INSTITUTO FEDERAL
São Paulo
Câmpus Avaré

RAFAEL RANAL SANTORO

PROJETO E CONSTRUÇÃO DE UM DISPOSITIVO ELETRÔNICO PARA CONTROLE, MONITORAMENTO E REGISTRO DA TEMPERATURA, LUMINOSIDADE, UMIDADE E PRESENÇA DE AMÔNIA EM UMA GRANJA DE AVES

Avaré

2022

RAFAEL RANAL SANTORO

PROJETO E CONSTRUÇÃO DE UM DISPOSITIVO ELETRÔNICO PARA CONTROLE, MONITORAMENTO E REGISTRO DA TEMPERATURA, LUMINOSIDADE, UMIDADE E PRESENÇA DE AMÔNIA EM UMA GRANJA DE AVES

Versão Original

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Biossistemas do Instituto Federal de São Paulo, como requisito parcial para obtenção do título de Bacharel em Engenharia de Biossistemas.

Orientador:

Prof. Me. Fábio Henrique Busquim Pereira

Coorientador:

Prof. Otávio Luiz Medeiros Tibagy

Avaré

2022

Catalogação na fonte
Instituto Federal de São Paulo – Campus Avaré

Santoro, Rafael Ranal
Projeto e construção de um dispositivo eletrônico para controle, monitoramento e registro da temperatura, luminosidade, umidade e presença de amônia em uma granja de aves / Rafael Ranal Santoro - Avaré, 2022. 98 p.

Orientador: Prof. Me. Fábio Henrique Busquim Pereira
Coorientador: Prof. Otávio Luiz Medeiros Tibagy
Monografia (Graduação – Engenharia de Biossistemas) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Campus Avaré, Avaré, 2022.

1. Microcontrolador. 2. Aviário. 3. Internet das coisas. 4. Protocolo MQTT 5. ESP-8266EX I. Pereira, Fábio Henrique Busquim. II. Tibagy, Otávio Luiz Medeiros III Título.

FORMULÁRIO N.º 3/2022 - CCM-AVR/DAE-AVR/DRG/AVR/IFSP

FOLHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO
IDENTIFICAÇÃO DO ALUNO
Nome: Rafael Ranal Santoro
Título: Projeto e construção de um dispositivo eletrônico para controle, monitoramento e registro da temperatura, luminosidade, umidade e níveis de amônia em uma granja de aves
Curso: Bacharelado em Engenharia de Biossistemas
BANCA EXAMINADORA
Nome: Fabio Henrique Busquim Pereira
Instituição/Departamento: IFSP/Avaré
Nota: 9,9 **Julgamento: (x) Aprovado ( ) Reprovado**
Assinatura: *[assinado eletronicamente]*
Nome: Newton Tamassia Pegolo
Instituição/Departamento: IFSP/Avaré
Nota: 9,8 **Julgamento: (x) Aprovado ( ) Reprovado**
Assinatura: *[assinado eletronicamente]*
Nome: Marcela Pavan Bagagli
Instituição/Departamento: IFSP/Avaré
Nota: 9,7 **Julgamento: (x) Aprovado ( ) Reprovado**
Assinatura: *[assinado eletronicamente]*

RESULTADO FINAL
Como parte das exigências para conclusão do Curso de Engenharia de Biossistemas, o candidato(a)/aluno(a), em sessão pública, foi considerado **aprovado** pela Comissão Examinadora, com média final **9,8**.

Avaré, 12 de agosto de **2022**.

Documento assinado eletronicamente por:

- **Fabio Henrique Busquim Pereira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 12/08/2022 22:36:43.
- **Marcela Pavan Bagagli**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 13/08/2022 12:35:04.
- **Newton Tamassia Pegolo**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 15/08/2022 10:58:37.

Este documento foi emitido pelo SUAP em 11/08/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 397281
Código de Autenticação: 7bafae574f

FORMULÁRIO N.º 3/2022 - CCM-AVR/DAE-AVR/DRG/AVR/IFSP

1ª via IFSP, 2ª via do(a) Aluno(a), 3ª via do(a) Co-orientador(a)

"Este documento não contém rasuras"

AGRADECIMENTOS

A Deus sobre todas as coisas, desde as mais pequeninas, como as sementes que sustentam os pássaros, até as mais grandiosas e esplêndidas, como todas as galáxias que ornam o céu nas noites escuras, pela intuição, pela consciência e pela vida.

Aos meus filhos, Sophia e Eli, que trazem força e inspiração apenas com um olhar ou sorriso, para sempre continuar, mesmo que à frente estejam as maiores adversidades.

À minha amada esposa Jaqueline, que por diversas vezes pediu por mais atenção e mesmo assim sustentou as atividades cotidianas para eu poder continuar com os estudos, principalmente com esse trabalho.

Aos meus pais, Adriana e Ricardo, pela criação, ensinamentos e conselhos.

Ao meu irmão, Tarsísio, e a minha colega Carol Grunow por me auxiliar com a revisão.

Aos professores; seres magníficos e mágicos, que buscaram sempre que tivesse o aprendizado mais cristalino, com todo o respeito, dedicação e ternura. Dentre eles: Fábio Crivelli, Estela, Otávio, Geza, Gustavo Pio, Celso, Danilo Budoya, Demétrio, Emerson, Fernanda Pazini, Julio, Lívia, Maressa, Newton, Rafael, Raíssa, Ronald, Vanda, Hugo, Ádria, André, Rodrigo Predolin, Adilson e tantos outros, igualmente importantes.

A professora Marcela, especialmente, pelo esforço em ensinar com qualidade distinta, às vezes até pela madrugada adentro e também por seus puxões de orelha.

Ao meu orientador Fabio Busquim, que com os seus conselhos fizeram com que esse trabalho se tornasse melhor, mais fluido e completo.

Aos meus colegas, Ruan, Danilo, Igor, Lucas Faxina, Leonardo, Mayara, Nathália, Ingrid, Jussara, entre tantos outros, pela ajuda e pelo respeito, que nunca nos faltou.

Aos meus queridos amigos, Jamil Simplício e Marcílio Barros, que nessa caminhada sempre se fizeram presentes e sustentamo-nos, mesmo com grandes adversidades, desde o início, com companheirismo, respeito, amizade e colaboração.

E ao Instituto Federal de São Paulo, com seus milhares de servidores, que contribuíram com a minha formação, inclusive aqueles que o fizeram anonimamente.

A todos os contribuintes, que possibilitam a existência da universidade pública no nosso país.

“Voici mon secret.
C’est très simple: on ne voit bien qu’avec le cœur.
L’essentiel est invisible pour les yeux”

Aqui está o meu segredo.
É muito simples: só vemos bem com o coração.
O essencial é invisível aos olhos.

(Antoine de Saint-Exupéry, 1900-1944)

RESUMO

SANTORO, Rafael Ranal. Projeto e construção de um dispositivo eletrônico para controle, monitoramento, registro da temperatura, luminosidade, umidade e presença de amônia em uma granja de aves. 2022. Monografia (Bacharelado em Engenharia de Biossistemas) – IFSP – Instituto Federal de São Paulo, Avaré, 2022.

Neste trabalho, objetivou-se a construção de um equipamento funcional para a coleta de parâmetros tais como: temperatura, umidade, luminosidade e presença de amônia, em aviários, registrando seus dados e disponibilizando na internet por meio de protocolo MQTT, seguindo o conceito de internet das coisas. Podendo este dispositivo também realizar o acionamento de dispositivos para controle das variáveis, por meio de relés. O protótipo foi projetado para ser utilizado em granjas de pequenos produtores de aves, podendo ser conectado ao sistema elétrico de ventiladores, resistências, exaustores, cortinas, etc. Utilizou-se como microcontrolador o ESP-8266EX em uma placa WeMos D1 Mini, e os sensores MQ-135, DHT22 e LDR GL-5528. O código-fonte foi desenvolvido na plataforma Arduino IDE, no formato de código livre. Os testes foram realizados em ambiente fechado e aberto, e ao final obteve-se um protótipo que manteve os valores de temperatura e umidade dentro de parâmetros estabelecidos, controlou a luminosidade por critério de tempo e valores obtidos pelo sensor de luminosidade, bem como detectou com sucesso a presença de amônia.

