

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS  
GERAIS – CAMPUS BAMBUÍ  
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Fábio Oliveira Tempesta

**PROTÓTIPO DE UM SISTEMA DE MONITORAMENTO E  
CONTROLE DE BAIXO CUSTO BASEADO EM IOT PARA  
AMBIENTES AVIÁRIOS**

Bambuí - MG  
2021

FÁBIO OLIVEIRA TEMPESTA

**PROTÓTIPO DE UM SISTEMA DE MONITORAMENTO E  
CONTROLE DE BAIXO CUSTO BASEADO EM IOT PARA  
AMBIENTES AVIÁRIOS**

Trabalho de conclusão de curso apresentado  
ao Curso Bacharelado em Engenharia de Com-  
putação do Instituto Federal de Educação, Ci-  
ênci a e Tecnologia de Minas Gerais – Campus  
Bambuí para obtenção do grau de Bacharel  
em Engenharia de Computação.

Bambuí - MG  
2021

Fábio Oliveira Tempesta

## **PROTÓTIPO DE UM SISTEMA DE MONITORAMENTO E CONTROLE DE BAIXO CUSTO BASEADO EM IOT PARA AMBIENTES AVIÁRIOS**

Trabalho de conclusão de curso apresentado  
ao Curso Bacharelado em Engenharia de Com-  
putação do Instituto Federal de Educação, Ci-  
ênci a e Tecnologia de Minas Gerais – Campus  
Bambuí para obtenção do grau de Bacharel  
em Engenharia de Computação.

Aprovado em 28 de setembro de 2021 pela banca examinadora:

---

Prof. Me. Itagildo Edmar Garbazza

Orientador

Instituto Federal Minas Gerais – Campus Bambuí

Bambuí - MG  
2021

## RESUMO

A produção avícola de grande porte vem crescendo cada dia mais, usufruindo da maioria das tecnologias de ponta disponíveis atualmente, gerando . O projeto consiste na implementação de um sistema de monitoramento e controle de ambientes de criação de frangos, que seja de baixo custo e baseado em internet das coisas, com o propósito de auxiliar o pequeno avicultor elevar a produtividade do seu negócio. Para alcançar os resultados esperados, será utilizado um microcontrolador ESP32 para orquestrar todo o sistema. Ele ficará responsável por hospedar o servidor web que será a interface da aplicação com o usuário, podendo ser acessado por celulares, tablets e computadores, processar os dados de temperatura do ar, temperatura da água e umidade do ar, e controlar os acionadores de forma autônoma, ou por um comando solicitado via interface.

**Palavras-chave:** Sistemas Embarcados. Internet das Coisas. Produção Aviária. Micro-controladores.

## ABSTRACT

Large poultry production grows day by day, taking advantage of most of the latest technologies available today, which generates a greater advantage compared to small poultry producers. The project consists of the implementation of a low-cost monitoring and control system for chicken farming environments, based on the Internet of Things, with the purpose of helping small poultry farmers to increase the productivity of their business. To achieve the expected results, an ESP32 microcontroller will be used to orchestrate the entire system. It will be responsible for hosting the web server that will be the user interface of the application, which can be accessed from cell phones, tablets and computers, processing the data of air temperature, water temperature and air humidity, and controlling the triggers Consequently. by a command requested through the interface.

**Keywords:** Embedded Systems. Internet of Things. Poultry Production. Microcontrollers.

## SUMÁRIO

1	INTRODUÇÃO . . . . .	8
1.1	Objetivo geral . . . . .	8
1.2	Objetivos específicos . . . . .	8
1.3	Justificativa . . . . .	8
1.4	Resultados esperados . . . . .	9
2	REFERENCIAL TEÓRICO . . . . .	10
2.1	Criação de frangos de corte . . . . .	10
2.1.1	<i>Condições ideais</i> . . . . .	10
2.1.2	<i>Sistemas de climatização</i> . . . . .	11
2.2	Microcontroladores . . . . .	12
2.2.1	<i>ESP32</i> . . . . .	13
2.3	Protocolo de comunicação <i>One Wire</i> . . . . .	14
2.4	Sensores . . . . .	16
2.4.0.1	Sensor de Temperatura DS18B20 a prova d'água . . . . .	16
2.4.0.2	Sensor de Umidade e Temperatura DHT11 . . . . .	19
2.5	Atuadores . . . . .	20
2.5.1	<i>Válvula Solenoide</i> . . . . .	21
2.5.2	<i>Relé eletromecânico</i> . . . . .	22
2.6	<i>Internet das Coisas e Sistemas Embarcados</i> . . . . .	23
2.7	Estado-da-arte . . . . .	24
3	METODOLOGIA . . . . .	25
3.1	Classificação da pesquisa . . . . .	25
3.2	Solução . . . . .	25
3.2.1	<i>Atuadores</i> . . . . .	27
3.3	Materiais e Tecnologias . . . . .	27
3.3.1	<i>Ferramentas utilizadas</i> . . . . .	27
3.3.2	<i>Materiais do protótipo</i> . . . . .	28
3.3.2.1	ESP32 Devkit v1 - DOIT . . . . .	28
3.3.2.2	Válvula Solenoide para água 127 VAC . . . . .	29
3.3.2.3	Módulo Relé 5V 4 Canais . . . . .	30
3.4	Ambiente de experimento . . . . .	31
3.5	Métodos e Procedimentos . . . . .	32
3.5.1	<i>Metodologia de desenvolvimento</i> . . . . .	32
3.5.2	<i>Definição dos requisitos</i> . . . . .	33
3.5.3	<i>Procedimentos</i> . . . . .	35

4	DESENVOLVIMENTO . . . . .	36
4.1	Leitura do sensor DHT11 . . . . .	36
4.2	Leitura dos sensores DS18B20 . . . . .	37
4.3	Ventilador . . . . .	38
4.4	Nebulizador . . . . .	39
4.5	Trocador de água . . . . .	41
4.6	Interface <i>Web</i> . . . . .	44
4.7	Custo material do protótipo . . . . .	44
	 REFERÊNCIAS . . . . .	46
5	IMPLEMENTAÇÃO DO CÓDIGO DO MICROCONTROLADOR ESP32 . . . . .	50
6	IMPLEMENTAÇÃO DO CÓDIGO HTML DA PÁGINA WEB	67
7	IMPLEMENTAÇÃO DO CÓDIGO CSS DA PÁGINA WEB .	91
8	IMPLEMENTAÇÃO DO CÓDIGO JAVASCRIPT DA PÁGINA WEB . . . . .	95
9	CIRCUITO ELÉTRICO GERAL DO PROJETO . . . . .	102

## 1 INTRODUÇÃO

Devido ao crescimento constante da produção avícola, espera-se que o consumo mundial de carne de frango em 2022 ultrapasse a carne suína (CONNOLLY, 2018). No Brasil, a carne tem uma grande relevância econômica, como o relatório anual da ABPA (Associação Brasileira de Proteína Animal) (ABPA, 2021) mostra que o Brasil fechou o ano de 2021 sendo o maior exportador de carne de frango do mundo, com 4,231 milhões de toneladas exportadas, e o terceiro maior produtor, com 13,845 milhões de toneladas de carne produzidas.

Diante de um cenário tão propício ao país, deve-se estar sempre atento em adotar um sistema de criação voltado à uma produção eficiente e de qualidade para continuar liderando a exportação desse setor que afeta positivamente a economia. Para essa finalidade, Silva Braga *et al.* (2018) reconhece que, em um sistema de criação, o bem-estar de animais está diretamente relacionado à produtividade e qualidade do produto.

Para uma criação de frangos obter alta produtividade, é necessário ficar atento a todas as necessidades básicas do animal. No Brasil, o ambiente para produção e bem-estar das aves nem sempre é compatível com as necessidades fisiológicas das mesmas, como a temperatura do ambiente, umidade do ar e a temperatura da água que esses animais consomem (BARBOSA, 2013; SOUZA, P. de, 2005).

### 1.1 Objetivo geral

Oferecer um protótipo de sistema embarcado de monitoramento e controle de um ambiente de criação de frangos para auxiliar no desenvolvimento e bem-estar animal, utilizando o conceito Internet das Coisas e, que seja acessível ao pequeno avicultor.

### 1.2 Objetivos específicos

- Desenvolver o hardware dos sensores e demais circuitos necessários;
- Desenvolver e configurar software dos sensores;
- Implementar a interface *Web*;
- Implementar os mecanismos dos atuadores;
- Testar e avaliar o sistema no ambiente de experimento.

### 1.3 Justificativa

Utilizando a tecnologia a favor da produção e bem-estar das aves, pode-se destacar os sistemas de monitoramento de ambiente em tempo real com acionadores

automáticos (CONNOLLY, 2018). O avicultor que implementa um sistema desse porte pode esperar alguns benefícios, como:

- maior precisão para admitir a temperatura de conforto térmico dos animais comparado aos métodos tradicionais;
- homogeneidade térmica do ambiente, diminuindo grandes variações de temperatura (FONTES, 2020; CONNOLLY, 2018);
- aumento da qualidade de vida dos animais, de modo que induz uma melhora no bem-estar, no desempenho de corte e reduz a taxa de mortalidade dos frangos (SILVA BRAGA *et al.* 2018);
- menor mão de obra aos produtores, dado que o controle do ambiente é todo feito de forma automática, reduzindo sua carga de trabalho;
- economia de energia elétrica (FONTES, 2020; CONNOLLY, 2018);
- captura de dados para realizar análises que pode influenciar na futura tomada de decisão.

Em contrapartida, um sistema desse porte ainda gera um custo inicial relativamente alto para implantação, devido às suas inúmeras funções. O que há na literatura atual é o sistema desenvolvido na pesquisa de doutorado de Marcelo Eduardo de Oliveira *et al.* (2018). Fontes (2020) declara que esse sistema é considerado de baixo-custo e que sua implantação teria um preço em torno de R\$ 3000,00.

O sistema desenvolvido há chances de suas funcionalidades gerar um produto inovador, pois o controle de aviários convencionais utilizando IoT com as funcionalidades presentes no trabalho ainda é pouco explorado na literatura.

#### 1.4 Resultados esperados

Espera-se que o protótipo inicial seja capaz de ler as temperaturas e a umidade do ar corretamente, podendo informar esses valores ao usuário via interface *Web* e também fazer um controle eficiente do ambiente. Com o protótipo funcionando sem erros, espera-se que ele esteja apto a realizar os primeiros testes em um ambiente real.

## 2 REFERENCIAL TEÓRICO

Este capítulo descreve os principais conceitos relacionados ao trabalho.

### 2.1 Criação de frangos de corte

A criação de frangos de corte vem crescendo cada vez mais nos últimos anos, tanto no contexto da economia, quanto no do tecnológico. Valéria Maria Nascimento Abreu e Paulo Giovanni de Abreu (2011) salienta que a avicultura de corte tem investido em inovações tecnológicas há pelo menos 20 anos, tendo como busca encontrar implementações de sistemas com maior eficiência na produção.

#### 2.1.1 *Condições ideais*

Os ambientes aviários que apresentam condições incompatíveis com as necessidades fisiológicas das aves, além de manter os animais em condições precárias, sofrem perdas de produtividade, pois, segundo Silva Braga *et al.* (2018), o bem-estar de animais está diretamente relacionado à qualidade e produtividade em um sistema de criação. Hötzell e Machado Filho (2004) acrescenta outros motivos que levam as pessoas a se preocuparem com o bem-estar de animais: inquietações de origem ética; o potencial efeito que o bem-estar animal possa ter na qualidade dos alimentos; as conexões entre bem-estar animal e comercialização internacional de seus produtos de origem animal.

Para gerar o apropriado aumento da produtividade da criação, estudos foram feitos em relação às necessidades básicas desses animais para um bom desenvolvimento e o que andam utilizando para alcançar o tal objetivo. Fontes (2020) esclarece a importância de uma temperatura adequada para as aves no seu ambiente de criação:

Quando a temperatura ambiente sobe acima da zona de conforto térmico, o animal começa a buscar alternativas para perder calor realizando alguns ajustes funcionais rápidos, como a vasodilatação e o aumento da frequência respiratória. No entanto, se a temperatura continuar aumentando, a ave reduz seu metabolismo, aumenta o consumo de água e tenta maximizar a perda de calor pelo suor. Se mesmo assim a temperatura não parar de subir, essas ações deixam de surtir efeito e, sem opções para aumentar a perda de calor, o animal pode morrer de hipertermia. Temperaturas abaixo do indicado também são perigosas. Num primeiro momento, em resposta ao frio, o animal busca o Sol, lugares secos, pouca ventilação, piso aquecido e reduz o consumo de água. Porém, se a temperatura ambiente seguir caindo, a ave aumenta o seu metabolismo realizando atividades de maior movimento e consumindo mais ração. Caso a temperatura siga despencando, o animal perde a capacidade de produzir calor e sua temperatura corporal começa a baixar rapidamente, podendo levá-lo à morte por hipotermia.

Como a água é uma possibilidade que esses animais têm para reduzir a temperatura corporal, deve-se ficar atento na temperatura da água dos bebedouros. Apesar desse parâmetro ser um pouco esquecido pelos ambientes de criação de aves, Barbosa (2013) declara que, devido ao fato de as aves não possuírem glândulas sudoríparas para uma maior perda de calor pelo suor, o consumo de água fria é a alternativa que esses animais têm para diminuir a temperatura corporal em casos de estresse calórico e, por consequência, uma melhora no desempenho.

Vale ressaltar que a temperatura de conforto dos frangos de corte varia de acordo com a idade do animal. Uma pesquisa (CASSUCE *et al.* 2013) que tinha a finalidade de encontrar os valores semanais da temperatura ideal para as aves obteve as temperaturas de 31,3 °C; 25,5 °C e 21,8 °C respectivamente, na primeira, segunda e terceira semana de vida em diante para um melhor desenvolvimento dos animais.

A umidade do ambiente também deve ser controlada, um valor muito alto pode não ser a melhor condição para o galpão de criação. A capacidade das aves em suportar o calor é inversamente proporcional ao teor de umidade relativa do ar, devido a umidade alta apresentar dificuldades aos frangos remover calor interno pelas vias aéreas, o que leva ao aumento da frequência respiratória (OLIVEIRA, R. F. M. d. *et al.* 2006). A Tabela 1 mostra a umidade do ar e temperatura ambiente em função da idade das aves.

Tabela 1 – Valores ideais de temperatura ambiente e de umidade do ar, em função da idade das aves

Idade (Semanas)	Temperatura Ambiente (°C)	Umidade do ar (%)
1	32 – 35	60 – 70
2	29 – 32	60 – 70
3	26 – 29	60 – 70
4	23 – 26	60 – 70
5	20 – 23	60 – 70
6	20	60 – 70
7	20	60 – 70

Fonte: Valéria Maria Nascimento Abreu e Paulo Giovanni de Abreu (2011).

### 2.1.2 Sistemas de climatização

Para manter proporcionar condições eficientes em termos de conforto térmico, próximas à zona termoneutra da ave, que é quando a dissipação da temperatura corporal é mínimo, a adoção de sistemas de ventilação com sistemas de resfriamento do ar é um bom caminho a se seguir (SOUZA, F. d., 2017).

Para combater as elevadas temperaturas e a umidade baixa do ambiente, será utilizado uma técnica de modificação ambiental artificial chamada resfriamento evaporativo do ar. De acordo com Sartor *et al.* (2001), esta técnica consiste em incorporar vapor d'água

diretamente no ar, utilizando um sistema de ventilação e nebulização, causando mudança no ponto de estado do ambiente (aumento da umidade e reduzindo a temperatura). Vale ressaltar que o nebulizador deve ser acionado apenas quando a umidade estiver relativamente baixa, pois uma umidade muito alta pode chegar a atrapalhar as aves remover o calor interno pela vias aéreas. Apesar de existirem técnicas mais novas para o resfriamento adiabático evaporativo do ar, que é quando há contato entre o ar e a água, os pequenos galpões continuam utilizando nebulizadores, pois teria um alto custo para mudar o galpão para incorporação da água por paredes porosas ou placas umedecidas, que são utilizados em aviários modernos.

A ventilação também deve ser controlada, um controle de forma inadequada pode, em vez de trazer benefícios ao ambiente, prejudicar as aves. Uma boa ventilação para o movimento de ar fresco no galpão, é um mecanismo de uso contínuo e diário que é essencial na produção de aves para um melhor crescimento e conforto, devido à ele remover o excesso de umidade do aviário e oferecer ar fresco aos animais, mas, por outro lado, uma ventilação inadequada pode provocar estresse, permitindo calor excessivo e acúmulo de umidade no galpão (GUERRA, 2021).

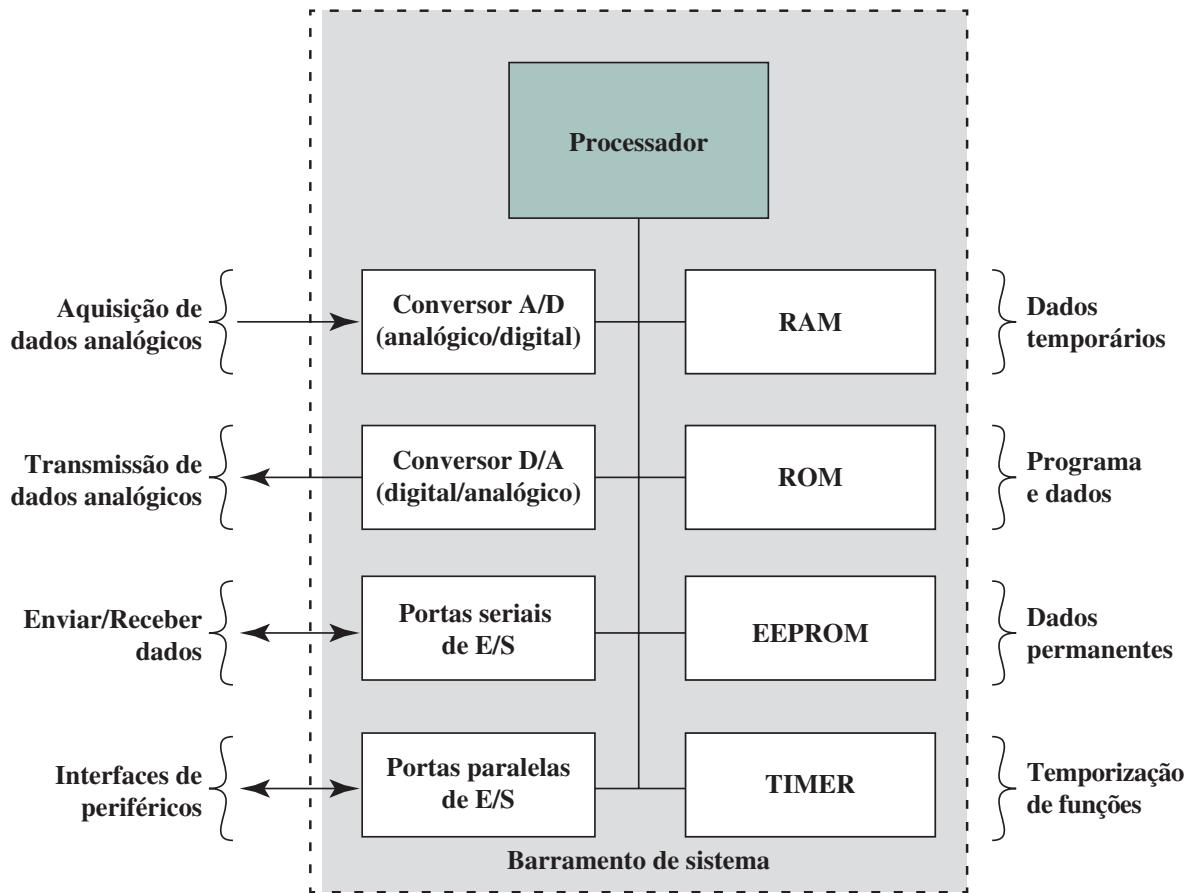
## 2.2 Microcontroladores

Vidal Pereira da Silva Júnior (2013) declara que utiliza o termo "Microcomputador-de-um-só-chip" para definir os microcontroladores desde o seu primeiro livro. O autor acrescenta que a principal característica do dispositivo está em reunir, em um único chip, todos os periféricos necessários para o projeto e fabricação de dispositivos eletrônicos dos mais diversos tipos, desde simples sinalizadores e luzes pisca-pisca até equipamentos médicos sofisticados. Esses dispositivos estão cada vez sendo mais utilizados, uma vez que o mercado atual estão progressivamente demandando sistemas embarcados, que são sistemas computacionais compactos e de custo acessível que atendem a uma demanda específica (CARDOSO, 2020).

É de referir que existe diferença entre microcontroladores e microprocessadores, cada um deles possui uma aplicação diferente, como poder de processamento demandado, custo, etc. Enquanto um microcontrolador é um chip simples que contém processador, memória não volátil para o programa, memória volátil para entrada e saída, um *clock* e uma unidade de controle de entrada e saída, o microprocessador contém apenas o processador que ocupa uma área de silício muito maior que o microcontrolador e possui uma eficiência de energia muito menor (STALLINGS, 2017). Ao observar a Figura 1, pode-se encontrar os elementos apontados em um chip microcontrolador típico.

Uma pesquisa recente de Rios *et al.* (2020) tinha a finalidade de comparar o desempenho dos principais microcontroladores do mercado para aplicações em *Internet das Coisas*, dentre eles, ESP8266, ESP32 e MK1000, e, mesmo utilizando apenas um núcleo do processador, o ESP32 demonstrou resultados melhores, comparado aos demais.

Figura 1 – Elementos de um chip microcontrolador típico.



Fonte: Stallings (2017)

Outra das vantagens da plataforma é o módulo *Wi-Fi* integrado, dessa maneira, gerando uma redução do custo do produto final, já que não será necessário comprar um módulo separado.

### 2.2.1 *ESP32*

O ESP32 é um microcontrolador criado pela Espressif<sup>1</sup>, que é uma empresa multinacional de semicondutores, e é denominado como irmão mais novo do ESP8266. Muito utilizado para aplicações em *Internet das Coisas* devido ao seu módulo *Wi-Fi* integrado, um custo relativamente baixo e a adoção da IDE Arduino<sup>2</sup> como plataforma de desenvolvimento de aplicações para o microcontrolador, tendo uma comunidade ampla ao seu lado e boas bibliotecas que são compatíveis ao dispositivo.

Como o ESP32 é um chip contendo várias características dentro dele, de modo geral é recomendado ter uma placa de desenvolvimento, que basicamente aloca o chip ESP32 e deixa praticamente pronto para implementação, como as portas especificadas, entrada micro-usb para carregamento de código e funções adicionais dependendo do

<sup>1</sup> <https://www.espressif.com/>

<sup>2</sup> <https://www.arduino.cc/en/software>

modelo. Além disso, deve ficar claro que existe vários tipos de ESP32, variando o tamanho, processador (número de núcleos e a frequência de clock), memória flash, protocolo *Wi-Fi*, número e tipos de pinos, e a faixa de temperatura de funcionamento.

O modelo do dispositivo adotado no trabalho foi o ESP32-WROOM-32, que, de acordo com o *datasheet* da Espressif (2021), tem as seguintes características:

- CPU: Xtensa® Dual-Core 32-bit LX6, clock de 80 a 240 MHz;
- memória ROM: 448 KBytes;
- memória RAM: 520 Kbytes;
- memória Flash: 4 MB;
- wireless padrão 802.11 b/g/n;
- conexão *Wi-Fi* 2,4 GHz a 2,5 GHz (máximo de 150 Mbps de transferência);
- *bluetooth* BLE 4.2;
- corrente de consumo médio de 80mA;
- tensão de operação de 3 V a 3,6 V;
- faixa de temperatura de trabalho recomendada: -40°C a 85°C.

As portas GPIO (General Purpose Input/Output - Entrada/Saída de Uso Geral), que são, basicamente, os pinos de entrada e saída, podem ter funções diferentes, como conversor de analógico para digital de 12 bits, conversor digital para analógico de 8 bits e sensor capacitivo. A Figura 2 mostra o esquemático dos pinos no chip.

### 2.3 Protocolo de comunicação *One Wire*

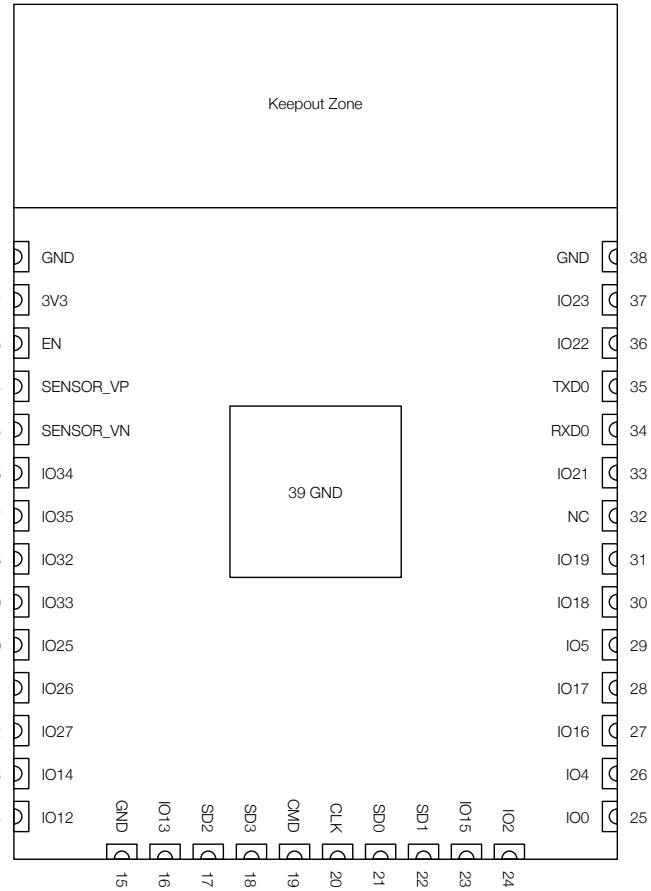
O protocolo de comunicação *1-Wire* (ou *One-Wire* - Um-Fio em português) foi desenvolvido pela empresa Dallas Semiconductor e posteriormente foi adquirida pela Maxim Integrated<sup>3</sup> (SACCO, 2015). A comunicação desse protocolo é do tipo mestre-escravo e é operada acontece em apenas uma linha de dados e um terra de referência (Figura 3). Um mestre inicia e controla a comunicação com um ou mais dispositivos escravos no barramento *One-Wire*(1-WIRE..., 2012).

