de condições durante o período de armazenamento de vegetais em depósito e de acordo com o autor ao armazenar vegetais, é importante que os valores ótimos de umidade e temperatura não se desviem desproporcionalmente, porque exceder os valores limite pode levar a uma deterioração da qualidade. Para atender o fim foram implantados sensores para a leitura das variáveis do ambiente do depósito e um sistema que gera notificações e interfaces com gráficos para facilitar monitoramento e acompanhamento foi desenvolvido.

O conceito de IoT também pode ser aplicado no cultivo de abelhas conforme Debauche et al. (2018), que desenvolveram sistemas automáticos e eficazes para monitorar o comportamento das abelhas. O uso de sensores na agricultura de precisão está se disseminando, especialmente na apicultura de precisão. De fato, a tecnologia da *Wireless Sensor Network* (WSN) tem sido amplamente utilizada por apicultores e muitos pesquisadores para monitorar especialmente as condições ambientais da colmeia, detectar enxames ou contar entrada e saída de abelhas.

A internet das coisas associada à computação em nuvem, oferece possibilidades de monitorar e acompanhar a saúde das colônias de abelhas. Os dados podem ser coletados e enviados automaticamente para um *gateway* em um determinado momento. Os dados para monitoramento das abelhas podem prover de múltiplas origens, como observações pontuais, imagens, vídeos, sons e séries temporais, etc. No passado o monitoramento das atividades das abelhas foi realizado manualmente, mas não foi possível detectar o comportamento de centenas de abelhas dentro e fora das colmeias.

Para atender de forma mais plena o conceito de IoT é necessário também a

inclusão de métodos que adicionem inteligência aos sistemas e uma forma de aplicar tal principio é utilizar o aprendizado de máquina do inglês *machine learning*, na análise de dados coletados pelos sensores. Segundo Mahdavinejad (2018), a IoT consiste em um conjunto de dispositivos que podem transferir dados entre si para otimizar o desempenho destes, essas ações ocorrem automaticamente e sem a necessidade de consciência ou contribuição humana. A IoT inclui quatro componentes principais: 1) sensores, 2) redes de processamento, 3) dados de análise sobre dados e 4) monitoramento do sistema.

Considerando que o volume de dados coletados na IoT vem aumentando, a IoT gera uma grande quantidade de dados caracterizada por sua velocidade em termos de tempo e dependência de localização, com variedade de múltiplas modalidades e qualidade de dados variáveis e para atender o componente 3 o processamento e análises inteligentes dos "*big data"* são a chave para o desenvolvimento de aplicativos inteligentes de IoT. Com sistemas que ajudam na tomada de decisões e a administração dos processos do agronegócio pode ficar mais simplificada o que é o caminho para avançar rumo ao conceito de fazendas inteligentes do inglês *smart farming*.

O conceito de *smart farming* segundo Wolfert (2017), é um desenvolvimento que enfatiza o uso da tecnologia da informação e comunicação no ciclo de gerenciamento ciber-físico das fazendas. Espera-se que as novas tecnologias, como a IoT e a computação em nuvem alavanquem esse desenvolvimento e introduzam mais robôs e inteligência artificial na agricultura.

2.2 AUTOMAÇÃO DO MONITORAMENTO DA AQUICULTURA

Nesse tópico é apresentado a importância do setor da aquicultura no agronegócio, sua representação no PIB, seu peso na balança comercial externa, as características principais da IoT e algumas tecnologias que podem ser integradas com a IoT a fim de atender aos seus princípios serão relacionadas.

2.2.1 Proposta de implementação de um *gateway* de IoT para a automatização de processos da aquicultura

Em virtude do crescimento da população mundial será necessária uma maior produção de alimentos para suprir a demanda global e o Brasil tem a oportunidade de aumentar a produção de alguns setores para atender ao mercado externo.

Segundo FAO (2018), a produção da aquicultura global (incluindo plantas aquáticas) em 2016 foi de 110.2 milhões de toneladas, com a primeira venda estimada em 243.5 bilhões de dólares, números que representam como o setor da aquicultura continua crescendo mais rápido que a maioria dos outros setores de produção de alimentos e o país que mais produziu peixes na aquicultura, no mesmo ano, foi a China seguida por Noruega, Vietnã e Tailândia.

A aquicultura Brasileira conforme dados de FAO (2018) representa o 13º lugar no ranking da média de produção dos maiores produtores de pescado no mundo entre os anos de 2001 e 2016, excluindo a produção de plantas aquáticas, em 2012 o Brasil produziu 707.461 toneladas e em 2013 a produção foi de 1.241.807 toneladas e nos demais anos segundo Peixe BR (2020) a produção Brasileira de peixes de cultivo, em toneladas, foi: 2014 (578.800), 2015 (638.000), 2016 (640.510), 2017 (691.700), 2018 (722.560) e 2019 (758.006), um crescimento que representa 31%.

A espécie mais cultivada na aquicultura brasileira é a de Tilápia do Nilo (Oreochromis niloticus), sendo que os estados que mais produziram esta espécie em 2019, de acordo com dados de Peixe BR (2020), foram: Paraná, São Paulo, Santa Catarina, Minas Gerais e Pernambuco. O Paraná participa com 33,8% do total e o Brasil reforça a posição de 4º maior produtor de tilápia do mundo.

O Brasil ainda pode melhorar sua posição no ranking de maiores produtores de peixe do mundo tendo em vista a grande extensão marítima e o tamanho das bacias hidrográficas que o país possui.

As principais tecnologias, atualmente, utilizadas no setor são as voltadas para: melhoramento genético, vacinas, identificação eletrônica, biometria, alimentação, probióticos, softwares de gestão: do manejo, do estoque, do

financeiro, etc.

É vantajoso automatizar os processos do complexo da aquicultura tendo em vista que esta ajuda a eliminar as atividades redundantes e rotineiras que geram estresse quando executadas por pessoas, exigindo mão de obra especializada o que cria oportunidades de empregos nas quais as pessoas são mais valorizadas tendo em vista que estas podem trabalhar na construção, venda, operação, manutenção das máquinas, entre outras ao invés de executar as atividades que causam desgaste físico e mental prejudicando a saúde dos trabalhadores.

Uma das vantagens de utilizar um *gateway* de IoT conforme Ray (2018), é que uma camada de *gateway* IoT facilita a infraestrutura do padrão inteligente. A camada de nó da IoT consiste em dispositivos IoT com menos recursos, menos processamento, menos potência e menor capacidade de consumo de energia. O *gateway* IoT vincula os dispositivos da camada de nó da IoT nas bordas da rede a uma infraestrutura de rede central de uma maneira programável remotamente.

Outra vantagem é atender ao requisito modularidade que segundo Lee (2000), agrega benefícios que se tratando de software não haverá necessidade de comprar recursos desnecessários e ainda será possível atualizar os recursos posteriores. No caso do hardware, a seleção de dispositivos de comunicação modulares tais como: transmissores, medidores, sensores e atuadores, significa que um sistema pode ser implementado rapidamente e modificado sem grandes atrasos. O design modular também agrega a vantagem de que uma vez que um sistema de monitoramento e controle de tanque de aquicultura seja projetado e implementado em suas instalações, ele poderá ser facilmente replicado para todos os outros sistemas de tanques.

Para Bokingkito Jr (2017), na gestão moderna da área de aquicultura, um controle remoto da qualidade da água e cultura intensiva controlada por computador é a futura tendência para a área da aquicultura.

Conforme Qiuwei (2015), um exemplo de interação entre máquinas ocorre quando o controlador detectar que a concentração de oxigênio dissolvido é menor que o limite inferior predefinido, instruções serão enviadas para que o aerador seja ligado. Quando detecta que a concentração de oxigênio dissolvido é maior do que o limite superior predefinido, o aerador é então desligado. Segundo Huan et al.

(2018), oxigênio dissolvido (OD) é um dos principais parâmetros de qualidade da água serve também para os produtos desta, logo suas mudanças refletem diretamente na qualidade dos resultados da aquicultura.

> Devido ao balanço entre a atividade fotossintética do fitoplâncton e a atividade respiratória das diferentes comunidades aquáticas (plâncton, peixes e organismos bentônicos), os níveis de oxigênio dissolvido (OD) nos sistemas aquiculturais flutuam durante o dia. (GUIMARÃES; LOHMANN, 2017, p. 37).

É importante considerar que além das tecnologias utilizadas outras podem ser projetadas e implementadas a fim de obter produtos mais eficientes e reduzir os custos para a automatização, um exemplo é o proposto por Chu et al. (2018) que criou um sensor plástico de fibra óptica que permite a detecção simultânea de H2O2 e OD e pode ser usado, por exemplo, para o sensoriamento de OD compensado por H2O2, assim como são os das aplicações da aquicultura.

Segundo Benavent (2018), o monitoramento dos peixes e a inspeção na aquicultura exigem manipulação extremamente delicada do cultivo a fim de evitar danos, contudo, métodos de amostragem são geralmente invasivos, caros, demorados e trabalhosos. Sensores ópticos e sistemas de visão de máquina demonstraram serem métodos muito apropriados para um desenvolvimento mais rápido, mais barato e menos invasivo para se trabalhar com peixes vivos.

Para Jiang (2018), alguns sistemas inteligentes podem contribuir para aumentar a produção e reduzir os custos aplicados na aquicultura de água doce quando usados para monitorizar variáveis ambientais da água em tempo real, tais como concentração de OD na água, temperatura da água, pH, etc.

> É importante desenvolver sistemas de cultura flexíveis, programáveis e modulares, facilitando a produção automática de espécies exigentes, tanto para fins científicos quanto para fins de aquicultura. Na verdade, os sistemas dedicados de cultura devem satisfazer as necessidades fisiológicas dos organismos alvos (temperatura, oxigênio dissolvido, pH, salinidade), reduzir a abundância de matéria orgânica em decomposição e concentração de poluentes (por exemplo, compostos nitrogenados), evitar a introdução de patógenos e reduzir o estresse que pode alterar padrões comportamentais e fisiológicos. (ZUPO, 2017, p. 156).

Um dos importantes controles para a aquicultura é o requisito da qualidade da água e nas bibliografias estudadas o tema mais abordado foi sobre automação da recirculação da água dos açudes visando, conforme Bokingkito Jr (2017), controlar mudanças repentinas no clima e temperatura que afetam a qualidade da água e são algumas das principais causas das mortes dos peixes tendo em vista

que a temperatura da água é uma variável chave de qualidade desta, pois ela influencia todas as demais variáveis de qualidade da mesma e dos organismos aquáticos.

Nos sistemas de recirculação automatizada deve ser dada a devida atenção ao consumo de energia elétrica haja visto que, segundo Schulz (2018), a ventilação da instalação é um dos principais consumidores de energia em um sistema de recirculação de aquicultura (RAS - *Recirculation Aquacultural System*).

Para Zupo (2017), uma unidade de processamento central programável controla as operações, ou seja, mudanças na temperatura da água, leve irradiância, abertura e fechamento de válvulas para descarga de alimentos não utilizados, circulação e filtragem da água e sistemas de desinfecção, de acordo com as informações recebidas por várias sondas. Vários dispositivos podem ser configurados para modificar a circulação e mudanças de água para satisfazer as necessidades dos organismos.

De acordo com Gehlert (2018), os sistemas de recirculação da aquicultura requerem um maior nível técnico de infraestrutura do que os sistemas abertos. Especialmente, em atividades como: tratamento de água, controle de temperatura e suprimento de oxigênio, os desafios são maiores.

A água da recirculação pode ser utilizada na aquaponia, de acordo com Romli et al. (2018), a palavra aquaponia surge da junção da aquicultura com a hidroponia, os peixes e também outros animais aquáticos produzem resíduos e o sistema de cultivo trata-os como nutrientes e absorve-os no canteiro hidropônico, após passar pelos canteiros a água limpa retorna ao tanque, esta é a forma como o sistema de recirculação da aquicultura realmente trabalha.

2.2.2 Controle da qualidade da água para aquicultura

Conforme Embrapa (2013), o ambiente aquático é o meio onde os peixes vivem e desenvolvem-se, estão em constante contato com a água, utilizando-a para a obtenção de oxigênio e liberação de gás carbônico, além de resíduos nitrogenados e outras substâncias de excreção. Em virtude disto os peixes precisam da água em condições específicas para que possam alimentar-se, crescer e se reproduzir.

A água possui aspectos de qualidade que são dos tipos físicos e químicos, os físicos são, por exemplo: temperatura, cor, turbidez e visibilidade ou transparência e alguns químicos são: pH, alcalinidade, dureza, oxigênio dissolvido, nitrogênio amoniacal, nitratos, fosfatos.

> Um peixe fora d'água morre. Um peixe em água de má qualidade também morre! A água boa para o peixe nem sempre é a água boa para bebermos. Ela deve ter componentes na qualidade certa. A produção de um viveiro está relacionada com a qualidade da água que o abastece, embora sejam raros os mananciais que não podem ser aproveitados para a piscicultura. (CECCARELLI, 2000, p. 105).

No contexto da aquicultura para que se tenha uma produção economicamente viável é necessário controlar adequadamente os níveis dos parâmetros físicos e químicos e os fundamentais para a aquicultura estão apresentados na Tabela 01.

Tabela 1 - Parâmetros Físicos e Químicos da Água.

Físicos	Químicos
Temperatura	pH
Coloração	Alcalinidade
Turbidez	Dureza
Visibilidade e transparência	Oxigênio dissolvido
Luminosidade	Amônia
Velocidade da corrente	Salinidade

Fonte: Baldisserotto (2009).

Na sequência alguns dos aspectos de qualidade da água considerados importantes para a aquicultura são detalhados.

2.2.2.1 Oxigênio dissolvido (OD)

O oxigênio dissolvido na água é consumido pela respiração dos peixes e de outros organismos, durante o dia as formas vegetais, incluindo as algas, realizam fotossíntese assim a água fica rica em OD. Durante a noite, quando não há energia luminosa, esses vegetais param a fotossíntese e realizam a respiração, a qual consome oxigênio do ambiente. O nível ideal de OD depende da espécie de peixe cultivada, do sistema de cultivo, da eminência ou não de manejo, etc.

De acordo com Ceccarelli (2000), o nível de OD abaixo de 3,0 mgO_2/L é considerado baixo e acima de 6,0 é o suficiente mas também depende do horário de medição se for entre 06:00 e 08:00 horas da manhã acima de 3,0 mgO_2/L está adequado.

2.2.2.2 Temperatura

A temperatura da água na aquicultura conforme Ceccarelli (2000) é um dos principais fatores que afeta o desenvolvimento e vida dos peixes, pois todas as atividades fisiológicas (respiração, digestão, reprodução, excreção, alimentação, movimentação, defesa imunológica, etc.) estão intimamente ligadas à temperatura da água.

Existem temperaturas ideais para reprodução, crescimento, conversão alimentar, resistência a doenças e manejo para cada espécie ou grupo de peixes. Há também limites superiores e inferiores de tolerância térmica. Para a maioria das espécies de peixes tropicais brasileiros, a faixa térmica ideal é de 24 a 30 ºC.

E para o caso da tilápia que é o peixe escolhido para este projeto de pesquisa a faixa térmica ideal conforme Kubitza (2011) é de 26 a 30 ºC.

2.2.2.3 Coloração

Segundo Ceccarelli (2000), as maiores produções de peixes são obtidas quando a coloração da água é levemente verde. Tal coloração indica a presença de

grande quantidade de algas ou fitoplâncton na água.

2.2.2.4 Turbidez

Conforme Baldisserotto (2009), a turbidez está relacionada com a quantidade de material insolúvel e em suspensão existente na água e que impede a passagem da luz. Sendo assim, experimentos envolvendo variações na turbidez também afetam a intensidade de luz. O material citado pode ser composto de material inorgânico (argila, por exemplo) ou fitoplâncton.

O controle da turbidez, segundo Parra et al. (2018), é importante pois é útil para tomar diferentes ações a fim de evitar mais danos na produção de peixe. Pode ser especialmente valioso para instalações internas com circuito de águas abertas.

Nas instalações onde larvas e reprodutores são mantidos, sensores são cruciais para garantir a qualidade da água nos tanques de produção. No entanto, diferentes tipos de turbidez podem causar diferentes efeitos nos peixes e em virtude disso algumas ações específicas devem ser tomadas. Por esta razão, é necessário ter um método automático para monitorar a turbidez e caracterizá-la. Para o autor, o método mais comum para medir a turbidez é a utilização de sensores ópticos. O sensor óptico funciona emitindo um feixe de luz e detectando a quantidade de luz que chega ao detector. Esta seria uma forma para automatizar a leitura da turbidez na aquicultura.

2.2.2.5 Visibilidade e transparência

De acordo com Ceccarelli (2000), a transparência da água indica genericamente a quantidade de plâncton, de matéria orgânica, de peixes, turbidez decorrente de chuvas, etc. A transparência diminui em função da profundidade e da turbidez. Quer dizer, quanto mais fundo o viveiro e mais barrenta a água, menos luz consegue chegar até o fundo. O raio solar (luz) é a fonte de energia essencial para todos os seres vivos, especialmente para as plantas clorofiladas (principalmente as algas), que produzem oxigênio por meio da fotossíntese.

Devido a isso a transparência é um fator de grande importância para a aquicultura. A transparência que tem relevância para ser mensurada está diretamente relacionada com a existência ou não, na água do viveiro, de pequenos vegetais e animais chamados plânctons.

Conforme Ceccarelli (2000), é difícil determinar uma transparência ideal para a criação de peixes, mas geralmente está entre 40 e 60 cm. Viveiros muito transparentes, acima de 60 cm, permitem desenvolvimento de macrófitas aquáticas, que competem com oxigênio dissolvido e alimento para plâncton, atrapalham na despesca. Transparência menor que 30 cm pode indicar excesso de matéria orgânica, que diminui os níveis de oxigênio dissolvido na água. Indica também excesso de partículas de argila, que podem obstruir as brânquias dos peixes, causar lesões e, consequentemente, possibilitar a manifestação de doenças.

2.2.2.6 PH

Segundo Ceccarelli (2000), de acordo com as substâncias que se combinam em suspenção num meio, as reações podem resultar em elementos que tornam o meio ácido, neutro ou básico (alcalino). Se como resultado dessas reações houver excesso de íon hidrogênio (H^+), o meio será ácido. Se o excesso for de íon hidroxila (OH^-), será básico. Se houver equivalência entre esses íons, o meio será neutro.

O pH é uma medida que indica o logaritmo negativo da concentração de íons de hidrogênio. Ele é expresso em uma escala arbitrária que varia de 0 a 14, sendo considerado o meio ácido em pH menor que 7,0, básico ou alcalino se superior a 7,0 e neutro se igual a 7,0. Os peixes vivem, geralmente, em pH na faixa de 5,0 a 9,5, mas o melhor para a piscicultura tropical é pH na faixa de 7,0 a 8,0 (ou seja, neutro ou ligeiramente alcalino).

As tilápias apresentam baixa sobrevivência quando mantidas em águas com pH abaixo de 4,0. Segundo Kubitza (2011), na criação de tilápias, o pH da água deve ser mantido preferencialmente entre 6,0 e 8,5. Abaixo de 5,0 e acima de 11 a mortalidade pode ser elevada.

2.2.2.7 Registro dos dados de qualidade da água

Uma importante tarefa no controle da água para a aquicultura é o arquivamento dos registros de qualidade de água, pois de acordo com Ceccarelli (2000), por menor que seja, toda piscicultura deve ter um registro específico da qualidade de água, assim como da produção e de tudo que acontecer com cada viveiro de criação pois cada viveiro é um ecossistema diferente do outro. Portanto, cada um responde diferentemente à adubação, ao preparo dos reprodutores, à produção de larvas, ao crescimento na engorda, etc.

Quem trabalha há muitos anos numa piscicultura consegue prever a evolução dos parâmetros zootécnicos, o surgimento de enfermidades e os índices de qualidade em função da coloração da água, comportamento dos peixes, e outras, contudo se a pessoa experiente deixar a piscicultura um substituto novato terá muito trabalho para adquirir tal experiência, trabalho esse que é muito bem mitigado caso haja relatórios detalhados a respeito dos acontecimentos na aquicultura. Com a análise de tais relatórios consegue-se conhecer a história de cada açude, seu potencial de resposta e, então, prever seu desempenho frente a situações novas.

Neste projeto todos os parâmetros da água são armazenados em uma base de dados conectada a uma aplicação que prove uma interface amigável com páginas web contendo tabelas com dados temporais das leituras realizadas por meio dos sensores e com os alertas gravados para sensores e atuadores. Tais informações servem para o acompanhamento de todo o histórico dos açudes.

2.2.3 Itens de automação da aquicultura

De acordo com Simbeye (2014), alguns dos itens presentes na aquicultura que podem ser automatizados são: alimentador, recirculação da água: abrindo a saída e acionando o abastecimento, aerador, controle de temperatura da água e o controle do nível da água que segundo Parra et al. (2017), pode ser medido por um sensor de distância a exemplo do GP2Y0A02YK0F, que é desenvolvido pela empresa SHARP, entre outros. O GP2Y0A02YK0F é composto por uma

combinação integrada de um detector sensitivo de posição, de um emissor infravermelho para diodo e de um circuito de processamento de sinais.

Os sensores a serem usados na aquicultura podem ser de tecnologias sem fio que conforme Pule (2017), ganharam popularidade dentro da comunidade de pesquisa, porque fornecem uma infraestrutura promissora para inúmeras aplicações de controle e monitoramento. Essas redes simples de baixo custo permitem que os processos de monitoramento sejam realizados remotamente, em tempo real e com um mínimo de intervenção humana.

Além das tecnologias de redes sem fio existentes a um bom tempo no mercado, outras novas já foram usadas com sucesso na aquicultura como, por exemplo, Espinosa-Faller (2012), que utilizou o protocolo ZigBee e afirma que a capacidade *multi-hop* de uma rede ZigBee, fornece um método para ampliar o alcance e aumentar a confiabilidade nas comunicações com fornecimento de baixo custo e tecnologia de fácil implantação e monitoramento podendo ser usada em aquicultura com alta densidade de peixes.

É possível também utilizar sensores e outros mecanismos de controles para auxiliar a tomada de decisão baseada no comportamento dos peixes, que segundo Lloret et al. (2018), se o peixe está estressado, o consumo de alimento, por exemplo, cai e o desempenho diminui. Muitos fatores, quando inadequados, podem causar estresse nos peixes tais como: Temperatura da água, turbidez, oxigênio dissolvido, etc. Existem muitos parâmetros que devem ser monitorados para ajudar os peixes a melhorarem seu desempenho, portanto, melhorar a sustentabilidade e a lucratividade do mercado das fazendas de peixes.

2.2.4 Tecnologias para implementar o conceito de IoT

Neste tópico a automação dos processos da produção rural será apresentada com exemplos de tecnologias que podem ser aplicadas para implementar os conceitos de IoT.

As dificuldades atuais para realizar a automação são, por exemplo: alta complexidade para quem não possui habilidade na área de tecnologia da informação e automação, custo da mão de obra qualificada e número reduzido de

materiais relacionados ao tema.

