UNIVERSIDADE FEDERAL DE SANTA CATARINA

RICARDO DO NASCIMENTO BOING

GERENCIAMENTO AUTONÔMICO DA ALIMENTAÇÃO DE FRANGOS EM AVIÁRIOS DE CRIAÇÃO ALTERNATIVA COM FOG E IOT

FLORIANÓPOLIS

2019 / 1

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

GERENCIAMENTO AUTONÔMICO DA ALIMENTAÇÃO DE FRANGOS EM AVIÁRIOS DE CRIAÇÃO ALTERNATIVA COM FOG E IOT

RICARDO DO NASCIMENTO BOING

Trabalho de Conclusão de Curso de Graduação em Ciências da Computação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Orientador: Dr. Carlos Becker Westphall

Co-orientador: Me. Hugo Vaz Sampaio

FLORIANÓPOLIS
2019 / 1

RICARDO DO NASCIMENTO BOING

GERENCIAMENTO AUTONÔMICO DA ALIMENTAÇÃO DE FRANGOS EM AVIÁRIOS DE CRIAÇÃO ALTERNATIVA COM FOG E IOT

Trabalho de Conclusão de Curso de Graduação em Ciências da Computação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Orientador:

_____

Dr. Carlos Becker Westphall

Co-orientador:

_____

Me. Hugo Vaz Sampaio

Banca examinadora:

_____

Dr. Carla Merkle Westphall

_____

Me. Leandro Loffi

Dedico este trabalho a meus pais, Nazir Pedro Boing e Cristiane do Nascimento.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me iluminar e abençoar durante todos esses anos.

A meus pais, Nazir Pedro Boing e Cristiane do Nascimento, por serem essas pessoas tão especiais, as quais considero como as mais importantes em minha vida. Sem sombra de dúvidas, são eles os melhores amigos que eu poderia ter.

Também presto aqui minha gratidão aos meus amigos de curso, que me acompanharam nessa longa jornada. Dentre eles, Raul Missfeldt Filho e João Guilherme Fritsche Colombo, que estiveram ao meu lado quando eu mais precisei.

Agradeço meu co-orientador, Me. Hugo Vaz Sampaio, orientador, Dr. Carlos Becker Westphall, e a banca, Me. Leandro Loffi e Dr. Carla Merkle Westphall, por terem contribuído no desenvolvimento deste trabalho.

“Honra teu pai e tua mãe, para que se prolonguem os teus dias na terra que o SENHOR, teu Deus, te dá.” (Êxodo 20:12)

RESUMO

A qualidade e a segurança alimentar são frequentemente alvos de discussões, principalmente devido à escassez de nutrientes ou por serem um meio de transmissão de doenças. Preocupados com a saúde humana e com a garantia do bem-estar animal, os consumidores começam a optar pelo consumo de alimentos orgânicos. A avicultura, um dos principais meios de obtenção de carne no mundo, tem parte do seu mercado voltado à criação orgânica. Considerando o interesse nos produtos orgânicos, este trabalho tem como foco a automatização de ambientes para criação orgânica de frangos. Com base nos paradigmas *IoT* e *Fog Computing*, é proposto um alimentador inteligente, cujo objetivo é abastecer de forma autonômica os locais de alimentação das aves.

Palavras-chaves: *IoT, Internet of Things*, *Fog Computing*, aviário, frangos, orgânico, orgânica, alimentador, alimentação, avicultura.

ABSTRACT

The quality and food safety are often the subject of discussions, mainly due to nutrient shortages or because they are a means of disease transmission. Concerned about human health and animal welfare, consumers are beginning to choose organic food. Poultry farming is one of the means of obtaining meat in the world. It has part of its market focused on organic farming. Considering the interest in organic products, this work focuses on the automation of environments for the organic creation of chickens. Based on IoT and Fog Computing paradigms, an intelligent feeder is proposed, with objective to supply food to the birds, in an autonomous way.

Palavras-chaves: IoT, Internet of Things, Fog Computing, fowl, chickens, organic, feeder, food, poultry farming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
Fog	*Fog Computing*
Cloud	*Cloud Computing*
OMS	Organização Mundial de Saúde
SaaS	*Software as a Service*
PaaS	*Platform as a Service*
IaaS	*Infrastructure as a Service*
EUA	Estados Unidos da América
MAPA	Ministério da Agricultura, Pecuária e Abastecimento
IA	Inseminação Artificial
TE	Transferência de Embriões
RFID	*Radio-Frequency Identification*
Wi-Fi	*Wireless Fidelity*
WSN	*Wireless Sensor Networks*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
IOS	*Iphone Operating System*
SD	*Secure Digital*

SUMÁRIO

1. Introdução

A agricultura industrial, por conta dos seus baixos custos de produção, vem crescendo cada vez mais em nível mundial. Dentre os assuntos decorrentes está a criação intensiva de aves, que são confinadas dentro de ambientes fechados, praticamente sem espaço para se locomoverem. Especialistas relatam que nessas condições as aves vivem estressadas e perdem os seus instintos naturais, possibilitando o surgimento de diversos tipos de doenças. Muitos profissionais criticam a adoção do sistema, afirmando que são cruéis e causam prejuízos à saúde do animal, mais tarde repassados ao consumidor (ANOMALY, 2015).

Segundo a Organização Mundial de Saúde (OMS), citada por FELTES (2017), estima-se que em 2010 quase uma em cada dez pessoas adoeceram no mundo por contaminação alimentar. Crianças, idosos e gestantes, por possuírem baixas condições físicas e imunológicas, estão entre os mais propensos a contaminação. O Brasil é um dos interessados em combater esse mal, já que é um grande fornecedor de alimentos. Em 2015, o país foi o segundo maior produtor e exportador de carne de frangos no mundo.

De acordo com LOUZADA (2017), a industrialização é um assunto de fundamental importância para questão alimentar. De acordo com sua publicação, a causa de doenças como diabetes, obesidade e câncer estão fortemente ligadas com a ingestão de alimentos industrializados ultraprocessados. LOUZADA (2017) destaca, ainda, a importância das vitaminas e sais minerais (adequados) para saúde, os quais nem sempre podem ser obtidos a partir desse tipo de alimento.

1.1. Motivação

A avicultura ocupava o segundo lugar no ranking mundial de produção de carnes, até o ano de 2010, com um total equivalente a quase 72 milhões de toneladas produzidas ao ano. Sendo um dos principais exportadores de carnes de frango, o Brasil produziu naquele ano o equivalente a pouco mais de 12 milhões de toneladas (ROSSA, 2012). Entre 2014 e outubro de 2018 o Brasil se tornou o maior exportador mundial de frangos e o segundo maior produtor mundial da carne, tendo como seu principal concorrente os EUA, que nesse mesmo período ficou em segundo colocado no número de exportações e como primeiro no número de produção (PSD Online, 2018).

1.2. Justificativas

Apesar do crescimento da agricultura industrial, um número cada vez maior de consumidores vem optando pelo consumo de alimentos orgânicos. ZANDER (2010), em seus estudos, afirma existir um descontentamento das pessoas em relação ao modo convencional de produção. Segundo ZANDER (2010) e APAOLAZA (2018) isso se justifica principalmente pela preocupação relacionada à saúde e bem-estar humano e animal. Isso porque muitas pessoas acreditam ser mais saudável o consumo de alimentos orgânicos e consideram importante o tratamento dado aos animais que serão abatidos.

1.3. Objetivos

Neste subcapítulo são descritos os objetivos deste trabalho. Primeiramente é apresentado uma descrição geral sobre o assunto, e, na sequência, são enumerados de forma mais específica cada um dos objetivos.

1.3.1. Objetivo geral

Com base na criação orgânica de frangos, esse trabalho possui como objetivo o desenvolvimento de um sistema para automatizar a reposição de alimentos em ambientes de confinamento das aves. O sistema deve ser projetado de acordo com as leis em vigor, destinadas ao desenvolvimento orgânico de animais, no Brasil, as quais visam manter o bem-estar físico, psicológico, comportamental, ambiental e nutricional das aves no decorrer de todas as etapas de criação.

1.3.2. Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Manter os espaços de alimentação abastecidos;
2. Emitir sinais para alertar as aves quando o alimentador for reabastecido;
3. Fornecer meios para cadastrar dias/horários para o reabastecimento automático do alimentador;
4. Fornecer uma forma de controlar manualmente o alimentador, sem a necessidade do cadastro de dias e horários.

1.4. Método de pesquisa e trabalho

A seguir são apresentados os métodos de trabalho.

1. Reunir informações sobre o processo de criação de frangos e as diferenças entre os sistemas intensivo (tradicional) e semi-intensivo (caipira e orgânico);
    1. Pesquisar sobre as exigências das leis nacionais que possam influenciar no processo de desenvolvimento do sistema;
    2. Realizar um levantamento sobre os modos de criação de frangos, atualmente em vigor, que sejam relevantes para o projeto;
2. Pesquisar por tecnologias que estejam em uso, que possam ser aprimoradas ou que sirvam de base para criação de novas soluções;
3. Desenvolver a proposta com base nas tecnologias e informações encontradas;
4. Implementar um protótipo para demonstração da proposta.

1.5. Organização do trabalho

Esse documento está organizado em 7 Capítulos. No Cap. 2 são descritos os principais conceitos e paradigmas computacionais, que foram considerados para implementação da proposta. O Cap. 3 apresenta o estado da arte na área avícola. No Cap. 4 é feita uma breve descrição sobre cinco artigos, que servirão como base para o desenvolvimento da proposta. No Cap. 5 é apresentada a proposta, cuja implementação é mostrada no Cap. 6. Por fim, o Cap. 7 destina-se a conclusão e sugestões para trabalhos futuros.

2. Conceitos básicos

Neste capítulo são apresentados os principais conceitos, da área de computação, que serão utilizados no decorrer do trabalho, sendo eles: *Internet of Things*, *Cloud Computing* e *Fog Computing*.

2.1. *Internet of Things*

Internet of Things (*IoT*) é um conceito de redes de computadores que introduz objetos do mundo real ao virtual, com o propósito de possibilitar a comunicação entre eles. Dentro dessa rede encontram-se sensores, atuadores, computadores pessoais e qualquer outro dispositivo considerado inteligente (MUNIR, 2017). São esses dispositivos que capturam dados e informações, que posteriormente são utilizados para fornecer melhores serviços à população, em áreas residenciais, hospitalares, industriais (ZANELLA, 2014), agrícolas, dentre outros. Cada objeto possui uma identidade e atributos próprios e podem se conectar a Internet, portanto, para acessá-los não é preciso necessariamente que exista uma conexão de rede local (XU, 2014).

Para facilitar e simplificar a usabilidade, um sistema *IoT* pode ser dividido em vários blocos funcionais. Um exemplo é mostrado em RAY (2018), que realiza a divisão em seis blocos, os quais são descritos na sequência do texto.

- **Dispositivo:** são "coisas" do nosso dia a dia, como relógios inteligentes, veículos, máquinas e roupas. Fazem parte dos objetivos de um dispositivo a realização de coleta de dados, envio de dados para servidores, comunicação com outros dispositivos e/ou aplicativos. A maioria dos dispositivos realizam o processamento de dados para gerar informações úteis, posteriormente utilizadas para alcançar um dado objetivo. Um exemplo é o processo de monitoramento de um jardim, em que são determinados os melhores horários para realizar a irrigação;
- **Comunicação:** é responsável pela comunicação entre dois ou mais dispositivos e entre dispositivos e servidores remotos. Os protocolos *IoT* funcionam, na maioria das vezes, na camada de enlace, rede, transporte e aplicação;
- **Serviços:** representam as funcionalidades do sistema *IoT*. Como serviços podem ser considerados a modelagem, busca e controle de dispositivos, ou a publicação e análise de dados;

- **Gerenciamento:** fornece ferramentas para gerenciar um sistema *IoT*;
- **Segurança:** disponibiliza mecanismos para garantir a proteção do sistema *IoT*, como a autenticação, autorização, privacidade e a integridade e segurança de dados;
- **Aplicação:** fornece meios para integração do usuário ao sistema *IoT*. A integração ocorre através de interfaces que possibilitam controlar e monitorar o sistema.

Uma rede *IoT* pode servir para solucionar vários tipos de problemas. STERGIOU (2018) cita alguns casos em que se faz o uso do paradigma:

- **Transporte:** auxilia na obtenção de soluções de problemas relacionados ao tráfego em rodovias, redução do consumo de combustível e a diminuição nos índices de mortes;
- **Redes elétricas:** são utilizadas para auxiliar na incorporação de energia renovável, melhorando a confiabilidade do sistema e tornando a energia mais barata;
- **Monitoramento remoto de pacientes hospitalares:** proporciona uma maior facilidade para atender os pacientes, gerando melhores condições e o aumento do número de pacientes atendidos;
- **Sensores de monitoramento de motores:** possibilitam a detecção de problemas de manutenção, tornando possível melhorar a reposição de peças e definir prioridades no agendamento de serviços.

2.2. *Cloud Computing*

Cloud Computing é um conceito computacional que fornece uma gama de recursos a outros computadores e dispositivos. Os recursos fornecidos pela *Cloud* ficam disponíveis de forma compartilhada, a qualquer hora e lugar, e são acessados através da Internet. Como exemplo tem-se o processamento e armazenamento de dados, serviços, aplicativos (MUNIR, 2017) e servidores. Por conta dos seus benefícios e da possibilidade de cobrança financeira, o paradigma deu origem a uma indústria multibilionária crescente em todo o mundo (ROMAN, 2018). Dentre as principais classificações de uma *Cloud* estão (SINGH, 2016):

- ***Public Cloud* (nuvem pública):** nesse modelo os recursos e serviços são públicos, portanto, qualquer organização, empresa, ambiente acadêmico ou indivíduo tem acesso. Por não existir um controle de clientes e fornecedores, existe uma maior vulnerabilidade em relação a segurança;

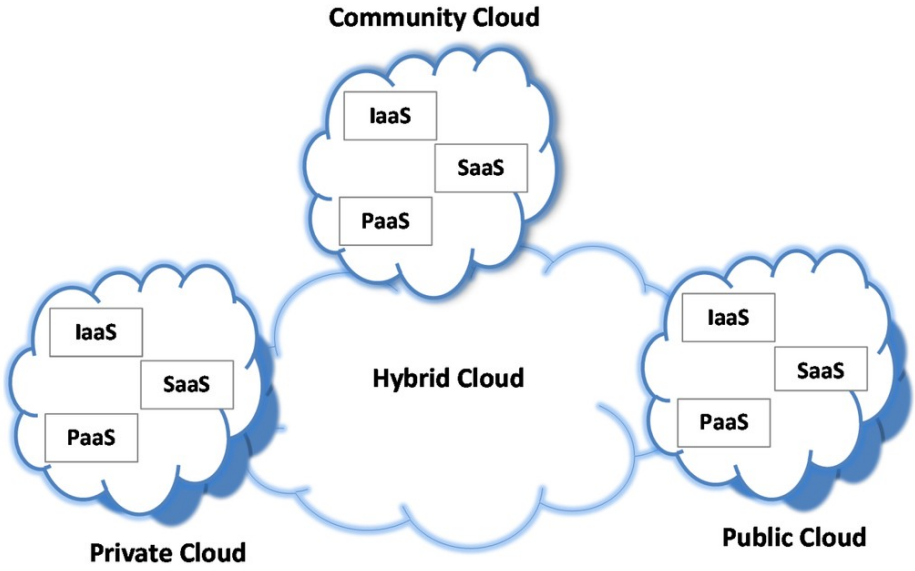
- ***Community Cloud* (nuvem comunitária):** é utilizada e gerenciada por um grupo mais restrito de organizações. Por possuírem um acesso mais controlado, os riscos relacionados de segurança são reduzidos;
- ***Private Cloud* (nuvem privada):** o uso e gerenciamento é exclusivo da organização a qual pertence. Por existir uma maior facilidade na identificação de clientes e os fornecedores, os riscos relacionados a segurança são menores do que os modelos público e comunitário;
- ***Hybrid Cloud* (nuvem híbrida):** representa a união de dois ou mais modelos de nuvens (pública, privada e comunitária).

O *National Institute of Standards and Technology* (*NIST*) descreve três modelos de infraestrutura (MELL, 2011):

- ***Software as a Service* (*SaaS*):** os aplicativos são acessados por meio de uma infraestrutura da própria *Cloud*. Os usuários fazem o acesso através de interfaces, como um *Browser* ou algum outro software, porém não possuem acesso a qualquer tipo de gerenciamento de infraestrutura, tais como rede, sistema operacional, e o armazenamento de dados. O gerenciamento de aplicativos também possuem acesso limitado, podendo existir algumas exceções;
- ***Platform as a Service* (*PaaS*):** fornece a possibilidade de hospedar aplicativos desenvolvidos pelo próprio usuário. Assim como em SaaS, o cliente não tem acesso ao gerenciamento da infraestrutura, porém possui liberdade para gerenciar os próprios aplicativos;
- ***Infrastructure as a Service* (*IaaS*):** são disponibilizados recursos de processamento, armazenamento, redes e outros recursos de computação. O usuário pode instalar sistemas operacionais e hospedar aplicativos, porém não controla a infraestrutura da nuvem.

A Fig. 1 mostra a relação entre as classificações e os modelos de *Cloud Computing*. Os três modelos (*IaaS*, *PaaS* e *SaaS*) são utilizados em *Public, Community, Private* e *Hybrid Cloud*.

Figura 1 - Representação dos modelos de infraestrutura e classificações de nuvens.



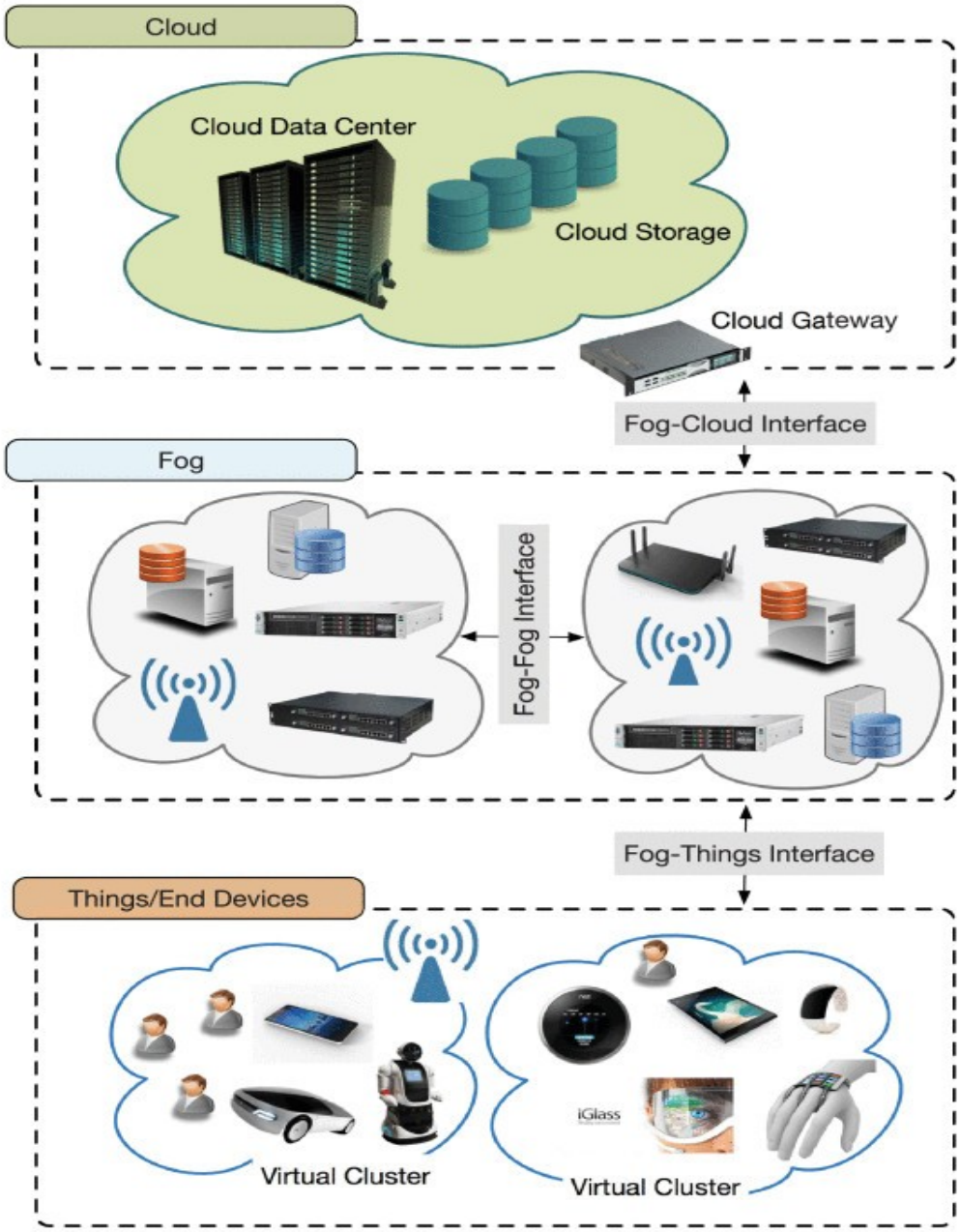
Fonte: KHALIL (2014)

2.3. *Fog Computing*

Fog Computing surgiu como um meio alternativo para intermediar a comunicação entre os dispositivos *IoT* e a *Cloud* (PERALA, 2018). O paradigma surgiu em 2014, criado pela Cisco (CHEN, 2018), e se caracteriza pelo fornecimento de serviços da *Cloud* dentro da borda da rede. O objetivo é aumentar o desempenho das aplicações ao processar e armazenar parte dos dados localmente, enviando para *Cloud* somente o necessário. Os resultados podem ser notados pela diminuição do tráfego de rede, redução no tempo de espera por serviços (GUPTA, 2017; PERALA, 2018; MUNIR, 2017), agilidade, acesso via rede sem fio e aplicações em tempo real (MUKHERJEE, 2018). Por consequência, é possível usar a *Cloud* para realizar outros serviços de maior complexidade, como aprendizado de máquina e processamento *Big Data* (CHEN, 2018).

A Fig. 2 ilustra a relação entre os dispositivos *IoT* e os serviços *Fog* e *Cloud Computing*. Os dispositivos realizam a troca de dados com a *Fog*, que por sua vez processa e envia/recebe apenas o necessário para *Cloud*.

Figura 2 - Representação da relação entre os conceitos *Cloud*, *Fog* e *IoT*.



Fonte: MUKHERJEE (2018)

3. Fundamentação teórica na área avícola

Neste capítulo são apresentados os conceitos fundamentais da área avícola e as diferentes formas de criação de frangos, sendo elas: produtos orgânicos, alimentos orgânicos no Brasil, avicultura intensiva e avicultura orgânica e caipira. Também são levantados os principais parâmetros utilizados para o controle de um galinheiro orgânico autonômico.

3.1. Produtos orgânicos

A classificação de alimentos como sendo orgânicos é considerada uma forma de dar credibilidade ao produto, pois além das informações visíveis na embalagem, o selo orgânico tende a comunicar ao consumidor que o produto atende a normas e padrões específicos de qualidade. Essas informações tem o propósito de dar mais transparência para o consumidor do que àquelas contidas em produtos convencionais (GIFFORD, 2011), e essas normas podem variar de um país para outro. De modo geral tem-se por objetivo a produção de alimentos que busquem manter o equilíbrio ecológico, evitando o uso de pesticidas, herbicidas, fertilizantes inorgânicos, antibióticos e hormônios de crescimento, em todas as etapas de produção (HONKANEN, 2006).

De acordo com GIFFORD (2011) e ROSSA (2012), nas últimas décadas é crescente o número de vendas relacionadas a produtos orgânicos na Europa e nos EUA. Dentre estes produtos está a carne de origem aviária, que corresponde a dois terços do mercado orgânico de carnes e é um dos primeiros tipos de alimentos a serem buscados na categoria orgânica. Estima-se que nos EUA cerca de dois terços da população compre orgânicos ocasionalmente, e 19% consuma esses produtos de forma semanal (GIFFORD, 2011). De acordo com alguns estudos, os consumidores acreditam que esses alimentos são mais saudáveis e possuem uma maior qualidade (ROSSA, 2012).

A produção de orgânicos no mundo correspondia a aproximadamente 35 milhões de hectares em 2008, distribuídas entre 154 países. A Oceania era a maior produtora, com 35% da produção mundial, em segundo a Europa e em terceiro a América Latina. No ano anterior (2007), o Brasil estava em quinto lugar entre as maiores áreas de cultivo, com 1,77 milhões de hectares (ROSSA, 2012).

3.2. Alimentos orgânicos no Brasil

De acordo com a Lei 10.831 de 2003, o objetivo da produção orgânica no Brasil é de diminuir o uso de contaminantes químicos no meio ambiente (ar, solo, água), de modo a preservar a diversidade biológica dos ecossistemas e proporcionar uma alimentação mais saudável ao consumidor (BRASIL, 2003). Um produto é dito como orgânico, seja ele processado ou *in natura*, caso seja obtido através de algum sistema orgânico de produção agropecuário ou por algum tipo de processo extrativista sustentável. Isto é, sua obtenção deve ser de modo a não causar prejuízos ao ecossistema local (BRASIL, 2003). Para sua comercialização o produto deve possuir uma certificação junto ao MAPA (Ministério da Agricultura, Pecuária e Abastecimento), obtido através de um órgão reconhecedor oficial (BRASIL, 2003).

3.3. Avicultura intensiva

Caracterizado pelo confinamento total das aves, o sistema intensivo é o modelo de criação mais empregado na avicultura. O objetivo desse sistema é o aumento da produção e sua eficiência, obtidos através de melhorias genéticas, aprimoramento na coleta de dados e instalação de novas tecnologias de manejo, nutrição, bem estar e higiene (VOGADO, 2016). Contudo, manter as aves confinadas é prejudicial a saúde, e ao contrário a esses investimentos tecnológicos, o sistema não converge para um bem-estar dos animais (SILVA, 2003).

No Brasil, a produção intensiva, reconhecida mundialmente, teve um forte investimento tecnológico em seu processo de produção, aliado a mudanças no controle sanitário e melhorias na genética das aves. Algumas medidas de melhoramento genético foram: Inseminação Artificial (IA); Transferência de Embriões (TE); micro manipulação e produção *in vitro* de embriões; clonagem e produção de animais transgênicos ou altamente modificados. A diminuição no tempo de crescimento dos frangos está entre os resultados desse investimento, que passou de 105 dias em 1930 para 45 dias em 1996 (VOGADO, 2016).

Com os avanços na avicultura industrial foram surgindo modificações nos hábitos alimentares e nas tradições da população. O frango caipira foi ao longo do tempo perdendo espaço na mesa do consumidor. Contudo, mesmo sendo substituído, existe uma maior preferência pela carne caipira devido suas características próprias na textura e no sabor da carne (VOGADO, 2016).

Apesar do sistema convencional ser o dominante no mercado, algumas formas alternativas, como a caipira e orgânica, estão voltando a ganhar espaço na mesa dos consumidores. A fim de minimizar os impactos do sistema industrial e melhorar o bem-estar das aves, a União Europeia impôs em 1999 algumas medidas mínimas para sistemas de criação. Em 2003, por exemplo, foram proibidas as instalações de gaiolas convencionais com menos de 550cm² por ave, além da necessidade do uso de lixas para as unhas (ROCHA, 2008). A Fig. 3 mostra o confinamento de frangos em um aviário de criação intensiva.

Figura 3 - Frangos em gaiolas no sistema avícola intensivo.

Fonte: EMBRAPA (2018)

3.4. Avicultura alternativa: caipira e orgânica

Os sistemas de produção caipira e orgânico são modelos alternativos ao sistema intensivo e são projetados para garantir a qualidade e o bem-estar dos animais. As raças que serão utilizadas devem ser aquelas que melhor se adaptam ao manejo e clima local, devendo ser garantido sua segurança e que estejam livres de medo, ansiedade, ou qualquer outro tipo de estresse. É preciso que o ambiente proporcione aos animais a liberdade para apresentarem suas características e comportamentos naturais, e que não passem fome, sede e não sofram de nenhum tipo de doença (FIGUEIREDO, 2012; DIAS, 2016). O sistema de criação e as raças são fundamentais para que se garanta uma carne mais saborosa e com menores índices de gordura (DIAS, 2016).

O Fator climático é também um dos fatores cruciais na criação das aves, pois as afeta diretamente. Isso pode ser percebido através de mudanças no comportamento, como respiração mais ofegante, diminuição na taxa de locomoção e o ato de manter as asas mais afastadas do corpo (DIAS, 2016). Apesar de no sistema orgânico os frangos necessitarem de liberdade para o acesso externo, onde devem existir grama e/ou algum outro tipo de

vegetação, isso pode, por conta do clima, vir a interferir no bem-estar animal e consequentemente na produção do sistema (BERG, 2002; DIAS, 2016). Dentre as consequências estão o aumento no consumo de água e ração, que podem vir a causar mudanças na taxa de crescimento das aves, afetando o rendimento e a qualidade da carne (DIAS, 2016; CASSUCE, 2013).

A fim de promover o bem-estar animal e viabilizar a inspeção em um sistema produtivo, o *Farm Animal Welfare Council* criou cinco conceitos para definir um sistema controlado por humanos que fosse capaz de proporcionar um bem-estar aos animais (PEREIRA, 2017):

- **Nutricional:** manter os animais livres de fome e sede;
- **Ambiental:** mantê-los livre de desconfortos;
- **Físico:** mantê-los livres de dores, lesões e doenças;
- **Comportamental:** dar liberdade para expressar seus comportamentos normais;
- **Psicológico:** deixá-los livres de medo e angústia.

4. Estado da arte

Com o objetivo de evidenciar a relevância da pesquisa, uma revisão sistemática foi desenvolvida com base nas ferramentas *Google Scholar*, *IEEE Explorer* e *ScienteDirect*. O resultado da revisão pode ser visto na Tab. 1, que apresenta a união do número de ocorrências de cada conjunto de palavras chaves, em inglês, nas três ferramentas de pesquisa. A última combinação, diferente das outras, não teve nenhum retorno nas ferramentas *IEEE Explorer* e *ScienteDirect*.

Tabela 1 - Resultado da revisão sistemática sobre cada palavra chave.

Palavras chave	**Resultados**	**Categoria**
automation	3.560.000	1
control	7.270.000	1
fog	1.120.000	1
iot	782.000	1
fog iot	16.700	2
feeding chickens	708.000	2
automation fog	38.500	2
automation iot	83.200	2
control fog	536.000	2
control iot	244.000	2
automation feeding	673.000	2
control feeding	3.930.000	2
iot feeding	15.400	2
fog feeding	80.000	2
automation feeding chickens	44.500	3
automation control feeding chickens	37.600	4
automation control feeding chickens fog iot	6.540	6
automation control feeding chickens alternative aviaries fog iot	130	8

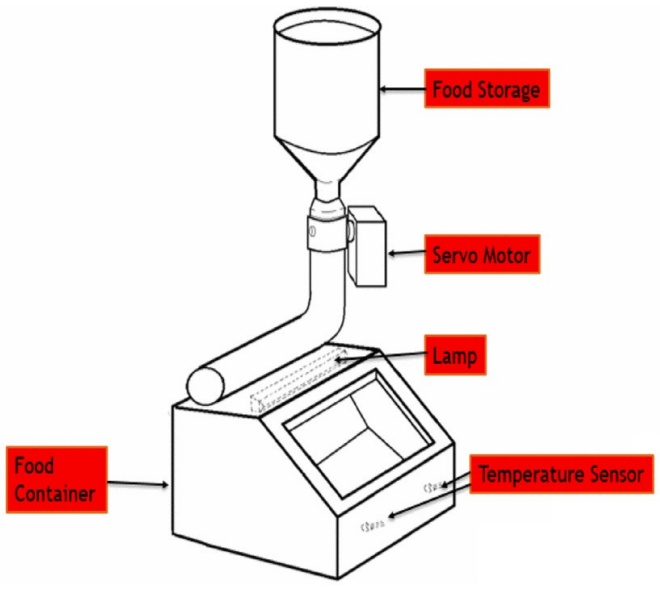
Fonte: Próprio (2019)

4.1. *Development of automatic chicken feeder using Arduino Uno*

A alimentação de frangos, criados para o abate, pode exigir tempo e muita mão de obra. Para realizar o trato de forma manual, o criador costuma fornecer o alimento semeando-o no chão do aviário. Existem ao menos três problemas em fazê-lo dessa forma: contaminação dos alimentos, decorrentes da presença de insetos e de fezes das aves; desperdício de alimentos, por falha humana, já que não é possível uma correta administração da quantia a ser distribuída; e a necessidade da presença dos responsáveis pelo fornecimento do alimento, o que dificulta que estes indivíduos venham realizar outras tarefas. Com o objetivo de suprir esses problemas, um alimentador de frangos é proposto a fim de automatizar por completo o processo de alimentação, possibilitando ao criador o cumprimento de outras obrigações sem que haja a necessidade de sua presença no local.

Ao projetar o alimentador, o autor considerou a necessidade de armazenar os alimentos, administrar as sobras, e registrar os horários de fornecimento e a quantidade a ser fornecida. A Fig. 5 mostra o projeto do tratador, o qual é composto por alguns sensores e atuadores controlados por uma placa Arduíno Uno. O processo de liberação do alimento acontece com a abertura da passagem entre local de armazenamento de comida e o recipiente, controlada pelo atuador servo motor. Para evitar o desperdício da comida deixada pelas aves, as sobras ficam retidas dentro do recipiente. Um sensor de temperatura e uma lâmpada são utilizados para que se garanta que os grãos permaneçam frescos.

Figura 4 - Alimentador automático de frangos.



Fonte: SOH (2017)

4.2. *A Smart Sensor Network for an Automated Urban Greenhouse*

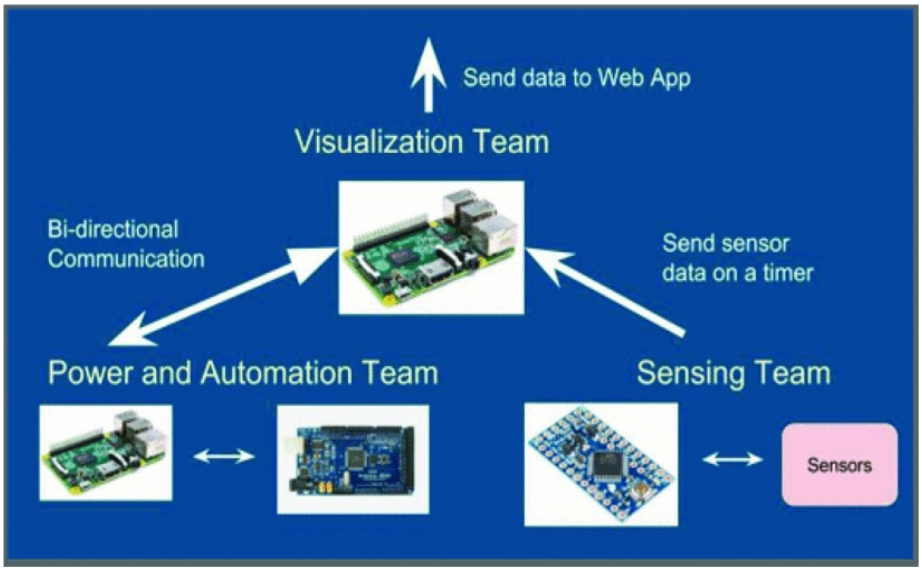
Considerando a dificuldade de obter alimentos frescos a preços acessíveis, na região da *Pennsylvania* (EUA), em MEAH (2019) é apresentado um projeto de redes de sensores inteligentes para automatização de uma estufa urbana. O projeto foi desenvolvido por alunos de um colégio de engenharia, o *York College of Pennsylvania*, e foi utilizado para apoiar a inicialização de alunos de escola primária no mundo científico. Dentre os objetivos do trabalho estão:

- Maximização da economia de energia;
- Gerenciamento de aquecedores, resfriadores e da ventilação;
- Uso de histórico de dados meteorológicos para estimativa da energia necessária dentro da estufa.