Cada escravo do protocolo possui um número de identificação (ID) de 64 bits exclusivo, inalterável e programado de fábrica, que serve como endereço do dispositivo no barramento de 1 fio (PROGRAMMABLE..., 2019). Ainda de acordo com o autor, os escravos, frequentemente, operam nas seguintes faixas de tensão:

---

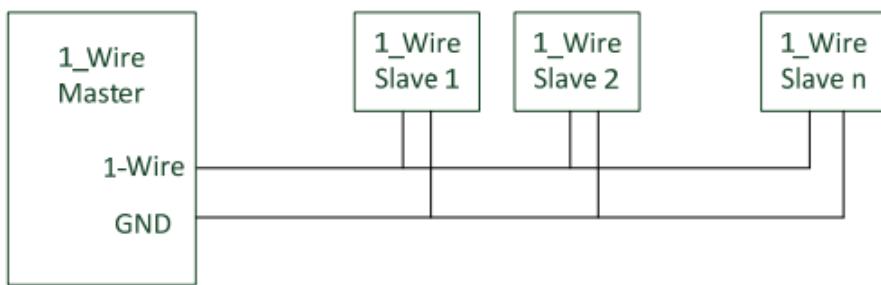
<sup>3</sup> <https://www.maximintegrated.com>

Figura 2 – Esquema dos pinos no chip ESP32-WROOM-32 (vista do topo).



Fonte: Espressif (2021)

Figura 3 – Esquema básico da rede *1-Wire*.



Fonte: Maceková (2012)

- 1.71V (mínima) até 1.89V (máxima);
- 1.71V (mínima) até 3.63V (máxima);
- 2.97V (mínima) até 3.63V (máxima);
- 2.8V (mínima) até 5.25V (máxima).

De acordo com a fabricante 1-WIRE... (2012), os dispositivos que utilizam o protocolo *1-Wire* são projetados para uso em um ambiente em que a comunicação

seja momentânea. Isso significa que, quando um dispositivo escravo ser desconectado do barramento *1-Wire*, ele poderá retornar a comunicação com o mestre facilmente.

Outra vantagem que o protocolo apresenta, é o seu modo *Parasitic Power*. Os dispositivos *1-Wire* que possui essa tecnologia podem funcionar independente de uma alimentação, de modo que eles conseguem "roubar" uma energia necessária de funcionamento pela própria linha de comunicação. Saduikis (2010) declara que, quando for utilizar esse tipo configuração, a linha de dados *1-Wire* deve apresentar uma tensão com o mesmo valor que o dispositivo precisa para se alimentar.

## 2.4 Sensores

Os sensores são dispositivos que são sensíveis a alguma forma de energia (luminosa, térmica ou cinética), e tem o objetivo de relacionar informações sobre uma grandeza que precisa ser medida (THOMAZINI; ALBUQUERQUE, 2020). No sistema embarcado, eles terão a finalidade de determinar as condições que precisam ser controladas para uma produção eficiente.

Deve-se selecionar os sensores de acordo com as necessidades da aplicação a que ele será submetido, que se mal avaliados podem levar a resultados inesperados e até mesmo desastrosos num projeto (BRAGA, 2012). Apesar do projeto necessitar de um cuidado especial das temperaturas e umidade, não é necessário sensores com precisões muito altas, pois existe uma faixa de tolerância aceitável para manter o ambiente nas condições ideais.

### 2.4.0.1 Sensor de Temperatura DS18B20 a prova d'água

O dispositivo apresenta um cabo de 1 metro de comprimento, uma ponta de aço inoxidável com dimensão de 6 mm x 50 mm (Figura 4), e, localizado dentro da proteção, o sensor de temperatura digital DS18B20. Segundo as especificações do fabricante PROGRAMMABLE... (2019), o sensor DS18B20 apresenta as seguintes características:

- tensão de operação de 3 V a 5,5 V;
- mede valores de temperatura de -55°C a 85°C;
- $\pm 0,5^\circ\text{C}$  de precisão na medição;
- flag de alarme para o microcontrolador, caso a temperatura saia da faixa configurada;
- método "*parasite power*"(possibilidade de alimentação pelo mesmo pino que são transferido os dados, caso seja feita a configuração correta);
- medição de temperatura de 9 a 12 bits (quanto mais bits, maior variação da precisão de medição e maior tempo de conversão);

- protocolo de comunicação *One Wire*.

Figura 4 – Sensor DS18B20 com a proteção a prova d’água.



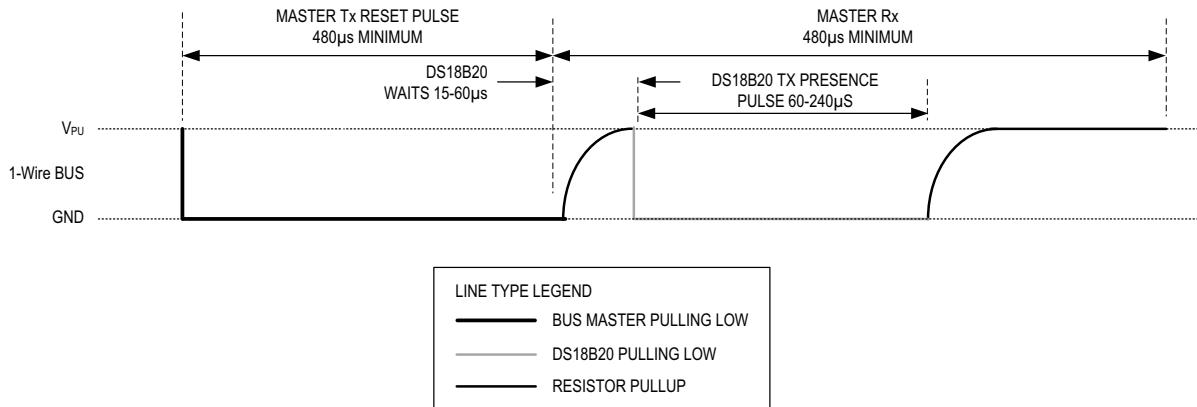
Fonte: O autor, 2021.

Ainda de acordo com *datasheet*, a comunicação com o sensor é feita através do protocolo *1-Wire*, podendo ser necessário a integração de um resistor de *pull-up* na linha de comunicação para uma maior confiança da troca de dados. O manual define uma sequência para comunicar com o sensor conforme os passos relatados:

- 1. Inicialização:** passo que tem a finalidade de inicializar a comunicação entre o sensor e o microcontrolador, consiste em primeiramente o microcontrolador mandar um pulso de *reset* para o sensor, colocando o pino de dados em nível lógico baixo por, ao menos,  $480 \mu s$ . Após isso, o microcontrolador libera a linha, deixando ela voltar para nível lógico alto novamente e, em seguida, fica aguardando (entre 15 a  $60 \mu s$ ) o sensor mandar um pulso de presença de  $60 \mu s$  a  $240 \mu s$ . A Figura 5 apresenta este passo por meio de gráficos para um melhor entendimento.
- 2. Comando ROM (*Read-Only Memory* - Memória Somente de Leitura):** segunda fase de comunicação que é utilizada para endereçar quais sensores da rede *One Wire* serão acessados. No *datasheet* mostra todos os comandos individualmente que pode ser utilizado nesta fase, por exemplo, o *Skip ROM* para utilizar todos sensores sem ter que selecioná-los.
- 3. Comando de Função:** fase que pode iniciar uma nova medição de temperatura, escrever uma nova configuração na memória do sensor, ler temperatura presente na memória, entre outras. Esses comandos são feitos de acordo com o código, geralmente em hexadecimal, informado no *datasheet*.

A Figura 6 mostra que o sensor possui uma memória RAM (*Random Access Memory* - Memória de Acesso Aleatório) interna de 9 registradores, cada um com a capacidade de 1 *byte*: os bytes 0 e 1 guarda valores da temperatura lida, o valor inicial do

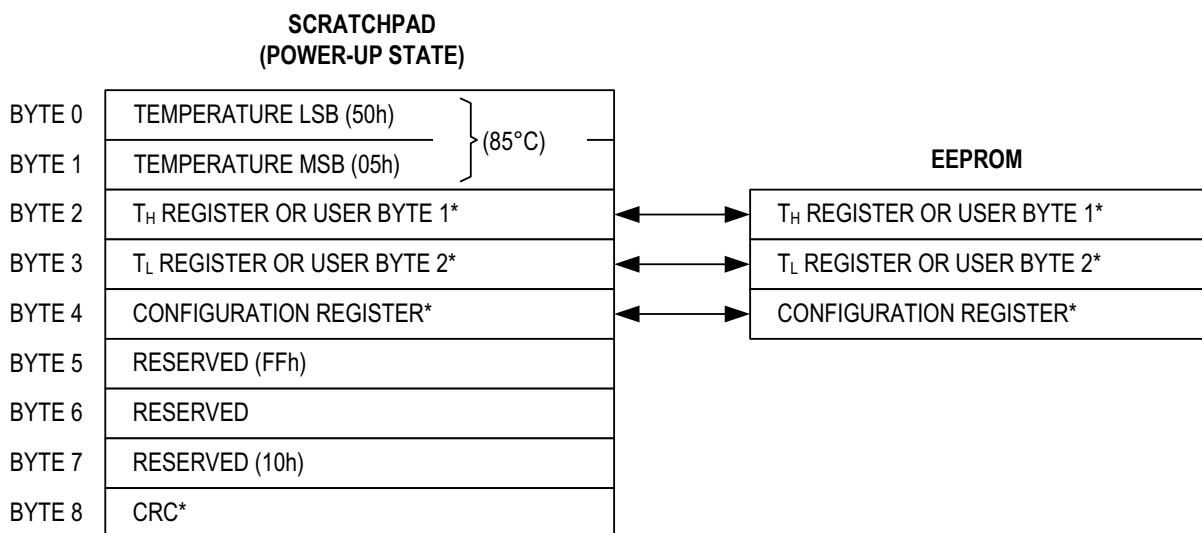
Figura 5 – Tempo de inicialização da comunicação.



Fonte: PROGRAMMABLE... (2019)

campo é de 85°C por padrão; os *bytes* 2 e 3 guarda os valores máximo e mínimo aceitável para o alarme não disparar; o byte 4 é responsável por guardar a resolução de temperatura de 9 a 12 bits; do *byte* 5 ao *byte* 7 é reservado para o sensor; o último byte contém o CRC (*Cyclic Redundancy Check* - Verificação de Redundância Cíclica), que é responsável por checar se houve algum erro na transmissão dos dados digitais por meio de bits de soma de verificação, de todos os registradores. A memória EEPROM (*Electrically Erasable Programmable Read-Only Memory* - Memória Somente de Leitura Programável Apagável) é responsável por salvar as configurações determinadas até quando o sensor estiver desligado, visto que a memória RAM não faz isso por ser uma memória temporária.

Figura 6 – Memória interna do sensor DS18B20.



\*POWER-UP STATE DEPENDS ON VALUE(S) STORED IN EEPROM.

Fonte: PROGRAMMABLE... (2019)

A Figura 7 mostra o formato da leitura de temperatura, em °C, que já vem calibrado de fábrica, dos registradores do *byte* 0 e 1. O primeiro conjunto de 8 bits ("LS

BYTE") é formado pelos bits menos significantes do valor da temperatura, de forma em que do bit 2 ao 0 pode ou não ser utilizado, dependendo da configuração utilizada no sensor. O segundo é formado pelos bits mais significantes, sendo que os bits de 15 a 11 são para determinar o sinal do valor (0 para positivo e 1 para negativo).

Figura 7 – Formato da temperatura registrada na memória.

	<b>BIT 7</b>	<b>BIT 6</b>	<b>BIT 5</b>	<b>BIT 4</b>	<b>BIT 3</b>	<b>BIT 2</b>	<b>BIT 1</b>	<b>BIT 0</b>
<b>LS BYTE</b>	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
	<b>BIT 15</b>	<b>BIT 14</b>	<b>BIT 13</b>	<b>BIT 12</b>	<b>BIT 11</b>	<b>BIT 10</b>	<b>BIT 9</b>	<b>BIT 8</b>
<b>MS BYTE</b>	S	S	S	S	S	$2^6$	$2^5$	$2^4$

S = SIGN

Fonte: PROGRAMMABLE... (2019)

#### 2.4.0.2 Sensor de Umidade e Temperatura DHT11

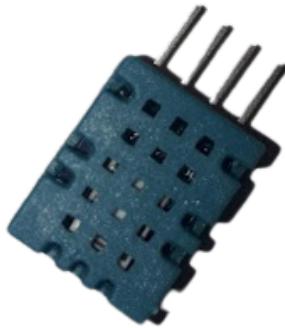
Dispositivo responsável por medir a temperatura e a umidade relativa do ar no ambiente aviário (Figura 8). O sensor digital é composto por um termistor do tipo NTC para temperatura e um sensor resistivo do tipo HR202 para umidade. Pode-se destacar as principais características do DHT11, de acordo com DHT11... (2019):

- dimensões: (23 x 12 x 5) mm;
- capacidade de medir umidade relativa na faixa de 20% a 90%;
- capacidade de medir temperatura na faixa de 0 °C a 50 °C;
- corrente de operação de 200  $\mu$ A a 500 mA;
- $\pm 5,0\%$  de precisão na medição da umidade;
- $\pm 2,0^\circ\text{C}$  de precisão na medição da temperatura;
- tensão de operação de 3 V a 5,5 V.

O presente dispositivo também utiliza comunicação *One Wire* para comunicar com o microcontrolador, como o sensor DS18B20. Pode ser necessário o uso de resistor *pull-up* na linha de comunicação para uma transferência de dados mais confiável. Por ser um sensor digital, ele manda os valores de temperatura e umidade por meio de bits, sendo 2 bytes para cada uma das grandezas medidas, um byte mais significativo e o outro menos significativo, e um byte para soma de verificação (DHT11..., 2019). A Figura 9 mostra o formato da temperatura, umidade e o byte de soma de verificação que deve informar, em decimal, quantos bits (dos 32) foram setados em 1.

O sensor DHT11 começa sua comunicação depois que o microcontrolador envia um sinal de nível lógico baixo através do pino de dados, por pelo menos 18 ms, fazendo

Figura 8 – Sensor DHT11.



Fonte: O autor, 2021.

Figura 9 – Formato da mensagem de comunicação do sensor DHT11.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
umidade (mais significativo)	BYTE 1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	+	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
umidade (menos significativo)	BYTE 2	2^{(-8)}	2^{(-7)}	2^{(-6)}	2^{(-5)}	2^{(-4)}	2^{(-3)}	2^{(-2)}	2^{(-1)}
	+	BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
temperatura (mais significativo)	BYTE 3	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	+	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
temperatura (menos significativo)	BYTE 4	2^{(-8)}	2^{(-7)}	2^{(-6)}	2^{(-5)}	2^{(-4)}	2^{(-3)}	2^{(-2)}	2^{(-1)}
	+	BIT 39	BIT 38	BIT 37	BIT 36	BIT 35	BIT 34	BIT 33	BIT 32
soma de verificação	BYTE 5	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Fonte: Adaptado de DHT11... (2019)

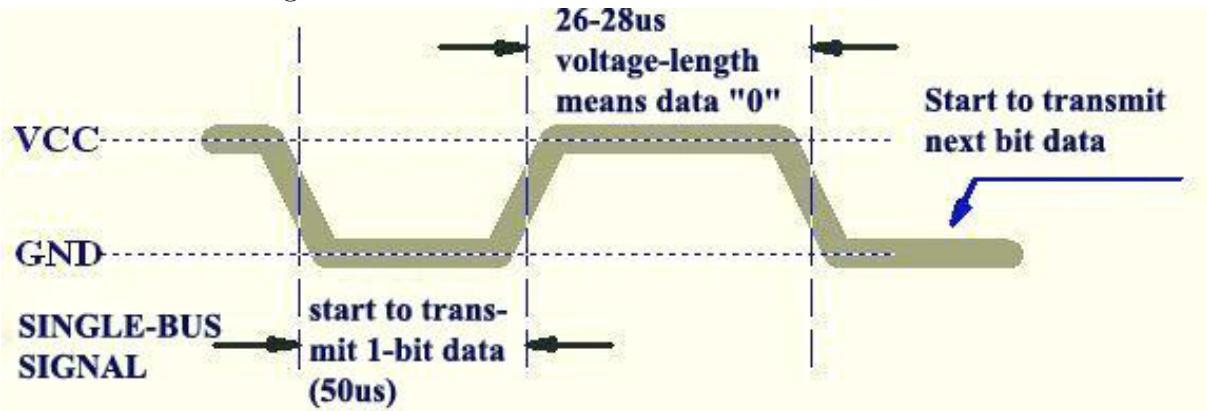
que o dispositivo saia do modo de baixo consumo de energia para o modo de operação e espere que o microcontrolador conclua o sinal. Logo em seguida, depois de  $20 \mu\text{s}$  a  $40 \mu\text{s}$ , o DHT11 envia um sinal de resposta de dados de 40 bits (os 5 bytes mostrados na Figura 9 em sequência), ficando em modo de baixo consumo de energia novamente até o microcontrolador chamá-lo novamente.

Antes de mandar os 40 bits, o sensor começa enviando um sinal em nível lógico baixo de  $80 \mu\text{s}$ , e, logo em seguida, deixa a linha de dados por  $80 \mu\text{s}$  no nível lógico alto. Após mandar esse sinal de resposta, ele começa mandar os bits pelo barramento, de forma que todos os bits de dados começam com o nível lógico baixo por  $50 \mu\text{s}$  e vai para o nível lógico alto. O que determina se o bit é "0" ou "1" é o tempo em que a linha ficou no nível lógico alto após sair do baixo, de forma que se for "0", o tempo é de  $26\text{-}28 \mu\text{s}$  (Figura 10), e quando o tempo for de  $50 \mu\text{s}$ , foi indicado o bit "1" (Figura 11). As Figuras 10 e 11 mostram, por meio de gráficos, o momento em que um determinado bit é caracterizado.

## 2.5 Atuadores

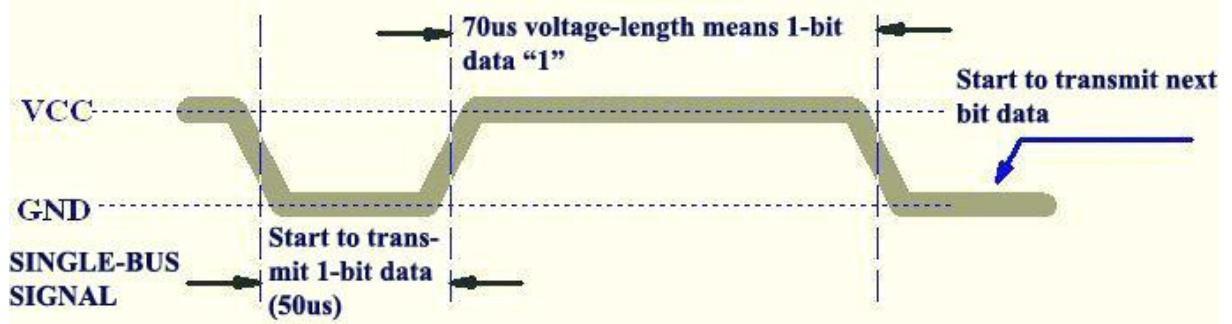
No âmbito da automação de sistemas industriais, Thomazini e Albuquerque (2020) argumenta que os atuadores são dispositivos que modificam uma variável controlada. Essa modificação ocorre depois de receber um sinal de comando do controlador, uma mudança normalmente mecânica, como uma alteração de posição ou velocidade (GROOVER,

Figura 10 – DHT11 indicando o bit com valor de 0.



Fonte: DHT11... (2019)

Figura 11 – DHT11 indicando o bit com valor de 1.



Fonte: DHT11... (2019)

2011). Pode-se citar, como exemplo de atuadores (pneumáticas, hidráulicas), as válvulas (estáticos, eletromecânicos), relés (estáticos, eletromecânicos), motores (*step-motor*, *syncro*, servomotor) e solenoides.

Groover (2011) classifica o tipo de amplificador utilizado pelos atuadores, em sua maioria, em três categorias:

- atuador elétrico: o mais comum das categorias, pode ser tanto linear (deslocamento linear de saída) como rotativo (deslocamento angular de saída);
- atuador hidráulico: utiliza fluido hidráulico para amplificar o sinal de comando do controlador, podendo ser linear ou rotativo;
- atuador pneumático: utiliza ar comprimido como energia propulsora, podendo oferecer movimento rotativo ou linear, como as outras categorias;

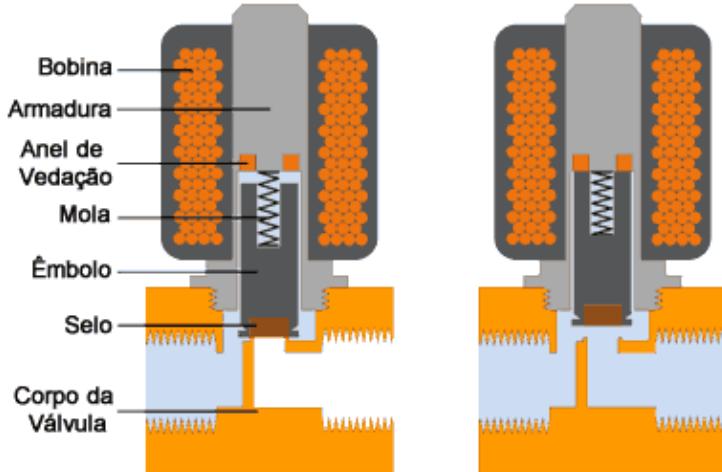
### 2.5.1 Válvula Solenoide

A válvula solenoide é um dispositivo eletromecânico que tem a finalidade de controlar o fluxo de fluidos. De modo geral, seu funcionamento abre ou fecha a passagem do fluido quando ela é energizada, baseado no deslocamento causado pela ação de um campo

magnético gerado por uma bobina elétrica com um núcleo ferromagnético móvel, que também é chamado de êmbolo (SILVA; LAGO, 2002; SILVEIRA, C. B., 2017). Quando o campo magnético é criado pela corrente elétrica, ela exerce uma força no êmbolo que por sua vez é puxado em direção ao centro da bobina.

Nos dispositivos do tipo normalmente fechado, quando estão em posição de repouso, o êmbolo tampa orifício por onde o fluido consegue circular, e só é liberado quando a válvula é energizada. No tipo normalmente aberto é o mesmo princípio, mas o êmbolo deixa o fluido circular normalmente e, quando uma corrente elétrica circula através da bobina, ele impede a circulação do fluido. Em questões de vias, a válvula mais simples existente é a do tipo 2/2, em que ela apresenta duas portas (uma de entrada e outra de saída) e duas posições possíveis (aberta ou fechada) (SILVEIRA, C. B., 2017). A Figura 12 apresenta o esquema de uma válvula normalmente fechada de 2/2 vias.

Figura 12 – Esquema típico de uma válvula solenoide.



Fonte: Cristiano Bertulucci Silveira (2017)

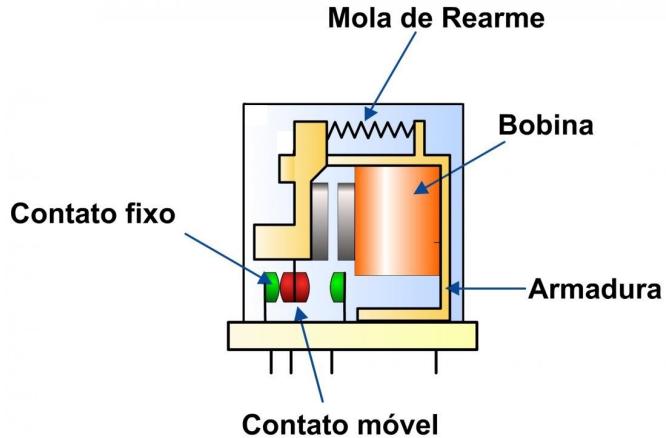
### **2.5.2 Relé eletromecânico**

O relé eletromecânico tem o princípio de fazer uma operação em um contato utilizando uma força mecânica em resposta a um estímulo (RUSH, 2011). Esse estímulo basicamente é dado pelo sinal de comando vindo do controlador. Conforme Stroski (2018), em suma, a força mecânica do atuador opera o contato por um eletroímã que recebe corrente elétrica e produz um campo magnético, atrairindo um pedaço metálico ligado ao contato móvel.

RUSH (2011) declara que, apesar dos relés eletromecânicos poderem ser classificados em diferentes tipos, somente o relé de atracamento (contato móvel) tem sido significativamente utilizado atualmente. Ainda conforme o autor, a movimentação da armadura que causa abertura ou fechamento do contato, com ela transportando um contato móvel que engata em um contato fixo. A Figura 13 mostra a organização dos componentes presentes internamente em um relé de atracamento típico. A mola de rearne é responsável

por restaurar o contato móvel para sua posição original no momento em que a bobina parar de ser alimentada.

Figura 13 – Relé típico de atracamento.



Fonte: Stroski (2018)

## 2.6 Internet das Coisas e Sistemas Embarcados

De acordo com Gogoni (2019), o conceito de *Internet das Coisas* (IoT - *Internet of Things*) se refere a uma enorme rede de dispositivos, que são além dos habituais, conectadas. O advento da IoT acabou gerando o aumento da comunicação entre máquinas pela *Internet*, o desenvolvimento de diversos utensílios, além de microdispositivos, como sensores que tornam-se partes integrantes da *Internet* (MAGRANI, 2018). Segundo Silva Braga *et al.* (2018), essa ferramenta se tornou uma ferramenta valiosa, à medida que tem proporcionado benefícios, na saúde e economia, desde as interações mais básicas do cotidiano, até aos avanços mais complexos. O fato é que essa tecnologia tende a crescer ainda mais com o passar dos tempos em consequência dos dispositivos eletrônicos reduzirem o preço e o tamanho.

Sinclair (2018) incentiva os negócios adotarem a tecnologia IoT em seus produtos, justificando os possíveis benefícios a se ganhar:

As empresas que não entram no mercado na hora certa, com uma oferta de IoT, enfrentarão fortes ventos contrários, e os retardatários não serão capazes de alcançar os precursores, tornando suas ofertas obsoletas. Há mais aqui, porém, do que uma simples corrida a pé. A IoT transforma a maneira como as empresas competem, e, no processo, muda as condições do campo de jogo, deixando-o de maneira que não é óbvia hoje. A IoT pode alterar as fronteiras da competição, eliminar os intermediários fracos, absorver categorias de produtos convencionais e transformar os modelos de negócios em si no mais importante de todos os atributos. Parece improvável? Pense na diferença em inteligência de mercado entre o Uber e as empresas convencionais de táxis. Talvez dez vezes, cem

vezes? Essa é a alavancagem decorrente de partir na frente com um produto IoT.