Na automação do controle de um determinado processo sensores e atuadores são geralmente utilizados para atender às necessidades de monitoramento e atuação nos processos. Os sensores são usados, basicamente, para coletar os dados de um processo e os atuadores servem como acionadores quando desejamos executar algum procedimento.

De acordo com Gunasekera (2018), que construiu uma infraestrutura de IoT para a produção agrícola o propósito principal da IoT para o contexto da área rural é atuar como um hub que vincula dispositivos de sensores heterogêneos e dados deles a várias aplicações. Como tal, deve: poder receber dados de sensores heterogêneos e armazená-los em um formato flexível, fornecer mecanismos para criar aplicativos que utilizem dados em tempo real e históricos, ser robusto e escalável à medida que aumenta o número de dispositivos, o volume de dados e o uso de aplicativos, ser implantável no local em vez de ser um serviço baseado em nuvem, fornecer recursos de gerenciamento de dispositivos de sensor ou seja, manter um registro de sensores e ter a capacidade de enviar comandos para dispositivos e ter o apoio de uma comunidade ou fornecedor ativo.

Para implementar o projeto de IoT proposto neste projeto foi utilizada a plataforma arduino, contudo este, naturalmente, não é a única tecnologia disponível no mercado que pode ser utilizada para concretizar a internet das coisas pois existem várias tecnologias à disposição para este trabalho e algumas delas são por exemplo: Raspberry Pi, BeagleBone, ESP8266, Photon, Intel Edison e LoRa, portanto, antes de estudar o arduino com maiores detalhes segue uma breve apresentação de cada uma.

O Raspberry Pi é um computador em uma única placa bem pequena que executa o sistema operacional Linux tendo nela portas USB e saída de vídeo HDMI, logo é possível conectar teclado, mouse, monitor e ter assim um computador completo. O Rapberry Pi segundo Monk (2016), tem ainda a vantagem de ser muito sofisticado e consumir pouca energia logo é muito mais adequado do que um notebook ou outro computador quando aplicado a cenários nos quais a energia pode acabar e a continuidade dos serviços é necessária, como é o caso de muitos sistemas de controle da IoT.

> Raspberry Pi é um computador de placa única usado para executar operações de computação e de redes remotas. Este computador é um dos componentes essenciais da Internet das Coisas (IoT), permitindo que o modelo acesse a internet e, portanto, contribuindo para a automação do processo. (DANKHARA, 2019, p. 700).

No caso do BeagleBone este apresenta as mesmas características listadas para o Raspberry Pi sendo que a principal diferença é com relação ao preço que no caso do BeagleBone este é mais caro. Tanto o Raspberry Pi quanto o BeagleBone possuem recursos bem acima do que é necessário para desenvolver um projeto básico de IoT sendo assim ambos atendem tranquilamente as demandas da IoT.

O componente ESP8266 produzido pela fabricante Chinesa Esppressif conforme Oliveira (2017) é um microcontrolador de 32 bits que inclui um núcleo microprocessado Tensilica L106, que funciona na frequência-padrão de 80 MHz, podendo chegar a 160 MHz. O processamento da pilha de protocolos WiFi utiliza 20% da capacidade total de processamento deste microcontrolador, portanto 80% da capacidade do processador do mesmo pode ser usado em aplicações do usuário.

O Photon também é um computador em uma placa, contudo em tamanho muito mais reduzido do que os já citados mesmo assim fornece tudo o que é necessário para desenvolver um projeto conectado de IoT. O photon apesar de possuir portas USB, por padrão, não possui saída de vídeo como alguns apresentados.

A Intel Edison é uma pequena placa baseada na plataforma Linux que foi especialmente projetada para ser embutida em projetos de IoT e é o competidor mais próximo do Photon apesar de seu preço ser mais elevado do que o do Photon.

LoRa é uma tecnologia de rádio frequência que permite a comunicação a longas distâncias. Uma rede LoRaWAN usa rádio frequência para transmitir dados de uma forma otimizada em distâncias que podem ser superiores a 15 quilômetro entre os pontos conectados. A distância de transmissão pode ser reduzida em ambientes urbanos, para o cenário rural ela é mais recomendada tendo em vista a ausência de banda larga em muitos casos e a lenta transição para a tecnologia 5G que em muitos países ainda nem está em operação.

Segundo Sinha (2017), LoRa opera em uma banda não licenciada abaixo de 1 GHz para operação de link de comunicação de longo alcance. LoRa é um

esquema de modulação de espectro de propagação proprietário que é derivado do chirp de modulação de espectro de dispersão (CSS) e que negocia a taxa de dados para sensibilidade dentro de uma largura de banda de canal fixo. CSS, que foi desenvolvido na década de 1940, era tradicionalmente usado em aplicações por causa de suas longas distâncias de comunicação e robustez contra interferência. LoRa é sua primeira implementação de baixo custo para uso comercial. O nome LoRa deriva da vantagem da capacidade de longo alcance do inglês *long-range* que se beneficia da grande provisão de link fornecido pelo espectro de expansão do esquema de modulação.

Uma grande vantagem da LoRaWAN é a duração da bateria que pode ser de até 10 anos. Devido a estas e a outras características esta tecnologia é também uma boa opção para a implementação da IoT no setor rural. Diferente das outras tecnologias apresentadas a LoRa não é um microcontrolador mas pode ser implementada usado o Raspberry Pi ou algum *gateway* compatível. Conforme Queralta (2019), baixa potência e comunicação de longo alcance têm aplicações evidentes em áreas rurais onde a cobertura de celular é ruim e a implantação de infraestrutura de redes Wi-Fi ou similares pode ser cara devido a terrenos difíceis ou da dispersão dos nós de sensores em grandes áreas, o autor afirma ainda que para o cenário agrícola a LoRaWAN e SigFox são as duas soluções mais populares do momento.

O arduino é a tecnologia mais utilizada atualmente como plataforma para a implementação da IoT e é também a mais difundida entre os estudantes. A tecnologia arduino conforme Monk (2014) é uma pequena placa de microcontrolador que contém uma conexão USB, tornando possível a ligação com um computador. Além disso, esta contém diversos terminais os quais permitem a conexão com dispositivos externos, como motores, relés, sensores luminosos, diodos a laser entre outro. Na Figura 1 uma placa arduino no modelo uno, o mais difundido entre os estudantes, e uma modelo mega 2560 podem ser observadas.

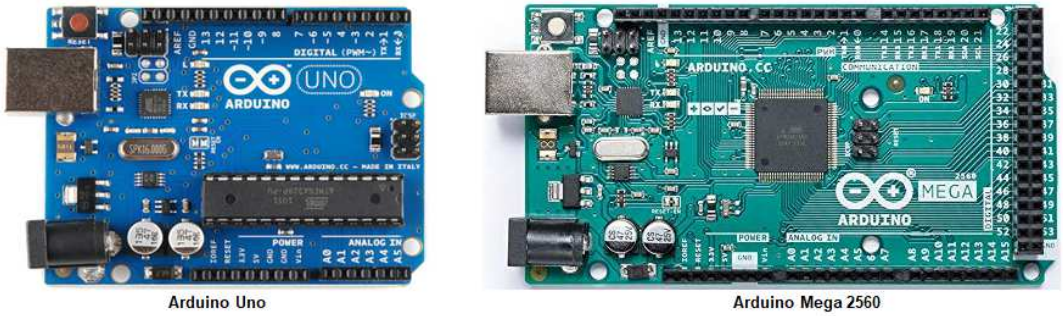


Figura 1 - Placas arduino modelos Uno e Mega 2560
Fonte: Arduino (2019).

Apesar de ser configurada por meio de um computador a placa arduino e as demais placas já apresentadas possuem a capacidade de trabalharem de forma autônoma. O projeto da placa arduino é aberto, ou seja, qualquer pessoa poderá construir placas compatíveis com o mesmo se assim desejar. As placas básicas do arduino são complementadas por outras placas acessórias (*Shields*), as quais podem ser encaixadas sobre as placas básicas, conectadas por meio de cabos ou até mesmo usando uma protoboard. Várias IDEs para configuração e programação do arduino são livres e estão disponíveis, gratuitamente, para download na internet, existe versões para: Windows, Linux e Mac OS. Para implementar a IoT utilizando o arduino ou outro microcontrolador uma shield de ethernet atende as demandas de conexão. Alguns microcontroladores já possuem os componentes de conexão de forma integrada (*onboard*).

3 METODOLOGIA

Neste capítulo os procedimentos gerais da pesquisa são descritos e as técnicas utilizadas para elaboração do gateway e do sistema web de monitoramento são listadas.

3.1 LOCAL DA PESQUISA

A pesquisa foi realizada no Instituto Federal do Paraná (IFPR) – Campus Foz do Iguaçu e na Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Medianeira. A maior parte da pesquisa bibliográfica foi realizada na UTFPR e os testes práticos foram realizados no IFPR tendo em vista que no campus Foz do Iguaçu tem viveiros e tanques que são usados para atender os cursos de aquicultura.

3.2 FINALIDADE DA PESQUISA

A pesquisa teve como finalidade coletar e salvar as informações referentes à aquicultura visando ter um registro histórico dos eventos gerados nos viveiros a fim de conhecer seus comportamentos e com isso ter a capacidade de prever eventos futuros e interferir com ações que otimizem os processos com intuito de melhorar os resultados.

3.3 TIPO DE PESQUISA

Para a elaboração deste trabalho foi feita uma revisão bibliográfica sobre: Internet das Coisas, automação, eletrônica, arduino e algoritmos para monitoramento de dados, tendo como objetivos: avaliar conceitos, coletar melhores práticas, mensurar os resultados apresentados nas bibliografias e apresentar os testes realizados em campo.

3.4 MÉTODOS DE PESQUISA

Foram utilizados métodos quantitativos de instrumentação, por meio de sensores para coleta dos dados da qualidade da água da aquicultura sendo eles: pH, turbidez, nível, temperatura e vazão, e dados do monitoramento interno do *gateway* desenvolvido que foram temperatura e umidade internas.

Na pesquisa houve também métodos de experimento, com a aplicação dos componentes implementados em campo, local onde os tanques de peixes foram instalados, os quais tiveram os comportamentos analisados a fim de mensurar precisão do *gateway* na coleta de dados, eficiência e resistência dos sensores e o desempenho do *gateway* e da aplicação web.

Logo após foram realizados experimentos em laboratório, feito isto, os experimentos foram implantados em campo para testes práticos.

Depois foi implementado um sistema web de análise dos dados coletados dos sensores o qual permite o gerenciamento remoto de todos os equipamentos implementados e instalados e o monitoramento em tempo real das coletas de dados e dos eventos ocorridos em campo.

3.5 PROJETO DO *GATEWAY*

Em outra etapa o *gateway* de IoT foi confeccionado utilizando tecnologias de preço acessível a fim de ampliar a possibilidade de produtores de baixo poder aquisitivo poderem utilizar tecnologias que ajudem a agregar valor aos seus produtos.

Para o design de um projeto de automação é necessário modelar e desenhar os processos e realizar a analise dos dados e feito isto se implementa então o algoritmo de fusão de dados com modificação de mais de um parâmetro, que de acordo com Khaire; Wahul (2018), é usado para a otimização de dados.

Na Figura 2 é apresentado o diagrama de bloco das tecnologias envolvidas na automatização da aquicultura proposta neste trabalho.

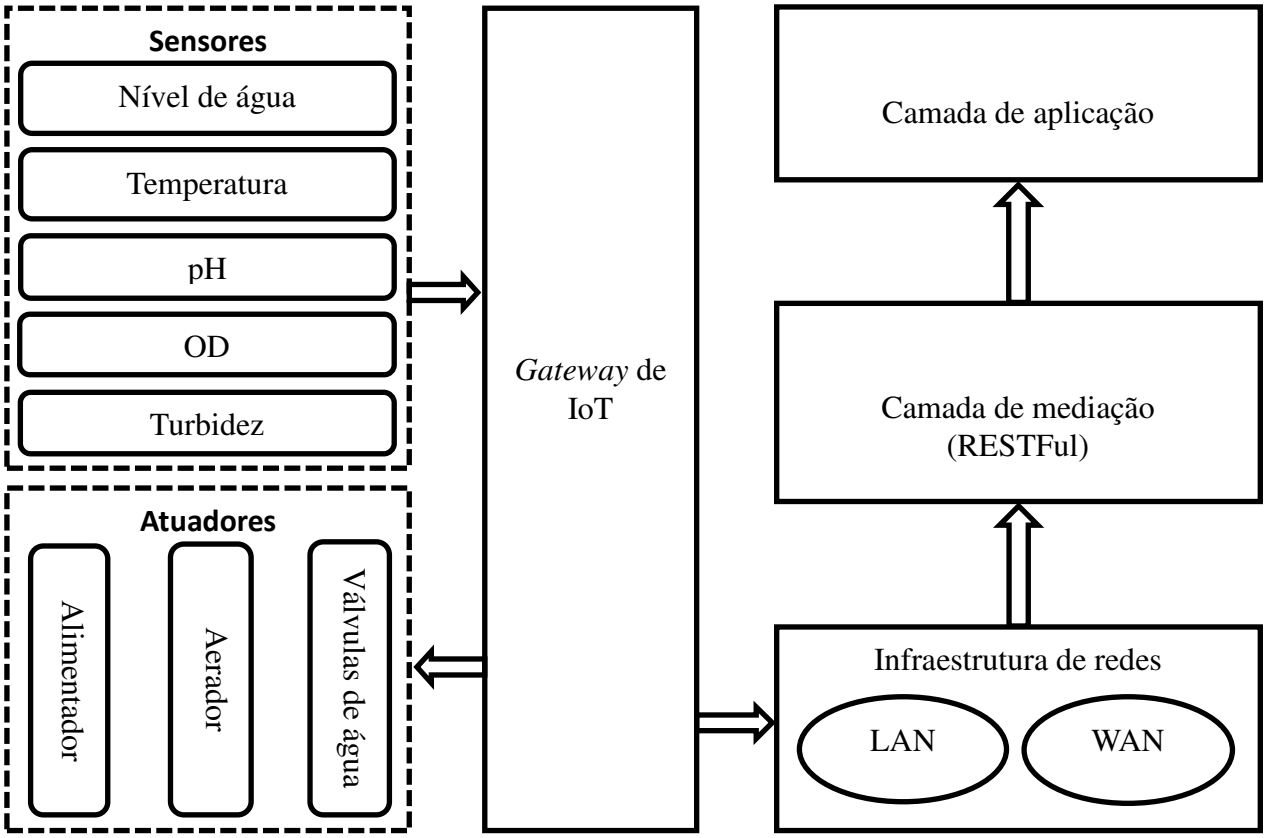


Figura 2: Diagrama de bloco das tecnologias envolvidas na automatização da aquicultura
Fonte: Traduzido e adaptado de Moses et al (2018 p. 32) e Bokingkito et al. (2017 p. 700).

Como observado na Figura 2 o *gateway* é o mediador entre a camada física e a lógica sendo um intermediário entre os componentes eletrônicos e softwares ao realizar uma automação baseada no conceito de IoT.

Quando se trata do complexo da aquicultura a simples automatização não é o suficiente para garantir a eficiência do monitoramento, é necessário que haja sistemas de redundância em itens mais críticos tendo em vista que de acordo com Xu et al. (2017), a aquicultura possui falhas e estas são complexas e cada parte em cada camada do sistema geral pode falhar. Devido ao ambiente hostil, o equipamento externo pode falhar na camada de aquisição de dados ou na camada de comunicação. Os sensores podem ser corroídos pela poluição e por micro-organismos.

As comunicações sem fio também podem ser interrompidas ou falharem facilmente devido às variáveis do ambiente ou até mesmo por erros humanos no complexo ambiente de aplicativos. Na camada de armazenamento e na camada de aplicativo, podem ocorrer falhas como as de software.

Falhas na fonte de alimentação por consequência de diferentes fontes de energia, como baterias, energia principal ou energia do painel fotovoltaico podem ocorrer. Quando essas falhas ocorrem, isso pode direcionar a decisões erradas como, desperdício de recursos e até ameaçar a segurança de produtos aquáticos, o que resultaria em perdas significativas de recursos econômicos e humanos.

Para automatizar a aquicultura foi proposta uma arquitetura tecnológica organizada em camadas conforme apresentado na Figura 3.

Figura 3: Representação visual da arquitetura de IoT proposta
Fonte: Traduzido e adaptado de Huan et al. (2018 p. 258).

O modelo proposto na Figura 3 é uma adaptação dos modelos propostos por Huan et al. (2018) e Bokingkito et al. (2017), e funciona da seguinte forma, a camada de sensores é a de mais baixo nível na qual a coleta dos dados é realizada, feito isto os mesmos são enviados para a camada de interface que é onde está o *gateway* de IoT o envio de dados pode ser feito por meio de cabeamento ou até mesmo via *wireless* que segundo Parra et al. (2017), tem a vantagem do rápido processamento na aquisição de dados, sendo que a camada de redes é responsável pelo desempenho das funções básicas de transmitir dados e informações para as aplicações por meio da rede. A camada mediadora permite a interoperabilidade entre aplicações heterogêneas, e por último a camada de aplicação é a que provê meios para interações com os usuários.

Em alguns modelos arquiteturais como, por exemplo, o apresentado por Jiang (2018), o nível *gateway* cria automaticamente a camada de redes sem fio e a administra por padrão ou por configuração manual. O nó de *gateway* não é responsável apenas por aceitar os dados de nós dos sensores, mas também por transmiti-los para o computador de monitoramento central a fim de realizar processamento adicional.

Já no modelo, aqui proposto, o *gateway* é representado pela camada de interface e é responsável por encapsular a complexidade existente na tarefa de interconectar a camada de sensores com a camada de redes, feito isto a camada de sensores fica responsável por coletar os dados e a camada de redes por realizar a transmissão dos mesmos usando a infraestrutura de redes.

Os autores Huan et al. (2018), elaboraram um diagrama de bloco no qual é possível ter uma visão arquitetural do sistema por eles proposto neste os componentes do sistema são organizados em quatro níveis de camada.

O autor Bokingkito (2017), apresenta em seu artigo um modelo arquitetural que está organizado nos seguintes níveis: camada de sensores, camada de redes, camada mediadora e camada de aplicação.

O *gateway* de IoT proposto neste trabalho pode ser usado também em experimentos para identificar a viabilidade de implantação de cultivo em áreas costeiras tendo em vista que de acordo com Schmidt (2018), a necessidade de garantir a segurança alimentar no futuro e as questões da variação da qualidade da água estuarina estará impulsionando a expansão da aquicultura em águas costeiras.

Para realização dos testes em campo, protótipos dos sistemas foram implementados e componentes foram confeccionados. O sistema para o servidor de aplicação foi implementado em linguagem de programação Java seguindo a especificação Java Enterprise Edition (JEE) e as interfaces para acesso via navegador foram implementadas em HTML e XHTML juntamente com os *frameworks* Java Server Faces (JSF) e Primefaces, a suíte de componentes eletrônicos parte da solução para o *gateway* foi confeccionada sobre a plataforma arduino. O nível de encapsulamento da complexidade foi mensurado considerando a usabilidade e facilidade de instalação proporcionada aos usuários.

Os equipamentos que foram usados para possibilitar o uso da IoT possuem Protocolo da Internet (IP), a fim de facilitar o entendimento da IoT a figura 4 apresenta a relação de tecnologias que estão, por esta, abrangidas.

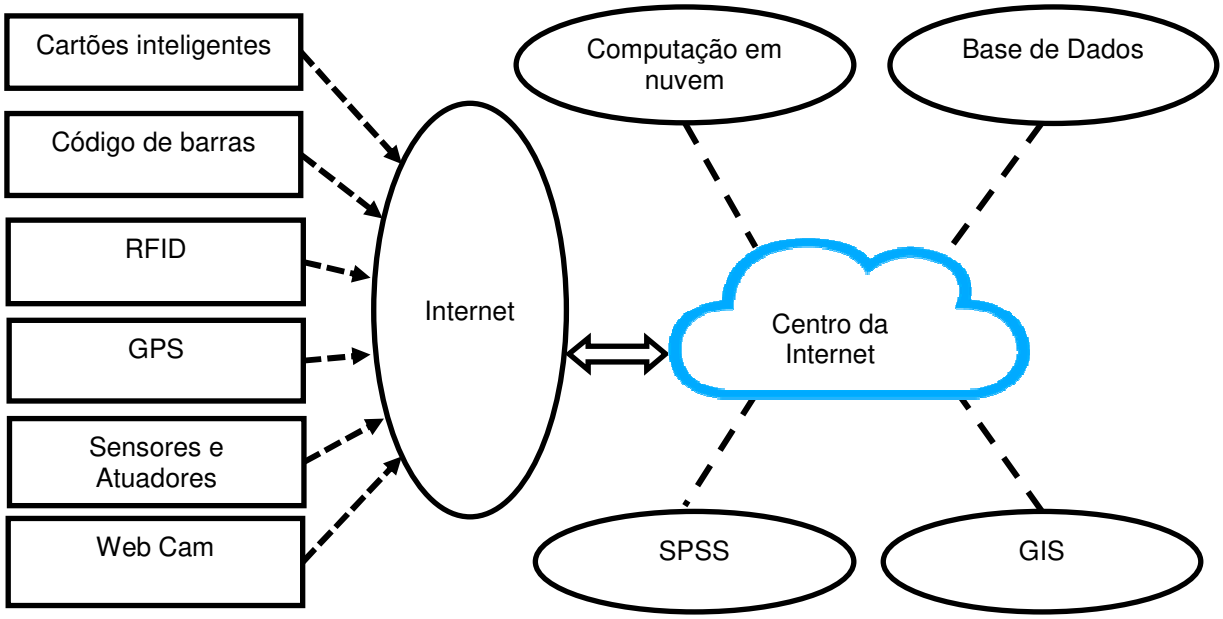


Figura 4: Sistema tecnológico de IoT
Fonte: Traduzido e adaptado de Ji et al. (2015 p. 1001).

Para implementar o *gateway* foi utilizada a plataforma *open-source* arduino e sensores compatíveis com a mesma plataforma foram utilizados para atender à camada 1 do modelo proposto na Figura 3. Uma placa de arduino Mega 2560 foi utilizada para atender à camada 2. Para a camada 3 foi usada uma placa Ethernet shield W5100, modelo compatível com arduino. Para a camada 4 foi implementado um servidor de aplicação baseado na especificação JEE.

A especificação JEE é um conjunto de boas práticas sinergicamente relacionadas no intuito de prover facilidades aos profissionais de software ao projetar aplicações corporativas com suporte a uma variedade de clientes o que contribui com a reação positiva e com a flexibilidade.

A JEE na sétima versão JEE7 é compatível com a computação em nuvem, do inglês *cloud computing*, que no contexto da IoT trás a vantagem de facilitar o provimento de dados para diversos dispositivos por meio da internet e maior confiabilidade no que diz respeito ao armazenamento dos dados. No projeto arquitetural, elaborado neste trabalho a comunicação é viabilizada por meio da disponibilização de serviços que foi implementada utilizando *web services* conforme o padrão REST que é um modelo a ser utilizado para se projetar arquiteturas de software distribuído, baseadas em comunicação via rede.