Para o desenvolvimento do projeto foi feita a divisão do sistema em três partes: sensoriamento; energia e automação; interface. No primeiro subsistema a equipe fez o uso de sensores para coletar dados de luz, umidade e temperatura dentro da estufa. Os dados são encaminhados posteriormente ao subsistema de energia e automação, que irá processá-los e intervir no ambiente a partir de equipamentos como aquecedores e lâmpadas. O protótipo desenvolvido possui 6 (seis) zonas de umidade, sendo que cada uma possui 2 (dois) sensores de umidade. Em paralelo, 5 (cinco) sensores de temperatura, luz e umidade foram distribuídos em plantas. Todo o processo de interação com o usuário foi desenvolvido no subsistema interface, onde ficam hospedados uma aplicação web e o banco de dados.

A Fig. 5 mostra a interação entre as partes do sistema. Um Raspberry Pi é utilizado para hospedagem da interface e intermediar a comunicação entre os subsistemas de sensoriamento e de energia e automação. A rede de sensores realiza a coleta de dados e transmite para o Raspberry Pi da interface. Na área de energia e automação foi instalado um segundo Raspberry Pi, em conjunto com um Arduíno Mega, para atuar no ambiente.

Figura 5 - Representação da integração entre os equipamentos dos 3 (três) subsistemas, em MEAH (2019): sensoriamento; energia e automação; interface.



Fonte: MEAH (2019)

4.3. *A remote pet feeder control system via MQTT protocol*

Popularmente aceito dentro da sociedade moderna, a criação de animais vem se tornando um hábito cada vez mais adotado entre as pessoas. Contudo, responsabilidades devem ser assumidas pelos proprietários, sendo uma delas a garantia da alimentação. Em WU (2018) é desenvolvido um alimentador de animais que, ao contrário da maioria, deixa de ser uma simples máquina estacionária para se tornar um alimentador móvel e inteligente. O equipamento *IoT* foi projetado para se assemelhar a um carrinho de brinquedo, possuindo duas rodas traseiras, cada uma conectada a um motor. Ainda, uma câmera IP foi instalada na parte frontal do carrinho para possibilitar o monitoramento pela Internet.

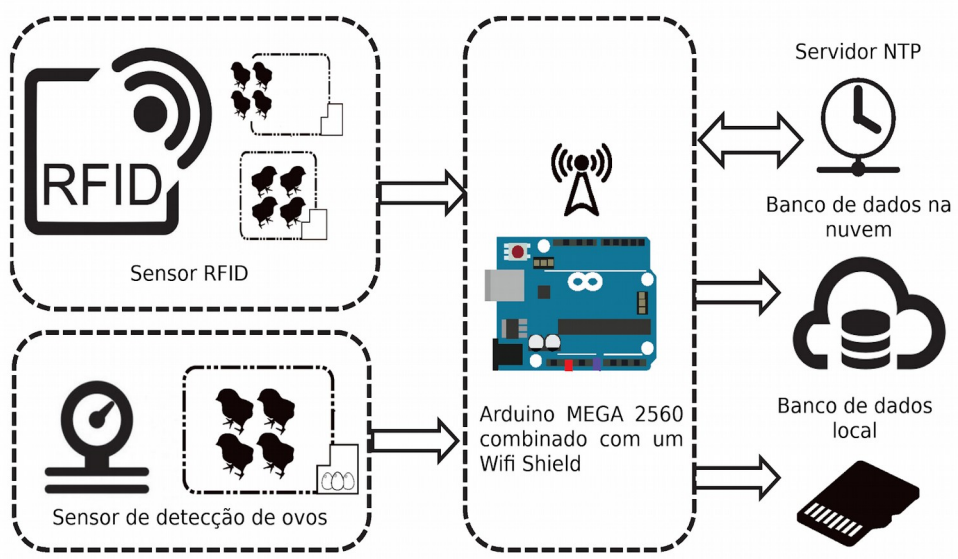
4.4. *An RFID-Based Smart Nest Box: An Experimental Study of Laying Performance and Behavior of Individual Hens*

O nível de produção de ovos pode ser afetado pelo ambiente de postura (ninho), idade das aves ou até mesmo pela raça do animal. Com a motivação de monitorar a produção de ovos, em CHIEN (2018) é apresentada uma solução que tem como objetivo a identificação de galinhas poedeiras que apresentam um rendimento abaixo do esperado. A ferramenta desenvolvida utiliza um leitor *RFID*, conectado a uma placa Arduíno MEGA 2560, cuja

serventia é identificar a presença de aves dentro de um ninho. Também é feito o uso de um sensor de pressão, para que seja detectado a existência de algum ovo.

A Fig. 9 apresenta um esquema do protótipo desenvolvido. No canto superior esquerdo é feita a ilustração referente a identificação das aves, pela antena *RFID*. No canto inferior esquerdo é mostrado o sensor de pressão, que assim como o sensor *RFID*, é conectado na placa Arduíno MEGA 2560. Por fim, os dados referentes a identidade das galinhas e o peso dos ovos são salvos em um cartão de memória (*SD*) e em um banco de dados na nuvem.

Figura 6 - Diagrama ilustrativo do sistema desenvolvido por CHIEN (2018) Adaptado.



Fonte: CHIEN (2018)

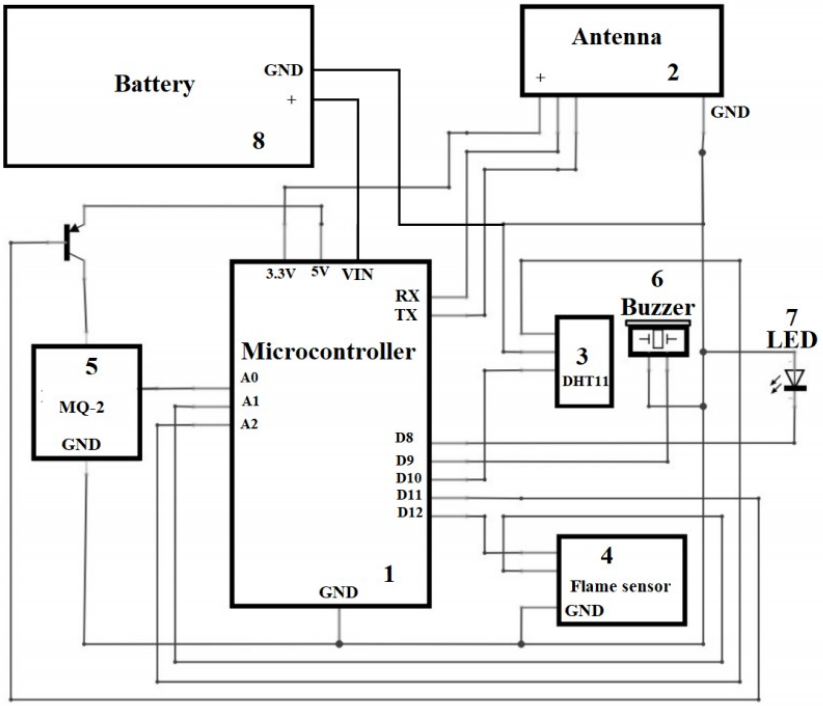
4.5. *Autonomic IoT Battery Management with Fog Computing*

O uso de energia é indispensável para dispositivos *IoT*, sendo em grande parte dependentes do uso de baterias para que funcionem. Para os usuários é crucial que existam técnicas para economizar energia, principalmente quando a exigência é de anos de durabilidade. Em SAMPAIO (2019) é desenvolvido um sistema *Fog* para alarmes de incêndio que depende do uso de baterias. O trabalho é dividido em duas partes, sendo a primeira referente ao hardware do dispositivo e na segunda são realizadas estimativas do consumo de energia.

O dispositivo *IoT* foi projetado e desenvolvido considerando o contexto de cozinhas em *Smart Home's*. Para a implementação do protótipo os autores fizeram o uso de um microcontrolador Arduíno Uno ATMega328p, um sensor de temperatura e umidade do ar

DHT11, um sensor de gás e fumaça MQ-2 e um sensor de chamas. O dispositivo *IoT* captura os dados através de portas analógicas do Arduíno, conectadas aos sensores, e os envia para *Fog* com uma antena Xbee Zigbee. O sistema *Fog*, por sua vez, processa os dados para gerar e apresentar a estimativa de consumo. A Fig. 6 apresenta o esquemático do dispositivo *IoT*, onde são mostradas as conexões entre: os componentes de hardware e a bateria; o Arduíno e os sensores; o Arduíno e a antena ZigBee.

Figura 7 - Esquemático do dispositivo *IoT* em SAMPAIO (2019).



Fonte: SAMPAIO (2019)

4.6. Proposta x trabalhos correlatos

Ao longo deste capítulo foram mencionados e descritos 5 (cinco) trabalhos correlatos. Dentre os trabalhos, em SOH (2017) apresenta-se uma proposta para automatização da alimentação de frangos. O alimentador desenvolvido é capaz de gerenciar o armazenamento e fornecimento dos alimentos, porém se limita ao uso em pequena escala. Para expandir esse alimentador, a proposta do Cap. 5 considerou a adaptação do carrinho proposto em WU (2018). No lugar de uma caixa de pequenas dimensões, como em SOH (2017), neste trabalho é utilizada uma caixa com um comprimento maior, em que um carrinho (nó *IoT*) se movimenta para liberar o alimento.

O desenvolvimento da proposta (Cap. 5) possui, ainda, a necessidade da criação de várias caixas de alimentação contendo seu próprio nó *IoT*. Uma rede de nós foi projetada, semelhante a rede de sensores apresentada em MEAH (2019). Na proposta (do alimentador) são considerados vários nós *IoT*, assim como na rede de sensores. A diferença é que em MEAH (2019) os nós são utilizados para obtenção de dados, enquanto os nós do alimentador se motivam em atuar no ambiente de aplicação.

Já em relação a criação de um nó *IoT* móvel, existe a necessita de uma maior flexibilidade em relação ao consumo de energia. Para suprir essa necessidade são necessárias baterias, pilhas, ou similares. O problema de usar esse tipo de fonte está na sua capacidade e duração, tornando-se essencial a realização de otimizações para diminuir o consumo de energia de cada nó. Em SAMPAIO (2019) apresenta-se um estudo sobre o consumo de energia em dispositivos para *Smart Home's*. Apesar de não ter sido considerado na implementação da proposta, esse trabalho evidencia a necessidade da redução do consumo de energia para aumentar o tempo de duração do dispositivo *IoT*.

O gerenciamento da alimentação pode diminuir o trabalho dentro de um aviário orgânico, mas não libera os trabalhadores de todo o esforço manual. O último trabalho foi mencionado por ter como objetivo a criação de um ninho inteligente. As propostas de gerenciar de forma inteligente a alimentação e os ninhos podem trazer maiores benefícios, caso utilizadas em conjunto.

5. Proposta de alimentador inteligente

Considerando o crescente interesse no consumo de alimentos orgânicos, em conjunto com a importância do mercado brasileiro de frangos, um alimentador inteligente é proposto para gerenciar a alimentação das aves de forma autonômica. O alimentador conta com um sistema *Fog*, dividido em três camadas, que considera desde a interação com o usuário até a criação e controle de um dispositivo *IoT*. No subcapítulo 5.1 é apresentada a hierarquia do sistema, enquanto no subcapítulo 5.2 são descritos o dispositivo *IoT* e a caixa de alimentação e no subcapítulo 5.3 é descrito o sistema *Fog*.

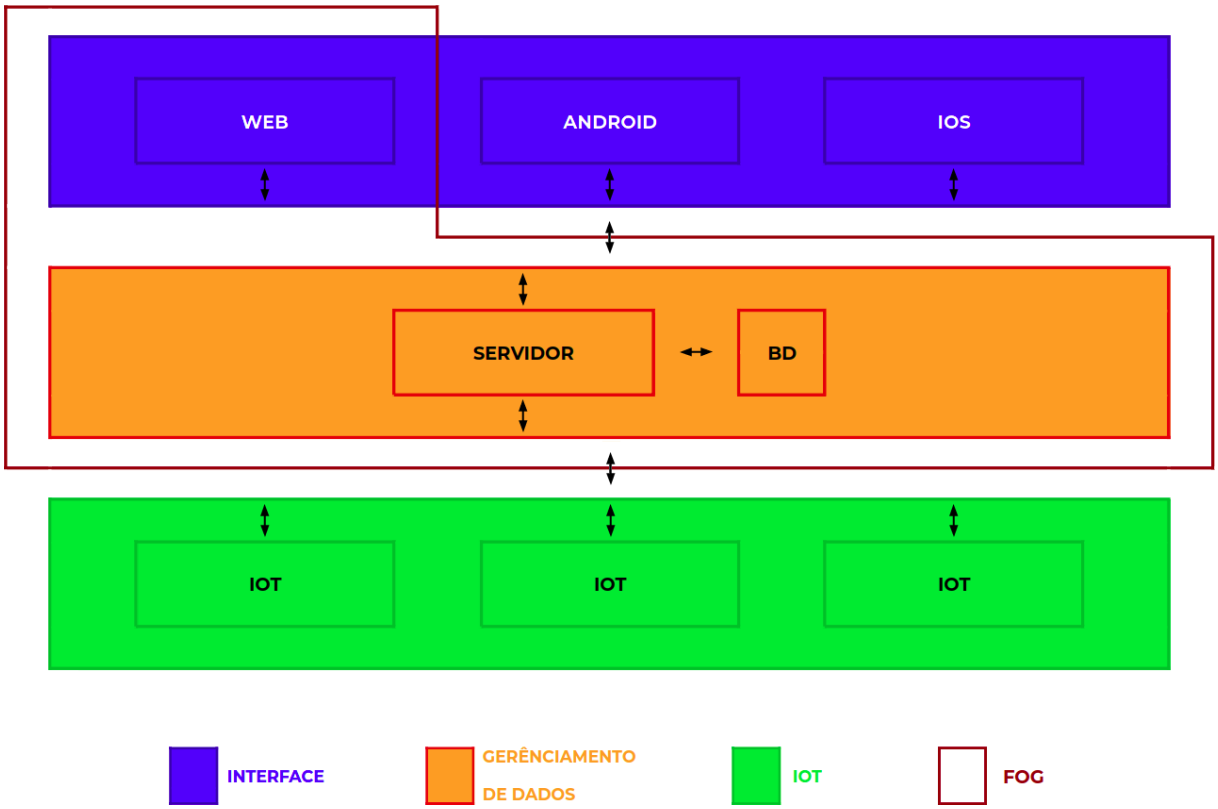
5.1. Arquitetura do sistema

A arquitetura do sistema é dividida em três camadas, conforme apresentado na Fig. 8. Uma camada para entrada de dados é responsável pela comunicação com os usuários, denominada como camada de interface. Abaixo dela encontra-se a camada de gerenciamento de dados, a qual se responsabiliza pelo armazenamento dos dados e a comunicação com os nós *IoT*. A última camada é a *IoT*, localizada abaixo da camada de gerenciamento de dados. Uma melhor descrição das camadas é apresentada na sequência do texto.

- **Camada de interface:** é responsável pela obtenção de dados, fornecidos pelos usuários, e pelo envio desses dados para camada de gerenciamento de dados. Na camada de interface é possível que sejam feitos o uso de uma ou mais aplicações, desenvolvidas para diferentes sistemas, tais como Android, iOS, Desktop e sistemas *Web*;
- **Camada de gerenciamento de dados:** se responsabiliza pelo armazenamento dos dados obtidos na camada de interface e pelo envio de comandos de controle para os dispositivos da camada *IoT*. Para comunicação com a interface, a camada conta com um servidor que receberá as informações e armazenará em um banco de dados. Já para comunicação com os dispositivos *IoT*, a camada possui uma aplicação que identificará e enviará aos dispositivos o comando para a realização de uma tarefa na data e horário especificados pelos usuários (armazenados no banco de dados);
- **Camada *IoT*:** representa a união de todos os dispositivos *IoT* do sistema. Os dispositivos recebem os comandos de controle, enviados pela camada de

gerenciamento de dados, e com base nos comandos é identificada e cumprida a tarefa em questão.

Figura 8 - Arquitetura do sistema *Fog*, composto por três camadas: camada de interface; camada de gerenciamento de dados; e camada *IoT*.

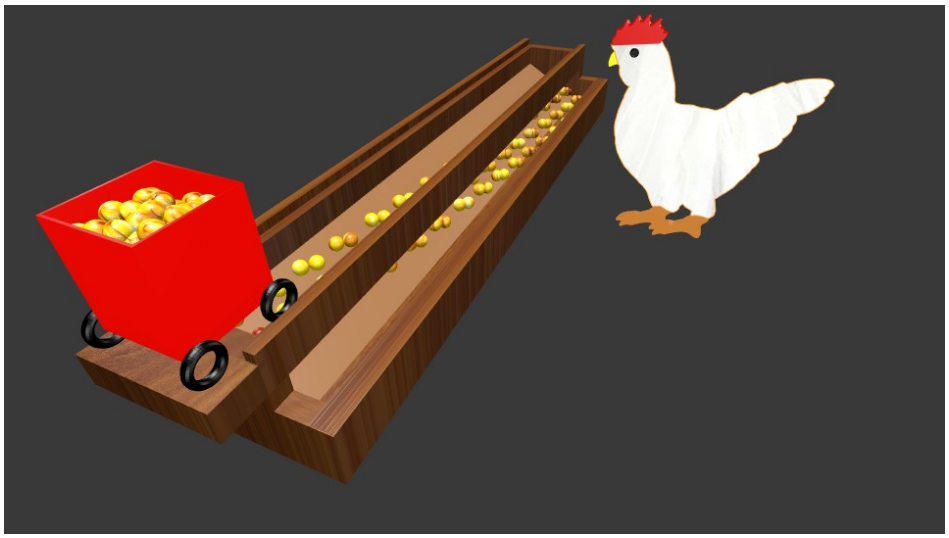


Fonte: PRÓPRIO (2019)

5.2. Nó *IoT* e a caixa de alimentação

O nó *IoT* é um equipamento baseado em um carrinho de brinquedo, cuja finalidade é armazenar o alimento das aves e liberá-lo apenas em datas e horários específicos. Para a realização da tarefa, o dispositivo *IoT* conta com o auxílio de uma base (caixa de alimentação) que receberá e disponibilizará o alimento para as aves. O dispositivo é posicionado em cima de dois trilhos, fixados na parte superior da caixa, e permanece imóvel até o recebimento de um comando enviado pelo servidor. Ao receber o comando, o nó caminha sobre os trilhos até o fim e depois retorna ao ponto de origem, liberando o alimento durante todo o trajeto. Na Fig. 9 uma galinha está de frente para o alimentador, logo após o nó *IoT* liberar o alimento.

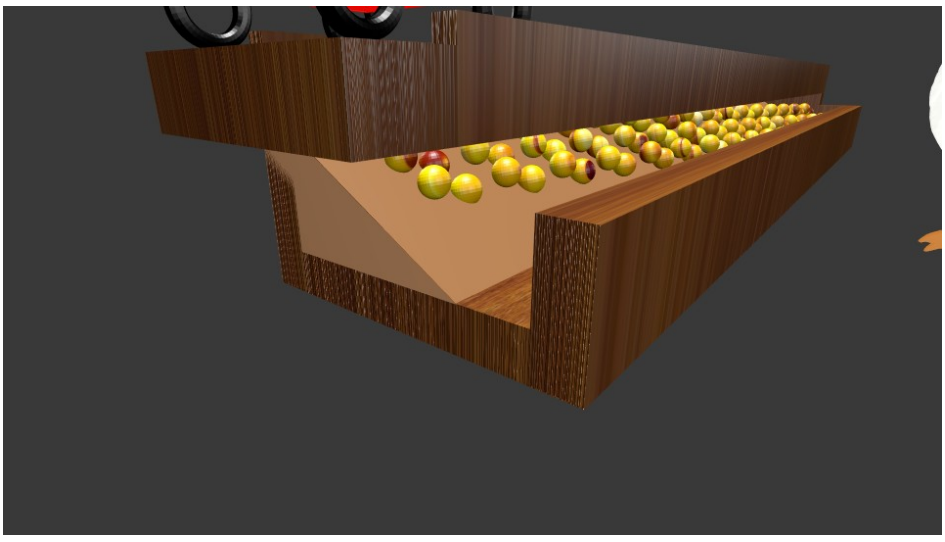
Figura 9 - Visão geral do alimentador, em 3D.



Fonte: PRÓPRIO (2019)

A caixa de alimentação conta, ainda, com uma área destinada ao acesso das aves, localizada na parte frontal, e uma rampa de desvio de alimentos, posicionada abaixo dos trilhos utilizados pelo nó *IoT*. O alimento, quando liberado, cai sobre a rampa e sofre um desvio até a parte frontal do alimentador. A Fig. 10 apresenta uma ilustração da rampa e da área de acesso às aves ao alimentador, onde o alimento fica armazenado até ser consumido.

Figura 10 - Visão lateral do alimentador, em 3D.

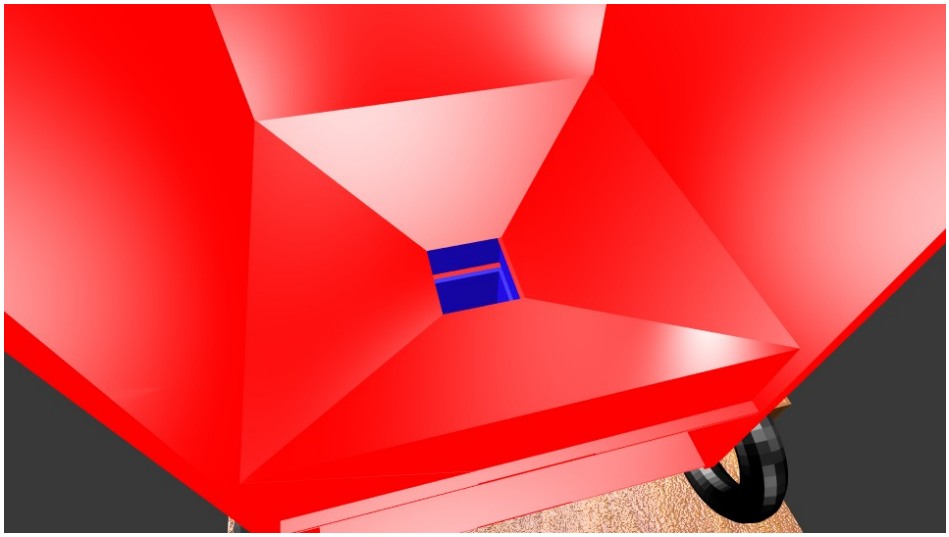


Fonte: PRÓPRIO (2019)

O projeto do carrinho, que consiste de um nó *IoT*, ocorre com a sua divisão em duas partes: superior e inferior. Na parte superior (Fig. 11) é feito o armazenamento do alimento, o qual passa por um processo de afunilamento para que ele deslize até o centro do dispositivo.

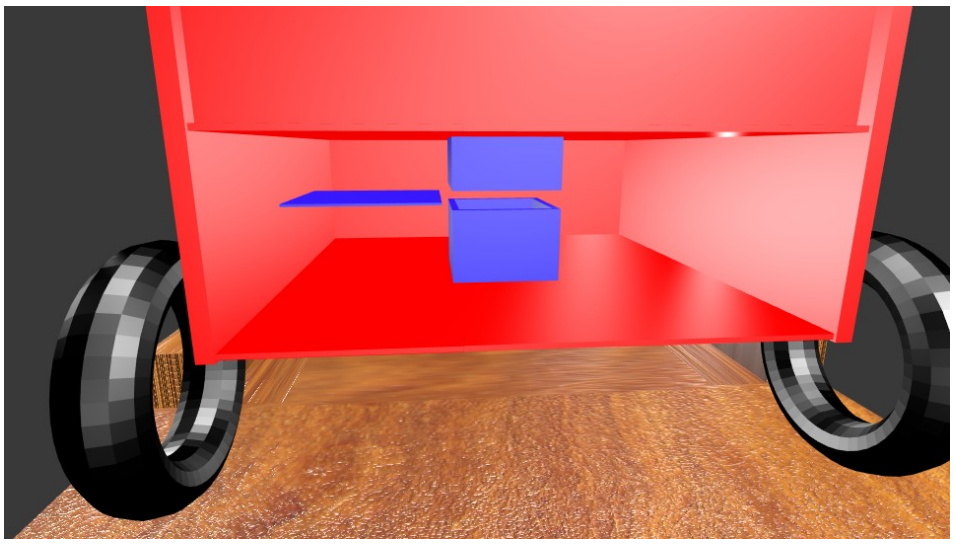
Após esse processo, o alimento cai dentro de um corredor que o levará até a parte inferior (Fig. 12). Quando isso ocorrer, um controlador de passagem decidirá se o alimento deve ou não ser liberado. Para isso conta-se com uma tampa de isolamento, que fica aberta ou fechada de acordo com a necessidade (liberação ou armazenagem do alimento, respectivamente).

Figura 11 - Visão interna da parte de cima do carrinho (nó *IoT*), em 3D.



Fonte: PRÓPRIO (2019)

Figura 12 - Visão interna da parte de inferior do carrinho (nó *IoT*), em 3D.



Fonte: PRÓPRIO (2019)

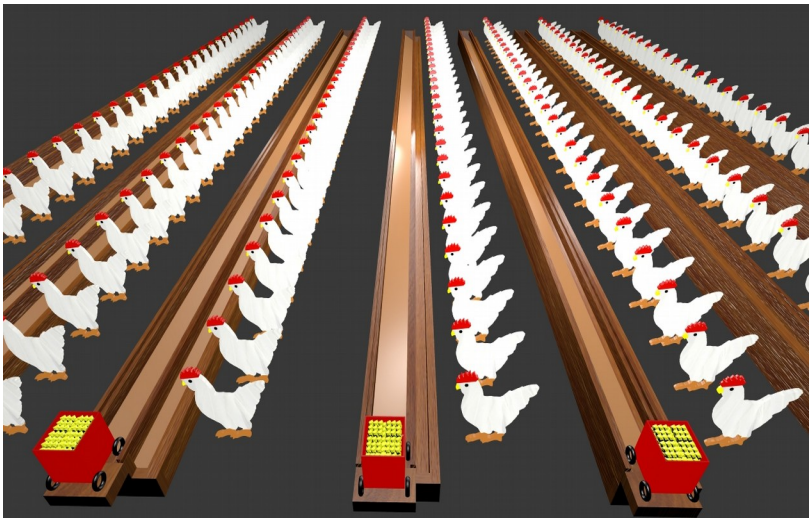
5.3. Sistema *Fog*

O sistema *Fog* é responsável por processar e administrar os dados fornecidos pelos usuários, assim como alertar as aves, por meio de efeitos sonoros, no momento em que a caixa de alimentação é abastecida. A partir do processamento dos dados, o sistema *Fog* gera as informações que serão utilizadas pelo nó *IoT*. Desse modo, o sistema *Fog* engloba toda camada de gerenciamento de dados, já que é ele quem repassará os comandos aos nós *IoT* e receberá as informações da camada de interface. Além disso, dependendo do tipo de aplicação da camada de interface, o sistema *Fog* pode ser dividido em duas partes para representar ambas as camadas. Uma aplicação de interface poderia ser um sistema *Web*, que ao invés de ser instalado no lado do cliente, como um aplicativo Android ou iOS, pode ficar no lado do servidor, se tornando, por consequência, um subsistema do sistema *Fog*.

5.4. Aplicação em larga escala

A automatização da alimentação de frangos, em granjas orgânicas, deve considerar o atendimento a um grande número de aves. A proposta do alimentador considera uma implementação em larga escala, em que vários nós *IoT* devem atuar para fornecer o alimento. Na Fig. 13 é apresentada uma ilustração em 3D em que os nós *IoT*, cada um posicionado em cima de uma caixa de alimentação própria, fazem o papel de reposição do alimento. Para uma melhor representação, na ilustração foram desenhados um grande número de frangos a espera do alimento, posicionados em frente às caixas de alimentação.

Figura 13 - Representação em 3D do alimentador inteligente aplicado em larga escala.



Fonte: PRÓPRIO (2019)

6. Desenvolvimento da proposta

Para o desenvolvimento da proposta foram criados um carrinho eletrônico, um sistema *Web* (em PHP) e um servidor (em Python). O carrinho eletrônico corresponde ao protótipo do nó *IoT*, enquanto que o sistema *Web* foi escolhido para servir como a aplicação da camada de interface. Já a camada de gerenciamento de dados conta com o servidor e com um banco de dados (MySQL), responsáveis por receber e armazenar os dados, inseridos pelos usuários, e por enviar os comandos de controle, ao nó *IoT*.

6.1. Sistema *Web*

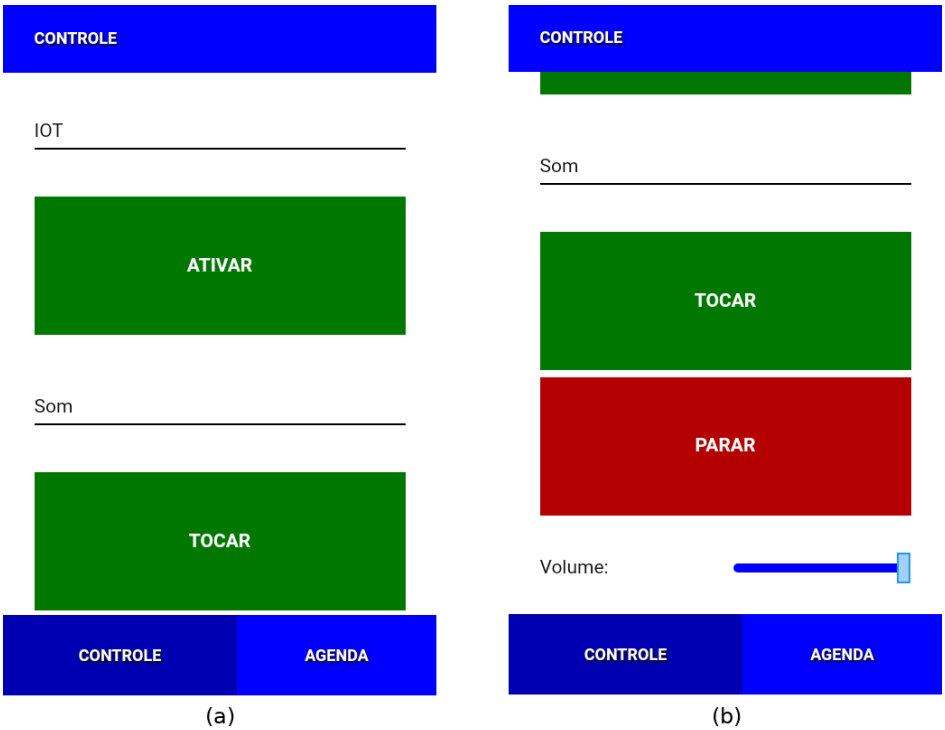
O sistema *Web* foi escolhido como a aplicação da camada de interface. O desenvolvimento levou em consideração todos os cenários de coleta de dados, desde o controle manual do nó *IoT* até o agendamento de horários das tarefas. O sistema conta, ao todo, com 3 (três) telas para interação com o usuário:

- Tela inicial (controle manual);
- Cadastro e atualização de agendamentos;
- Agendamento de horários.

6.1.1. Tela inicial: controle manual

A tela inicial é utilizada para controlar, de forma manual, o nó *IoT* e o som de alerta aos frangos. Para controlar o nó *IoT* é disponibilizado 1 (um) botão, cuja finalidade é enviar um comando ao nó *IoT* para que ele forneça o alimento para as aves. Já para o controle do som, a tela conta com dois botões, para ativar e desativar, e um campo para controlar o volume do som. O botão para ativar o nó *IoT* é mostrado na Fig. 14 (a). Já os botões de controle do som aparecem na Fig. 14 (b).

Figura 14 - Tela de controle manual do nó *IoT*.

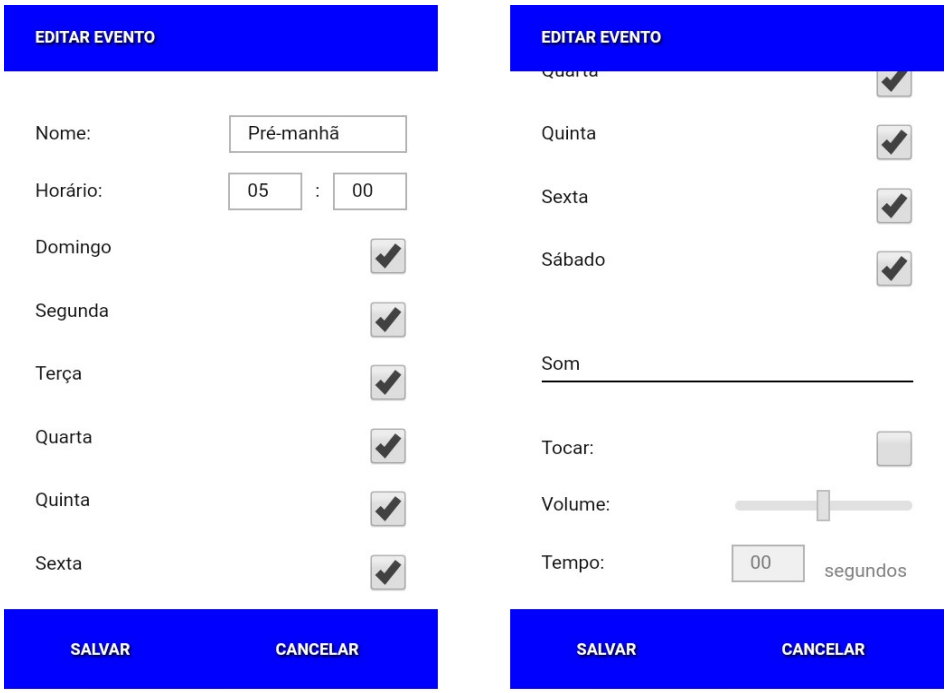


Fonte: PRÓPRIO (2019)

6.1.2. Cadastro e edição de agendamentos

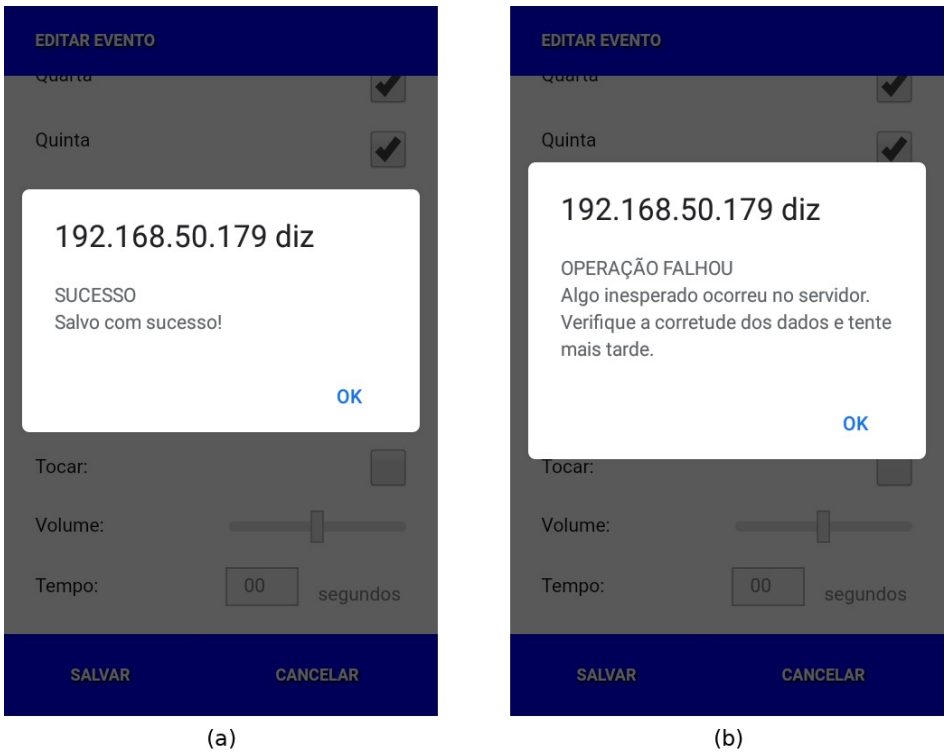
O cadastro de agendamentos considera os dados sobre os dias da semana em que um evento ocorrerá, o horário, se um efeito sonoro será disparado e o tempo e intensidade em que ocorrerá. O preenchimento de um campo para o nome também é exigido, pois facilitará a identificação dos registros quando forem apresentados aos usuários. As Fig. 15 (a) e Fig. 15 (b) mostram a tela de edição, que corresponde também a tela de cadastro. A Fig. 16 (a) e Fig. 16 (b) mostram as mensagens de sucesso e falha/erro na operação solicitada, respectivamente.