Palavras-chave: microcontrolador; aviário; internet das coisas; protocolo MQTT; ESP-8266EX.

ABSTRACT

SANTORO, Rafael Ranal. Design and construction of an electronic device to control, monitor, record temperature, luminosity, humidity and presence of ammonia in a poultry farm. 2022. Monograph (Bachelor's degree in Biosystems Engineering) – IFSP – Federal Institute of São Paulo, Avaré, 2020.

The objective of this work was to build a functional equipment for the collection of parameters, temperature, humidity, luminosity and presence of ammonia, in aviaries, recording its data and making it available on the internet through the MQTT protocol, following the concept of internet of things. This device can also activate devices to control the variables, through relays. The prototype was designed to be used in farms of small poultry producers, and can be connected to the electrical system of fans, resistors, exhaust fans, curtains, etc. The ESP-8266EX was used as a microcontroller on a WeMos D1 Mini board, and the MQ-135, DHT22 and LDR GL-5528 sensors. The source code was developed on the Arduino IDE platform, in open source format. The tests were carried out in closed and open environments, and in the end, a prototype was obtained that kept the temperature and humidity values within established parameters, controlled the luminosity by time criterion and values obtained by the luminosity sensor, as well as detected with successfully the presence of ammonia.

Keywords: microcontroller; aviary, internet of things; MQTT protocol; ESP-8266EX.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS E ABREVIATURAS

CSV	Comma Separated Values (Valores separados por vítgulas)
DTH22	Humidity and Temperature Sensor (Sensor de umidade relativa e temperatura)
ESP-8266EX	Espressif Chip
FLASH	Electrically-Erasable Programmable Read-Only Memory (Memória somente de leitura programável apagável eletricamente)
FZEA	Faculdade de Zootecnia e Engenharia de Alimentos
FSH	Hormônio folículo estimulante
GND	Ground (Terra)
GnRH	Hormônio liberador de gonadotrofina
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
IDE	Integrated Development Environment (Ambiente de desenvolvimento integrado)
IoT	Internet of Things (Internet das Coisas)
IP	Internet Protocol (Protocolo de Internet)
LCD	Liquid Cystal Display (Tela de Cristal Líquido)
LDR	Light Dependent Resistor (Resistor dependente de luz)
LED	Light Emissor Diode (Diodo emissor de luz)
LH	Hormônio luteinizante
MQTT	Message Queuing Telemetry Transport (Transporte de Enfileiramento de Mensagens por Telemetria)
MySQL	My Structured Query Language (Minha linguagem de pesquisa estruturada)
PCB	Printed Circuit Board (Placa de circuito impresso)
PHP	Hypertext Preprocessor (Pré-processador de Hypertexto)
PWM	Pulse Width Modulation (Modulação por largura de pulso)
RFID	Radio Frequency Identification (Identificação por Radiofreqüência)
RTC	Real Time Clock (Relógio de tempo real)
SD	Secure Digital (Segurança Digital)
TCP	Transmission Control Protocol (Protocolo de Controle e Transmissão)
USB	Universal Serial Bus (Barramento serial universal)
USP	Universidade de São Paulo
Wi-Fi	Wireless Fidelity (Fidelidade sem fio)

LISTA DE SÍMBOLOS

CO	monóxido de carbono	A	amperagem
CO_2	dióxido de carbono	mA	miliampere
C_7H_8	tolueno	mm	milímetro
NH_3	amônia	cm	centímetro
ppm	partes por milhão	m	metro
%	percentual	kB	quilo-bytes
pH	potencial hidrogeniônico	MB	megabytes
Kg	quilograma	Mbps	megabit por segundo
mg	miligrama	R^2	coeficiente de determinação
m^3	metro cúbico	W	watts
hab	habitante	mW	miliwatt
R_0	Resistência do sensor no ar limpo	di/dt	velocidade de contração no tempo
Rs	Resistência do sensor	L	indutância
º	grau	Ohm ou Ω	unidade de resistência
ºC	grau Celsius	kΩ	quilo-ohm
Lux	unidade de luminosidade	MΩ	mega-ohm
Hz	Hertz	V_{BE}(sat)	voltagem de saturação para a base
kHz	quilo-hertz	Ic	corrente para o coletor
MHz	mega-hertz	I_B	corrente para a base
GHz	giga-hertz	±	tolerância
V	voltagem		

SUMÁRIO

1. INTRODUÇÃO

O Brasil é um dos grandes produtores mundiais de carne de frango, com 68% de sua produção destinada ao mercado interno, tendo como consumo 42,84 Kg/hab. Situa-se na terceira posição mundial de produção com 12.245 mil toneladas, atrás apenas de China com 13.750 mil toneladas e Estados Unidos com 19.941 mil toneladas. No ano de 2019 foram exportadas 4.214 mil toneladas que geraram uma receita de 6,994 bilhões de dólares. O Oriente Médio e a Ásia são os principais mercados compradores do frango brasileiro. (ABPA, 2021).

Na produção de ovos, o Brasil destina 99,59% de sua produção para o mercado interno. Em 2019 a produção de ovos no país foi de 49 bilhões de unidades, isso representa um grande crescimento, em relação aos quase 29 bilhões de unidades produzidos em 2010, um aumento de quase 41%. O consumo *per capita* aumentou de 148 para 230 unidades. O estado com maior alojamento de pintainhas em 2019 foi São Paulo, com 32.97% (ABPA, 2021).

A relação mais comum entre as principais indústrias de abate de frango e os criadores de frangos não é uma relação de compra e venda, mas um negócio regido pelo sistema de integração, em que os frigoríficos adiantam os pintinhos, os remédios e a ração para os produtores e em troca o produtor é condicionado a comercializar as aves exclusivamente com essa empresa. Após ser descontado o custo, o produtor é finalmente remunerado (MONITOR, 2016).

Nesse contexto entre grandes empresas e pequenos produtores, as empresas possuem grande poder de barganha, exigindo padrões de qualidade contratualmente, como o tamanho dos aviários e peso final de abate, o que pode demandar altos investimentos dos produtores (MONITOR, 2016).

Devido ao número grande de parâmetros envolvidos na produção de aves, antes da adoção de dispositivos eletrônicos que medem e controlam, era necessário o emprego de muito mais mão de obra, o que elevava os custos. Com o emprego dos dispositivos eletrônicos e o uso da Internet das Coisas (IoT), foi possível reduzir essa mão de obra, isso possibilita também que o produtor tenha na palma das mãos o controle à distância do ambiente de produção. Pode, assim, conhecer a temperatura e outras condições à distância, além de também poder atuar remotamente na correção das condições, como em uma tela de celular (ASCOM, 2018).

Os sensores são os componentes que, de maneira relativamente mais fácil, podem ser implementados. Os que controlam a temperatura, amônia, dióxido de carbono, luminosidade e ventilação são amplamente utilizados em aviários modernos. Sensores como RFID, podem fornecer dados a respeito da maneira como as aves se movimentam em ambientes abertos, e através das gaiolas, proporcionando dados que podem ser utilizados para aprimorar a eficiência na produção (CONNOLLY, 2022).

O Brasil é um país tropical, com intensidade de calor nos meses de dezembro a março. Essa variação de calor interfere negativamente, retirando as aves da zona de conforto térmico. Em altas temperaturas as aves reduzem o consumo de ração, a fim de diminuir a produção interna de calor e por consequência há perda de peso pelas aves. Para que as aves permaneçam em conforto climático, diversos métodos de resfriamento podem ser utilizados, sendo o mais eficaz deles a ventilação, controlando a entrada e saída de calor, removendo o excesso de umidade, dióxido de carbono e amônia. (SOARES; FERREIRA, 2020).