Por último a camada 5 foi atendida com a implementação de uma aplicação

web, que foi também desenvolvida seguindo a especificação JEE7 com os frameworks JSF na versão 2.0, Primefaces, JQUERY, Spring Security, Hibernate, Java Persistence API (JPA) e a linguagem de programação Java na versão 7 juntamente com a tecnologia Enterprise Java Bean (EJB) na versão 3.0.

A linguagem Java foi selecionada tendo em vista as suas características positivas tais como: multiplataforma, portabilidade, escalabilidade e conjunto de bibliotecas e *frameworks* disponíveis. Dentre as características apresentadas a mais relevante para este projeto é a capacidade de ser escalável haja vista o atendimento da intenção futura de implementar um ambiente em nuvem a fim de disponibilizar um repositório no qual os usuários do *gateway* possam implantar os seus servidores de aplicação e armazenar de forma segura os dados coletados tornando assim a atividade de automatização ainda menos onerosa haja vista que o usuário não terá a necessidade de adquirir um servidor local e não gastará com consumo de energia.

O registro dos experimentos e testes realizados em laboratório e em campo foi realizado em vídeos e fotos.

4 RESULTADOS E DISCUSSÕES

Neste capítulo os resultados obtidos a partir do desenvolvimento desta pesquisa são apresentados e detalhados e algumas discussões a respeito das experiências vivenciadas no decorrer dos estudos e trabalhos são apresentadas.

4.1 IMPLEMENTAÇÃO DA APLICAÇÃO WEB

Na primeira etapa a aplicação do *gateway*, sistema de monitoramento da aquicultura (SISMAQUI), foi implementada afim de facilitar a gestão, controle e operação dos componentes instalados em campo, local onde fica os dois tanques de peixes, a Figura 5 apresenta o diagrama de classes da SISMAQUI.

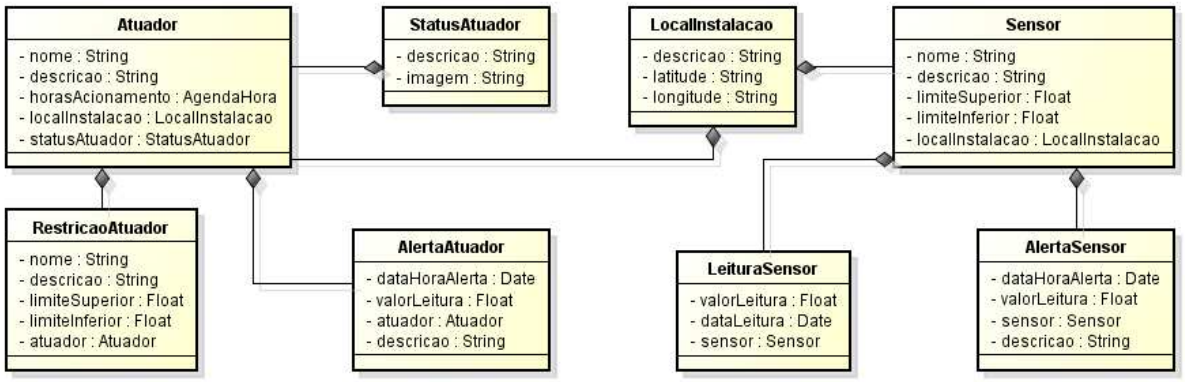


Figura 5: Diagrama de classes da aplicação SISMAQUI
Fonte: Elaborado pelos autores.

O diagrama de classes apresentado na Figura 5 refere-se ao documento que facilita o entendimento dos códigos implementados entre os analistas e desenvolvedores, nele é possível especificar os atributos correspondentes, em grande parte, aos dados que serão armazenados na base de dados, os relacionamentos e os comportamentos que são as funções de um sistema. O diagrama possibilita ver que para um sensor, os dados a serem armazenados são: nome, descrição, limite superior, limite inferior e local de instalação.

Na Figura 6 é possível visualizar a interface implementada para a tela *dashboard* da aplicação. Uma tela *dashboard* é uma interface que possibilita ter uma visão geral e resumida dos dados importantes em uma aplicação.

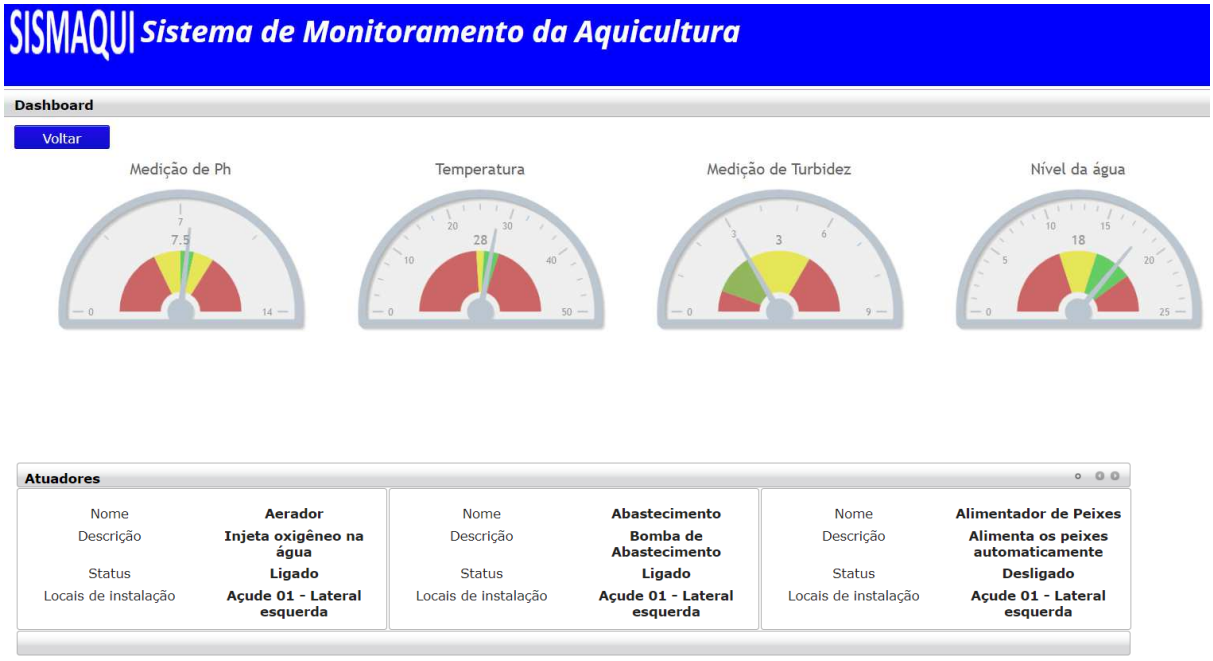


Figura 6: Tela *dashboard* da aplicação SISMAQUI

Fonte: Elaborado pelos autores.

No primeiro componente do tipo metergauge, o qual tem um ponteiro, permite monitorar os dados de pH da água, o segundo é referente à temperatura da água, o terceiro ao nível de turbidez da água e o último à medição do nível da água do tanque. Na tabela abaixo dos componentes metergauge os atuadores instalados são listados em uma datagrid, sendo eles: alimentador automatizado, aerador e circulador de água.

A Figura 7 mostra a interface implementada para monitoramento e avaliação das curvas de tendências temporais dos dados coletados para os sensores de: pH, temperatura, turbidez e nível da água.

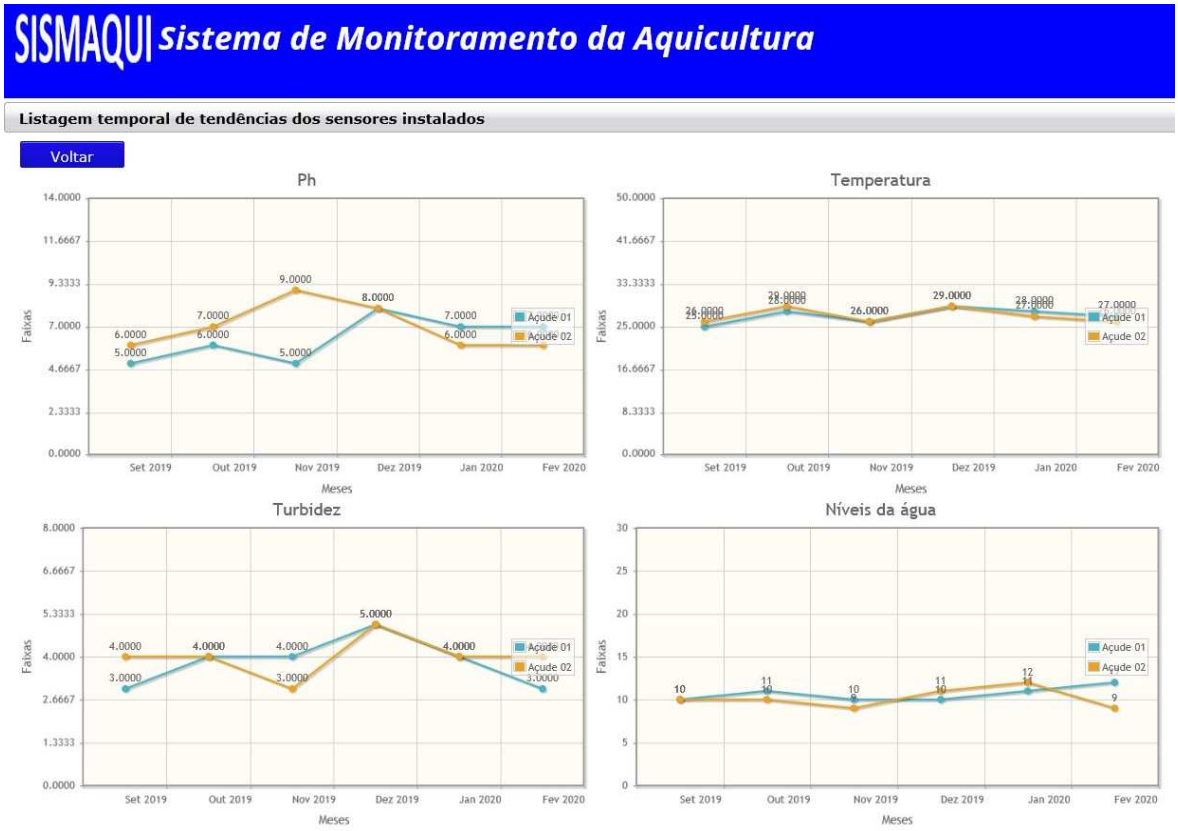


Figura 7: Tela visualização das tendências dos dados coletados

Fonte: Elaborado pelos autores.

Nesta aplicação é possível monitorar, remotamente, as operações de campo tais como: Alertas disparados para os sensores e atuadores, status dos sensores e atuadores, como por exemplo, no atuador alimentador de peixes automatizado seria possível verificar o nível do suprimento de ração e quando o nível está abaixo de 20% a aplicação envia notificação, automaticamente, via email aos operadores responsáveis pelo reabastecimento, para que seja resolvido o problema.

A aplicação está internacionalizada e traduzida para 5 idiomas: Português, inglês, espanhol, francês e italiano. E foi desenvolvida visando ser responsiva e portável e está homologada para os navegadores Google Chrome, Internet Explorer e Mozilla Firefox e sua responsividade foi testada nas resoluções de 1024 x 600 até 3840 x 2160.

Na aplicação o usuário é cadastrado, necessariamente, com um determinado perfil que pode ser, por exemplo: administrador, visualizador ou outro. Quando possuir o perfil de administrador o usuário tem um amplo controle sobre a aplicação e quando possuir perfil de visualizador tem disponível algumas funcionalidades tais como: Gerenciamento de sensores, atuadores, locais de

instalação, geração de relatórios em PDF das leituras dos sensores e dos alertas gerados entre outras. Maiores detalhes dos códigos desenvolvidos para a aplicação SISMAQUI estão no Anexo C, neste é possível visualizar as classes implementadas para a aplicação e ter noção da estrutura arquitetural da mesma. Nem todas as camadas e classes implementadas estão sendo utilizadas, todavia, foram codificadas tendo em vista que é uma boa prática de desenvolvimento projetar as arquiteturas prevendo necessidades futuras de expansão da aplicação.

4.2 CONFECÇÃO E IMPLEMENTAÇÃO DO *GATEWAY* DE IOT

Na segunda etapa o *gateway* de IoT foi confeccionado e seus códigos embutidos foram implementados. A integração das tecnologias para conexão dos componentes com sistemas e aplicações foi feita conforme o diagrama de arquitetura conceitual apresentado na Figura 8.

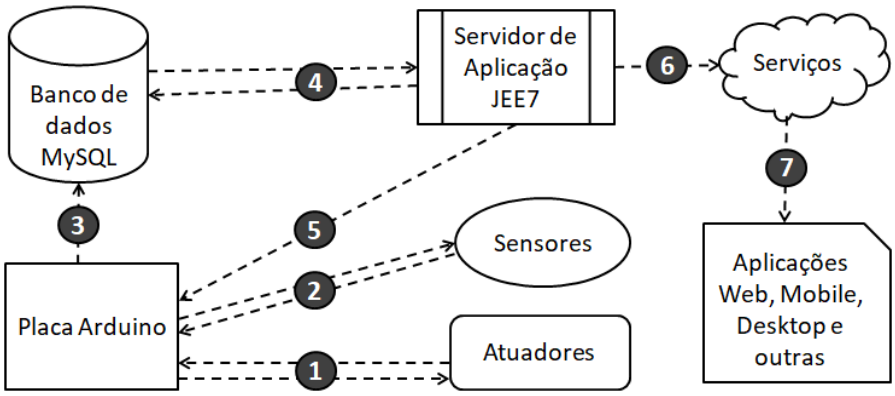


Figura 8: Diagrama de arquitetura conceitual
Fonte: Elaborado pelos autores.

Na Figura 8 é possível, também, verificar a sequência lógica da atividade de monitoramento, inicialmente, os dados são coletados pelos sensores e/ou atuadores e lidos pela plataforma arduino a qual envia os mesmos para o banco de dados MySQL o que os armazena, permanentemente, deixando-os disponíveis para a aplicação que os acessa via conexão HTTP, feito isto as informações são disponibilizadas como serviço. O servidor de aplicação também é responsável pelo processamento e notificações de alertas gerados pelo sistema.

Com intuito de resolver os incidentes o mais rápido possível foi implementado o módulo de notificação que informa, via email, aos responsáveis pela

aquicultura as ocorrências geradas em campo.

O módulo de notificação foi implementado utilizando a biblioteca EthernetClient.h para arduino as declarações necessárias para configurar o envio de email são exibidas na Figura 9.

```
250 #include <EthernetClient.h>
251 #define time 1000
252 EthernetClient emailclient;
253 byte emailserver[] = { 200, 147, 99, 132 }; //IP do servidor SMTP
254
255   void enviaemail(String msg, String valorexcedido)
256   {
257     delay(time);
258     Serial.println("conectando...");
259     if (emailclient.connect(emailserver, 587)) //Porta do servidor do email.
260     {
261     Serial.println("conectado!");
262     Serial.println("enviando email...");
263     Serial.println();
264     emailclient.println("EHLO localhost");
265     recebe();    delay(time);
266     emailclient.println("AUTH LOGIN");
267     recebe();    delay(time);
268     emailclient.println("loginembasede64"); // Email de login em base de 64
269     recebe();    delay(time);
270     emailclient.println("senhaembasede64"); // Senha do email em base de 64
271     recebe();    delay(time);
272     emailclient.println("mail from: <usuario@bol.com.br>"); //Remetente
273     recebe();    delay(time);
274     emailclient.println("rcpt to: <usuario@gmail.com>"); //Destinatário
275     recebe();    delay(time);
276     emailclient.println("data");
277     recebe();    delay(time);
278     emailclient.println("Subject: Monitoramento da Aquicultura"); // Assunto
279     recebe();    delay(time);
280     emailclient.println(msg+", valor excedido: "+valorexcedido); // Corpo
281     recebe();    delay(time);
282     emailclient.println("."); // Indica fim do email.
```

Figura 9: Configurações para envio de email

Fonte: Elaborado pelos autores.

Na Figura 10 o código que verifica as exceções dos limites superiores e inferiores para envio de notificação é apresentado.

```
787     if (temp[i]>30){ //Se a temperatura for maior que 30°C liga
788      ligaBombaRecirculacao();
789        //Envia email de alerta
790        enviaemail("Temperatura acima do recomendado",tempaguaconvertida);
791        //Salva alerta
792          sprintf(sentenca, INSERIR_ALERTA_SENSOR, "Temperatura acima do recomendado",
793          "Sensor de Temperatura",tempaguaconvertida);
794          Serial.println(sentenca);
795          cur_mem->execute(sentenca);
796     }else if (temp[i]<24){ //Se a temperatura for menor que 24°C liga
797        ligaBombaRecirculacao();
798          //Envia email de alerta
799          enviaemail("Temperatura abaixo do recomendado",tempaguaconvertida);
800          //Salva alerta na base de dados
801          sprintf(sentenca, INSERIR_ALERTA_SENSOR, "Temperatura abaixo do recomendado",
802          "Sensor de Temperatura",tempaguaconvertida);
803          Serial.println(sentenca);
804          cur_mem->execute(sentenca);
805     }else{
806      desligaBombaRecirculacao(); //Caso contrário desliga
```

Figura 10: Verificação de limites excedidos para envio automático de notificação

Fonte: Elaborado pelos autores.

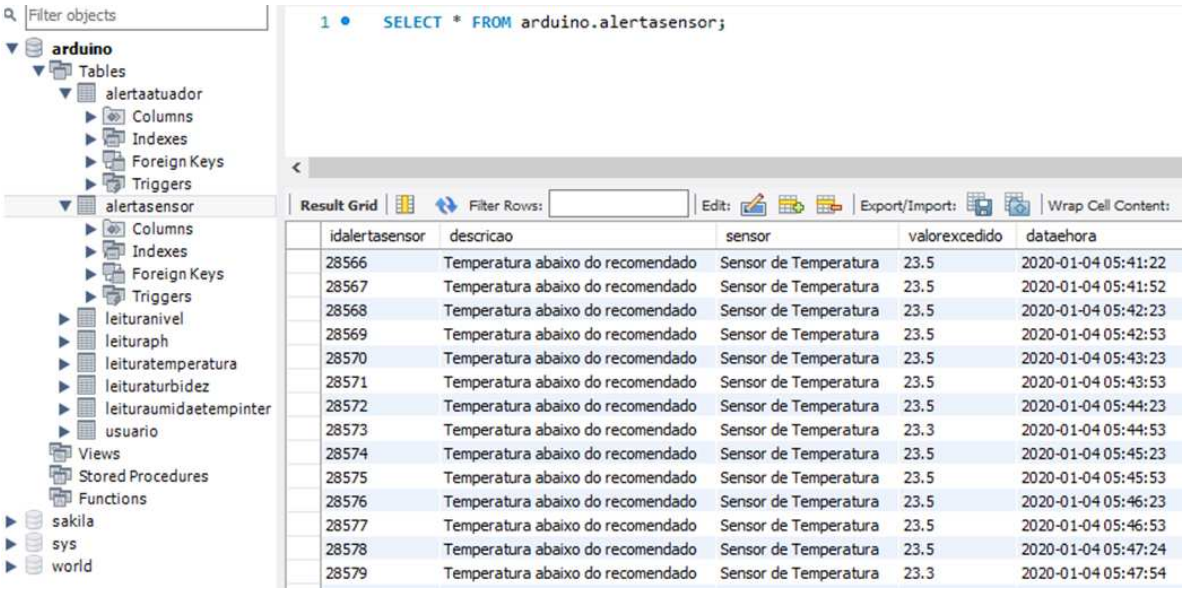
Na linha 787 é verificado se a temperatura da água está acima de 30º C se sim a função de envio de email é chamada com os parâmetros correspondentes à temperatura que está acima do recomendado conforme a linha 790, já na linha 796 é verificado se a temperatura da água está abaixo de 24º C se sim a função de envio de email é chamada com os parâmetros correspondentes à temperatura abaixo do recomendado conforme a linha 799. Os limites 24º C e 30º C são os definidos para a maioria das espécies de peixes tropicais brasileiros. Na Figura 11 é possível observar um exemplo de notificação via email que fora gerado pelo próprio SISMAQUI.

Figura 11: Notificações recebidas via email

Fonte: Elaborado pelos autores.

Além de enviar por email o sistema também salva as exceções na base de

dados a fim de garantir a rastreabilidade e futuras análises dos dados coletados. A Figura 12 apresenta as exceções do limite inferior configurado para a temperatura da água salvo na base de dados.



idalertasensor	descricao	sensor	valorexcedido	dataehora
28566	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:41:22
28567	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:41:52
28568	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:42:23
28569	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:42:53
28570	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:43:23
28571	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:43:53
28572	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:44:23
28573	Temperatura abaixo do recomendado	Sensor de Temperatura	23.3	2020-01-04 05:44:53
28574	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:45:23
28575	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:45:53
28576	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:46:23
28577	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:46:53
28578	Temperatura abaixo do recomendado	Sensor de Temperatura	23.5	2020-01-04 05:47:24
28579	Temperatura abaixo do recomendado	Sensor de Temperatura	23.3	2020-01-04 05:47:54

Figura 12: Exceções ao limite inferior da temperatura da água salvas na base de dados
Fonte: Elaborado pelos autores.

Conforme observação na Figura 12, para cada alerta gerado é salvo: uma descrição que informa o que ocorreu, o sensor correspondente, o valor limite excedido que pode ser inferior ou superior e a data e hora em que a exceção verificada gerou o alerta.

A visão da distribuição e interconexão dos componentes pode ser visualizada no diagrama de componentes conforme a Figura 13.

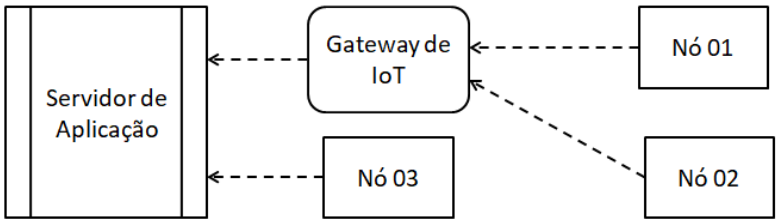


Figura 13: Diagrama de componentes
Fonte: Elaborado pelos autores.

No projeto do componente *gateway* foram definidos os sensores que seriam compatíveis com o mesmo sendo que não há garantia de pleno funcionamento para sensores diferentes destes com o componente implementado.

Para medir o nível da água foi utilizado o sensor ultrassônico de medição de distância HC-SR04. Neste sensor o transmissor emite oito rajadas de onda ultrassônica de 40khz direcional quando acionado e inicia um temporizador. Pulsos ultrassônicos viajam para fora até encontrarem um objeto, o objeto faz com que a onda seja refletida de volta para a unidade.