## 2.7 Estado-da-arte

No artigo de Alecrim, Campos e Tadayuki Yanagi Júnior (2013), foi desenvolvido um sistema para controle e supervisão do ambiente térmico para aviários, conseguindo um bom controle de temperatura para os animais. Apesar de ser uma fonte de embasamento, o sistema desse projeto não é conectado à rede.

Um projeto relacionado a essa área é a tese de doutorado em Ciências da Engenharia Ambiental de Marcelo Eduardo de Oliveira *et al.* (2018) e Marcelo Eduardo de Oliveira (2019), que consistiu em desenvolver todo o sistema com vários sensores de temperatura e umidade no ambiente de criação de frangos para fazer o controle automático ou remotamente. O trabalho foi interessante, mas com um sistema embarcado de custo um pouco elevado comparado ao protótipo presente desenvolvido. Isso se deu ao alto número de sensores, atuadores e dispositivos de alto processamento, como o Raspberry Pi 3 Model B+. O sistema também não monitorava a temperatura da água que os animais consomem, que é uma variável relevante à vista disso.

Na monografia de Lima Junior (2017), foi implementado um sistema conectado à *Internet* com o objetivo de alcançar um baixo custo e menor complexidade. Foi utilizado um sensor DHT11 para captar a temperatura e umidade do ambiente, e a possibilidade de acionar ventiladores e lâmpadas. Seus controladores consistiam em um Rasberry Pi 3 Model B e um Arduino Uno Rev3, sendo, respectivamente, um para processar e hospedar a interface Web, e o outro para processar as informações dos sensores e controlar os atuadores. Um interessante trabalho para época, visto que a tecnologia IoT era ainda mais nova que atualmente. Visto que um dos objetivos do trabalho era alcançar um baixo preço, observa-se que essa ideia pode ser aperfeiçoada, de modo que pode-se adotar apenas um microcontrolador, como o ESP32.

O trabalho de Ramadiani *et al.* (2021) desenvolveu um sistema que monitora a temperatura e umidade pelo sensor DHT11, utilizando o controlador Arduino ATMega328. O controle era feito quando a temperatura tornasse alta, uma bomba d'água acionava o nebulizador no ambiente. Teve um baixo custo de desenvolvimento, ajudou os produtores a manter a temperatura do ambiente aviário mais estável. A desvantagem é dada pela interface ser feita por meio de um LCD, e o controle ser feito por apenas nebulizadores, faltando renovação do ar.

### **3 METODOLOGIA**

Este capítulo tem a finalidade de elucidar todas as etapas metodológicas necessárias para o desenvolvimento do presente trabalho.

#### **3.1 Classificação da pesquisa**

Uma pesquisa pode ser classificada por várias características. Algumas das modalidades de classificação de Gerhardt e Denise Tolfo Silveira (2009) são utilizadas: quanto à abordagem, este estudo se encaixa em uma pesquisa quantitativa, visto que o sistema a ser desenvolvido tem preocupação apenas com números, cálculos e lógicas matemática, como os valores das temperaturas e umidade, que são caracterizado como atributos mensuráveis; quanto à natureza, o objetivo do trabalho é gerar conhecimentos para aplicação prática da produção avícola, à vista disso, sendo classificada como uma pesquisa aplicada.

Para Gil (2002), as pesquisas podem ser classificadas com base em seus objetivos e com base nos procedimentos técnicos utilizados, sendo a caracterização do presente trabalho definida por essas modalidades, respectivamente: pesquisa exploratória, pois levanta-se a hipótese de que é possível criar um sistema de baixo custo que irá auxiliar em pequenas produções avícolas, e que pode ser caracterizado como um aprimoramento de ideias; como o processo de pesquisa é dado pela participação ativa do pesquisador e do objeto estudo (sistema), de forma em que será feito testes com a finalidade de encontrar o melhor resultado, sendo desenvolvido uma prototipagem final, a modalidade pode ser classificada como pesquisa-ação.

No âmbito da computação, Wazlawick (2009) fomenta as possíveis classificações da área, que são denominadas "Estilos de Pesquisa Correntes em Computação". Nesse contexto, esta pesquisa se encaixa melhor no estilo "Apresentação de algo Presumivelmente Melhor", devido o produto de desenvolvimento ter algumas semelhanças com alguns trabalhos relacionados, mas, até o momento presente, não encontra-se nenhum sistema de monitoramento e controle térmico conectado à *Internet* com foco no baixo custo.

#### **3.2 Solução**

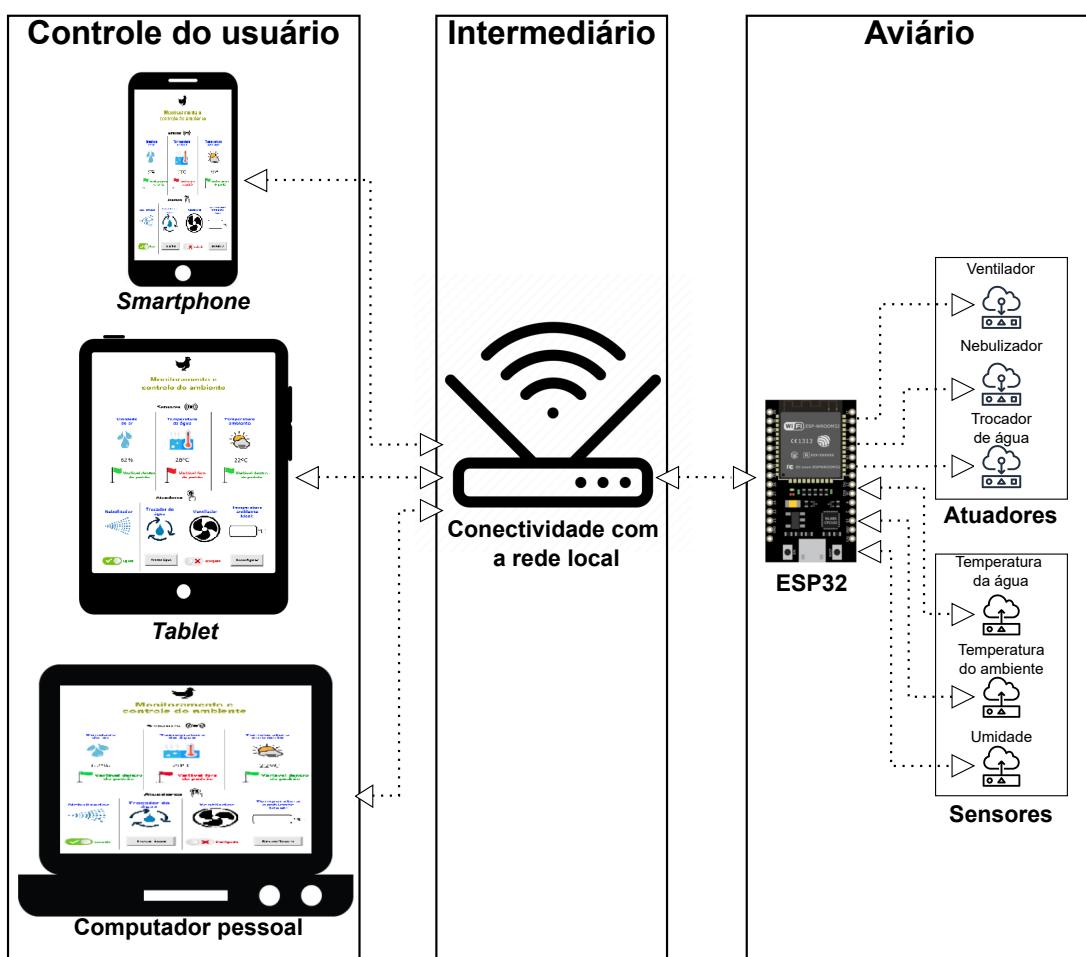
Como solução do problema identificado, foi desenvolvido um sistema de monitoramento e controle do ambiente aviário de baixo custo, desde que o pequeno avicultor já tenha alguns equipamentos básicos de criação do aviário, como ventiladores e nebulizadores instalados no local.

Toda a parte de controle e monitoramento do sistema é feita pelo usuário, podendo optar por um controle automático ou manual. Cada atuador tem seu modo de operação (automático ou manual). Quando o atuador está no modo automático, seu

funcionamento depende dos parâmetros de acionamento que podem ser alterados pelo usuário. Caso optar pelo modo manual, o atuador pode ser ligado/desligado pelo sistema.

Esse controle e monitoramento feito pelo usuário é realizado através de uma página *Web* hospedada no microcontrolador, que é acessada através do *WiFi*. O ESP32 consegue ser programado para construir uma interface *Web* que é acessada através do IP (*Internet Protocol*) responsável por identificar o microcontrolador dentro da rede. Esse servidor *Web* se comunica com o cliente por meio do protocolo HTTP. Um esquemático dessa conexão é mostrada na Figura 14.

Figura 14 – Esquemático da conexão do microcontrolador com a rede local.



Fonte: O Autor, 2021.

Para monitorar a temperatura e umidade do ambiente aviário, e a temperatura da água do bebedouro das aves foi necessário três sensores digitais. Dois sensores DS18B20, um responsável por medir a temperatura da água localizada dentro da caixa d'água do aviário. E um sensor DHT11 que mede a temperatura e a umidade do ambiente aviário.

### 3.2.1 Atuadores

O sistema tem uma flexibilidade neste quesito, em razão de que instalar um atuador pode vir a gerar custos inviáveis ao pequeno produtor. O ideal é que já tenha instalado os equipamentos e que seja feita uma adaptação para o sistema atuar neles. Caso ele não tenha condições de adquirir os atuadores, tem a opção de ficar apenas com a parte de monitoramento que avisa da irregularidade, tendo que tomar medidas por conta própria.

Pode ser controlado três atuadores que tentam melhorar as condições do ambiente: um ventilador para renovação do ar ambiente e melhorar a sensação térmica das aves; um nebulizador para fazer o resfriamento evaporativo do ar; um trocador de água que contém uma válvula solenoide para realizar a troca da água presente no bebedouro tipo *nipple*, que, de modo geral, é um bebedor que possui vários bicos de ingestão da água contida dentro de um cano instalado de forma longitudinal.

Os atuadores podem ser acionados remotamente via interface *Web*, ou funcionar de maneira automática. Cada atuador tem sua lógica de acionamento e parâmetros de acionamento, que podem ser configurados pelo usuário ou deixar funcionando pelo valor predefinido pelo sistema.

## 3.3 Materiais e Tecnologias

Além dos materiais e tecnologias fomentadas no capítulo 2, o trabalho fará o uso de algumas ferramentas de desenvolvimento (Seção 3.3.1) e também alguns materiais que estão incluídos no sistema embarcado (Seção 3.3.2).

### 3.3.1 Ferramentas utilizadas

Para desenvolver os códigos do sistema embarcado foi utilizado as plataformas apresentada na Tabela 2. As plataformas foram executadas em um *Notebook Samsung Expert X23 NP300E5M i5-7200U* com um sistema operacional *Windows 10 Pro* 64-bits.

Tabela 2 – Plataformas de *software* utilizadas para o desenvolvimento.

Ferramenta	Utilização
Google Chrome	Navegador utilizado para testes da interface <i>Web</i> .
<i>Visual Studio Code</i>	Editor de código-fonte para o desenvolvimento da interface <i>Web</i> .
<i>Arduino IDE</i>	Ambiente de desenvolvimento do código presente no microcontrolador.

Fonte: O Autor, 2021.

Tabela 3 – Tecnologias utilizadas no desenvolvimento do código.

Ferramenta	Utilização
HTML ( <i>HyperText Markup Language</i> — Linguagem de Marcação de Hipertexto)	Linguagem base para construir os elementos da página Web.
CSS	Linguagens utilizadas para construção da página Web.
JavaScript	Linguagens utilizadas para construção da página Web.
Bootstrap	Framework para organizar e gerenciar o layout da página Web.

Fonte: O Autor, 2021.

Tabela 4 – Ferramentas de *Hardware* e *Software* que serão utilizados para o desenvolvimento do sistema embarcado.

Ferramenta	Utilização
Notebook Samsung Expert X23 NP300E5M i5-7200U com Windows 10	Aparelho utilizado para implementar todo o código de desenvolvimento.
Google Chrome	Navegador utilizado para testes da interface Web.
Fonte de alimentação 5V	Componente responsável por alimentar o microcontrolador.
Protoboard	Placa de ensaio utilizada para simulações do circuito do sistema.
Jumpers	Cabos utilizados para conectar os componentes eletrônicos.
Cabo USB	Responsável por transferir o código do computador para o microcontrolador e também transferir energia da fonte para ele.
HTML, JavaScript e CSS	Linguagens utilizadas para construção da aplicação Web.
Visual Studio Code	Editor de código-fonte para o desenvolvimento da interface Web.
Bootstrap	Framework para organizar e gerenciar o layout da página Web.
Arduino IDE	Ambiente de desenvolvimento do código presente no microcontrolador.
Atuadores e Sensores	Equipamentos presente no aviário responsáveis por fazer a leitura dos valores a serem tratados.

Fonte: O Autor, 2021.

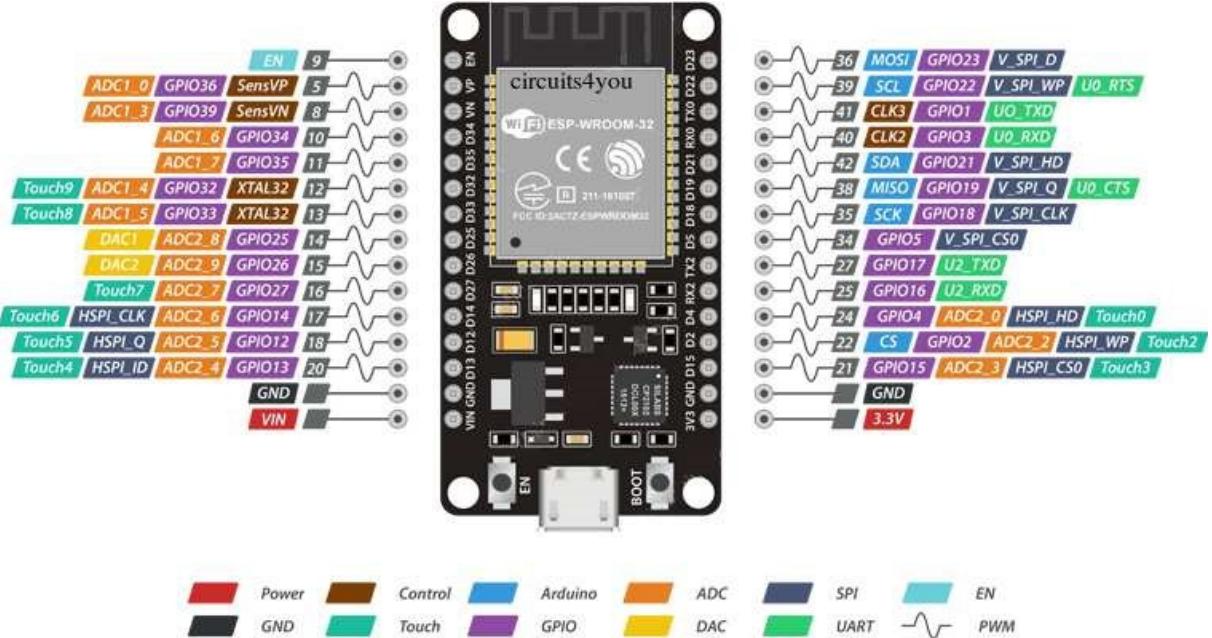
### 3.3.2 Materiais do protótipo

#### 3.3.2.1 ESP32 Devkit v1 - DOIT

Ele é responsável por controlar todo sistema e fazer papel de servidor Web. A placa de desenvolvimento (Figura 16) contém 30 pinos, podendo ser digitais ou analógicos.

Esses pinos são mostrados especificadamente na Figura 15.

Figura 15 – ESP32 Devkit v1 - *Pinout*.



Fonte: Adaptado de circuits4you (2018)

A Seção 2.2 do referencial teórico entra em detalhes sobre o chip microcontrolador ESP32, que está presente na placa de desenvolvimento.

Figura 16 – ESP32 Devkit v1 - DOIT.



Fonte: O Autor, 2021.

### 3.3.2.2 Válvula Solenoide para água 127 VAC

O equipamento é introduzido no cano do bebedouro tipo *nipple* para fazer o controle do trocamento da água. Em alguns casos, será necessário ser instalado no cano da água que circula para o nebulizador. A Válvula Solenoide (Figura 17) tem um ângulo de

180 graus entre o cano de entrada e saída da água, tem um funcionamento normalmente fechada (abre quando a corrente elétrica é circulada), suporta água de temperatura máxima de 60°C, entrada e saída do cano de 3/4", corrente nominal de 74 mAh e uma vazão máxima de 40 L/min.

Figura 17 – Válvula Solenoide para água 127 V VA 04.



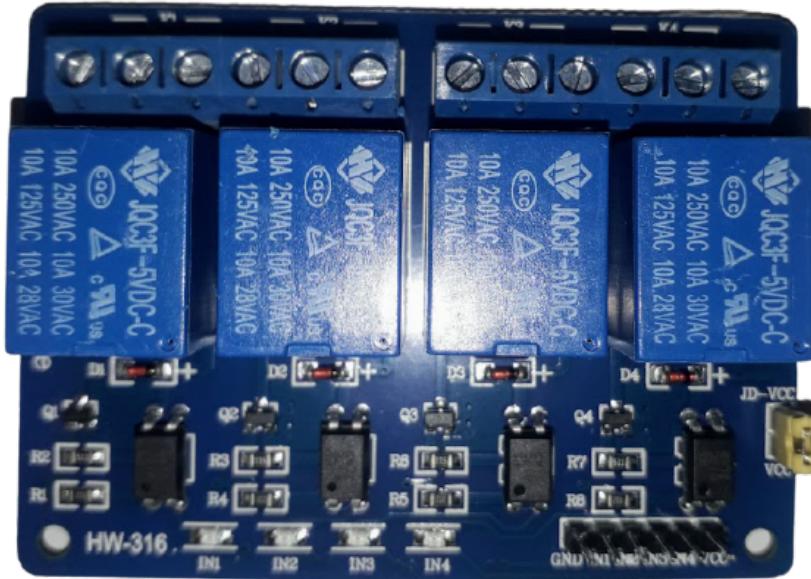
Fonte: O Autor, 2021.

### 3.3.2.3 Módulo Relé 5V 4 Canais

O Módulo Relé 5 V 4 Canais (Figura 18) é composto por 4 relés, dispositivos eletromecânicos que através da aplicação de uma corrente, gera uma indução eletromagnética que fecha ou abre um contato. Ele funciona como um interruptor, de forma que consegue-se isolar os circuitos de baixa tensão com os de alta tensão, evitando queimar equipamentos.

Ele servirá para acionar as válvulas solenoide, e os equipamentos do aviário que estão sendo controlados por um contator, podendo ser utilizado no controle de cargas AC (*Alternating Current*, ou, em português, Corrente Alternada) de até 10 A. O acionamento será controlado pelo ESP32, de modo que ação é mandada para o módulo relé que interrompe ou efetua o funcionamento do equipamento determinado.

Figura 18 – Módulo Relé 5V 4 Canais.



Fonte: O Autor, 2021.

### 3.4 Ambiente de experimento

Com o sistema todo desenvolvido, passado por testes e verificado que está funcionando perfeitamente, o sistema passou para um ambiente de teste real. Esse ambiente está localizado no Instituto Federal De Educação Ciência E Tecnologia De Minas Gerais - *Campus Bambuí*, na cidade de Bambuí, Minas Gerais, em um aviário. A Figura 19 mostra o ambiente aviário.

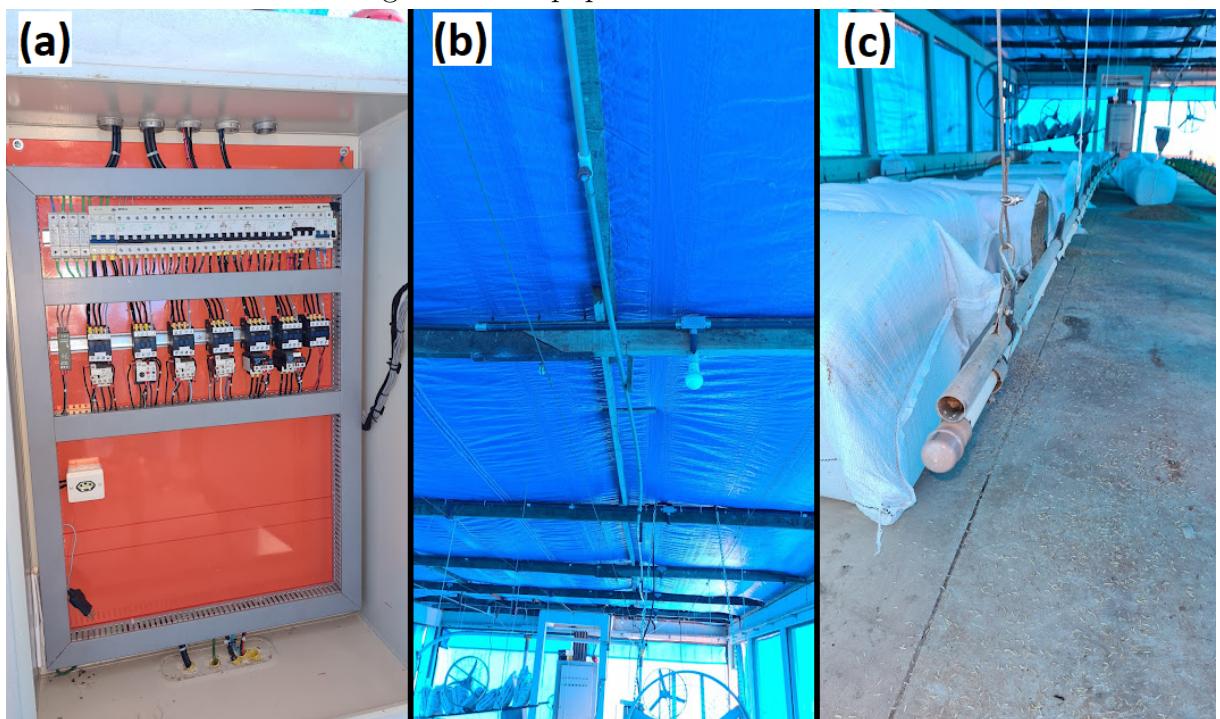
Figura 19 – Aviário de experimento.



Fonte: O Autor, 2021.

O sistema foi instalado no aviário com o objetivo de avaliar o funcionamento do sistema e sua eficiência. O sensor de temperatura e umidade deve ficar localizado no meio do aviário, na altura dos frangos de corte. Um dos canos dos bebedouros tipo *nipple* foi adaptado para funcionar com o sistema, adicionando as válvulas de entrada e saída da água, juntamente com o sensor de temperatura da água. O acionamento do ventilador e nebulizador será feito com a ajuda do contator instalada no ambiente, de modo que os relés do microcontrolador mandará sinais para ligar/desligar o atuador devido. A Figura 20 mostra os equipamentos presentes no aviário.

Figura 20 – Equipamentos do aviário.



Legenda: (a) Painel de força e comando; (b) Nebulizador e (c) Bebedouro tipo *nipple*.

Fonte: O Autor, 2021.

### 3.5 Métodos e Procedimentos

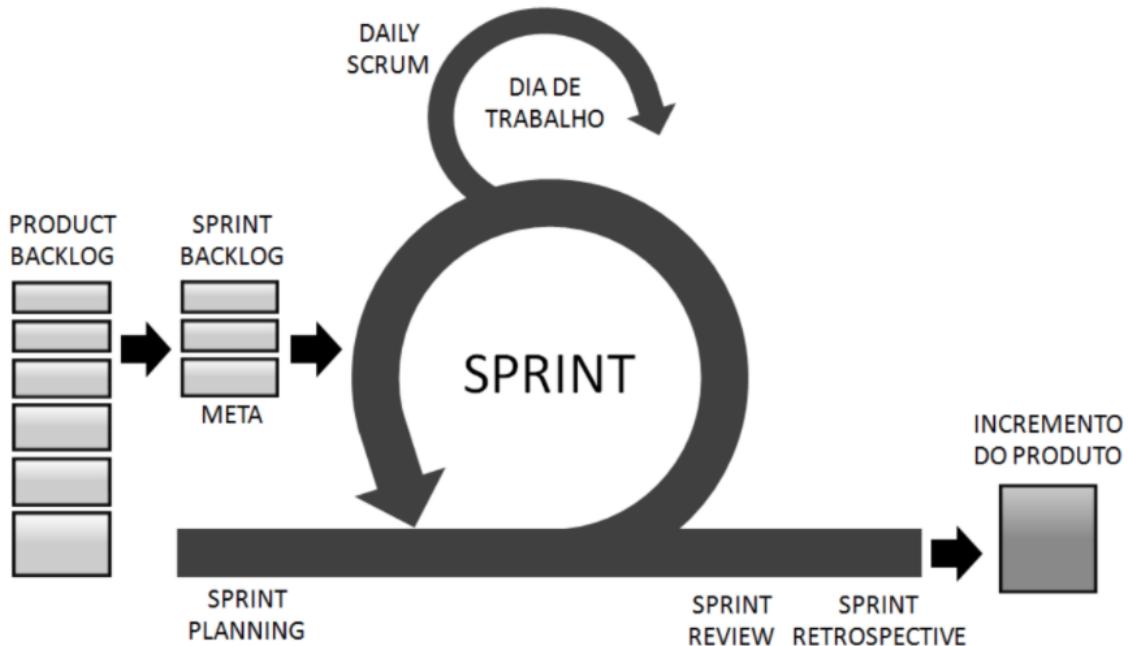
#### 3.5.1 Metodologia de desenvolvimento

Para o processo de desenvolvimento, foi utilizado o *Scrum*, que é um *framework* de metodologia ágil. Sabbagh (2014) aponta que a ferramenta vinha sendo utilizada por organizações de diversos tamanho, de multinacionais a *startups*, e, quando aplicada de maneira correta em um projeto, pode-se esperar os seguintes benefícios: entregas frequentes de retorno ao investimento dos clientes; redução dos riscos do projeto; maior qualidade no produto gerado; mudanças utilizadas como vantagem competitiva; visibilidade do progresso do projeto; redução do desperdício; aumento de produtividade.