A amônia é um gás incolor, irritante, produto da decomposição de dejetos provocada por microrganismos. Muitos criadores desconhecem os efeitos maléficos da amônia em seus galpões. A presença pode ocasionar além da redução de peso das aves, lesões nos olhos, cegueira e subdesenvolvimento de carcaça. Com as perdas de peso de subdesenvolvimento de carcaça o lote perde o padrão exigido pelos abatedores, o que ocasiona perda financeira. A amônia também paralisa e, em maiores concentrações, danifica os cílios das vias respiratórias, abrindo uma janela para infecções, como por bactérias *Escherichia coli* e aumenta as chances de as aves adquirirem Bronquite e Newcastle. (LOTT; DONALD, 2003).

A incidência de luz a que as aves são submetidas afeta diretamente a produção de hormônios, como o liberador de gonadotrofina (GnRH), que atua nas funções reprodutivas, comportamentais e características secundárias. Entre os efeitos da ausência do controle de luminosidade estão a demora de 3 a 4 semanas na idade do início da produção dos primeiros ovos, picos atrasados e baixos de produção, problemas de eclosão e sobrepeso das fêmeas. A utilização de luz nos primeiros dias de vida pode estimular o consumo de ração e o ganho de peso ideal (CFMV, 2011).

Neste trabalho, foi realizado o monitoramento e controle de variáveis relevantes para a criação de aves, a fim de obter-se controle sobre o ambiente de produção utilizando sensores e atuadores. Os dados coletados foram armazenados e disponibilizados em um dispositivo conectado à internet, possibilitando fácil acesso dos produtores.

2. OBJETIVOS

2.1. Objetivo Geral

Desenvolver um equipamento funcional e confiável, de custo reduzido, para a utilização em granjas de aves, com o monitoramento, registro e controle da variação de temperatura, umidade, luminosidade e presença de gás amônia ao longo do tempo, a fim de automatizar e otimizar a produção, possibilitando a melhora da qualidade de vida das aves no ambiente de produção.

2.2. Objetivos Específicos

- Projetar equipamento eletrônico com a escolha de componentes e levantamento de custos.
- Desenvolver equipamento de controle e automação, para controlar as variáveis de temperatura, umidade, luminosidade e presença de amônia.
- Realizar o registro das mudanças das variáveis, para a posterior análise e embasamento na tomada de decisão dos pequenos produtores, quanto a modificações no ambiente de produção no escopo da ambiência.
- Permitir através do equipamento desenvolvido a consulta “on-line”, em tempo real, das condições atuais no ambiente de produção por meio da internet, utilizando apenas um navegador em qualquer dispositivo, como celular, *tablet* ou computador.
- Elaborar um documento que possa ser utilizado como ponto de partida, ou exemplo de aplicação de técnicas da engenharia, com foco na automação e programação para a comunidade acadêmica.

3. JUSTIFICATIVA

Nos dias atuais, já se sabe que mudanças nas variáveis da ambiência afetam diretamente os níveis de produção nas granjas, em parâmetros de perda de peso, perda de indivíduos, acometimento de doenças, mudanças nas taxas de reprodução, entre outros.

Muitos produtores, por serem pequenos, não possuem recursos financeiros ou conhecimentos técnicos suficientes para adquirir ou elaborar sistemas automatizados de controle de ambiência, sistemas esses que permitirão ao produtor se ausentar da área de produção, não sendo necessária a intervenção manual no período, podendo este dispensar mais atenção para outras atividades relativas à produção. Além do ganho de tempo o produtor pode economizar na energia, pois as medidas de contenção, como refrigeração, não serão acionadas sem necessidade, ou ficarão mais tempo acionadas do que o estritamente necessário.

O sistema de automação também permite que as aves tenham uma qualidade de vida superior, uma vez que não estarão expostas a variações abruptas das intempéries do tempo, assim recebendo um tratamento mais humanizado.

A elaboração deste equipamento também permitirá que conhecimentos acerca de automação e programação, possam ser compartilhados com a comunidade científica de interesse e, futuramente, aprimorado.

4. REVISÃO DE LITERATURA

4.1. O estado da arte

Considerando um contexto regional e de desenvolvimentos de baixo custo com aplicação em pequenos produtores, alguns projetos estão apresentados na literatura, entre os quais apresentam relevância os descritos a seguir.

Em um trabalho realizado na Universidade de Uberaba (2020), foi desenvolvido um sistema para controle na cunicultura (criação de coelhos), com o objetivo de acionar o sistema de ventiladores a depender das temperaturas e umidades internas e externas ao galpão, registrá-las e posteriormente serem extraídas por meio de um cartão de memória do tipo SD. No trabalho foi utilizada a plataforma Arduino como microcontrolador e foram utilizados os sensores de temperatura e umidade DHT22. O sistema de ventilação era acionado por meio de um relé eletromecânico. Foi utilizado um painel do tipo LCD, a fim de possibilitar no local a leitura dos valores diretamente no aparelho, este modelo não envia os dados pela internet ou Wi-Fi. O trabalho obteve êxito, conseguindo acionar o sistema de ventilação e mantendo a temperatura e umidade dentro de valores pré-estabelecidos (SANTOS; JÚNIOR, 2020).

Outro trabalho, desenvolvido na USP (Universidade de São Paulo), objetivando o controle de temperatura e umidade, aplicou vários sensores sem fio, espalhados pelo galpão. A luminosidade também foi mensurada. Os dados coletados pelos sensores foram enviados para um computador, que fez o processamento e o controle dos ventiladores, aparelhos de ar-condicionado e cortinas do ambiente. Nesse sistema as informações foram enviadas pela internet e os dispositivos conectados podiam ser acionados remotamente pelo produtor. Foi observada houve uma redução de 80% no consumo da energia elétrica e os objetivos quanto a temperatura e umidade foram alcançados, melhorando o conforto térmico das aves (AGEVOLUTION, 2020).

Na FZEA (Faculdade de Zootecnia e Engenharia de Alimentos da USP), desenvolveu-se uma solução apenas para monitoramento dos parâmetros de luminosidade, temperatura e umidade. Os sensores utilizavam o microcontrolador ESP-8266EX, sensor de luminosidade LDR e o sensor de temperatura e umidade DHT11. Os dados coletados eram então enviados a um servidor que podia ser acessado para consulta por meio de uma página em PHP (Pré-processador de Hipertexto), sendo os dados armazenados em um banco de dados MySQL. O

projeto alcançou seu objetivo armazenando os dados de maneira satisfatória no banco de dados (LIBERA; OLIVEIRA; TECH, 2018).

4.2. Efeitos da Amônia na avicultura

A amônia é um gás tóxico muito comum em galpões de criação de aves. O gás se origina do ácido úrico e da proteína não digerida presente na cama de frango, sendo produto da degradação por microrganismos presentes no esterco e na cama de frango (HAN *et al*., 2021). Esse processo ocorre em certos níveis de temperatura e pH (EMBRAPA, 2011).

A presença da amônia pode causar cegueira, queimaduras nos coxins plantares, irritação ocular, irritações na pele, calos no peito, perda de peso, baixa uniformidade e suscetibilidade a doenças (COBB-VANTRESS, 2009). A partir de 10ppm já ocorre a deterioração dos cílios do epitélio traqueal e em concentrações acima de 75ppm podem ocorrer ulcerações no globo ocular, ocasionando dificuldade de alimentar-se, aumento de estresse e mortalidade (EMBRAPA, 2011). Outro efeito da presença de amônia é a alteração na microbiota dos cecos das aves, grupos expostos a níveis de amônia acima de 25 e 35ppm, tiveram problemas quanto ao crescimento (HAN et al., 2021).

Na Figura 1, pode-se observar diferentes graus de lesões em coxims plantares de frangos de corte.

Figura 1 - Coxins plantares lesionados por pododermatite em diferentes graus
Fonte: Adaptado de RORIZ, 2016.