O receptor ultrassônico detecta a onda refletida e interrompe o cronômetro de parada. A velocidade da explosão ultrassônica é de 340m/segundo no ar. Baseado no número de contagem do temporizador, a distância pode ser calculada entre o objeto e transmissor, a fórmula de mensuração TRD é expressa como: D = C X T, que é conhecido como a fórmula de medição de tempo / taxa / distância, onde D é a distância mensurada, e R é a velocidade de propagação (Taxa) no ar (velocidade do som) e T representa o tempo. Nesta aplicação, T é dividido por 2 como T é o dobro do tempo do valor do transmissor para o objeto de volta ao receptor.

No caso da medição do pH foi utilizado o sensor SEN161 que é especialmente projetado para controladores arduino sendo de conexão prática e outros recursos. Utiliza um conector BNC plugado na interface PH2.0 a qualquer entrada analógica do controlador arduino. É necessário utilizar água pura para limpar o sensor antes e depois do uso e para preservá-lo deve ser mantido na solução 3N KCL, o mesmo deve ser frequentemente calibrado com a solução padrão no intuito de manter a precisão na leitura dos dados.

Já para a medição de temperatura e umidade interna do *gateway* foi utilizado o sensor DTH22 que utiliza técnica exclusiva de coleta de sinal digital e umidade e tecnologia de detecção, assegurando sua confiabilidade e estabilidade. Seus sensores são conectados a um chip de computador com 8 bits.

Para medir a temperatura da água foi utilizado o sensor do tipo termômetro digital DS18B20 o qual fornece medições de temperatura em Celsius de 9 bits a 12 bits e possui função de alarme com pontos de gatilho superior e inferior programáveis pelo usuário e não voláteis. O DS18B20 se comunica através de um barramento de 1 fio que, por definição, requer apenas uma linha de dados, e terra, para comunicação com um microprocessador central. Possui uma faixa de temperatura operacional de -55º C a + 125º C e é preciso para ± 0,5º C na faixa de -10º C a + 85º C. Além disso, o DS18B20 pode derivar energia diretamente da linha de dados (“*parasite power*”), eliminando a necessidade de uma fonte de alimentação

externa.

Para a medição da turbidez foi usado o sensor SEN0189 que funciona da seguinte forma, utiliza luz para detectar partículas suspensas em água, medindo a transmitância da luz e taxa de dispersão, que muda com a quantidade de totais de sólidos em suspensão (TSS) na água. À medida que o TSS aumenta, o líquido aumenta o nível de turbidez.

E por último para a medição da vazão da água da recirculação foi utilizado o sensor YF-S201 que é um sensor de fluxo de líquido de 1/2", que contém internamente um sensor de cata-vento para medir quanto líquido foi movido por ela. Quando a água passa pelo rotor, são gerados pulsos proporcionais a velocidade do rotor. Há um sensor de efeito hall magnético integrado que gera um pulso elétrico. O sensor de efeito hall é vedado a partir da tubulação de água permitindo assim que o sensor fique seco e seguro.

O YF-S201 vem com três fios: vermelho (Power 5-24VDC), Preto (Terra) e amarelo (Efeito de hall de saída de pulso). Ao contar os impulsos a partir da saída do sensor, pode-se facilmente calcular o fluxo de água. Cada pulso é de aproximadamente 2,25 mililitros, não é um sensor de precisão, e a taxa de pulso faz variar um pouco dependendo da taxa de fluxo, pressão de fluido e orientação do sensor. Ele precisa de uma calibração cuidadosa. O sinal de pulso é uma onda quadrada simples sendo assim fácil de registrar e converter em litros por minuto. A Figura 14 apresenta alguns dos sensores eletrônicos utilizados no projeto.

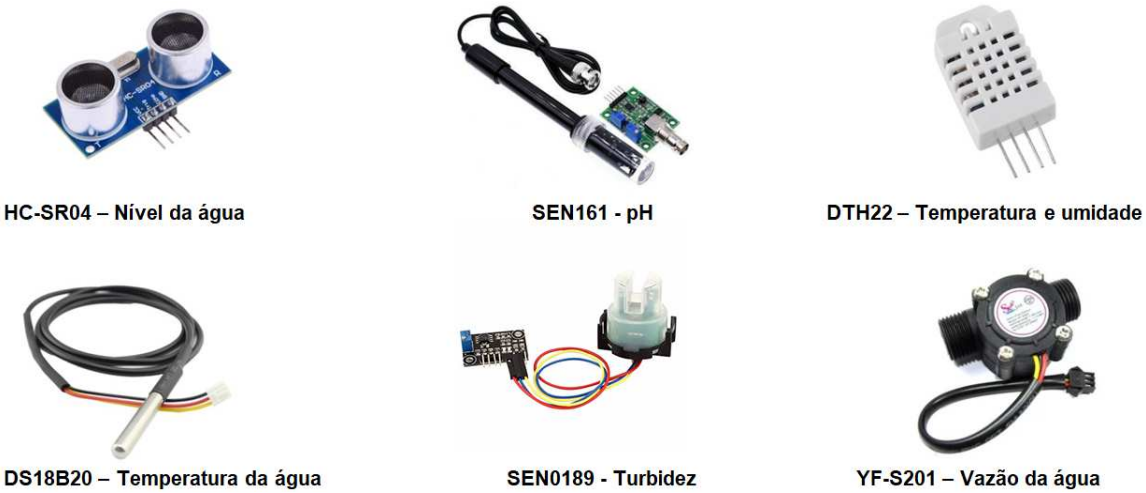


Figura 14: Sensores eletrônicos utilizados no projeto

Fonte: Elaborado pelos autores.

Além dos sensores eletrônicos vários outros componentes eletrônicos

compatíveis com arduino foram utilizados para a confecção do *gateway.*

Para obter as informações de tempo foi utilizado o DS1307, módulo (RTC - *Real Time Clock*), este é de baixo consumo de energia, possui calendário completo e mais 56 bytes de SRAM, sendo capaz de fornecer informações como segundos, minutos, dia, data, mês e ano. O RTC possui ainda uma EEPROM 24C32, que pode ser usada para gravar dados. Utiliza o protocolo de comunicação I2C. O I2C é um protocolo de baixa velocidade de comunicação criado pela Philips para comunicação entre placa mãe e dispositivos, sistemas embarcados e circuitos de celulares.

Para disponibilizar os dados do *gateway* na internet foi utilizado uma placa Ethernet Shield W5100. Esta Shield baseia-se no chip WIZnet ethernet W5100 (datasheet) que fornece acesso à rede (IP) nos protocolos TCP ou UDP e é facilmente utilizado usando a biblioteca Ethernet Library e SD Library. A W5100 é compatível tanto com o arduino Uno quanto com o Mega e possui um slot para cartão micro-SD que pode ser usado para armazenar arquivos que podem servir na rede.

Para exibir as informações em campo foi incluso no *gateway* um display LCD 20x4 (20 colunas por 4 linhas), este é um equipamento com fundo azul que possui capacidade de suportar a exibição de até 20 caracteres por linha em uma tela de 4 linhas, sendo especialmente indicado para aqueles que desenvolvem projetos com arduino. Compatível com um grande número de sistemas microcontroladores, o display LCD 20x4 é um equipamento que pode ser utilizado junto ao arduino, PIC, Atmel, Raspberry Pi, entre outros tipos.

O display LCD 20x4 conta com uma luz de fundo em LED que facilita a visualização de dados junto a tela, contribuindo assim para o aperfeiçoamento do projeto realizado pelos projetistas. O display é geralmente utilizado em projetos eletrônicos ou robóticos com o objetivo de aumentar a integração do projeto com os espectadores, trazendo também informações úteis ao operador do sistema. O mesmo pode ser operado tanto em 4 quanto em 8-bits.

Para facilitar a conexão dos componentes em teste foi utilizada uma *protoboard* de 830 pontos, também conhecida como *breadboard*, placa de ensaio ou matriz de contato, é uma placa com furos e conexões pré-definidas, que visa auxiliar a montagem de teste de circuitos eletrônicos experimentais de forma

simples e ágil. A grande vantagem da placa de ensaio na montagem de circuitos eletrônicos é a facilidade de inserção de componentes, uma vez que não necessita soldagem.

O controle dos atuadores é feito utilizando um módulo de relés a fim de eliminar o trabalho do circuito de ativação com transistores, relés, conectores, leds e diodos. As especificações dos relés do módulo selecionado são as seguintes: Modelo: SRD-05VDC-SL-C, tensão da operação: 5VDC, permite controlar cargas de 220V AC, corrente típica da operação: 15~20mA, LED indicador de status, Pinagem: Normal Aberto, Normal Fechado e Comum, Tensão de saída: (30 VDC a 10A) ou (250VAC a 10A), furos de 3mm para fixação nas extremidades da placa e tempo de resposta: 5~10ms.

Para alimentação elétrica do *gateway* foi utilizado uma fonte chaveada, tensão de entrada: Bivolt 100~240VAC 47~64Hz, tensão de saída: 9VDC e corrente de saída máxima: 1A. A Figura 15 apresenta alguns dos componentes eletrônicos acima descritos.

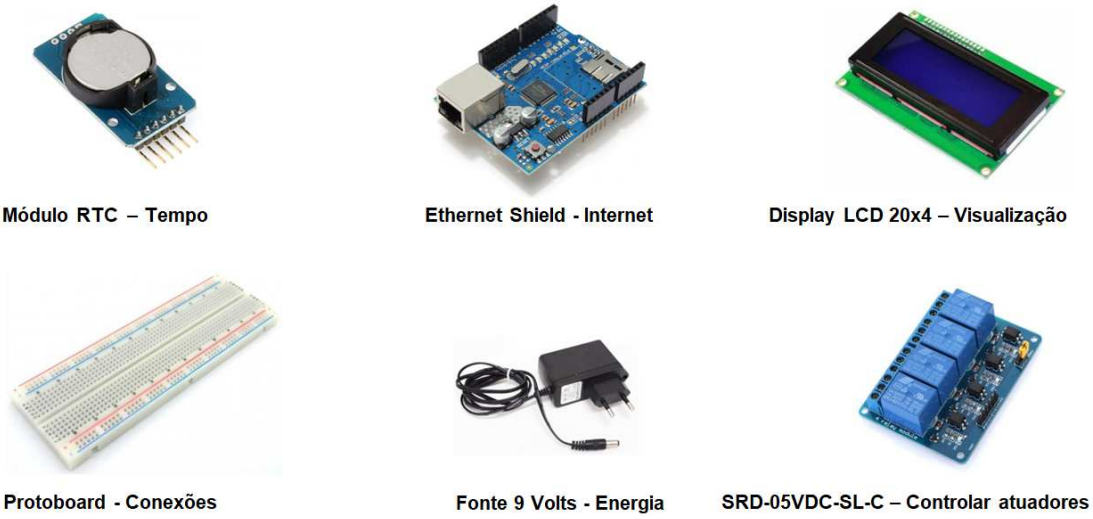


Figura 15: Alguns dos componentes eletrônicos utilizados no projeto
Fonte: Elaborado pelos autores.

Para montar o *gateway* foi utilizada uma caixa especialmente projetada para ser resistente à exposição solar e também à resistência de chuva. Os detalhes da montagem do *gateway* podem ser verificados na Figura 16.

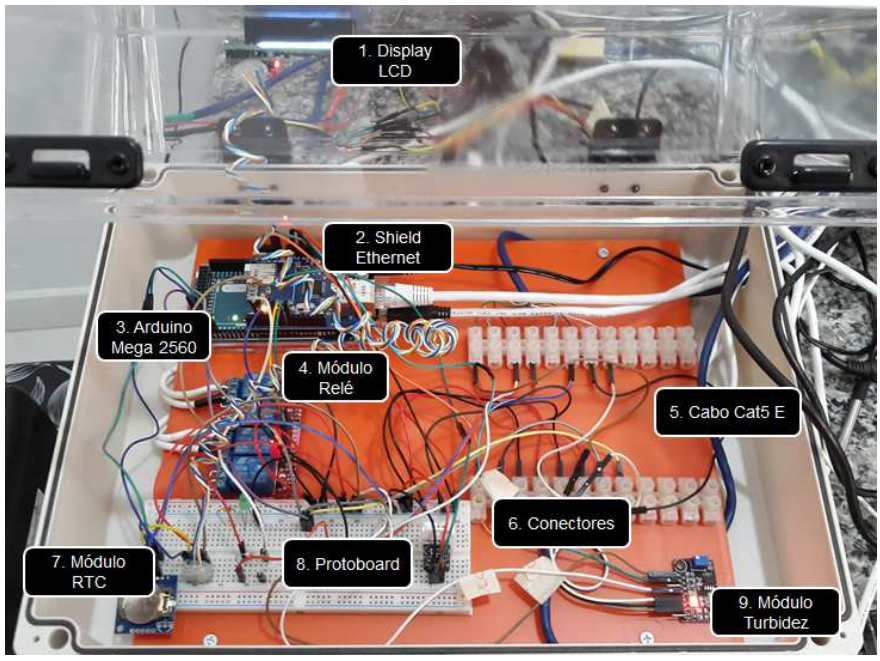


Figura 16: *Gateway* do sistema de monitoramento da aquicultura
Fonte: Elaborado pelos autores.

Na Figura 16 o “componente 1. Display LCD” exibe as informações dos sensores e atuadores em tempo real; o “componente 2. Shield Ethernet” está conectado à placa arduino e permite a conexão com a internet; o “componente 3. Arduino Mega 2560” é o microcontrolador do *gateway*; o “componente 4. Módulo Relé” é o acionador dos atuadores: bomba de recirculação, bomba de acionamento e alimentador automatizado; o “componente 5. Cabo Cat5 E” conduz dados e energia até os sensores; o “componente 6. Conectores”, serve para simplificar e tornar mais prática a conexão dos componentes externos, que neste caso, são os sensores e atuadores que se conecta aos cabeamentos internos do *gateway* via Cabo Cat5 E; o “componente 7. Módulo RTC” fornece informação de data e tempo para o sistema; o “componente 8. Protoboard” serve para fazer as conexões necessárias entre os componentes do sistema; o “componente 9. Módulo Turbidez” é parte do sensor de turbidez e foi incluso no *gateway* a fim de obter maior desempenho e aumentar a vida útil do sensor, protegendo contra umidade.

A Figura 17 mostra os demais componentes, sensores e atuadores, conectados ao *gateway* confeccionado e ao lado o sistema web de monitoramento da aquicultura.

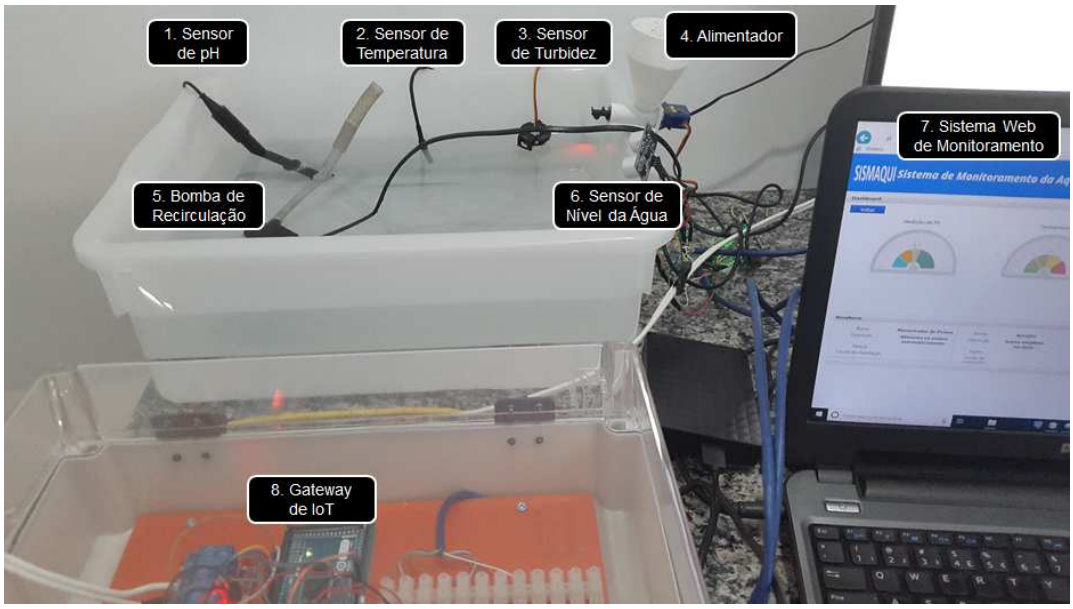


Figura 17: Componentes conectados ao *gateway* de IoT
Fonte: Elaborado pelos autores.

Para implementação dos sensores apresentados na Figura 17 no arduino foi utilizado a IDE padrão do arduino que usa linguagem wiring que é basicamente C/C++, os detalhes do código implementado no arduino podem ser observados na sequência, a Figura 18 apresenta as funções implementadas para ligar e desligar a bomba de recirculação da água dos açudes.

```
116 //=== Funções Ligar e Desligar Bomba para Recirculação ===
117 void ligaBombaRecirculacao(){
118   Serial.println("A bomba de recirculação está ligada");
119   delay(100);
120   digitalWrite(porta_rele2_bomba_recirculacao, HIGH); //Liga rele 2
121   delay(200);
122 }
123
124 void desligaBombaRecirculacao(){
125   Serial.println("A bomba de recirculação está desligada");
126   delay(100);
127   digitalWrite(porta_rele2_bomba_recirculacao, LOW); //Desliga rele 2
128   delay(200);
129 }
```

Figura 18: Funções para ligar e desligar a bomba de recirculação
Fonte: Elaborado pelos autores.

A chamada das funções apresentadas na Figura 18 é feita no laço de repetição (loop), conforme pode ser observado na Figura 19.

```
631 //=== Loop para Sensor de Temperatura da Água ========
632     float temp[qtdeSensores];
633     int x,y,i,j;
634     sensors.requestTemperatures();
635     for(i=0;i<qtdeSensores;i++){
636       temp[i] = sensors.getTempC(sensores[i]);
637     Serial.print("Temperatura da Água ");
638       //Serial.print(i); Para usar vários sensores
639         Serial.print(": ");
640           Serial.print(temp[i]);
641           lcd.setCursor(0, 0);
642           lcd.print("Temp Agua: ");
643           lcd.print(temp[i]);
644           Serial.println("°C");
645           lcd.print("C");
646           tempagua = (float(temp[i]));
647      //== Verifica se precisa ligar a bomba ==
648      if (temp[i]>30){
649       ligaBombaRecirculacao();
650      }else{
651       desligaBombaRecirculacao();
652      }
653   }
654   delay(10);
```

Figura 19: Condições para ligar ou desligar a bomba de recirculação

Fonte: Elaborado pelos autores.

Na linha 648 do código apresentado na Figura 19 é verificado se a temperatura da água está acima de 30ºC, se sim o sistema liga a bomba de recirculação se não passa para a linha 651 e a bomba permanece desligada ou é desligada se já estiver ligada.

A Figura 20 apresenta as funções implementadas para ligar e desligar a bomba de abastecimento de água para os açudes.

```
131 //=== Funções Ligar e Desligar Bomba de Abastecimento ===
132 void ligaBombaAbastecimento(){
133   Serial.println("A bomba de abastecimento está ligada");
134   delay(100);
135   digitalWrite(porta_rele3_bomba_abastecimento, HIGH); //Liga rele 3
136   delay(200);
137 }
138
139 void desligaBombaAbastecimento(){
140   Serial.println("A bomba de abastecimento está desligada");
141   delay(100);
142   digitalWrite(porta_rele3_bomba_abastecimento, LOW); //Desliga rele 3
143   delay(200);
144 }
```

Figura 20: Funções para ligar e desligar a bomba de abastecimento

Fonte: Elaborado pelos autores.

A chamada das funções apresentadas na Figura 20 também é feita no loop, conforme pode ser observado na Figura 21.

```
170 //=== Bloco no Loop para Sensor de Nível da Água - Ultrasônico ========
171   delay(10);
172   Serial.print("Nível da Água: ");
173   int dist = sonar.ping();
174   Serial.print(sonar.convert_cm(dist)); // Converte distância para centimetros
175   Serial.println("cm");
176     if (sonar.convert_cm(dist)>20){
177       ligaBombaAbastecimento();
178     } else {
179       desligaBombaAbastecimento();
180     }
```

Figura 21: Condições para ligar ou desligar a bomba de abastecimento

Fonte: Elaborado pelos autores.

Tendo em vista que para todo atuador controlado seria necessário implementar uma função de ligar e desligar um determinado equipamento uma forma de otimizar e deixar o código sintetizado, foi implementando uma função genérica conforme pode ser visto na Figura 22.

```
298   void ligaOuDesligaEquipamento(int porta, String statusrequerido, String msg){
299     Serial.println(msg);
300     delay(100);
301     digitalWrite(porta, 'statusrequerido');
302     delay(100);
303   }
```

Figura 22: Função genérica para ligar ou desligar equipamentos

Fonte: Elaborado pelos autores.

Na linha 298 da Figura 22 pode ser observado que os parâmetros necessários devem ser passados ao chamar a função. Um exemplo de utilização da função genérica está ilustrado na Figura 23.

```
715   delay(10);
716   Serial.print("Nível da Água: ");
717   int dist = sonar.ping();
718   Serial.print(sonar.convert_cm(dist)); //Converte distância em centimetros
719   nivelagua = sonar.convert_cm(dist);
720   Serial.println("cm");
721     if (sonar.convert_cm(dist)>20){
722       ligaOuDesligaEquipamento(porta_rele3_bomba_abastecimento,"HIGH",
723       "A bomba de abastecimento está ligada"); //ligaBombaAbastecimento();
724     } else {
725       ligaOuDesligaEquipamento(porta_rele3_bomba_abastecimento,"LOW",
726       "A bomba de abastecimento está desligada");//desligaBombaAbastecimento();
727     }
```

Figura 23: Utilização da função genérica para ligar ou desligar equipamentos
Fonte: Elaborado pelos autores.

Na linha 721 da Figura 23 é verificada que, se a distância entre o sensor ultrassônico e a água for maior que 20 cm, a função genérica é tida com parâmetros que correspondem a função de ligar a bomba de abastecimento de acordo com o que pode ser observado na linha 722, caso contrário, conforme a linha 725, a função genérica é chamada com parâmetros que correspondem ao evento de desligar a bomba de abastecimento.

No sistema também foi implantado códigos para leitura dos sensores de turbidez e pH sendo que fora necessário a implementação de cálculos matemáticos para calibragem das leituras de ambos. O código implementado para leitura do sensor de turbidez pode ser visualizado na Figura 24.

```
189 //=== Loop Sensor de Turbidez ========
190   int sensorValue = analogRead(A0);// Leitura do pino analógico 0:
191   float voltage = sensorValue * (5.0 / 1024.0);
192   Serial.print("Turbidez:");
193   Serial.print(voltage); //Imprime a saída
194   Serial.println("V");
195   delay(500);
196 //=== Loop Sensor de Turbidez ========
```

Figura 24: Implementação para o sensor de turbidez
Fonte: Elaborado pelos autores.

Na linha 191 da Figura 24 o código converte as leituras analógicas (Com valores de 0 a 1023) para voltagem (0 - 5V).