O orientador teve o papel de *Product Owner* e *Scrum Master*, e a equipe de desenvolvimento será composta apenas pelo orientando do trabalho, acontecendo reuniões de *Sprint* em um período de 7 dias (a cada uma semana) para novas orientações e observações do projeto.

O *Product Backlog* contará com estórias de todos os tipos necessários para desenvolver o trabalho, como revisões bibliográficas de materiais, consulta a manual, desenvolvimento de *hardware* e *software*, simulações, entre outras. O *Product Backlog* é composto por todas as tarefas que precisam ser feitas, o *Sprint Backlog* ficam as tarefas que foram almejadas para acabar no presente *Sprint*, como pode-se ver o ciclo completo do Scrum na Figura 21.

Figura 21 – O ciclo do Scrum.



Fonte: Sabbagh (2014)

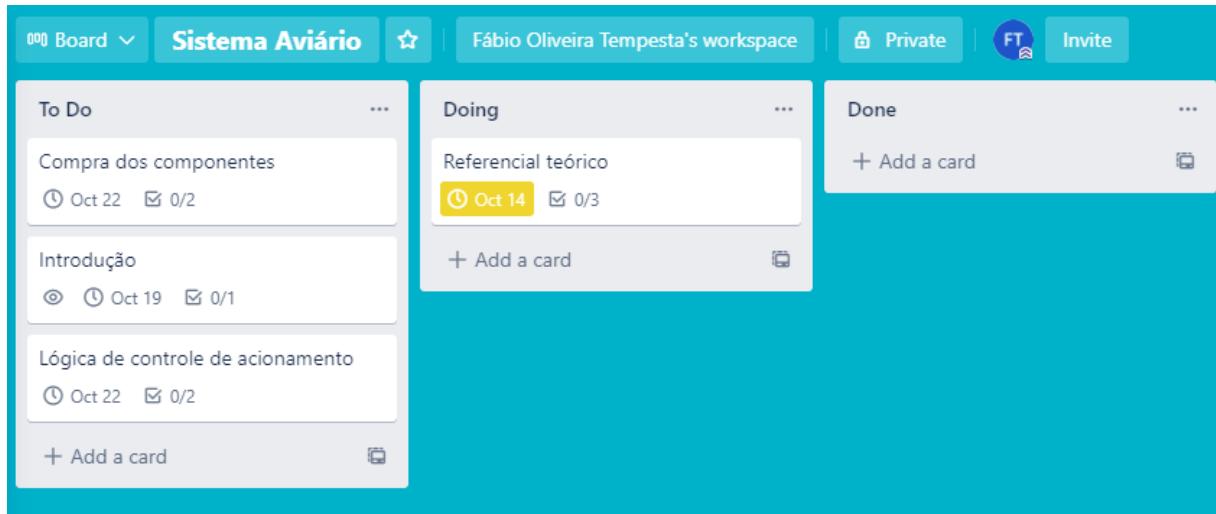
Para um melhor controle do fluxo de desenvolvimento de trabalho, foi adotada a ferramenta Trello<sup>1</sup>, que auxilia no gerenciamento de projetos em listas, para utilizar um quadro de *Kanban*, como está apresentado na Figura 22 com as atividades que precisam ser feitas com uma lista inclusa em cada cartão.

### 3.5.2 Definição dos requisitos

Após realizar estudos, obter relatos de especialistas da área de avicultura e uma visita realizada ao aviário de ambiente de experimento, foi feito um levantamento de requisitos com a finalidade de suprir as reais necessidades do usuário.

<sup>1</sup> <https://www.trello.com/>

Figura 22 – Quadro de *Kanban* com tarefas iniciais.



Fonte: O Autor, 2021.

Os requisitos são divididos em duas categorias maiores: não funcionais e funcionais. Os requisitos não funcionais são aqueles relacionados a forma que o sistema será desenvolvido para tornar realidade o que está sendo planejado. Já os requisitos funcionais descrevem os problemas e necessidades que devem ser resolvidos pelo sistema.

## 1. Requisitos Não Funcionais

### a) Requisitos Operacionais

- i. O sistema deve funcionar em qualquer navegador *Web*
- ii. O usuário poderá escolher entre funcionamento automático e manual dos equipamentos
- iii. O usuário poderá ligar e desligar qualquer um dos atuadores

### b) Requisitos de Desempenho

- i. Os valores das condições do ambiente captadas pelos sensores deve ser atualizada na interface a cada cinco minutos

### c) Requisitos Segurança

- i. O sistema deverá ter uma alguma forma de autenticação antes do acesso à interface

### d) Requisitos Cultural e Político

- i. Nenhum requisito cultural e político foi proposto

## 2. Requisitos Funcionais

### a) Interface

- i. A página *Web* deve ser simples e de fácil compreensão

b) Custo

- i. O preço final do produto deve ser acessível aos pequenos produtores

### **3.5.3 Procedimentos**

Como foi argumentado no capítulo 1, o presente trabalho é caracterizado pelo desenvolvimento de um sistema que auxiliará os pequenos ambientes de criação de frango. Assim, foi necessário iniciar o primeiro passo realizando pesquisas mais aprofundadas sobre os animais e os locais de criação. Após a fundamentação, foi feito uma atividade de confirmação, que será realizada de forma eficaz e com rapidez, dos equipamentos necessários para o desenvolvimento do projeto, analisando os manuais e comparando com os requisitos da aplicação, para maior segurança do projeto e da compra. O ambiente de desenvolvimento de *software* e *hardware* foi sendo preparado durante o período de chegada dos equipamentos.

Após a chegada do controlador e dos demais sensores, iniciou-se o processo de implementação dos componentes de *software*, configuração dos sensores e, consequentemente, permitir a leitura das variáveis do ambiente de criação. Juntamente a isto, foi desenvolvido o acionador responsável por trocar a água de consumo dos animais, tanto a parte mecânica, quanto a parte lógica.

Com os sensores e atuadores funcionando corretamente no controlador, foi desenvolvida a interface do servidor *Web*, estudando as linguagens de desenvolvimento *Web*, focando em um baixo custo de processamento, devido à pequena capacidade do microcontrolador trabalhar como servidor. Em seguida, com o servidor *Web* operando internamente, foi feito as devidas configurações para a interface poder operar externamente.

Para desenvolver a automação dos atuadores, foram realizados estudos e testes, com a finalidade dos acionamentos do sistema serem feitos de forma inteligente e eficaz. Em paralelo a isso, foi feita as calibragens dos sensores para o sistema ter os valores mais próximos do real.

O acabamento do projeto consistiu na validação da precisão dos sensores comparada com resultados obtidos em sensores industriais e da aplicação do sistema no ambiente de experimento. Esse experimento consistiu na captação dos valores dos sensores a cada hora durante dias. Na primeira instância, o aviário terá seu funcionamento de forma tradicional, sem o auxílio do sistema. Na outra instância, o aviário funcionará de forma automática, sendo controlada pela aplicação. No final das captações, foi comparado os resultados dos sensores com e sem o auxílio do sistema embarcado.

## 4 DESENVOLVIMENTO

Este capítulo aborda o desenvolvimento do protótipo do sistema, tanto na parte lógica quanto na parte física.

As Seções 4.1 e 4.1 são referentes à leitura dos sensores. Todas as duas ligações elétricas integram um resistor de  $4,7k\ \Omega$  para fazer o papel de resistor de *pull-up* na linha de dados *1-Wire*. Como os dois sensores tem em comum o protocolo *1-Wire*, as suas ligações também são caracterizadas pelas conexões de três fios: um para dados, uma alimentação de 3,3 V e o GND.

Em sequência, as Seções 4.3, 4.4 e 4.5 demonstram o desenvolvimento realizado nos atuadores. Todos os atuadores utilizam o módulo relé para fazer o acionamento e desligamento do ventilador pelo microcontrolador. As ligações elétricas dos atuadores mostram o módulo relé alimentado por uma tensão de 5 V, por uma fonte externa. Essa fonte precisa ter seu terra (GND) conectado com o terra do microcontrolador para o módulo relé ter referência dos sinais de controle do ESP32.

O módulo relé entra em ação quando o ESP32 envia um sinal de uma porta digital para alguma entrada do módulo para atracar/desatracar o relé. O relé responsável por aquela entrada libera a passagem de corrente da fase da tomada enquanto ele estiver desatracado, devido os atuadores serem conectados na configuração normalmente fechado. Como o módulo relé do protótipo atraca os dispositivos eletromecânicos quando a tensão de entrada é baixa (aproximadamente 0 V), o ESP32 deve mandar uma tensão alta (aproximadamente 3,3 V) para liberar a corrente da fase e, por consequência, fechar o contato do contator elétrico, alimentando o ventilador.

### 4.1 Leitura do sensor DHT11

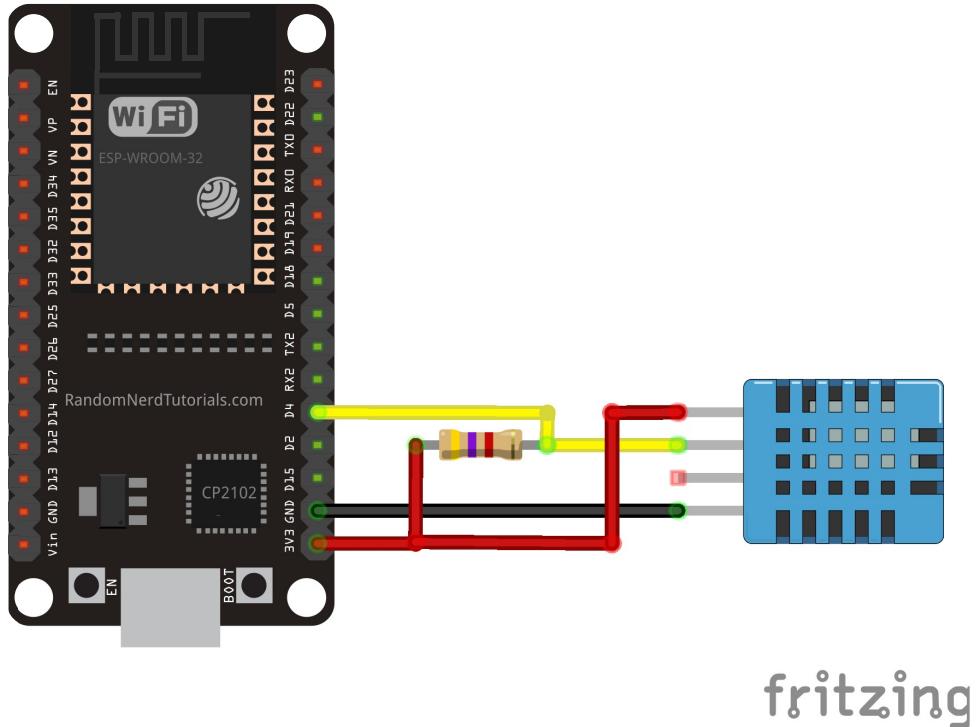
O *hardware* de leitura do sensor de temperatura e umidade do ar (DHT11) foi desenvolvido da forma que o microcontrolador ESP32 Devkit v1 comunica com o sensor para captar os valores de umidade e temperatura. O esquema de ligação elétrica pode ser observado na Figura 23.

A porta de 3,3 V do microcontrolador (3V3) alimenta tanto a porta de alimentação do sensor quanto a linha de comunicação dos dispositivos. Logo acima, a porta GND do ESP32 faz ligação apenas com a porta GND do sensor DHT11. A porta D4 é responsável pela comunicação, ela pode requisitar e receber dados do sensor.

A implementação lógica do controlador utilizou a biblioteca "DHT.h"<sup>1</sup> para ficar responsável por toda a conversão dos sinais digitais. Após requisitar o sensor de uma leitura, a biblioteca já traz os valores de temperatura prontos para ser utilizado.

<sup>1</sup> <https://github.com/adafruit/DHT-sensor-library>

Figura 23 – Ligações elétricas para leitura do sensor DHT11.



Fonte: O Autor, 2021.

## 4.2 Leitura dos sensores DS18B20

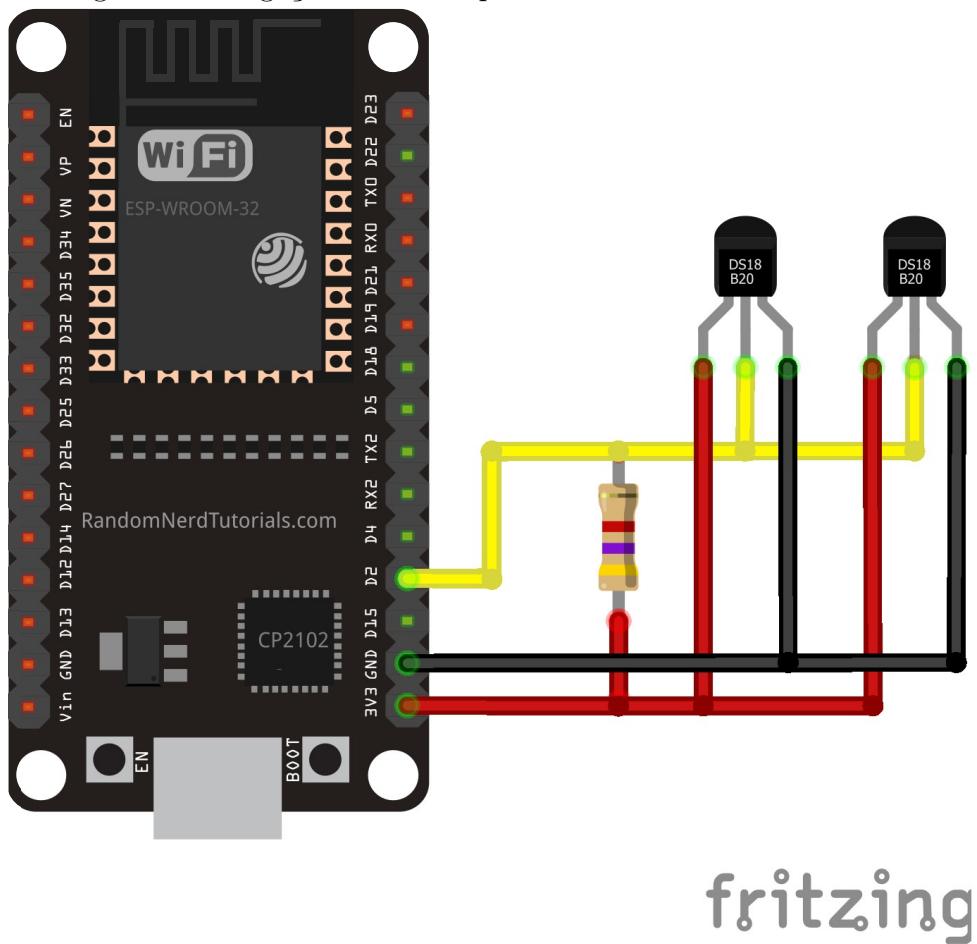
A ligação elétrica do sensor DS18B20 é semelhante ao DHT11. O ESP32, além de alimentar, faz a comunicação com os sensores, sendo que os sensores só respondem quando são requisitados.

Na Figura 24 mostra esquema de ligação elétrica dos sensores com o controlador. Vale lembrar que os sensores do presente trabalho possuem o revestimento de proteção contra água, já o da imagem não. A porta D2 do controlador fica responsável por comunicar com os sensores via protocolo *1-Wire*. Para diferenciar os sensores, os dispositivos que utilizam o protocolo de comunicação *1-Wire* possuem um número de identificação (ID) de 64 bits exclusivo, inalterável e programado de fábrica.

Os dois sensores tem suas alimentações e GND conectados no GND e na alimentação de 3,3 V do ESP32. A comunicação *1-Wire* (fio amarelo) conta com uma alimentação de 3,3V e um resistor de *pull-up* para um funcionamento correto.

Na implementação do controlador, foi necessário a utilização de duas bibliotecas disponíveis na própria IDE Arduino, como mostra na Tabela 5.

Figura 24 – Ligações elétricas para leitura do sensor DS18B20.



Fonte: O Autor, 2021.

Tabela 5 – Bibliotecas utilizadas no código de leitura do sensor DS18B20.

Biblioteca	Autor	Descrição
DallasTemperature.h	Paul Stoffregen	Biblioteca feita para utilizar os sensores de temperatura da <i>Maxim Integrated</i> (antiga Dallas).
OneWire.h	Miles Burton	Biblioteca para acessar dispositivos que utilizam o protocolo <i>1-Wire</i> .

Fonte: O Autor, 2021.

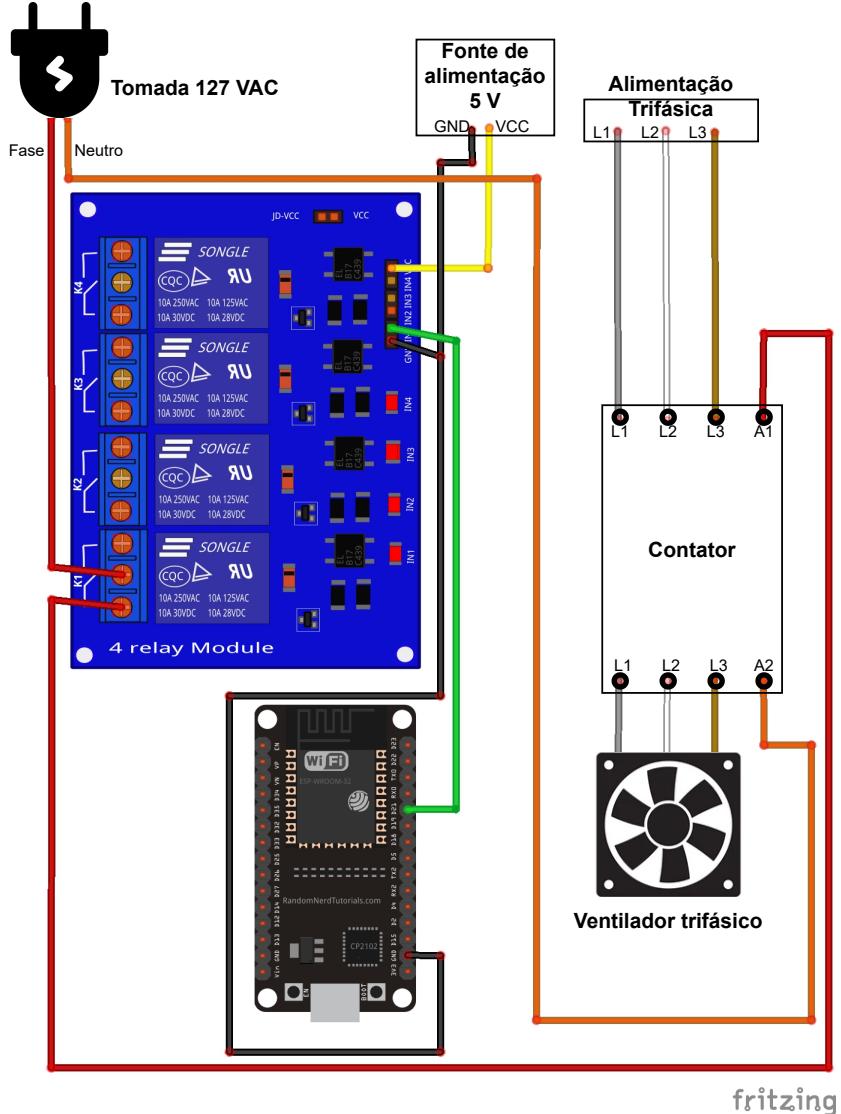
#### 4.3 Ventilador

A Figura 25 demonstra a ligação elétrica desenvolvida para ventiladores industriais que são alimentados por uma tensão trifásica. O ventilador trifásico é ligado assim que o contator elétrico é atracado. Para o atracamento ocorrer, a fase conectada no relé K1, que está embutido no módulo, deve estar liberada para chegar até o contator elétrico. Os 10 amperes do relé são suficiente para o atracamento do contator de maneira segura.

O sinal responsável por liberar a passagem da fase vem da porta digital D21

do microcontrolador. O sinal chega até a primeira entrada do módulo relé (IN1).

Figura 25 – Ligações elétricas para para funcionamento do ventilador trifásico.



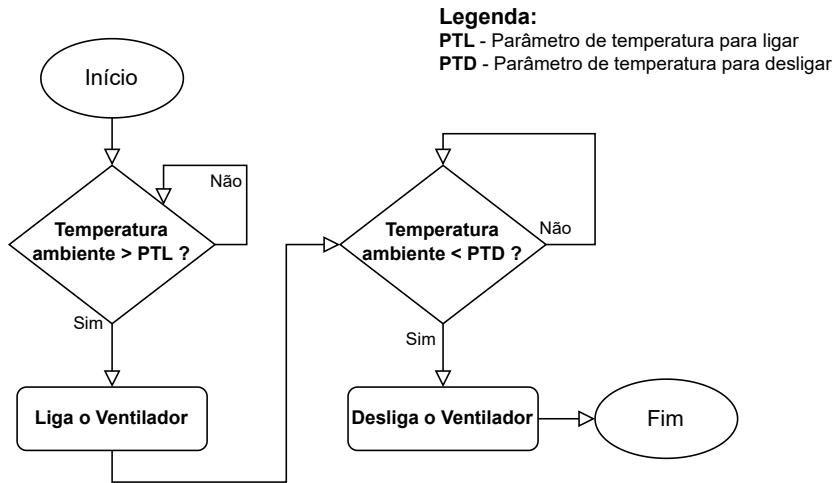
Fonte: O Autor, 2021.

A sua lógica de acionamento no modo automático depende de dois parâmetros de temperatura, um para ligar e o outro para desligar. Esses valores são dados na unidade de medida de °C. O parâmetro de ligar funciona de modo que quando a temperatura ambiente medida pelo sensor estiver acima do valor dele, o ventilador deve ser acionado. Já o parâmetro de desligar atua quando o valor informado no sensor estiver abaixo do valor desse parâmetro. A Figura 26 apresenta esse funcionamento por meio de fluxograma, em que o ventilador começa em um estado inicial desligado.

#### 4.4 Nebulizador

A Figura 27 demonstra a ligação elétrica desenvolvida para aviários com bomba de água com motor trifásico para ajudar a água chegar aos canos do nebulizador. O ESP32,

Figura 26 – Lógica de acionamento do ventilador em modo automático.



Fonte: O Autor, 2022.

nesse caso, envia um sinal de uma porta digital (D19) que é responsável por duas entradas do módulo relé (IN2 e IN3). As ligações poderiam utilizar apenas um relé para fazer o acionamento dos equipamentos, mas foi utilizado dois para garantir a segurança de que a corrente não exceda os 10 amperes.

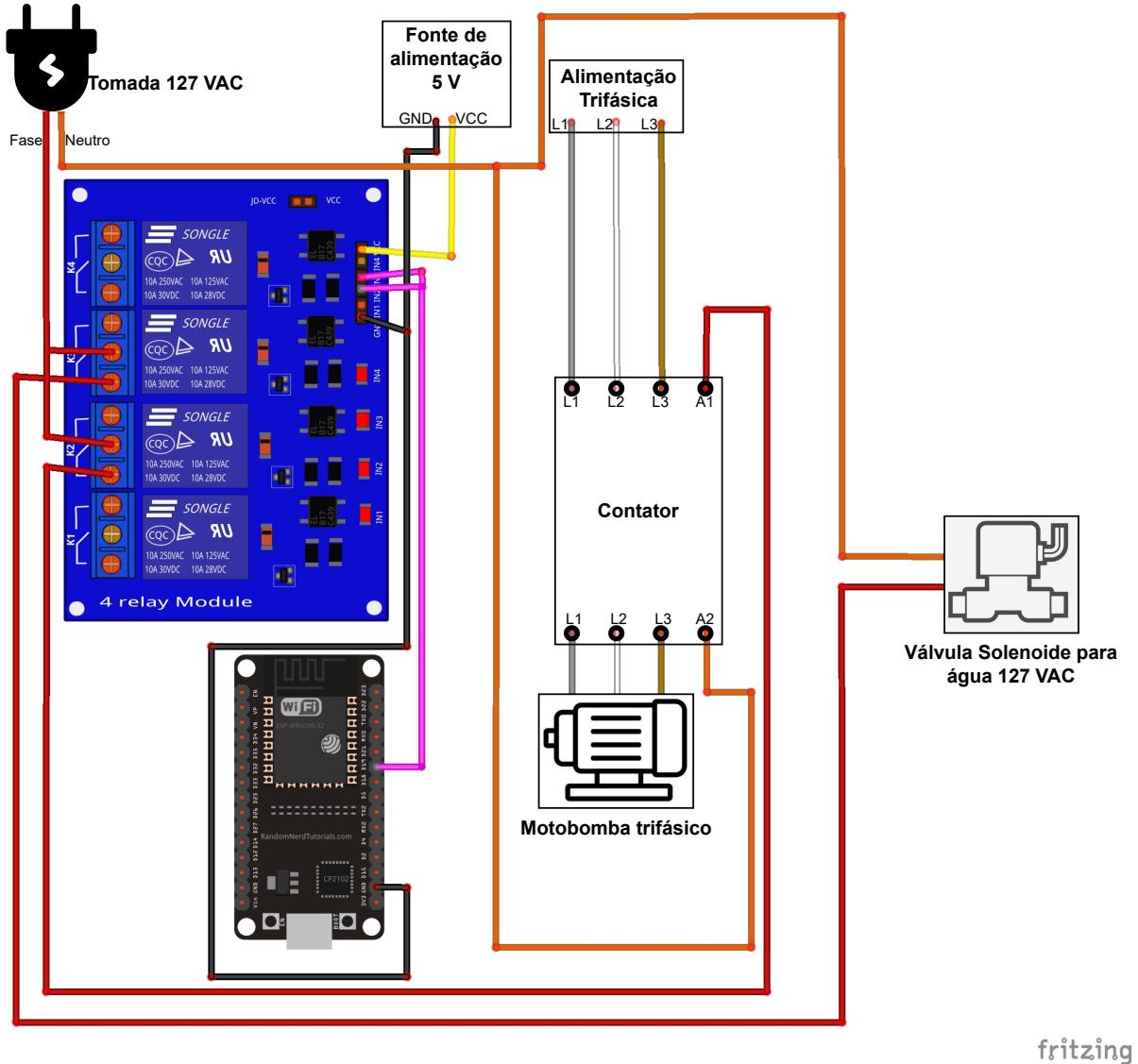
Quando o nebulizador deve ser ligado, o relé K2 e K3 libera a passagem de corrente da fase da tomada. O relé K2 faz que a alimentação da tomada chegue até a válvula solenoide, liberando o fluxo de água nos canos do nebulizador. A fase do relé K3 chega até ao contator, atracando ele e, por consequência, a motobomba sendo alimentada pela rede trifásica.