Figura 15 - Tela de cadastro e edição de agendamentos.



Fonte: PRÓPRIO (2019)

Figura 16 - Mensagens de sucesso (a) e erro/falha (b) nas operações de cadastro e edição de agendamentos.



Fonte: PRÓPRIO (2019)

6.1.3. Agenda de horários

Após a criação dos agendamentos, uma lista dos registros é disponibilizada na tela agenda de horários. A Fig. 17 mostra como ocorre a listagem, apresentando o nome, horário e os dias programados para ocorrência do evento. A tela conta com campos de seleção para remover um ou mais agendamentos. Em caso de ao menos um agendamento ser selecionado, um botão vermelho é mostrado para solicitar a remoção do registro. Já no caso de nenhum campo ser selecionado, um botão verde é utilizado para abrir a tela de cadastro de um novo agendamento. Um link fica disponível no nome do agendamento para abrir a página de edição.

Figura 17 - Tela de listagem dos agendamentos para ocorrência de eventos do nó *IoT*.

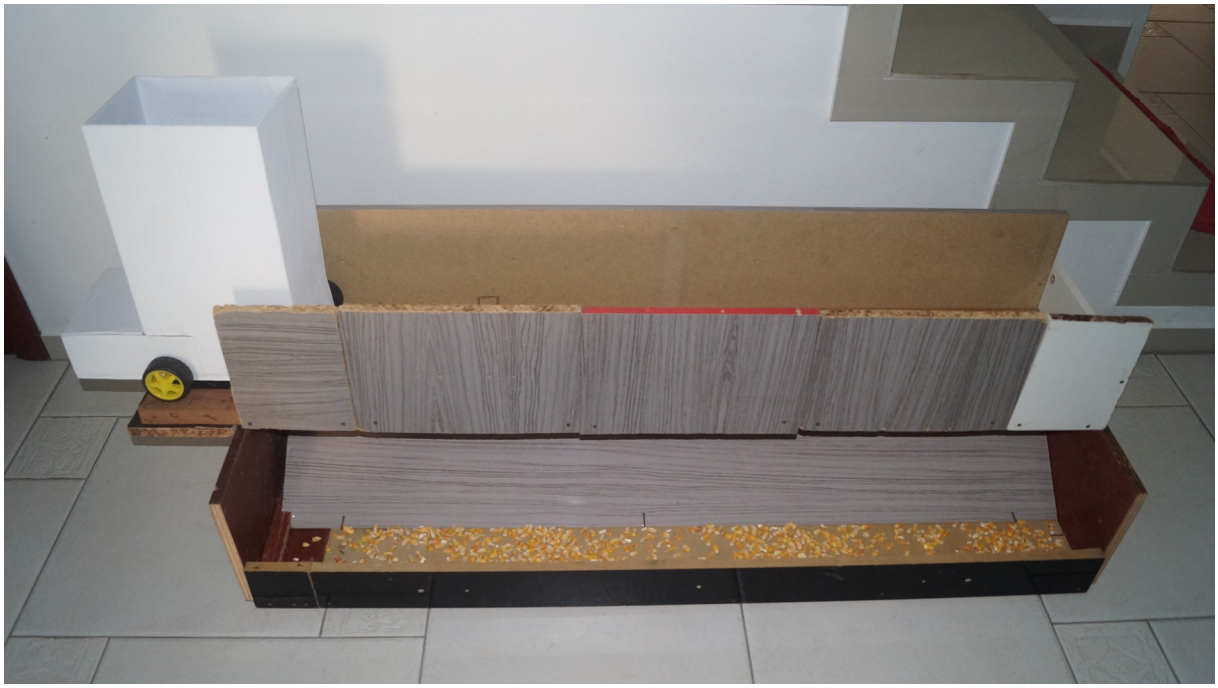
Fonte: PRÓPRIO (2019)

6.2. Protótipo nó *IoT*

A proposta deste trabalho considera a implantação de um sistema para utilização em larga escala. A implementação da proposta, contudo, considera apenas um nó *IoT*, desenvolvido como protótipo. Uma caixa de alimentação, também protótipo, foi montada para atender as necessidades do dispositivo. Na Fig. 18 é apresentada uma visão geral do

dispositivo em cima da caixa de alimentação. Já na Fig. 19 é possível a visualização geral do carrinho, sem a caixa de alimentação. Por último, a Fig. 20 mostra a parte externa superior do dispositivo, onde ficam armazenados os alimentos.

Figura 18 - Visão geral do nó *IoT* e da caixa de alimentação.

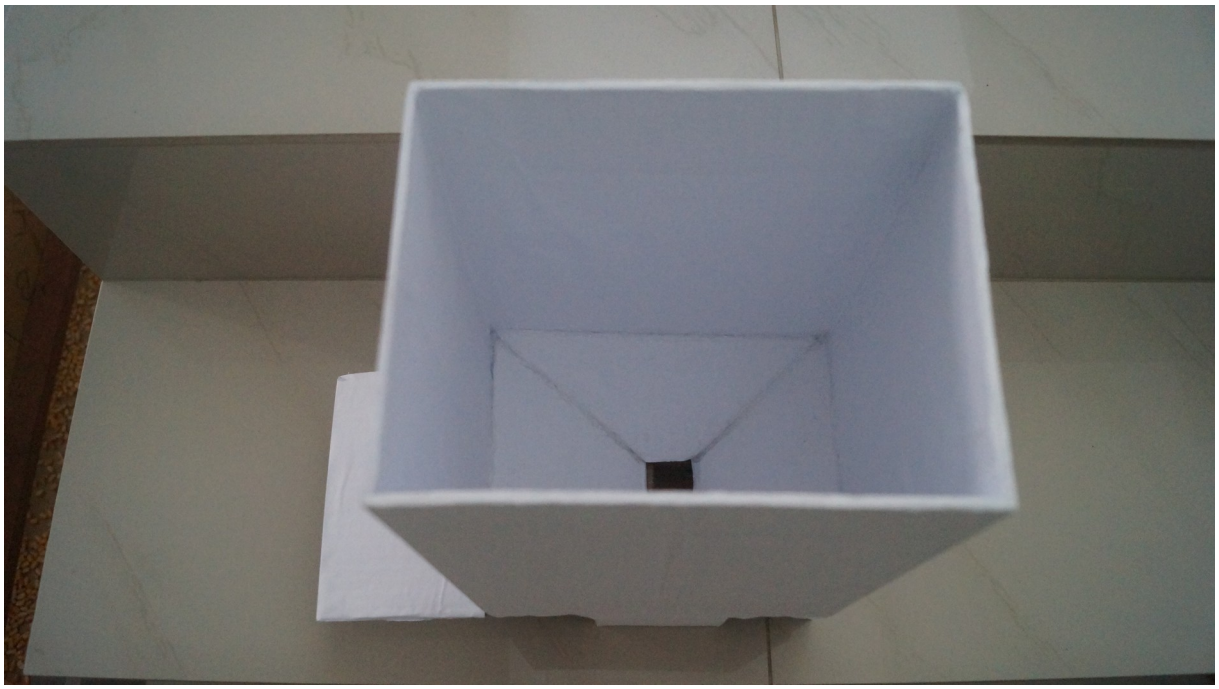


Fonte: PRÓPRIO (2019)

Figura 19 - Visão geral do nó *IoT*.

Fonte: PRÓPRIO (2019)

Figura 20 - Parte externa superior do nó *IoT*.



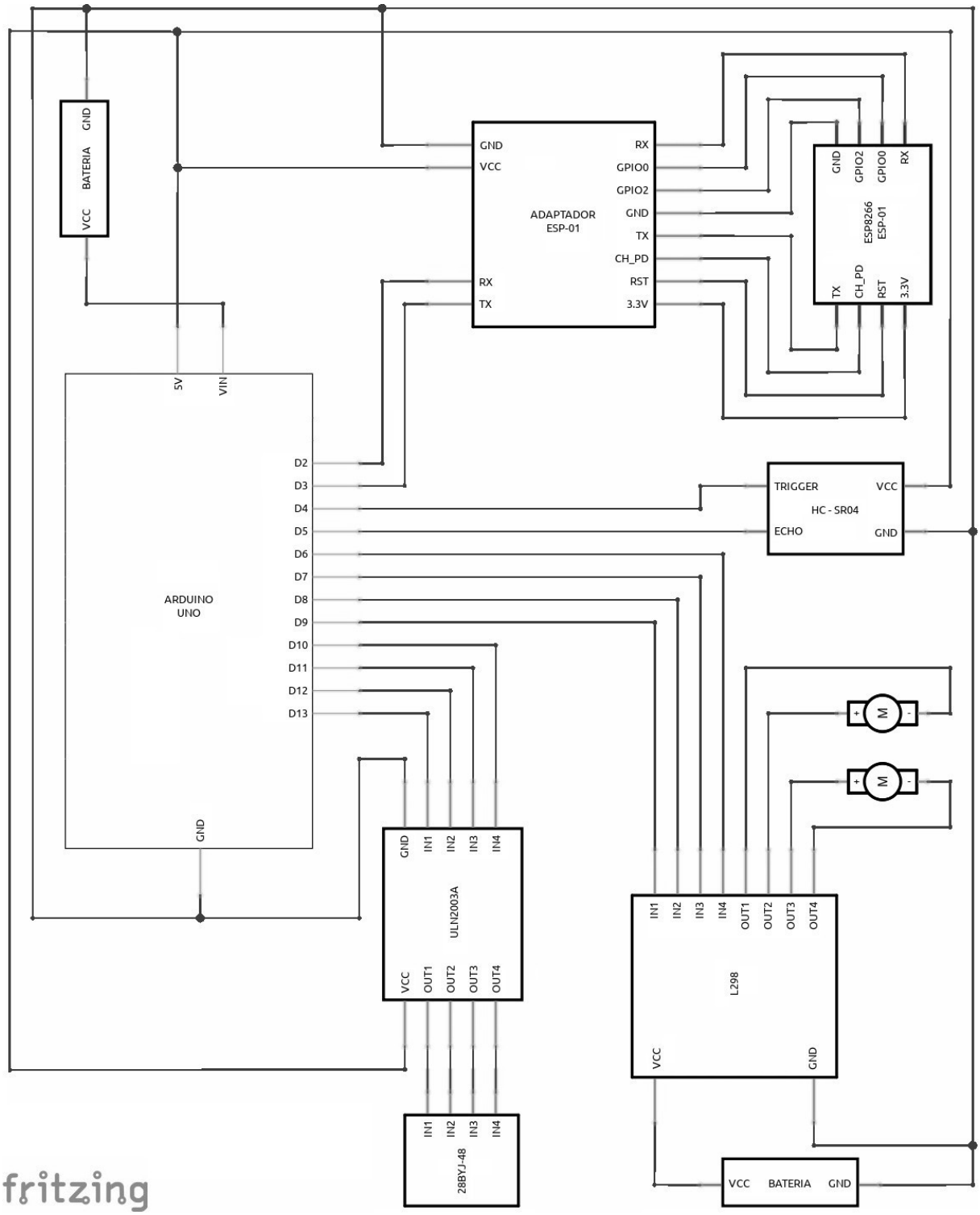
Fonte: PRÓPRIO (2019)

6.2.1. Esquema lógico do circuito eletrônico

Para o desenvolvimento do nó *IoT* foram utilizados um Arduíno (UNO), dois motores DC de 210RPM, um driver para motores DC (L298N), uma antena Wi-Fi (ESP8266 ESP-01), um driver de motores de passos (ULN2003), um motor de passos (28BYJ-48) e um sensor de distância (HC-SR04). Na parte dianteira do carrinho foram instalados os dois motores DC, cada um conectado a uma roda plástica, e o sensor de distância. As duas rodas traseiras, por não se conectarem aos motores, foram interligadas através de um eixo. O motor de passos foi posicionado no centro do dispositivo, cuja finalidade é o controle da passagem dos alimentos. Na parte externa traseira do dispositivo é fixado o Arduíno e uma fonte de 6V, formada por 5 (cinco) pilhas AA de 1.2V (2600mAh).

Na Fig. 21 é apresentado o esquema lógico do nó *IoT*. O Arduíno é conectado a antena ESP8266 ESP-01, ao driver L298N, ao driver ULN2003 e ao sensor HC-SR04. Uma fonte de 6V alimenta o driver L298N, e uma segunda fonte, também de 6V, alimenta o Arduíno. O Arduíno fica responsável por alimentar o driver ULN2003 e o sensor de distância, enquanto os motores são energizados pelos respectivos drivers.

Figura 21 - Esquema lógico do nó *IoT*.



Fonte: PRÓPRIO (2019)

6.2.2. Movimentação

O funcionamento do nó *IoT* fica sob responsabilidade do Arduíno. Ao receber o comando de acionamento, através da antena ESP8266 ESP-01, o Arduíno emite um sinal para o driver L298N para que este realize o acionamento dos motores DC. O acionamento dos motores fará com que o carrinho se movimente, em um sentido, até o sensor HC-SR04 detectar um obstáculo. Após a detecção, o Arduíno altera o sentido de movimentação do carrinho para que ele volte ao ponto de partida.

A movimentação do carrinho conta, ainda, com o uso da função millis(), da biblioteca Arduíno. O objetivo da função millis() é obter o intervalo de tempo entre a inicialização do Arduíno até o instante da chamada da função, cujos valores são fundamentais para o cálculo do tempo de movimentação. Para o cálculo, o algoritmo subtrai o valor obtido no início do trajeto pelo valor obtido no momento da detecção do obstáculo.

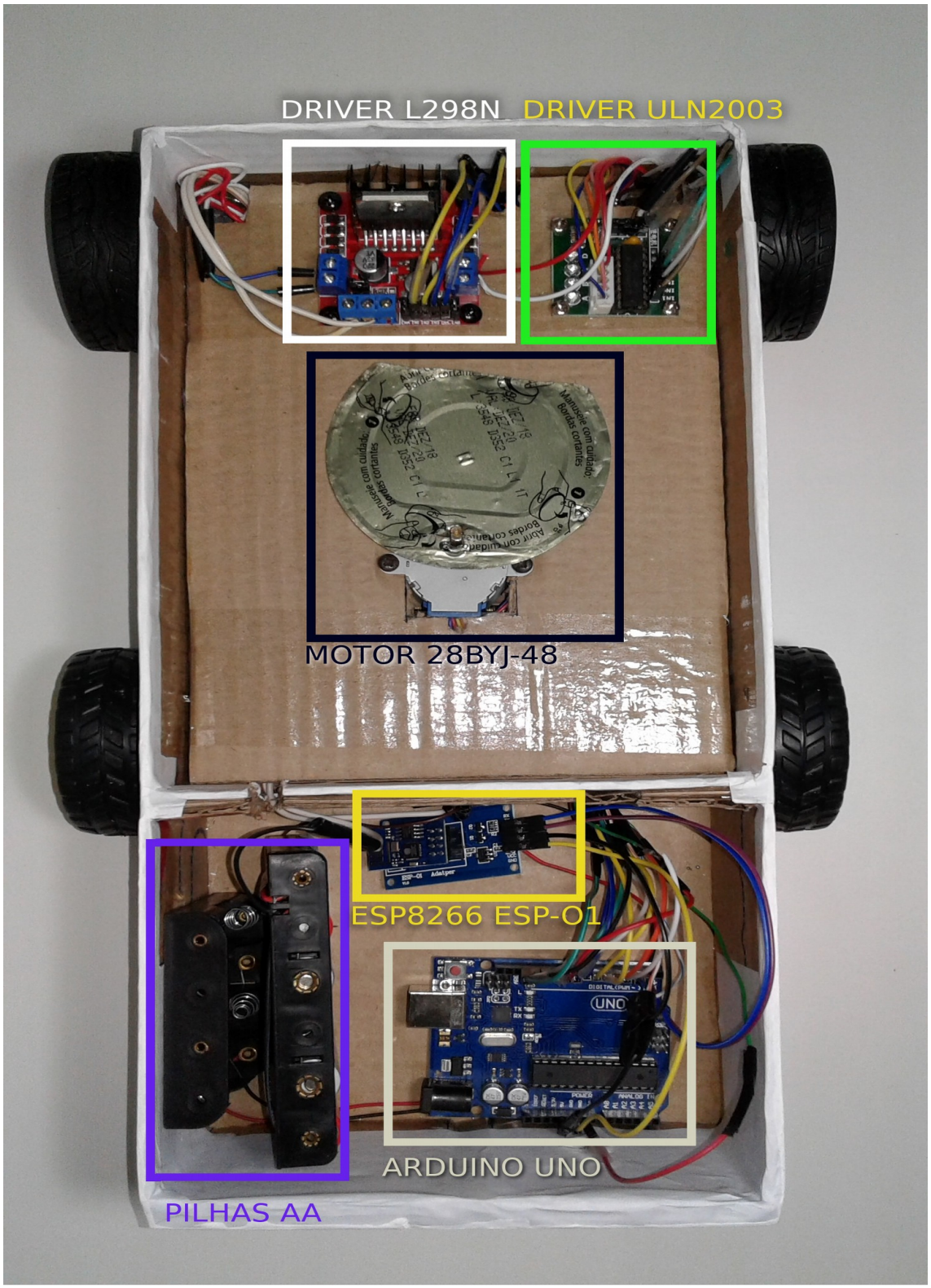
Para a implementação do retorno do dispositivo, ao ponto de origem, a função millis() é novamente utilizada. Com base no instante em que o carrinho chega ao final dos trilhos da caixa de alimentação, uma verificação, contida dentro de um loop, identificará o momento em que o tempo de movimentação atual seja igual ao calculado para ir da origem ao fim da pista. Nesse momento, considera-se que o dispositivo retornou ao ponto de origem, portanto os motores DC são desativados.

6.2.3. Reservatório

Além do acionamento dos motores DC, o Arduíno controla o driver ULN2003 para que seja aberta ou fechada a passagem do alimento. No momento em que os motores DC são acionados, o Arduíno emite um sinal ao driver ULN2003 para que o motor 28BYJ-48 faça um giro de 180 graus na horizontal, permanecendo (o motor) desativado até que o carrinho retorne ao ponto de origem. Ao retornar, o Arduíno determina que o motor 28BYJ-48 realize um giro de 180 graus no sentido inverso, e em seguida o motor é novamente desativado.

A Fig. 22 apresenta a parte interna do dispositivo, onde é possível visualizar o motor 28BYJ-48 conectado a uma placa de metal, para abrir e fechar o reservatório. Também é possível a visualização do driver L298N e do driver ULN2003, um pouco acima do motor 28BYJ-48. Abaixo do motor 28BYJ-48 estão o Arduíno, as pilhas AA e a antena ESP8266 ESP-01.

Figura 22 - Visão da parte interna do dispositivo *IoT*.



Fonte: PRÓPRIO (2019)

6.3. Servidor

O servidor, responsável pelo gerenciamento dos dados, conta com 3 (três) *Sockets TCP/IP*. O primeiro *Socket* atua para troca de dados com a interface. O segundo se destina a recepção de dados enviados pela camada *IoT*, e é utilizado para cadastrar um nó *IoT* no servidor. O último *Socket* atua como emissor, enviando os comandos de operação para todos os nós *IoT* cadastrados.

A hospedagem do servidor é feita em um Raspberry Pi 3, que conta com um módulo *Wi-Fi* para comunicação *Wireless* e entradas para conexões com dispositivos de hardware externos, como os alto falantes. Para o funcionamento da gerência de dados, o servidor conta com um banco de dados MySQL para o armazenamento de dados, recebidos da camada de interface, e processados pelo servidor. A Fig. 23 apresenta o esquema do banco de dados, que conta com a tabela evento_agendamento. Na tabela são armazenados os registros de agendamento dos dias e horários em que o nó *IoT* irá operar, além da possível emissão de sons de alerta para as aves.

Figura 23 - Esquema do banco de dados utilizado para armazenar o agendamento de eventos do nó *IoT*.

Fonte: PRÓPRIO (2019)

6.3.1. Comunicação entre interface e servidor

Para gerenciar as requisições, o servidor conta com a padronização da inserção (Fig. 24), remoção (Fig. 25), atualização (Fig. 26), seleção individual (Fig. 27) e seleção de todos os agendamentos de eventos da tabela evento. Desse modo, independente da aplicação da camada de interface, as solicitações devem seguir um modelo pré-definido para possibilitar a comunicação entre as camadas. A solicitação para seleção de todos os eventos possui apenas 1 (um) byte, correspondente ao caractere "e". Os dados, de ambas as operações, são definidos da seguinte forma:

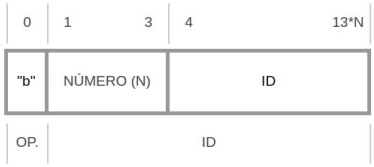
- **Operação (op):** possui 1 (um) byte (caractere) para informar o tipo de operação a ser realizada. A Tab. 2 apresenta os valores de cada operação;
- **Dias da semana (d, s, t, q, q, s, s):** correspondem, juntos, ao conjunto de 7 bytes para habilitar/desabilitar o funcionamento do carrinho nos 7 dias da semana: domingo, segunda, terça, quarta, quinta, sexta e sábado, nessa ordem. Cada um dos 7 bytes conterá o valor 0, para desabilitar, ou 1, para habilitar;
- **Horário (hora, minuto):** retém um total de 4 bytes, que representam, juntos, a hora e minuto em que o agendamento ocorrerá. Os 2 primeiros bytes se referem a hora, no intervalo entre 0 e 23, e os dois últimos aos minutos, dentro do intervalo de 0 a 59;
- **Som (habilitado, volume, tempo de duração):** se referem a 1 byte para habilitação/desabilitação do alerta sonoro, 3 bytes para o intervalo entre 0 a 100 da intensidade do volume e de 2 bytes para o intervalo de tempo entre 0 a 59 segundos;
- **ID (id):** formado por 10 bytes para representar o código único de registro do agendamento (id), no banco de dados;
- **Nome (tamanho, nome):** possui um tamanho variável de bytes, que vai de 2 a 32 bytes. Os dois primeiros bytes correspondem ao tamanho do nome (ilustrado na sequência com a letra N), que pode variar de 0 a 30 caracteres. Os bytes seguintes são preenchidos com os caracteres do nome.

Figura 24 - Mensagem de solicitação para registrar um agendamento.

0	1	2	3	4	5	6	7	8	9	10	11	12	13 15	16	17	18	19	20 N
"a"	D	S	T	Q	Q	S	S	HORA		MINUTO		HAB	VOLUME	TEMPO DE DURAÇÃO		TAMANHO (N)		NOME
OP.	DIAS DA SEMANA							HORÁRIO				SOM				NOME		

Fonte: PRÓPRIO (2019)

Figura 25 - Mensagem de solicitação para remoção de um agendamento.

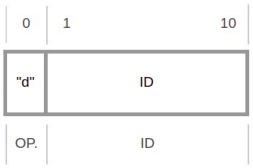


Fonte: PRÓPRIO (2019)

Figura 26 - Mensagem de solicitação para edição dos dados de um agendamento.

Fonte: PRÓPRIO (2019)

Figura 27 - Mensagem de solicitação dos dados de um agendamento.



Fonte: PRÓPRIO (2019)

Tabela 2 - Bytes de identificação enviados na mensagem da interface para o servidor.

Operação	**Byte de comando**
Inserir	a
Remover	b
Editar	c
Selecionar agendamento	d
Selecionar todos os agendamentos	e

Fonte: PRÓPRIO (2019)

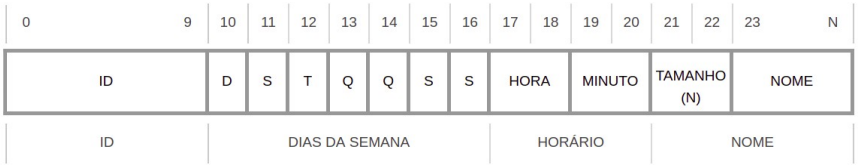
A troca de dados conta, ainda, com os padrões para o envio da resposta do servidor para a interface. Os modelos de dados são os mesmos utilizados para representar as solicitações da interface para o servidor. As respostas para solicitação de inserção, remoção e

atualização correspondem a 1 (um) byte contendo o valor 0 (zero), para erro/falha, ou o valor 1 (um), para sucesso. Já as respostas para seleção de um agendamento ou de todos os agendamentos são representadas pelas Fig. 28 e Fig. 29, respectivamente.

Figura 28 - Mensagem de resposta do servidor para a solicitação dos dados de um agendamento.

Fonte: PRÓPRIO (2019)

Figura 29 - Mensagem de resposta do servidor para a solicitação dos dados de todos os agendamentos.



Fonte: PRÓPRIO (2019)

6.3.2. Comunicação entre servidor e *IoT*

A comunicação entre o servidor e o nó *IoT* ocorre em duas etapas, sendo a primeira de configuração e a segunda de controle. A primeira etapa consiste no recebimento de uma solicitação, do nó *IoT* ao servidor, para que o servidor comece a enviar comandos de controle ao dispositivo. Ao receber a solicitação o servidor armazena o endereço de rede do dispositivo em uma lista, e assim que um comando tenha de ser enviado, o servidor o envia a todos os nós cadastrados na lista.

O comando esperado por um nó *IoT* corresponde a um byte de controle: caractere "a". O recebimento do comando força o dispositivo a percorrer a caixa de alimentação, e durante o percurso deve-se liberar o alimento armazenado. Qualquer dado diferente do caractere "a" será ignorado. Se o nó já estiver realizando uma operação, o caractere "a" também será ignorado até que a tarefa seja finalizada.

7. Conclusão e trabalhos futuros

Nessa seção são apresentadas as conclusões e os trabalhos futuros. Na subseção 7.1 são descritos os resultados obtidos com o desenvolvimento deste trabalho. Na seção 7.2 são apresentados possíveis problemas que poderiam ser solucionados futuramente, com base nos resultados apresentados nesse trabalho.

7.1. Conclusão

O objetivo deste trabalho foi contribuir com a avicultura orgânica, e para isso foram propostas ferramentas que possibilitam a automação parcial da criação de frangos. O trabalho levou em consideração o tempo exigido para alimentar as aves de forma manual, em conjunto com o crescente interesse em produtos orgânicos. Um levantamento foi realizado para encontrar ferramentas que pudessem auxiliar na criação de um alimentador de frangos, resultando em 5 (cinco) trabalhos correlatos, mencionados no Cap. 4. Considerando o levantamento realizado, a proposta e implementação de um sistema inteligente, que gerencia o fornecimento dos alimentos, foram apresentados nos Cap. 5 e Cap. 6. Considerando, ainda, o grande número de aves que devem ser atendidas, o trabalho tratou da implementação para um contexto em larga escala.

O desenvolvimento da proposta, em grande parte, acontece com o projeto de um dispositivo *IoT*. O dispositivo é responsável por cumprir com o papel de repositor do alimento dentro de uma caixa de alimentação (utilizada para alimentação das aves). Um servidor, em conjunto com um sistema *Web*, foram instalados na *Fog* e utilizados em conjunto para gerenciar o dispositivo *IoT*. O usuário cadastra as informações necessárias, através do sistema web, como os dias e horários de atuação do nó *IoT*, e o servidor as processa e se encarrega de controlar o dispositivo *IoT*.

Alguns problemas surgiram durante a etapa de implementação do protótipo, do nó *IoT*, e gerou a necessidade da realização da troca de equipamentos de hardware. Inicialmente foram utilizadas antenas NRF24L01 para comunicação entre a *Fog* e os dispositivos *IoT*, porém uma sequencia de erros levaram a substituição desta antena pela antena *Wi-Fi* ESP8266 ESP-01. Já no processo de finalização do protótipo, um problema de alimentação forçou a separação da fonte de energia entre o Arduíno UNO e os motores DC, pois ao realizar a inversão do sentido de movimentação dos motores o Arduíno sofria um processo de reinicialização.

7.2. Trabalhos futuros

A proposta para automação da alimentação de frangos, apresentadas neste trabalho, consideram o controle dos alimentos enquanto ainda estão estocados ou sendo liberados. Contudo, os cuidados necessários com as sobras não foram devidamente realizados. Desse modo, um trabalho futuro deveria considerar o gerenciamento inteligente dessas sobras, para que possíveis parasitas, como os ratos, não tenham acesso aos alimentos.

Além do aprimoramento da ferramenta, um estudo sobre o uso em outros animais poderia ser realizado. O trabalho é motivado pela criação de frangos, porém animais domésticos, como cães e gatos, talvez possam ser alimentados pelo equipamento.

Por fim, apesar do protótipo apresentado depender do uso de pilhas AA, a economia sobre o consumo de energia não foi considerada na implementação. Um estudo sobre os gastos realizados, além de medidas para diminuição desses gastos, poderiam ser exploradas para melhorar a eficiência do dispositivo.

REFERÊNCIAS

ALVES, S. P.; SILVA, I. J. O.; PIEDADE, S. M. S. Avaliação do bem estar de aves poedeiras comerciais: efeitos do sistema de criação e do ambiente bioclimático sobre o desempenho das aves e a qualidade de ovos. *Revista Brasileira de Zootecnia*, v. 36, n. 5, p. 1388-1394, 2007.

AMMAD-UDDIN, M.; AYAZ, M.; AGGOUNE, E-H.; SAJJAD, M. Wireless sensor network: A complete solution for poultry farming. *2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT)*, p. 321-325, 2014.

ANOMALY, J. What's Wrong With Factory Farming? *Public Health Ethics*, v. 8, n. 3, p. 246-254, 2015.

APAOLAZA, V.; HARTMANN, P.; SOUZA, C.; LÓPEZ, C. M. Eat organic – Feel good? The relationship between organic food consumption, health concern and subjective wellbeing. *Food Quality and Preference*, v. 63, p. 51-62, 2018.

Avicultura sofrerá com mudanças climáticas - Portal Embrapa. Disponível em: <www.embrapa.br/busca-de-noticias/-/noticia/7546116/avicultura-sofrera-com-mudancas-climaticas>. Acesso em ago. 2018.

BARACHO, M. S.; NAAS, I. A.; BETIN, P. S.; MOURA, D. J. Factors that Influence the Production, Environment, and Welfare of Broiler Chicken: A Systematic Review. *Brazilian Journal of Poultry Science*, v. 20, n. 3, p. 617-624, 2018.

BERG, C. Health and Welfare in Organic Poultry Production. *Acta Veterinaria Scandinavica*, v. 43, n. 1, p. 37-45, 2002.

BÍBLIA. Português. Bíblia Sagrada. Tradução de João Ferreira de Almeida. Barueri - SP: Sociedade Bíblica do Brasil, 1993.

BRASIL. Decreto Federal 6. 323, de 27 de dezembro de 2007. Regulamenta a Lei no 10.831, de 23 de dezembro de 2003, que dispõe sobre a agricultura orgânica, e dá outras providências.

BRASIL. Lei nº 10831, de 23 de dezembro de 2003. Diário Oficial da República Federativa do Brasil, Poder Executivo, Brasília, DF, 23 dez. 2003. Seção 1, p. 8.

CASSUCE, D. C.; TINÔCO, I. F. F.; BAÊTA, F. C.; ZOLNIER, S.; CECON, P. R.; VIEIRA, M. F. A. Thermal comfort temperature update for broiler chickens up to 21 days of age., v. 33, n. 1, p. 28-36, 2013.

CHEN, Y.; AZHARI, M. Z.; LEU, J. Design and implementation of a power consumption management system for smart home over fog-cloud computing View Document. *2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, 2018.

CHIEN, Y-R.; CHEN, Y-X. An RFID-Based Smart Nest Box: An Experimental Study of Laying Performance and Behavior of Individual Hens. *SENSORS 2018*, v. 18, n. 3, p. 859-870 , 2018.

CHIEOCHAN, O.; SAOKAEW, A.; BOONCHIENG, E. IOT for smart farm: A case study of the Lingzhi mushroom farm at Maejo University. *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2017.

DAMASCENO, F. A.; CASSUCE, D. C.; ABREU, L. H. P.; SCHIASSI, L.; TINÔCO, I. F. F. Effect of thermal environment on performance of broiler chickens using fuzzy modeling. *Revista Ceres*, v. 64, n. 4, p. 337-343, 2017.

DIAS, A. N.; MACIEL, M. P.; AIURA, A. L. O.; AROUCA, C. L. C.; SILVA, D. B.; MOURA, V. H. S. Linhagens de frangos caipiras criadas em sistema semi-intensivo em região de clima quente. *Pesquisa Agropecuária Brasileira*, v. 51, n. 12, p. 2010-2017, 2016.

FELTES, M. M. C.; ARISSETO-BRAGOTTO, A. P.; BLOCK, J. M. Food quality, food-borne diseases, and food safety in the Brazilian food industry. *Food Quality and Safety*, v. 1, n. 1, p. 13-27, 2017.

FIGUEIREDO, E. A. P.; SOARES, J. P. G. Sistemas orgânicos de produção animal: dimensões técnicas e econômicas. *Anais da 49a Reunião Anual da Sociedade Brasileira de Zootecnia*, 2012.

GIFFORD, K.; BERNARD, J. C. The effect of information on consumers' willingness to pay for natural and organic chicken. *International Journal of Consumer Studies*, v. 35, n. 3, p. 282-289, 2011.

GUPTA, H.; DASTJERDI, A. V.; GHOSH, S. K.; BUYYA, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Journal of Software: Practice and Experience*, v. 47, n. 9, p. 1275-1296, 2017.

HONKANEN, P.; VERPLANKEN, B.; OLSEN, S. O. Ethical values and motives driving organic food choice. *Journal of Consumer Behaviour*, v. 5, n. 5, p. 420-430, 2006.

KHALIL, I. M.; KHREISHAH, A.; AZEEM, M. Cloud Computing Security: A Survey. *Computers*, v. 3, n. 1, p. 1-35, 2014.