O código da Figura 25 foi implementado para realizar a leitura do pH.

```
595  static unsigned long samplingTime = millis();
596    static unsigned long printTime = millis(); static float pHValue,voltagePh;
597    if(millis()-samplingTime > samplingInterval){
598        pHArray[pHArrayIndex++]=analogRead(SensorPin);
599        if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
600        voltagePh = mediadoarray(pHArray, ArrayLenth)*5.0/1024;
601        pHValue = 3.5*voltagePh+Offset;
602        samplingTime=millis();
603    }
604    if(millis() - printTime > printInterval){
605      Serial.print("pH = Voltagem:");
606      Serial.print(voltagePh,2);
607      voltagemph = (float(voltagePh));
608      Serial.print(";  Valor: ");
609      Serial.println(pHValue,2);
610      valorph = (float(pHValue));
611      lcd.setCursor(0, 0);
612      lcd.print("Voltagem pH: "); lcd.print(voltagePh,2);
613      delay(300); lcd.setCursor(0, 1);
614      lcd.print("Valor pH: "); lcd.print(pHValue,2);
615      delay(100); printTime=millis();
616    }
```

Figura 25: Implementação para o sensor de pH
Fonte: Elaborado pelos autores .

Para que o valor da leitura do pH seja mais preciso foi implementado um código que retorna a média de uma sequência de valores armazenados na função mediadoarray. O código da função que armazena os valores, mediadoarray, em uma lista para cálculo das médias está apresentado na Figura 26.

```
839  double mediadoarray(int* arr, int number){
840    int i; int max,min; double avg; long amount=0;
841    if(number<=0){
842      Serial.println("Número inválido para cálculo da média!/n"); return 0;
843    }
844    if(number<5){   //Menos do que 5, calcula diretamente
845      for(i=0;i<number;i++){
846        amount+=arr[i];
847      } avg = amount/number; return avg;
848    }else{
849      if(arr[0]<arr[1]){
850        min = arr[0];max=arr[1];
851      }else{
852        min=arr[1];max=arr[0];
853      }
854      for(i=2;i<number;i++){
855        if(arr[i]<min){
856          amount+=min;       //para array menor que min
857          min=arr[i];
858        }else {
859          if(arr[i]>max){
860            amount+=max;     //para array maior que max
861            max=arr[i];
862          }else{
863            amount+=arr[i]; //para min<=array<=max
864          }
865        }
866      } avg = (double)amount/(number-2);
867    } return avg;
868  }
```

Figura 26: Função que retorna a média dos valores
Fonte: Elaborado pelos autores.

Na segunda etapa o conceito de IoT foi totalmente implementado no *gateway*, o módulo ethernet foi incluso, portanto, o componente já pode ser identificado na rede de internet permitindo assim que todos os sensores e atuadores, nele conectado, sejam também acessados e controlados.

Por meio da conexão de internet estabelecida o sistema também grava as informação coletadas no banco de dados MySQL que está em um servidor remoto possibilitando o detalhamento posteriormente dos dados que são continuamente coletados e armazenados no banco de dados do sistema que garante maior rastreabilidade ao processo.

A conexão com o banco de dados fora feita utilizando o conector MySQL para arduino (Biblioteca MySQL_Connection), que é uma biblioteca para o arduino. A vantagem deste conector em relação a outras formas de conexão estudadas é a desnecessidade de se ter um servidor intermediário.

Em caso de falha na rede o contingenciamento das gravações poderá ser feito via cartão de memória SSD, sendo utilizado como um buffer, pois assim que a conexão reestabelece, o sistema passa a atualizar a base de dados com as informações gravadas no cartão.

Com a conexão na internet também foi possível implementar o sistema de notificações via email, o qual envia alertas para o operador quando os limites inferiores ou superiores são excedidos.

4.3 TESTES EM CAMPO

Na terceira e última etapa, após as implementações, o *gateway* foi implantado em campo conforme pode ser observado na Figura 27.

Figura 27: *Gateway* de IoT implantado em campo
Fonte: Elaborado pelos autores.

Como pode ser visto na Figura 27, o Gateway foi incluso em uma caixa com tampa a fim de proteger os componentes eletrônicos usados (tomadas, *access point*, fontes de energia, etc.) da chuva.

Para os testes com os peixes foram usados dois tanques, um com

monitoramento do sistema e o outro sem o monitoramento, ambos com capacidade de 1000 litros e com 5 peixes em cada um, os peixes foram colocados com 40 dias de vida. O sistema ficou continuamente ligado, ou seja, 24 horas por dia. O tanque utilizado conjuntamente com os sensores pode ser observado na Figura 28.

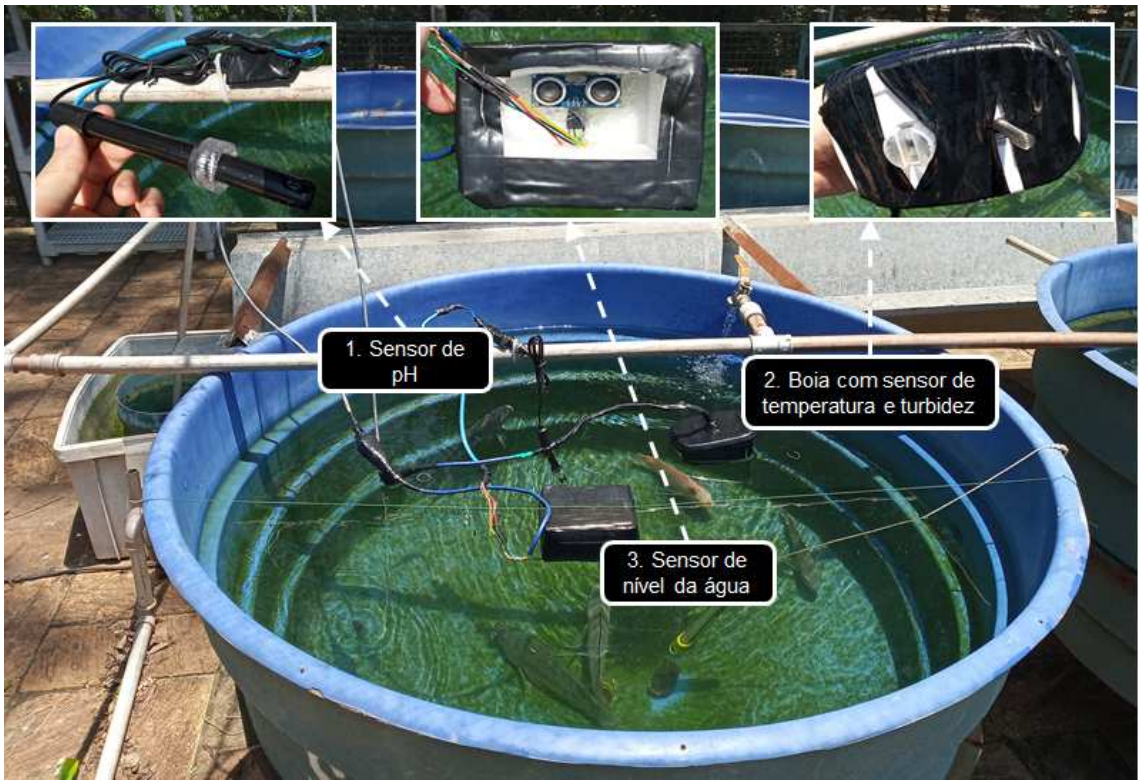


Figura 28: Tanques de peixes com monitoramento automatizado
Fonte: Elaborado pelos autores.

Como observado na Figura 28 os sensores estão instalados no tanque e recebem energia e conduzem os dados via cabeamentos Cat5E. Como pode ser observado no item 02, foi necessário proteger alguns sensores da exposição direta ao tempo e até mesmo dos peixes que poderiam danificá-los por estar de fácil acesso.

Na Figura 29 é apresentado o sistema de recirculação que serve para filtrar e oxigenar a água dos peixes.

Figura 29: Sistema de recirculação
Fonte: Elaborado pelos autores.

Na Figura 29 o "item 1. Saída de Água" serve para tirar a água depositada a fim de liberar espaço para entrada da água para recirculação que é feita conforme "item 2. Entrada de Água"; o "item 3. Bomba d'Água" envia a água da saída novamente para a entrada; o "item 4. Sensor de Vazão" mede o fluxo de água que está indo para as entradas; o "item 5. Fuga do Excesso" serve para evitar que a água limitada nas válvulas das entrada extrapole o nível máximo da caixa de filtragem.

O *gateway* permaneceu implantado por um período de 4 meses e 12 dias, com inicio em 23 de setembro de 2019 e término em 06 de fevereiro de 2020.

O sistema foi configurado para realizar uma leitura a cada 16 segundos, em média. Os totais das leituras por sensor e de alertas gerados por atuadores estão apresentadas na Tabela 2.

Tabela 2 – Dados Coletados Pelo *Gateway*

Leituras	
Sensor	**Totais de Leituras**
Nível	31.116
pH	44.952
Temperatura da água	218.481
Turbidez	45.018
Umidade e Temperatura Interna	218.474
Alertas	
Equipamento	**Totais de Alertas Gerados**
Sensor de temperatura	48.237 (Gerados conforme os limites da temperatura)
Alimentador de peixes	218.462 (Gerados automaticamente para testar a função)

Fonte: Elaborado pelos autores.

O total de leituras válidas realizadas pelos sensores, conforme a soma dos dados apresentados na Tabela 2 é de 558.041 registros e a soma dos alertas gerados é de 266.699. É importante ressaltar que os dados de alerta do alimentador automatizado de peixes foram gerados automaticamente com intuito de testar as funções de controle e persistência dos dados ou seja não havia um alimentador físico sendo monitorado.

No decorrer do período de leitura houve algumas interrupções devido à manutenção que estava sendo realizada na infraestrutura de internet do IFPR – Campus Foz do Iguaçu e também em virtude da necessidade de alteração da rota do cabeamento da internet tendo em vista a instalação de estufas que foram implantadas enquanto a coleta era realizada.

Durante o tempo em que estiveram nos tanques os peixes foram alimentados com ração todos os dias com duas alimentações diárias no período da manhã e da tarde. Os peixes dos dois tanques foram pesados um total de 5 (cinco) vezes com intuito de acompanhar o desenvolvimento dos mesmos, a Tabela 3 apresenta o histórico das pesagens realizadas.

Tabela 3 – Pesagens Realizadas

Datas				
17/09/2019	18/11/2019	17/01/2020	31/01/2020	07/02/2020
Pesos				
Tanque 1				
0,102	0,130	0,166	0,198	0,205
0,151	0,195	0,248	0,254	0,260
0,095	0,120	0,150	0,164	0,173
0,106	0,129	0,170	0,172	0,180
0,092	0,115	0,148	0,152	0,161
Tanque 2				
0,155	0,198	0,272	0,295	0,300
0,140	0,187	0,226	0,255	0,260
0,102	0,125	0,178	0,198	0,204
0,157	0,218	0,278	0,289	0,294
0,065	0,080	0,100	0,116	0,122

Fonte: Elaborado pelos autores.

Os dados do tanque 1, apresentados na Tabela 3, são referentes ao tanque que teve o monitoramento automatizado, o tanque 2 não teve monitoramento do *gateway*. O peso médio dos peixes confinados nos tanques foi: tanque 1 igual a 0,161.44 kg e tanque 2 igual a 0,192.56 kg. O desvio padrão do tanque 1 foi de 31.4302 e do tanque 2 foi de 57.0150.

4.4 COMPARAÇÕES COM TRABALHOS DE OUTROS AUTORES

Para elaboração deste trabalho várias bibliografias com temas relacionados ao desta pesquisa, foram selecionadas e estudadas com a finalidade de conhecer: os setores de aplicação da IoT, as tecnologias utilizadas, analisar os processos de implantação utilizados por outros autores, as boas práticas recomendadas e os problemas não solucionados.

Identificou-se trabalhos cujos resultados possuem certas similaridades ao deste, contudo os objetivos são diferentes tendo em vista que a maior parte teve

como objetivo a automatização propriamente dita enquanto este visou facilitar a automatização por meio do encapsulamento da complexidade existente nos processos necessários para a automação da aquicultura utilizando para isto um *gateway* de IoT.

Um exemplo de trabalho similar ao dessa pesquisa é o de Kyaw; Keong (2017), que implementou um sistema de aquaponia inteligente para fazenda urbana que igualmente apresenta mecanismos de aquisição de dados por meio de sensores, unidades de alarme, unidade central de processamento e assim como esse também disponibiliza interfaces web para monitoramento em tempo real dos dados coletados e exibe o histórico das coletas e dos alertas gerados e utilizou também o arduino como plataforma contudo é voltado para a aquaponia e não para a aquicultura.

Outro exemplo está no trabalho de Lloret et al. (2018), que elaborou o projeto e implantação de sensores de baixo custo para monitoramento da qualidade da água e do comportamento dos peixes em tanques de aquicultura durante o processo de alimentação haja visto também que ter utilizado sensores sobre a plataforma arduino e sensores para monitorar itens de qualidade da água da aquicultura tais como: temperatura e turbidez, todavia, o foco fora diferente desse projeto pois a ideia principal, aqui, não foi apenas automatizar porem aplicar o conceito de IoT visando reduzir a complexidade das tarefas de automatização reduzindo o custo como um todo e não apenas dos sensores.

4.5 DISCUSSÕES COMPLEMENTARES

Um dos resultados técnico-científico alcançado com esta pesquisa foi o estudo e seleção das técnicas mais adequadas para otimizar a utilização de recursos no complexo da aquicultura.

Entre os resultados socioeconômicos esperados estão: o fortalecimento do agronegócio no âmbito do comércio, aumento da lucratividade por meio da redução de custos e melhoria dos resultados e geração de empregos no setor.

5 CONSIDERAÇÕES FINAIS

É prudente avaliar totalmente os novos locais de aquicultura ou propostos, antes de qualquer investimento financeiro substancial em infraestrutura e pessoal. Medições da temperatura da água, turbidez, pH, salinidade e oxigênio dissolvido podem ser usados para obter informações sobre a qualidade física, química e biológica da água e das condições dentro de uma fazenda, para identificar sua adequação para a aquicultura, tanto para as espécies de interesse como para avaliar o risco potencial de algas nocivas ou tóxicas.

Este último pode provocar o fechamento da coleta de mariscos em determinados períodos. Infelizmente, os sistemas de monitoramento científico comercial podem ter um custo proibitivo para pequenas organizações e empresas desde a compra até a operação, contudo a utilização de um *gateway* de IoT pode prover uma relevante redução do valor total tendo em vista que a maior parte do trabalho necessário para a automatização já está implementada neste componente.

5.1 ABRANGÊNCIA DO ESTUDO

O estudo realizado neste trabalho atende à automatização do complexo da aquicultura, contudo não se limita a este, grande parte do conhecimento aqui compilado poderá ser aplicado em outros complexos do ramo do agronegócio após a realização de determinadas adequações tais como: tratamento para outros tipos de sensores, modelagem para utilização de diferentes atuadores e tratamento das exceções e restrições existentes em cada setor produtivo.

O *gateway* de IoT desenvolvido também pode ser utilizado, após as devidas adequações, para automatização dos processos produtivos de: hidroponia, aquaponia, green houses, ranicultura, aviários, suinocultura, etc.

Após o levantamento bibliográfico realizado nas disciplinas de seminários foi possível comprovar a viabilidade em continuar com o tema do projeto de pesquisa, contudo foi necessário realizar algumas adaptações e redução do escopo inicialmente proposto.

5.2 DIFICULDADES ENCONTRADAS

As principais dificuldades para realizar o projeto são fragilidade e curto tempo de vida dos equipamentos eletrônicos quando utilizados em ambientes de intensa umidade e necessidade de calibrar alguns sensores selecionados antes do uso. Os sensores que apresentaram aspectos negativos foram SEN0189 (Turbidez) e ultrassônico HC-SR04 (Nível da água).

Por mais que o SEN0189 tenha sido isolado com vedações para evitar a entrada de água a própria umidade do local faz com que a água se infiltre na parte interior do componente e isso reduz sua precisão e devido ao contato da parte elétrica com a água poderia ter ocorrido curto-circuito. O HC-SR04 após implantado coletou, normalmente, os dados por 45 dias neste período começou a ser corroído por ferrugem e após 60 dias estava completamente danificado, contudo em outro projeto este pode ser substituído pelo HC-SR04 a prova d'água.

Para a efetivação da utilização da IoT na aquicultura e em outros complexos do agronegócio ainda é necessário o desenvolvimento de sensores mais resistentes a umidade, poeira, calor e a outras variáveis presentes no meio rural e ainda mais focados em reduzir a necessidade de interação humana ao contrário de alguns como o sensor de pH que precisa ser calibrado antes do uso, já que estes somente serão confeccionados se houver demanda e para isto é importante começar com o que está no mercado.

5.3 TRABALHOS FUTUROS

Trabalhos futuros que poderão ser realizados dando continuidade ou expandindo a abrangência deste projeto são, por exemplo, automatização de processos rurais baseada em IoT integrada com ferramentas analíticas, *datamining* e *machine learning* visando agregar valor à camada de inteligência da IoT, automatização baseada em IoT dos processos produtivos de: aquaponia, green houses, ranicultura, suinocultura, aviários, pecuária, etc.

Implementação de *gateway* de IoT baseado em protocolo *Message Queuing Telemetry Transport* (MQTT) focado em automatização de processos de complexos do agronegócio.

REFERÊNCIAS

ARDUINO. **Arduino Products.** <https://www.arduino.cc/en/Main/Products>. Acesso em: 12 abr. 2019.

BALDISSEROTTO, Bernardo. **Fisiologia de peixes aplicada à piscicultura**. Editoraufsm, Santa Maria – RS. 2009.

BENAVENT, Muñoz, P. et al. Enhanced fish bending model for automatic tuna sizing using computer vision. **Computers and Electronics in Agriculture**, Espanha, v. 150, 2018. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

BOKINGKITO JR, Paul B., LLANTOS, Orven E. Design and Implementation of Real-Time Mobile-based Water Temperature Monitoring System. **Procedia Computer Science**, Indonésia, v. 124, 2017. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

CAMARGO, Tiago F. B. et al. Termal comfort monitoring in aviaries by a real-time data acquisition system. **Agriambi**, Brasil, v. 23, 2019. Disponível em: < https://www.scielo.com>. Acesso em: 10 dez. 2019.

CASTRO, Nesly Diana C., CHAMORRO, Luis Eduardo F., VITERI, Carlos Andrés M. Uma red de sensores inalámbricos para la automatización y control del Riego localizado. **Revista de Ciencias Agrícolas**, Colombia, v. 33, 2016. Disponível em: < https://www.scielo.com>. Acesso em: 10 dez. 2019.

CECCARELLI, Paulo S., SENHORINI, José A., VOLPATO, Gilson L.. **Dicas em Piscicultura.** Santana Gráfica Editora, Bocatu – SP. 2000.

CHU, Cheng-Shane. SU, Chih-Jen. Optical fiber sensor for dual sensing of H2O2 and DO based on CdSe/ZnS QDs and Ru(dpp)32+ embedded in EC matrix. **Sensors and Actuators B: Chemical**, Taiwan, v. 255, 2018. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=6Bh377Yf5iGDrcJIubS&preferences

Saved=>. Acesso em: 26 jun. 2018.

DANKHARA, Fenil, PATEL, Kartik, DOSHI, Nishant. Analysis of robust weed detection techniques based on the internet of things (IoT). **Procedia Computer Science**, India, 2019. Disponível em: <https://www.sciencedirect.com>. Acesso em: 04 dez. 2019.

DEBAUCHE, Olivier. et al. Web Monitoring of Bee Helth for Researchers and Beekeepers Based on the Internet of Things. **Procedia Computer Science**, Belgium, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 12 dez. 2019.

EMBRAPA. **Piscicultura de água doce: Multiplicando conhecimentos.** Embrapa Editora, Brasília – DF. 2013.

ESPINOSA-FALLER, Francisco J.. RENDÓN-RODRÍGUEZ, Guillermo E.. A ZigBee Wireless Sensor Network for Monitoring an Aquaculture Recirculating System. **Journal of Applied Research and Technology**, México, v. 10, no.3, p. 380-387, 2012. Disponível em: <http://www.scielo.org/php/index.php/>. Acesso em: 07 ago. 2018.

FAO – Food and Agriculture Organization of the United Nations. **The state of world fisheries and aquaculture.** Rome, 2018. Disponível em: < http://www.fao.org/home/en/>. Acesso em: 19 mar. 2019.

GEHLERT, G. et al. Analysis and optimisation of dynamic facility ventilation in recirculation aquacultural systems. **Aquacultural Engineering**, Alemanha, v. 80, p. 1–10, 2018. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

GUIMARÃES, Kevin Manoel. LOHMANN, Daniel. Automação de Tanques para Aquicultura. **Revista Ilha Digital**, Brasil, v. 6, p. 34 – 47, 2017. Disponível em: <https://scholar.google.com.br/>. Acesso em: 18 ago. 2018.

GUNASEKERA, Kutila. et al. Experinces in building an IoT infrastructure for agriculture education. **Procedia Computer Science**, Sri Lanka, 2018. Disponível em: < https://www.sciencedirect.com>. Acesso em: 11 dez. 2019.

HUAN, Juan, CAO, Weijian. QIN, Yilin. Prediction of dissolved oxygen in aquaculture based on EEMD and LSSVM optimized by the Bayesian evidence framework. **Computers and Electronics in Agriculture**, China, v. 150, p. 257–265, 2018. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

JIANG, Jianming. A wireless sensor network-based monitoring system for freshwater fishpond aquaculture. **Biosystems Engineering**, Austrália, v. 172, p. 57 - 66, 2018. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

JI, Changbo. et al. An IoT and Mobile Cloud based Architecture for Smart Planting. **Materials and Information Technology Applications**, China, 2015. Disponível em: <https://scholar.google.com.br/>. Acesso em: 18 ago. 2018.

FOUGHALI, Karim, KARIM, Fathalah, FRIHIDA, Ali. Monitoring system using web of things in precision agriculture. **Procedia Computer Science,** Tunisia, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 10 dez. 2019.

KHAIRE, Supriya R., WAHUL, Revati M.. Water Quality Data Gathering and Analysis System using IoT Environment. **JASC: Journal of Applied Science and Computations**, India, v. 5, 2018. Disponível em: <https://scholar.google.com.br/>. Acesso em: 18 ago. 2018.

KOTHA, Harika Devi. GUPTA, V Mnssvk. IoT Application, A Survey. **International Journal of Engineering & Technology**, India, v. 7 (2.7), p. 891-896, 2018. Disponível em: <https://scholar.google.com.br/>. Acesso em: 18 ago. 2018.

KYAW, Thu Ya. KEONG, Andrew Ng. Smart Aquaponics System for Urban Farming. **Procedia Computer Science**, Singapura, 2017. Disponível em: <

https://www.sciencedirect.com>. Acesso em: 11 set. 2019.

KUBITZA, Fernando. **Tilápia: Tecnologia e planejamento na produção comercial.** Acqua Imagem Editora, Jundiaí – SP. 2011.