A cama de frango com alto nível de umidade colabora com o aumento nos níveis de volatilização da amônia formada nas camas. Alterações no trato nutricional, utilização de condicionadores, compostagem e a redução da umidade da cama, podem ajudar a reduzir os níveis desse gás no galpão (EMBRAPA, 2011). Para reduzir os efeitos e proporcionar ar de

maior qualidade no aviário, o processo de ventilação mínima deve ser utilizado (COBB VANTRESS, 2009).

A amônia não representa apenas riscos para as aves, mas também para humanos, que podem detectar a substância pelo odor entre 5 e 50ppm. A exposição em níveis de 100 a 500ppm pode causar, irritação nas mucosas superficiais em uma hora, níveis de 400 a 700ppm, provocam irritação imediata dos olhos nariz e garganta, de 2000 a 3000ppm há severa irritação dos olhos, tosse intensa, podendo ser fatal, em níveis próximos a 5000ppm, ocorrem espasmos respiratórios e asfixia, 10.000ppm é considerado o nível fatal (EMBRAPA, 2011).

Idealmente, o nível de amônia no ambiente de criação de aves deve ser menor que 10ppm (ROSS, 2018). O nível máximo permitido em instrução normativa no Brasil é de 20ppm para os trabalhadores (BRASIL, 2022).

4.3. Efeitos da variação de temperatura e umidade na avicultura

As inovações adotadas nos aviários visam a dispersão do calor, a fim de que as aves possam expressar todo o seu potencial de crescimento. A observação dos métodos construtivos, orientação, materiais utilizados, dimensões, localização, entre outros aspectos são insuficientes para adequar o ambiente a temperatura ideal para as aves. Túneis de ventilação e sistemas de resfriamento evaporativo têm sido utilizados para evitar a influência da temperatura externa. (ABREU; ABREU).

As aves assim como os humanos são animais homeotérmicos, ou seja, são capazes de regular a temperatura corporal. Em aves a energia é 80% utilizada para a manutenção da homeostase e apenas 20% para a crescimento. No entanto, esse mecanismo só é eficiente em determinada faixa de temperatura, assim os aviários devem manter as temperaturas ambientais dentro das condições de conforto para as aves. A temperatura ideal para a criação de aves varia de 20 a 35°C dependendo da idade da ave (ABREU; ABREU, 2004).

O aumento da temperatura corporal é uma das causadoras de morte durante o verão, sendo que podem-se evidenciar as aves que estão passando por situação de estresse térmico pela apresentação de asas e bicos abertos. As aves, diferentemente de humanos não transpiram, podendo perder calor para o ambiente apenas pela ingestão de água fria e por meio da respiração. Níveis de umidade altos podem umedecer a cama, e favorecer o aparecimento

de fungos e bactérias que comprometem a saúde das aves. Restrições alimentares podem ser impostas para reduzir os efeitos do calor. Geralmente retira-se a alimentação durante o dia, para reduzir o calor metabólico das aves, e durante a noite, a luminosidade artificial intermitente se encarrega de acordar as aves para que elas ingiram ração, compensando a restrição do período diurno (SCOLARI, 2008).

Com exposições as diferentes condições ambientais os animais apresentam diferentes comportamentos em relação ao calor e frio, e podem ser vistos na Tabela 1.

Tabela 1 - Conduta das aves em relação as condições ambientais de calor e frio

Calor	Frio
Buscam sombra	Buscam sol
Buscam lugares frescos	Buscam lugares secos
Expõem-se ao vento	Refugiam-se do vento
Buscam pisos frios	Buscam pisos quentes
Aumentam o consumo de água	Diminuem o consumo de água
Diminuem o consumo de alimento	Aumentam o consumo de alimento

Fonte: Adaptado de ABREU e ABREU, 2004.

O desenvolvimento ideal das aves é atingido quando elas se encontram na zona de conforto térmico, não tendo sensações de calor ou frio (ABREU; ABREU, 2004).

Podemos observar na Figura 2, que na faixa de temperatura de A a A', a ave se encontra em conforto térmico, entre B e B' em uma zona de conforto térmico modesto em que a temperatura corporal e produção de calor permanecem constantes. Já entre C e B a homeostase atua mantendo a temperatura corporal mesmo com a redução da temperatura ambiental, e entre D e C já pode ser observada a hipotermia, zona em que a homeostase já não é suficiente para repor o calor necessário para a manutenção da vida. Entre B' e C' é possível notar que a homeostase atua para reduzir o calor corporal, mantendo a temperatura corporal constante. Já entre C' e D' o quadro já é de hipertermia, em que a temperatura corporal é diretamente afetada pelo calor ambiente. Quadros de temperatura inferiores a D e superiores a D' representam as mortes de aves por hipotermia e hipertermia, respectivamente.

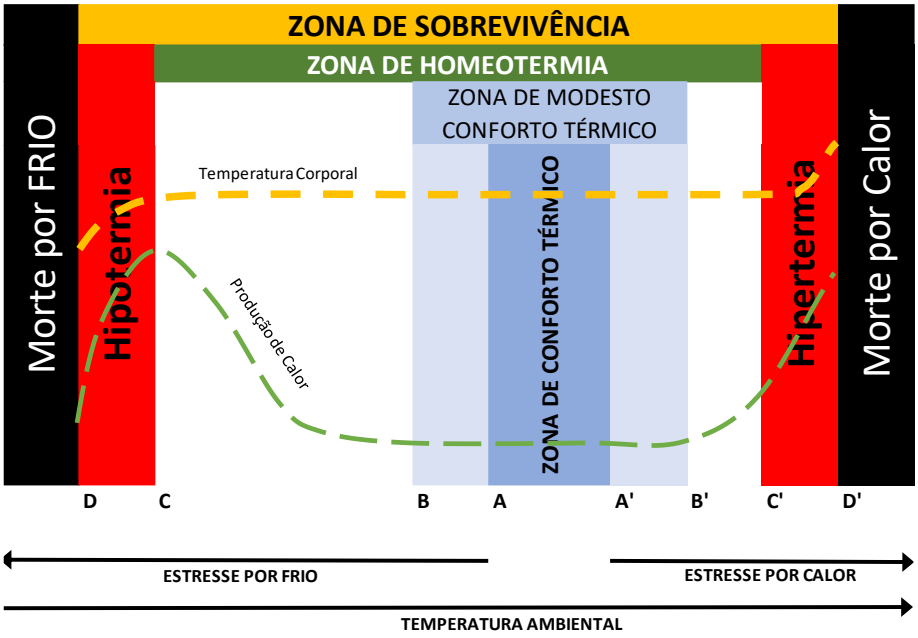


Figura 2 - Esquema das temperaturas ambientais críticas
Fonte: Adaptado de ABREU; ABREU, 2004.

O aumento da umidade do ar apresenta correspondência com elevação dos níveis de amônia no ambiente, redução da condição das características aceitáveis da cama de frango, aparecimento de lesões nos pés e pernas, além de queimaduras por amônia. A mudança de umidade pode influenciar rapidamente a qualidade da carcaça e peso corporal (WEAVER; MEIJERHOF, 1991).

As aves trocam calor com o ambiente por condução, convecção e radiação nas formas sensíveis e por evaporação e condensação nas formas latentes. A condução não é um processo muito favorecido, pelo fato das aves possuírem pequena área de contato dos pés com o solo. A convecção é o meio sensível mais eficiente, desde que haja gradiente de temperatura satisfatório para que ocorra a remoção do calor pelo ar; já na radiação, há dependência da orientação das superfícies consideradas, como o galpão ou o animal. Quando o gradiente de temperatura é pequeno, os meios sensíveis de troca de calor perdem a eficiência, e os modos de perda de calor latente são então priorizados. As perdas de calor passam então a ser por meio da evaporação na superfície da pele, mas principalmente pela evaporação proveniente do trato respiratório (BAÊTA, 1998).

É possível observar na Tabela 2, a temperatura e umidade relativa recomendadas para a criação de aves de acordo com a idade.