O nebulizador, tanto no modo automático quanto no modo manual, opera juntamente com o ventilador, precisando ligá-lo em casos que ele esteja desligado.

A sua lógica de acionamento no modo automático depende de dois parâmetros que podem ser configurados na interface Web. Como dito na subseção 2.1.2, o resfriamento evaporativo funciona melhor em umidades baixas. Nesse caso, os parâmetros de acionamento do nebulizador consiste na temperatura mínima de acionamento, em °C, e a umidade relativa máxima, em porcentagem.

A lógica de acionamento do nebulizador é demonstrada na Figura 28 por meio de um fluxograma, em que o nebulizador começa desligado. Seguindo o fluxograma, o equipamento é acionado quando o sensor DHT11 estiver informando uma temperatura acima do seu devido parâmetro juntamente com uma umidade relativa abaixo do valor do parâmetro responsável por ela. Para ser desligado, algum dos valores do sensor devem estar fora da faixa do parâmetro de acionamento com a soma de um valor mínimo para evitar que o equipamento ligue e desligue frequentemente em um curto período de tempo.

Figura 27 – Ligações elétricas para para funcionamento do Nebulizador.



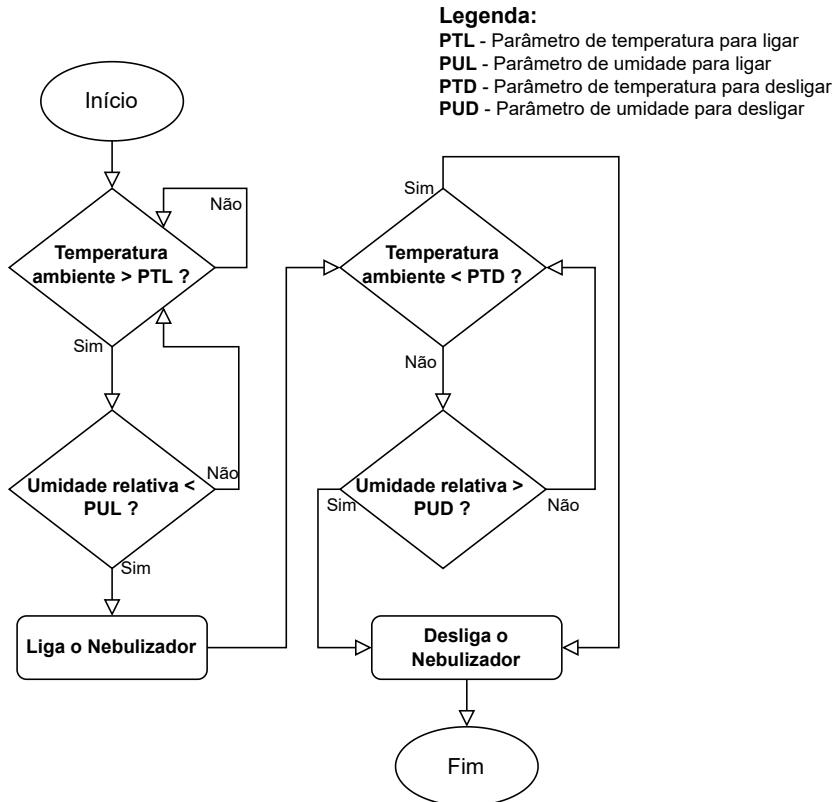
Fonte: O Autor, 2021.

#### 4.5 Trocador de água

O trocador de água não faz o uso de alimentação trifásica para seus equipamentos, como o nebulizador e ventilador. A Figura 29 demonstra as ligações elétricas para o funcionamento do mecanismo. O ESP32 envia um sinal de uma porta digital (D18) para a quarta entrada do módulo relé (IN4) para a conexão da fase conectada no relé K4 liberar a passagem de corrente para a válvula solenoide. Com a válvula alimentada, a água presente no cano do bebedouro é retirada. Quando D18 manda um sinal baixo para a entrada, a válvula é fechada novamente.

Esse atuador tenta resolver problema de temperatura elevada da água dos bebedouros. Como a caixa de água do aviário deve ser instalados em locais fora da exposição do sol, existe uma chance considerável dela estar um pouco mais fresca que a água localizada nos canos dos bebedouros. A automatização foi desenvolvida para

Figura 28 – Lógica de acionamento do nebulizador em modo automático.



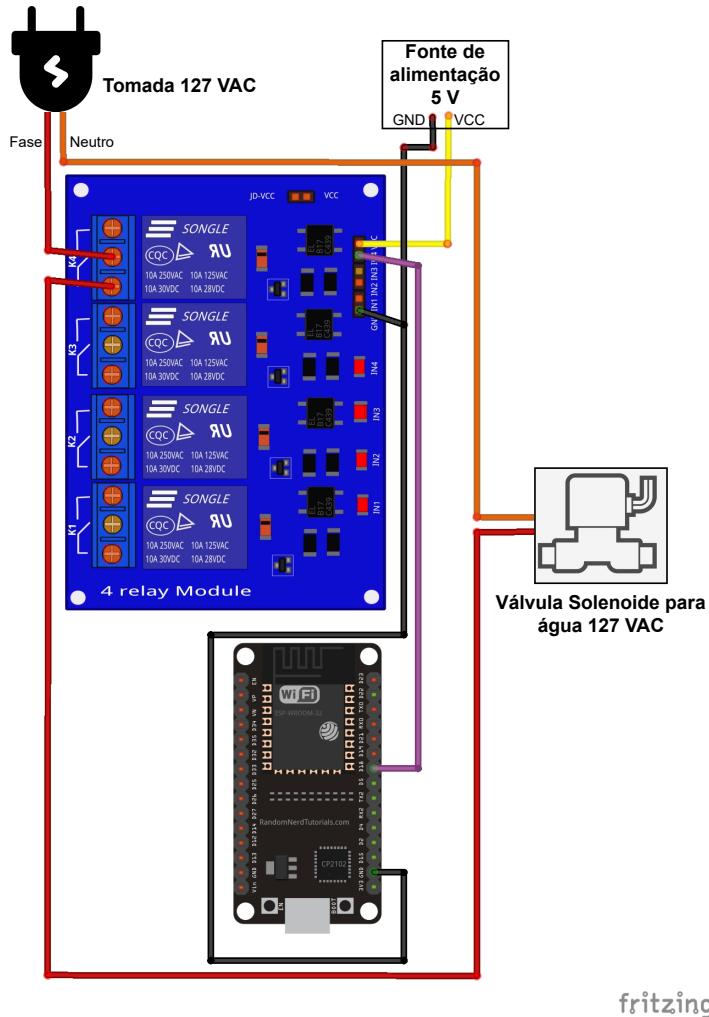
Fonte: O Autor, 2022.

bebedouros tipo *nipple*, que são comumente utilizados nos aviários. Para esta implantação, foi adicionado uma válvula solenoide (VA), que tem o objetivo de trocar a água "velha" pela "nova", no bebedouro. Para uma troca inteligente, um sensor foi instalado dentro da água do bebedouro (S1) e outro dentro da água da caixa d'água (S2). A Figura 30 ilustra esse funcionamento, de forma que a entrada de água do bebedouro fica sempre aberta, deixando a água presente dentro dos canos pressurizada. Quando a válvula VA abre, a água presente no bebedouro cai dentro do recipiente, podendo ser reutilizada para outra atividade, e, em seguida, a água que estava na caixa chega aos canos.

Quando o equipamento está no modo de operação automático, seu acionamento depende de duas variáveis que podem ser modificadas através da interface Web. Uma delas se trata da diferença limite de temperatura entre os sensores S1 e S2 para realizar o acionamento, tendo um valor predefinido de 5°C. A outra se refere ao tempo, em segundos, da duração da troca de água, ou seja, o tempo que a válvula deve ficar aberta para conseguir trocar a água "velha" pela "nova".

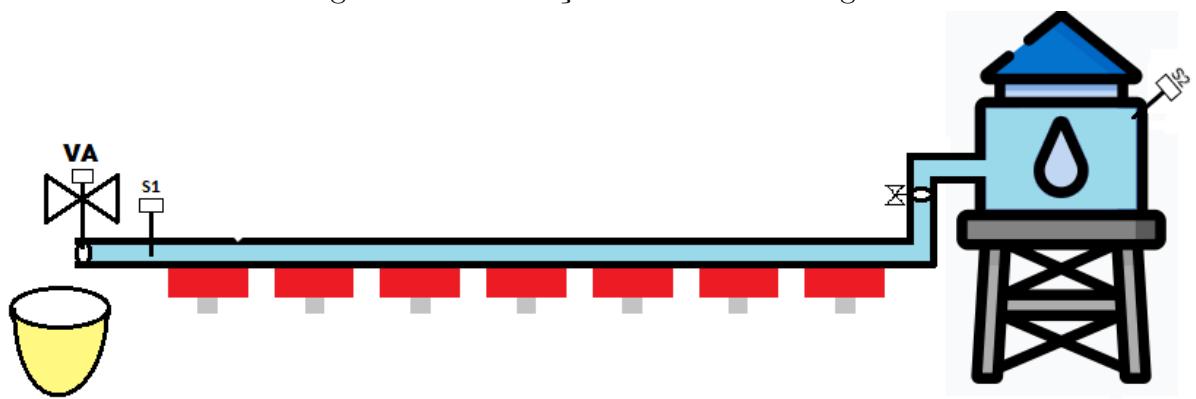
No fluxograma que apresenta sua lógica de acionamento (Figura 31), o trocador de água tem seu estado inicial com a válvula fechada. Se a temperatura medida em S1 exceder a temperatura medida em S2 mais o valor do parâmetro de temperatura para ligar (PTL), o trocador abrirá a válvula VA. A válvula apenas fechará se o usuário interromper

Figura 29 – Ligações elétricas para para funcionamento do trocador de água.



Fonte: O Autor, 2021.

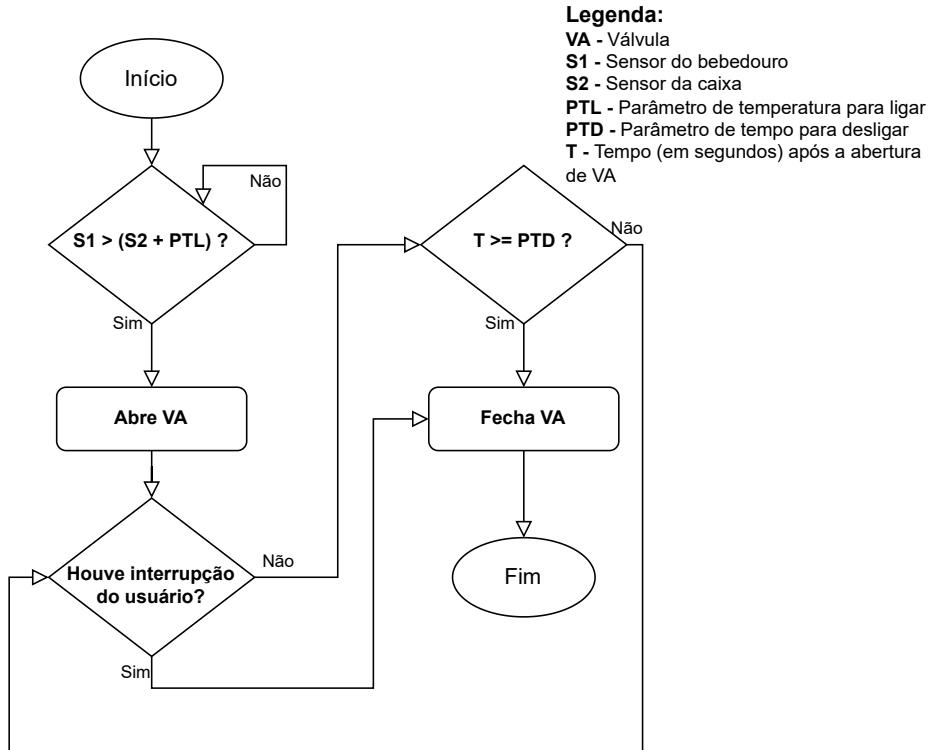
Figura 30 – Ilustração do trocador de água



Fonte: O Autor, 2021.

a troca de água via interface Web, ou após a troca de água ter sido concluída. Essa conclusão da troca de água acontece após o tempo, em segundos, que a válvula VA for aberta atingir o valor do parâmetro responsável (PTD).

Figura 31 – Lógica de acionamento do trocador em modo automático.



Fonte: O Autor, 2022.

#### 4.6 Interface Web

O usuário faz o acesso à interface utilizando um computador, celular *smartphone*, *tablet*, ou qualquer dispositivo tecnológico que execute um navegador de internet.

A página Web é dividida em duas partes: atuadores e sensores (Figura 32). Na parte dos sensores é informado o valor atual de cada sensor. Na parte dos atuadores é apresentado o status atual e o modo de operação de cada atuador. Quando um atuador está no modo automático, a interface apresenta os parâmetros responsáveis pela lógica de acionamento na sua parte inferior, tanto para ligar quanto para desligar o atuador. Caso esse atuador estiver no modo manual, essa parte inferior do atuador apresenta ao usuário a opção de ligar ou desligá-lo.

#### 4.7 Custo material do protótipo

Para mensurar o valor dos materiais presentes no sistema embarcado, foi levantado o preço de cada dispositivo e componente em lojas online, no mercado brasileiro, obtendo-se um valor médio. A Tabela 6 a seguir, apresenta o custo médio encontrado no levantamento e a quantidade em unidades gasto do material.

Figura 32 – Captura de tela da página Web.

### SENSORES (•)

<b>TEMPERATURA AMBIENTE</b> <b>26,2°C</b> 	<b>UMIDADE RELATIVA</b> <b>51%</b> 	<b>TEMPERATURA DA ÁGUA DO BEBEDOURO</b> <b>22,1°C</b> 	<b>TEMPERATURA DA ÁGUA DA CAIXA</b> <b>21,3°C</b> 
---	--	--	---

### ATUADORES ☰

<b>NEBULIZADOR</b>  <b>► Status: DESLIGADO</b> <b>► Modo de operação</b> <input type="button" value="Automático"/> <input type="button" value="Manual"/> <b>► Parâmetros do modo automático</b> <input type="radio"/> Ligar quando: <input type="radio"/> Desligar quando: A umidade relativa do ar estiver maior que <input type="text" value="90"/> % <input type="button" value="Salvar"/> ou A temperatura ambiente estiver menor que <input type="text" value="20,00"/> °C <input type="button" value="Salvar"/>	<b>TROCADOR DE ÁGUA</b>  <b>► Status: DESLIGADO</b> <b>► Modo de operação</b> <input type="button" value="Automático"/> <input type="button" value="Manual"/> <b>► Parâmetros do modo automático</b> <input type="radio"/> Ligar quando: <input type="radio"/> Desligar após: A diferença de temperatura entre o bebedouro e a caixa estiver maior que <input type="text" value="5,40"/> °C <input type="button" value="Salvar"/>	<b>VENTILADOR</b>  <b>► Status: LIGADO</b> <b>► Modo de operação</b> <input type="button" value="Automático"/> <input type="button" value="Manual"/> <b>► Acionamento do atuador</b> <input type="button" value="Desligar"/>
--	--	--

Fonte: O Autor, 2022.

Tabela 6 – Preços do protótipo

Item	Quantidade	Valor (R\$)
ESP32 - Devkit v1	1	40,70
Válvula Solenoide para Água 127V	2	118,40
Sensor de temperatura a prova de água DS18B20	2	38,30
Jumpers de 20 cm	30	12,80
Módulo Relé 5 V - 4 canais	1	29,45
Cabo Flexível Paralelo 300V (2 X 4,0mm) / Por Metro	3	26,85
Sensor de Umidade e Temperatura DHT11	1	14,50
Fonte 5 V DC	1	33,39

Fonte: O autor, 2022.

## REFERÊNCIAS

- ABPA. **Relatório Anual**. 2021. Disponível em: [http://abpa-br.org/wp-content/uploads/2021/04/ABPA\\_Relatorio\\_Annual\\_2021\\_web.pdf](http://abpa-br.org/wp-content/uploads/2021/04/ABPA_Relatorio_Annual_2021_web.pdf). Acesso em: 3 ago. 2021.
- ABREU, V. M. N.; ABREU, P. G. de. Os desafios da ambiência sobre os sistemas de aves no Brasil. **Revista Brasileira de Zootecnia**, Embrapa Suínos e Aves, Viçosa, v. 40, p. 1–14, 2011.
- ALECRIM, P. D. de; CAMPOS, A. T.; JÚNIOR, T. Y. Sistema automatizado embarcado em microcontrolador para controle e supervisão do ambiente térmico para aviários. **Revista Científica**, Jaboticabal, v. 41, n. 1, p. 33–45, 2013.
- BARBOSA, T. M. A importância da água na avicultura. Brasília, Distrito Federal, 2013.
- BRAGA, N. C. **Como funcionam os Sensores de temperatura (ART764)**. Instituto Newton C. Braga, 2012. Disponível em: <https://www.newtoncbraga.com.br/index.php/como-funciona/6097-art764>. Acesso em: 3 ago. 2021.
- CARDOSO, M. **O Que É Um Microcontrolador?** IEEE RAS UFCG - Capítulo Estudantil de Robótica e Automação, 2020. Disponível em: <https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/>. Acesso em: 27 out. 2021.
- CASSUCE, D. C. *et al.* Thermal comfort temperature update for broiler chickens up to 21 days of age. **Revista Engenharia Agrícola**, Jaboticabal, v. 33, n. 1, p. 28–36, 2013.
- CIRCUITS FOR YOU. **ESP32 DevKit ESP32-WROOM GPIO Pinout**. Dez. 2018. Disponível em: <https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>. Acesso em: 26 mar. 2022.
- CONNOLLY, A. **Era digital: o futuro da tecnologia avícola**. 2018. Disponível em: <https://www.aviculturainustrial.com.br/imprensa/era-digital-o-futuro-da-tecnologia-avicola/20180522-093843-w047>. Acesso em: 3 ago. 2021.
- D-ROBOTICS UK. **DHT11 Humidity & Temperature Sensor**. Jul. 2019. Disponível em: <https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1>. Acesso em: 3 nov. 2021.
- ESPRESSIF. **ESP32-WROOM-32 Datasheet**. Versão 3.7. 2021. Disponível em: [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf). Acesso em: 3 ago. 2021.
- FONTES, H. **Sistema que controla clima de aviário pode evitar perdas na produção de frangos**. Jornal da USP, 2020. Disponível em: <https://jornal.usp.br/ciencias/sistema-que-controla-clima-de-aviario-pode-evitar-perdas-na-producao-de-frangos/>. Acesso em: 3 ago. 2021.
- GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. Porto Alegre: Editora da UFRGS, 2009.

- GIL, A. C. **Como elaborar projetos de pesquisa.** 4. ed.: São Paulo: Editora Atlas, 2002.
- GOGONI, R. **O que é Internet das Coisas?** 2019. Disponível em: <https://tecnoblog.net/responde/o-que-e-internet-das-coisa/>. Acesso em: 20 jan. 2022.
- GROOVER, M. P. **Automação industrial e sistemas de manufatura.** 3. ed.: São Paulo: Editora Pearson Education do Brasil, 2011.
- GUERRA, R. R. NOÇÕES BÁSICAS EM VENTILAÇÃO PARA AVIÁRIOS. In: EM TEMPOS DE GRÃOS CAROS, QUAIS AS ESTRATÉGIAS? Rio Claro, São Paulo: Revista aviNews Brasil, 2021. P. 7–13.
- HÖTZEL, M. J.; MACHADO FILHO, L. C. P. Bem-estar animal na agricultura do século XXI. **Revista de Etologia**, São Paulo, v. 6, p. 3–15, jun. 2004.
- JÚNIOR, V. P. d. S. **Microcontroladores PIC 16F e 18F–Teoria e Prática.** 2013.
- LIMA JUNIOR, F. P. d. **Sistema embarcado para controle e supervisão de ambiente em aviário utilizando web service.** 2017. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Federal do Rio Grande do Norte, Macaíba, Rio Grande do Norte.
- MACEKOVÁ, L. 1-wire-the technology for sensor networks. **Revista Acta Electrotechnica et Informatica**, v. 12, n. 4, p. 52, 2012.
- MAGRANI, E. **A internet das coisas.** 1. ed.: Rio de Janeiro: Editora FGV, 2018.
- MAXIM INTEGRATED. **1-WIRE COMMUNICATION THROUGH SOFTWARE.** 2012. Disponível em: <https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html>. Acesso em: 21 jan. 2022.
- \_\_\_\_\_. **Programmable Resolution 1-Wire Digital Thermometer.** Jul. 2019. Rev. 6. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Acesso em: 28 out. 2021.
- OLIVEIRA, M. E. d. **Implementação e avaliação de um sistema automatizado de monitoramento e controle térmico em um aviário convencional utilizando tecnologia IoT.** 2019. Tese (Doutor em Ciências da Engenharia Ambiental) – Universidade de São Paulo, São Carlos.
- OLIVEIRA, M. E. d. *et al.* **DESENVOLVIMENTO DE SENsoRES PARA MONITORAMENTO DE AMBIENTE AVIÁRIO COM êNFASE EM CONTROLE TÉRMICO/- DEVELOPMENT OF SENSORS FOR MONITORING THE AVIAN ENVIRONMENT WITH EMPHASIS IN THERMAL CONTROL.** **Revista Brasileira de Engenharia de Biossistemas**, v. 12, n. 3, p. 234–240, 2018.

- OLIVEIRA, R. F. M. d. *et al.* Efeitos da temperatura e da umidade relativa sobre o desempenho e o rendimento de cortes nobres de frangos de corte de 1 a 49 dias de idade. **Revista Brasileira de Zootecnia**, v. 35, p. 797–803, 2006.
- RAMADIANI, R. *et al.* Temperature and humidity control system for broiler chicken coops. **Indonesian Journal of Electrical Engineering e Computer Science**, v. 22, n. 3, p. 1327–1333, 2021.
- RIOS, A. d. A. *et al.* COMPARATIVO ENTRE PLATAFORMAS DE DESENVOLVIMENTO DE SISTEMAS MICROCONTROLADOS PARA APLICAÇÕES EM INTERNET DAS COISAS. In: XVIII CONFERÊNCIA DE ESTUDOS DE ENGENHARIA ELÉTRICA, 2020, Uberlândia, MG, Brasil. **Anais**. Uberlândia, MG: Universidade Federal de Uberlândia, 2020. P. 6.
- RUSH, P. **Proteção e Automação de Redes: conceito e aplicação**. 1. ed.: São Paulo: Editora Blucher, 2011.
- SABBAGH, R. **Scrum: Gestão ágil para projetos de sucesso**. São Paulo: Editora Casa do Código, 2014.
- SACCO, F. **Módulo Tiny RTC 1-Wire – Parte 3**. 2015. Disponível em: <https://www.embarcados.com.br/modulo-tiny-rtc-1-wire-parte-3/>. Acesso em: 21 jan. 2022.
- SADUIKIS, P. **1-Wire Parasitic Power**. 2010. Disponível em: <https://os.mbed.com/users/snatch59/notebook/1-wire-parasitic-power/>. Acesso em: 22 jan. 2022.
- SARTOR, V. *et al.* Sistemas de resfriamento evaporativo e o desempenho de frangos de corte. **Revista Scientia Agricola**, Piracicaba, v. 58, n. 1, p. 17–20, 2001.
- SILVA, J. A. F. d.; LAGO, C. L. d. Módulo eletrônico de controle para válvulas solenóides. **Revista Química Nova**, São Paulo, v. 25, n. 5, p. 842–843, 2002.
- SILVA BRAGA, J. da *et al.* O modelo dos “Cinco Domínios” do bem-estar animal aplicado em sistemas intensivos de produção de bovinos, suíños e aves. **Revista Brasileira de Zoociências**, Juiz de Fora, v. 19, n. 2, p. 204–226, 2018.
- SILVEIRA, C. B. **Como Funciona a Válvula Solenoide e Quais os Tipos?** 2017. Disponível em: <https://www.citisystems.com.br/valvula-solenoide/>. Acesso em: 4 nov. 2021.
- SINCLAIR, B. **IoT: Como Usar a Internet das Coisas para Alavancar seus Negócios**. 1. ed.: São Paulo: Editora Autêntica Business, 2018.
- SOUZA, F. d. **Galpões climatizados: uma ferramenta eficiente aliada à alta produtividade**. 2017. Disponível em: <https://www.aviculturainustrial.com.br/imprensa/galpoes-climatizados-uma-ferramenta-eficiente-aliada-a-alta-produtividade/20170201-144126-c248>. Acesso em: 26 out. 2021.

- SOUZA, P. de. Avicultura e clima quente: Como administrar o bem-estar às aves? **Refresque: Indústria e Comércio Ltda.**, 2005.
- STALLINGS, W. **Arquitetura e Organização de Computadores**. 10. ed.: São Paulo: Editora Pearson Education do Brasil, 2017.
- STROSKI, P. N. **Conheça o relé eletromecânico**. 2018. Disponível em: <https://www.electricallibrary.com/2018/08/06/conheca-o-rele-eletromecanico/>. Acesso em: 21 jan. 2022.
- THOMAZINI, D.; ALBUQUERQUE, P. U. B. d. **Sensores industriais: fundamentos e aplicações**. 9. ed.: Bela Vista: Editora Saraiva Educação, 2020.
- WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Editora Elsevier, 2009.