LEE, Phillip G. Process control and artificial intelligence software for aquaculture. **Aquacultural Engineering**, EUA, v. 23, p. 13–36, 2000. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=6Bh377Yf5iGDrcJIubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

LLORET, Jaime. et al. Design and Deployment of Low-Cost Sensors for Monitoring the Water Quality and Fish Behavior in Aquaculture Tanks during the Feeding Process. **Sensors**, Espanha, v. 750, 2018. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=6Bh377Yf5iGDrcJIubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

MA, Zeliang. et al. Design and application of electronic tongue system for orange juice quality detection using internet of things. **IFAC PapersOnLine**, China, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 03 dez. 2019.

MAHDAVINEJAD, Mohammad S. et al. Machine learning for internet of things data analysis: a survey. **Digital Communications and Networks**, Iran, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 12 dez. 2019.

MONK, Simon. **Guia do Maker para o Apocalipse Zumbi.** Novatec. São Paulo – SP. 2016.

MONK, Simon. **Projetos com Arduino e Android.** Bookman. Porto Alegre – RS. 2014.

MOSES, M. Balasingh., PARAMESWARI, M. Online measurement of water quality and reporting system using prominent rule controller based on aqua care-IOT.

Design Automation for Embedded Systems, Índia, 2018. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

OLIVEIRA, Sérgio. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi.** Novatec. São Paulo – SP. 2017.

PARRA, Lorena. et al. Design and deployment of a smart system for data gathering in aquaculture tanks using wireless sensor networks. **International Journal Of Communication Systems**, Espanha. Brasil. Portugal., v. 30, 2017. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=6Bh377Yf5iGDrcJIubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

PARRA, Lorena. Design and development of low cost smart turbidity sensor for water quality monitoring in fish farms. **Aquacultural Engineering**, Espanha, v. 81, p. 10–18, 2018. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

PEIXE BR – Associação Brasileira da Piscicultura. **Anuário 2020 Peixe BR da Piscicultura.** Brasil, 2020. Disponível em: <https://www.peixebr.com.br/anuario-2020/>. Acesso em: 14 mar. 2020.

POPOVIC, Tomo. et al. Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: A case study. **Computers and Electronics in Agriculture**, Montenegro, v. 140, p. 255–265, 2017. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

PULE, Mompoloki, YAHYA, Abid. CHUMA, Joseph. Wireless sensor networks: A survey on monitoring water quality. **Journal of Applied Research and Technology**, Botswana, v. 15, p. 562–570, 2017. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

QIUWEI, Bai. et al. Real-time remote monitoring system for aquaculture water

quality. **International Journal Of Agricultural And Biological Engineering**, China, v. 8, 2015. Disponível em: <http://apps-webofknowledge.ez48. periodicos. capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=General Search&SID=6Bh377Yf5iGDrcJIubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

QUERALTA, J. P., et al. Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN. **Procedia Computer Science,** Filand, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 10 dez. 2019.

RAY, P.P. A survey on Internet of Things architectures. **Journal of King Saud University – Computer and Information Sciences**, India, v. 30, p. 291–319, 2018. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago. 2018.

ROMLI, Muhamad Asmi. Aquaponic Growbed Water Level Control Using Fog Architecture. **Journal of Physics**, Malásia, v. 1018, 2018. Disponível em: <https://scholar.google.com.br/>. Acesso em: 18 ago. 2018.

SARQUIS, José Buainain. **Comércio Internacional e Crescimento Econômico no Brasil**. Funag. Brasília – DF. 2011.

SIMBEYE, Daudi S. ZHAO, Jimin. YANG, Shifeng. Design and deployment of wireless sensor networks for aquaculture monitoring and control based on virtual instruments. **Computer and Electronics in Agriculture**, China, v. 102, p. 31 – 42, 2014. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes. gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch &SID=6Bh377Yf5iGDrcJIubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

SCHMIDT, Wiebke. Design and operation of a low-cost and compact autonomous buoy system for use in coastal aquaculture and water quality monitoring. **Aquacultural Engineering**, Reino Unido, v. 80, p. 28–36, 2018. Disponível em: <https://www-sciencedirect.ez48.periodicos.capes.gov.br/>. Acesso em: 07 ago.

2018.

SCHULZ, C. et al. Analysis and optimisation of dynamic facility ventilation in recirculation aquacultural systems. **Aquacultural Engineering**, Alemanha, v. 80, 2018. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

SINHA, Rashmi S., WEI, Yiqiao, HWANG, Seung-Hoon. A survey on LPWA technology: LoRa and NB-IoT. **ICT Express 3**, Republic of Korea, 2017. Disponível em: <https://www.sciencedirect.com>. Acesso em: 03 dez. 2019.

TERVONEN, Jouni. Experiment of the quality control of vegetable storage based on the Internet-of-Things. **Procedia Computer Science,** Filand, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 02 dez. 2019.

WANG, Jizhang. et al. Manage system for internet of things of greenhouse based on GWT. **Information Processing in Agriculture**, China, 2018. Disponível em: <https://www.sciencedirect.com>. Acesso em: 02 dez. 2019.

WOLFERT, Sjaak. et al. Big Data in Smart Farming – A review. **Agricultural Systems**, The Netherlands, 2017. Disponível em: <https://www.sciencedirect.com>. Acesso em: 03 dez. 2019.

XU, Jing. et al. Application of Fault Tree Analysis and Fuzzy Neural Networks to Fault Diagnosis in the Internet of Things (IoT) for Aquaculture. **Sensors**, China, v. 153, 2017. Disponível em: <http://apps-webofknowledge.ez48.periodicos.capes.gov.br/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=6Bh377Yf5iGDrcJlubS&preferencesSaved=>. Acesso em: 26 jun. 2018.

ZUPO, V. et al. Automated culture of aquatic model organisms: shrimp larvae husbandry for the needs of research and aquaculture. **Animal**, Itália, v. 12:1, p. 155–163, 2017. Disponível em: <https://www.scopus.com/search/form.uri?display=basic>. Acesso em: 26 jun. 2018.

APÊNDICE A - Artigos publicados

Título: Internet das coisas (IoT) aplicada ao agronegócio: Projeto e implementação de um *gateway* de IoT sobre a plataforma arduino para simplificar a automatização da aquicultura
Periódico: Brazilian Journal of Development
ISSN: 2525-8761
DOI: 10.34117/bjdv5n11-292
Qualis: B2
Link: http://www.brazilianjournals.com/index.php/BRJD/article/view/4899

Título: Projeto e implementação de um *gateway* de internet das coisas (IoT) para otimização e monitoramento de processos do agronegócio
Periódico: Brazilian Journal of Development
ISSN: 2525-8761
DOI: 10.34117/bjdv6n1-023
Qualis: B2
Link: http://www.brazilianjournals.com/index.php/BRJD/article/view/5845

Título: Automatização de processos rurais: Proposta de implementação de um *gateway* de Internet das Coisas (IoT) para simplificar a automatização da aquicultura
Periódico: The Journal of Engineering and Exact Sciences
ISSN: 2527-1075
DOI: 10.18540/jcecvl6iss1pp0001-0007
Qualis: B4
Link: https://periodicos.ufv.br/ojs/jcec

Título: Automação de processos do agronegócio auxiliada pela internet das coisas (IoT): Uma proposta de implementação de um *gateway* de IoT para simplificar a automatização da aquicultura
Evento: ConBRepro 2018: VIII Congresso Brasileiro de Engenharia de Produção
Link: http://aprepro.org.br/conbrepro/2018/anais.php?ordem01=autor&ordem02= au tor

APÊNDICE B - Capítulos publicados

Título: Automação de processos do agronegócio auxiliada pela internet das coisas (IoT): Uma proposta de implementação de um *gateway* de IoT para simplificar a automatização da aquicultura

Editora: Stellata Editora

ISBN: 978-85-94105-05-9

Link: https://stellata.com.br/wp-content/uploads/2019/02/Coletânea-Brasileira-de-Emgnharia-de-Produção-4.pdf

ANEXO A - Levantamento orçamentário

A Tabela 4 apresenta os recursos financeiros que foram gastos para execução do projeto, uma parcela destes recursos foram arcadas pelos autores e o restante foi provido pelos laboratórios das instituições envolvidas.

Tabela 4 - Levantamento Orçamentário para o *Gateway*

Produto	Preço
Placa arduino mega 2560	R$ 95,00
Módulo sensor + pH eletrodo sonda bnc arduino/phmetro	R$ 120,00
Sensor temperatura ntc 10k prova d'agua sonda 2 metro	R$ 20,00
Protoboard compatível com arduino	R$ 30,00
Jumpers de tamanhos variados	R$ 20,00
Resistores 330Ω	R$ 5,00
Resistores 1kΩ	R$ 2,50
Resistor de 15Ω	R$ 11,00
Resistor de 270 Ω e 1/4 W	R$ 25,00
BD139 transistor de potência	R$ 10,00
Medidor de vazão de água arduino	R$ 33,00
Ethernet Shield	R$ 56,90
Sensor de turbidez SEN0189	R$ 240,00
Sensor umidade e temperatura DTH22	R$ 42,90
Display LCD 20x4	R$ 45,00
Sensor ultrassônico HC-SR04	R$ 7,90
Módulo RTC – Tempo	R$ 15,00
Módulo relé 4 canais arduino SRD-05VDC-SL-C	R$ 25,00
Interface I2C	R$ 10,00
Caixa a prova d'água para componentes eletrônicos	R$ 120,00
Total	**R$ 934,20**

Fonte: Elaborado pelos autores.

ANEXO B - Diagrama de circuito do *gateway*

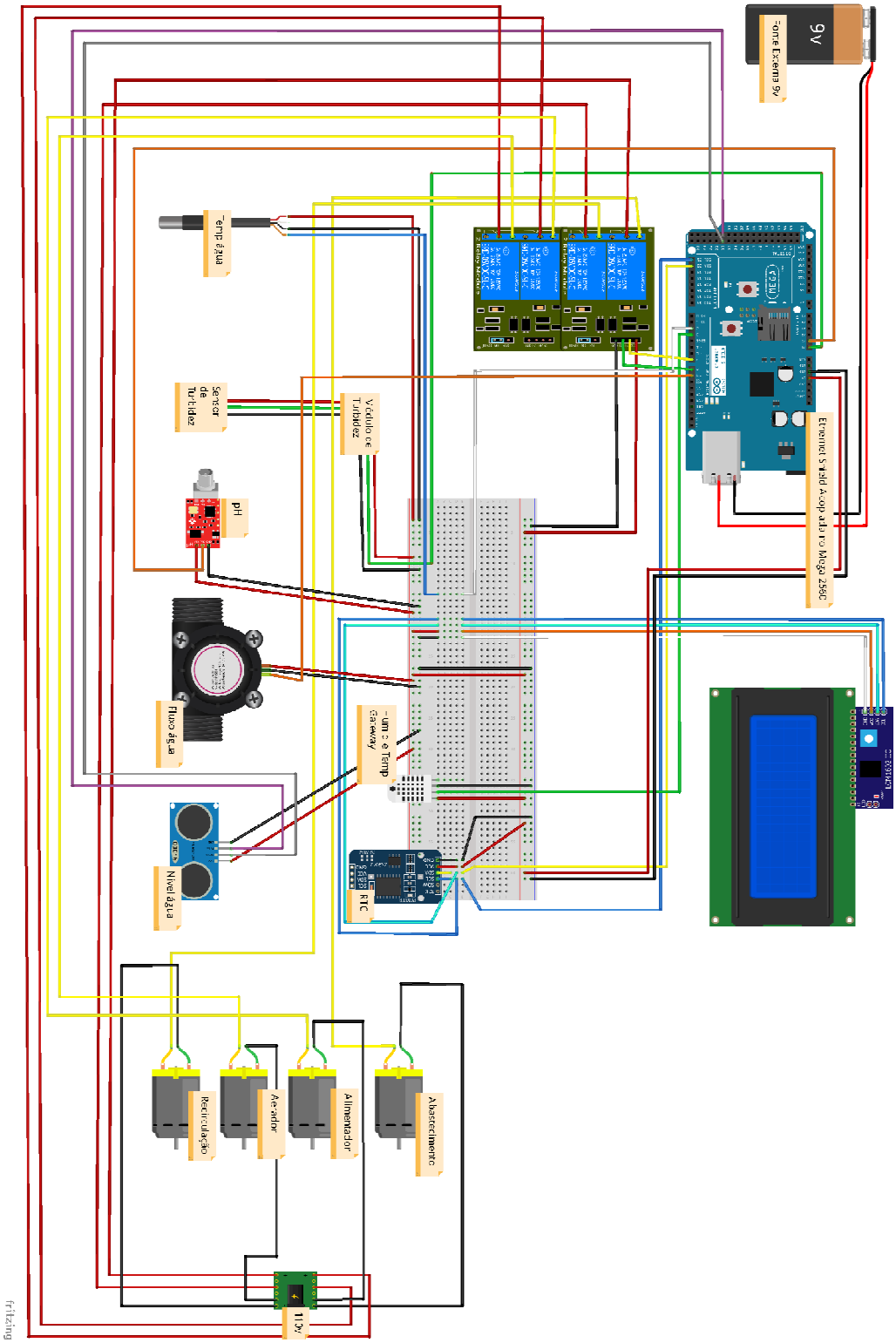


Figura 30: Diagrama de circuito do *gateway*

Fonte: Elaborado pelos autores.

ANEXO C - Códigos do *gateway* implementados no arduino

```
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <MySQL_Encrypt_Sha1.h>
#include <MySQL_Packet.h>
#include <NewPing.h>
#include <RTClib.h>

//=== Declarações MySQL Connector ===
#include <SPI.h>
#include <Ethernet.h>
#define LM35 A0
int leitura;
float leituraconvertida; //converter antes de salvar
float tempagua;
char tempaguaconvertida[10];
float turbidez;
char turbidezconvertida[10];
float tempinterna;
char tempinternaconvertida[10];
float umidainterna;
char umidainternaconvertida[10];
float nivelagua;
char nivelaguaconvertida[10];
float valorph;
char valorphconvertida[10];
float voltagemph;
char voltagemphconvertida[10];
char sentenca[128];
char valortemp[10];
char dia[2];
char mes[2];
char ano[4];
char horas[2];
char minutos[2];
char segundos[2];
byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEF };
IPAddress server_addr(192,000,00,000);  //Endereço IP do servidor de Banco de Dados
char user[] = "nomeusuarioBD";
char password[] = "senhausuarioBD*";
char INSERIR_PH[] = "INSERT INTO leituraph (valorph,voltagemph) VALUES ('%s','%s')";
char INSERIR_TEMP[] = "INSERT INTO leituratemperatura (temperatura) VALUES ('%s')";
char INSERIR_TURB[] = "INSERT INTO leituraturbidez (turbidez) VALUES ('%s')";
char INSERIR_UMID_TEMP_INTERNA[] = "INSERT INTO leituraumidaetempinter
(umidade,temperatura)
VALUES ('%s','%s')";
char INSERIR_NIVEL[] = "INSERT INTO leituranivel (nivel) VALUES ('%s')";
char INSERIR_ALERTA_SENSOR[] = "INSERT INTO alertasensor (descricao,sensor,
valorexcedido)
VALUES ('%s','%s','%s')";