LOUZADA, M. L. C.; RICARDO, C. Z.; STEELE, E. M.; LEVY, R. B.; CANNON, G.; MONTEIRO, C. A. The share of ultra-processed foods determines the overall nutritional quality of diets in Brazil. *Public Health Nutrition*, v. 21, n. 1, p. 94-102, 2017.

MEAH, K.; FORSYSTH, J.; MOSCOLA, J. A Smart Sensor Network for an Automated Urban Greenhouse. *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, p. 23-27, 2019.

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*, 2011.

MOURA, G. R. S.; MENDONÇA, M. O.; SALGADO, H. R.; CASTRO, J. O.; SOUZA, R. T. Galinhas semipesadas em postura criadas sobre diferentes tipos de cama. *Revista Brasileira de Saúde e Produção Animal*, v. 18, n. 2, p. 378-387, 2017.

MUKHERJEE, M.; SHU, L.; WANG, D. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. *IEEE Communications Surveys & Tutorials*, v. 20, n. 3, p. 1826-1857, 2018.

MUNIR, A.; KANSAKAR, P.; KHAN, S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. *IEEE Consumer Electronics Magazine*, v. 6, n. 3, p. 74-82, 2017.

NAVANEETH, B. S.; MURTY, A. K. AUTOMATIC POULTRY FEEDER. *International Journal of Advance Engineering and Research Development*, v. 2, n. 7, p. 338-343, 2015.

PERALA, S. S. N.; GALANIS, I.; ANAGNOSTOPOULOS, I. Fog Computing and Efficient Resource Management in the era of Internet-of-Video Things (IoVT). 2018 *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

PEREIRA, D. C. O.; MIRANDA, K. O. S.; FILHO, L. C. D.; PEREIRA, G. V.; PIEDADE, S. M. S.; BERNO, P. R. Presence of roosters in an alternative egg production system aiming at animal welfare. *Brazilian Journal of Animal Science*, v. 46, n. 3, p. 175-184, 2017.

PSD Online. Disponível em: <apps.fas.usda.gov/psdonline/app/index.html#/app/downloads>. Acesso em ago. 2018.

RAY, P. P. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, v. 30, n. 3, p. 291-319, 2018.

ROCHA, J. S. R.; LARA, L. J. C.; BAIÃO N. C. PRODUÇÃO E BEM-ESTAR ANIMAL - ASPECTOS ÉTICOS E TÉCNICOS DA PRODUÇÃO INTENSIVA DE AVES. *Ciência Veterinária nos Trópicos*, v. 11, n. 1, p. 49-55, 2008.

ROMAN, R.; LOPEZ, J.; MASAHIRO, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, v. 78, n. 2, p. 680-698, 2018.

ROSSA, L. S.; STERTZ, S. C.; MACEDO, R. E. F. Regulamentação, mercado e qualidade da carne de frango orgânico no Brasil – Revisão. *Revista Acadêmica: Ciência Animal*, v. 10, n. 1, p. 29-44, 2012.

SAMPAIO, H. V.; JESUS, A. L. C.; BOING, R. N.; WESTPHALL, C. B. Autonomic IoT Battery Management with Fog Computing. GPC: *International Conference on Green, Pervasive, and Cloud Computing*, v. 11484, p. 89-103, 2019.

SILVA, M. A. N.; FILHO, P. H.; ROSÁRIO, M. F.; COELHO, A. A. D.; SAVINO, V. J. M.; GARCIA, A. A. F.; SILVA, I. J. O.; MENTEN, J. F. M. Influência do Sistema de Criação sobre o Desempenho, a Condição Fisiológica e o Comportamento de Linhagens de Frangos para Corte. *Revista Brasileira de Zootecnia*, v. 32, n. 1, p. 208-213, 2003.

SINGH, S.; JEONG, Y-S.; PARK, J. H. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, v. 75, p. 200-222, 2016.

SOH, Z. H. C.; ISMAIL, M. H.; OTTHAMAN, F. H.; SAFIE, M. K.; ZUKRI, M. A. A.; ABDULLAH, S. A. C. Development of automatic chicken feeder using Arduino Uno. *2017 International Conference on Electrical, Electronics and System Engineering (ICEESE)*, p. 120-124, 2017.

STERGIOU, C.; PSANNIS, K. E.; KIM, B-G.; GUPTA, B. Secure integration of IoT and Cloud Computing. *Future Generation Computer Systems*, v. 78, n. 3, p 964-975, 2018.

VOGADO, G. M. S.; VOGADO, K. T. S.; FONSECA, W. J. L.; FONSECA, W. L.; VOGADO, W. F.; OLIVEIRA, A. M.; OLIVEIRA, N. M.; LUZ, C. S. M. EVOLUÇÃO DA AVICULTURA BRASILEIRA. *Nucleus Animalium*, v. 8, n. 1, p. 49-58, 2016.

XU, L. D.; HE, W.; LI, S. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, v. 10, n. 4, p. 2233-2243, 2014.

WU, W-C.; CHENG, K-C.; LIN, P-Y. A remote pet feeder control system via MQTT protocol. *2018 IEEE International Conference on Applied System Invention (ICASI)*, p. 487-489, 2018.

ZANDER, K.; HAMM, U. Consumer preferences for additional ethical attributes of organic food. *Food Quality and Preference*, v. 21, n. 5, p. 495-503, 2010.

ZANELLA, A.; BUI, N.; CASTELLANI, A.; VANGELISTA, L.; ZORZI, M. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, v. 1, n. 1, p. 22-32, 2014.

ZANINELLI, M.; REDAELLI, V.; TIRLONI, E.; CRISTIAN, B.; DELL'ORTO, V.; SAVOINI, G. First Results of a Detection Sensor for the Monitoring of Laying Hens Reared in a Commercial Organic Egg Production Farm Based on the Use of Infrared Technology. *SENSORS*, v. 16, n. 10, 2016.

APÊNDICE A – Artigo no formato SBC

Gerenciamento Autonômico da Alimentação de Frangos em Aviários de Criação Alternativa com Fog e IoT

Ricardo N. Boing[1]

[1]Departamento de Informática e Estatísticas
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

ricardoboing.ufsc@gmail.com

Abstract. *The quality and food safety are often the subject of discussions, mainly due to nutrient shortages or because they are a means of disease transmission. Concerned about human health and animal welfare, consumers are beginning to choose organic food. Poultry farming is one of the means of obtaining meat in the world. It has part of its market focused on organic farming. Considering the interest in organic products, this work focuses on the automation of environments for the alternative creation of chickens. Based on IoT and Fog Computing paradigms, an intelligent feeder is proposed, with objective to supply food to the birds, in an autonomous way.*

Resumo. *A qualidade e a segurança alimentar são frequentemente alvos de discussões, principalmente devido à escassez de nutrientes ou por serem um meio de transmissão de doenças. Preocupados com a saúde humana e com a garantia do bem-estar animal, os consumidores começam a optar pelo consumo de alimentos orgânicos. A avicultura, um dos principais meios de obtenção de carne no mundo, tem parte do seu mercado voltado à criação orgânica. Considerando o interesse nos produtos orgânicos, este trabalho tem como foco a automatização de ambientes para criação alternativa de frangos. Com base nos paradigmas IoT e Fog Computing, é proposto um alimentador inteligente, cujo objetivo é abastecer de forma autonômica os locais de alimentação das aves.*

1. Introdução

A agricultura industrial, por conta dos seus baixos custos de produção, vem crescendo cada vez mais em nível mundial. Dentre os assuntos decorrentes está a criação intensiva de aves, que são confinadas dentro de ambientes fechados, praticamente sem espaço para se locomoverem. Especialistas relatam que nessas condições as aves vivem estressadas e perdem os seus instintos naturais, possibilitando o surgimento de diversos tipos de doenças. Muitos profissionais criticam a adoção do sistema, afirmando que são cruéis e causam prejuízos à saúde do animal, mais tarde repassados ao consumidor (ANOMALY, 2015).

Segundo a Organização Mundial de Saúde (OMS), citada por FELTES (2017), estima-se que em 2010 quase uma em cada dez pessoas adoeceram no mundo por contaminação alimentar. Crianças, idosos e gestantes, por possuírem baixas condições físicas e imunológicas, estão entre os mais propensos a contaminação. O Brasil é um dos interessados em combater esse mal, já que é um grande fornecedor de alimentos. Em 2015, o país foi o segundo maior produtor e exportador de carne de frangos no mundo.

De acordo com LOUZADA (2017), a industrialização é um assunto de fundamental importância para questão alimentar. De acordo com sua publicação, a causa de doenças como

diabetes, obesidade e câncer estão fortemente ligadas com a ingestão de alimentos industrializados ultraprocessados. LOUZADA (2017) destaca, ainda, a importância das vitaminas e sais minerais (adequados) para saúde, os quais nem sempre podem ser obtidos a partir desse tipo de alimento.

1.1. Motivação

A avicultura ocupava o segundo lugar no ranking mundial de produção de carnes, até o ano de 2010, com um total equivalente a quase 72 milhões de toneladas produzidas ao ano. Sendo um dos principais exportadores de carnes de frango, o Brasil produziu naquele ano o equivalente a pouco mais de 12 milhões de toneladas (ROSSA, 2012). Entre 2014 e outubro de 2018 o Brasil se tornou o maior exportador mundial de frangos e o segundo maior produtor mundial da carne, tendo como seu principal concorrente os EUA, que nesse mesmo período ficou em segundo colocado no número de exportações e como primeiro no número de produção (PSD Online, 2018).

1.2. Justificativas

Apesar do crescimento da agricultura industrial, um número cada vez maior de consumidores vem optando pelo consumo de alimentos orgânicos. ZANDER (2010), em seus estudos, afirma existir um descontentamento das pessoas em relação ao modo convencional de produção. Segundo ZANDER (2010) e APAOLAZA (2018) isso se justifica principalmente pela preocupação relacionada à saúde e bem-estar humano e animal. Isso porque muitas pessoas acreditam ser mais saudável o consumo de alimentos orgânicos e consideram importante o tratamento dado aos animais que serão abatidos.

1.3. Objetivos

Nesta subseção são descritos os objetivos deste trabalho. Primeiramente é apresentado uma descrição geral sobre o assunto, e, na sequência, são enumerados de forma mais específica cada um dos objetivos.

1.3.1. Objetivo geral

Com base na criação orgânica de frangos, esse trabalho possui como objetivo o desenvolvimento de um sistema para automatizar a reposição de alimentos em ambientes de confinamento das aves. O sistema deve ser projetado de acordo com as leis em vigor, destinadas ao desenvolvimento orgânico de animais, no Brasil, as quais visam manter o bem-estar físico, psicológico, comportamental, ambiental e nutricional das aves no decorrer de todas as etapas de criação.

1.3.2. Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Manter os espaços de alimentação abastecidos;
2. Emitir sinais para alertar as aves quando o alimentador for reabastecido;
3. Fornecer meios para cadastrar dias/horários para o reabastecimento automático do alimentador;
4. Fornecer uma forma de controlar manualmente o alimentador, sem a necessidade do cadastro de dias e horários.

1.4. Método de pesquisa e trabalho

A seguir são apresentados os métodos de trabalho.

1. Reunir informações sobre o processo de criação de frangos e as diferenças entre os sistemas intensivo (tradicional) e semi-intensivo (caipira e orgânico);
2. Pesquisar sobre as exigências das leis nacionais que possam influenciar no processo de desenvolvimento do sistema;
3. Realizar um levantamento sobre os modos de criação de frangos, atualmente em vigor, que sejam relevantes para o projeto;
4. Pesquisar por tecnologias que estejam em uso, que possam ser aprimoradas ou que sirvam de base para criação de novas soluções;
5. Desenvolver a proposta com base nas tecnologias e informações encontradas;
6. Implementar um protótipo para demonstração da proposta.

1.5. Organização do trabalho

Esse documento está organizado em 7 Seções. Na Seção 2 são descritos os principais conceitos e paradigmas computacionais, que foram considerados para implementação da proposta. A Seção 3 apresenta o estado da arte na área avícola. Na Seção 4 é feita uma breve descrição sobre cinco artigos, que servirão como base para o desenvolvimento da proposta. Na Seção 5 é apresentada a proposta, cuja implementação é mostrada na Seção 6. Por fim, a Seção 7 destina-se a conclusão e sugestões para trabalhos futuros.

2. Conceitos básicos

Neste capítulo são apresentados os principais conceitos, da área de computação, que serão utilizados no decorrer do trabalho, sendo eles: *Internet of Things*, *Cloud Computing* e *Fog Computing*.

2.1. *Internet of Things*

Internet of Things (*IoT*) é um conceito de redes de computadores que introduz objetos do mundo real ao virtual, com o propósito de possibilitar a comunicação entre eles. Dentro dessa rede encontram-se sensores, atuadores, computadores pessoais e qualquer outro dispositivo considerado inteligente (MUNIR, 2017). São esses dispositivos que capturam dados e informações, que posteriormente são utilizados para fornecer melhores serviços à população, em áreas residenciais, hospitalares, industriais (ZANELLA, 2014), agrícolas, dentre outros. Cada objeto possui uma identidade e atributos próprios e podem se conectar a *Internet*, portanto, para acessá-los não é preciso necessariamente que exista uma conexão de rede local (XU, 2014).

Para facilitar e simplificar a usabilidade, um sistema *IoT* pode ser dividido em vários blocos funcionais. Um exemplo é mostrado em RAY (2018), que realiza a divisão em seis blocos, os quais são descritos na sequência do texto.

- **Dispositivo:** são "coisas" do nosso dia a dia, como relógios inteligentes, veículos, máquinas e roupas. Fazem parte dos objetivos de um dispositivo a realização de coleta de dados, envio de dados para servidores, comunicação com outros dispositivos e/ou aplicativos. A maioria dos dispositivos realizam o processamento de dados para gerar informações úteis, posteriormente utilizadas para alcançar um dado objetivo. Um

exemplo é o processo de monitoramento de um jardim, em que são determinados os melhores horários para realizar a irrigação;

- **Comunicação:** é responsável pela comunicação entre dois ou mais dispositivos e entre dispositivos e servidores remotos. Os protocolos *IoT* funcionam, na maioria das vezes, na camada de enlace, rede, transporte e aplicação;
- **Serviços:** representam as funcionalidades do sistema *IoT*. Como serviços podem ser considerados a modelagem, busca e controle de dispositivos, ou a publicação e análise de dados;
- **Gerenciamento:** fornece ferramentas para gerenciar um sistema *IoT*;
- **Segurança:** disponibiliza mecanismos para garantir a proteção do sistema *IoT*, como a autenticação, autorização, privacidade e a integridade e segurança de dados;
- **Aplicação:** fornece meios para integração do usuário ao sistema *IoT*. A integração ocorre através de interfaces que possibilitam controlar e monitorar o sistema.

Uma rede *IoT* pode servir para solucionar vários tipos de problemas. STERGIOU (2018) cita alguns casos em que se faz o uso do paradigma:

- **Transporte:** auxilia na obtenção de soluções de problemas relacionados ao tráfego em rodovias, redução do consumo de combustível e a diminuição nos índices de mortes;
- **Redes elétricas:** são utilizadas para auxiliar na incorporação de energia renovável, melhorando a confiabilidade do sistema e tornando a energia mais barata;
- **Monitoramento remoto de pacientes hospitalares:** proporciona uma maior facilidade para atender os pacientes, gerando melhores condições e o aumento do número de pacientes atendidos;
- **Sensores de monitoramento de motores:** possibilitam a detecção de problemas de manutenção, tornando possível melhorar a reposição de peças e definir prioridades no agendamento de serviços.

1.3.2. *Cloud Computing*

Cloud Computing é um conceito computacional que fornece uma gama de recursos a outros computadores e dispositivos. Os recursos fornecidos pela *Cloud* ficam disponíveis de forma compartilhada, a qualquer hora e lugar, e são acessados através da *Internet*. Como exemplo tem-se o processamento e armazenamento de dados, serviços, aplicativos (MUNIR, 2017) e servidores. Por conta dos seus benefícios e da possibilidade de cobrança financeira, o paradigma deu origem a uma indústria multibilionária crescente em todo o mundo (ROMAN, 2018). Dentre as principais classificações de uma *Cloud* estão (SINGH, 2016):

- ***Public Cloud* (nuvem pública):** nesse modelo os recursos e serviços são públicos, portanto, qualquer organização, empresa, ambiente acadêmico ou indivíduo tem acesso. Por não existir um controle de clientes e fornecedores, existe uma maior vulnerabilidade em relação a segurança;
- ***Community Cloud* (nuvem comunitária):** é utilizada e gerenciada por um grupo mais restrito de organizações. Por possuírem um acesso mais controlado, os riscos relacionados de segurança são reduzidos;
- ***Private Cloud* (nuvem privada):** o uso e gerenciamento é exclusivo da organização a qual pertence. Por existir uma maior facilidade na identificação de clientes e os

fornecedores, os riscos relacionados a segurança são menores do que os modelos público e comunitário;

- ***Hybrid Cloud* (nuvem híbrida):** representa a união de dois ou mais modelos de nuvens (pública, privada e comunitária).

O National Institute of Standards and Technology (NIST) descreve três modelos de infraestrutura (MELL, 2011):

- ***Software as a Service (SaaS)*:** os aplicativos são acessados por meio de uma infraestrutura da própria Cloud. Os usuários fazem o acesso através de interfaces, como um Browser ou algum outro software, porém não possuem acesso a qualquer tipo de gerenciamento de infraestrutura, tais como rede, sistema operacional, e o armazenamento de dados. O gerenciamento de aplicativos também possuem acesso limitado, podendo existir algumas exceções;
- ***Platform as a Service (PaaS)*:** fornece a possibilidade de hospedar aplicativos desenvolvidos pelo próprio usuário. Assim como em SaaS, o cliente não tem acesso ao gerenciamento da infraestrutura, porém possui liberdade para gerenciar os próprios aplicativos;
- ***Infrastructure as a Service (IaaS)*:** são disponibilizados recursos de processamento, armazenamento, redes e outros recursos de computação. O usuário pode instalar sistemas operacionais e hospedar aplicativos, porém não controla a infraestrutura da nuvem.

A Fig. 1 mostra a relação entre as classificações e os modelos de *Cloud Computing*. Os três modelos (*IaaS*, *PaaS* e *SaaS*) são utilizados em *Public*, *Community*, *Private* e *Hybrid Cloud*.

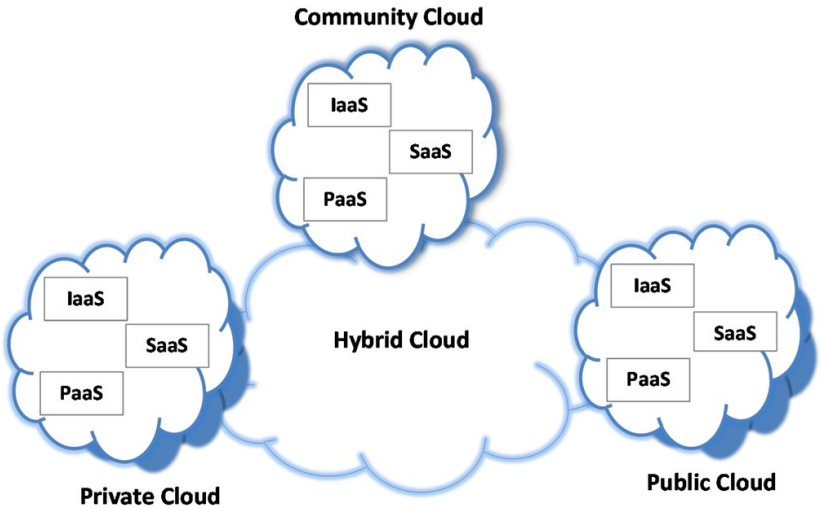


Figura 1. Representação dos modelos de infraestrutura e classificações de nuvens.
Fonte: KHALIL (2014)

2.3. *Fog Computing*

Fog Computing surgiu como um meio alternativo para intermediar a comunicação entre os dispositivos *IoT* e a *Cloud* (PERALA, 2018). O paradigma surgiu em 2014, criado pela Cisco (CHEN, 2018), e se caracteriza pelo fornecimento de serviços da *Cloud* dentro da borda da rede. O objetivo é aumentar o desempenho das aplicações ao processar e armazenar parte dos

dados localmente, enviando para *Cloud* somente o necessário. Os resultados podem ser notados pela diminuição do tráfego de rede, redução no tempo de espera por serviços (GUPTA, 2017; PERALA, 2018; MUNIR, 2017), agilidade, acesso via rede sem fio e aplicações em tempo real (MUKHERJEE, 2018). Por consequência, é possível usar a *Cloud* para realizar outros serviços de maior complexidade, como aprendizado de máquina e processamento Big Data (CHEN, 2018).

A Fig. 2 ilustra a relação entre os dispositivos *IoT* e os serviços *Fog* e *Cloud Computing*. Os dispositivos realizam a troca de dados com a *Fog*, que por sua vez processa e envia/recebe apenas o necessário para *Cloud*.

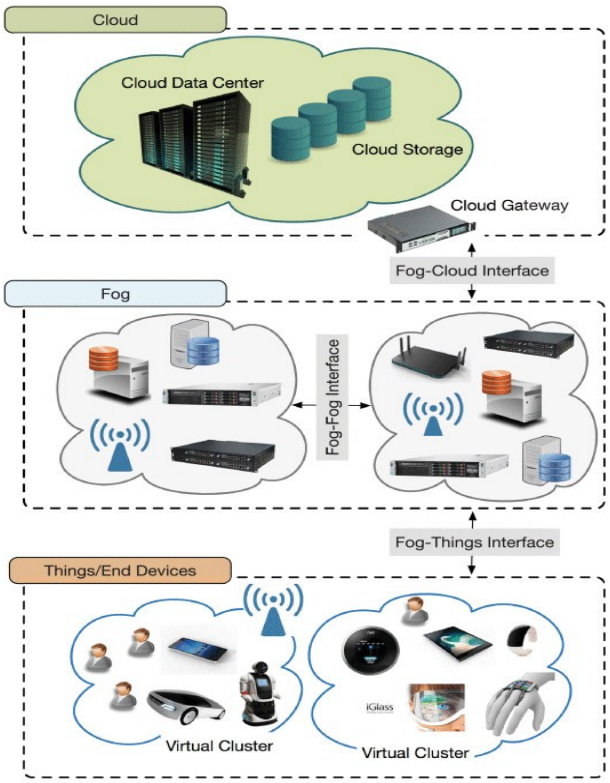


Figura 2. Representação da relação entre os conceitos *Cloud, Fog* e *IoT*.
Fonte: MUKHERJEE (2018)

3. Produtos orgânicos e avicultura

Nesta seção são apresentados os conceitos fundamentais da área avícola e as diferentes formas de criação de frangos, sendo elas: produtos orgânicos, alimentos orgânicos no Brasil, avicultura intensiva e avicultura orgânica e caipira. Também são levantados os principais parâmetros utilizados para o controle de um galinheiro orgânico autonômico.

3.1. Produtos orgânicos

A classificação de alimentos como sendo orgânicos é considerada uma forma de dar credibilidade ao produto, pois além das informações visíveis na embalagem, o selo orgânico tende a comunicar ao consumidor que o produto atende a normas e padrões específicos de qualidade. Essas informações tem o propósito de dar mais transparência para o consumidor do que àquelas contidas em produtos convencionais (GIFFORD, 2011), e essas normas podem variar de um país para outro. De modo geral tem-se por objetivo a produção de alimentos que busquem manter o equilíbrio ecológico, evitando o uso de pesticidas, herbicidas, fertilizantes

inorgânicos, antibióticos e hormônios de crescimento, em todas as etapas de produção (HONKANEN, 2006).

3.2. Avicultura intensiva

Caracterizado pelo confinamento total das aves, o sistema intensivo é o modelo de criação mais empregado na avicultura. O objetivo desse sistema é o aumento da produção e sua eficiência, obtidos através de melhorias genéticas, aprimoramento na coleta de dados e instalação de novas tecnologias de manejo, nutrição, bem estar e higiene (VOGADO, 2016). Contudo, manter as aves confinadas é prejudicial a saúde, e ao contrário a esses investimentos tecnológicos, o sistema não converge para um bem-estar dos animais (SILVA, 2003).

No Brasil, a produção intensiva, reconhecida mundialmente, teve um forte investimento tecnológico em seu processo de produção, aliado a mudanças no controle sanitário e melhorias na genética das aves. Algumas medidas de melhoramento genético foram: Inseminação Artificial (IA); Transferência de Embriões (TE); micro manipulação e produção in vitro de embriões; clonagem e produção de animais transgênicos ou altamente modificados. A diminuição no tempo de crescimento dos frangos está entre os resultados desse investimento, que passou de 105 dias em 1930 para 45 dias em 1996 (VOGADO, 2016).

Com os avanços na avicultura industrial foram surgindo modificações nos hábitos alimentares e nas tradições da população. O frango caipira foi ao longo do tempo perdendo espaço na mesa do consumidor. Contudo, mesmo sendo substituído, existe uma maior preferência pela carne caipira devido suas características próprias na textura e no sabor da carne (VOGADO, 2016).

Apesar do sistema convencional ser o dominante no mercado, algumas formas alternativas, como a caipira e orgânica, estão voltando a ganhar espaço na mesa dos consumidores. A fim de minimizar os impactos do sistema industrial e melhorar o bem-estar das aves, a União Europeia impôs em 1999 algumas medidas mínimas para sistemas de criação. Em 2003, por exemplo, foram proibidas as instalações de gaiolas convencionais com menos de 550cm² por ave, além da necessidade do uso de lixas para as unhas (ROCHA, 2008). A Fig. 3 mostra o confinamento de frangos em um aviário de criação intensiva.

Figura 3. Frangos em gaiolas no sistema avícola intensivo.
Fonte: EMBRAPA (2018)

3.4. Avicultura alternativa: caipira e orgânica

Os sistemas de produção caipira e orgânico são modelos alternativos ao sistema intensivo e são projetados para garantir a qualidade e o bem-estar dos animais. As raças que serão

utilizadas devem ser aquelas que melhor se adaptam ao manejo e clima local, devendo ser garantido sua segurança e que estejam livres de medo, ansiedade, ou qualquer outro tipo de estresse. É preciso que o ambiente proporcione aos animais a liberdade para apresentarem suas características e comportamentos naturais, e que não passem fome, sede e não sofram de nenhum tipo de doença (FIGUEIREDO, 2012; DIAS, 2016). O sistema de criação e as raças são fundamentais para que se garanta uma carne mais saborosa e com menores índices de gordura (DIAS, 2016).

O Fator climático é também um dos fatores cruciais na criação das aves, pois as afeta diretamente. Isso pode ser percebido através de mudanças no comportamento, como respiração mais ofegante, diminuição na taxa de locomoção e o ato de manter as asas mais afastadas do corpo (DIAS, 2016). Apesar de no sistema orgânico os frangos necessitarem de liberdade para o acesso externo, onde devem existir grama e/ou algum outro tipo de vegetação, isso pode, por conta do clima, vir a interferir no bem-estar animal e consequentemente na produção do sistema (BERG, 2002; DIAS, 2016). Dentre as consequências estão o aumento no consumo de água e ração, que podem vir a causar mudanças na taxa de crescimento das aves, afetando o rendimento e a qualidade da carne (DIAS, 2016; CASSUCE, 2013).

A fim de promover o bem-estar animal e viabilizar a inspeção em um sistema produtivo, o Farm Animal Welfare Council criou cinco conceitos para definir um sistema controlado por humanos que fosse capaz de proporcionar um bem-estar aos animais (PEREIRA, 2017):

- **Nutricional:** manter os animais livres de fome e sede;
- **Ambiental:** mantê-los livre de desconfortos;
- **Físico:** mantê-los livres de dores, lesões e doenças;
- **Comportamental:** dar liberdade para expressar seus comportamentos normais;
- **Psicológico:** deixá-los livres de medo e angústia.

4. Objetivos específicos

Com o objetivo de evidenciar a relevância da pesquisa, uma revisão sistemática foi desenvolvida com base nas ferramentas *Google Scholar, IEEE Explorer* e *ScienteDirect*. O resultado da revisão pode ser visto na Tab. 1, que apresenta a união do número de ocorrências de cada conjunto de palavras chaves, em inglês, nas três ferramentas de pesquisa. A última combinação, diferente das outras, não teve nenhum retorno nas ferramentas *IEEE Explorer* e *ScienteDirect*.

Tabela 1. Resultado da revisão sistemática sobre cada palavra chave.

Palavras chave	Resultados	Categoria
automation	3.560.000	1
control	7.270.000	1
fog	1.120.000	1
iot	782.000	1
fog iot	16.700	2
feeding chickens	708.000	2
automation fog	38.500	2

automation iot	83.200	2
control fog	536.000	2
control iot	244.000	2
automation feeding	673.000	2
control feeding	3.930.000	2
iot feeding	15.400	2
fog feeding	80.000	2
automation feeding chickens	44.500	3
automation control feeding chickens	37.600	4
automation control feeding chickens fog iot	6.540	6
automation control feeding chickens alternative aviaries fog iot	130	8

4.1. *Development of automatic chicken feeder using Arduino Uno*

A alimentação de frangos, criados para o abate, pode exigir tempo e muita mão de obra. Para realizar o trato de forma manual, o criador costuma fornecer o alimento semeando-o no chão do aviário. Existem ao menos três problemas em fazê-lo dessa forma: contaminação dos alimentos, decorrentes da presença de insetos e de fezes das aves; desperdício de alimentos, por falha humana, já que não é possível uma correta administração da quantia a ser distribuída; e a necessidade da presença dos responsáveis pelo fornecimento do alimento, o que dificulta que estes indivíduos venham realizar outras tarefas. Com o objetivo de suprir esses problemas, um alimentador de frangos é proposto a fim de automatizar por completo o processo de alimentação, possibilitando ao criador o cumprimento de outras obrigações sem que haja a necessidade de sua presença no local.

Ao projetar o alimentador, o autor considerou a necessidade de armazenar os alimentos, administrar as sobras, e registrar os horários de fornecimento e a quantidade a ser fornecida. O projeto do tratador é composto por alguns sensores e atuadores que são controlados por uma placa Arduíno Uno. O processo de liberação do alimento acontece com a abertura da passagem entre local de armazenamento de comida e o recipiente, controlada por um atuador servo motor. Para evitar o desperdício da comida deixada pelas aves, as sobras ficam retidas dentro do recipiente. Um sensor de temperatura e uma lâmpada são utilizados para que se garanta que os grãos permaneçam frescos.

4.2. A Smart Sensor Network for an Automated Urban Greenhouse

Considerando a dificuldade de obter alimentos frescos a preços acessíveis, na região da Pennsylvania (EUA), em MEAH (2019) é apresentado um projeto de redes de sensores inteligentes para automatização de uma estufa urbana. O projeto foi desenvolvido por alunos de um colégio de engenharia, o York College of Pennsylvania, e foi utilizado para apoiar a inicialização de alunos de escola primária no mundo científico. Dentre os objetivos do trabalho estão:

- Maximização da economia de energia;
- Gerenciamento de aquecedores, resfriadores e da ventilação;
- Uso de histórico de dados meteorológicos para estimativa da energia necessária dentro da estufa.

Para o desenvolvimento do projeto foi feita a divisão do sistema em três partes: sensoriamento; energia e automação; interface. No primeiro subsistema a equipe fez o uso de sensores para coletar dados de luz, umidade e temperatura dentro da estufa. Os dados são encaminhados posteriormente ao subsistema de energia e automação, que irá processá-los e intervir no ambiente a partir de equipamentos como aquecedores e lâmpadas. O protótipo desenvolvido possui 6 (seis) zonas de umidade, sendo que cada uma possui 2 (dois) sensores de umidade. Em paralelo, 5 (cinco) sensores de temperatura, luz e umidade foram distribuídos em plantas. Todo o processo de interação com o usuário foi desenvolvido no subsistema interface, onde ficam hospedados uma aplicação web e o banco de dados.

4.3. A remote pet feeder control system via MQTT protocol

Popularmente aceito dentro da sociedade moderna, a criação de animais vem se tornando um hábito cada vez mais adotado entre as pessoas. Contudo, responsabilidades devem ser assumidas pelos proprietários, sendo uma delas a garantia da alimentação. Em WU (2018) é desenvolvido um alimentador de animais que, ao contrário da maioria, deixa de ser uma simples máquina estacionária para se tornar um alimentador móvel e inteligente. O equipamento *IoT* foi projetado para se assemelhar a um carrinho de brinquedo, possuindo duas rodas traseiras, cada uma conectada a um motor. Ainda, uma câmera *IP* foi instalada na parte frontal do carrinho para possibilitar o monitoramento pela Internet.

4.4. An RFID-Based Smart Nest Box: An Experimental Study of Laying Performance and Behavior of Individual Hens

O nível de produção de ovos pode ser afetado pelo ambiente de postura (ninho), idade das aves ou até mesmo pela raça do animal. Com a motivação de monitorar a produção de ovos, em CHIEN (2018) é apresentada uma solução que tem como objetivo a identificação de galinhas poedeiras que apresentam um rendimento abaixo do esperado. A ferramenta desenvolvida utiliza um leitor *RFID*, conectado a uma placa Arduíno MEGA 2560, cuja serventia é identificar a presença de aves dentro de um ninho. Também é feito o uso de um sensor de pressão, para que seja detectado a existência de algum ovo.

4.5. Autonomic IoT Battery Management with Fog Computing

O uso de energia é indispensável para dispositivos *IoT*, sendo em grande parte dependentes do uso de baterias para que funcionem. Para os usuários é crucial que existam técnicas para economizar energia, principalmente quando a exigência é de anos de durabilidade. Em SAMPAIO (2019) é desenvolvido um sistema *Fog* para alarmes de incêndio que depende do uso de baterias. O trabalho é dividido em duas partes, sendo a primeira referente ao hardware do dispositivo e na segunda são realizadas estimativas do consumo de energia.

O dispositivo *IoT* foi projetado e desenvolvido considerando o contexto de cozinhas em Smart Home's. Para a implementação do protótipo os autores fizeram o uso de um microcontrolador Arduíno Uno ATMega328p, um sensor de temperatura e umidade do ar DHT11, um sensor de gás e fumaça MQ-2 e um sensor de chamas. O dispositivo *IoT* captura os dados através de portas analógicas do Arduíno, conectadas aos sensores, e os envia para *Fog* com uma antena Xbee Zigbee. O sistema *Fog*, por sua vez, processa os dados para gerar e apresentar a estimativa de consumo.

5. Proposta de alimentador inteligente

Considerando o crescente interesse no consumo de alimentos orgânicos, em conjunto com a importância do mercado brasileiro de frangos, um alimentador inteligente é proposto para gerenciar a alimentação das aves de forma autonômica. O alimentador conta com um sistema

Fog, dividido em três camadas, que considera desde a interação com o usuário até a criação e controle de um dispositivo *IoT*. Na subseção 5.1 é apresentada a hierarquia do sistema, enquanto na subseção 5.2 são descritos o dispositivo *IoT* e a caixa de alimentação e na subseção 5.3 é descrito o sistema *Fog*.

5.1. Arquitetura do sistema

A arquitetura do sistema é dividida em três camadas, conforme apresentado na Fig. 4. Uma camada para entrada de dados é responsável pela comunicação com os usuários, denominada como camada de interface. Abaixo dela encontra-se a camada de gerenciamento de dados, a qual se responsabiliza pelo armazenamento dos dados e a comunicação com os nós *IoT*. A última camada é a *IoT*, localizada abaixo da camada de gerenciamento de dados. Uma melhor descrição das camadas é apresentada na sequência do texto.