## 5 IMPLEMENTAÇÃO DO CÓDIGO DO MICROCONTROLADOR ESP32

```

1 //---Bibliotecas---
2 #include <WiFi.h>
3 #include <OneWire.h>
4 #include <DallasTemperature.h>
5 #include <SPIFFS.h>
6 #include "DHT.h"
7 #include "ESPAsyncWebServer.h"
8
9
10 //---Pinos de entrada e saída---
11 #define PIN_DS18B20 2
12 #define PIN_DHT11 4
13 #define PIN_NIPPLE 27
14 #define PIN_FAN 19
15 #define PIN_NEBULIZER 21
16
17 //---Configuração de bibliotecas---
18 #define DHTYPE DHT11
19 DHT dht(PIN_DHT11, DHTYPE);
20 OneWire oneWire(PIN_DS18B20);
21 DallasTemperature ds18b20(&oneWire);
22 AsyncWebServer server(80);
23
24 //---Conexão na rede local sem fio---
25 const char* ssid      = "Bethpsi";
26 const char* password = "raulfabio21";
27
28 //---Variáveis globais---
29 //Valores dos parâmetros
30
31 int parameter_nebulizer_on_humidity = 90;
32 int parameter_nebulizer_off_humidity = 90;
33 float parameter_nebulizer_on_temperature = 20.0;
34 float parameter_nebulizer_off_temperature = 20.0;
35
36 float parameter_nipple_on_deltat = 5.4;
37 int parameter_nipple_off_time = 8000;
38
39 float parameter_fan_on_temperature = 22.2;
40 float parameter_fan_off_temperature = 22.2;
41
42
43 //Valores atuais rebido dos sensores
44 float current_nipple_temp = 0;
45 float current_box_temp = 0;

```

```
46 float current_climate_temp = 0;
47 float current_climate_humidity = 0;
48
49 //Variáveis de tempo para função millis()
50 long update_time = 0;
51 long exchanger_time = 0;
52
53 //Variável para auxiliar desligar ou manter o ventilador ligado após
      desligar o nebulizador
54 bool fan_on_directly = 0;
55
56 //Modos de operação
57 bool operation_mode_nebulizer = false; //false = automático | true = manual
58 bool operation_mode_fan = false; //false = automático | true = manual
59 bool operation_mode_exchanger = true; //false = automático | true = manual
60
61
62
63
64 bool stringToBool(String n){ //qualquer valor diferente de 1 retornará
      false
65   return (n == "1");
66 }
67
68 String getStatusActuator(char n){
69   //'n' para nebulizador; 'e' para o trocador de agua; 'f' para o
      ventilador
70   int pin;
71   switch (n) {
72     case 'n':
73       pin = PIN_NEBULIZER;
74       break;
75     case 'e':
76       pin = PIN_NIPPLE;
77       break;
78     case 'f':
79       pin = PIN_FAN;
80       break;
81   }
82
83   if(digitalRead(pin) == HIGH)
84     return "1";
85   else
86     return "0";
87
88 }
```

```

90 void printActuators(){
91   Serial.println("-----");
92   Serial.print("N: ");
93   Serial.print(getStatusActuator('n'));
94   Serial.print(" ");
95   Serial.println(operation_mode_nebulizer);
96   Serial.print("F: ");
97   Serial.print(getStatusActuator('f'));
98   Serial.print(" ");
99   Serial.println(operation_mode_fan);
100  Serial.print("E: ");
101  Serial.print(getStatusActuator('e'));
102  Serial.print(" ");
103  Serial.println(operation_mode_exchanger);
104  Serial.println("-----");
105 }
106 }
107
108 void setNebulizer(bool state){
109 if (state){
110   Serial.println("ligou");
111   digitalWrite(PIN_NEBULIZER, HIGH);
112   if (digitalRead(PIN_FAN)==HIGH)
113     fan_on_directly = 1;
114   else{
115     fan_on_directly=0;
116     digitalWrite(PIN_FAN, HIGH);
117   }
118 }else{
119   Serial.println("desligou nebulizador");
120   digitalWrite(PIN_NEBULIZER, LOW);
121   Serial.println("desligou nebulizador");
122   if(fan_on_directly == 0)
123     digitalWrite(PIN_FAN, LOW);
124 }
125 printActuators();
126 }
127
128 void setExchanger(bool state, int time_exchange){
129 //Liga trocador de água
130 //timeExchange = 0 quando a duração da troca for da variavel
131 //parameter_nipple_off_time
132 if (state){
133   digitalWrite(PIN_NIPPLE, HIGH);
134   if (time_exchange == 0)
135     exchanger_time = millis() + parameter_nipple_off_time;
136   else

```

```

136     exchanger_time = millis() + time_exchange;
137 }else{
138     digitalWrite(PIN_NIPPLE, LOW);
139 }
140 }
141
142 void automaticModeNebulizer(){
143 if(!operation_mode_nebulizer){
144
145 //Liga Nebulizador
146 if(digitalRead(PIN_NEBULIZER) == LOW){
147     if( (current_climate_humidity < parameter_nebulizer_on_humidity) && (
148         current_climate_temp > parameter_nebulizer_on_temperature) ){
149         setNebulizer(1);
150         Serial.println("Nebulizador ligado via modo automático");
151     }
152
153 //Desliga Nebulizador
154 }else{
155     if( (current_climate_humidity > parameter_nebulizer_off_humidity) ||
156         (current_climate_temp < parameter_nebulizer_off_temperature) ){
157         setNebulizer(0);
158         Serial.println("Nebulizador desligado via modo automático");
159     }
160 }
161
162 void automaticModeExchanger(){
163 if(!operation_mode_exchanger){
164     if( (current_nipple_temp - current_box_temp) >
165         parameter_nipple_on_deltat){
166         Serial.println("Água trocada via modo automático");
167         setExchanger(1, 0);
168     }
169 }
170
171 void automaticModeFan(){
172 if(!operation_mode_fan){ //Liga Ventilador
173     if (digitalRead(PIN_FAN) == LOW){
174         if (current_climate_temp > parameter_fan_on_temperature){
175             digitalWrite(PIN_FAN, HIGH);
176             Serial.println("Ventilador ligado via modo automático");
177         }
178     }else{ //Desliga Ventilador

```

```

179     if ((current_climate_temp < parameter_fan_off_temperature) && (
180         digitalRead(PIN_NEBULIZER) == LOW)) {
181         digitalWrite(PIN_FAN, LOW);
182         Serial.println("Ventilador ligado via modo automático");
183     }
184 }
185 }
186
187 void changeWaterOff() {
188     if(digitalRead(PIN_NIPPLE) == HIGH) && (exchanger_time < millis())){
189         digitalWrite(PIN_NIPPLE, LOW);
190         Serial.println("Trocador de água terminou sua ação!");
191     }
192 }
193
194 void printSensorsValue() {
195     Serial.println("VALORES DOS SENSORES");
196     // Sensores DS18B20
197     Serial.print("Temperatura do bebedouro: ");
198     Serial.print(current_nipple_temp);
199     Serial.println("°C");
200
201     Serial.print("Temperatura da água da caixa: ");
202     Serial.print(current_box_temp);
203     Serial.println("°C");
204
205     //Sensor DHT11
206
207
208     if (isnan(current_climate_temp) || isnan(current_climate_humidity))
209         Serial.println("Falha em ler o sensor DHT11!");
210     else{
211         Serial.print("Temperatura do aviário: ");
212         Serial.print(current_climate_temp);
213         Serial.print(" °C\n");
214         Serial.print("Umidade do aviário: ");
215         Serial.print(current_climate_humidity);
216         Serial.println(" % ");
217         Serial.println();
218     }
219
220 }
221
222
223 void updateSensorsValue() {
224

```

```

225 Serial.println("Atualizando valores dos sensores.");
226 ds18b20.requestTemperatures();
227 current_nipple_temp = ds18b20.getTempCByIndex(0);
228 current_box_temp = ds18b20.getTempCByIndex(1);
229 current_climate_temp = dht.readTemperature();
230 current_climate_humidity = dht.readHumidity();
231 }
232
233 void setup() {
234
235   Serial.begin(9600);
236   pinMode(PIN_NIPPLE, OUTPUT);
237   pinMode(PIN_FAN, OUTPUT);
238   pinMode(PIN_NEBULIZER, OUTPUT);
239   digitalWrite(PIN_NIPPLE, LOW);
240   digitalWrite(PIN_FAN, LOW);
241   digitalWrite(PIN_NEBULIZER, LOW);
242
243   ds18b20.begin();
244   dht.begin();
245
246   Serial.print("Conectando em ");
247   Serial.println(ssid);
248
249   WiFi.begin(ssid, password);
250
251   long time_out_connect = 30000 + millis();
252   while (WiFi.status() != WL_CONNECTED) {
253     delay(500);
254     Serial.print(".");
255     if (time_out_connect > millis()) {
256       WiFi.reconnect();
257       long time_out_connect = 30000 + millis();
258     }
259   }
260   if (!SPIFFS.begin()) {
261     Serial.println("An Error has occurred while mounting SPIFFS");
262     return;
263   }
264
265   Serial.println("");
266   Serial.println("WiFi connected.");
267   Serial.println("IP address: ");
268   Serial.println(WiFi.localIP());
269
270   // URL para raiz (index)
271   server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {

```

```

272     //request->send_P(200, "text/html", index_html, processor);
273     //Send index.htm with default content type
274     request->send(SPIFFS, "/index.html", "text/html");
275 }
276
277 // HTTP basic authentication
278 /*server.on("/login", HTTP_GET, [] (AsyncWebServerRequest *request) {
279     if(!request->authenticate("user", "pass"))
280         return request->requestAuthentication();
281     request->send(200, "text/plain", "Login Success!");
282 }) ;*/
283
284 // --- URL para arquivos estáticos ---
285
286 //CSS e JS pessoais
287 server.on("/styles.css", HTTP_GET, [] (AsyncWebServerRequest *request) {
288     request->send(SPIFFS, "/styles.css", "text/css");
289 });
290 server.on("/scripts.js", HTTP_GET, [] (AsyncWebServerRequest *request) {
291     request->send(SPIFFS, "/scripts.js", "text/javascript");
292 });
293
294 //Bootstrap
295 server.on("/bootstrap/bootstrap.min.css", HTTP_GET, [] (
296     AsyncWebServerRequest *request) {
297     request->send(SPIFFS, "/bootstrap/bootstrap.min.css", "text/css");
298 });
299 server.on("/bootstrap/bootstrap.min.js", HTTP_GET, [] (
300     AsyncWebServerRequest *request) {
301     request->send(SPIFFS, "/bootstrap/bootstrap.min.js", "text/javascript")
302     ;
303 });
304
305 //Imagens
306 server.on("/favicon.ico", HTTP_GET, [] (AsyncWebServerRequest *request) {
307     request->send(SPIFFS, "/img/favicon.ico", "image/png");
308 });
309
310 server.on("/img/logo.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
311     request->send(SPIFFS, "/img/logo.png", "image/png");
312 });
313
314 server.on("/img/air_temperature.png", HTTP_GET, [] (AsyncWebServerRequest
315     *request) {
316     request->send(SPIFFS, "/img/air_temperature.png", "image/png");
317 });

```

```

315 server.on("/img/exchanger.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
316     request->send(SPIFFS, "/img/exchanger.png", "image/png");
317 });
318
319 server.on("/img/box_temperature.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
320     request->send(SPIFFS, "/img/box_temperature.png", "image/png");
321 });
322
323 server.on("/img/fan.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
324     request->send(SPIFFS, "/img/fan.png", "image/png");
325 });
326
327 server.on("/img/humidity.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
328     request->send(SPIFFS, "/img/humidity.png", "image/png");
329 });
330
331 server.on("/img/nebulizer.png", HTTP_GET, [] (AsyncWebServerRequest *request) {
332     request->send(SPIFFS, "/img/nebulizer.png", "image/png");
333 });
334
335 server.on("/img/nipple_temperature.png", HTTP_GET, [] (
336     AsyncWebServerRequest *request) {
337     request->send(SPIFFS, "/img/nipple_temperature.png", "image/png");
338 });
339
340
341
342 //---Rotas para requisições de origem da Página Web---
343 //URL para requisitar a temperatura temperatura em c_str, porque deve
344 //ser mandado em vetor de char request é um ponteiro
345 //request é um ponteiro que aponta pra qual tipo de requisição vai ser
346 //feita send_P é para enviar uma pagina web grande send é para
347 //respostas simples
348
349 //URL para requisitar o valor atual de cada sensor
350 server.on("/sensors/getall", HTTP_GET, [] (AsyncWebServerRequest *request) {
351
352     String response = String(current_climate_temp) + " " + String(
353         current_climate_humidity) + " " + String(current_box_temp) + " " +
354         String(current_nipple_temp);
355     request->send(200, "text/plain", response.c_str());
356 });

```

```

351
352 //URL para acionar/desligar o nebulizador
353 server.on(
354     "/actuators/status/set/nebulizer",
355     HTTP_POST,
356     [] (AsyncWebServerRequest * request) {
357         int params = request->params();
358         for (int i = 0; i < params; i++) {
359             AsyncWebParameter *p = request->getParam(i);
360
361             if (p->name() == "value") {
362                 setNebulizer(stringToBool(p->value()));
363                 request->send(200);
364                 Serial.println("Parâmetro de acionamento do ventilador modificado
365                               via interface Web!");
366                 Serial.println(stringToBool(p->value()));
367             }
368         }
369     });
370
371
372 //URL para ligar o trocador de agua
373 server.on(
374     "/actuators/status/set/exchanger",
375     HTTP_POST,
376     [] (AsyncWebServerRequest * request) {
377         int params = request->params();
378         bool flagState, flagTime, currentState;
379         int currentTime = 0;
380         for (int i = 0; i < params; i++) {
381             AsyncWebParameter *p = request->getParam(i);
382
383             if (p->name() == "value") {
384                 flagState = true;
385                 currentState = (stringToBool(p->value()));
386             }
387             if (p->name() == "time") {
388                 if (currentTime = p->value().toInt())
389                     flagTime = true;
390             }
391         }
392         if (flagState) {
393             if (currentState) {
394                 if (flagState) {
395                     setExchanger(currentState, currentTime * 1000);

```

```

397         request->send(200);
398     }else{
399         request->send(304);
400     }
401 }else{
402     setExchanger(currentState, 0);
403     request->send(200);
404 }
405 }else{
406     request->send(304);
407 }
408 });
409
410 //URL para ligar/desligar o ventilador
411 server.on(
412     "/actuators/status/set/fan",
413     HTTP_POST,
414     [] (AsyncWebServerRequest * request) {
415         int params = request->params();
416         for (int i = 0; i < params; i++) {
417             AsyncWebParameter *p = request->getParam(i);

418             if (p->name() == "value") {
419                 if (stringToBool(p->value())) {
420                     digitalWrite(PIN_FAN, HIGH);
421                     request->send(200);
422                     Serial.println("Parâmetro de acionamento do ventilador modificado
423                                     via interface Web!");
424                 }else{
425                     if(digitalRead(PIN_NEBULIZER)==HIGH) {
426                         request->send(409, "text/plain", String("Ventilador não pode
427                                         ser desligado quando o nebulizador estiver em funcionamento!
428                                         ").c_str());
429                     }else{
430                         digitalWrite(PIN_FAN, LOW);
431                         request->send(200);
432                         Serial.println("Parâmetro de acionamento do ventilador
433                                         modificado via interface Web!");
434                     }
435                 }
436             }
437         }
438     });
439
440 //URL para capturar o status atual de cada atuador
441 server.on("/actuators/status/getall", HTTP_GET, [] (AsyncWebServerRequest
442             *request) {

```

```

439     String response = String(getStatusActuator('e')) + " " + String(
440         getStatusActuator('f')) + " " + String( getStatusActuator('n'));
441     request->send(200, "text/plain", response.c_str());
442 );
443
444 //URL para capturar o valor atual de cada parâmetro de acionamento
445 server.on("/actuators/parameter/getall", HTTP_GET, [](
446     AsyncWebServerRequest *request) {
447
448     String response = String(parameter_nebulizer_on_humidity) + " " + String(
449         parameter_nebulizer_off_humidity) + " " + String(
450             parameter_nebulizer_on_temperature)
451         + " " + String(parameter_nebulizer_off_temperature) + " " + String(
452             parameter_nipple_on_deltat) +
453             " " + String(parameter_nipple_off_time / 1000) + " " + String(
454                 parameter_fan_on_temperature) + " " + String(
455                     parameter_fan_off_temperature);
456
457     request->send(200, "text/plain", response.c_str());
458 );
459
460 //URL para capturar o modo de operação de cada atuador
461 server.on("/actuators/opmode/getall", HTTP_GET, [] (AsyncWebServerRequest
462     *request) {
463
464     String response = String(operation_mode_nebulizer) + " " + String(
465         operation_mode_exchanger) + " " + String(operation_mode_fan);
466     request->send(200, "text/plain", response.c_str());
467 );
468
469 //URL para modificar o modo de operação do nebulizador
470 server.on(
471     "/actuators/opmode/set/nebulizer",
472     HTTP_POST,
473     [] (AsyncWebServerRequest * request) {
474
475     int params = request->params();
476     for (int i = 0; i < params; i++) {
477
478         AsyncWebParameter *p = request->getParam(i);
479
480         if (p->name() == "value") {
481
482             operation_mode_nebulizer = stringToBool(p->value());
483             Serial.println(stringToBool(p->value()));
484             request->send(200);
485             Serial.println("Nebulizador trocou seu modo de operação!");
486         }
487     }
488 };
489
490 //URL para modificar o modo de operação do trocador de água

```

```

477 server.on(
478     "/actuators/opmode/set/exchanger",
479     HTTP_POST,
480     [] (AsyncWebServerRequest * request) {
481         int params = request->params();
482         for (int i = 0; i < params; i++) {
483             AsyncWebParameter *p = request->getParam(i);
484
485             if (p->name() == "value") {
486                 operation_mode_exchanger = stringToBool(p->value());
487                 request->send(200);
488                 Serial.println("Trocador trocou seu modo de operação!");
489             }
490         }
491     });
492
493 //URL para modificar o modo de operação do ventilador
494 server.on(
495     "/actuators/opmode/set/fan",
496     HTTP_POST,
497     [] (AsyncWebServerRequest * request) {
498         int params = request->params();
499         for (int i = 0; i < params; i++) {
500             AsyncWebParameter *p = request->getParam(i);
501
502             if (p->name() == "value") {
503                 operation_mode_fan = stringToBool(p->value());
504                 request->send(200);
505                 Serial.println("Ventilador trocou seu modo de operação!");
506             }
507         }
508     });
509
510 //URL para modificar o parâmetro de temperatura do ventilador
511 server.on(
512     "/actuators/parameter/fan/on/temp",
513     HTTP_POST,
514     [] (AsyncWebServerRequest * request) {
515         int params = request->params();
516         for (int i = 0; i < params; i++) {
517             AsyncWebParameter *p = request->getParam(i);
518
519             if (p->name() == "value") {
520
521                 if (float current_value = (p->value().toFloat())) {
522                     parameter_fan_on_temperature = current_value;
523                     request->send(200);

```

```

524     Serial.println("Parâmetro de acionamento do ventilador
525         modificado via interface Web!");
526     }else{
527         Serial.println("Erro em modificar o parâmetro de acionamento do
528             ventilador via interface Web!");
529             request->send(304);
530         }
531     });
532 }
533 server.on(
534     "/actuators/parameter/fan/off/temp",
535     HTTP_POST,
536     [] (AsyncWebServerRequest * request){
537         int params = request->params();
538         for (int i = 0; i < params; i++){
539             AsyncWebParameter *p = request->getParam(i);
540
541             if (p->name() == "value") {
542
543                 if (float current_value = (p->value().toFloat())){
544                     parameter_fan_off_temperature = current_value;
545                     request->send(200);
546                     Serial.println("Parâmetro de acionamento do ventilador
547                         modificado via interface Web!");
548                 }else{
549                     Serial.println("Erro em modificar o parâmetro de acionamento do
550                         ventilador via interface Web!");
551                     request->send(304);
552                 }
553             }
554         });
555 //URL para modificar o parâmetro de variação da temperatura do trocador
556 //de água
557 server.on(
558     "/actuators/parameter/exchanger/on/deltatemp",
559     HTTP_POST,
560     [] (AsyncWebServerRequest * request){
561         int params = request->params();
562         for (int i = 0; i < params; i++){
563             AsyncWebParameter *p = request->getParam(i);
564
565             if (p->name() == "value") {
566                 if (float current_value = (p->value().toFloat())){

```

```

566     parameter_nipple_on_deltat = current_value;
567     request->send(200);
568     Serial.println("Parâmetro de acionamento do trocador modificado
569     via interface Web!");
570 }else{
571     request->send(304);
572     Serial.println("Erro em modificar o parâmetro de acionamento do
573     trocador via interface Web!");
574 }
575 });
576
577 //URL para modificar o parâmetro de variação da temperatura do trocador
578 //de água
579 server.on(
580     "/actuators/parameter/exchanger/off/time",
581     HTTP_POST,
582     [] (AsyncWebServerRequest * request) {
583     int params = request->params();
584     for (int i = 0; i < params; i++) {
585         AsyncWebParameter *p = request->getParam(i);
586
587         if(p->name() == "value"){
588             if(float current_value = (p->value().toFloat())){
589                 parameter_nipple_off_time = current_value * 1000;
590                 request->send(200);
591                 Serial.println("Parâmetro de acionamento do trocador modificado
592                 via interface Web!");
593             }else{
594                 request->send(304);
595                 Serial.println("Erro em modificar o parâmetro de acionamento do
596                 trocador via interface Web!");
597             }
598         }
599     });
600
601 //URL para modificar o parâmetro de umidade do nebulizador
602 server.on(
603     "/actuators/parameter/nebulizer/on/humidity",
604     HTTP_POST,
605     [] (AsyncWebServerRequest * request) {
606     int params = request->params();
607     for (int i = 0; i < params; i++) {
608         AsyncWebParameter *p = request->getParam(i);

```

```

608     if (p->name () == "value") {
609         if (float current_value = (p->value ().toFloat ())){
610             parameter_nebulizer_on_humidity = current_value;
611             request->send(200);
612             Serial.println("Parâmetro de umidade de acionamento do
613                             nebulizador modificado via interface Web!");
614         }else{
615             request->send(304);
616             Serial.println("Erro em modificar o parâmetro de acionamento do
617                             nebulizador via interface Web!");
618         }
619     });
620
621 //URL para modificar o parâmetro de umidade do nebulizador
622 server.on(
623     "/actuators/parameter/nebulizer/off/humidity",
624     HTTP_POST,
625     [] (AsyncWebServerRequest * request){
626         int params = request->params ();
627         for (int i = 0; i < params; i++) {
628             AsyncWebParameter *p = request->getParam(i);
629
630             if (p->name () == "value") {
631                 if (float current_value = (p->value ().toFloat ())){
632                     parameter_nebulizer_off_humidity = current_value;
633                     request->send(200);
634                     Serial.println("Parâmetro de umidade de acionamento do
635                             nebulizador modificado via interface Web!");
636                 }else{
637                     request->send(304);
638                     Serial.println("Erro em modificar o parâmetro de acionamento do
639                             nebulizador via interface Web!");
640                 }
641             });
642
643 //URL para modificar o parâmetro de temperatura do nebulizador
644 server.on(
645     "/actuators/parameter/nebulizer/on/temp",
646     HTTP_POST,
647     [] (AsyncWebServerRequest * request){
648         int params = request->params ();
649         for (int i = 0; i < params; i++) {
650             AsyncWebParameter *p = request->getParam(i);

```

```

651     if(p->name() == "value") {
652         if(float current_value = (p->value().toFloat())){
653             parameter_nebulizer_on_temperature = current_value;
654             request->send(200);
655             Serial.println("Parâmetro de temperatura de acionamento do
656                             nebulizador modificado via interface Web!");
657         }else{
658             Serial.println("Erro em modificar o parâmetro de acionamento do
659                             ventilador via interface Web!");
660             request->send(304);
661         }
662     });
663
664 //URL para modificar o parâmetro de temperatura do nebulizador
665 server.on(
666     "/actuators/parameter/nebulizer/off/temp",
667     HTTP_POST,
668     [] (AsyncWebServerRequest * request){
669         int params = request->params();
670         for (int i = 0; i < params; i++) {
671             AsyncWebParameter *p = request->getParam(i);
672             if(p->name() == "value") {
673                 if(float current_value = (p->value().toFloat())){
674                     parameter_nebulizer_off_temperature = current_value;
675                     request->send(200);
676                     Serial.println("Parâmetro de temperatura de acionamento do
677                             nebulizador modificado via interface Web!");
678                 }else{
679                     Serial.println("Erro em modificar o parâmetro de acionamento do
680                             ventilador via interface Web!");
681                     request->send(304);
682                 }
683             });
684
685         server.begin();
686     }
687 }
688
689 void loop() {
690
691     if (update_time < millis()){
692         printActuators();
693         updateSensorsValue();

```

```
694     printSensorsValue();  
695  
696     automaticModeExchanger();  
697     automaticModeFan();  
698     automaticModeNebulizer();  
699  
700     update_time = millis() + 5000;  
701 }  
702  
703 changeWaterOff();  
704  
705 }
```

## 6 IMPLEMENTAÇÃO DO CÓDIGO HTML DA PÁGINA WEB

```

1 <!DOCTYPE HTML>
2 <html lang="pt-BR">
3 <!-- Required meta tags -->
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6
7 <!-- Bootstrap CSS -->
8 <link href="/bootstrap/bootstrap.min.css" rel="stylesheet"
9   integrity="sha384-EVSTQN3+azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWspd3yD65VohhpuuC0mLASjC" crossorigin=
  "anonymous">
10
11 <link rel="stylesheet" href="/styles.css">
12
13 <title>Controle Aviário</title>
14 </head>
15
16 <body>
17
18   <nav id="navComputer" class="d-block navbar navbar-expand-sm navbar-light
      bg-warning">
19     <div class="container-fluid">
20       <a class="navbar-brand" href="/">
21         
22       </a>
23       <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
          " data-bs-target="#navbarSupportedContent"
          aria-controls="navbarSupportedContent" aria-expanded="false"
          aria-label="Toggle navigation">
24         <span class="navbar-toggler-icon"></span>
25       </button>
26       <div class="collapse navbar-collapse" id="navbarSupportedContent">
27         <ul class="navbar-nav me-auto mb-2 mb-lg-0">
28           <li class="nav-item">
29             <a class="nav-link active" aria-current="page" href="/">
30               MONITORAMENTO E CONTROLE DO AMBIENTE AVIÁRIO</a>
31           </li>
32
33         </ul>
34         <ul class="navbar-nav ml-auto">
35           <li class="nav-item">
36             <a class="nav-link" href="/login.php">SOBRE</a>
37           </li>
38         </ul>
39     </div>