```

```
char INSERIR_ALERTA_ATUADOR[] = "INSERT INTO alertaatuador (descricao,atuador)
VALUES ('%s','%s')";
char BANCODEDADOS[] = "USE arduino";
EthernetClient client;
MySQL_Connection conn((Client *)&client);
//=== Declarações MySQL Connector ===

//=== Declarações Acesso Remoto ===
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    #define qtdePortasDigitaisDoArduino 53
#else
    #define qtdePortasDigitaisDoArduino 13
#endif
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xFF};
IPAddress ip(192, 000, 00, 00); //IP do Gateway
EthernetServer server(80);
String HTTP_req;
String URLValue;
void processaPorta(byte porta, byte posicao, EthernetClient cl);
void lePortaDigital(byte porta, byte posicao, EthernetClient cl);
void lePortaAnalogica(byte porta, byte posicao, EthernetClient cl);
String getURLRequest(String *requisicao);
bool mainPageRequest(String *requisicao);
const byte qtdePinosDigitais = 11;
byte pinosDigitais[qtdePinosDigitais] = { 3     , 5     , 6     , 7     , 8     , 9
, A2    , A3    , A4    , A5};
byte modoPinos[qtdePinosDigitais]     = { OUTPUT, OUTPUT, OUTPUT, OUTPUT, OUTPUT,
OUTPUT, OUTPUT, OUTPUT, OUTPUT, OUTPUT};
const byte qtdePinosAnalogicos = 2;
byte pinosAnalogicos[qtdePinosAnalogicos] = {A0, A1};
//=== Declarações Acesso Remoto ===

//=== Declarações Sensor de Temperatura da Água ===
  #include <DallasTemperature.h>
  #include<OneWire.h>
  #include "Wire.h";
  #include "LiquidCrystal_I2C.h";
  #define ONE_WIRE_BUS 2
  #define qtdeSensores 1
  OneWire oneWire(ONE_WIRE_BUS);
  DallasTemperature sensors(&oneWire);
  DeviceAddress sensores[qtdeSensores] = {{0x28, 0xFF, 0xC0, 0x37, 0x57, 0x16, 0x03,
  0x01}};
//=== Declarações Sensor de Temperatura da Água ===

//=== Declarações Sensor de Umidade e Temperatura do Ar ===
  #include <DHT.h>
  #define DHTPIN 3 //4     // Define o pino digital 4 para conexão
  #define DHTTYPE DHT22   // Define tipo DHT 22  (AM2302)
  DHT dht(DHTPIN, DHTTYPE); //Inicializa o sensor DHT para 16mhz
  int chk;
  float humi;  //Armazena o valor da umidade
  float tempe; //Armazena o valor da temperatura
//=== Declarações Sensor de Umidade e Temperatura do Ar ===

//=== Declarações Módulo Tiny RTC I2C ===
  RTC_DS1307 RTC;
  int intervalortc = 1000;
```

```
109 //=== Declarações Módulo Tiny RTC I2C ===
110   LiquidCrystal_I2C lcd(0x27,20,4);
111 
112 //=== Declarações Sensor de pH ===
113   #define SensorPin A1            //pH porta analógica
114   #define Offset 0.00            //compensação de desvio
115   #define samplingInterval 20
116   #define printInterval 800
117   #define ArrayLenth  40
118   int pHArray[ArrayLenth];   //Guarda o valor da média sensor feedba
119   int pHArrayIndex=0;
120 //=== Declarações Sensor de pH ===
121 
122 
123 //=== Declarações Sensor de Nível da Água - Ultrasônico ===========
124   #define TRIGGER_PIN 30
125   #define ECHO_PIN 31
126   #define MAX_DISTANCE 200
127   NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
128 //=== Declarações Sensor de Nível da Água - Ultrasônico ===========
129 
130 //=== Declarações Atuador para Alimentação dos Peixes - Relés =====
131   int porta_rele1_alimentador = 5;
132 //=== Declarações Atuador para Alimentação dos Peixes - Relés =====
133 
134 //=== Declarações Atuador Bomba para Recirculação ================
135   int porta_rele2_bomba_recirculacao = 6;
136 //=== Declarações Atuador Bomba para Recirculação ================
137 
138 //=== Declarações Atuador Bomba para Abastecimento ================
139   int porta_rele3_bomba_abastecimento = 7;
140 //=== Declarações Atuador Bomba para Abastecimento ================
141 
142 //=== Declarações Sensor de Vazão ================
143   int Pulso; //Variável para a quantidade de pulsos
144   int j=0; //Variável para contagem
145   float vazaoagua; //Variável para armazenar o valor em L/min
146   float valormedia=0; //Variável para tirar a média a cada 1 minuto
147 //=== Declarações Sensor de Vazão ================
148 
149 void setup() {
150 
151 //=== Setup Sensor de Temperatura da Água ===
152   int x,y;
153   Serial.begin(2000000);
154   for(int i=0; i<qtdeSensores; i++)
155   sensors.setResolution(sensores[i], 10);
156 //=== Setup Sensor de Temperatura da Água ===
157 
158 //============ Setup Acesso Remoto ========================
159     Ethernet.begin(mac_addr, ip);
160     server.begin();
161     for (int nP=0; nP < qtdePinosDigitais; nP++) {
```

```
162         pinMode(pinosDigitais[nP], modoPinos[nP]);
163     }
164 //============ Setup Acesso Remoto ========================
165 
166 //============ Setup LCD ==================
167 //Inicializa o LCD e o backlight
168   lcd.init();
169   lcd.backlight();
170   Wire.begin();
171   lcd.begin(20,4);
172 //============ Setup LCD ==================
173 
174 //============ Setup MySQL Connector ========================
175    while (!Serial);
176    //Ethernet.begin(mac_addr);
177    Serial.println("Conectando...");
178    if (conn.connect(server_addr, 3306, user, password))
179    {
180       delay(1000);
181       MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
182       cur_mem->execute(BANCODEDADOS);
183       delete cur_mem;
184    }
185    else
186    {
187       Serial.println("A conexão falhou");
188       while(!conn.connect(server_addr, 3306, user, password)){
189         conn.close();
190         Serial.println("Gravando no cartao");
191         lcd.print("Gravando no cartao");
192         delay(5000);
193         lcd.clear();
194       }
195    }
196 //============ Setup MySQL Connector ========================
197 
198 //============ Setup Tiny RTC I2C ===============
199     Wire.begin();
200     RTC.begin();
201   if (! RTC.isrunning()) {
202     Serial.println("RTC is NOT running!");
203     RTC.adjust(DateTime(__DATE__, __TIME__));
204   }
205 //============ Setup Tiny RTC I2C ===============
206 
207 //=== Setup Atuador para Alimentação dos Peixes - Relés ================
208   pinMode(porta_rele1_alimentador, OUTPUT);
209 //=== Setup Atuador para Alimentação dos Peixes - Relés ================
```

```
210
211 //=== Setup Atuador Bomba para Recirculação ================
212   pinMode(porta_rele2_bomba_recirculacao, OUTPUT);
213 //=== Setup Atuador Bomba para Recirculação ================
214
215 //=== Setup Atuador Bomba para Abastecimento ================
216   pinMode(porta_rele3_bomba_abastecimento, OUTPUT);
217 //=== Setup Atuador Bomba para Abastecimento ================
218
219 //=== Setup Sensor de Vazão ================
220   pinMode(3, INPUT);
221   attachInterrupt(0, incrpulso, RISING);
222 //=== Setup Sensor de Vazão ================
223
224 //=== Setup Sensor Temperatura e Umidade do Ar no Gateway ===
225   dht.begin();
226 //=== Setup Sensor Temperatura e Umidade do Ar no Gateway ===
227 }
228
229 //=== Função salvar alerta sensor ===
230   void alertaSensor(String descricao, String sensor, String
231   valorexcedido){
232     MySQL_Cursor *cur_memaux = new MySQL_Cursor(&conn);
233     sprintf(sentenca, INSERIR_ALERTA_SENSOR, descricao,sensor,
234     valorexcedido);
235     Serial.println(sentenca);
236     cur_memaux->execute(sentenca);
237     delete cur_memaux;
238   }
239
240 //=== Função salvar alerta atuador ===
241   void alertaAtuador(String descricao, String atuador){
242     MySQL_Cursor *cur_memaux = new MySQL_Cursor(&conn);
243     sprintf(sentenca, INSERIR_ALERTA_ATUADOR, descricao,atuador);
244     Serial.println(sentenca);
245     cur_memaux->execute(sentenca);
246     delete cur_memaux;
247   }
248
249 // === Função para enviar email de alerta ===
250 #include <EthernetClient.h>
251 #define time 1000
252 EthernetClient emailclient;
253 byte emailserver[] = { 200, 147, 99, 132 }; //IP do servidor SMTP
254
255   void enviaemail(String msg, String valorexcedido)
256   {
```

```
257     delay(time);
258     Serial.println("conectando...");
259     if (emailclient.connect(emailserver, 587)) //Porta do servidor do email.
260     {
261     Serial.println("conectado!");
262     Serial.println("enviando email...");
263     Serial.println();
264     emailclient.println("EHLO localhost");
265     recebe();
266     delay(time);
267     emailclient.println("AUTH LOGIN");
268     recebe();
269     delay(time);
270     emailclient.println("loginembasede64");
271     // Email de login em base de 64: http://base64-encoder-online.waraxe.us/
272     recebe();
273     delay(time);
274     emailclient.println("senhaembasede64");
275     // Senha do email em base de 64: http://base64-encoder-online.waraxe.us/
276     recebe();
277     delay(time);
278     emailclient.println("mail from: <usuario@bol.com.br>"); //Remetente
279     recebe();
280     delay(time);
281     emailclient.println("rcpt to: <usuario@gmail.com>"); //Destinatário
282     recebe();
283     delay(time);
284     emailclient.println("data");
285     recebe();
286     delay(time);
287     emailclient.println("Subject: Monitoramento da Aquicultura"); // Assunto
288     recebe();
289     delay(time);
290     emailclient.println(msg+", valor excedido: "+valorexcedido); // Corpo
291     recebe();
292     delay(time);
293     emailclient.println("."); // Indica fim do email.
294     recebe();
295     delay(time);
296     emailclient.println();
297     recebe();
298     delay(time);
299     Serial.println("email enviado!");
300     delay(time);
301     if (emailclient.connected())
302     {
303     Serial.println();
304     Serial.println("desconectando...");
305     emailclient.stop();
306     Serial.println();
307     Serial.println();
308     }
```

```
309     }
310     else
311     {
312     Serial.println("connection failed");
313     }
314     Serial.println("Ready. Press 'e' to connect.");
315   }
316
317
318   void recebe()
319   {
320     while (client.available())
321     {
322       char c = client.read();
323       Serial.print(c);
324     }
325   }
326 // === Função para enviar email de alerta ===
327
328   void loop(void) {
329   MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
330
331
332 //=== Loop Módulo Tiny RTC I2C ===
333     DateTime now = RTC.now();//Recuperando a data e hora atual
334     Serial.print("Data: ");
335     Serial.print(now.day(), DEC);//Imprimindo o dia
336     Serial.print('/');
337     Serial.print(now.month(), DEC);//Recuperando o mes
338     Serial.print('/');
339     Serial.print(now.year(), DEC);//Recuperando o ano
340     Serial.print(' ');
341     Serial.print("Hora: ");
342     Serial.print(now.hour(), DEC);//Recuperando a hora
343     Serial.print(':');
344     Serial.print(now.minute(), DEC);//Reci[erando os minutos
345     Serial.print(':');
346     Serial.print(now.second(), DEC);//Recuperando os segundos
347     Serial.println();
348     //Exibe as informações na tela LCD
349     lcd.setCursor(0, 0);
350     lcd.print("======SISMAQUI======");
351     lcd.setCursor(0, 1);
352     lcd.print("Data: ");
353     lcd.print(now.day(), DEC);
354     lcd.print("/");
355     lcd.print(now.month(), DEC);//Recuperando o mes
356     lcd.print("/");
357     lcd.print(now.year(), DEC);//Recuperando o ano
358     lcd.setCursor(0, 2);
359     lcd.print("Hora: ");
360     lcd.print(now.hour(), DEC);//Recuperando a hora
```

```
361     lcd.print(':');
362     lcd.print(now.minute(), DEC);//Reci[erando os minutos
363     lcd.print(':');
364     lcd.print(now.second(), DEC);//Recuperando os segundos */
365     delay(intervalortc);
366     lcd.clear();
367 //=== Loop Módulo Tiny RTC I2C ===
368
369 //=== Loop Sensor de pH ========
370   static unsigned long samplingTime = millis();
371     static unsigned long printTime = millis();
372     static float pHValue,voltagePh;
373     if(millis()-samplingTime > samplingInterval)
374     {
375         pHArray[pHArrayIndex++]=analogRead(SensorPin);
376         if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
377         voltagePh = avergearray(pHArray, ArrayLenth)*5.0/1024;
378         pHValue = 3.5*voltagePh+Offset;
379         samplingTime=millis();
380     }
381     if(millis() - printTime > printInterval)
382     {
383       Serial.print("pH = Voltagem:");
384       Serial.print(voltagePh,2);
385       voltagemph = (float(voltagePh));
386       Serial.print(";  Valor: ");
387       Serial.println(pHValue,2);
388       valorph = (float(pHValue));
389       lcd.setCursor(0, 0);
390       lcd.print("Voltagem pH: ");
391       lcd.print(voltagePh,2);
392       delay(300);
393       lcd.setCursor(0, 1);
394       lcd.print("Valor pH: ");
395       lcd.print(pHValue,2);
396       delay(100);
397       Serial.println("============ Novas leituras ===============");
398 //          digitalWrite(LED,digitalRead(LED)^1);
399           printTime=millis();
400     }
401 //=== Loop Sensor de pH ========
402
403 //=== Loop para Sensor de Nível da Água - Ultrasônico ======
404   delay(10);
405   Serial.print("Nível da Água: ");
406   int dist = sonar.ping();
407   Serial.print(sonar.convert_cm(dist));//Converte distância em cm
408   nivelagua = sonar.convert_cm(dist);
409   Serial.println("cm");
410     if (sonar.convert_cm(dist)>20){
411       ligaOuDesligaEquipamento(porta_rele3_bomba_abastecimento,
412       "HIGH", "A bomba de abastecimento está ligada");
```

```
413     } else {
414       ligaOuDesligaEquipamento(porta_rele3_bomba_abastecimento,
415       "LOW", "A bomba de abastecimento está desligada");
416     }
417 //=== Loop Sensor de Nível da Água - Ultrasônico ========
418 
419 //=== Loop Sensor de Vazão da Recirculação ========
420   Pulso = 0; //Inicia do 0 para contar os giros das pás internas
421   sei(); //liga interrupção
422   delay (100); //Espera 2 segundos //Se reduzir trava
423   cli(); //Desliga interrupção
424   vazaoagua = Pulso / 5.5; //Converte para Litros/minuto
425   valormedia=valormedia+vazaoagua; //Soma para calcular a média
426   j++;
427   Serial.print("Vazão da recirculação: ");
428   Serial.print(vazaoagua); //Imprime na serial o valor da vazão
429   delay (100);
430   Serial.println(" L/minuto");
431   if(j==60)
432   {
433     valormedia = valormedia/60;
434     Serial.print("\n Media por minuto = ");
435     Serial.print(valormedia);
436     Serial.println(" Litros/minutos - ");
437     valormedia = 0; //Torna variável valormedia em 0
438     j=0; //Torna a variável 0,para uma nova contagem
439   }
440 //=== Loop Sensor de Vazão da Recirculação ========
441 
442 //=== Loop Sensor de Temperatura da Água ============
443     float temp[qtdeSensores];
444     int x,y,i,j;
445     //Serial.print("Lendo..\n\r");
446     sensors.requestTemperatures();
447     for(i=0;i<qtdeSensores;i++){
448       temp[i] = sensors.getTempC(sensores[i]);
449     Serial.print("Temperatura da Água ");
450         Serial.print(": ");
451           Serial.print(temp[i]);
452           lcd.setCursor(0, 0);
453           lcd.print("Temp Agua: ");
454           lcd.print(temp[i]);
455           Serial.println("°C");
456           lcd.print("C");
457           tempagua = (float(temp[i]));
458           dtostrf(tempagua, 4, 1, tempaguaconvertida);
459      if (temp[i]>30){ //Se a temperatura for maior que 30°C liga
460       ligaOuDesligaEquipamento(porta_rele2_bomba_recirculacao,
461        "HIGH",
462         enviaemail("Temperatura acima do recomendado",
463         tempaguaconvertida);
464           sprintf(sentenca, INSERIR_ALERTA_SENSOR,
465           "Temperatura acima do recomendado",
```

```
466           "Sensor de Temperatura",tempaguaconvertida);
467           Serial.println(sentenca);
468           cur_mem->execute(sentenca);
469                }else if (temp[i]<24){ //Temp menor que 24°C liga
470         ligaOuDesligaEquipamento(porta_rele2_bomba_recirculacao,
471         "HIGH",
472           //Envia email de alerta
473           enviaemail("Temperatura abaixo do recomendado",
474           tempaguaconvertida);
475           sprintf(sentenca, INSERIR_ALERTA_SENSOR,
476           "Temperatura abaixo do recomendado",
477           "Sensor de Temperatura",tempaguaconvertida);
478           Serial.println(sentenca);
479           cur_mem->execute(sentenca);
480       }else{
481        ligaOuDesligaEquipamento(porta_rele2_bomba_recirculacao,
482        "LOW",
483       }
484   }
485   delay(10);
486 //=== Loop Sensor de Temperatura da Água ================
487
488   delay(100); //Não alterar
489
490 //=== Loop Sensor de Turbidez ========
491   int sensorValue = analogRead(A0);
492   float voltage = sensorValue * (5.0 / 1024.0);
493   Serial.print("Turbidez:");
494   Serial.print(voltage);
495   lcd.setCursor(0, 1);
496   lcd.print("Turbidez: ");
497   lcd.print(voltage);
498   Serial.println("V");
499   lcd.print("V");
500   delay(100); //Se reduzir trava
501 //=== Loop Sensor de Turbidez ========
502
503 //=== Loop Sensor de Temperatura e Humidade do Ar ========
504     humi = float(dht.readHumidity());
505     tempe = float(dht.readTemperature());
506     Serial.print("Umidade Interna: ");
507     Serial.print(humi);
508     Serial.print(" %, Temperatura Interna: ");
509     Serial.print(tempe);
510     Serial.println(" Celsius");
511     lcd.setCursor(0, 2);
512     lcd.print("Umid Gateway: ");
513     lcd.print(humi);
514     lcd.setCursor(0, 3);
515     lcd.print("Temp Gateway: ");
516     lcd.print(tempe);
517     delay(2000);
518     lcd.clear();
519     delay(500);
520 //==== Loop Sensor de Temperatura e Humidade do Ar =========
```

```
//============ Loop MySQL Connector ========================
   Serial.println("Executando sentença");
   int idia = now.day();
   int imes = now.month();
   int iano = now.year();
   int ihoras = now.hour();
   int iminutos = now.minute();
   int isegundos = now.second();
   leitura = analogRead(LM35);
   leituraconvertida = (float(analogRead(LM35))*5/(1023))/0.01;
   dtostrf(leituraconvertida, 4, 1, valortemp);
   dtostrf(tempagua, 4, 1, tempaguaconvertida);
   turbidez = (float(voltage));
   dtostrf(turbidez, 4, 1, turbidezconvertida);
   tempinterna = (float(tempe));
   dtostrf(tempinterna, 4, 1, tempinternaconvertida);
   umidainterna = (float(humi));
   dtostrf(umidainterna, 4, 1, umidainternaconvertida);
   dtostrf(nivelagua, 4, 1, nivelaguaconvertida);
   dtostrf(valorph, 4, 1, valorphconvertida);
   dtostrf(voltagemph, 4, 1, voltagemphconvertida);
   sprintf(sentenca, INSERIR_TEMP, tempaguaconvertida);
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   sprintf(sentenca, INSERIR_TURB, turbidezconvertida);
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   sprintf(sentenca, INSERIR_UMID_TEMP_INTERNA,
   umidainternaconvertida,tempinternaconvertida);
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   sprintf(sentenca, INSERIR_PH, valorphconvertida,
   voltagemphconvertida);
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   sprintf(sentenca, INSERIR_NIVEL, nivelaguaconvertida);
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   sprintf(sentenca, INSERIR_ALERTA_ATUADOR,
   "O nivel de racao esta baixo","Alimentador");
   Serial.println(sentenca);
   cur_mem->execute(sentenca);
   delete cur_mem;
   delay(10000);
  //============ Loop MySQL Connector ========================

} //=== Fim do Loop Geral ===

//=== Função para arredondamento do valor das leituras do pH ====
  double mediadoarray(int* arr, int number){
    int i;
    int max,min;
    double avg;
    long amount=0;
```

```
575     if(number<=0){
576       Serial.println("Número inválido para cálculo da média!/n");
577       return 0;
578     }
579     if(number<5){
580       for(i=0;i<number;i++){
581         amount+=arr[i];
582       }
583       avg = amount/number;
584       return avg;
585     }else{
586       if(arr[0]<arr[1]){
587         min = arr[0];max=arr[1];
588       }
589       else{
590         min=arr[1];max=arr[0];
591       }
592       for(i=2;i<number;i++){
593         if(arr[i]<min){
594           amount+=min;         //para array menor que min
595           min=arr[i];
596         }else {
597           if(arr[i]>max){
598             amount+=max;     //para array menor que min
599             max=arr[i];
600           }else{
601             amount+=arr[i]; //para min<=arra<=max
602           }
603         }//if
604       }//for
605       avg = (double)amount/(number-2);
606     }//if
607     return avg;
608   }
609
610 //=== Função para incremento de pulsos no sensor de vazão ===
611   void incrpulso ()
612   {
613     Pulso++;
614   }
```

ANEXO D - Exemplos de códigos da aplicação web

Códigos da classe Sensor

```
1   package br.com.itbs.itbusinessSchool.domain;
2   import java.io.Serializable;
3   import javax.faces.bean.ViewScoped;
4   import javax.persistence.Entity;
5   import javax.persistence.GeneratedValue;
6   import javax.persistence.GenerationType;
7   import javax.persistence.Id;
8   import javax.persistence.JoinColumn;
9   import javax.persistence.ManyToOne;
10  import javax.persistence.Table;
11
12  @Entity
13  @Table
14  @ViewScoped
15  public class Sensor implements Serializable {
16      @Id
17      @GeneratedValue(strategy = GenerationType.AUTO)
18      private Long id;
19      private String nome;
20      private String descricao;
21      private Long limiteSuperior;
22      private Long limiteInferior;
23
24      @ManyToOne @JoinColumn(name="idLocalInstalacao")
25      private LocalInstalacao localInstalacao;
26
27      //================================ Construtores ================================
28      public Sensor() {
29      }
30
31      public Sensor(Long id, String nome, String descricao, Long limiteSuperior,
32              Long limiteInferior, LocalInstalacao localInstalacao) {
33          this.id = id;
34          this.nome = nome;
35          this.descricao = descricao;
36          this.limiteSuperior = limiteSuperior;
37          this.limiteInferior = limiteInferior;
38          this.localInstalacao = localInstalacao;
39      }
40
41      //============================ Métodos sobrescritos ============================
42      @Override
44      public int hashCode() {
44          int hash = 0;
45          hash += (id != null ? id.hashCode() : 0);
46          return hash;
47      }
48
49      @Override
51      public boolean equals(Object object) {
51          if (!(object instanceof Sensor)) {
52              return false;
53          }
54          Sensor other = (Sensor) object;
56          if ((this.id == null && other.id != null) || (this.id != null &&
                  !this.id.equals(other.id))) {
```

```
57              return false;
58          }
59          return true;
60      }
61
62      @Override
63      public String toString() {
64          return "br.coop.integrada.testenetbeans4.dao.Sensor[ id=" + id + " ]";
65      }
66
67      //============================== Getters e Setters ==============================
68      public Long getId() {
69          return id;
70      }
71
72      public void setId(Long id) {
73          this.id = id;
74      }
75
76      public String getNome() {
77          return nome;
78      }
79
80      public void setNome(String nome) {
81          this.nome = nome;
82      }
83
84      public String getDescricao() {
85          return descricao;
86      }
87
88      public void setDescricao(String descricao) {
89          this.descricao = descricao;
90      }
91
92      public Long getLimiteSuperior() {
93          return limiteSuperior;
94      }
95
96      public void setLimiteSuperior(Long limiteSuperior) {
97          this.limiteSuperior = limiteSuperior;
98      }
100     public Long getLimiteInferior() {
101         return limiteInferior;
102     }
103
104     public void setLimiteInferior(Long limiteInferior) {
105         this.limiteInferior = limiteInferior;
106     }
107
108     public LocalInstalacao getLocalInstalacao() {
109         return localInstalacao;
110     }
111
112     public void setLocalInstalacao(LocalInstalacao localInstalacao) {
113         this.localInstalacao = localInstalacao;
114     }
115 }
```

Códigos da classe GenericDAO

```java
package br.com.itbs.itbusinessSchool.dao;
import java.io.Serializable;
import java.lang.reflect.ParameterizedType;
import java.sql.Connection;
import java.sql.SQLException;
import javax.annotation.Resource;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.sql.DataSource;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Order;
import javax.persistence.EntityManagerFactory;
import java.util.List;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

public abstract class GenericDAO<T> implements Serializable {

    @PersistenceContext(unitName="TesteNetbeansPU")
    protected EntityManager entityManager;

    @Resource(mappedName="jdbc/netbeansteste")
    DataSource dataSource;


    public GenericDAO() {
        this.persistentClass = (Class<T>) ((ParameterizedType)
                getClass().getGenericSuperclass()).getActualTypeArguments()[0];
    }

    private Class<T> persistentClass;

    public Class<T> getPersistentClass() {
        return this.persistentClass;
    }


    public void excluir(T entity) {
        entity = entityManager.merge(entity);
        entityManager.remove(entity);
    }

    public T buscarPorId(int id) {
        try {
            return (T) entityManager.find(getPersistentClass(), id);
        } catch (NoResultException noResultException) {
            return null;
        }
    }

    public T buscarUmPorCampoCompleto(String campo, Object valor) {
        try {
            return (T) entityManager.createQuery("SELECT obj FROM " +
                    persistentClass.getName() + " obj "
                    + "FETCH ALL PROPERTIES "
                    + "WHERE " + campo + " = :_valor ")
```

```
61                      .setParameter("_valor", valor)
62                      .setMaxResults(1)
63                      .getSingleResult();
64              } catch (NoResultException noResultException) {
65                  return null;
66              }
67          }
68
69          public List<T> buscarTodos() {
70              return getSession().createCriteria(persistentClass)
71              .setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY).list();
72          }
73
74          public List<T> buscarTodosPorObjetoCompleto(Long id, Object objeto) {
75             EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpa");
76             EntityManager em = emf.createEntityManager();
77             EntityTransaction entr = em.getTransaction();
78             entr.begin();
79             Query query = em.createNativeQuery("SELECT * FROM T WHERE " + id +
80                     " = :_objeto ").setParameter("_objeto", objeto);
81              List stList = query.getResultList();
82              return query.getResultList();
83          }
84
85          public T create(T entity) {
86              entity = entityManager.merge(entity);
87              entityManager.createQuery(entity.toString());
88              return entity;
89          }
90
91          public T salvar(T entity) {
92              entity = entityManager.merge(entity);
93              return entity;
94          }
95
96          public List<T> buscarListaPorCriterio(Criterion... criterion) {
97              Criteria crit = getSession().createCriteria(getPersistentClass());
98              for (Criterion c : criterion) {
99                  crit.add(c);
100             }
101             crit.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
102             return crit.list();
103         }
104
105         public List<T> buscarListaPorCriterio(Order order, Criterion... criterion) {
106             Criteria crit = getSession().createCriteria(getPersistentClass());
107             for (Criterion c : criterion) {
108                 crit.add(c);
109             }
110             crit.addOrder(order);
111             crit.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
112             return crit.list();
113         }
114
115         public T buscarUmPorCriterio(Criterion... criterion) {
116             Criteria crit = getSession().createCriteria(getPersistentClass());
117             for (Criterion c : criterion) {
118                 crit.add(c);
119             }
120             return (T) crit.uniqueResult();
121         }
122
```

```
123      public T buscarUmPorValorMaior(Criterion... criterion) {
124          Criteria crit = getSession().createCriteria(getPersistentClass());
125          for (Criterion c : criterion) {
126              crit.add(c);
127          }
128          return (T) crit.uniqueResult();
129      }
130
131      public Session getSession() {
132         return (Session) entityManager.unwrap(Session.class);
133      }
134
135      public Connection getConnection() {
136          try {
137              return dataSource.getConnection();
138          } catch (SQLException ex) {
139              return null;
140          }
141      }
142  }
```

Códigos da classe SensorDAO

```
1   package br.com.itbs.itbusinessSchool.dao;
2   import br.com.itbs.itbusinessSchool.domain.Sensor;
3
4   public class SensorDAO extends GenericDAO<Sensor>{
5
6   }
```

Códigos da classe GenericBO

```
1   package br.com.itbs.itbusinessSchool.bo;
2   import br.com.itbs.itbusinessSchool.dao.GenericDAO;
3   import java.io.Serializable;
4   import java.util.List;
5   import javax.annotation.PostConstruct;
6   import org.hibernate.criterion.Criterion;
7   import org.hibernate.criterion.Order;
8
9   public abstract class GenericBO<E> implements Serializable {
10
11      protected GenericDAO<E> genericDAO;
12
13      @PostConstruct
14      protected abstract void inicializar();
15
16      public E salvar(E entidade) {
17          return genericDAO.salvar(entidade);
18      }
19
20      public void excluir(E entidade) {
21          genericDAO.excluir(entidade);
22      }
23
24      public List<E> buscarTodos(String ordenarPor) {
25          return genericDAO.buscarListaPorCriterio(Order.asc(ordenarPor));
26      }
27
28      public List<E> buscarTodos() {
29          return genericDAO.buscarTodos();
```

```
30      }
31
32      public E buscarPorId(int id) {
33          return (E)genericDAO.buscarPorId(id);
34      }
35
36      public E buscarUmPorCampoCompleto(String campo, Object valor) {
37          return (E)genericDAO.buscarUmPorCampoCompleto(campo, valor);
38      }
39
40      public List<E> buscarListaPorCriterio(Criterion... criterion) {
41          return genericDAO.buscarListaPorCriterio(criterion);
42      }
43
44      public List<E> buscarListaPorCriterio(Order order, Criterion... criterion){
45          return genericDAO.buscarListaPorCriterio(order, criterion);
46      }
47
48      public E buscarUmPorCriterio(Criterion... criterion) {
49          return genericDAO.buscarUmPorCriterio(criterion);
50      }
51  }
```