Tabela 2 - Temperatura e umidade recomendada na criação de aves em função da idade.

Idade (dias)	% de Umidade Relativa	Temperatura ºC
0	30-50%	32-33
7	40-60%	29-30
14	50-60%	27-28
21	50-60%	24-26
28	50-65%	21-23
35	50-70%	19-21
42	50-70%	18
49	50-70%	17
56	50-70%	16

Fonte: Adaptado de Cobb-Vantress, 2009.

4.4. Efeitos da exposição à luminosidade na avicultura (fotoperíodo)

A luz, na avicultura tem papel fundamental no controle da reprodução e idade da maturidade sexual das aves. Ao receber o estímulo luminoso, o organismo da ave ativa o relógio circadiano, tendo a sensibilidade máxima entre 10 e 15 horas. Os efeitos são principalmente provocados pela duração do tempo de exposição e a intensidade da luz. Como efeito da exposição a luz, fotorreceptores hipotalâmicos, são excitados e produzem efeitos nos neurônios hipotalâmicos, resultando na produção de hormônio liberador de gonadotrofina (GnRH), o hormônio luteinizante (LH) e o hormônio folículo estimulante (FSH). A produção de hormônio luteinizante só é estimulada pela exposição à luz em dias mais longos, que em condições naturais ocorre entre o solstício de inverno e o solstício de verão. Os principais objetivos ao se manejar a luz são: aumentar a produção de ovos, aumentar o tamanho dos ovos produzidos e sincronizar os períodos de produção das aves, esse último também evita diferenças de maturidade sexual entre fêmeas e machos (CFMV, 2011).

A suposição de que a exposição contínua ou quase total de iluminação das aves melhora o seu desempenho é errada, no período de até 7 dias após o nascimento é recomendada a exposição por até 23 horas de luz e 1 hora de escuridão, essa alternância mínima proporciona menor índice de mortalidade e morbidade. Após os sete dias um intervalo de escuridão de 4-6 horas deve ser considerado. Legislações locais devem ser observadas quanto aos programas de iluminação para criação de aves. Programas em que a alternância gradual de

duração e intensidade entre as mudanças de luminosidade podem trazer efeitos benéficos, como evitar a aglomeração nos comedouros quando as luzes acendem (ROSS, 2018).

Os programas de luz se mostram essenciais para o bem-estar dos lotes e para o desempenho das aves. Programas de luz que preconizam 6 horas de escuro melhoram o desenvolvimento do sistema imune das aves. Ao se planejar um programa, deve-se ter em conta as especificidades da região, condições climáticas, tipo de construção e os objetivos do produtor. É recomendada a incidência de 25 Lux medido na altura da ave para atingir o ganho de peso. Após a ave atingir 160 gramas, a intensidade deve ser reduzida gradativamente para 5-10 Lux (COBB-VANTRESS, 2009).

4.5. Componentes para o monitoramento e controle de temperatura, umidade, luminosidade e presença de amônia

4.5.1. Placa de desenvolvimento

A placa WeMos D1 Mini é uma pequena placa com 4MB de memória *flash* com microcontrolador ESP-8266EX. A tensão de operação é de 3,3 V, possui 11 pinos de entrada e saída digitais e 1 pino de entrada analógica (3,2 V máx.), com velocidade de *clock* (número de ciclos de cálculo por segundo) de 80/160Mhz, tamanho de 34,2 por 25,6 mm e pesando apenas 3 gramas (WEMOS 2021).

O microcontrolador ESP-8266EX já possui integração com Wi-Fi, possui gerenciamento de baixa energia embarcado, têm amplificadores de potência e de recepção de baixo ruído e design compacto. Requer reduzidos circuitos externos para operar, o que favorece a utilização em aplicações de internet das coisas (EXPRESSIF SYSTEMS, 2020).

Na Tabela 3 pode-se observar a configuração de cada pino de saída em relação aos pinos do microcontrolador ESP-8266EX.

Tabela 3 - Relação entre os pinos do microcontrolador ESP-8266EX e pinos de saída da placa WeMos D1 Mini

Pino WeMos	Função	Pino ESP-8266EX
TX	TXD	TXD
RX	RXD	RXD
A0	Entrada Analógica, máx 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

Fonte: Adaptado de WEMOS 2021.

A corrente máxima que pode ser utilizada na porta de entrada e saída do ESP-8266EX é de 12 mA, a voltagem típica de operação para saída nas portas de entrada e saída no ESP-8266EX é de 3,3 V (EXPRESSIF SYSTEMS, 2020).

4.5.2. Arduino versus ESP-8266EX

Na Itália em 2005, um grupo de estudantes, buscando encontrar uma solução mais apropriada para a utilização de microcontroladores disponíveis até então, desenvolveram um novo projeto e o batizaram de Arduino, em homenagem a um bar local. Sua versão inicial era volumosa e não possuía interface USB, mas as versões posteriores se mostraram muito baratas e de fácil utilização. Utiliza código aberto, o que permitiu a diversos fabricantes de placas utilizarem seu padrão para a fabricação de inúmeras placas similares. A placa Arduino é uma plataforma utilizada para colher dados do ambiente e tomar decisões baseadas em dados, podendo realizar múltiplos acionamentos. Estão presentes em impressoras 3D, robôs, drones, entre outros (AMARIEI, 2015).

As placas Arduino, apesar de serem bastante difundidas e muito utilizadas, não possuem Wi-Fi integrado, não possuem memória *flash* adicional integrada e também possuem baixo poder de processamento, 8 a 16 MHz de *clock*. A capacidade de se conectar com Wi-Fi e não possuir memória *flash* podem ser contornadas com o uso de placas auxiliares adicionais, mas elevariam o custo do projeto, e não há como elevar mais a taxa de processamento do microcontrolador para além de 16 MHz. Dada a especificidade do projeto, em que o microcontrolador deverá enviar as informações colhidas em diferentes protocolos, além de rodar um servidor HTTP internamente, a placa com o microcontrolador ESP-8266EX apresenta-se mais vantajosa, em detrimento do microcontrolador ATmega16U2, o mais comumente utilizado nas placas Arduino.

A Tabela 4, apresenta uma síntese das principais diferenças entre esses microcontroladores discutidos nesse trabalho.

Tabela 4 - Principais características dos microcontroladores ESP-8266EX e ATmega16U2

Característica	**ESP-8266EX**	**ATmega16U2**
Tensão de operação	2,5 até 3,6 V	2,7 até 5,5 V
Corrente de operação	Valor médio - 80 mA	21 mA em 16 MHz e 5 V
CPU	Tensilica L106 32-bit processor	AVR 8-Bit RISC CPU
Temperatura de operação	-40 até +125 ºC	-40 até +85 ºC
Frequência de operação	80 MHz / 160 MHz	8 MHz em 2,7 V e 16 MHz em 4,5 V
Wi-Fi	802.11 b/g/n até 72.2 Mbps em 2,4 até 2,5 GHz	Não possui
Modos Wi-Fi	Estação / Ponto de Acesso / Modo Promíscuo	Não possui
Memória FLASH	4 MB	16 KB
Memória RAM/SRAM	32 KB	512 B
Memória ROM/EEPROM	4 KB	512 B
Linhas de I/O	13 sendo 9 PWM	22 sendo 6 PWM
Conversor analógico-digital	1 com 10-bit de resolução	6 com 10-bit de resolução
Conversor digital-analógico	Não possui	Não possui
Watchdog Timer	1	1

Fonte: Adaptado de ATMEL e EXPRESSIF SYSTEMS, 2020.

4.5.3. Resistor Dependente de Luz

O sensor de luminosidade mais comum e de baixo custo é o LDR - Light Dependent Resistor, ou Resistor Dependente de Luz. É um sensor em que a resistência elétrica é modificada de acordo com o nível de exposição a luz (STEVAN; FARINELI, 2019).

O LDR modelo LDR GL5528 da empresa LIDA Optical & Eletronic Company, que pode ser visto na Figura 3, é uma das opções disponíveis no mercado.