```

```

40 </nav>
41
42
43
44
45
46 <div class="container-fluid d-block">
47   <br>
48   <div id="sensors" class="container-fluid">
49     <p class="h2 text-center text-uppercase">sensores
50
51     <svg xmlns="http://www.w3.org/2000/svg" width="40" fill="
52       currentColor" class="bi bi-broadcast"
53       viewBox="0 0 20 20">
54       <path
55         d="M3.05 3.05a7 7 0 0 0 0 9.9.5.5 0 0 1-.707.707 8 8 0 0 1 0-11
56           .314.5.5 0 0 1 .707.707zm2.122 2.122a4 4 0 0 0 0 5.656.5.5 0
57           1 1-.708.708 5 5 0 0 1 0-7.072.5.5 0 0 1 .708.708zm5.656-
58           .708a.5.5 0 0 1 .708 0 5 5 0 0 1 0 7.072.5.5 0 1 1-.708-.708
59           4 4 0 0 0 0-5.656.5.5 0 0 1 0-.708zm2.122-2.12a.5.5 0 0 1
60           .707 0 8 8 0 0 1 0 11.313.5.5 0 0 1-.707-.707 7 7 0 0 0 0-9
61           .9.5.5 0 0 1 0-.707zM10 8a2 2 0 1 1-4 0 2 2 0 0 1 4 0z" />
62   </svg>
63 </p>
64 <div class="row border border-4 border-dark">
65   <!-- Sensores expandidos -->
66   <div id="expanded_climate_temperature"
67     class="d-lg-flex d-none col-12 col-lg-3 d-flex flex-column
68       border-end border-4 border-dark border-bottom-lg p-1">
69     <p class="h5 text-center text-uppercase">Temperatura Ambiente</
70       p>
71     <div class="mt-auto">
72       <p class="h5 text-center text-uppercase">
73         <span class="fw-bold" id="climate_temperature2" style="color: #000000;">---

```

```

74   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
75     " fill="currentColor"
76     class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
77     >
78     <path
79       d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
80         -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
81         0 0 1-1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
82         .70714.096 4.096a.5.5 0 1 1-.707.707z" />
83   </svg>
84 </button>
85
86 </div>
87 <div id="expanded_climate_humidity">
88   <div class="d-lg-flex d-none col-12 col-lg-3 d-flex flex-column
89     border-end border-4 border-dark border-bottom-lg
90     border-end-lg-0 position-relative p-1">
91     <p class="h5 text-center text-uppercase">Umidade relativa
92     </p>
93     <div class="mt-auto">
94       <p class="h5 text-center text-uppercase">
95         <span class="fw-bold" id="climate_humidity2"> --- </span>
96       </p>
97       
99     </div>
100
101   <button id="expanded_climate_humidity_button" type="button"
102     onclick='retractDiv("expanded_climate_humidity", "actuators"
103     )'
104     class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1 d-none
105     "> Recolher Item
106   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
107     " fill="currentColor"
108     class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
109     >
110     <path
111       d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
112         -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
113         0 0 1-1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
114         .70714.096 4.096a.5.5 0 1 1-.707.707z" />
115   </svg>
116 </button>
117 </div>
118 <div id="expanded_nipple_temperature">
119   <div class="d-lg-flex d-none col-12 col-lg-3 d-flex flex-column
120     border-end border-4 border-dark position-relative p-1">

```

```

105 <p class="h5 text-center text-uppercase">Temperatura da Água do
106   Bebedouro
107 </p>
108
109 <div class="mt-auto">
110   <p class="h5 text-center text-uppercase">
111     <span class="fw-bold" id="nipple_temperature2"> --- </span>
112   </p>
113   
115 </div>
116
117 <button id="expanded_nipple_temperature_button" type="button"
118   onclick='retractDiv("expanded_nipple_temperature", "actuators")'
119   class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1 d-none"
120     > Recolher Item
121   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
122     fill="currentColor"
123     class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
124       >
125     <path
126       d="M2 0a2 2 0 0 0-2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
127         -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
128           0 0 1-1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
129             .70714.096 4.096a.5.5 0 1 1-.707.707z" />
130   </svg>
131 </button>
132 </div>
133 <div id="expanded_box_temperature">
134   <div class="d-lg-flex d-none col-12 col-lg-3 d-flex flex-column
135     position-relative p-1">
136     <p class="h5 text-center text-uppercase">Temperatura da Água da
137       Caixa
138     </p>
139     <div class="mt-auto">
140       <p class="h5 text-center text-uppercase">
141         <span class="fw-bold" id="box_temperature2"> --- </span>
142       </p>
143       
145     </div>
146
147     <button id="expanded_box_temperature_button" type="button"
148       onclick='retractDiv("expanded_box_temperature", "actuators")'
149         '

```

```

138   class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1 d-none"
139     "> Recolher Item
140   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
141     " fill="currentColor"
142     class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
143     ">
144   <path
145     d="M2 0a2 2 0 0 0-2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
146       -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
147       0 0 1-1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
148       .70714.096 4.096a.5.5 0 1 1-.707.707z" />
149   </svg>
150 </button>
151 </div>
152
153 <!-- Sensores recolhidos -->
154 <div
155   class="item-mobile d-lg-none col-6 col-lg-3 d-flex
156     flex-column border-end border-4 border-dark
157     border-bottom-lg position-relative p-1">
158   <p class="h6 text-center text-uppercase">Temperatura Ambiente
159     <br> <span class="fw-bold"
160       id="climate_temperature1"> --- </span></p>
161   <button type="button" onclick='expandDiv("expanded_climate_temperature", "actuators")'
162     class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1">
163     Expandir item
164   <svg xmlns="http://www.w3.org/2000/svg" width="16" fill=
165     "currentColor"
166     class=" bi bi-arrow-down-right-square-fill" viewBox="0 0
167       16 16">
168   <path
169     d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2
170       v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
171       .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
172       0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
173   </svg>
174 </button>
175 </div>
176 <div
177   class="item-mobile d-block d-lg-none col-6 col-lg-3 d-flex
178     flex-column border-end border-4 border-dark
179     border-bottom-lg border-end-lg-0 position-relative p-1">
180   <p class="h6 text-center text-uppercase">Umidade relativa <br
181     >
182   <span class="fw-bold" id="climate_humidity1"> --- </span>
```

```

166 </p>
167 <button type="button" onclick='expandDiv("expanded_climate_humidity", "actuators")' class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1">
168   Expandir item
169   <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="currentColor"
170     class="bi bi-arrow-down-right-square-fill" viewBox="0 0 16 16">
171     <path
172       d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2 2
173         v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
174           .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
175             0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
176   </svg>
177 </button>
178 </div>
179 <div
180   class="item-mobile d-block d-lg-none col-6 col-lg-3 d-flex
181     flex-column border-end border-4 border-dark
182     position-relative p-1">
183   <p class="h6 text-center text-uppercase">Temperatura da Água
184     do Bebedouro <br>
185     <span class="fw-bold" id="nipple_temperature1"> --- </span>
186   </p>
187   <button type="button" onclick='expandDiv("expanded_nipple_temperature", "actuators")' class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1">
188     Expandir item
189     <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="currentColor"
190       class="bi bi-arrow-down-right-square-fill" viewBox="0 0 16 16">
191       <path
192         d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2 2
193           v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
194             .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
195               0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
196   </svg>
197 </button>
198 </div>
199 <div class="item-mobile d-block d-lg-none col-6 col-lg-3 d-flex
200   flex-column position-relative p-1">
201   <p class="h6 text-center text-uppercase">Temperatura da Água
202     da Caixa <br>
203     <span class="fw-bold" id="box_temperature1"> --- </span>
204   </p>

```

```

194   <button type="button" onclick='expandDiv("expanded_box_temperature", "actuators")' class="btn btn-outline-dark btn-sm mt-auto ms-auto p-1">
195     Expandir item
196   <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="currentColor" class="bi bi-arrow-down-right-square-fill" viewBox="0 0 16 16">
197     <path d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0 0-2-2H2a2 2 0 0 0 0-2 2
198       v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
199         .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
200         0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
201   </svg>
202   </button>
203 </div>
204 </div>
205
206 <br> <br>
207 <div id="actuators" class="container-fluid align-bottom">
208   <p id="actuators" class="h2 text-center text-uppercase"> atuadores
209
210
211   <svg xmlns="http://www.w3.org/2000/svg" width="28" fill="currentColor" class="bi bi-toggles2" viewBox="0 0 20 20">
212     <path d="M9.465 10H12a2 2 0 1 1 0 4H9.465c.34-.588-.535-1.271-.535
213       -2 0-.729-.195-1.412-.535-2z" />
214     <path d="M6 15a3 3 0 1 0 0-6 3 3 0 0 0 6zm0 1a4 4 0 1 1 0-8 4 4 0 0
215       1 0 8zm.535-10a3.975 3.975 0 0 1-.409-1H4a1 1 0 0 1 0-2h2
216       .126c.091-.355.23-.69.41-1H4a2 2 0 1 0 0 4h2.535z" />
217     <path d="M14 4a4 4 0 1 1-8 0 4 4 0 0 1 8 0z" />
218   </svg>
219 </p>
220 <div class="row border border-4 border-dark">
221   <!-- Atuadores expandidos -->
222   <div id="expanded_nebulizer" class="d-lg-flex d-none col-12 col-lg-4 d-flex flex-column border-end border-4 border-dark px-0 pt-2">
223     <p class="h5 text-center text-uppercase">Nebulizador</p>
224     
225     <p class="text-center">
226       <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="currentColor"

```

```

227     class="bi bi-play-fill pb-1 negative-me" viewBox="0 0 16 16"
228         ">
229     <path
230         d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-1.233-
231             .697V4.308c0-.63.692-1.01 1.233-.696l6.363 3.692a
232                 .802.802 0 0 1 0 1.393z" />
233     </svg>
234     <span class="fw-bold">Status:</span> <span class="fw-bold
235         text-uppercase" id="status_nebulizer1">
236         --- </span>
237     </p>
238
239
240     <p class=" text-center mb-0">
241         <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16
242             " fill="currentColor"
243             class"bi bi-wrench-adjustable-circle-fill" viewBox="0 0 16
244                 16">
245         <path
246             d="M6.705 8.139a.25.25 0 0 0-.288-.376l-1.5.5.159.474.808
247                 -.27-.595.894a.25.25 0 0 0 .287.376l.808-.27-.595.894a
248                 .25.25 0 0 0 .287.376l1.5-.5-.159-.474-.808.27.596-
249                 .894a.25.25 0 0 0-.288-.376l-.808.27.596-.894z" />
250         <path
251             d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16Zm-6.202-4.751 1.988
252                 -1.657a4.5 4.5 0 0 1 7.537-4.623L7.497 6.511 2.5 1.333
253                 3.11c-.56.251-1.18.39-1.833.39a4.49 4.49 0 0 1-1.592-
254                 .29L4.747 14.2a7.031 7.031 0 0 1-2.949-2.951zM12.496 8
255                 a4.491 4.491 0 0 1-1.703 3.526L9.497 8.512.959-1.11c
                     .027.2.04.403.04.61Z" />
256     </svg>
257
258
259     <span class="fw-bold"> Modo de operação </span>
260     </p>
261     <div class="switch-button my-0">
262         <input class="switch-button-checkbox" type="checkbox" id="
263             op_mode_nebulizer"
264             onclick='buttonSetOpModeActuator(this.id, "nebulizer")'></i
265             nput>
266         <label class="switch-button-label" for=""><span class="
267             switch-button-label-span">Automático</span></label>
268     </div>
269
270
271     <hr class="mt-2 mb-1">
272
273
274     <div>
```

```

256 <div id="automatic_nebulizer" class="text-center d-flex
257   flex-column">
258
259
260   <p class="m-1">
261     <svg xmlns="http://www.w3.org/2000/svg" width="16"
262       height="16" fill="currentColor"
263       class="bi bi-sliders" viewBox="0 0 16 16">
264       <path fill-rule="evenodd"
265         d="M11.5 2a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0 0-3zM9.05
266           3a2.5 2.5 0 0 1 4.9 0H16v1h-2.05a2.5 2.5 0 0 1-4.9
267           0H0V3h9.05zM4.5 7a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0
268           0-3zM2.05 8a2.5 2.5 0 0 1 4.9 0H16v1H6.95a2.5 2.5
269           0 0 1-4.9 0H0V8h2.05zm9.45 4a1.5 1.5 0 1 0 0 3
270           1.5 1.5 0 0 0 0-3zm-2.45 1a2.5 2.5 0 0 1 4.9 0
271           H16v1h-2.05a2.5 2.5 0 0 1-4.9 0H0v-1h9.05z" />
272     </svg>
273     <span class="fw-bold"> Parâmetros do modo automático</s
274       pan>
275   </p>
276   <div class="d-flex flex-row ps-2 ">
277     <div class="form-check form-check-inline me-4">
278       <input class="form-check-input" type="radio" name="
279         flexRadioNebulizer" id="param_on_nebulizer"
280         onclick="showHideDivParam(1, 'nebulizer')" checked>
281       <label class="form-check-label text-start negative-ms"
282         for="param_on_nebulizer">
283         Ligar quando:
284       </label>
285     </div>
286     <div class="form-check form-check-inline me-1 ms-auto">
287       <input class="form-check-input" type="radio" name="
288         flexRadioNebulizer" id="param_off_nebulizer"
289         onclick="showHideDivParam(0, 'nebulizer')">
290       <label class="form-check-label text-start negative-ms"
291         for="param_off_nebulizer">
292         Desligar quando:
293       </label>
294     </div>
295   </div>
296
297   <div id="card_param_nebulizer" class="flip-container mt-3">
298     <div class="flipper">
299       <div id="card_front_nebulizer" class="front">

```

```
290 <div class="row px-2">
291   <div class="row">
292     <div class="col-9">
293       <label for="nebulizer_parameter_on_humidity"
294         class="col-form-label col-form-label-sm pb-0">
295         "A
296         umidade relativa do ar estiver menor que</
297         label>
298       </div>
299     </div>
300
301     <div class="col-3 pe-0">
302       <input type="number" class="form-control
303         form-control-sm"
304         id="nebulizer_parameter_on_humidity">
305     </div>
306     <div class="col-9 ps-1 pe-2 d-flex flex-row
307       align-items-center ">
308       <span class="font-size-sm ">% </span>
309       <div class="ms-auto">
310         <button type="submit" class="btn px-1
311           font-size-sm bg-warning"
312           onclick='buttonSetParameter("nebulizer/on/
313             humidity", "
314             nebulizer_parameter_on_humidity")'>Salvar<
315             /button>
316       </div>
317     </div>
318   </div>
319
320   <div class="col-12 pe-0">
321     <label class="col-form-label col-form-label-sm
322       pb-0 fw-bold">e</label>
323   </div>
324
325
326
327   <div class="row px-2 mt-1">
328     <div class="row">
329       <div class="col-9">
330         <label for="nebulizer_parameter_on_temperature"
331           class="col-form-label col-form-label-sm pb-0">
332           "A temperatura ambiente estiver maior que<
333           /label>
334         </div>
335       </div>
336     </div>
337   </div>
```

```
325
326     <div class="col-3 pe-0">
327
328         <input type="number" class="form-control
329             form-control-sm"
330             id="nebulizer_parameter_on_temperature">
331     </div>
332     <div class="col-9 ps-1 pe-2 d-flex flex-row
333             align-items-center ">
334         <span class="font-size-sm ">°C </span>
335         <div class="ms-auto">
336             <button type="submit" class="btn px-1
337                 font-size-sm bg-warning mb-2"
338                 onclick='buttonSetParameter("nebulizer/on/
339                     temperature", "
340                     nebulizer_parameter_on_temperature") '>
341                     Salvar</button>
342             </div>
343         </div>
344     </div>
345
346     </div>
347
348
349
350
351     <div id="card_back_nebulizer" class="back hidden-face">
352         <div class="row px-2">
353             <div class="row">
354                 <div class="col-9">
355                     <label for="nebulizer_parameter_off_humidity"
356                         class="col-form-label col-form-label-sm pb-0"
357                         >A umidade relativa do ar estiver maior
358                         que</label>
359                 </div>
360             </div>
361         </div>
362
363
364
365         <div class="col-3 pe-0">
366             <input type="number" class="form-control
367                 form-control-sm"
368                 id="nebulizer_parameter_off_humidity">
369         </div>
370         <div class="col-9 ps-1 pe-2 d-flex flex-row
371             align-items-center ">
372             <span class="font-size-sm ">% </span>
373             <div class="ms-auto">
374                 <button type="submit"
375                     onclick='buttonSetParameter("nebulizer/off/
376                         humidity", "
377                         nebulizer_parameter_off_humidity") '>
```

```

360           class="btn px-1 font-size-sm bg-warning">
361             Salvar</button>
362         </div>
363     </div>
364
365     <div class="col-12 pe-0">
366       <label classclass="row px-2 mt-1">
372     <div class="row">
373       <div class="col-9">
374         <label for="nebulizer_parameter_off_temperature
375           "
376           class="col-form-label col-form-label-sm pb-0"
377             >A temperatura ambiente estiver menor que<
378           /label>
379         </div>
380       </div>
381
382     <div class="col-3 pe-0">
383
384       <input type="number" class="form-control
385         form-control-sm"
386         id="nebulizer_parameter_off_temperature">
387     </div>
388     <div class="col-9 ps-1 pe-2 d-flex flex-row
389       align-items-center ">
390       <span class="font-size-sm ">°C </span>
391       <div class="ms-auto">
392         <button type="submit" class="btn px-1
393           font-size-sm bg-warning mb-2"
394           onclick='buttonSetParameter("nebulizer/off/
395             temperature", "
396             nebulizer_parameter_off_temperature")'>
397             Salvar</button>
398         </div>
399       </div>
400     </div>
401   </div>
402
403 </div>

```

```

396   </div>
397
398
399
400   </div>
401
402   <div id="manual_nebulizer" class="text-center d-none
403     flex-column">
404     <p class="m-1">
405       <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="
406         currentColor"
407         class="bi bi-play-fill pb-1 negative-me" viewBox="0 0
408           16 16">
409         <path
410           d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-1
411             .233-.697v4.308c0-.63.692-1.01 1.233-.696l6.363
412               3.692a.802.802 0 0 1 0 1.393z" />
413       </svg>
414       <span class="fw-bold"> Acionamento do atuador</span>
415     </p>
416     <div class="">
417       <button id="button_act_nebulizer" value="0" type="submit"
418         class="btn bg-warning btn-lg mt-2 mb-2"
419         onclick='buttonSetStatusActuator(this.value, "nebulizer
420           ")'>Desligar</button>
421     </div>
422   </div>
423
424   <button id="expanded_nebulizer_button" type="button" onclick='
425     retractDiv("expanded_nebulizer", "sensors")'
426     class="btn btn-outline-dark btn-sm d-none p-1 mt-5 ms-auto
427       mb-2 me-2"> Recolher Item
428   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
429     fill="currentColor"
430     class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
431     >
432     <path
433       d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
434             -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
435               0 0 1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
436                 .70714.096 4.096a.5.5 0 1 1-.707.707z" />
437   </svg>
438   </button>
439 </div>
440
441 <div id="expanded_exchanger"

```

```

428     class="d-lg-flex d-none col-12 col-lg-4 d-flex flex-column
429         border-end border-4 border-dark px-0 pt-2">
430     <p class="h5 text-center text-uppercase">Trocador de Água</p>
431     
432     <p class=" text-center">
433         <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="
434             currentColor"
435             class="bi bi-play-fill pb-1 negative-me" viewBox="0 0 16 16
436             ">
437             <path
438                 d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-1.233-
439                 .697V4.308c0-.63.692-1.01 1.233-.696l6.363 3.692a
440                 .802.802 0 0 1 0 1.393z" />
441         </svg>
442         <span class="fw-bold">Status:</span> <span class="fw-bold
443             text-uppercase"
444             id="status_exchanger1">---</span>
445         </p>
446
447         <p class=" text-center mb-0">
448             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16
449                 " fill="currentColor"
450                 class="bi bi-wrench-adjustable-circle-fill" viewBox="0 0 16
451                 16">
452                 <path
453                     d="M6.705 8.139a.25.25 0 0 0-.288-.376l-1.5.5.159.474.808
454                     -.27-.595.894a.25.25 0 0 0 .287.376l.808-.27-.595.894a
455                     .25.25 0 0 0 .287.376l1.5-.5-.159-.474-.808.27.596-
456                     .894a.25.25 0 0 0-.288-.376l-.808.27.596-.894z" />
457                 <path
458                     d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 16Zm-6.202-4.751 1.988
459                     -1.657a4.5 4.5 0 0 1 7.537-4.623L7.497 6.511 2.5 1.333
460                     3.11c-.56.251-1.18.39-1.833.39a4.49 4.49 0 0 1-1.592-
461                     .29L4.747 14.2a7.031 7.031 0 0 1-2.949-2.951zM12.496 8
462                     a4.491 4.491 0 0 1-1.703 3.526L9.497 8.512.959-1.11c
463                     .027.2.04.403.04.61z" />
464             </svg>
465
466             <span class="fw-bold"> Modo de operação </span>
467         </p>
468         <div class="switch-button my-0">
469             <input class="switch-button-checkbox" type="checkbox" id="
470                 op_mode_exchanger"
471                 onclick='buttonSetOpModeActuator(this.id, "exchanger")'><i
472                 nput>
473             <label class="switch-button-label" for=""><span class="
474                 switch-button-label-span">Automático</span></label>

```

```
456
457     </div>
458     <hr class="mt-2 mb-1">
459     <div>
460         <div id="automatic_exchanger" class="text-center d-flex flex-column">
461             <p class="m-1">
462                 <img alt="Sliders icon" class="bi bi-sliders" width="16" height="16" style="fill:currentColor;" viewBox="0 0 16 16" />
463                 <path fill-rule="evenodd" d="M11.5 2a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0 0-3zM9.05 3a2.5 2.5 0 0 1 4.9 0H16v1h-2.05a2.5 2.5 0 0 0 1-4.9 0H0V3h9.05zM4.5 7a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0 0-3zM2.05 8a2.5 2.5 0 0 1 4.9 0H16v1H6.95a2.5 2.5 0 0 0 1-4.9 0H0V8h2.05zm9.45 4a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0 0-3zm-2.45 1a2.5 2.5 0 0 1 4.9 0H16v1h-2.05a2.5 2.5 0 0 1 4.9 0H0v-1h9.05z" />
464             </path>
465         </div>
466         <span class="fw-bold"> Parâmetros do modo automático </span>
467             <span> pan </span>
468         </p>
469         <div class="d-flex flex-row ps-2">
470             <div class="form-check form-check-inline me-4">
471                 <input class="form-check-input" type="radio" name="flexRadioExchanger" id="param_on_exchanger" onclick="showHideDivParam(1, 'exchanger')" checked>
472                 <label class="form-check-label text-start negative-ms" for="param_on_exchanger">
473                     Ligar quando:
474                 </label>
475             </div>
476             <div class="form-check form-check-inline me-1 ms-auto">
477                 <input class="form-check-input" type="radio" name="flexRadioExchanger" id="param_off_exchanger" onclick="showHideDivParam(0, 'exchanger')">
478                 <label class="form-check-label text-start negative-ms" for="param_off_exchanger">
479                     Desligar após:
480                 </label>
481             </div>
482         </div>
483     </div>
484
485     <div id="card_param_exchanger" class="flip-container mt-3">
486         <div class="flipper">
487             <div id="card_front_exchanger" class="front">
```

```

490 <div class="row px-2">
491   <div class="row">
492     <div class="col-9">
493       <label for="exchanger_parameter_on_temperature"
494         class="col-form-label col-form-label-sm pb-0">
495         >
496         A diferença de temperatura entre o bebedouro
497         e a caixa estiver maior que </label>
498       </div>
499     </div>
500
501     <div class="col-3 pe-0">
502       <input type="number" class="form-control
503         form-control-sm"
504         id="exchanger_parameter_on_temperature">
505     </div>
506     <div class="col-9 ps-1 pe-2 d-flex flex-row
507       align-items-center ">
508       <span class="font-size-sm ">°C </span>
509       <div class="ms-auto">
510         <button type="submit" class="btn px-1
511           font-size-sm bg-warning mb-2"
512             onclick='buttonSetParameter("exchanger/on/
513               deltat", "
514               exchanger_parameter_on_temperature")'>
515               Salvar</button>
516             </div>
517           </div>
518         </div>
519       <div id="card_back_exchanger" class="back hidden-face">
520         <div class="row px-2 mt-4">
521
522           <div class="col-3 pe-0">
523             <input type="number" class="form-control
524               form-control-sm" id="
525               exchanger_parameter_off_time">
526           </div>

```

```
526 <div class="col-9 ps-1 pe-2 d-flex flex-row align-items-center">
527   <span class="font-size-sm">segundos </span>
528   <div class="ms-auto">
529     <button type="submit" class="btn px-1 font-size-sm bg-warning mb-2" onclick='buttonSetParameter("exchanger/off/time", "exchanger_parameter_off_time")'> Salvar</button>
530   </div>
531 </div>
532 </div>
533 </div>
534 </div>
535 </div>
536 </div>
537 </div>
538
539 <div id="manual_exchanger" class="text-center d-none flex-column">
540   <p class="m-1">
541     <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="currentColor" class="bi bi-play-fill pb-1 negative-me" viewBox="0 0 16 16">
542       d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-1.233-.697V4.308c0-.63.692-1.01 1.233-.696l6.363 3.692a.802.802 0 0 1 0 1.393z" />
543   </svg>
544   <span class="fw-bold"> Acionamento do atuador</span>
545 </p>
546 <div class="">
547   <div id="exchanger_off" class="row pe-2 mt-4 align-items-center">
548
549
550
551
552   <div class="col-5 pe-0">
553     <span class=""> Duração: </span>
554
555   </div>
556   <div class="col-2 ps-1 pe-2 d-flex flex-row align-items-center">
557     <input type="number" class="form-control form-control-sm" id="exchanger_parameter_time_act" >
558
559 </div>
```

```

560      <div class="col-4 pe-0 text-start">
561        <span class="">segundos </span>
562
563      </div>
564    </div>
565    <button id="button_act_exchanger" value="0" type="submit"
566      class="btn bg-warning btn-lg mt-2 mb-2"
567      onclick='buttonSetStatusActuator(this.value, "exchanger"
568      ")'>Interromper</button>
569    </div>
570  </div>
571
572  <button id="expanded_exchanger_button" type="button" onclick='
573    retractDiv("expanded_exchanger", "sensors")'
574    class="btn btn-outline-dark btn-sm d-none p-1 mt-5 ms-auto
575    mb-2 me-2"> Recolher Item
576  <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
577    fill="currentColor"
578    class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16"
579    >
580    <path
581      d="M2 0a2 2 0 0 0-2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
582      -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
583      0 0 1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
584      .70714.096 4.096a.5.5 0 1 1-.707.707z" />
585    </svg>
586  </button>
587 </div>
588
589 <div id="expanded_fan" class="d-lg-flex d-none col-12 col-lg-4
590   d-flex flex-column px-0 pt-2">
591   <p class="h5 text-center text-uppercase">Ventilador</p>
592   
593   <p class=" text-center">
594     <svg xmlns="http://www.w3.org/2000/svg" width="16" fill=
595       "currentColor"
596       class="bi bi-play-fill pb-1 negative-me" viewBox="0 0 16 16"
597       >
598       <path
599         d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-1.233-
600         .697V4.308c0-.63.692-1.01 1.233-.696l6.363 3.692a
601         .802.802 0 0 1 0 1.393z" />
602     </svg>
603     <span class="fw-bold">Status:</span> <span class="fw-bold
604       text-uppercase" id="status_fan1">---</span>
605   </p>