- **Camada de interface:** é responsável pela obtenção de dados, fornecidos pelos usuários, e pelo envio desses dados para camada de gerenciamento de dados. Na camada de interface é possível que sejam feitos o uso de uma ou mais aplicações, desenvolvidas para diferentes sistemas, tais como Android, iOS, Desktop e sistemas *Web*;
- **Camada de gerenciamento de dados:** se responsabiliza pelo armazenamento dos dados obtidos na camada de interface e pelo envio de comandos de controle para os dispositivos da camada *IoT*. Para comunicação com a interface, a camada conta com um servidor que receberá as informações e armazenará em um banco de dados. Já para comunicação com os dispositivos *IoT*, a camada possui uma aplicação que identificará e enviará aos dispositivos o comando para a realização de uma tarefa na data e horário especificados pelos usuários (armazenados no banco de dados);
- **Camada *IoT*:** representa a união de todos os dispositivos *IoT* do sistema. Os dispositivos recebem os comandos de controle, enviados pela camada de gerenciamento de dados, e com base nos comandos é identificada e cumprida a tarefa em questão.

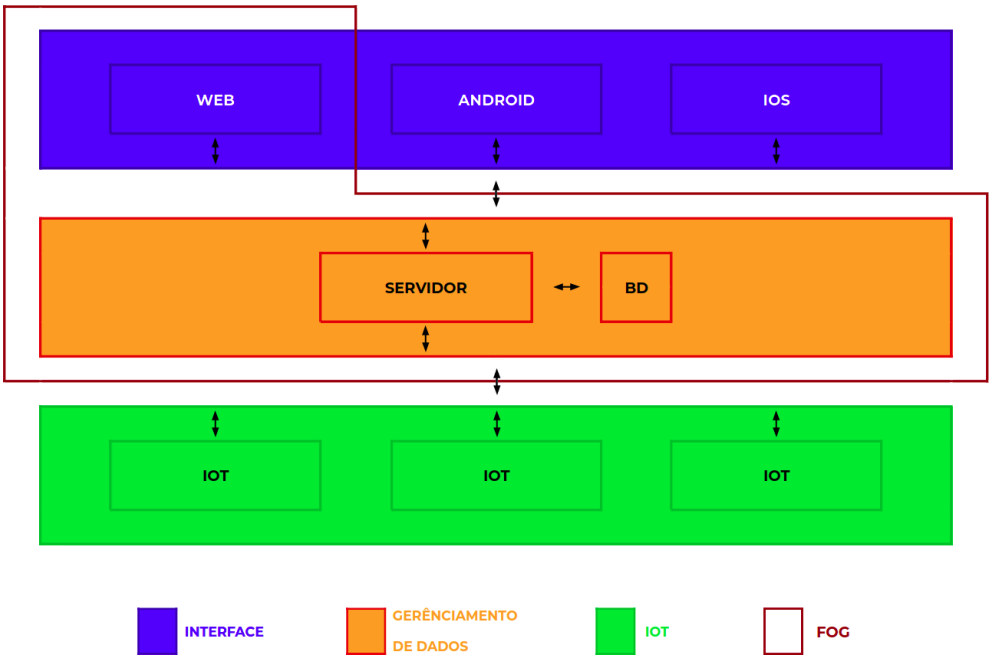


Figura 4. Arquitetura do sistema *Fog*, composto por três camadas: camada de interface; camada de gerenciamento de dados; e camada *IoT*.

5.2. Nó *IoT* e a caixa de alimentação

O nó *IoT* é um equipamento baseado em um carrinho de brinquedo, cuja finalidade é armazenar o alimento das aves e liberá-lo apenas em datas e horários específicos. Para a realização da tarefa, o dispositivo *IoT* conta com o auxílio de uma base (caixa de alimentação) que receberá e disponibilizará o alimento para as aves. O dispositivo é posicionado em cima de dois trilhos, fixados na parte superior da caixa, e permanece imóvel até o recebimento de um comando enviado pelo servidor. Ao receber o comando, o nó caminha sobre os trilhos até o fim e depois retorna ao ponto de origem, liberando o alimento durante todo o trajeto. Na Fig. 5 uma galinha está de frente para o alimentador, logo após o nó *IoT* liberar o alimento.

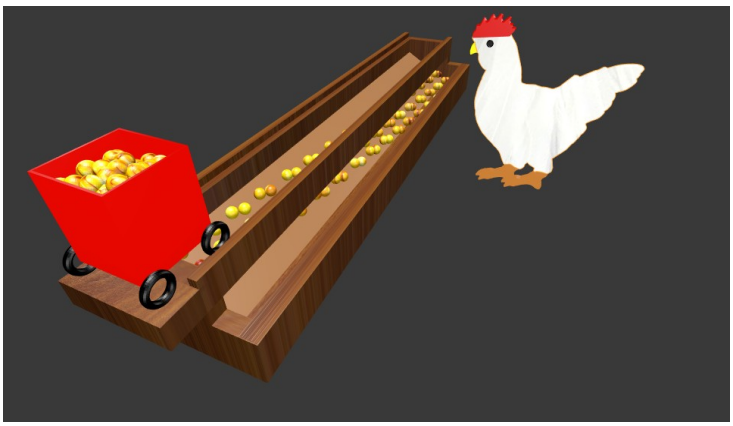


Figura 5. Visão geral do alimentador, em 3D.

A caixa de alimentação conta, ainda, com uma área destinada ao acesso das aves, localizada na parte frontal, e uma rampa de desvio de alimentos, posicionada abaixo dos trilhos utilizados pelo nó *IoT*. O alimento, quando liberado, cai sobre a rampa e sofre um desvio até a parte frontal do alimentador. A Fig. 6 apresenta uma ilustração da rampa e da área de acesso às aves ao alimentador, onde o alimento fica armazenado até ser consumido.

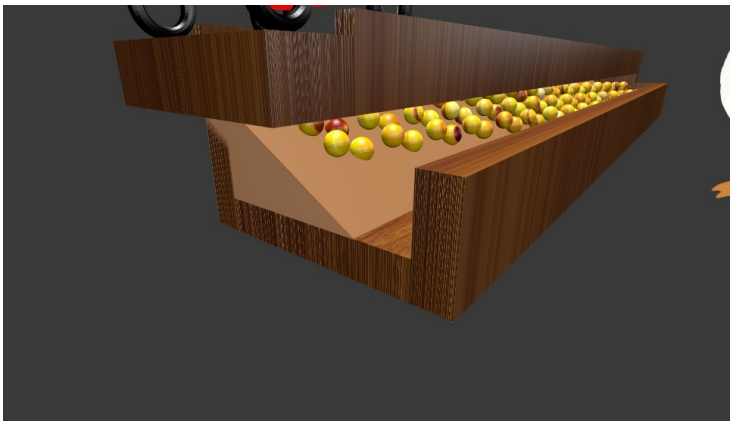


Figura 6. Visão lateral do alimentador, em 3D.

O projeto do carrinho, que consiste de um nó *IoT*, ocorre com a sua divisão em duas partes: superior e inferior. Na parte superior (Fig. 7) é feito o armazenamento do alimento, o qual passa por um processo de afunilamento para que ele deslize até o centro do dispositivo. Após esse processo, o alimento cai dentro de um corredor que o levará até a parte inferior (Fig. 8). Quando isso ocorrer, um controlador de passagem decidirá se o alimento deve ou não ser liberado. Para isso conta-se com uma tampa de isolamento, que fica aberta ou fechada de acordo com a necessidade (liberação ou armazenagem do alimento, respectivamente).

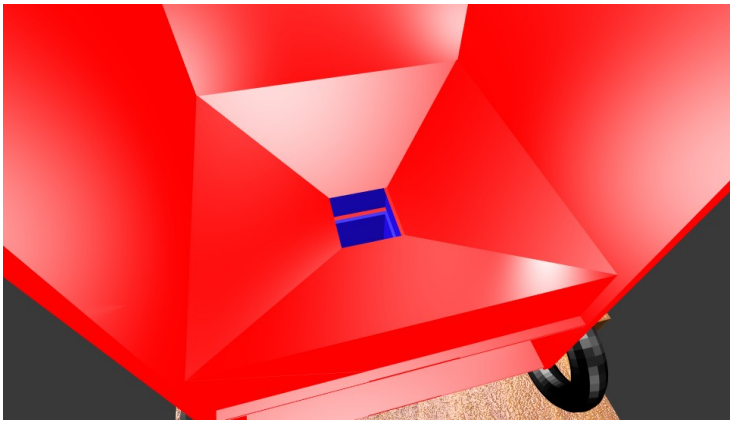


Figura 7. Visão interna da parte de cima do carrinho (nó *IoT*), em 3D.

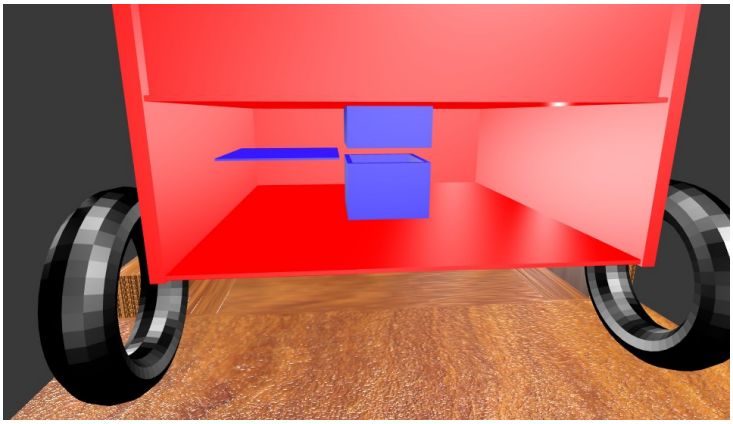


Figura 8. Visão interna da parte de inferior do carrinho (nó *IoT*), em 3D.

5.3. Sistema *Fog*

O sistema *Fog* é responsável por processar e administrar os dados fornecidos pelos usuários, assim como alertar as aves, por meio de efeitos sonoros, no momento em que a caixa de alimentação é abastecida. A partir do processamento dos dados, o sistema *Fog* gera as informações que serão utilizadas pelo nó *IoT*. Desse modo, o sistema *Fog* engloba toda camada de gerenciamento de dados, já que é ele quem repassará os comandos aos nós *IoT* e receberá as informações da camada de interface. Além disso, dependendo do tipo de aplicação da camada de interface, o sistema *Fog* pode ser dividido em duas partes para representar ambas as camadas. Uma aplicação de interface poderia ser um sistema *Web*, que ao invés de ser instalado no lado do cliente, como um aplicativo Android ou iOS, pode ficar no lado do servidor, se tornando, por consequência, um subsistema do sistema *Fog*.

5.4. Aplicação em larga escala

A automatização da alimentação de frangos, em granjas orgânicas, deve considerar o atendimento a um grande número de aves. A proposta do alimentador considera uma implementação em larga escala, em que vários nós *IoT* devem atuar para fornecer o alimento. Na Fig. 9 é apresentada uma ilustração em 3D em que os nós *IoT*, cada um posicionado em cima de uma caixa de alimentação própria, fazem o papel de reposição do alimento. Para uma melhor representação, na ilustração foram desenhados um grande número de frangos a espera do alimento, posicionados em frente às caixas de alimentação.

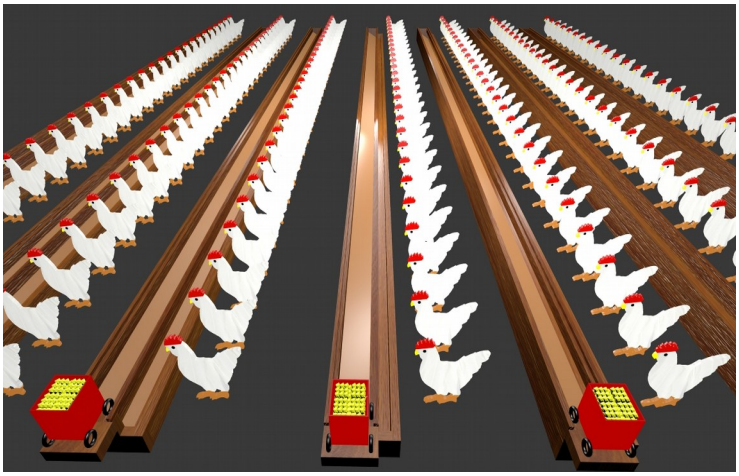


Figura 9. Representação em 3D do alimentador inteligente aplicado em larga escala.

6. Desenvolvimento da proposta

Para o desenvolvimento da proposta foram criados um carrinho eletrônico, um sistema *Web* (em PHP) e um servidor (em Python). O carrinho eletrônico corresponde ao protótipo do nó *IoT*, enquanto que o sistema *Web* foi escolhido para servir como a aplicação da camada de interface. Já a camada de gerenciamento de dados conta com o servidor e com um banco de dados (MySQL), responsáveis por receber e armazenar os dados, inseridos pelos usuários, e por enviar os comandos de controle, ao nó *IoT*.

6.1. Sistema Web

O sistema *Web* foi escolhido como a aplicação da camada de interface. O desenvolvimento levou em consideração todos os cenários de coleta de dados, desde o controle manual do nó *IoT* até o agendamento de horários das tarefas. O sistema conta, ao todo, com 3 (três) telas para interação com o usuário:

- Tela inicial (controle manual);
- Cadastro e atualização de agendamentos;
- Agendamento de horários.

6.1.1. Tela inicial: controle manual

A tela inicial é utilizada para controlar, de forma manual, o nó *IoT* e o som de alerta aos frangos. Para controlar o nó *IoT* é disponibilizado 1 (um) botão, cuja finalidade é enviar um comando ao nó *IoT* para que ele forneça o alimento para as aves. Já para o controle do som, a tela conta com dois botões, para ativar e desativar, e um campo para controlar o volume do som. O botão para ativar o nó *IoT* é mostrado na Fig. 10 (a). Já os botões de controle do som aparecem na Fig. 10 (b).

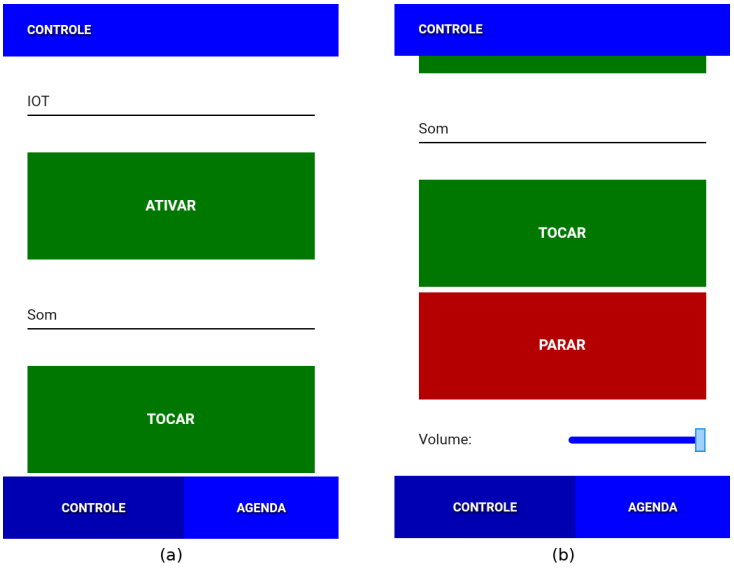


Figura 10. Tela de controle manual do nó *IoT*.

6.1.2. Cadastro e edição de agendamentos

O cadastro de agendamentos considera os dados sobre os dias da semana em que um evento ocorrerá, o horário, se um efeito sonoro será disparado e o tempo e intensidade em que ocorrerá. O preenchimento de um campo para o nome também é exigido, pois facilitará a identificação dos registros quando forem apresentados aos usuários. As Fig. 11 (a) e Fig. 11 (b) mostram a tela de edição, que corresponde também a tela de cadastro.

Figura 11. Tela de cadastro e edição de agendamentos.

6.1.3. Agenda de horários

Após a criação dos agendamentos, uma lista dos registros é disponibilizada na tela agenda de horários. A Fig. 12 mostra como ocorre a listagem, apresentando o nome, horário e os dias programados para ocorrência do evento. A tela conta com campos de seleção para remover um ou mais agendamentos. Em caso de ao menos um agendamento ser selecionado, um botão vermelho é mostrado para solicitar a remoção do registro. Já no caso de nenhum campo ser selecionado, um botão verde é utilizado para abrir a tela de cadastro de um novo

agendamento. Um link fica disponível no nome do agendamento para abrir a página de edição.

Figura 12. Tela de listagem dos agendamentos para ocorrência de eventos do nó *IoT*.

6.2. Protótipo nó *IoT*

A proposta deste trabalho considera a implantação de um sistema para utilização em larga escala. A implementação da proposta, contudo, considera apenas um nó *IoT*, desenvolvido como protótipo. Uma caixa de alimentação, também protótipo, foi montada para atender as necessidades do dispositivo. Na Fig. 13 é apresentada uma visão geral do dispositivo em cima da caixa de alimentação. Já na Fig. 14 é possível a visualização geral do carrinho, sem a caixa de alimentação. Por último, a Fig. 15 mostra a parte externa superior do dispositivo, onde ficam armazenados os alimentos.

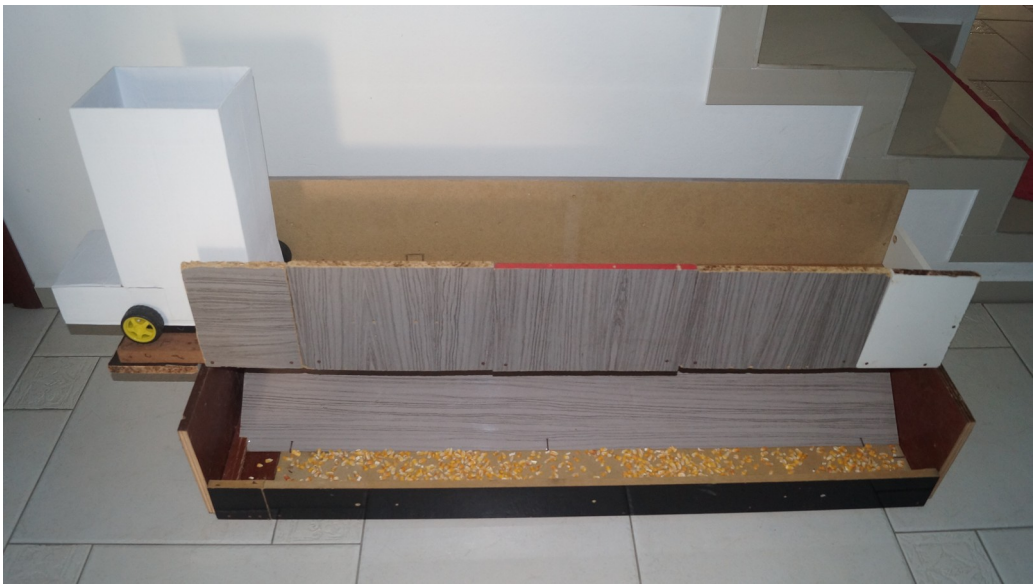


Figura 13. Visão geral do nó *IoT* e da caixa de alimentação.

Figura 14. Visão geral do nó *IoT*.

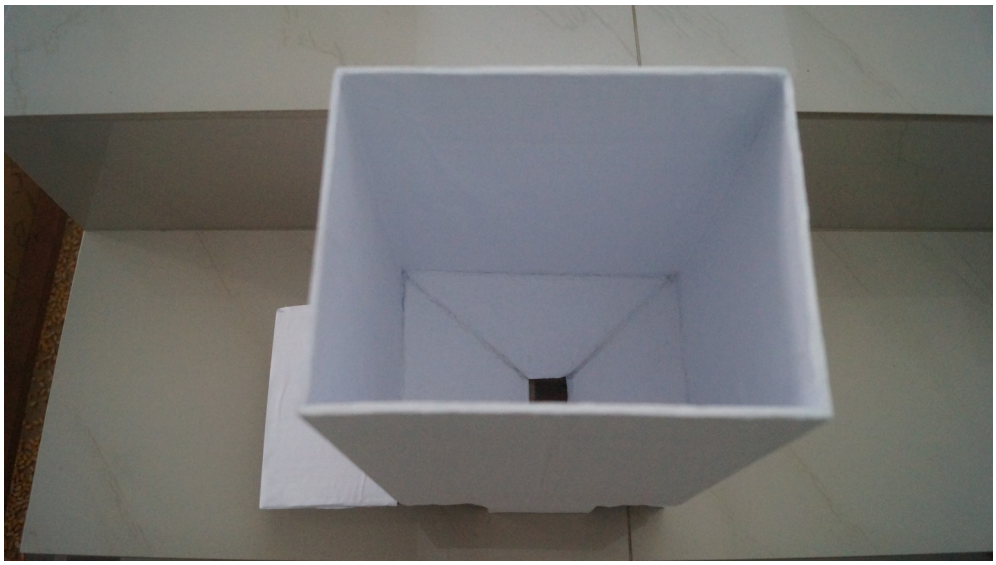


Figura 15. Parte externa superior do nó *IoT*.

6.2.1. Esquema lógico do circuito eletrônico

Para o desenvolvimento do nó *IoT* foram utilizados um Arduíno (UNO), dois motores DC de 210RPM, um driver para motores DC (L298N), uma antena *Wi-Fi* (ESP8266 ESP-01), um driver de motores de passos (ULN2003), um motor de passos (28BYJ-48) e um sensor de distância (HC-SR04). Na parte dianteira do carrinho foram instalados os dois motores DC, cada um conectado a uma roda plástica, e o sensor de distância. As duas rodas traseiras, por não se conectarem aos motores, foram interligadas através de um eixo. O motor de passos foi posicionado no centro do dispositivo, cuja finalidade é o controle da passagem dos alimentos. Na parte externa traseira do dispositivo é fixado o Arduíno e uma fonte de 6V, formada por 5 (cinco) pilhas AA de 1.2V (2600mAh).

Na Fig. 16 é apresentado o esquema lógico do nó *IoT*. O Arduíno é conectado a antena ESP8266 ESP-01, ao driver L298N, ao driver ULN2003 e ao sensor HC-SR04. Uma fonte de 6V alimenta o driver L298N, e uma segunda fonte, também de 6V, alimenta o Arduíno. O Arduíno fica responsável por alimentar o driver ULN2003 e o sensor de distância, enquanto os motores são energizados pelos respectivos drivers.

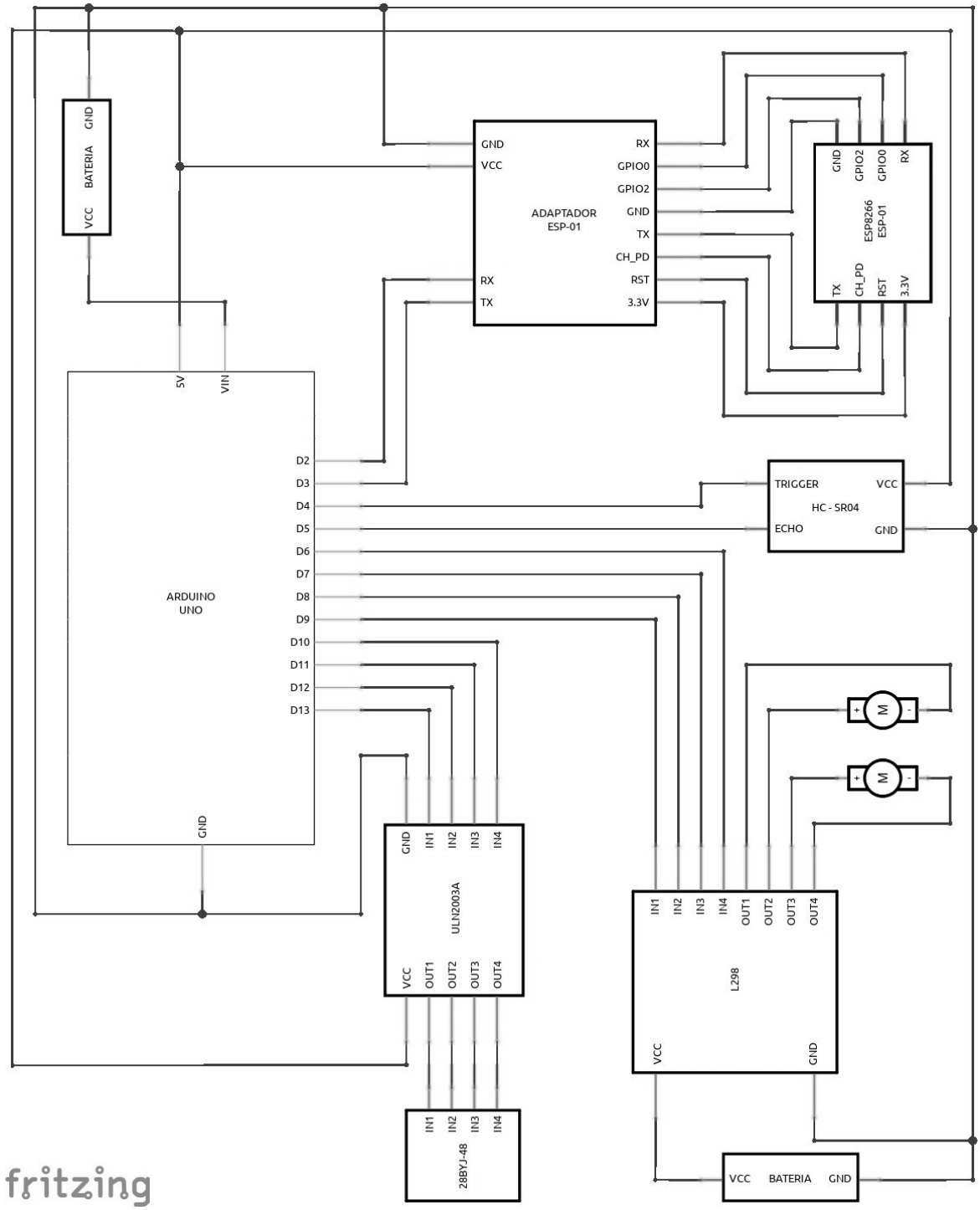


Figura 16. Esquema lógico do nó *IoT*.

6.2.2. Movimentação

O funcionamento do nó *IoT* fica sob responsabilidade do Arduíno. Ao receber o comando de acionamento, através da antena ESP8266 ESP-01, o Arduíno emite um sinal para o driver

L298N para que este realize o acionamento dos motores DC. O acionamento dos motores fará com que o carrinho se movimente, em um sentido, até o sensor HC-SR04 detectar um obstáculo. Após a detecção, o Arduíno altera o sentido de movimentação do carrinho para que ele volte ao ponto de partida.

A movimentação do carrinho conta, ainda, com o uso da função millis(), da biblioteca Arduíno. O objetivo da função millis() é obter o intervalo de tempo entre a inicialização do Arduíno até o instante da chamada da função, cujos valores são fundamentais para o cálculo do tempo de movimentação. Para o cálculo, o algoritmo subtrai o valor obtido no início do trajeto pelo valor obtido no momento da detecção do obstáculo.

O retorno ao ponto de origem ocorre com o uso da função delay(), da biblioteca Arduíno, que recebe como parâmetro o tempo calculado anteriormente. A função delay() realiza a suspensão do Arduíno durante esse tempo, porém os motores DC continuam ligados. Ao despertar, o Arduíno desativa os motores DC para que o carrinho termine o processo de movimentação.

6.2.3. Reservatório

Além do acionamento dos motores DC, o Arduíno controla o driver ULN2003 para que seja aberta ou fechada a passagem do alimento. No momento em que os motores DC são acionados, o Arduíno emite um sinal ao driver ULN2003 para que o motor 28BYJ-48 faça um giro de 180 graus na horizontal, permanecendo (o motor) desativado até que o carrinho retorne ao ponto de origem. Ao retornar, o Arduíno determina que o motor 28BYJ-48 realize um giro de 180 graus no sentido inverso, e em seguida o motor é novamente desativado.

A Fig. 17 apresenta a parte interna do dispositivo, onde é possível visualizar o motor 28BYJ-48 conectado a uma placa de metal, para abrir e fechar o reservatório. Também é possível a visualização do driver L298N e do driver ULN2003, um pouco acima do motor 28BYJ-48. Abaixo do motor 28BYJ-48 estão o Arduíno, as pilhas AA e a antena ESP8266 ESP-01.

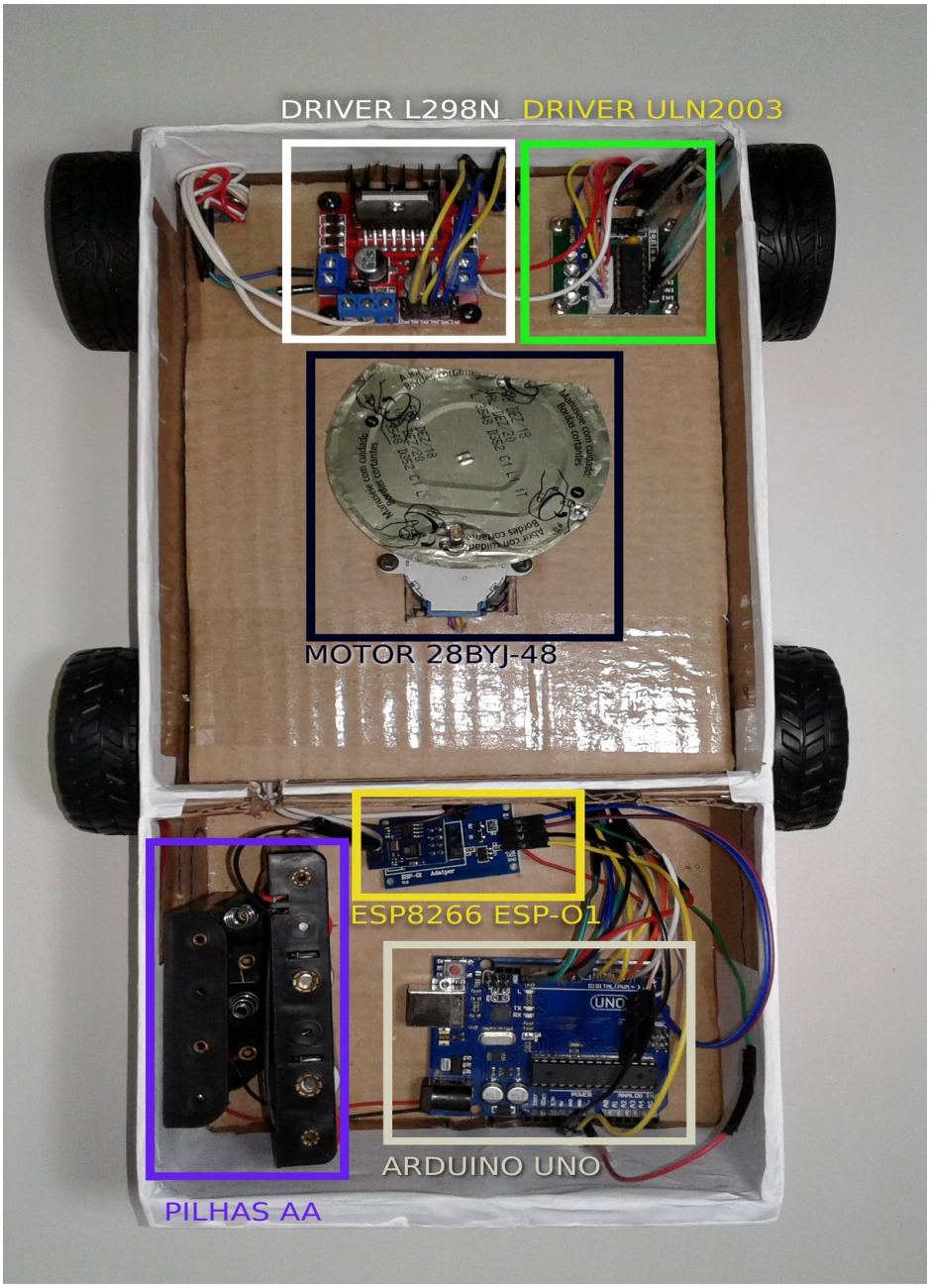


Figura 17. Visão da parte interna do dispositivo *IoT*.

6.3. Servidor

O servidor, responsável pelo gerenciamento dos dados, conta com 3 (três) *Sockets TCP/IP*. O primeiro *Socket* atua para troca de dados com a interface. O segundo se destina a recepção de dados enviados pela camada *IoT*, e é utilizado para cadastrar um nó *IoT* no servidor. O último *Socket* atua como emissor, enviando os comandos de operação para todos os nós *IoT* cadastrados.

A hospedagem do servidor é feita em um Raspberry Pi 3, que conta com um módulo *Wi-Fi* para comunicação *Wireless* e entradas para conexões com dispositivos de hardware externos, como os alto falantes. Para o funcionamento da gerência de dados, o servidor conta com um banco de dados MySQL para o armazenamento de dados, recebidos da camada de interface, e processados pelo servidor. A Fig. 18 apresenta o esquema do banco de dados, que conta com a tabela evento_agendamento. Na tabela são armazenados os registros de

agendamento dos dias e horários em que o nó *IoT* irá operar, além da possível emissão de sons de alerta para as aves.

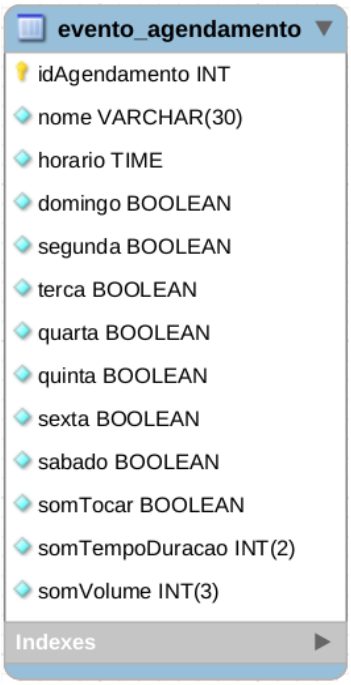


Figura 18. Esquema do banco de dados utilizado para armazenar o agendamento de eventos do nó *IoT*.

6.3.1. Comunicação entre interface e servidor

Para gerenciar as requisições, o servidor conta com a padronização da inserção (Fig. 19), remoção (Fig. 20), atualização (Fig. 21), seleção individual (Fig. 22) e seleção de todos os agendamentos de eventos da tabela evento. Desse modo, independente da aplicação da camada de interface, as solicitações devem seguir um modelo pré-definido para possibilitar a comunicação entre as camadas. A solicitação para seleção de todos os eventos possui apenas 1 (um) byte, correspondente ao caractere "e". Os dados, de ambas as operações, são definidos da seguinte forma:

- **Operação (op):** possui 1 (um) byte (caractere) para informar o tipo de operação a ser realizada. A Tab. 2 apresenta os valores de cada operação;
- **Dias da semana (d, s, t, q, q, s, s):** correspondem, juntos, ao conjunto de 7 bytes para habilitar/desabilitar o funcionamento do carrinho nos 7 dias da semana: domingo, segunda, terça, quarta, quinta, sexta e sábado, nessa ordem. Cada um dos 7 bytes conterá o valor 0, para desabilitar, ou 1, para habilitar;
- **Horário (hora, minuto):** retém um total de 4 bytes, que representam, juntos, a hora e minuto em que o agendamento ocorrerá. Os 2 primeiros bytes se referem a hora, no intervalo entre 0 e 23, e os dois últimos aos minutos, dentro do intervalo de 0 a 59;
- **Som (habilitado, volume, tempo de duração):** se referem a 1 byte para habilitação/desabilitação do alerta sonoro, 3 bytes para o intervalo entre 0 a 100 da intensidade do volume e de 2 bytes para o intervalo de tempo entre 0 a 59 segundos;
- **ID (id):** formado por 10 bytes para representar o código único de registro do agendamento (id), no banco de dados;
- **Nome (tamanho, nome):** possui um tamanho variável de bytes, que vai de 2 a 32 bytes. Os dois primeiros bytes correspondem ao tamanho do nome (ilustrado na

sequência com a letra N), que pode variar de 0 a 30 caracteres. Os bytes seguintes são preenchidos com os caracteres do nome.