Códigos da Classe SensorBO

```
1   package br.com.itbs.itbusinessSchool.bo;
2   import br.com.itbs.itbusinessSchool.dao.SensorDAO;
3   import br.com.itbs.itbusinessSchool.domain.Sensor;
4   import javax.annotation.PostConstruct;
5   import javax.ejb.Stateless;
6   import javax.inject.Inject;
7
8   @Stateless
9   public class SensorBO extends GenericBO<Sensor>{
10
11      @Inject
12      private SensorDAO sensorDAO;
13
14      @Override
15      @PostConstruct
17      protected void inicializar() {
18          this.genericDAO = sensorDAO;
19      }
20  }
```

Códigos da classe SensorController

```
1   package br.com.itbs.itbusinessSchool.view;
2   import br.com.itbs.itbusinessSchool.domain.Sensor;
3   import java.lang.reflect.InvocationTargetException;
4   import java.lang.reflect.Method;
5   import javax.faces.FacesException;
6   import javax.annotation.Resource;
7   import javax.transaction.UserTransaction;
8   import br.com.itbs.itbusinessSchool.view.util.JsfUtil;
9   import br.com.itbs.itbusinessSchool.view.util.PagingInfo;
10  import java.util.List;
11  import javax.faces.component.UIComponent;
12  import javax.faces.context.FacesContext;
13  import javax.faces.convert.Converter;
14  import javax.faces.model.SelectItem;
15  import javax.persistence.EntityManagerFactory;
16  import javax.persistence.PersistenceUnit;
17
18  public class SensorController {
```

```
19
20      public SensorController() {
21          pagingInfo = new PagingInfo();
22          converter = new SensorConverter();
23      }
24      private Sensor sensor;
25      private List<Sensor> sensorItems = null;
26      private SensorFacade jpaController = null;
27      private SensorConverter converter = null;
28      private PagingInfo pagingInfo = null;
29      @Resource
        private UserTransaction utx = null;
31      @PersistenceUnit(unitName = "TesteNetbeansPU")
        private EntityManagerFactory emf = null;
33
34      public PagingInfo getPagingInfo() {
35          if (pagingInfo.getItemCount() == -1) {
36              pagingInfo.setItemCount(getJpaController().count());
37          }
38          return pagingInfo;
39      }
40
41      public SensorFacade getJpaController() {
42          if (jpaController == null) {
43              FacesContext facesContext = FacesContext.getCurrentInstance();
44              jpaController = (SensorFacade) facesContext.getApplication().
45                      getELResolver().getValue(facesContext.getELContext(), null,
46                              "sensorJpa");
47          }
48          return jpaController;
49      }
50
51      public SelectItem[] getSensorItemsAvailableSelectMany() {
52          return JsfUtil.getSelectItems(getJpaController().findAll(), false);
53      }
54
55      public SelectItem[] getSensorItemsAvailableSelectOne() {
56          return JsfUtil.getSelectItems(getJpaController().findAll(), true);
57      }
58
59      public Sensor getSensor() {
60          if (sensor == null) {
61              sensor = (Sensor) JsfUtil.getObjectFromRequestParameter
62              ("jsfcrud.currentSensor", converter, null);
63          }
64          if (sensor == null) {
65              sensor = new Sensor();
66          }
67          return sensor;
68      }
69
70      public String listSetup() {
71          reset(true);
72          return "sensor_list";
73      }
74
75      public String createSetup() {
76          reset(false);
77          sensor = new Sensor();
78          return "sensor_create";
79      }
80
81      public String create() {
```

```
82          try {
83              utx.begin();
84          } catch (Exception ex) {
85          }
86          try {
87              Exception transactionException = null;
88              getJpaController().create(sensor);
89              try {
90                  utx.commit();
91              } catch (javax.transaction.RollbackException ex) {
92                  transactionException = ex;
93              } catch (Exception ex) {
94              }
95              if (transactionException == null) {
96                  JsfUtil.addSuccessMessage("O sensor foi cadastrado com "
97                          + "sucesso.");
98              } else {
99                  JsfUtil.ensureAddErrorMessage(transactionException, "Ocorreu "
100                         + "um erro ao tentar salvar os dados.");
101             }
102         } catch (Exception e) {
103             try {
104                 utx.rollback();
105             } catch (Exception ex) {
106             }
107             JsfUtil.ensureAddErrorMessage(e, "Ocorreu um erro ao tentar salvar"
108                     + " os dados.");
109             return null;
110         }
111         return listSetup();
112     }
113
114     public String detailSetup() {
115         return scalarSetup("sensor_detail");
116     }
117
118     public String editSetup() {
119         return scalarSetup("sensor_edit");
120     }
121
122     private String scalarSetup(String destination) {
123         reset(false);
124         sensor = (Sensor) JsfUtil.getObjectFromRequestParameter("jsfcrud."
125                 + "currentSensor", converter, null);
126         if (sensor == null) {
127             String requestSensorString = JsfUtil.getRequestParameter("jsfcrud."
128                     + "currentSensor");
129             JsfUtil.addErrorMessage("O sensor com a id " + requestSensorString
130                     + " não existe.");
131             return relatedOrListOutcome();
132         }
133         return destination;
134     }
135
136     public String edit() {
137         String sensorString = converter.getAsString(FacesContext.
138                 getCurrentInstance(), null, sensor);
139         String currentSensorString = JsfUtil.getRequestParameter("jsfcrud."
140                 + "currentSensor");
141         if (sensorString == null || sensorString.length() == 0 || !sensorString
142                 .equals(currentSensorString)) {
143             String outcome = editSetup();
```

```
144             if ("sensor_edit".equals(outcome)) {
145                 JsfUtil.addErrorMessage("Não foi possível editar o sensor. "
146                         + "Tente novamente.");
147             }
148             return outcome;
149         }
150         try {
151             utx.begin();
152         } catch (Exception ex) {
153         }
154         try {
155             Exception transactionException = null;
156             getJpaController().edit(sensor);
157             try {
158                 utx.commit();
159             } catch (javax.transaction.RollbackException ex) {
160                 transactionException = ex;
161             } catch (Exception ex) {
162             }
163             if (transactionException == null) {
164                 JsfUtil.addSuccessMessage("O sensor foi alterado com sucesso.");
165             } else {
166                 JsfUtil.ensureAddErrorMessage(transactionException, "Ocorreu "
167                         + "um erro ao tentar salvar os dados.");
168             }
169         } catch (Exception e) {
170             try {
171                 utx.rollback();
172             } catch (Exception ex) {
173             }
174             JsfUtil.ensureAddErrorMessage(e, "Ocorreu um erro ao tentar salvar"
175                     + " os dados.");
176             return null;
177         }
178         return detailSetup();
179     }
180 
181     public String remove() {
182         String idAsString = JsfUtil.getRequestParameter("jsfcrud."
183                 + "currentSensor");
184         Long id = new Long(idAsString);
185         try {
186             utx.begin();
187         } catch (Exception ex) {
188         }
189         try {
190             Exception transactionException = null;
191             getJpaController().remove(getJpaController().find(id));
192             try {
193                 utx.commit();
194             } catch (javax.transaction.RollbackException ex) {
195                 transactionException = ex;
196             } catch (Exception ex) {
197             }
198             if (transactionException == null) {
199                 JsfUtil.addSuccessMessage("O sensor foi removido com sucesso.");
200             } else {
201                 JsfUtil.ensureAddErrorMessage(transactionException, "Ocorreu "
202                         + "um erro ao tentar salvar os dados.");
203             }
204         } catch (Exception e) {
205             try {
206                 utx.rollback();
```

```
            } catch (Exception ex) {
            }
            JsfUtil.ensureAddErrorMessage(e, "Ocorreu um erro ao tentar salvar"
                    + " os dados.");
            return null;
        }
        return relatedOrListOutcome();
    }

    private String relatedOrListOutcome() {
        String relatedControllerOutcome = relatedControllerOutcome();
        if (relatedControllerOutcome != null) {
            return relatedControllerOutcome;
        }
        return listSetup();
    }

    public List<Sensor> getSensorItems() {
        if (sensorItems == null) {
            getPagingInfo();
            sensorItems = getJpaController().findRange(new int[]{pagingInfo.
                    getFirstItem(), pagingInfo.getFirstItem() + pagingInfo.
                            getBatchSize()});
        }
        return sensorItems;
    }

    public String next() {
        reset(false);
        getPagingInfo().nextPage();
        return "sensor_list";
    }

    public String prev() {
        reset(false);
        getPagingInfo().previousPage();
        return "sensor_list";
    }

    private String relatedControllerOutcome() {
        String relatedControllerString = JsfUtil.getRequestParameter("jsfcrud."
                + "relatedController");
        String relatedControllerTypeString = JsfUtil.getRequestParameter(""
                + "jsfcrud.relatedControllerType");
        if (relatedControllerString != null && relatedControllerTypeString !=
                null) {
            FacesContext context = FacesContext.getCurrentInstance();
            Object relatedController = context.getApplication().getELResolver()
                    .getValue(context.getELContext(), null,
                            relatedControllerString);
            try {
                Class<?> relatedControllerType = Class.forName(
                        relatedControllerTypeString);
                Method detailSetupMethod = relatedControllerType.getMethod(""
                        + "detailSetup");
                return (String) detailSetupMethod.invoke(relatedController);
            } catch (ClassNotFoundException e) {
                throw new FacesException(e);
            } catch (NoSuchMethodException e) {
                throw new FacesException(e);
            } catch (IllegalAccessException e) {
                throw new FacesException(e);
```

```
269                 } catch (InvocationTargetException e) {
270                     throw new FacesException(e);
271                 }
272             }
273             return null;
274         }
275 
276         private void reset(boolean resetFirstItem) {
277             sensor = null;
278             sensorItems = null;
279             pagingInfo.setItemCount(-1);
280             if (resetFirstItem) {
281                 pagingInfo.setFirstItem(0);
282             }
283         }
284 
285         public void validateCreate(FacesContext facesContext, UIComponent component
286                 , Object value) {
287             Sensor newSensor = new Sensor();
288             String newSensorString = converter.getAsString(FacesContext.
289                     getCurrentInstance(), null, newSensor);
290             String sensorString = converter.getAsString(FacesContext.
291                     getCurrentInstance(), null, sensor);
292             if (!newSensorString.equals(sensorString)) {
293                 createSetup();
294             }
295         }
296 
297         public Converter getConverter() {
298             return converter;
299         }
300     }
```

Códigos da classe SensorFacade

```
1   package br.com.itbs.itbusinessSchool.view;
2   import br.com.itbs.itbusinessSchool.domain.Sensor;
3   import javax.ejb.Stateless;
4   import javax.persistence.EntityManager;
5   import javax.persistence.PersistenceContext;
6 
7   @Stateless
8   public class SensorFacade extends AbstractFacade<Sensor> {
9       @PersistenceContext(unitName = "TesteNetbeansPU")
10      private EntityManager em;
11 
12      @Override
13      protected EntityManager getEntityManager() {
14          return em;
15      }
16 
17      public SensorFacade() {
18          super(Sensor.class);
19      }
20  }
```

Códigos da classe SensorConverter

```
1   package br.com.itbs.itbusinessSchool.view;
2   import br.com.itbs.itbusinessSchool.domain.Sensor;
3   import javax.faces.component.UIComponent;
4   import javax.faces.context.FacesContext;
5   import javax.faces.convert.Converter;
```

```
6
7   public class SensorConverter implements Converter {
8
9       public Object getAsObject(FacesContext facesContext, UIComponent component,
10                                String string) {
11          if (string == null || string.length() == 0) {
12              return null;
13          }
14          Long id = new Long(string);
15          SensorController controller = (SensorController)
16                  facesContext.getApplication().getELResolver().getValue
17          (facesContext.getELContext(), null, "sensor");
18          return controller.getJpaController().find(id);
19      }
20
21      public String getAsString(FacesContext facesContext, UIComponent component,
22                                Object object) {
23          if (object == null) {
24              return null;
25          }
26          if (object instanceof Sensor) {
27              Sensor o = (Sensor) object;
28              return o.getId() == null ? "" : o.getId().toString();
29          } else {
30              throw new IllegalArgumentException("O objeto " + object +
31                 " é um tipo " + object.getClass().getName() +
32                 "; tipo esperado: br.com.itbs.itbusinessSchool.domain.Sensor");
33          }
34      }
35  }
```

Exemplos de códigos da interface

Códigos da página de listagem dos sensores cadastrados

```
1   <!DOCTYPE html>
2   <html xmlns="http://www.w3.org/1999/xhtml"
3         xmlns:f="http://xmlns.jcp.org/jsf/core"
4         xmlns:h="http://xmlns.jcp.org/jsf/html"
5         xmlns:p="http://xmlns.jcp.org/jsf/passthrough"
6         xmlns:jsf="http://xmlns.jcp.org/jsf"
7         xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
8         xmlns:t="http://myfaces.apache.org/tomahawk"
9         xmlns:c="http://java.sun.com/jsp/jstl/core"
10        class="identity-ui"
11        xmlns:pr="http://primefaces.org/ui">
12      <h:head>
13          <script type="text/javascript" src="jquery.js"></script>
14          <script type="text/javascript"></script>
15          <link rel="shortcut icon" href="fishico.png"> </link>
16          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
17
18          </meta>
19          <title>SISMAQUI</title>
20          <link rel="stylesheet" type="text/css"
21                href="/ITBusinessSchool/faces/styles.css" />
22      </h:head>
23
24      <h:body>
25          <TABLE width="100%" height="660" BORDER="0" CELLSPACING="0">
```

```
26          <TR bordercolor="#CD0000" style="height: 100%; width: 99%;">
27              <TD width="99%" bgcolor="#CD0000" height="25" COLSPAN="2">
28                  <ui:include src="/header2.xhtml" /></TD>
29          </TR>
30
31          <TR bordercolor="#FFFFFF">
32              <TD width="100%" bgcolor="#FFFFFF" height="70"
33                  style="background-color: #0000FF;">
34                  <ui:include src="/menublueadmin.xhtml" />
35              </TD>
36          </TR>
37
38          <TR style="height: 805px; width: 99%; border-left-color: #000000;">
39              <td width="100%" bgcolor="#FFFFFF" COLSPAN="1">
40                  <pr:panel id="basic"
41                            header="Listagem de sensores cadastrados"
                              style="margin-bottom:1px; width: 99,20%;
43                            height: 800px; border-color: white;">
44                      <h:panelGroup id="messagePanel" layout="block">
45                          <h:messages errorStyle="color: red"
46                                      infoStyle="color: green"
47                                      layout="table"/>
48                      </h:panelGroup>
49                      <h:panelGrid columns="2" cellpadding="7" >
50                          <h:form>
51                              <pr:dataTable value="#{sensor.sensorItems}"
52                                    var="item" widgetVar="sensorTable"
53                                    paginator="true" rows="8"
54                                    paginatorPosition="bottom"
55                                    style="width: 1950px;"
56                                    emptyMessage="Nenhum registro encontrado">
57                                  <f:facet name="header">
58                                      <pr:outputPanel>
59                                         <h:outputText
60                                             value="Localizar por nome:" />
61                                          <pr:inputText id="globalFilter"
62                                             onkeyup="PF('sensorTable').
63                                             filter()" style="width:150px"
64                                             placeholder="Nome"/>
65                                      </pr:outputPanel>
66                                  </f:facet>
67
68                                  <pr:column headerText="Nome"
69                                             sortBy="#{item.nome}" >
70                                      <h:outputText value="#{item.nome}" />
                                    </pr:column>
72
73                                  <pr:column headerText="Descrição">
74                                    <h:outputText value="#{item.descricao}"/>
75                                  </pr:column>
76
77                                  <pr:column headerText="Limite superior" >
78                                    <h:outputText
79                                        value="#{item.limiteSuperior}" />
80                                  </pr:column>
81
82                                  <pr:column headerText="Limite inferior" >
83                                   <h:outputText
84                                       value="#{item.limiteInferior}" />
85                                  </pr:column>
86
87                                  <pr:column headerText="Local de instalação">
88                                      <h:outputText
```

```
                                        value="#{item.localInstalacao.
                                                descricao}"/>
                                    <h:panelGroup
                                        rendered="#{item.localInstalacao
                                                != null}">
                                    </h:panelGroup>
                                </pr:column>

                                <pr:column headerText="Ações" width="110">
                                    <h:outputText value="  "/>
                                    <h:commandLink value="    "
                                                action="#{sensor.
                                                        detailSetup}">
                                        <h:graphicImage value="/resources/
                                                imagensLegais/detail2.png"/>
                                        <f:param name="jsfcrud.
                                        currentSensor" value="#{jsfcrud_class
                                       ['br.com.itbs.itbusinessSchool.view.util
                                        .JsfUtil'].jsfcrud_method['
                                        getAsConvertedString'][item][sensor.
                                        converter].jsfcrud_invoke}"/>
                                    </h:commandLink>
                                    <h:outputText value="  "/>
                                    <h:commandLink value="     " action="#
                                                {sensor.editSetup}">
                                        <h:graphicImage value="/resources/
                                     imagensLegais/edit2.png"/>
                                      <f:param name="jsfcrud.currentSensor"
                                     value="#{jsfcrud_class['br.com.itbs.
                                      itbusinessSchool.view.util.JsfUtil'].
                                      jsfcrud_method['getAsConvertedString'
                                ][item][sensor.converter].jsfcrud_invoke}"/>
                                    </h:commandLink>
                                    <h:outputText value="  "/>
                                    <h:commandLink value="
                                      "action="#{sensor.remove}" >
                                        <h:graphicImage value="/resources/
                                                imagensLegais/remove2.png" />
                                      <f:param name="jsfcrud.currentSensor"
                                      value="#{jsfcrud_class['br.com.itbs.
                                     itbusinessSchool.view.util.JsfUtil'].
                                     jsfcrud_method['getAsConvertedString']
                                      [item][sensor.converter].
                                      jsfcrud_invoke}"/>
                                    </h:commandLink>
                                </pr:column>
                            </pr:dataTable>
                            <div>
                            </div>
                            <div id="buttonbar" >
                                <p>
                                    <form jsf:id="form5">
                                        <h:commandButton value="#{msg.novo}"
                        action="cadastrar" class="SantosBlueAquaButtonHover"/>
                                    </form>
                                </p>
                            </div>
                        </h:form>
                    </h:panelGrid>

                </pr:panel>
            </td>
```

```
            </TR>
        </TABLE>
    </h:body>
</html>
```

Códigos da página de cadastro de sensores

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://xmlns.jcp.org/jsf/passthrough"
      xmlns:jsf="http://xmlns.jcp.org/jsf"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:t="http://myfaces.apache.org/tomahawk"
      xmlns:c="http://java.sun.com/jsp/jstl/core"
      class="identity-ui"
      xmlns:pr="http://primefaces.org/ui">

    <h:head>
        <script type="text/javascript" src="jquery.js"></script>
        <script type="text/javascript"></script>
        <link rel="shortcut icon" href="fishico.png"> </link>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        </meta>
        <title>SISMAQUI</title>
        <link rel="stylesheet" type="text/css" href="/ITBusinessSchool/faces/
              styles.css" />
    </h:head>

    <h:body>
        <TABLE width="100%" height="660" BORDER="0" CELLSPACING="0">
            <TR bordercolor="#CD0000">
                <TD width="700" bgcolor="#CD0000" height="25" COLSPAN="2">
                    <ui:include src="/header2.xhtml" /></TD>
            </TR>

            <TR bordercolor="#FFFFFF">
                <TD width="100%" bgcolor="#FFFFFF" height="70"
                    style="background-color: #0000FF;">
                    <ui:include src="/menublueadmin.xhtml" />
                </TD>
            </TR>

            <TR bordercolor="#CD0000">
                <td width="100%" bgcolor="#FFFFFF" COLSPAN="1">
                    <div style="color:#000000"> <strong> </strong> </div>
                    <h:form>
                    <h:outputScript name="locale-primefaces.js" library="js" />
                        <pr:panel id="basic" header="Cadastro de Sensor"
                        style="margin-bottom:1px; width: 99,20%; height: 800px;
                        border-color: white;">

                                <h:panelGrid columns="3" cellpadding="5">
                                  <h:outputLabel value="#{msg.usuarioNome}"
                                  for="nome"/>
                                  <pr:inputText id="nome"
                                  value="#{sensor.sensor.nome}"
                                 required="true" label="#{msg.usuarioNome}"
                                  style="width: 300px;" requiredMessage="
                                  #{msg.informenomeusuario}"/>
                                    <pr:message for="nome" />
                                    <h:outputLabel value="Descrição"
                                    for="descricao"/>
```

```
58                                  <pr:inputText id="descricao" type="text"
59                                       value="#{sensor.sensor.descricao}"
60                                      required="true" label="#{msg.login}"
61                                       style="width: 300px"
62                                  requiredMessage="#{msg.informeloginusuario}"/>
63                                  <pr:message for="descricao" />
64                                  <h:outputLabel value="Limite superior"
65                                   for="limiteSuperior"/>
66                                  <pr:inputText id="limiteSuperior"
67                                   type="text" value="#{sensor.sensor.
68                                   limiteSuperior}" required="true"
69                                   label="#{msg.senha}" style="width:
70                                   300px" requiredMessage="#{msg.
71                                   informesenhausuario}" />
72                                  <pr:message for ="limiteSuperior" />
73                                  <h:outputLabel value="Limite inferior"
74                                    for="limiteInferior"/>
75                                  <pr:inputText id="limiteInferior"
76                                  type="text" value="#{sensor.sensor.
77                                  limiteInferior}" required="true"
78                                 label="#{msg.senha}" style="width: 300px"
79                                  requiredMessage="#{msg.
80                                   informesenhausuario}" />
81                                  <pr:message for ="limiteInferior" />
82                                  <h:outputLabel value="#{msg.local}"
83                                                for="local"/>
85                                   <pr:selectOneListbox id="local"
86                                  binding=""value="#{sensor.sensor.
87                                  localInstalacao}" var="l"
88                                  scrollHeight="123" style="width: 340px"
89                                  required="true" label="#{msg.local}"
90                                  requiredMessage="
91                                  #{msg.informelocalusuario}">
92                                      <pr:column width="25">
93                                          <pr:graphicImage
94                                           value="/resources/bandeiras/#
95                                  {l.bandeira}" width="25" height="20"/>
96                                      </pr:column>
97                                      <pr:column>
98                                          #{l.descricao} - #{l.sigla}
99                                      </pr:column>
100                                     <f:selectItems value="
101                                 #{localInstalacao.localInstalacaoItems}"
102                                     var ="local"
104                                     itemValue ="#{local}"
105                                     itemLabel ="#{local.descricao}">
106                                     </f:selectItems>
107                                 </pr:selectOneListbox>
108                                 <pr:message for="local" />
109
110                                 <div id="buttonbar">
111                                     <p>
112                                         <h:form >
113                          <h:commandButton class="SantosBlueAquaButtonHover"
114                         accesskey=""value="#{msg.salvar}"
115                         action="#{sensor.create}"
116                         actionListener="#{captchaView.submit}" >
117                                                 </h:commandButton>
118                                         </h:form>
119                                     </p>
120                                 </div>
                                </h:panelGrid>
                            </pr:panel>
```

```
121                                 <div>
122                                 </div>
123                             </h:form>
124                         </td>
125                     </TR>
126                 </TABLE>
127             </h:body>
128         </html>
```