```

```

592
593   <p class=" text-center mb-0">
594     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
595       " fill="currentColor"
596       class="bi bi-wrench-adjustable-circle-fill" viewBox="0 0 16
597         16">
598       <path
599         d="M6.705 8.139a.25.25 0 0 0-.288-.376l-1.5.5.159.474.808
600           -.27-.595.894a.25.25 0 0 0 .287.376l.808-.27-.595.894a
601             .25.25 0 0 0 .287.376l1.5-.5-.159-.474-.808.27.596-
602               .894a.25.25 0 0 0-.288-.376l-.808.27.596-.894z" />
603     </path>
604   </svg>
605
606   <span class="fw-bold"> Modo de operação </span>
607 </p>
608 <div class="switch-button my-0">
609   <input class="switch-button-checkbox" type="checkbox" id="op_mode_fan"
610     onclick='buttonSetOpModeActuator(this.id, "fan")'></input>
611   <label class="switch-button-label" for=""><span class="switch-button-label-span">Automático</span></label>
612 </div>
613 <hr class="mt-2 mb-1">
614 <div>
615   <div id="automatic_fan" class="text-center d-flex flex-column">
616     <p class="m-1">
617       <svg xmlns="http://www.w3.org/2000/svg" width="16"
618         height="16" fill="currentColor"
619         class="bi bi-sliders" viewBox="0 0 16 16">
620         <path fill-rule="evenodd"
621           d="M11.5 2a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0 0-3zM9.05
622             3a2.5 2.5 0 0 1 4.9 0H16v1h-2.05a2.5 2.5 0 0 1-4.9
623               0H0V3h9.05zM4.5 7a1.5 1.5 0 1 0 0 3 1.5 1.5 0 0 0
624                 0-3zM2.05 8a2.5 2.5 0 0 1 4.9 0H16v1H6.95a2.5 2.5
625                   0 0 1-4.9 0H0V8h2.05zm9.45 4a1.5 1.5 0 1 0 0 3
626                     1.5 1.5 0 0 0 0-3zm-2.45 1a2.5 2.5 0 0 1 4.9 0
627                       H16v1h-2.05a2.5 2.5 0 0 1-4.9 0H0v-1h9.05z" />
628     </svg>
629   </div>
630 </div>

```

```
618         <span class="fw-bold"> Parâmetros do modo automático</span>
619     </p>
620     <div class="d-flex flex-row ps-2 ">
621         <div class="form-check form-check-inline me-4">
622             <input class="form-check-input" type="radio" name="flexRadioFan" id="param_on_fan"
623             onclick="showHideDivParam(1, 'fan')" checked>
624             <label class="form-check-label text-start negative-ms" for="param_on_fan">
625                 Ligar quando:
626             </label>
627         </div>
628         <div class="form-check form-check-inline me-1 ms-auto">
629             <input class="form-check-input" type="radio" name="flexRadioFan" id="param_off_fan"
630             onclick="showHideDivParam(0, 'fan')">
631             <label class="form-check-label text-start negative-ms" for="param_off_fan">
632                 Desligar quando:
633             </label>
634         </div>
635     </div>
636     <div id="card_param_fan" class="flip-container mt-3">
637         <div class="flipper">
638             <div id="card_front_fan" class="front">
639                 <div class="row px-2">
640                     <div class="row">
641                         <div class="col-9">
642                             <label for="fan_parameter_on_temperature" class="col-form-label col-form-label-sm pb-0">
643                                 <b>A</b>
644                                 temperatura ambiente estiver maior que</label>
645                         <br>
646                     </div>
647                 <div class="col-3 pe-0">
648                     <input type="number" class="form-control form-control-sm" id="fan_parameter_on_temperature">
649                 </div>
650                 <div class="col-9 ps-1 pe-2 d-flex flex-row align-items-center">
651                     <span class="font-size-sm ">°C </span>
652                     <div class="ms-auto">
653                         <button type="submit" class="btn px-1 font-size-sm bg-warning mb-2">
```

```

653         onclick='buttonSetParameter("fan/on/
654             temperature", "
655             fan_parameter_on_temperature")'>Salvar</b>
656             button>
657         </div>
658     </div>
659     <div id="card_back_fan" class="back hidden-face">
660         <div class="row px-2">
661             <div class="row">
662                 <div class="col-9">
663                     <label for="fan_parameter_off_temperature"
664                         class="col-form-label col-form-label-sm pb-0
665                         ">A
666                         temperatura
667                         ambiente estiver menor que</label>
668                     </div>
669                 <div class="col-3 pe-0">
670                     <input type="number" class="form-control
671                         form-control-sm" id="
672                         fan_parameter_off_temperature">
673                 </div>
674             <div class="col-9 ps-1 pe-2 d-flex flex-row
675                 align-items-center ">
676                 <span class="font-size-sm ">°C </span>
677                 <div class="ms-auto">
678                     <button type="submit" class="btn px-1
679                         font-size-sm bg-warning mb-2"
680                         onclick='buttonSetParameter("fan/off/
681                             temperature", "
682                             fan_parameter_off_temperature")'>Salvar</b>
683             button>
684         </div>
685     </div>
686
687     <div id="manual_fan" class="text-center d-none flex-column">
688         <p class="m-1">
689             <svg xmlns="http://www.w3.org/2000/svg" width="16" fill="
690                 currentColor"

```

```

686         class="bi bi-play-fill pb-1 negative-me" viewBox="0 0
687             16 16">
688     <path
689         d="m11.596 8.697-6.363 3.692c-.54.313-1.233-.066-
690             .233-.697V4.308c0-.63.692-1.01 1.233-.696l6.363
691                 3.692a.802.802 0 0 1 0 1.393z" />
692     </svg>
693     <span class="fw-bold"> Acionamento do atuador</span>
694   </p>
695   <div class="">
696     <button id="button_act_fan" value="0" type="submit"
697         class="btn bg-warning btn-lg mt-2"
698         onclick='buttonSetStatusActuator(this.value, "fan")'>
699             Desligar</button>
700   </div>
701   </div>
702 </div>
703 <button id="expanded_fan_button" type="button" onclick='
704     retractDiv("expanded_fan", "sensors")'
705     class="btn btn-outline-dark btn-sm d-none p-1 mt-5 ms-auto
706         mb-2 me-2"> Recolher Item
707 <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
708         fill="currentColor"
709         class="bi bi-arrow-up-left-square-fill" viewBox="0 0 16 16">
710         <path
711             d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
712                 -2V2a2 2 0 0 0-2-2H2zm8.096 10.803L6 6.707v2.768a.5.5
713                     0 0 1-1 0V5.5a.5.5 0 0 1 .5-.5h3.975a.5.5 0 1 1 0 1H6
714                         .70714.096 4.096a.5.5 0 1 1-.707.707z" />
715 </svg>
716 </button>
717 </div>
718 <!-- Atuadores recolhidos -->
719 <div
720     class="item-mobile d-block d-lg-none col-12 col-lg-4 d-flex
721         flex-column border-bottom border-4 border-dark p-2">
722     <p class="h6 text-center text-uppercase">Nebulizador</p>
723     <p class=" text-center mb-0">Status: <span class="fw-bold
724         text-uppercase" id="status_nebulizer2"> ---</span>
725     </p>
726     <p class=" text-center mb-0">Modo de operação: <span class="fw-bold
727         text-uppercase" id="opmode_nebulizer"> ---</span></p>
728     <button type="button" class="btn btn-outline-dark btn-sm
729         mt-auto ms-auto p-1">
```

```

717     onclick='expandDiv("expanded_nebulizer", "sensors")'>
718         Expandir item
719         <svg xmlns="http://www.w3.org/2000/svg" width="16" fill=
720             currentColor"
721             class=" bi bi-arrow-down-right-square-fill" viewBox="0 0
722                 16 16">
723             <path
724                 d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2 2
725                     v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
726                     .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
727                     0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
728         </svg>
729     </button>
730 </div>
731 <div
732     class="item-mobile d-block d-lg-none col-12 col-lg-4 d-flex
733         flex-column border-bottom border-4 border-dark p-2">
734         <p class="h6 text-center text-uppercase">Trocador de Água</p>
735         <p class=" text-center mb-0">Status: <span class="fw-bold
736             text-uppercase"
737                 id="status_exchanger2">---</span>
738             </p>
739             <p class=" text-center mb-0">Modo de operação: <span class="fw-bold
740                 text-uppercase" id="opmode_exchanger">
741                 --- </span></p>
742             <button type="button" class="btn btn-outline-dark btn-sm
743                 mt-auto ms-auto p-1"
744                 onclick='expandDiv("expanded_exchanger", "sensors")'>
745                     Expandir item
746                     <svg xmlns="http://www.w3.org/2000/svg" width="16" fill=
747                         currentColor"
748                         class=" bi bi-arrow-down-right-square-fill" viewBox="0 0
749                             16 16">
750                         <path
751                             d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2 2
752                                 v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
753                                 .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
754                                 0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
755                     </svg>
756                 </button>
757             </div>
758
759             <div class="item-mobile d-block d-lg-none col-12 col-lg-4
760                 d-flex flex-column p-2">
761                 <p class="h6 text-center text-uppercase">Ventilador</p>
762                 <p class=" text-center mb-0">Status: <span class="fw-bold
763                     text-uppercase" id="status_fan2">---</span>

```

```

746      </p>
747      <p class=" text-center mb-0">Modo de operação: <span class="fw-bold text-uppercase"
748          id="opmode_fan">---</span></p>
749      <button type="button" class="btn btn-outline-dark btn-sm
750          mt-auto ms-auto p-1"
751          onclick='expandDiv("expanded_fan", "sensors")'> Expandir
752          item
753          <svg xmlns="http://www.w3.org/2000/svg" width="16" fill=
754              "currentColor"
755              class=" bi bi-arrow-down-right-square-fill" viewBox="0 0
756              16 16">
757              <path
758                  d="M14 16a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2a2 2 0 0 0-2 2
759                  v12a2 2 0 0 0 2 2h12zM5.904 5.197 10 9.293V6.525a
760                  .5.5 0 0 1 1 0V10.5a.5.5 0 0 1-.5.5H6.525a.5.5 0 0 1
761                  0-1h2.768L5.197 5.904a.5.5 0 0 1 .707-.707z" />
762          </svg>
763          </button>
764      </div>
765      </div>
766
767      <script src="bootstrap/bootstrap.min.js"
768          integrity="sha384-cVKIPhGWic2Al4u+LWgxfKTRIcfu0JTxR+EQDz/
769          bgldoEyl4H0zUF0QKbrJ0EcQF"
770          crossorigin="anonymous"></script>
771      <script src="/scripts.js"> </script>
772  </body>
773
774  </html>

```

## 7 IMPLEMENTAÇÃO DO CÓDIGO CSS DA PÁGINA WEB

```
1 img {
2   display: block;
3   margin-left: auto;
4   margin-right: auto;
5 }
6
7 .button-align-right {
8   position: absolute;
9   bottom: 1px;
10  right: 1px;
11 }
12
13
14 .navbar .dropdown-menu {
15   background-color: #f0ad4e;
16 }
17
18 .font-size-sm {
19   font-size: 0.875rem !important;
20 }
21
22 *
23   box-sizing: border-box;
24 }
25
26 .switch-button {
27   background: rgb(238, 235, 235);
28   border-radius: 30px;
29   overflow: hidden;
30   width: 240px;
31   text-align: center;
32   letter-spacing: 1px;
33   color: black;
34   position: relative;
35   padding-right: 120px;
36   margin: auto;
37 }
38
39 .switch-button:before {
40   content: "Manual";
41   position: absolute;
42   top: 0;
43   bottom: 0;
44   right: 0;
45   width: 120px;
```

```
46  display: flex;
47  align-items: center;
48  justify-content: center;
49  z-index: 3;
50  pointer-events: none;
51 }
52
53 .switch-button-checkbox {
54   cursor: pointer;
55   position: absolute;
56   top: 0;
57   left: 0;
58   bottom: 0;
59   width: 100%;
60   height: 100%;
61   opacity: 0;
62   z-index: 2;
63 }
64
65 .switch-button-checkbox:checked+.switch-button-label:before {
66   transform: translateX(120px);
67   transition: transform 300ms linear;
68 }
69
70 .switch-button-checkbox+.switch-button-label {
71   position: relative;
72   padding: 5px 0;
73   display: block;
74   user-select: none;
75   pointer-events: none;
76 }
77
78 .switch-button-checkbox+.switch-button-label:before {
79   content: "";
80   background: #ffc107;
81   height: 100%;
82   width: 100%;
83   position: absolute;
84   left: 0;
85   top: 0;
86   border-radius: 30px;
87   transform: translateX(0);
88   transition: transform 300ms;
89 }
90
91 .switch-button-checkbox+.switch-button-label .switch-button-label-span {
92   position: relative;
```

```
93 }
94
95 .form-check-input[type="radio"]:checked {
96   background-color: #ffc107;
97   border-color: #ffc107;
98 }
99
100 .negative-ms {
101   margin-left: -0.4rem;
102 }
103
104 .negative-me {
105   margin-right: -0.3rem;
106 }
107
108 .flip-container.hover .flipper {
109   transform: rotateY(180deg);
110 }
111
112
113 .flipper {
114
115   transition: 0.5s;
116   transform-style: preserve-3d;
117
118   position: relative;
119 }
120
121 .front,
122 .back {
123   -webkit-backface-visibility: hidden;
124   -moz-backface-visibility: hidden;
125   -o-backface-visibility: hidden;
126   backface-visibility: hidden;
127 }
128
129 .back {
130   -webkit-transform: rotateY(180deg);
131   -moz-transform: rotateY(180deg);
132   -o-transform: rotateY(180deg);
133   transform: rotateY(180deg);
134 }
135
136 .hidden-face{
137   position: absolute;
138   top: 0;
139   left: 0;
```

```
140 }
141
142 /* Chrome, Safari, Edge, Opera */
143 input::-webkit-outer-spin-button,
144 input::-webkit-inner-spin-button {
145   -webkit-appearance: none;
146   margin: 0;
147 }
148
149 /* Firefox */
150 input[type=number] {
151   -moz-appearance: textfield;
152 }
153
154
155
156 @media screen and (max-width: 991px) {
157   .border-bottom-lg {
158     border-bottom: 4px solid black;
159   }
160
161   .border-end-lg-0 {
162     border-right: 0 !important;
163   }
164 }
```

## 8 IMPLEMENTAÇÃO DO CÓDIGO JAVASCRIPT DA PÁGINA WEB

```

1
2
3 updateSensors();
4 updateActuatorsState();
5 updateActuatorsOpMode();
6 updateActuatorsParam();
7
8 function buttonSetOpModeActuator(id_checkbox, actuator_name) {
9     let opmode = 0;
10    console.log(document.getElementById(id_checkbox).checked);
11    let value = document.getElementById(id_checkbox).checked;
12    if (value)
13        opmode = 1;
14
15    document.getElementById(id_checkbox).checked = !value;
16    var xhttp = new XMLHttpRequest();
17    xhttp.open("POST", "/actuators/opmode/set/" + actuator_name, true);
18    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'
19        );
20    xhttp.onreadystatechange = function () {
21        if (this.readyState == 4 && this.status == 200) {
22            updateOpMode(opmode, actuator_name, id_checkbox);
23        }
24    };
25    xhttp.send("value=" + opmode);
26}
27
28 function updateOpMode(value, actuator_name, id_checkbox) {
29     var automatic_div = document.getElementById("automatic_" + actuator_name)
30         ;
31     var manual_div = document.getElementById("manual_" + actuator_name);
32     if (value == "1") {
33         document.getElementById(id_checkbox).checked = true;
34         if (manual_div.classList.contains('d-none')) {
35             manual_div.classList.remove('d-none');
36             automatic_div.classList.add('d-none');
37         }
38         document.getElementById("opmode_" + actuator_name).innerHTML = "Manual"
39         ;
40     } else {
41         document.getElementById(id_checkbox).checked = false;
42         if(automatic_div.classList.contains('d-none')) {
43             automatic_div.classList.remove('d-none');
44             manual_div.classList.add('d-none');
45         }
46     }
47 }
```

```

43     document.getElementById("opmode_" + actuator_name).innerHTML = "Automá
        tico";
44 }
45 }
46
47 function dNoneAllItemsClass(class_name) {
48     var sensors = document.getElementsByClassName(class_name);
49     for (var i = 0; i < sensors.length; i++) {
50         sensors[i].classList.add('d-none');
51     }
52 }
53
54 function dFlexAllItemsClass(class_name) {
55     var sensors = document.getElementsByClassName(class_name);
56     for (var i = 0; i < sensors.length; i++) {
57         sensors[i].classList.remove('d-none');
58     }
59 }
60
61 function expandDiv(div_id_expand, div_id_none) {
62     var div = (document.getElementById(div_id_none));
63     div.classList.add('d-none');
64     dNoneAllItemsClass("item-mobile");
65     var div_expand = document.getElementById(div_id_expand);
66     var buttonRetract = document.getElementById(div_id_expand + "_button");
67     buttonRetract.classList.remove('d-none');
68     div_expand.classList.remove('d-none');
69     div_expand.classList.remove('border-4');
70     div_expand.classList.add('border-0');
71 }
72
73 function retractDiv(div_id_retract, div_id_block) {
74     var div = (document.getElementById(div_id_block));
75     div.classList.remove('d-none');
76     dFlexAllItemsClass("item-mobile");
77     var div_retract = document.getElementById(div_id_retract);
78     var buttonRetract = document.getElementById(div_id_retract + "_button");
79     buttonRetract.classList.add('d-none');
80     div_retract.classList.add('d-none');
81     div_retract.classList.add('border-4');
82     div_retract.classList.remove('border-0');
83 }
84
85 function updateActuatorsParam() {
86     var xhttp = new XMLHttpRequest();
87     xhttp.onreadystatechange = function () {
88         if (this.readyState == 4 && this.status == 200) {

```

```

89     var array = this.responseText.split(" ");
90
91     //index = |0: ligar nebulizador umidade| |1: desligar nebulizador
92     //umidade| |2: ligar nebulizador temperatura|
93     //|3: desligar nebulizador temperatura| |4: ligar trocador de água
94     //variação temperatura| |5: desligar trocador de água tempo|
95     //|6: ligar ventilador temperatura| |7: desligar ventilador
96     //temperatura|
97     document.getElementById("nebulizer_parameter_on_humidity") .
98         placeholder = array[0].replace(".", ",");
99     document.getElementById("nebulizer_parameter_off_humidity") .
100        placeholder = array[1].replace(".", ",");
101    document.getElementById("nebulizer_parameter_on_temperature") .
102        placeholder = array[2].replace(".", ",");
103    document.getElementById("nebulizer_parameter_off_temperature") .
104        placeholder = array[3].replace(".", ",");
105    document.getElementById("exchanger_parameter_on_temperature") .
106        placeholder = array[4].replace(".", ",");
107    document.getElementById("exchanger_parameter_off_time") .placeholder =
108        array[5].replace(".", ",");
109    document.getElementById("exchanger_parameter_time_act") .placeholder =
110        array[5].replace(".", ",");
111    document.getElementById("fan_parameter_on_temperature") .placeholder =
112        array[6].replace(".", ",");
113    document.getElementById("fan_parameter_off_temperature") .placeholder =
114        array[7].replace(".", ",");
115
116    }
117
118    xhttp.open("GET", "/actuators/parameter/getall", true);
119    xhttp.send();
120
121
122    function showHideDivParam(radioCheck, actuator_name) {
123        var divParam = document.getElementById("card_param_" + actuator_name);
124        if (radioCheck) {
125            divParam.classList.remove('hover');
126            document.getElementById("card_front_" + actuator_name).classList.remove(
127                'hidden-face');
128            document.getElementById("card_back_" + actuator_name).classList.add('
129                hidden-face');
130        } else {
131            divParam.classList.add('hover');
132            document.getElementById("card_back_" + actuator_name).classList.remove(
133                'hidden-face');
134        }
135    }

```

```

120     document.getElementById("card_front_" + actuator_name).classList.add('
121         hidden-face');
122 }
123
124 function updateActuatorState(state, actuator_name) {
125     var button = document.getElementById("button_act_" + actuator_name);
126     if (state == true) {
127
128         if (actuator_name == "exchanger") {
129             button.innerHTML = "Interromper a troca";
130             document.getElementById("exchanger_off").classList.add('d-none');
131         }
132     else
133         button.innerHTML = "Desligar";
134
135     button.value = "0";
136     document.getElementById("status_" + actuator_name + "1").innerHTML = "
137         Ligado";
138     document.getElementById("status_" + actuator_name + "2").innerHTML = "
139         Ligado";
140     document.getElementById("status_" + actuator_name + "1").style.color =
141         "green";
142     document.getElementById("status_" + actuator_name + "2").style.color =
143         "green";
144 } else {
145     if (actuator_name == "exchanger") {
146         document.getElementById("exchanger_off").classList.remove('d-none');
147         button.innerHTML = "Realizar a troca";
148     }
149     else
150         button.innerHTML = "Ligar";
151
152     button.value = "1";
153     document.getElementById("status_" + actuator_name + "1").innerHTML = "
154         Desligado";
155     document.getElementById("status_" + actuator_name + "2").innerHTML = "
156         Desligado";
157     document.getElementById("status_" + actuator_name + "1").style.color =
158         "red";
159     document.getElementById("status_" + actuator_name + "2").style.color =
160         "red";
161 }
162
163 function updateActuatorsState() {
164     var xhttp = new XMLHttpRequest();

```

```

158 xhttp.onreadystatechange = function () {
159   if (this.readyState == 4 && this.status == 200) {
160     var array = this.responseText.split(" ");
161     //array index = |0: trocador| |1: ventilador| |2: nebulizador|
162     updateActuatorState(array[0], "exchanger");
163     updateActuatorState(array[1], "fan");
164     updateActuatorState(array[2], "nebulizer");
165
166   }
167 };
168 xhttp.open("GET", "/actuators/status/getall", true);
169 xhttp.send();
170 }
171
172
173 function buttonSetParameter(url, inputID) {
174   var xhttp = new XMLHttpRequest();
175   xhttp.open("POST", "/actuators/parameter/" + url, true);
176   xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
177   xhttp.onreadystatechange = function () {
178     if (this.readyState == 4) {
179       if (this.status == 200) {
180         alert("Parâmetro reconfigurado!");
181       }
182       if (this.status == 304) {
183         alert("Erro em Modificar o parâmetro de acionamento do equipamento!");
184       }
185     }
186   }
187 };
188 xhttp.send("value=" + document.getElementById(inputID).value);
189 }
190
191 function updateActuatorsOpMode() { //atualizar o modo de operação dos
192   atuadores
193   xhttp = new XMLHttpRequest();
194   xhttp.onreadystatechange = function () {
195     if (this.readyState == 4 && this.status == 200) {
196       var array = this.responseText.split(" ");
197
198       //array index = |0: nebulizador| |1: trocador de água| |2:
199       //ventilador|
200
201       updateOpMode(array[0], "nebulizer", "op_mode_nebulizer");
202     }
203   }
204 }
```

```

201
202     updateOpMode(array[1], "exchanger", "op_mode_exchanger");
203
204
205     updateOpMode(array[2], "fan", "op_mode_fan");
206 }
207 };
208 xhttp.open("GET", "/actuators/opmode/getall", true);
209 xhttp.send();
210 }
211
212 function updateSensors() {
213     var xhttp = new XMLHttpRequest();
214     xhttp.onreadystatechange = function () {
215         if (this.readyState == 4 && this.status == 200) {
216             var array = this.responseText.split(" ");
217
218             //array index = |0: temperatura ambiente| |1: umidade ambiente
219             //relativa| |2: temperatura da caixa| |3: temperatura do bebedouro|
220
221             updateSensor(array[3], "nipple_temperature1", "°C", "-127.00");
222             updateSensor(array[3], "nipple_temperature2", "°C", "-127.00");
223
224             updateSensor(array[2], "box_temperature1", "°C", "-127.00");
225             updateSensor(array[2], "box_temperature2", "°C", "-127.00");
226
227             updateSensor(array[1], "climate_humidity1", "%", "nan");
228             updateSensor(array[1], "climate_humidity2", "%", "nan");
229
230             updateSensor(array[0], "climate_temperature1", "°C", "nan");
231             updateSensor(array[0], "climate_temperature2", "°C", "nan");
232         }
233     };
234     xhttp.open("GET", "/sensors/getall", true);
235     xhttp.send();
236 }
237
238 function updateSensor(value, sensor_span, unit_measurement, text_error) {
239     if (value == text_error) {
240         document.getElementById(sensor_span).innerHTML = "Falha na leitura do
241             sensor!";
242         document.getElementById(sensor_span).style.color = "red";
243     } else {
244         document.getElementById(sensor_span).style.color = "black";
245         document.getElementById(sensor_span).innerHTML = value +
246             unit_measurement;
247     }

```

```
245 }
246
247 function buttonsetStatusActuator(button_value, actuator_name) {
248     var xhttp = new XMLHttpRequest();
249
250     xhttp.open("POST", "/actuators/status/set/" + actuator_name, true);
251     xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
252     xhttp.onreadystatechange = function () {
253         if (this.readyState == 4 && this.status == 200) {
254             updateActuatorState(button_value, actuator_name);
255         }
256     };
257     var requestSend = "value=" + button_value;
258     if (actuator_name == "exchanger")
259         requestSend += "&time=" + document.getElementById("exchanger_parameter_time_act").value;
260
261     xhttp.send(requestSend);
262 }
263
264 //taxa de atualização de 5 segundos
265 setInterval(function () {
266     //atualizando o valor dos sensores
267     /*updateSensors();
268     updateActuatorsState();
269     updateActuatorsOpMode();
270     updateActuatorsParam();*/
271 }, 5000);
```

**9 CIRCUITO ELÉTRICO GERAL DO PROJETO**