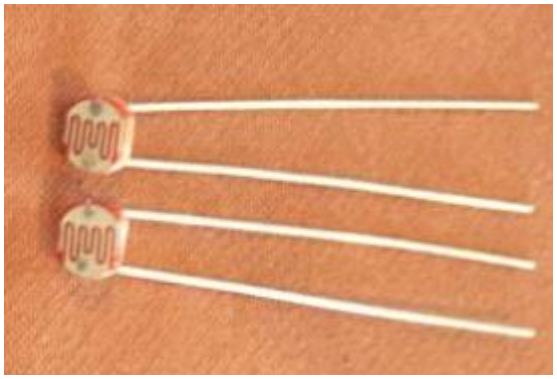


Figura 3 - Foto do sensor do tipo LDR
Fonte: LIDA OPTICAL & ELETRONIC CO.

É um sensor pequeno, encapsulado com epóxi, apresenta resposta rápida, possui alta sensitividade e desempenho confiável. O intervalo de temperatura de trabalho varia de -30 até 70 ºC, resistência à luz em 10 Lux (25 ºC) de 8 até 20 KΩ, resistência no escuro em 0 Lux 1 MΩ (mínimo), margem de erro de 0,1 entre 10 e 100 Lux, voltagem máxima de trabalho de 150 V e dissipação de energia a 25 ºC de 100 mW (LIDA OPTICAL & ELETRONIC CO.).

4.5.4. MQ-135 - Gas Sensor (Sensor de Gás)

O sensor de gás, tem como principal função o monitoramento de gases nocivos à saúde ou gases inflamáveis (STEVAN; FARINELI, 2019).

O sensor MQ-135, possui a capacidade de medir o gás amônia (NH_3), além de hidrocarbonetos aromáticos, monóxido de carbono (CO), dióxido de carbono (CO_2), tolueno (C_7H_8), sulfeto, álcool e fumaça. (SNS; WINSEN, 2015). Uma foto do sensor pode ser vista na Figura 4.

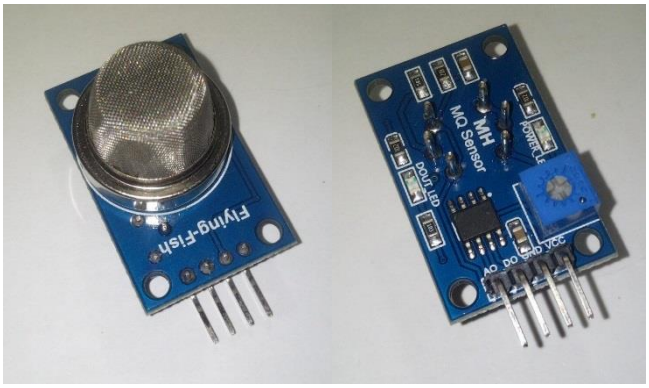


Figura 4 - Vista posterior e anterior do sensor de gás MQ-135
Fonte: Elaboração própria.

O sensor é do tipo semicondutor, encapsulado com baquelite e revestido por uma capa metálica. A voltagem da resistência de aquecimento é de 5±0,1V de corrente alternada ou contínua. O intervalo de detecção varia de 10 a 1000 ppm (gás amônia, tolueno, hidrogênio e fumaça). O tempo de pré-aquecimento para leituras estabilizadas é de 48 horas. O consumo da resistência interna é menor que 950 mW. O tempo de resposta após mudança abrupta de gases no ambiente é de aproximadamente 20 segundos. (WINSEN, 2015).

4.5.5. Sensor Digital de Temperatura e Umidade

Baseado em compostos cerâmicos, o sensor DHT22 fornece sinais digitais por meio de um controlador interno, fornecendo informações acerca da temperatura e umidade relativa do ar sendo utilizado para monitoramento. O range de operação deve ser observado a depender da aplicação (STEVAN; FARINELI, 2019).

As principais aplicações são em sistemas de climatização, desumidificadores, equipamentos de teste e inspeção, bens de consumo, setor automotivo, automação e controle, data loggers, aplicações domésticas, reguladores de umidade, aplicações médicas, estações meteorológicas, e outros controles de temperatura e umidade (AOSONG, 2019).

A faixa de medição de umidade varia de 0% a 100%, a faixa de medição de temperatura varia de -40 °C até 80 °C, a tensão de alimentação nominal é de 5 V, tensão mínima de 3,3 V e máxima 5,5 V, corrente de 2,5 mA em uso, precisão de medição de umidade de ±2% e de temperatura ±0,5 °C, resolução de 0,1% para umidade relativa e de

0,1 °C para temperatura e possui tempo de resposta de 2 segundos (STEVAN; FARINELI, 2019).

4.5.6. Relés

Um relé é um interruptor controlado eletricamente, quando alimentado por corrente elétrica aciona um eletroímã, o magnetismo produzido então puxa uma alavanca (armadura), que fecha o contato no circuito, é possível ouvir o mecanismo em funcionamento quando há o acionamento dos contatos (SHAMIEH; McCOMB, 2012).

O relé possibilita o isolamento do circuito controlado do circuito do microcontrolador, utiliza apenas um pino de controle para o acionamento e permite o controle de lâmpadas, eletrodomésticos e motores. Isso permite a um microcontrolador controlar um circuito de potência com carga elevada sem afetar o microcontrolador, como por exemplo uma cafeteira, que poderia ser acionada à distância por rede sem fio ou Bluetooth (STEVAN; FARINELI, 2019).

Ao se utilizar um relé, é necessário ter em conta que o equipamento que faz o acionamento, principalmente para o caso de corrente contínua, deve ser protegido. As linhas de força do campo magnético da bobina interna do relé começam a se retrair após o corte de corrente, ao desligar o dispositivo. Nesta retração, as espiras das bobinas do relé são cortadas gerando então a indução de uma tensão, e esta terá polaridade inversa àquela que a criou e podendo atingir valores altos. O valor da tensão dependerá da velocidade de contração no tempo (di/dt) e da indutância da bobina (L). Se o equipamento que fez o acionamento não estiver preparado para receber essa tensão, poderá ser danificado (BRAGA, 2009).

Diodos colocados em paralelo com um equipamento eletrônico sensível protegem o equipamento contra grandes elevações de voltagem (SHAMIEH; McCOMB, 2012).

A corrente de acionamento para o relé RAS-1210 é de 30 mA em 12V (SUN HOLD).

4.5.7. Transistor

Um transistor NPN é a junção de duas partes compostas por semicondutores tipo N e uma parte ao centro de semicondutor tipo P, cada parte contém um terminal. Para controlar o transistor, controla-se a junção base-emissor, controlando assim o fluxo de corrente elétrica que atravessa o transistor. O ganho de corrente da base para o coletor é limitado, há um limite de elétrons livres disponível no emissor e se a voltagem for aumentada na base, esse limite é então atingido, ficando o transistor saturado. Quando saturado, o transistor funciona como uma válvula, deixando fluir a corrente, é como se um fio estivesse ligado diretamente entre o coletor e o emissor, nessa configuração o transistor funciona como um interruptor eletricamente controlado. Um dos seus principais usos é para acionamento de dispositivos que necessitam de uma maior corrente, em relação a corrente disponível no dispositivo acionador, sendo comumente utilizados para o acionamento de relés (SHAMIEH; McCOMB, 2012).

A corrente no coletor Ic em corrente contínua para o transistor BC-548 é de 100 mA, a voltagem de saturação para a base V_{BE}(sat) para Ic em 100 mA, com corrente de base I_B em 5 mA, é de 0,9 V para o transístor BC-548 (FAIRCHILD, 2014).