Figura 19. Mensagem de solicitação para registrar um agendamento.

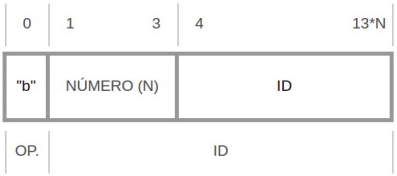


Figura 20. Mensagem de solicitação para remoção de um agendamento.

Figura 21. Mensagem de solicitação para edição dos dados de um agendamento.

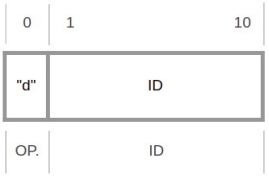


Figura 22. Mensagem de solicitação dos dados de um agendamento.

Tabela 2. Bytes de identificação enviados na mensagem da interface para o servidor.

Operação	Byte de comando
Inserir	a
Remover	b
Editar	c
Selecionar agendamento	d
Selecionar todos os agendamentos	e

A troca de dados conta, ainda, com os padrões para o envio da resposta do servidor para a interface. Os modelos de dados são os mesmos utilizados para representar as solicitações da interface para o servidor. As respostas para solicitação de inserção, remoção e atualização correspondem a 1 (um) byte contendo o valor 0 (zero), para erro/falha, ou o valor 1 (um), para sucesso. Já as respostas para seleção de um agendamento ou de todos os agendamentos são representadas pelas Fig. 23 e Fig. 24, respectivamente.

Figura 23. Mensagem de resposta do servidor para a solicitação dos dados de um agendamento.

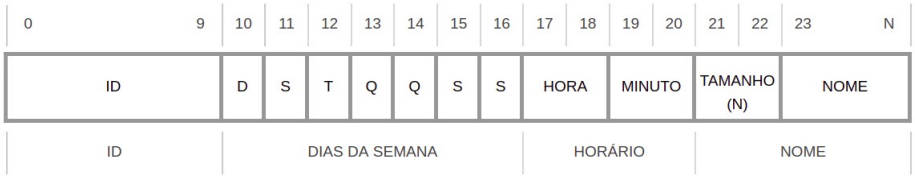


Figura 24. Mensagem de resposta do servidor para a solicitação dos dados de todos os agendamentos.

6.3.2. Comunicação entre servidor e *IoT*

A comunicação entre o servidor e o nó *IoT* ocorre em duas etapas, sendo a primeira de configuração e a segunda de controle. A primeira etapa consiste no recebimento de uma solicitação, do nó *IoT* ao servidor, para que o servidor comece a enviar comandos de controle ao dispositivo. Ao receber a solicitação o servidor armazena o endereço de rede do dispositivo em uma lista, e assim que um comando tenha de ser enviado, o servidor o envia a todos os nós cadastrados na lista.

O comando esperado por um nó *IoT* corresponde a um byte de controle: caractere "a". O recebimento do comando força o dispositivo a percorrer a caixa de alimentação, e durante o percurso deve-se liberar o alimento armazenado. Qualquer dado diferente do caractere "a" será ignorado. Se o nó já estiver realizando uma operação, o caractere "a" também será ignorado até que a tarefa seja finalizada.

7. Conclusão e trabalhos futuros

Nessa seção são apresentadas as conclusões e os trabalhos futuros. Na subseção 7.1 são descritos os resultados obtidos com o desenvolvimento deste trabalho. Na seção 7.2 são apresentados possíveis problemas que poderiam ser solucionados futuramente, com base nos resultados apresentados nesse trabalho.

7.1. Conclusão

O objetivo deste trabalho foi contribuir com a avicultura orgânica, e para isso foram propostas ferramentas que possibilitam a automação parcial da criação de frangos. O trabalho levou em consideração o tempo exigido para alimentar as aves de forma manual, em conjunto com o crescente interesse em produtos orgânicos. Um levantamento foi realizado para encontrar ferramentas que pudessem auxiliar na criação de um alimentador de frangos, resultando em 5 (cinco) trabalhos correlatos, mencionados na Seção 4. Considerando o levantamento realizado, a proposta e implementação de um sistema inteligente, que gerencia o fornecimento dos alimentos, foram apresentados nas Seção 5 e Seção 6. Considerando, ainda, o grande número de aves que devem ser atendidas, o trabalho tratou da implementação para um contexto em larga escala.

O desenvolvimento da proposta, em grande parte, acontece com o projeto de um dispositivo *IoT*. O dispositivo é responsável por cumprir com o papel de repositor do alimento dentro de uma caixa de alimentação (utilizada para alimentação das aves). Um servidor, em conjunto com um sistema *Web*, foram instalados na *Fog* e utilizados em conjunto para

gerenciar o dispositivo *IoT*. O usuário cadastra as informações necessárias, através do sistema web, como os dias e horários de atuação do nó *IoT*, e o servidor as processa e se encarrega de controlar o dispositivo *IoT*.

Alguns problemas surgiram durante a etapa de implementação do protótipo, do nó *IoT*, e gerou a necessidade da realização da troca de equipamentos de hardware. Inicialmente foram utilizadas antenas NRF24L01 para comunicação entre a *Fog* e os dispositivos *IoT*, porém uma sequencia de erros levaram a substituição desta antena pela antena *Wi-Fi* ESP8266 ESP-01. Já no processo de finalização do protótipo, um problema de alimentação forçou a separação da fonte de energia entre o Arduíno UNO e os motores DC, pois ao realizar a inversão do sentido de movimentação dos motores o Arduíno sofria um processo de reinicialização.

7.2. Trabalhos futuros

A proposta para automação da alimentação de frangos, apresentadas neste trabalho, consideram o controle dos alimentos enquanto ainda estão estocados ou sendo liberados. Contudo, os cuidados necessários com as sobras não foram devidamente realizados. Desse modo, um trabalho futuro deveria considerar o gerenciamento inteligente dessas sobras, para que possíveis parasitas, como os ratos, não tenham acesso aos alimentos.

Além do aprimoramento da ferramenta, um estudo sobre o uso em outros animais poderia ser realizado. O trabalho é motivado pela criação de frangos, porém animais domésticos, como cães e gatos, talvez possam ser alimentados pelo equipamento.

Por fim, apesar do protótipo apresentado depender do uso de pilhas AA, a economia sobre o consumo de energia não foi considerada na implementação. Um estudo sobre os gastos realizados, além de medidas para diminuição desses gastos, poderiam ser exploradas para melhorar a eficiência do dispositivo.

Referências

ALVES, S. P.; SILVA, I. J. O.; PIEDADE, S. M. S. Avaliação do bem estar de aves poedeiras comerciais: efeitos do sistema de criação e do ambiente bioclimático sobre o desempenho das aves e a qualidade de ovos. Revista Brasileira de Zootecnia, v. 36, n. 5, p. 1388-1394, 2007.

AMMAD-UDDIN, M.; AYAZ, M.; AGGOUNE, E-H.; SAJJAD, M. Wireless sensor network: A complete solution for poultry farming. 2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT), p. 321-325, 2014.

ANOMALY, J. What’s Wrong With Factory Farming? Public Health Ethics, v. 8, n. 3, p. 246-254, 2015.

APAOLAZA, V.; HARTMANN, P.; SOUZA, C.; LÓPEZ, C. M. Eat organic – Feel good? The relationship between organic food consumption, health concern and subjective wellbeing. Food Quality and Preference, v. 63, p. 51-62, 2018.

Avicultura sofrerá com mudanças climáticas - Portal Embrapa. Disponível em: <www.embrapa.br/busca-de-noticias/-/noticia/7546116/avicultura-sofrera-com-mudancas-climaticas>. Acesso em ago. 2018.

BARACHO, M. S.; NAAS, I. A.; BETIN, P. S.; MOURA, D. J. Factors that Influence the Production, Environment, and Welfare of Broiler Chicken: A Systematic Review. Brazilian Journal of Poultry Science, v. 20, n. 3, p. 617-624, 2018.

BERG, C. Health and Welfare in Organic Poultry Production. Acta Veterinaria Scandinavica, v. 43, n. 1, p. 37-45, 2002.

CASSUCE, D. C.; TINÔCO, I. F. F.; BAÊTA, F. C.; ZOLNIER, S.; CECON, P. R.; VIEIRA, M. F. A. Thermal comfort temperature update for broiler chickens up to 21 days of age. , v. 33, n. 1, p. 28-36, 2013.

CHEN, Y.; AZHARI, M. Z.; LEU, J. Design and implementation of a power consumption management system for smart home over fog-cloud computing View Document. 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG) , 2018.

CHIEN, Y-R.; CHEN, Y-X. An RFID-Based Smart Nest Box: An Experimental Study of Laying Performance and Behavior of Individual Hens. SENSORS 2018, v. 18, n. 3, p. 859-870 , 2018.

CHIEOCHAN, O.; SAOKAEW, A.; BOONCHIENG, E. IOT for smart farm: A case study of the Lingzhi mushroom farm at Maejo University. 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2017.

DAMASCENO, F. A.; CASSUCE, D. C.; ABREU, L. H. P.; SCHIASSI, L.; TINÔCO, I. F. F. Effect of thermal environment on performance of broiler chickens using fuzzy modeling. Revista Ceres, v. 64, n. 4, p. 337-343, 2017.

DIAS, A. N.; MACIEL, M. P.; AIURA, A. L. O.; AROUCA, C. L. C.; SILVA, D. B.; MOURA, V. H. S. Linhagens de frangos caipiras criadas em sistema semi-intensivo em região de clima quente. Pesquisa Agropecuária Brasileira, v. 51, n. 12, p. 2010-2017, 2016.

FELTES, M. M. C.; ARISSETO-BRAGOTTO, A. P.; BLOCK, J. M. Food quality, food-borne diseases, and food safety in the Brazilian food industry. Food Quality and Safety, v. 1, n. 1, p. 13-27, 2017.

FIGUEIREDO, E. A. P.; SOARES, J. P. G. Sistemas orgânicos de produção animal: dimensões técnicas e econômicas. Anais da 49a Reunião Anual da Sociedade Brasileira de Zootecnia, 2012.

GIFFORD, K.; BERNARD, J. C. The effect of information on consumers' willingness to pay for natural and organic chicken. International Journal of Consumer Studies, v. 35, n. 3, p. 282-289, 2011.

GUPTA, H.; DASTJERDI, A. V.; GHOSH, S. K.; BUYYA, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Journal of Software: Practice and Experience, v. 47, n. 9, p. 1275-1296, 2017.

HONKANEN, P.; VERPLANKEN, B.; OLSEN, S. O. Ethical values and motives driving organic food choice. Journal of Consumer Behaviour, v. 5, n. 5, p. 420-430, 2006.

KHALIL, I. M.; KHREISHAH, A.; AZEEM, M. Cloud Computing Security: A Survey. Computers, v. 3, n. 1, p. 1-35, 2014.

LOUZADA, M. L. C.; RICARDO, C. Z.; STEELE, E. M.; LEVY, R. B.; CANNON, G.; MONTEIRO, C. A. The share of ultra-processed foods determines the overall nutritional quality of diets in Brazil. Public Health Nutrition, v. 21, n. 1, p. 94-102, 2017.

MEAH, K.; FORSYSTH, J.; MOSCOLA, J. A Smart Sensor Network for an Automated Urban Greenhouse. 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), p. 23-27, 2019.

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology, 2011.

MOURA, G. R. S.; MENDONÇA, M. O.; SALGADO, H. R.; CASTRO, J. O.; SOUZA, R. T. Galinhas semipesadas em postura criadas sobre diferentes tipos de cama. Revista Brasileira de Saúde e Produção Animal, v. 18, n. 2, p. 378-387, 2017.

MUKHERJEE, M.; SHU, L.; WANG, D. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. IEEE Communications Surveys & Tutorials, v. 20, n. 3, p. 1826-1857, 2018.

MUNIR, A.; KANSAKAR, P.; KHAN, S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. IEEE Consumer Electronics Magazine, v. 6, n. 3, p. 74-82, 2017.

NAVANEETH, B. S.; MURTY, A. K. AUTOMATIC POULTRY FEEDER. International Journal of Advance Engineering and Research Development, v. 2, n. 7, p. 338-343, 2015.

PERALA, S. S. N.; GALANIS, I.; ANAGNOSTOPOULOS, I. Fog Computing and Efficient Resource Management in the era of Internet-of-Video Things (IoVT). 2018 IEEE International Symposium on Circuits and Systems (ISCAS) , 2018.

PEREIRA, D. C. O.; MIRANDA, K. O. S.; FILHO, L. C. D.; PEREIRA, G. V.; PIEDADE, S. M. S.; BERNO, P. R. Presence of roosters in an alternative egg production system aiming at animal welfare. Brazilian Journal of Animal Science, v. 46, n. 3, p. 175-184, 2017.

PSD Online. Disponível em: <apps.fas.usda.gov/psdonline/app/index.html#/app/downloads>. Acesso em ago. 2018.

RAY, P. P. A survey on Internet of Things architectures. Journal of King Saud University - Computer and Information Sciences, v. 30, n. 3, p. 291-319, 2018.

ROCHA, J. S. R.; LARA, L. J. C.; BAIÃO N. C. PRODUÇÃO E BEM-ESTAR ANIMAL - ASPECTOS ÉTICOS E TÉCNICOS DA PRODUÇÃO INTENSIVA DE AVES. Ciência Veterinária nos Trópicos, v. 11, n. 1, p. 49-55, 2008.

ROMAN, R.; LOPEZ, J.; MASAHIRO, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, v. 78, n. 2, p. 680-698, 2018.

ROSSA, L. S.; STERTZ, S. C.; MACEDO, R. E. F. Regulamentação, mercado e qualidade da carne de frango orgânico no Brasil – Revisão. Revista Acadêmica: Ciência Animal, v. 10, n. 1, p. 29-44, 2012.

SAMPAIO, H. V.; JESUS, A. L. C.; BOING, R. N.; WESTPHALL, C. B. Autonomic IoT Battery Management with Fog Computing. GPC: International Conference on Green, Pervasive, and Cloud Computing, v. 11484, p. 89-103, 2019.

SILVA, M. A. N.; FILHO, P. H.; ROSÁRIO, M. F.; COELHO, A. A. D.; SAVINO, V. J. M.; GARCIA, A. A. F.; SILVA, I. J. O.; MENTEN, J. F. M. Influência do Sistema de Criação sobre o Desempenho, a Condição Fisiológica e o Comportamento de Linhagens de Frangos para Corte. Revista Brasileira de Zootecnia, v. 32, n. 1, p. 208-213, 2003.

SINGH, S.; JEONG, Y-S.; PARK, J. H. A survey on cloud computing security: Issues, threats, and solutions. Journal of Network and Computer Applications, v. 75, p. 200-222, 2016.

SOH, Z. H. C.; ISMAIL, M. H.; OTTHAMAN, F. H.; SAFIE, M. K.; ZUKRI, M. A. A.; ABDULLAH, S. A. C. Development of automatic chicken feeder using Arduino Uno. 2017

International Conference on Electrical, Electronics and System Engineering (ICEESE), p. 120-124, 2017.

STERGIOU, C.; PSANNIS, K. E.; KIM, B-G.; GUPTA, B. Secure integration of IoT and Cloud Computing. Future Generation Computer Systems, v. 78, n. 3, p 964-975, 2018.

VOGADO, G. M. S.; VOGADO, K. T. S.; FONSECA, W. J. L.; FONSECA, W. L.; VOGADO, W. F.; OLIVEIRA, A. M.; OLIVEIRA, N. M.; LUZ, C. S. M. EVOLUÇÃO DA AVICULTURA BRASILEIRA. Nucleus Animalium, v. 8, n. 1, p. 49-58, 2016.

XU, L. D.; HE, W.; LI, S. Internet of Things in Industries: A Survey. IEEE Transactions on Industrial Informatics, v. 10, n. 4, p. 2233-2243, 2014.

WU, W-C.; CHENG, K-C.; LIN, P-Y. A remote pet feeder control system via MQTT protocol. 2018 IEEE International Conference on Applied System Invention (ICASI), p. 487-489, 2018.

ZANDER, K.; HAMM, U. Consumer preferences for additional ethical attributes of organic food. Food Quality and Preference, v. 21, n. 5, p. 495-503, 2010.

ZANELLA, A.; BUI, N.; CASTELLANI, A.; VANGELISTA, L.; ZORZI, M. Internet of Things for Smart Cities. IEEE Internet of Things Journal, v. 1, n. 1, p. 22-32, 2014.

ZANINELLI, M.; REDAELLI, V.; TIRLONI, E.; CRISTIAN, B.; DELL'ORTO, V.; SAVOINI, G. First Results of a Detection Sensor for the Monitoring of Laying Hens Reared in a Commercial Organic Egg Production Farm Based on the Use of Infrared Technology. SENSORS, v. 16, n. 10, 2016.

APÊNDICE B – Código do Servidor *Fog*

Para o desenvolvimento do sistema *Fog* foi utilizada a versão 3 da linguagem Python.

main.py

```
001 from threading import Thread, Lock
002
003 from servidor import ativar_servidor_interface, ativar_servidor_iot,
    no_iot_enviar_comando
004 from classes.agendamento import *
005
006 import time
007
008 def cumprir_agendamento(mutex, eventoAtual):
009         while True:
010                 mutex.acquire()
011
012                 if eventoAtual.disparar_agora():
013                         no_iot_enviar_comando()
014                 mutex.release()
015
016                 # Dorme ate alterar o minuto do relogio
017                 # Ex:
018                 # agora 19:30:40 e precisa acordar 19:31:00, entao: dorme 20s
019                 proximoMinuto = 60-int(datetime.now().strftime("%S"))
020                 time.sleep(proximoMinuto)
021
022 if __name__ == "__main__":
023         eventoAtual = Evento()
024         eventoAtual.buscar_proximo_evento()
025
026         mutex = Lock()
027
028         listaDeNosIot = []
029         threadServidorIot = Thread(target=ativar_servidor_iot, args=(mutex,))
030         threadServidorInterface = Thread(target=ativar_servidor_interface,
    args=(mutex,eventoAtual))
031         threadAgendamento = Thread(target=cumprir_agendamento,
    args=(mutex,eventoAtual))
032
033         threadServidorInterface.start()
034         threadServidorIot.start()
035         threadAgendamento.start()
036
037         threadServidorIot.join()
038         threadServidorInterface.join()
039         threadAgendamento.join()
```

servidor.py

```
001 import socket
002 import pygame
003
```

```
004 from threading import Thread, Lock
005 from bd.agendamento import *
006
007 #SERVER_HOST = "192.168.25.36"
008 SERVER_HOST = "ricardoboing"
009 SERVER_INTERFACE_PORT = 8086
010 SERVER_IOT_PORT = 8081
011 CLIENTE_IOT_PORT = 8080
012 IOT_ADDRESS_LIST = []
013
014 MUSIC_ADDRESS = "musica.mp3"
015
016 ############################## SERVER IOT ##############################
017 # Recebe solicitacoes de dispositivos iot para serem cadastrados (IP)
018 # em uma lista. Posteriormente, comandos de controle sao enviados aos
019 # IP's cadastrados nessa lista.
020 #
021 # ativar_servidor_interface, no_iot_enviar_comando
022 ########################################################################
023 def ativar_servidor_iot(mutex):
024     global IOT_ADDRESS_LIST
025
026     serverSocket = socket.socket()
027     serverSocket.bind((SERVER_HOST, SERVER_IOT_PORT))
028     serverSocket.listen()
029
030     print("SERVER CONNECT - HOST: %s | PORT: %s (IOT)" %(SERVER_HOST,
    SERVER_IOT_PORT))
031
032     try:
033         while True:
034             conexao, ip = serverSocket.accept()
035             print("\nIOT CONNECT -------\n")
036             print("IP: %s" %(ip[0]))
037
038             # Um iot nao pode ser cadastrado em paralelo com
039             # o gerenciamento de dados da camada de interface
040             mutex.acquire()
041
042             # Cadastra o ip do iot, caso ainda nao esteja na lista
043             if ip[0] not in IOT_ADDRESS_LIST:
044                 IOT_ADDRESS_LIST.append(ip[0])
045                 print("IOT CADASTRADO")
046
047             mutex.release()
048
049             print("\nIOT DISCONNECT -------\n")
050     except:
051         pass
052     finally:
053         conexao.close()
054     serverSocket.close()
055
056 ########################## SERVER INTERFACE ##########################
057 # Recebe dados da camada de interface e realiza o gerenciamento
```

```
058 # desses dados.
059 #
060 # ativar_servidor_iot, no_iot_enviar_comando, som_configurar
061 ##########################################################################
062 def ativar_servidor_interface(mutex, eventoAtual):
063     serverSocket = socket.socket()
064     serverSocket.bind((SERVER_HOST, SERVER_INTERFACE_PORT))
065     serverSocket.listen()
066 
067     print("SERVER CONNECT - HOST: %s | PORT: %s (INTERFACE)" %
    (SERVER_HOST, SERVER_INTERFACE_PORT))
068 
069     # Inicializa a biblioteca para ativar/desativar efeitos sonoros
070     # utilizados para alertar as aves.
071     pygame.init()
072     pygame.mixer.music.load(MUSIC_ADDRESS)
073 
074     try:
075         while True:
076             conexao = serverSocket.accept()[0]
077             print("CLIENT CONNECT -------\n")
078 
079             operacao = str(ler_conteudo_conexao(conexao,1))
080 
081             # As operacoes nao podem ser realizadas em paralelo com
082             # o cadastrado dos IP's dos dispositivos.
083             mutex.acquire()
084 
085             # INSERIR EVENTO
086             if operacao == 'a':
087                 print("_INSERIR EVENTO")
088                 valorDeRetorno = bd_agendamento_insert(conexao)
089                 eventoAtual.buscar_proximo_evento()
090 
091             # REMOVER EVENTO
092             elif operacao == 'b':
093                 print("_REMOVER EVENTO")
094                 valorDeRetorno = bd_agendamento_remove(conexao)
095                 eventoAtual.buscar_proximo_evento()
096 
097             # UPDATE EVENTO
098             elif operacao == 'c':
099                 print("_UPDATE EVENTO")
100                 valorDeRetorno = bd_agendamento_update(conexao)
101                 eventoAtual.buscar_proximo_evento()
102 
103             # SELECT EVENTO
104             elif operacao == 'd':
105                 print("_SELECT EVENTO")
106                 valorDeRetorno = bd_agendamento_select(conexao)
107 
108             # SELECT_ALL EVENTO NO BANCO DE DADOS
109             elif operacao == 'e':
110                 print("_SELECT_ALL EVENTO")
111                 valorDeRetorno = bd_agendamento_select_all()
```

```
112
113             # CONTROLE_NO_IOT
114             elif operacao == 'f':
115                 print("_CONTROLE_NO_IOT")
116                 valorDeRetorno = no_iot_enviar_comando()
117
118             # CONTROLE_SOM
119             elif operacao == 'g':
120                 print("_CONTROLE_SOM")
121                 comando = ler_conteudo_conexao(conexao,1)
122
123                 # Comando de alterar o volume do som precisa dos 3 bytes do volume
124                 if comando != "a":
125                     comando += ler_conteudo_conexao(conexao,3)
126                     valorDeRetorno = som_configurar(comando)
127
128             # GET ESTIMATIVA DE CONSUMO
129             elif operacao == "h":
130                 print("_GET ESTIMATIVA DE CONSUMO")
131                 valorDeRetorno = bd_no_iot_select_estimativa_de_consumo()
132
133             # REINICIAR ESTIMATIVA DA BATERIA
134             elif operacao == "i":
135                 print("_REINICIAR ESTIMATIVA DA BATERIA")
136                 valorDeRetorno = bd_no_iot_reset_estimativa_da_bateria()
137
138             # SELECT CONFIGURACOES NO_IOT
139             elif operacao == "j":
140                 print("_SELECT CONFIGURACOES NO_IOT")
141                 valorDeRetorno = bd_no_iot_select_configuracoes()
142
143             # UPDATE CONFIGURACOES NO_IOT
144             elif operacao == "k":
145                 print("_UPDATE CONFIGURACOES NO_IOT")
146                 valorDeRetorno = bd_no_iot_update_configuracoes(conexao)
147
148             mutex.release()
149
150             conexao.sendall(valorDeRetorno.encode())
151
152             print(valorDeRetorno.encode());
153
154             print ("\nCLIENT DISCONNECT -------\n");
155         finally:
156             conexao.close()
157         serverSocket.close()
158
159     ############################ CONFIGURAR SOM ############################
160     # Com base no comando especificado eh alterado o volume, iniciado ou
161     # parado a execucao do efeito sonoro (que alerta as aves sobre o
162     # alimento).
163     #
164     # comando:
165     # a: parar som
166     # b: iniciar som e alterar volume
```

```
167   # c: alterar volume
168   #
169   # ativar_servidor_interface
170   ###############################################################################
171   def som_configurar(comando):
172       operacao = comando[0:1]
173
174       # PARAR
175       if operacao == "a":
176           pygame.mixer.music.stop()
177       else:
178       # ALTERAR VOLUME [, INICIAR SOM]
179           volume = int(comando[1:])
180           volume /= 100
181
182           pygame.mixer.music.set_volume(volume)
183
184       # INICIAR
185       if operacao == "b":
186           pygame.mixer.music.play(-1)
187
188       # Sucesso/erro/falha nao implementados
189       return "1"
190
191   ############################ CONTROLAR IOT ############################
192   # Envia um comando de controle ao dispositivo iot de acordo com o IP
193   # do dispositivo. Os IP's sao cadastrados, inicialmente, atraves de
194   # um socket, na funcao ativar_servidor_iot.
195   #
196   # ativar_servidor_iot, ativar_servidor_interface
197   ###############################################################################
198   def no_iot_enviar_comando():
199       global IOT_ADDRESS_LIST
200       global mutex
201
202       # O comando eh enviado para todos os dispositivos iot cadastrados
203       for address in IOT_ADDRESS_LIST:
204           try:
205               with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
206                   server.connect((address, CLIENTE_IOT_PORT))
207                   server.sendall("a".encode())
208           except:
209               print("Erro ao se conectar com iot. IP: %s" %(address))
210               pass
211       # Sucesso/erro/falha nao implementados
212       return "1"
```

util.py

```
001   # Acrescenta zeros ao lado esquerdo do valor numerico informado
002   def formatar_digitos(valor, numeroDeDigitosEsperado):
003       numeroDeDigitosEsperado = int(numeroDeDigitosEsperado)
004       valorStr = str(valor)
005       lenghtValor = len(valorStr)
006
007       # Se o valor for maior que o numeroDeDigitosEsperado entao corta o
```

```
008     # valor e retorna. O corte acontece do valor mais a esquerda para
009     # o mais a direita
010     if lenghtValor > numeroDeDigitosEsperado:
011         valorStr = valorStr[0:numeroDeDigitosEsperado]
012         return valorStr
013 
014     numeroDeDigitosEmFalta = numeroDeDigitosEsperado - lenghtValor
015 
016     # Se o valor numerico for menor que o requisitado entao preenche o
017     # lado esquerdo com zeros (0)
018     for c in range(numeroDeDigitosEmFalta):
019         valorStr = "0"+valorStr
020 
021     return valorStr
```

classes/agendamento.py

```
001 from datetime import datetime, date
002 from bd.sql import *
003 
004 # Gerar dados aleatorios para testes
005 def gerador_aleatorio_de_eventos():
006     query = "insert into
    evento(nome,horario,domingo,segunda,terca,quarta,quinta,sexta,sabado,somTocar,somTe
    mpoDuracao,somVolume) values "
007 
008     values = ""
009     for c in range(0,5000):
010         if c != 0:
011             values += ","
012 
013         nome = c+10
014         horario1 = (c*3) % 24
015         horario2 = (c*7) % 60
016         horario  = str(horario1)+":"+str(horario2)
017         domingo  =  ((c*4) % 11) % 2
018         segunda  =  ((c*5) % 10) % 2
019         terca    =  ((c*6) % 9) % 2
020         quarta   =  ((c*7) % 8) % 2
021         quinta   =  ((c*8) % 7) % 2
022         sexta    =  ((c*3) % 6) % 2
023         sabado   =  ((c*4) % 9) % 2
024         somTocar =  ((c*5) % 4) % 2
025         values  += "(\"evento_agendamento %s\",\"%s\",%s,%s,%s,%s,%s,%s,%s,%s,5,50)"
    %(nome,horario,domingo,segunda,terca,quarta,quinta,sexta,sabado,somTocar)
026 
027     query += values+";"
028     #print(query)
029     banco(query)
030 
031 class Evento:
032     def __init__(self):
033         self.idEvento = 0
034         self.diaEvento = ""
035         self.horarioEvento = ""
036         self.tocarSom = False
```

```
037         self.volumeSom = 0
038         self.somTempoDuracaoSom = 0
039         self.listaDiaDaSemana = ["quinta", "sexta", "sabado", "domingo", "segunda", "terca",
    "quarta"]
040
041     def buscar_proximo_evento(self):
042         diaNumericoDaSemana = date.today().weekday()
043         horarioAtual = datetime.now().strftime("%H:%M")
044
045         for c in range (0,7):
046             diaNumericoDaSemanaASerBuscado = (diaNumericoDaSemana + c) % 7
047             diaStringDaSemana =
    self.listaDiaDaSemana[diaNumericoDaSemanaASerBuscado]
048
049             queryCampos =
    "idAgendamento,horario,somTocar,somVolume,somTempoDuracao"
050             queryTabela = "evento_agendamento"
051             queryCondicao = "%s=1 AND horario > \"%s\"" %(diaStringDaSemana,
    horarioAtual)
052             queryAdicional = "ORDER BY horario ASC LIMIT 1"
053
054             tuplas = sql_select(queryCampos, queryTabela, queryCondicao, queryAdicional)
055
056             possuiTupla = False
057             for tupla in tuplas:
058                 possuiTupla = True
059                 self.idEvento = tupla[0]
060                 self.diaEvento = diaStringDaSemana
061                 self.horarioEvento = str(tupla[1])
062                 self.tocarSom = tupla[2]
063                 self.volumeSom = tupla[3]
064                 self.somTempoDuracaoSom = tupla[4]
065
066             if possuiTupla:
067                 break
068
069             horarioAtual = "00:00"
070
071     def disparar_agora(self):
072         horarioAtual = str(datetime.now().strftime("%H:%M:00"))
073         print(horarioAtual)
074         print(self.horarioEvento)
075
076         # SE horarioEvento <= horarioAtual
077         if horarioAtual >= self.horarioEvento:
078             self.buscar_proximo_evento()
079             return 1
080
081         return 0
082
083     def menor_dia(self,dia1,dia2):
084         indiceDia1 = -1
085         indiceDia2 = -1
086         for c in range(0,6):
087             if listaDiaDaSemana[c] == dia1:
```

```
088             indiceDia1 = c
089         if listaDiaDaSemana[c] == dia2:
090             indiceDia2 = c
091
092     if indiceDia1 < indiceDia2:
093         return 1
094     if indiceDia1 > indiceDia2:
095         return 2
096     return 0
```