4.5.8. Relógio de Tempo Real

O RTC (Real Time Clock), relógio de tempo real é um dispositivo eletrônico de baixa energia que atua como um relógio, armazenando a hora, minuto, segundo, e o dia da semana, dia do mês, mês e ano atuais. O tempo é controlado por um cristal oscilador de quartzo que opera na frequência de 1 Hz até 32.768 kHz. O dispositivo opera na faixa de 4,5 V até 5,5 V em corrente contínua. O RTC também faz uso de uma bateria, que é acionada quando ocorre a interrupção da energia que alimenta o módulo, passando a fornecer energia para que a memória e o oscilador continuem operando, possibilitando que o dispositivo armazene os dados relativos ao tempo mesmo nessa condição (MAXIM INTEGRATED, 2015).

4.5.9. Alarme ou buzina

O *buzzer*, ou buzina, possui função de alarme, consiste em um dispositivo piezoelétrico encapsulado que vibra e produz som quando acionados por corrente. Existem dois tipos principais, os passivos, que funcionam como um autofalante, que necessitam de modulação do sinal para produzir sons, e os ativos, que possuem internamente um oscilador de sinal que produz um tom fixo. Os *buzzers* devem ser acionados por relé ou transistores, devido a corrente necessária para acionamento poder ser maior que a corrente disponível no microcontrolador. (STEVAN; FARINELI, 2019).

O *buzzer*, possui tensão de operação de 4 até 8 V em corrente contínua, emite som ≥85 dB à 10cm em 6 V, com frequência de ressonância de 2300 ± 300 Hz, tom contínuo, peso de 2 gramas e temperatura de operação de -25 °C até 80 °C (PRO-SIGNAL, 2016).

4.6. IoT (Internet of Things) - Internet das Coisas

Até pouco tempo, a forma mais habitual de interagir com a internet dava-se por meio de um computador, e utilizando-se de um navegador procurava-se por assuntos de interesse, tendo textos e imagens reproduzidos na tela. Links até então podiam ser utilizados para levar a outras páginas com outros conteúdos relacionados. Hoje em dia, por meio de IoT (Internet of Things), ou Internet das Coisas, vários dispositivos podem ser utilizados, com diferentes tipos de sensores e até mesmo eletrodomésticos, podem ser conectados mais facilmente a internet. Dispositivos dotados de acelerômetros podem até mesmo monitorar aspectos da saúde das pessoas por meio de dispositivos vestíveis. Sistemas de automação residencial se tornaram possíveis, podendo controlar aspectos como iluminação e aquecimento de uma casa por meio de um smartphone (MONK, 2018).

4.7. Protocolo MQTT

O protocolo MQTT - Message Queuing Telemetry Transport, ou Transporte de Enfileiramento de Mensagens por Telemetria, é um protocolo de transmissão de mensagens que foi desenvolvido pela IBM, com o foco na utilização para transmissão de dados em dispositivos baseados em IoT, Internet das Coisas. Esse protocolo utiliza o protocolo TCP/IP para envio e recepção das mensagens. Tem por principal característica o modo de funcionamento de publicação e assinatura, em que um dispositivo emissor, envia uma mensagem a um servidor, chamado de *broker*, que disponibiliza essa mensagem para os assinantes, que podem coletar os dados em um momento posterior ao do envio para o *broker*. É possível também que uma informação tenha vários assinantes, sendo uma mesma mensagem disponibilizada para muitos dispositivos. Desse modo o emissor e o receptor são desacoplados entre si, tanto em espaço quanto no tempo, sendo ideal para utilização em redes não confiáveis (SOARES, 2019).

Abaixo, na Figura 5, pode-se observar um esquema do tráfego de dados através do protocolo MQTT.

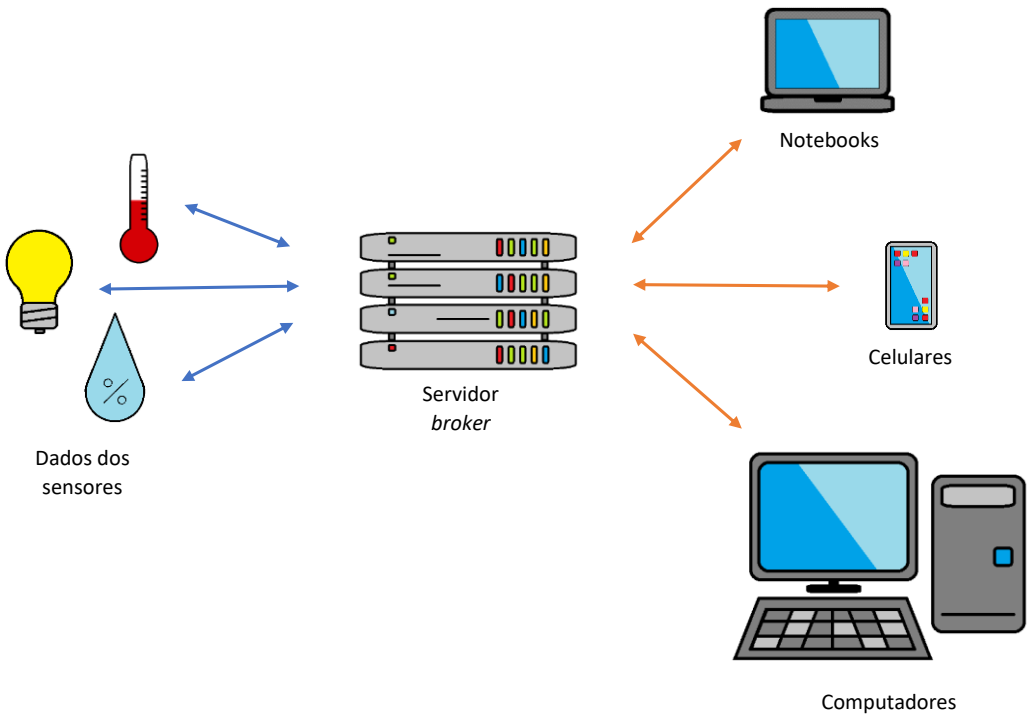


Figura 5 – Esquema de transmissão de dados no protocolo MQTT
Fonte: Elaboração própria.

4.8. Ambiente de desenvolvimento

IDE (Integrated Development Environment) ou Ambiente de Desenvolvimento Integrado, é um programa de computador que possui ferramentas para criar outros programas e é composto geralmente por um editor de código-fonte, em que há entrada dos códigos-fonte, um compilador, que traduz o código-fonte para a linguagem de máquina do respectivo sistema operacional que executará o programa e também um debugger, que é utilizado para testar a execução dos códigos e sinalizar a localização de inconsistências no código de entrada. Existem várias IDEs diferentes no mercado, para as mais variadas linguagens de programação, elas possuem funcionalidades específicas e podem ser executadas nos diferentes tipos de sistema operacional (RED HAT, 2019).

Para programar o microcontrolador ESP-8266EX, é possível utilizar a linguagem MicroPython e Arduino (WEMOS, 2021).

Na linguagem Arduino não há referências completas a todos os comandos do C++, sendo possível apenas a utilização de parte do C++ (SUHANKO, 2019).

A estrutura principal de um código de programação para utilização em Arduino é dividida em duas parte principais, a função setup() {} e a função loop() {}, na função setup são carregadas as bibliotecas, variáveis e funções que serão utilizadas ao longo da execução do programa, nesta etapa de execução também são configuradas as condições inicias das portas de entrada e saída do microcontrolador, ocorrendo uma única vez e sempre que o dispositivo for energizado, já na função loop, o trecho de código é executado continuamente, se repetindo desde o início, a partir da primeira linha, sempre que alcança seu final, podendo utilizar as bibliotecas, variáveis e funções configuradas no setup (PROTTO, 2020).

4.9. Licença de uso de software

As pessoas que tiverem acesso a esse código podem de forma geral, ter a liberdade de utilizar o software para qualquer finalidade, compartilhar com quem desejar, modificar de acordo com suas necessidades e compartilhar as mudanças que fez. As pessoas não podem assumir a autoria do código e nem solicitar ao autor qualquer tipo de garantia sob nenhuma hipótese (GNU, 2022).