bd/agendamento.py

```
001 from bd.sql import *
002 from util import *
003
004 def bd_agendamento_remove(conexao):
005     numeroDeIds = int( ler_conteudo_conexao(conexao,3) );
006
007     queryCondicaoValoresIn = ""
008     for c in range(numeroDeIds):
009         idEvento = ler_conteudo_conexao(conexao,10)
010
011         if c > 0:
012             queryCondicaoValoresIn += ","
013         queryCondicaoValoresIn += idEvento
014
015     queryTabela = "evento_agendamento"
016     queryCondicaoCampo = "idAgendamento"
017
018     sql_delete_where_in(queryTabela, queryCondicaoCampo, queryCondicaoValoresIn)
019
020     return "1"
021
022 # Atualiza as informacoes de um evento de acordo
023 # com os dados recebidos em uma conexao socket
024 def bd_agendamento_update(conexao):
025     idEvento = ler_conteudo_conexao(conexao,10)
026     domingo  = ler_conteudo_conexao(conexao,1)
027     segunda  = ler_conteudo_conexao(conexao,1)
028     terca    = ler_conteudo_conexao(conexao,1)
029     quarta   = ler_conteudo_conexao(conexao,1)
030     quinta   = ler_conteudo_conexao(conexao,1)
031     sexta    = ler_conteudo_conexao(conexao,1)
032     sabado   = ler_conteudo_conexao(conexao,1)
033
034     somHabilitado = ler_conteudo_conexao(conexao,1)
035
036     if somHabilitado == "1":
037         somVolume = ler_conteudo_conexao(conexao,3)
038         somTempoDuracao = ler_conteudo_conexao(conexao,2)
039     else:
040         somVolume = '000'
041         somTempoDuracao = '00'
042
043     horario = ler_conteudo_conexao(conexao,2)
044     horario += ":"
```

```
045     horario += ler_conteudo_conexao(conexao,2)
046
047     lengthNome = int(ler_conteudo_conexao(conexao,2))
048     nome = ler_conteudo_conexao(conexao,lengthNome)
049
050     queryCampos = "somTempoDuracao=%s," %(somTempoDuracao)
051     queryCampos += "somVolume=%s," %(somVolume)
052     queryCampos += "somTocar=%s," %(somHabilitado)
053     queryCampos += "horario='%s'," %(horario)
054     queryCampos += "nome='%s'," %(nome)
055     queryCampos += "domingo=%s," %(domingo)
056     queryCampos += "segunda=%s," %(segunda)
057     queryCampos += "terca=%s," %(terca)
058     queryCampos += "quarta=%s," %(quarta)
059     queryCampos += "quinta=%s," %(quinta)
060     queryCampos += "sexta=%s," %(sexta)
061     queryCampos += "sabado=%s " %(sabado)
062
063     queryTabela = "evento_agendamento"
064     queryCondicao = "idAgendamento=%s" %(idEvento)
065
066     sql_update(queryTabela, queryCampos, queryCondicao)
067
068     return "1"
069
070 def bd_agendamento_insert(conexao):
071     domingo = ler_conteudo_conexao(conexao,1)
072     segunda = ler_conteudo_conexao(conexao,1)
073     terca   = ler_conteudo_conexao(conexao,1)
074     quarta  = ler_conteudo_conexao(conexao,1)
075     quinta  = ler_conteudo_conexao(conexao,1)
076     sexta   = ler_conteudo_conexao(conexao,1)
077     sabado  = ler_conteudo_conexao(conexao,1)
078
079     somHabilitado = ler_conteudo_conexao(conexao,1)
080
081     if somHabilitado == "1":
082         somVolume = ler_conteudo_conexao(conexao,3)
083         somTempoDuracao = ler_conteudo_conexao(conexao,2)
084     else:
085         somVolume = '000'
086         somTempoDuracao = '00'
087
088     horario = ler_conteudo_conexao(conexao,2)
089     horario += ":"
090     horario += ler_conteudo_conexao(conexao,2)
091
092     lengthNome = int(ler_conteudo_conexao(conexao,2))
093     nome = conexao.recv(lengthNome).decode('utf-8')
094
095     queryCampos =
    "nome,horario,domingo,segunda,terca,quarta,quinta,sexta,sabado,somTocar,somVolume,s
    omTempoDuracao"
096     queryTabela = "evento_agendamento"
097     queryDados = "'%s'," %(nome)
```

```
098     queryDados += "'%s'," %(horario)
099     queryDados += "%s," %(domingo)
100     queryDados += "%s," %(segunda)
101     queryDados += "%s," %(terca)
102     queryDados += "%s," %(quarta)
103     queryDados += "%s," %(quinta)
104     queryDados += "%s," %(sexta)
105     queryDados += "%s," %(sabado)
106     queryDados += "%s," %(somHabilitado)
107     queryDados += "%s," %(somVolume)
108     queryDados += "%s" %(somTempoDuracao)
109
110     sql_insert(queryCampos, queryTabela, queryDados)
111
112     return "1"
113
114 def bd_agendamento_select_all():
115     queryCampos =
    "domingo,segunda,terca,quarta,quinta,sexta,sabado,horario,idAgendamento,nome"
116     queryTabela = "evento_agendamento"
117     queryAdicional = "ORDER BY horario ASC"
118
119     data = sql_select_all(queryCampos, queryTabela, queryAdicional)
120
121     retorno = ""
122
123     for tupla in data:
124         domingo  = str(tupla[0])
125         segunda  = str(tupla[1])
126         terca    = str(tupla[2])
127         quarta   = str(tupla[3])
128         quinta   = str(tupla[4])
129         sexta    = str(tupla[5])
130         sabado   = str(tupla[6])
131
132         horario = formatar_horario(tupla[7])
133         idEvento = formatar_id(tupla[8])
134         nome = formatar_nome(str(tupla[9]))
135
136         retorno += idEvento
137         retorno += domingo
138         retorno += segunda
139         retorno += terca
140         retorno += quarta
141         retorno += quinta
142         retorno += sexta
143         retorno += sabado
144         retorno += horario
145         retorno += "%s" %(nome)
146
147     print(len(data))
148
149     totalDeTuplas = formatar_digitos(len(data), 3)
150
151     retorno = totalDeTuplas + retorno
```

```
152     return retorno
153
154 def bd_agendamento_select(conexao):
155     idEvento = ler_conteudo_conexao(conexao,10)
156
157     queryCampos =
"domingo,segunda,terca,quarta,quinta,sexta,sabado,horario,somTocar,somVolume,somTe
mpoDuracao,nome"
158     queryTabela = "evento_agendamento"
159     queryCondicao = "idAgendamento=%s" %(idEvento)
160     queryAdicional = ""
161
162     data = sql_select(queryCampos, queryTabela, queryCondicao, queryAdicional)
163
164     retorno = ""
165
166     for tupla in data:
167         domingo = str(tupla[0])
168         segunda = str(tupla[1])
169         terca   = str(tupla[2])
170         quarta  = str(tupla[3])
171         quinta  = str(tupla[4])
172         sexta   = str(tupla[5])
173         sabado  = str(tupla[6])
174
175         horario = formatar_horario(tupla[7])
176         som = formatar_som(tupla[8],tupla[9],tupla[10])
177         nome = formatar_nome(str(tupla[11])) # nao pode ter ascento
178
179         retorno += domingo
180         retorno += segunda
181         retorno += terca
182         retorno += quarta
183         retorno += quinta
184         retorno += sexta
185         retorno += sabado
186         retorno += horario
187         retorno += som
188         retorno += nome
189
190     return retorno
191
192 def formatar_horario(horario):
193     horario = str(horario).split(":")
194
195     hora = formatar_digitos(horario[0],2)
196     minuto = formatar_digitos(horario[1],2)
197
198     return hora+minuto
199
200 def formatar_som(somHabilitado,somVolume,somTempoDuracao):
201     somHabilitado = str(somHabilitado)
202     somVolume = formatar_digitos(somVolume,3)
203     somTempoDuracao = formatar_digitos(somTempoDuracao,2)
204
```

```
205     if somHabilitado == "1":
206         return somHabilitado+somVolume+somTempoDuracao
207     return somHabilitado
208 
209 def formatar_nome(nome):
210     nomeEncode = nome.encode("utf8")
211     lengthNome = str(len(nomeEncode))
212     lengthNome = formatar_digitos(lengthNome, 2)
213 
214     return lengthNome+nome
215 
216 def formatar_id(id):
217     return formatar_digitos(id, 10)
```

bd/sql.py

```
001 import pymysql
002 
003 DATABASE_HOST = "localhost"
004 DATABASE_USER = "boing"
005 DATABASE_PASSWORD = "boing12345"
006 DATABASE_NAME = "tcc"
007 
008 # Le o conteudo recebido em uma conexao socket
009 def ler_conteudo_conexao(conexao, numeroBytes):
010     return str(conexao.recv(numeroBytes).decode('utf-8'))
011 
012 def banco(query):
013     print(query)
014 
015     # Conecta ao banco de dados
016     conexao = pymysql.connect(
017         host=DATABASE_HOST,
018         user=DATABASE_USER,
019         password=DATABASE_PASSWORD,
020         db=DATABASE_NAME
021     )
022 
023     try:
024         # Insere query ao banco de dados
025         with conexao.cursor() as cursor:
026             cursor.execute(query)
027             retorno = cursor.fetchall()
028         conexao.commit()
029 
030     finally:
031         # Cancela operacao
032         conexao.rollback()
033 
034     conexao.close()
035 
036     return retorno
037 
038 def sql_insert(campos, tabela, dados):
039     query = "INSERT INTO %s(%s) VALUES(%s);" %(tabela, campos, dados)
040     banco(query)
```

```
041
042 def sql_select(campos, tabela, condicao, adicional):
043     query = "SELECT %s FROM %s WHERE %s %s;" %(campos, tabela, condicao,
    adicional)
044     return banco(query)
045
046 def sql_select_all(campos, tabela, adicional):
047     query = "SELECT %s FROM %s %s;" %(campos, tabela, adicional)
048     return banco(query)
049
050 def sql_update(tabela, campos, condicao):
051     query = "UPDATE %s SET %s" %(tabela, campos)
052
053     if condicao != "":
054         query += "WHERE %s" %(condicao)
055
056     query += ";"
057
058     banco(query)
059
060 def sql_delete_where_in(tabela, condicaoCampo, condicaoValoresIn):
061     query = query = "DELETE FROM %s WHERE %s in (%s);" %
    (tabela,condicaoCampo,condicaoValoresIn)
062     banco(query)
```

APÊNDICE C – Código do dispositivo *IoT* (protótipo)

O dispositivo *IoT* foi desenvolvido para um Arduíno UNO. Desse modo, aconselha-se o uso da IDE do fabricante: Arduino IDE.

Carrinho.ino

```
001  #include "Carrinho.hpp"
002  #include <SoftwareSerial.h>
003  #include <Ultrasonic.h>
004  SoftwareSerial antennaEsp(2, 3);
005  Carrinho carrinho(6, 7, 8, 9, 10, 11, 12, 13);
006  Ultrasonic ultrasonic(4, 5);
007
008  int estado = 0;
009  unsigned long tempoInicial = 0;
010  unsigned long tempoFinal = 0;
011  unsigned long intervaloEntreTempos = 0;
012  unsigned long tempoVoltaEsperado = 0;
013
014  float dist = 0;
015
016  void setup() {
017          Serial.begin(115200);
018    antennaEsp.begin(115200);
019    esp_configure();
020  }
021  void esp_configure() {
022    // Configuracoes iniciais
023    esp_data_send("RST");
024    esp_data_send("CWMODE=1");
025    esp_data_send("CWJAP=\"GVT-DFDC\",\"12345\"");
026
027    // Mostra ip do "no" dentro da rede
028    esp_data_send("CIFSR");
029
030    while(!antennaEsp.find("OK"));
031
032    // Registra "no" iot no servidor
033    esp_data_send("CIPSTART=\"TCP\",\"192.168.25.36\",8081");
034    delay(2000);
035
036    // Fecha conexao com o servidor
037    esp_data_send("CIPCLOSE");
038    delay(2000);
039
040    // Cria servidor socket na porta 8080
041    esp_data_send("CIPMUX=1");
042    esp_data_send("CIPSERVER=1,8080");
043    delay(2000);
044  }
045  void esp_data_send(String comando) {
046    // Todo comando deve iniciar com "AT+" e terminar com "\r\n"
```

```
047     comando = "AT+"+comando+"\r\n";
048
049     String resposta;
050     antennaEsp.print(comando);
051
052     delay(2000);
053
054     while(antennaEsp.available()) {
055       char data = antennaEsp.read();
056       resposta += data;
057     }
058
059     Serial.print(resposta);
060   }
061   float get_sensor_distancia() {
062     long microsec = ultrasonic.timing();
063
064     return ultrasonic.convert(microsec, Ultrasonic::CM);
065   }
066
067   void loop() {
068     // Carrinho parado: verifica comando recebido (antena)
069          if (estado == 0) {
070                    if (antennaEsp.available()) {
071           delay(2000);
072           while (antennaEsp.available()) antennaEsp.read();
073
074           tempoInicial = millis();
075           carrinho.andar_sentido_sul();
076                              carrinho.abrir_reservatorio();
077           estado = 1;
078       }
079           // Carrinho andando (sentido 1 - ida)
080     } else if (estado == 1) {
081       float distancia = get_sensor_distancia();
082
083       if (distancia < 7 && distancia > 0) {
084         carrinho.andar_sentido_norte();
085         tempoFinal = millis();
086         estado = 2;
087
088         tempoVoltaEsperado = 2*tempoFinal - tempoInicial;
089       }
090           // Carrinho andando (sentido 2 - volta)
091           } else {
092       unsigned long tempoAtual = millis();
093       if (tempoVoltaEsperado <= tempoAtual) {
094         estado = 0;
095
096         carrinho.parar();
097         carrinho.fechar_reservatorio();
098       }
099     }
100   }
```

Carrinho.hpp

```
001 #ifndef CARRINHO_HPP
002 #define CARRINHO_HPP
003
004 #include "Roda.hpp"
005 #include "Reservatorio.hpp"
006
007 class Carrinho {
008 private:
009         Roda *rodaA, *rodaB;
010         Reservatorio *reservatorio;
011
012 public:
013         Carrinho();
014         Carrinho(int pinA1, int pinA2, int pinB1, int pinB2, int pinC1, int pinC2, int
    pinC3, int pinC4);
015         ~Carrinho();
016
017         void andar_sentido_norte();
018         void andar_sentido_sul();
019         void parar();
020         void abrir_reservatorio();
021         void fechar_reservatorio();
022
023 };
024
025 #endif
```

Reservatorio.hpp

```
001 #ifndef RESERVATORIO_HPP
002 #define RESERVATORIO_HPP
003
004 #include <Stepper.h>
005 #include "Reservatorio.hpp"
006
007 class Reservatorio {
008 private:
009         bool isAberto;
010         Stepper* motor;
011
012 public:
013         Reservatorio();
014         Reservatorio(int pin1, int pin2, int pin3, int pin4);
015         ~Reservatorio();
016
017         void abrir();
018         void fechar();
019
020 };
021
022 #endif
```

Roda.hpp

```
001 #ifndef RODA_HPP
002 #define RODA_HPP
003
004 class Roda {
005 private:
006         int pinA, pinB;
007
008 public:
009         Roda();
010         Roda(int pinA, int pinB);
011         ~Roda();
012
013         void parar();
014         void girar_sentido_horario();
015         void girar_sentido_anti_horario();
016
017 };
018
019 #endif
```

Carrinho.cpp

```
001 #ifndef CARRINHO_CPP
002 #define CARRINHO_CPP
003
004 #include "Carrinho.hpp"
005
006 Carrinho::Carrinho() {}
007 Carrinho::Carrinho(int pinA1, int pinA2, int pinB1, int pinB2, int pinC1, int pinC2, int
    pinC3, int pinC4) {
008         this->rodaA = new Roda(pinA1, pinA2);
009         this->rodaB = new Roda(pinB1, pinB2);
010
011         this->reservatorio = new Reservatorio(pinC1, pinC2, pinC3, pinC4);
012 }
013 Carrinho::~Carrinho() {}
014
015 void Carrinho::andar_sentido_norte() {
016         this->rodaA->girar_sentido_horario();
017         this->rodaB->girar_sentido_horario();
018 }
019 void Carrinho::andar_sentido_sul() {
020         this->rodaA->girar_sentido_anti_horario();
021         this->rodaB->girar_sentido_anti_horario();
022 }
023 void Carrinho::parar() {
024         this->rodaA->parar();
025         this->rodaB->parar();
026 }
027 void Carrinho::abrir_reservatorio() {
028         this->reservatorio->abrir();
029 }
030 void Carrinho::fechar_reservatorio() {
031         this->reservatorio->fechar();
032 }
```

```
033
034 #endif
```

Reservatorio.cpp

```
001 #ifndef RESERVATORIO_CPP
002 #define RESERVATORIO_CPP
003
004 #include "Reservatorio.hpp"
005 #include <Arduino.h>
006
007 Reservatorio::Reservatorio() {}
008 Reservatorio::Reservatorio(int pin1, int pin2, int pin3, int pin4) {
009         this->motor = new Stepper(500, pin1, pin2, pin3, pin4);
010         this->motor->setSpeed(60);
011
012         this->isAberto = false;
013 }
014 Reservatorio::~Reservatorio() {}
015
016 void Reservatorio::abrir() {
017         if (!this->isAberto) {
018                 this->motor->step(1024);
019                 this->isAberto = true;
020         }
021 }
022 void Reservatorio::fechar() {
023         if (this->isAberto) {
024                 this->motor->step(-1024);
025                 this->isAberto = false;
026         }
027 }
028
029 #endif
```

Roda.cpp

```
001 #ifndef RODA_CPP
002 #define RODA_CPP
003
004 #include "Roda.hpp"
005 #include <Arduino.h>
006
007 Roda::Roda() {}
008 Roda::Roda(int pinA, int pinB) {
009         pinMode(pinA, OUTPUT);
010         pinMode(pinB, OUTPUT);
011
012         this->pinA = pinA;
013         this->pinB = pinB;
014 }
015 void Roda::parar() {
016         digitalWrite(this->pinA, LOW);
017         digitalWrite(this->pinB, LOW);
018 }
019 void Roda::girar_sentido_horario() {
```

```
020             digitalWrite(this->pinA, HIGH);
021             digitalWrite(this->pinB, LOW);
022     }
023     void Roda::girar_sentido_anti_horario() {
024             digitalWrite(this->pinA, LOW);
025             digitalWrite(this->pinB, HIGH);
026     }
027
028     #endif
```

APÊNDICE D – Código do sistema *Web*

Para a correta execução do sistema *Web* é necessário obter o arquivo jquery-3.3.1.min.js. O arquivo deve ser obtido no site do desenvolvedor e ser movido para pasta js.

index.php

```
001 <!DOCTYPE html>
002 <html>
003 <?php include_once "class/ClienteServer.php"; ?>
004 <?php include_once "include/head.php"; ?>
005 <body>
006         <header>
007                 <span>CONTROLE</span>
008         </header>
009         <main class="controle">
010                 <section class="iot_movimentacao">
011                         <div class="titulo">
012                                 <span>IOT</span>
013                         </div>
014
015                         <div class="botoes">
016                                 <ul>
017                                         <li>
018                                                 <button id="iot_ativar"
    class="ativar">ATIVAR</button>
019                                         </li>
020                                 </ul>
021                         </div>
022                 </section>
023                 <section class="som">
024                         <div class="titulo">
025                                 <span>Som</span>
026                         </div>
027                         <div class="botoes">
028                                 <ul>
029                                         <li>
030                                                 <button id="som_tocar"
    class="tocar">TOCAR</button>
031                                         </li>
032                                         <li>
033                                                 <button id="som_parar"
    class="parar">PARAR</button>
034                                         </li>
035                                 </ul>
036                         </div>
037                         <div>
038                                 <label for="som_volume">Volume:</label>
039                                 <input type="range" id="som_volume" value="100">
040                         </div>
041                 </section>
042         </main>
043         <footer>
```

```
044                     <ul>
045                             <li data-active="active">
046                                     <a href="javascript: void(0);">CONTROLE</a>
047                             </li>
048                             <li>
049                                     <a href="agenda.php">AGENDA</a>
050                             </li>
051                     </ul>
052             </footer>
053     </body>
054     <?php include_once "include/scripts.php"; ?>
055     </html>
```

evento.php

```
001     <!DOCTYPE html>
002     <html>
003     <?php include_once "class/ClienteServer.php"; ?>
004     <?php include_once "include/head.php"; ?>
005     <?php include_once "util/util.php"; ?>
006     <?php
007             $id = $_GET["id"];
008
009             $tipoDaPagina = "insert";
010             $title = "NOVO AGENDAMENTO";
011
012             $semanaDomingo = "";
013             $semanaSegunda = "";
014             $semanaTerca   = "";
015             $semanaQuarta  = "";
016             $semanaQuinta  = "";
017             $semanaSexta   = "";
018             $semanaSabado  = "";
019
020             $horarioHora   = "";
021             $horarioMinuto = "";
022
023             $nome = "";
024             $somTocar = "";
025             $somVolume = "50";
026             $somTempoDuracao = "00";
027             $somDisabled = "disabled='disabled'";
028
029             if ($id != "") {
030                     $tipoDaPagina = "update";
031                     $title = "EDITAR AGENDAMENTO";
032
033                     $pacote = "d";
034                     $pacote .= formatar_digitos($id, 10, "0");
035
036                     $clienteServer = new ClienteServer();
037                     $clienteServer->criarConexao();
038                     $clienteServer->conectar();
039                     $clienteServer->enviar($pacote);
040
041                     $semanaDomingo = ($clienteServer->ler(1) == "0")? "" : "
```

```
checked=\"checked\"";
042                 $semanaSegunda = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
043                 $semanaTerca   = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
044                 $semanaQuarta  = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
045                 $semanaQuinta  = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
046                 $semanaSexta   = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
047                 $semanaSabado  = ($clienteServer->ler(1) == "0")? "" : "
checked=\"checked\"";
048
049                 $horarioHora   = $clienteServer->ler(2)."";
050                 $horarioMinuto = $clienteServer->ler(2)."";
051
052                 $somTocar = $clienteServer->ler(1)."";
053                 if ($somTocar == "1") {
054                         $somVolume = $clienteServer->ler(3);
055                         $somTempoDuracao = $clienteServer->ler(2);
056                         $somTocar = "checked=\"checked\"";
057                         $somDisabled = "";
058                 } else {
059                         $somTocar = "";
060                 }
061
062                 $lengthNome = $clienteServer->ler(2);
063                 $nome = $clienteServer->ler($lengthNome);
064
065                 $clienteServer->desconectar();
066         }
067 ?>
068 <body>
069         <header>
070                 <span><?php echo $title; ?></span>
071         </header>
072         <main class="evento" data-operacao=<?php echo "".$tipoDaPagina.""; ?> data-
id=<?php echo "".$id.""; ?>>
073                 <form>
074                         <section class="evento">
075                                 <div>
076                                         <label for="nome">Nome:</label>
077                                         <input type="text" name="nome" id="nome"
placeholder="De manha" required="required" value=<?php echo "".$nome.""; ?>>
078                                 </div>
079                                 <div class="hora">
080                                         <label for="horario_hora">Horário:</label>
081                                         <div>
082                                                 <input type="text" class="number"
name="horario_minuto" id="horario_minuto" placeholder="00" required="required"
value=<?php echo "\"".$horarioMinuto."\""; ?>>
083                                                 <div>
084                                                         <span>:</span>
085                                                 </div>
```

```
086                                                 <input type="text" class="number"
    name="horario_hora" id="horario_hora" placeholder="01" required="required" value=<?
    php echo "\"".$horarioHora."\""; ?>>
087                                         </div>
088                                 </div>
089                                 <div>
090                                         <ul>
091                                                 <li>
092                                                         <label
    for="dia_domingo">Domingo</label>
093                                                         <input type="checkbox"
    name="dia_domingo" id="dia_domingo" <?php echo $semanaDomingo; ?>>
094                                                 </li>
095                                                 <li>
096                                                         <label
    for="dia_segunda">Segunda</label>
097                                                         <input type="checkbox"
    name="dia_segunda" id="dia_segunda" <?php echo $semanaSegunda; ?>>
098                                                 </li>
099                                                 <li>
100                                                         <label
    for="dia_terca">Terça</label>
101                                                         <input type="checkbox"
    name="dia_terca" id="dia_terca" <?php echo $semanaTerca; ?>>
102                                                 </li>
103                                                 <li>
104                                                         <label for="dia_quarta">Quarta</
    label>
105                                                         <input type="checkbox"
    name="dia_quarta" id="dia_quarta" <?php echo $semanaQuarta; ?>>
106                                                 </li>
107                                                 <li>
108                                                         <label for="dia_quinta">Quinta</
    label>
109                                                         <input type="checkbox"
    name="dia_quinta" id="dia_quinta" <?php echo $semanaQuinta; ?>>
110                                                 </li>
111                                                 <li>
112                                                         <label
    for="dia_sexta">Sexta</label>
113                                                         <input type="checkbox"
    name="dia_sexta" id="dia_sexta" <?php echo $semanaSexta; ?>>
114                                                 </li>
115                                                 <li>
116                                                         <label
    for="dia_sabado">Sábado</label>
117                                                         <input type="checkbox"
    name="dia_sabado" id="dia_sabado" <?php echo $semanaSabado; ?>>
118                                                 </li>
119                                         </ul>
120                                 </div>
121                         </section>
122                         <section class="som">
123                                 <div class="titulo">
124                                         <span>Som</span>
```

```
125                                     </div>
126                                     <div>
127                                             <label
for="som_input_checkbox">Tocar:</label>
128                                             <input type="checkbox"
name="som_input_checkbox" id="som_input_checkbox" <?php echo $somTocar; ?>>
129                                     </div>
130                                     <div>
131                                             <label
for="som_input_volume">Volume:</label>
132                                             <input type="range" name="som_input_volume"
id="som_input_volume" value=<?php echo """.$somVolume."""; ?> <?php echo
$somDisabled; ?>>
133                                     </div>
134                                     <div class="tempo">
135                                             <label for="som_input_tempo">Tempo:</label>
136                                             <div>
137                                                     <input type="text" class="number"
name="som_input_tempo" id="som_input_tempo" placeholder="0" value=<?php echo """.
$somTempoDuracao."""; ?> <?php echo $somDisabled; ?>>
138                                                     <span class="disabled">segundos</span>
139                                             </div>
140                                     </div>
141                             </section>
142                     </form>
143             </main>
144             <footer>
145                     <ul>
146                             <li>
147                                     <a href="javascript: void(0);" class="salvar" data-
href="agenda.php">SALVAR</a>
148                             </li>
149                             <li>
150                                     <a href="agenda.php"
class="cancelar">CANCELAR</a>
151                             </li>
152                     </ul>
153             </footer>
154     </body>
155     <?php include_once "include/scripts.php"; ?>
156     </html>
```

agenda.php

```
001     <!DOCTYPE html>
002     <html>
003     <?php
004     header("Content-type: text/html;charset=utf-8");
005     include_once "class/ClienteServer.php";
006     include_once "include/head.php";
007     ?>
008
009     <?php
010             $pacote = "e";
011             $clienteServer = new ClienteServer();
012             $clienteServer->criarConexao();
```

```
013             $clienteServer->conectar();
014             $clienteServer->enviar($pacote);
015     ?>
016     <body>
017             <header>
018                     <span>HORÁRIOS</span>
019             </header>
020             <main class="lista_de_evento">
021                     <section>
022                             <table>
023                                     <?php
024                                             $numeroDeAgendamentos =
        intval($clienteServer->ler(3));
025                                             for ($c = 0; $c < $numeroDeAgendamentos; $c+
        +) {
026                                                     $id = $clienteServer->ler(10);
027
028                                                     if ($id == "") {
029                                                             break;
030                                                     }
031
032                                                     $id = intval($id);
033
034                                                     $semanaDomingo = ($clienteServer-
        >ler(1) == "0")? "" : " class=\"bold\"";
035                                                     $semanaSegunda = ($clienteServer-
        >ler(1) == "0")? "" : " class=\"bold\"";
036                                                     $semanaTerca   = ($clienteServer->ler(1)
        == "0")? "" : " class=\"bold\"";
037                                                     $semanaQuarta  = ($clienteServer->ler(1)
        == "0")? "" : " class=\"bold\"";
038                                                     $semanaQuinta  = ($clienteServer->ler(1)
        == "0")? "" : " class=\"bold\"";
039                                                     $semanaSexta   = ($clienteServer->ler(1)
        == "0")? "" : " class=\"bold\"";
040                                                     $semanaSabado  = ($clienteServer->ler(1)
        == "0")? "" : " class=\"bold\"";
041
042                                                     $horarioHora   = $clienteServer->ler(2);
043                                                     $horarioMinuto = $clienteServer->ler(2);
044                                                     $horario = $horarioHora.":".
        $horarioMinuto;
045
046                                                     $lengthNome = intval($clienteServer-
        >ler(2));
047
048                                                     $nome = $clienteServer-
        >ler($lengthNome);
049                                     ?>
050                                                     <tr data-id=<?php echo "\"".$id."\""; ?>>
051                                                             <td class="horario">
052                                                                     <span><?php echo
        $horario; ?></span>
053                                                             </td>
054                                                             <td class="nome">
```

```
055                                                                 <a <?php echo "href='evento.php?id=".$id."'"; ?>><?php echo $nome; ?></a>
056                                                                 <ul>
057                                                                     <li <?php echo "\"".$semanaDomingo."\""; ?>
058                                                                         <span>D</span>
059                                                                     </li>
060                                                                     <li <?php echo "\"".$semanaSegunda."\""; ?>
061                                                                         <span>S</span>
062                                                                     </li>
063                                                                     <li <?php echo "\"".$semanaTerca."\""; ?>
064                                                                         <span>T</span>
065                                                                     </li>
066                                                                     <li <?php echo "\"".$semanaQuarta."\""; ?>
067                                                                         <span>Q</span>
068                                                                     </li>
069                                                                     <li <?php echo "\"".$semanaQuinta."\""; ?>
070                                                                         <span>Q</span>
071                                                                     </li>
072                                                                     <li <?php echo "\"".$semanaSexta."\""; ?>
073                                                                         <span>S</span>
074                                                                     </li>
075                                                                     <li <?php echo "\"".$semanaSabado."\""; ?>
076                                                                         <span>S</span>
077                                                                     </li>
078                                                                 </ul>
079                                                             </td>
080                                                             <td class="status">
081                                                                 <input type="checkbox">
082                                                             </td>
083                                                         </tr>
084                                 <?php
085                                     }
086
087                                     $clienteServer->desconectar();
088                                 ?>
089                             </table>
090                     </section>
091             </main>
092             <footer>
093                     <input type="button" id="insert_event" class="insert" value="+" data-disabled="enabled" data-href="evento.php">
```

```
094                 <input type="button" class="remove" value="-" data-
    disabled="disabled">
095                 <ul>
096                     <li>
097                         <a href="index.php">CONTROLE</a>
098                     </li>
099                     <li data-active="active">
100                         <a href="javascript: void(0);">AGENDA</a>
101                     </li>
102                 </ul>
103             </footer>
104 </body>
105 <?php include_once "include/scripts.php"; ?>
106 </html>
```

util/util.php

```
001 <?php
002
003 function formatar_digitos($valor, $numeroDeDigitosEsperado, $digitoDePreenchimento)
    {
004     $numeroDeDigitosEsperado = intval($numeroDeDigitosEsperado);
005     $valorStr = $valor."";
006     $lenghtValor = strlen($valorStr);
007     $digitoDePreenchimento = "".$digitoDePreenchimento;
008
009     if ($lenghtValor > $numeroDeDigitosEsperado) {
010         $valorStr = substr($valorStr, 0, $numeroDeDigitosEsperado);
011         return $valorStr;
012     }
013
014     $numeroDeDigitosEmFalta = $numeroDeDigitosEsperado - $lenghtValor;
015
016     for ($c = 0; $c < $numeroDeDigitosEmFalta; $c++) {
017         $valorStr = $digitoDePreenchimento.$valorStr;
018     }
019
020     return $valorStr;
021 }
022 ?>
```

include/head.php

```
001 <head>
002         <title>Sistema Fog</title>
003         <link rel="stylesheet" type="text/css" href="css/default.css">
004         <meta name="viewport" content="width=device-width, height=device-height
    minimum-scale=1, maximum-scale=1">
005         <meta charset="utf-8"/>
006 </head>
```

include/scripts.php

```
001 <script type="text/javascript" src="js/jquery-3.3.1.min.js"></script>
002 <script type="text/javascript" src="js/script.js"></script>
```

class/ClienteServer.php

```
001 <?php
002 header("Content-type: text/html;charset=utf-8");
003
004 class ClienteServer {
005         private $host, $port;
006         private $socket;
007
008         function ClienteServer() {
009                 $this->host = "192.168.50.179";//"192.168.25.16";
010                 $this->port = 8086;
011         }
012
013         function criarConexao() {
014                 $this->socket = socket_create(AF_INET, SOCK_STREAM, 0);
015                 return $this->socket? 1 : 0;
016         }
017         function conectar() {
018                 return socket_connect($this->socket, $this->host, $this->port)? 1 : 0;
019         }
020         function desconectar() {
021                 return socket_close($this->socket)? 1 : 0;
022         }
023         function enviar($dado) {
024                 return socket_write ($this->socket, $dado, strlen($dado))? 1 : 0;
025         }
026         function ler($bytes) {
027                 return socket_read ($this->socket, $bytes);
028         }
029 };
030 ?>
```

action/controle/iot.php

```
001 <?php
002 include "../../class/ClienteServer.php";
003 include "../../util/util.php";
004
005 $pacote = "f";
006 $clienteServer = new ClienteServer();
007
008 if ($clienteServer->criarConexao() == 0) {
009         // FALHOU
010         echo json_encode(false);
011         return;
012 }
013 if ($clienteServer->conectar() == 0) {
014         // FALHOU
015         echo json_encode(false);
016         return;
017 }
018 if ($clienteServer->enviar($pacote) == 0) {
019         // FALHOU
020         echo json_encode(false);
021         return;
022 }
```

```
023
024 $respostaDoServidor = $clienteServer->ler(1);
025 $clienteServer->desconectar();
026
027 if ($respostaDoServidor != "1") {
028         // FALHOU
029         echo json_encode(false);
030         return;
031 }
032
033 echo json_encode(true);
034
035
036 ?>
```

action/controle/som.php

```
001 <?php
002 include "../../class/ClienteServer.php";
003 include "../../util/util.php";
004
005 $operacao = $_GET["operacao"];
006 $comando = "";
007
008 switch ($operacao) {
009         case "parar":
010                 $comando = "a";
011                 break;
012         case "tocar":
013                 $comando = "b";
014                 $comando .= formatar_digitos($_GET["valor"], 3, "0");
015                 break;
016         case "alterar_volume":
017                 $comando = "c";
018                 $comando .= formatar_digitos($_GET["valor"], 3, "0");
019                 break;
020         default:
021                 break;
022 }
023
024 $pacote = "g";
025 $pacote .= $comando;
026
027 $clienteServer = new ClienteServer();
028
029 if ($clienteServer->criarConexao() == 0) {
030         // FALHOU
031         echo json_encode(false);
032         return;
033 }
034 if ($clienteServer->conectar() == 0) {
035         // FALHOU
036         echo json_encode(false);
037         return;
038 }
039 if ($clienteServer->enviar($pacote) == 0) {
```

```
040             // FALHOU
041             echo json_encode(false);
042             return;
043     }
044
045     $respostaDoServidor = $clienteServer->ler(1);
046     $clienteServer->desconectar();
047
048     if ($respostaDoServidor != "1") {
049             // FALHOU
050             echo json_encode(false);
051             return;
052     }
053
054     echo json_encode(true);
055
056
057     ?>
```

action/evento/insert.php

```
001     <?php
002     include "../../class/ClienteServer.php";
003     include "../../util/util.php";
004
005     // Dados gerais
006     $nome = $_GET["nome"];
007     $horario = $_GET["horario"];
008
009     // Dados sobre dias de acontecimento do evento
010     $segunda = $_GET["segunda"];
011     $terca = $_GET["terca"];
012     $quarta = $_GET["quarta"];
013     $quinta = $_GET["quinta"];
014     $sexta = $_GET["sexta"];
015     $sabado = $_GET["sabado"];
016     $domingo = $_GET["domingo"];
017
018     // Dados do som
019     $somTocar = $_GET["somTocar"];
020     $somVolume = $_GET["somVolume"];
021     $somTempo = $_GET["somTempo"];
022
023     // Formacao do pacote a ser enviado (de acordo com o protocolo definido)
024     $pacote = "a";
025     $pacote .= ($domingo == "1"  || $domingo == 1)?  "1" : "0";
026     $pacote .= ($segunda == "1"  || $segunda == 1)?  "1" : "0";
027     $pacote .= ($terca == "1"    || $terca == 1)?    "1" : "0";
028     $pacote .= ($quarta == "1"   || $quarta == 1)?   "1" : "0";
029     $pacote .= ($quinta == "1"   || $quinta == 1)?   "1" : "0";
030     $pacote .= ($sexta == "1"    || $sexta == 1)?    "1" : "0";
031     $pacote .= ($sabado == "1"   || $sabado == 1)?   "1" : "0";
032
033     if ($somTocar == "1" || $somTocar == 1) {
034             $pacote .= "1";
035
```

```
036             $pacote .= formatar_digitos($somVolume, 3, 0);
037             $pacote .= formatar_digitos($somTempo, 2, 0);
038     } else {
039             $pacote .= "0";
040     }
041
042     $pacote .= formatar_digitos($horario, 4, 0);
043     $pacote .= formatar_digitos(strlen($nome), 2, 0);
044     $pacote .= substr($nome, 0, 100);
045
046     $clienteServer = new ClienteServer();
047
048     if ($clienteServer->criarConexao() == 0) {
049             // FALHOU
050             echo json_encode(false);
051             return;
052     }
053     if ($clienteServer->conectar() == 0) {
054             // FALHOU
055             echo json_encode(false);
056             return;
057     }
058     if ($clienteServer->enviar($pacote) == 0) {
059             // FALHOU
060             echo json_encode(false);
061             return;
062     }
063
064     $respostaDoServidor = $clienteServer->ler(1);
065     $clienteServer->desconectar();
066
067     if ($respostaDoServidor != "1") {
068             // FALHOU
069             echo json_encode(false);
070             return;
071     }
072
073     echo json_encode(true);
074
075     ?>
```

action/evento/remove.php

```
001     <?php
002     include "../../class/ClienteServer.php";
003     include "../../util/util.php";
004
005     // Dados gerais
006     $id = explode(",", $_GET["id"]);
007     $pacote = "b";
008     $pacote .= formatar_digitos(count($id), 3, 0);
009
010     for ($c = 0; $c < count($id); $c++) {
011             $pacote .= formatar_digitos($id[$c], 10, 0);
012     }
013
```

```
014 $clienteServer = new ClienteServer();
015 if ($clienteServer->criarConexao() == 0) {
016         // FALHOU
017         echo json_encode(false);
018         return;
019 }
020 if ($clienteServer->conectar() == 0) {
021         // FALHOU
022         echo json_encode(false);
023         return;
024 }
025 if ($clienteServer->enviar($pacote) == 0) {
026         // FALHOU
027         echo json_encode(false);
028         return;
029 }
030
031 $respostaDoServidor = $clienteServer->ler(1);
032 $clienteServer->desconectar();
033
034 if ($respostaDoServidor != "1") {
035         // FALHOU
036         echo json_encode(false);
037         return;
038 }
039
040 echo json_encode(true);
041 ?>
```

action/controle/update.php

```
001 <?php
002 include "../../class/ClienteServer.php";
003 include "../../util/util.php";
004
005 // Dados gerais
006 $id = $_GET["id"];
007 $nome = $_GET["nome"];
008 $horario = $_GET["horario"];
009
010 // Dados sobre dias de acontecimento do evento
011 $segunda = $_GET["segunda"];
012 $terca = $_GET["terca"];
013 $quarta = $_GET["quarta"];
014 $quinta = $_GET["quinta"];
015 $sexta = $_GET["sexta"];
016 $sabado = $_GET["sabado"];
017 $domingo = $_GET["domingo"];
018
019 // Dados do som
020 $somTocar = $_GET["somTocar"];
021 $somVolume = $_GET["somVolume"];
022 $somTempo = $_GET["somTempo"];
023
024 // Formacao do pacote a ser enviado (de acordo com o protocolo definido)
025 $pacote = "c";
```

```
026 $pacote .= formatar_digitos($id, 10, 0);
027 $pacote .= ($domingo == "1"  || $domingo == 1)?  "1" : "0";
028 $pacote .= ($segunda == "1"  || $segunda == 1)?  "1" : "0";
029 $pacote .= ($terca == "1"    || $terca == 1)?    "1" : "0";
030 $pacote .= ($quarta == "1"   || $quarta == 1)?   "1" : "0";
031 $pacote .= ($quinta == "1"   || $quinta == 1)?   "1" : "0";
032 $pacote .= ($sexta == "1"    || $sexta == 1)?    "1" : "0";
033 $pacote .= ($sabado == "1"   || $sabado == 1)?   "1" : "0";
034
035 if ($somTocar == "1" || $somTocar == 1) {
036         $pacote .= "1";
037
038         $pacote .= formatar_digitos($somVolume, 3, 0);
039         $pacote .= formatar_digitos($somTempo, 2, 0);
040 } else {
041         $pacote .= "0";
042 }
043
044 $pacote .= formatar_digitos($horario, 4, 0);
045 $pacote .= formatar_digitos(strlen($nome), 2, 0);
046
047 $pacote .= substr($nome, 0, 100);
048
049 $clienteServer = new ClienteServer();
050
051 if ($clienteServer->criarConexao() == 0) {
052         // FALHOU
053         echo json_encode(false);
054         return;
055 }
056 if ($clienteServer->conectar() == 0) {
057         // FALHOU
058         echo json_encode(false);
059         return;
060 }
061 if ($clienteServer->enviar($pacote) == 0) {
062         // FALHOU
063         echo json_encode(false);
064         return;
065 }
066
067 $respostaDoServidor = $clienteServer->ler(1);
068 $clienteServer->desconectar();
069
070 if ($respostaDoServidor != "1") {
071         // FALHOU
072         echo json_encode(false);
073         return;
074 }
075
076 echo json_encode(true);
077 ?>
```

css/default.php

```
001  * {
002          margin: 0px;
003          padding: 0px;
004          list-style-type: none;
005          text-decoration: none;
006          color: rgb(15,15,15);
007          font-family: sans-serif;
008          -webkit-box-sizing: border-box;
009          -moz-box-sizing: border-box;
010          box-sizing: border-box;
011          z-index: 1;
012          font-size: 14px;
013  }
014  html, body {
015          width: 100%;
016          height: 100%;
017  }
018  div.centralizado {
019          width: 100%;
020          margin: auto;
021          padding: 0px;
022          position: relative;
023          height: 100%;
024          min-height: 100%;
025  }
026
027  /* ----------- Button ----------- */
028  button, a.button{
029          padding: 5px 13px;
030          border-width: 1px;
031          border-style: solid;
032          border-radius: 3px;
033          cursor: pointer;
034          outline: none;
035  }
036  button.green, a.button.green {
037          color: white;
038          background-color: rgb(5, 159, 42);
039          background-image: -webkit-linear-gradient(top, rgb(5, 159, 42), rgb(0, 95, 10));
040          background-image: -moz-linear-gradient(top, rgb(20, 160, 60), rgb(0, 130, 30));
041     background-image: -o-linear-gradient(top, rgb(20, 160, 60), rgb(0, 130, 30));
042     border-color: rgb(0, 130, 20);
043          text-shadow: 1px 1px black;
044  }
045  button.green:hover, a.button.green:hover {
046          background-color: rgb(17, 169, 52);
047          background-image: -webkit-linear-gradient(top, rgb(17, 169, 52), rgb(5, 105, 20));
048          background-image: -moz-linear-gradient(top, rgb(17, 169, 52), rgb(5, 105, 20));
049     background-image: -o-linear-gradient(top, rgb(17, 169, 52), rgb(5, 105, 20));
050          border-color: rgb(0, 149, 32);
051  }
052  button.black, a.button.black {
053          color: white;
054          background-color: rgb(40, 40, 40);
```

```
055             background-image: -webkit-linear-gradient(top, rgb(50, 50, 50), rgb(0, 0, 0));
056             background-image: -moz-linear-gradient(top, rgb(50, 50, 50), rgb(0, 0, 0));
057        background-image: -o-linear-gradient(top, rgb(50, 50, 50), rgb(0, 0, 0));
058             border-color: rgb(10,10,10);
059             text-shadow: 1px 1px black;
060     }
061     button.black:hover, a.button.green:hover {
062             background-color: rgb(0, 106, 194);
063             background-image: -webkit-linear-gradient(top, rgb(30, 30, 30), rgb(0, 0, 0));
064             background-image: -moz-linear-gradient(top, rgb(30, 30, 30), rgb(0, 0, 0));
065        background-image: -o-linear-gradient(top, rgb(30, 30, 30), rgb(0, 0, 0));
066             border-color: rgb(10,10,10);
067     }
068     button.red, a.button.red {
069             color: white;
070             background-color: rgb(180, 0, 0);
071             background-image: -webkit-linear-gradient(top, rgb(180, 0, 0), rgb(80, 0, 0));
072             background-image: -moz-linear-gradient(top, rgb(180, 0, 0), rgb(80, 0, 0));
073        background-image: -o-linear-gradient(top, rgb(180, 0, 0), rgb(80, 0, 0));
074             border-color: rgb(100,0,0);
075             text-shadow: 1px 1px black;
076     }
077     button.red:hover, a.button.green:hover {
078             background-color: rgb(170, 0, 0);
079             background-image: -webkit-linear-gradient(top, rgb(170, 0, 0), rgb(70, 0, 0));
080             background-image: -moz-linear-gradient(top, rgb(170, 0, 0), rgb(70, 0, 0));
081        background-image: -o-linear-gradient(top, rgb(170, 0, 0), rgb(70, 0, 0));
082             border-color: rgb(100,0,0);
083     }
084
085     /* ----------- Input ----------- */
086     input[type='number'] {
087             width: 80px;
088     }
089     input[type='number'],
090     input[type='text'] {
091             height: 20px;
092             padding: 13px;
093             border: 1px rgb(180,180,180) solid;
094     }
095     input[type='file'],
096     input[type='range'] {
097             width: 41vw;
098     }
099     input[type='file']:disabled {
100             color: rgb(130, 130, 130);
101     }
102     input[type='checkbox'] {
103             width: 28px;
104             height: 28px;
105             cursor: pointer;
106     }
107     input[type='number']:disabled,
108     input[type='text']:disabled {
109             background-color: rgb(240,240,240);
```

```
110         color: rgb(120,120,120);
111 }
112
113 /* ----------- Input[range] ----------- */
114 input[type='range'] {
115         -webkit-appearance: none;
116         background-color: blue;
117         margin: 10px 0;
118         border-radius: 5px;
119         height: 7px;
120         cursor: pointer;
121 }
122 input[type=range]:focus {
123         outline: none;
124 }
125 input[type='range']:disabled {
126         -webkit-appearance: none;
127         background-color: rgb(225, 225, 225);
128         margin: 10px 0;
129         border-radius: 5px;
130         height: 7px;
131         cursor: pointer;
132 }
133 input[type=range]::-webkit-slider-thumb {
134         width: 10px;
135         height: 23px;
136         border: 1px solid #2497E3;
137         background-color: #A1D0FF;
138         -webkit-appearance: none;
139         box-shadow: 0px 0px 0px #000000;
140 }
141 input[type=range]::-ms-thumb {
142         width: 10px;
143         height: 23px;
144         border: 1px solid #2497E3;
145         background-color: #A1D0FF;
146         -webkit-appearance: none;
147         box-shadow: 0px 0px 0px #000000;
148 }
149 input[type=range]::-moz-range-thumb {
150         width: 10px;
151         height: 23px;
152         border: 1px solid #2497E3;
153         background-color: #A1D0FF;
154         -webkit-appearance: none;
155         box-shadow: 0px 0px 0px #000000;
156 }
157 input[type=range]:disabled::-webkit-slider-thumb {
158         background-color: rgb(225, 225, 225);
159         border-color: rgb(190,190,190);
160 }
161 input[type=range]:disabled::-ms-thumb {
162         background-color: rgb(225, 225, 225);
163         border-color: rgb(190,190,190);
164 }
```

```
input[type=range]:disabled::-moz-range-thumb {
	background-color: rgb(225, 225, 225);
	border-color: rgb(190,190,190);
}

/* ----------- Header ----------- */
header {
	width: 100%;
	float: left;
	background-color: blue;
	position: fixed;
	z-index: 100;
	padding: 0px 23px;
}
header span,
header a {
	font-size: 12px;
	font-weight: bold;
	text-shadow: 1px 1px 2px black;
}
header span {
	color: white;
	float: left;
	margin: 18px 0px;
}
header a {
	color: yellow;
	float: right;
	padding: 18px;
	background-color: rgba(0,0,0,0.2);
}

/* ----------- Footer ----------- */
footer {
	width: 100vw;
	height: 60px;
	float: none;
	padding: 0px;
	position: fixed;
	bottom: 0;
	background-color: blue;
}
footer ul {
	width: 100%;
	height: 100%;
	float: left;
	display: table;
}
footer li {
	height: 100%;
	display: table-cell;
	vertical-align: middle;
	text-align: center;
	cursor: pointer;
}
```

```
footer li:hover {
	background-color: rgba(0,0,0,0.2);
}
footer li[data-active='active'] {
	background-color: rgba(0,0,0,0.3);
}
footer input[type='button'] {
	width: 55px;
	height: 55px;
	border-radius: 30px;
	font-size: 20px;
	font-weight: bold;
	margin-bottom: 35px;
	border-width: 0px;
	color: white;
	text-shadow: 1px 1px 2px black;
	cursor: pointer;
	position: absolute;
	bottom: 40px;
}
footer input[type='button'].insert {
	left: 0;
	margin-left: 10px;
	background-color: rgb(0, 170, 0);
	background-image: -webkit-linear-gradient(top, rgb(0, 200, 0), rgb(0, 110, 0));
	background-image: -moz-linear-gradient(top, rgb(0, 200, 0), rgb(0, 110, 0));
   background-image: -o-linear-gradient(top, rgb(0, 200, 0), rgb(0, 110, 0));
   display: block;
}
footer input[type='button'].remove {
	right: 0;
	margin-right: 10px;
	background-color: rgb(230, 0, 0);
	background-image: -webkit-linear-gradient(top, rgb(230, 0, 0), rgb(150, 0, 0));
	background-image: -moz-linear-gradient(top, rgb(230, 0, 0), rgb(150, 0, 0));
   background-image: -o-linear-gradient(top, rgb(230, 0, 0), rgb(150, 0, 0));
   display: block;
}
footer input[type='button'].remove[data-disabled='disabled'],
footer input[type='button'].insert[data-disabled='disabled'] {
	display: none;
}
footer a,
footer button {
	padding: 0px 15px;
	color: white;
	font-size: 12px;
	font-weight: bold;
	text-shadow: 1px 1px 2px black;
}
footer button {
	background-image: none;
	border-width: 0px;
	border-radius: 0px;
	margin: 0px;
```

```
}
footer button:hover {
	background-color: blue;
	background-image: none;
}

/* ----------- Section ----------- */
section {
	width: 100%;
	max-width: 100vw;
	float: left;
	padding: 0px 23px;
	margin-top: 20px;
}
section input {
	float: left;
}
section input[type='file'] {
	height: 25vh;
	padding-top: 2px;
}

/* ----------- Main ----------- */
main {
	width: 100%;
	min-height: 100%;
	padding-bottom: 80px;
	float: left;
	position: relative;
	padding-top: 50px;
}
main div {
	width: 100%;
	float: left;
	margin-top: 15px;
	display: table;
}
main div.titulo {
	width: 100%;
	float: left;
	border-bottom: 1px black solid;
	margin-bottom: 20px;
	padding-bottom: 5px;
}
main label {
	width: 100vw;
	height: 27px;
	display: table-cell;
	vertical-align: middle;
}
main span.disabled {
	color: rgb(130, 130, 130);
}
main button {
	float: left;
```

```
330         /*margin-left: 2vw;*/
331 }
332 main section div div {
333         width: 42vw;
334         margin: 0px;
335 }
336
337 /* ----------- Main.EstimativaDeConsumo ----------- */
338 main.estimativa_de_consumo table {
339         width: 100%;
340         border: 1px rgb(200,200,200) solid;
341         border-collapse: collapse;
342         margin-top: 12px;
343 }
344 main.estimativa_de_consumo table td span {
345         float: right;
346 }
347 main.estimativa_de_consumo table th,
348 main.estimativa_de_consumo table td {
349         width: 20%;
350         padding: 10px;
351         border: 1px rgb(200,200,200) solid;
352 }
353 main.estimativa_de_consumo table th span,
354 main.estimativa_de_consumo table td span {
355         font-size: 12px;
356 }
357 main.estimativa_de_consumo table th {
358         text-align: left;
359         background-color: rgb(230,230,230);
360 }
361 main.estimativa_de_consumo table tr.head th {
362         text-align: center;
363 }
364 main.estimativa_de_consumo table tr button {
365         float: none;
366         margin: 0px;
367 }
368 main.estimativa_de_consumo table td input {
369         width: 100%;
370 }
371 main.estimativa_de_consumo table#tarifa td {
372         width: 50%;
373 }
374 main.estimativa_de_consumo button {
375         width: 100%;
376         height: 15vh;
377         color: white;
378         font-weight: bold;
379         background-color: rgb(0,180,0);
380         border-color: rgb(0,180,0);
381         margin-top: 20px;
382 }
383
384 /* ----------- Main.Manual ----------- */
```

```
main.manual section div div {
	width: 55vw;
	margin: 0px;
	float: right;
}
main.manual section button,
main.manual section input {
	float: right;
}

/* ---------- Main.horarios ---------- */
main.lista_de_evento {
	padding-bottom: 60px;
}
main.lista_de_evento section {
	width: 100%;
	padding: 0px;
	margin: 0px;
}
main.lista_de_evento section table {
	width: 100%;
	border-collapse: collapse;
	margin-bottom: 90px;
}
main.lista_de_evento section td {
	padding: 30px 10px;
	border-bottom: 1px blue solid;
}
main.lista_de_evento section td.horario {
	text-align: center;
}
main.lista_de_evento section td.horario span {
	color: rgb(0,0,220);
	font-size: 32px;
	font-family: sans-serif;
	font-weight: bold;
}
main.lista_de_evento section td.nome a {
	font-weight: bold;
	border-bottom-width: 1px;
	border-bottom-style: solid;
	border-bottom-color: rgb(255,255,255);
}
main.lista_de_evento section td.nome a:hover {
	border-bottom-color: black;
}
main.lista_de_evento section td.nome ul {
	width: 100%;
	float: left;
}
main.lista_de_evento section td.nome ul li {
	width: auto;
	float: left;
	margin-top: 10px;
	margin-right: 5px;
```

```
440 }
441 main.lista_de_evento section td.nome li span {
442         color: rgb(170,170,170);
443         font-weight: normal;
444 }
445 main.lista_de_evento section td.nome li.bold span {
446         color: darkblue;
447         font-weight: bold;
448 }
449 main.lista_de_evento section td.status input {
450         position: relative;
451         margin: auto;
452         color: red;
453 }
454 main.lista_de_evento section tr.disabled td.horario span,
455 main.lista_de_evento section tr.disabled td.nome span,
456 main.lista_de_evento section tr.disabled td.nome a {
457         color: rgb(170,170,170);
458 }
459 main.lista_de_evento section tr.disabled td.nome li.bold span {
460         color: rgb(120,120,120);
461 }
462
463 /* ---------- Main.Evento ---------- */
464 main.evento section input.number {
465         width: 17vw;
466         vertical-align: bottom;
467         margin-right: 10px;
468 }
469 main.evento section.evento input.number {
470         vertical-align: middle;
471         margin-right: 0px;
472 }
473 main.evento section.evento input#nome {
474         width: 41vw;
475 }
476 main.evento section.evento ul label {
477         width: 30vw;
478         height: 37px;
479         float: left;
480 }
481 main.evento section.evento ul {
482         width: 100%;
483         float: left;
484 }
485 main.evento section.evento li {
486         width: 100%;
487         float: left;
488         margin: 5px 0px;
489 }
490 main.evento section.evento input {
491         float: right;
492 }
493 main.evento section div.hora div {
494         float: right;
```

```
}
main.evento section div.hora div div {
	width: auto;
}
main.evento section div.hora span {
	width: 7vw;
	text-align: center;
	vertical-align: middle;
}
main.evento section div.tempo span,
main.evento section div.distancia span {
	margin-left: 10px;
}
main.evento section div div span {
	height: 28px;
	float: none;
	display: table-cell;
	vertical-align: bottom;
}

/* ----------- Main.Evento ----------- */

main.controle section.iot div.botoes {
	width: 100%;
	float: left;
}
main.controle section div.botoes ul,
main.controle section div.botoes li {
	width: 100%;
	float: left;
	margin-bottom: 5px;
}
main.controle section button {
	width: 100%;
	height: 20vh;
	float: left;
	color: white;
	border-radius: 0px;
	font-weight: bold;
}
main.controle section.iot_movimentacao button.ativar {
	background-color: rgb(0,120,0);
	border-color: rgb(0,120,0);
}

main.controle section.som button.tocar {
	background-color: rgb(0,120,0);
	border-color: rgb(0,120,0);
}
main.controle section.som button.parar {
	background-color: rgb(180,0,0);
	border-color: rgb(180,0,0);
}

/* ----------- Main.Configuracoes ----------- */
```

```
main.configuracoes input[type='number'],
main.configuracoes input[type='text'] {
	width: 17vw;
	vertical-align: bottom;
}
```

js/script.js

```
$(document).ready(function() {
	var main = $("main")[0];
	var pagina = $(main).attr("class");

	switch (pagina) {
		case "estimativa_de_consumo":
			page_estimativa_de_consumo();
			break;
		case "evento":
			page_evento();
			break;
		case "lista_de_evento":
			page_lista_de_evento();
			break;
		case "configuracoes":
			page_configuracoes();
			break;
		case "controle":
			page_controle();
			break;
		default:
			break;
	}

	$("footer li").click(function() {
		$(this).find("a")[0].click();
	});
});
function Ajax(url, data, function_sucess, function_error) {
	if (data != undefined && data != "") {
		data = "?"+data;
	} else {
		data = "";
	}

	$.ajax({
		url: url+data,
		dataType: "json",
		contentType: "application/json; charset=utf-8",
		success: function(d) {
			function_sucess(d);
		},
		type: 'GET',
		error: function(d) {
			function_error(d);
		}
	});
```

```
048 }
049
050 const message_box_type = {
051         SUCESS: 0,
052         ERROR: 1,
053         FAIL: 2
054 };
055 // Futuramente implementar algo melhor (nao usar esse alert padrao)
056 function message_box(box_type, message) {
057         switch(box_type) {
058                 case message_box_type.SUCESS:
059                         alert("SUCESSO\n"+message);
060                         break;
061                 case message_box_type.ERROR:
062                         alert("ERRO\n"+message);
063                         break;
064                 case message_box_type.FAIL:
065                         alert("OPERAÇÃO FALHOU\n"+message);
066                         break;
067                 default:
068                         break;
069         }
070 }
071
072 /* ---------- [Class] FrontIndexTarifa ---------- */
073 class FrontEstimativaDeConsumo {
074         static reiniciarEstimativaDeConsumo() {
075                 function function_sucess(data) {
076                         message_box(
077                                 message_box_type.SUCESS,
078                                 "Tempo de duração da bateria\natualizado com sucesso!"
079                         );
080                 }
081                 function function_error(data) {
082                         message_box(
083                                 message_box_type.FAIL,
084                                 "Algo inesperado ocorreu no servidor.\nVerifique a
    corretude dos dados e tente mais tarde."
085                         );
086                 }
087
088
        BackEstimativaDeConsumo.reiniciarEstimativaDeConsumo(function_sucess,
    function_error);
089         }
090 };
091 class BackEstimativaDeConsumo {
092         static reiniciarEstimativaDeConsumo(front_function_sucess,
    front_function_error) {
093                 function function_sucess(data) {
094                         console.log("Sucess:");
095                         console.log(data);
096
097                         if (data) {
```

```
098                                     front_function_sucess(data);
099                             } else {
100                                     front_function_error(data);
101                             }
102                     }
103                     function function_error(data) {
104                             front_function_error(data);
105                     }
106
107                     Ajax("action/estimativa_de_consumo/reiniciar_estimativa.php", "",
    function_sucess, function_error);
108             }
109     }
110
111     /* ---------- [FunctionPage] Index ---------- */
112     function page_estimativa_de_consumo() {
113             $("main button").click(function() {
114                     FrontEstimativaDeConsumo.reiniciarEstimativaDeConsumo($(this));
115             });
116     }
117
118     /* ---------- [Class] FrontConfiguracoes ---------- */
119     class FrontConfiguracoes {
120             static salvar(button) {
121                     var noIotTempoAtividade = $("#no_iot_tempo_atividade").val();
122
123                     if ($(button).prop("disabled") == true) {
124                             return;
125                     }
126
127                     if (noIotTempoAtividade == "" || noIotTempoAtividade == undefined) {
128                             message_box(
129                                     message_box_type.ERROR,
130                                     "Insira um tempo de atividade para o nó iot."
131                             );
132                             return;
133                     }
134
135                     var bateriaCapacidade = $("#bateria_capacidade").val();
136
137                     if (bateriaCapacidade == "" || bateriaCapacidade == undefined) {
138                             message_box(
139                                     message_box_type.ERROR,
140                                     "Insira a capacidade (corrente, em mA) da\nbateria
    utilizada no nó iot."
141                             );
142                             return;
143                     }
144
145                     function function_sucess(data) {
146                             message_box(
147                                     message_box_type.SUCESS,
148                                     "Salvo com sucesso!"
149                             );
```

```
150                             $(button).prop("disabled", false)
151                             $(window.document.location).attr("href", $(button).attr("data-
    href"));
152                     }
153                     function function_error(data) {
154                             message_box(
155                                     message_box_type.FAIL,
156                                     "Algo inesperado ocorreu no servidor.\nVerifique a
    corretude dos dados e tente mais tarde."
157                             );
158                             $(button).prop("disabled", false)
159                     }
160
161                     $(button).prop("disabled", true)
162
163                     BackConfiguracoes.update(function_sucess, function_error);
164             }
165     }
166
167     /* ---------------------- [Class] BackConfiguracoes ---------------------- */
168     class BackConfiguracoes {
169             static update(front_function_sucess, front_function_error) {
170                     // Dados gerais
171                     var noIotTempoAtividade = $("#no_iot_tempo_atividade").val()+"";
172                     var noIotVelocidade = $("#no_iot_velocidade").val()+""
173                     var bateriaCapacidade = $("#bateria_capacidade").val()+"";
174                     var consumoAtivado = $("#consumo_ativado").val()+"";
175                     var consumoDesativado = $("#consumo_desativado").val()+"";
176
177                     function function_sucess(data) {
178                             console.log("Sucess:");
179                             console.log(data);
180
181                             if (data) {
182                                     front_function_sucess(data);
183                             } else {
184                                     front_function_error(data);
185                             }
186                     }
187                     function function_error(data) {
188                             console.log(data);
189                             front_function_error(data);
190                     }
191
192                     var dado = "";
193                     dado += "noIotTempoAtividade="+noIotTempoAtividade;
194                     dado += "&noIotVelocidade="+noIotVelocidade;
195                     dado += "&bateriaCapacidade="+bateriaCapacidade;
196                     dado += "&consumoAtivado="+consumoAtivado;
197                     dado += "&consumoDesativado="+consumoDesativado;
198
199                     Ajax("action/configuracoes/configuracoes.php", dado, function_sucess,
    function_error);
200             }
```

```
}


/* ---------- [FunctionPage] Configuracoes ---------- */
function page_configuracoes() {
	var parametrosUrl = window.location.search.replace("?", "");
	var parametroPage = parametrosUrl.split("=");

	$($("footer a")[0]).attr("data-href", parametroPage[1]+".php");
	$($("footer a")[1]).attr("href", parametroPage[1]+".php");

	$("footer a.salvar").click(function() {
		FrontConfiguracoes.salvar($(this));
	});
}

/* ---------- [Class] Controle ---------- */

class BackControle {
	static iot_ativar() {
		function function_sucess(data) {
			console.log("sucesso");
			console.log(data);
		}
		function function_error(data) {
			console.log("erro");
			console.log(data);
		}

		Ajax("action/controle/iot.php", "", function_sucess, function_error);
	}


	static som_tocar() {
		function function_sucess(data) {
			console.log("sucesso");
			console.log(data);
		}
		function function_error(data) {
			console.log("erro");
			console.log(data);
		}

		var dado = "";
		dado += "operacao=tocar";
		dado += "&valor="+$("#som_volume").val();
		console.log(dado);
		Ajax("action/controle/som.php", dado, function_sucess, function_error);
	}
	static som_parar() {
		function function_sucess(data) {
			console.log("sucesso");
			console.log(data);
		}
```

```
255                     function function_error(data) {
256                             console.log("erro");
257                             console.log(data);
258                     }
259
260                     Ajax("action/controle/som.php", "operacao=parar", function_sucess,
    function_error);
261             }
262             static som_alterar_volume(button) {
263                     function function_sucess(data) {
264                             console.log("sucesso");
265                             console.log(data);
266                     }
267                     function function_error(data) {
268                             console.log("erro");
269                             console.log(data);
270                     }
271
272                     var dado = "";
273                     dado += "operacao=alterar_volume";
274                     dado += "&valor="+$(button).val();
275
276                     Ajax("action/controle/som.php", dado, function_sucess, function_error);
277             }
278     }
279
280     /* ---------------------- [FunctionPage] Controle ---------------------- */
281     function page_controle() {
282             $("button#iot_ativar").click(function() {
283                     console.log("iot_parar");
284                     BackControle.iot_ativar();
285             });
286
287
288             $("button#som_tocar").click(function() {
289                     console.log("som_tocar");
290                     BackControle.som_tocar();
291             });
292             $("button#som_parar").click(function() {
293                     console.log("som_parar");
294                     BackControle.som_parar();
295             });
296             $("input#som_volume").change(function() {
297                     console.log("volume");
298                     BackControle.som_alterar_volume($(this));
299             });
300     }
301
302     /* ---------------------- [Class] EventoSom ---------------------- */
303     class FrontEvento {
304             static ativarSom() {
305                     $("#som_input_tempo").prop("disabled", false);
306                     $("#som_input_volume").prop("disabled", false);
307                     $("#som_input_arquivo").prop("disabled", false);
```

```
308
309                     $("#som_input_tempo").prop("required", true);
310                     $("#som_input_volume").prop("required", true);
311                     $("#som_input_arquivo").prop("required", true);
312                     console.log("ativar");
313             }
314             static desativarSom() {
315                     $("#som_input_tempo").prop("disabled", true);
316                     $("#som_input_volume").prop("disabled", true);
317                     $("#som_input_arquivo").prop("disabled", true);
318
319                     $("#som_input_tempo").prop("required", false);
320                     $("#som_input_volume").prop("required", false);
321                     $("#som_input_arquivo").prop("required", false);
322                     console.log("desativar");
323             }
324             static salvar(button) {
325                     var nome = $("#nome").val();
326
327                     if (nome == "" || nome == undefined) {
328                             message_box(
329                                     message_box_type.ERROR,
330                                     "Insira um nome para o evento."
331                             );
332                             return;
333                     }
334
335                     function function_sucess(data) {
336                             message_box(
337                                     message_box_type.SUCESS,
338                                     "Salvo com sucesso!"
339                             );
340                             $(window.document.location).attr("href", $(button).attr("data-
    href"));
341                     }
342                     function function_error(data) {
343                             message_box(
344                                     message_box_type.FAIL,
345                                     "Algo inesperado ocorreu no servidor.\nVerifique a
    corretude dos dados e tente mais tarde."
346                             );
347                     }
348
349                     BackEvento.salvar(function_sucess, function_error);
350             }
351     }
352     class BackEvento {
353             static salvar(front_function_sucess, front_function_error) {
354                     // Dados gerais
355                     var nome = $("#nome").val();
356                     var horario = "";
357
358                     var horarioHora = $("#horario_hora").val()+"";
359                     if (horarioHora.length < 2) {
```

```
360                     horario += "0" +horarioHora;
361             } else {
362                     horario += horarioHora.substring(0,2);
363             }
364
365             var horarioMinuto = $("#horario_minuto").val() +"";
366             if (horarioMinuto.length < 2) {
367                     horario += "0" +horarioMinuto;
368             } else {
369                     horario += horarioMinuto.substring(0,2);
370             }
371
372             var segunda = $("#dia_segunda").prop("checked");
373             var terca = $("#dia_terca").prop("checked");
374             var quarta = $("#dia_quarta").prop("checked");
375             var quinta = $("#dia_quinta").prop("checked");
376             var sexta = $("#dia_sexta").prop("checked");
377             var sabado = $("#dia_sabado").prop("checked");
378             var domingo = $("#dia_domingo").prop("checked");
379
380             // Dados do som
381             var somTocar = $("#som_input_checkbox").prop("checked");
382             var somVolume = $("#som_input_volume").val();
383             var somTempo = $("#som_input_tempo").val();
384
385             // String dados
386             var dado = "nome=" + nome;
387             dado += "&horario=" + horario;
388             dado += "&segunda=" + (segunda? "1" : "0");
389             dado += "&terca=" + (terca? "1" : "0");
390             dado += "&quarta=" + (quarta? "1" : "0");
391             dado += "&quinta=" + (quinta? "1" : "0");
392             dado += "&sexta=" + (sexta? "1" : "0");
393             dado += "&sabado=" + (sabado? "1" : "0");
394             dado += "&domingo=" + (domingo? "1" : "0");
395
396             dado += "&somTocar=" + (somTocar? "1" : "0");
397             dado += "&somVolume=" + somVolume;
398             dado += "&somTempo=" + somTempo;
399
400             function function_sucess(data) {
401                     console.log("Sucess:");
402                     console.log(data);
403
404                     if (data) {
405                             front_function_sucess(data);
406                     } else {
407                             front_function_error(data);
408                     }
409             }
410             function function_error(data) {
411                     console.log(data);
412                     front_function_error(data);
413             }
```

```
414
415                     var operacaoASerRealizada = $($("main").get(0)).attr("data-operacao");
416                     var urlDaPagina = "insert";
417
418                     if (operacaoASerRealizada == "update") {
419                             urlDaPagina = "update";
420
421                             var id = $($("main").get(0)).attr("data-id");
422                             dado += "&id="+id;
423                     }
424
425                     Ajax("action/evento/"+urlDaPagina+".php", dado, function_sucess,
    function_error);
426             }
427     }
428
429     /* ---------------------- [FunctionPage] Evento ---------------------- */
430     function page_evento() {
431             $("#som_input_checkbox").change(function() {
432                     var checked = $(this).prop("checked");
433                     console.log(checked);
434
435                     if (checked) {
436                             FrontEvento.ativarSom();
437                     } else {
438                             FrontEvento.desativarSom();
439                     }
440             });
441             $("footer a.salvar").click(function() {
442                     FrontEvento.salvar($(this));
443             });
444     }
445
446     /* ---------------------- [Class] ListaDeEvento ---------------------- */
447     class FrontListaDeEvento {
448             static select() {
449                     console.log("oi");
450                     var arrayCheckbox = $("table input:checkbox:checked");
451                     console.log(arrayCheckbox);
452                     console.log($(arrayCheckbox));
453
454                     if ($(arrayCheckbox).length <= 0) {
455                             $("input.remove:button").attr("data-disabled", "disabled");
456                             $("input.insert:button").attr("data-disabled", "enabled");
457                     } else {
458                             $("input.remove:button").attr("data-disabled", "enabled");
459                             $("input.insert:button").attr("data-disabled", "disabled");
460                     }
461             }
462             static remove(arrayTr) {
463                     $("input:checkbox").prop("disabled", true);
464                     var dataId = "";
465
466                     for (var c = 0; c < $(arrayTr).length; c++) {
```

```
467                             var tr = $(arrayTr)[c];
468
469                             if ($(tr).find("input").prop("checked")) {
470                                     if (dataId != "") {
471                                             dataId += ",";
472                                     }
473                                     dataId += $(tr).attr("data-id");
474                             }
475                     }
476
477                     function function_sucess(data) {
478                             console.log("Sucess:");
479                             console.log(data);
480
481                             $("input:checkbox").prop("disabled", false);
482
483                             for (var c = 0; c < $(arrayTr).length; c++) {
484                                     var tr = $(arrayTr)[c];
485
486                                     if ($(tr).find("input").prop("checked")) {
487                                             $(tr).remove();
488                                     }
489                             }
490
491                             FrontListaDeEvento.select();
492                     }
493                     function function_error(data) {
494                             message_box(
495                                     message_box_type.FAIL,
496                                     "Algo inesperado ocorreu no servidor. Tente mais tarde."
497                             );
498
499                             $("input:checkbox").prop("disabled", false);
500                     }
501
502                     BackListaDeEvento.remove(dataId, function_sucess, function_error);
503             }
504             static insert() {
505                     var button = $("input#insert_event");
506                     $(window.document.location).attr("href", $(button).attr("data-href"));
507             }
508     }
509     class BackListaDeEvento {
510             static remove(dataId, front_function_sucess, front_function_error) {
511                     function function_sucess(data) {
512                             console.log("Sucess:");
513                             console.log(data);
514
515                             if (data) {
516                                     front_function_sucess(data);
517                             } else {
518                                     front_function_error(data);
519                             }
520                     }
```

```
521                     function function_error(data) {
522                             front_function_error(data);
523                     }
524
525                     Ajax("action/evento/remove.php", "id="+dataId, function_sucess,
    function_error);
526             }
527     }
528
529     /* ---------- [FunctionPage] Horarios ---------- */
530     function page_lista_de_evento() {
531             $("table input:checkbox").click(function() {
532                     console.log("active checkbox");
533                     var tr = $(this).parents("tr")[0];
534
535                     FrontListaDeEvento.select(tr);
536             });
537             $("input.insert:button").click(function() {
538                     FrontListaDeEvento.insert();
539             });
540             $("input.remove:button").click(function() {
541                     var arrayTr = $("table tr");
542                     FrontListaDeEvento.remove(arrayTr);
543             });
544     }
```