5. MATERIAIS E MÉTODOS

5.1. Materiais utilizados

5.1.1. Placa de desenvolvimento

Esta pequena placa foi a escolhida para este projeto, por apresentar características integradas importantes, como o Wi-Fi e memória interna flash, necessárias para a comunicação e armazenamento dos dados, respectivamente. É responsável pelo processamento de todos os sinais dos sensores, processamento, armazenamento e rotinas de controle. Abaixo na Figura 6, pode-se observar a foto da placa WeMos D1 Mini equipada com o microcontrolador ESP-8266EX.

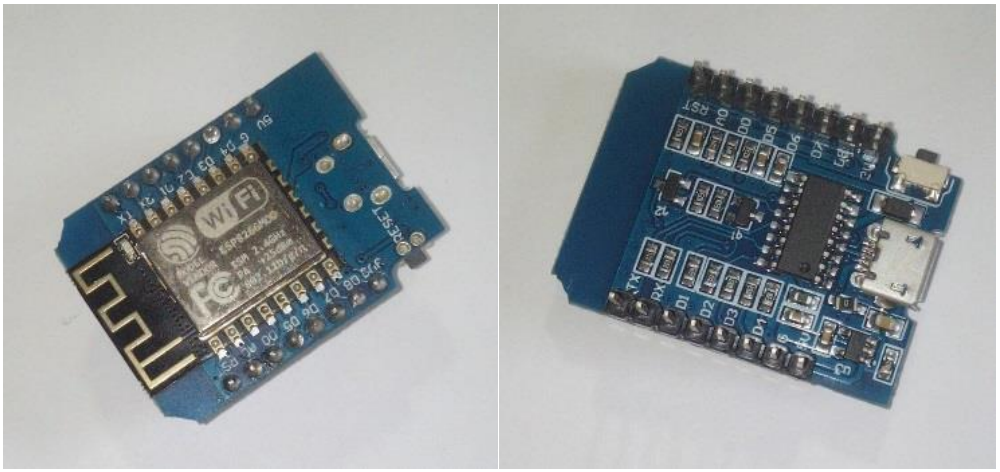


Figura 6 - WEMOS D1 mini
Fonte: Elaboração própria.

5.1.2. Resistor Dependente de Luz

Na ficha técnica desse componente, é fornecido um gráfico com o comportamento da resistência em função da luminosidade, em escala logarítmica, como pode ser observado na Figura 7.

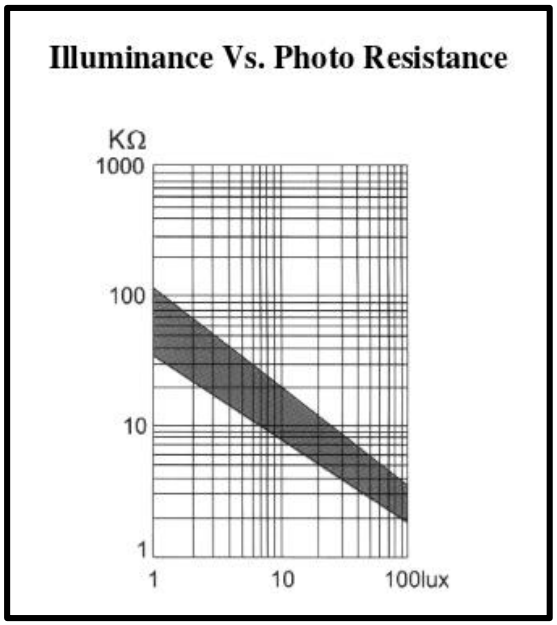


Figura 7 - Gráfico da relação entre resistência e luminosidade
Fonte: LIDA OPTICAL & ELETRONIC CO.

Para trabalhar com unidade de luminosidade na programação do microcontrolador, foi necessário aplicar o método dos mínimos quadrados, assim podemos obter uma função matemática que se aproxime do comportamento do gráfico. No Apêndice C, constam os dados utilizados para a realização dos cálculos.

Após a aplicação do método, foi obtido R^2 de 0.9901, e em seguida foi feita a reversão da linearização, obtendo-se a seguinte fórmula:

$$y = 2.242.931.x^{-1,3011}$$

Onde:

y: luminosidade em Lux
x: resistência (Ω)

O sinal do sensor recebido na porta analógica da placa controladora é uma variação de tensão, sendo necessária sua conversão em resistência. Isso é feito utilizando-se da fórmula de divisão de tensão que é baseada na lei de Ohm:

$$U = R.I$$

Onde:

U: tensão em volts
R: resistência (Ω)
I: corrente (A)

A fórmula derivada para divisão de tensão utilizada foi a seguinte:

$$Usaída = Uentrada\frac{R2}{R1 + R2}$$

Onde:

Usaída: tensão em volts lida no microcontrolador
Uentrada: tensão em volts que alimenta o LDR
R1: resistência conhecida do divisor de tensão (Ω)
R2: resistência do sensor LDR (Ω)

Conhecendo-se então o valor da resistência do LDR, pôde-se obter a luminosidade em Lux que incide sobre o sensor no momento da leitura, utilizando-se a fórmula obtida com o método dos mínimos quadrados.

5.1.3. Sensor Digital de Temperatura e Umidade

O sensor DHT22 possui características que o torna adequado, trabalhando em uma ampla faixa de temperatura e umidade, operar com nível baixo de corrente e pouca variação no nível de precisão como visto na Figura 8. A Figura 9, apresenta uma foto do sensor em questão, instalado no dispositivo.

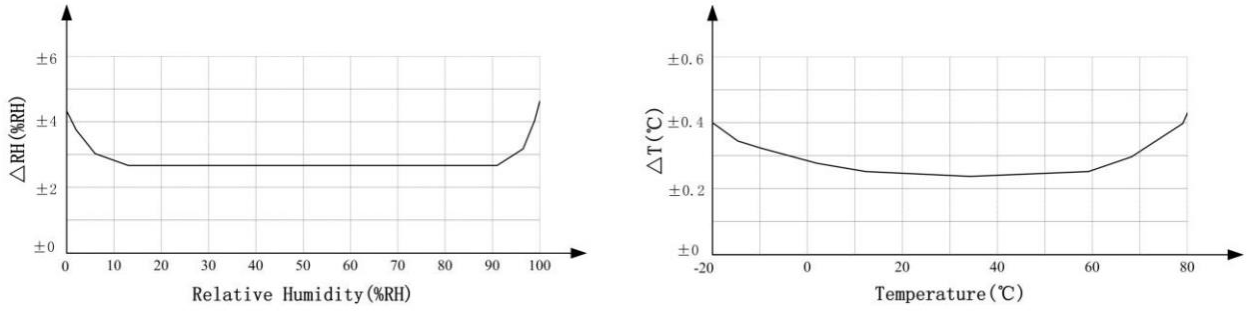


Figura 8 - Precisões do sensor de temperatura e umidade
Fonte: AOSONG, 2019.

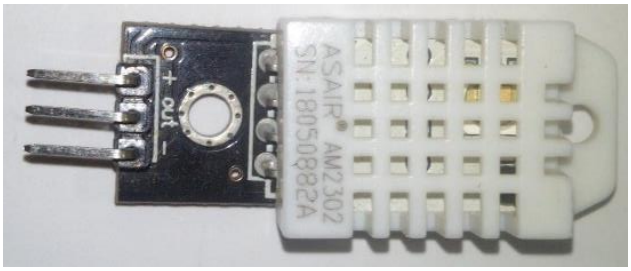


Figura 9 - Sensor de temperatura e umidade
Fonte: Elaboração própria.

5.1.4. MQ-135 - Gas Sensor (Sensor de Gás)

Na ficha técnica desse componente, é apresentado um gráfico que relaciona resistência com a concentração medida no sensor em ppm. Na Figura 10, é apresentado o gráfico fornecido pelo fabricante. De acordo com a ficha técnica, R_0 representa a resistência do sensor no ar limpo e R_S a resistência do sensor apresentada na presença de gás detectável.

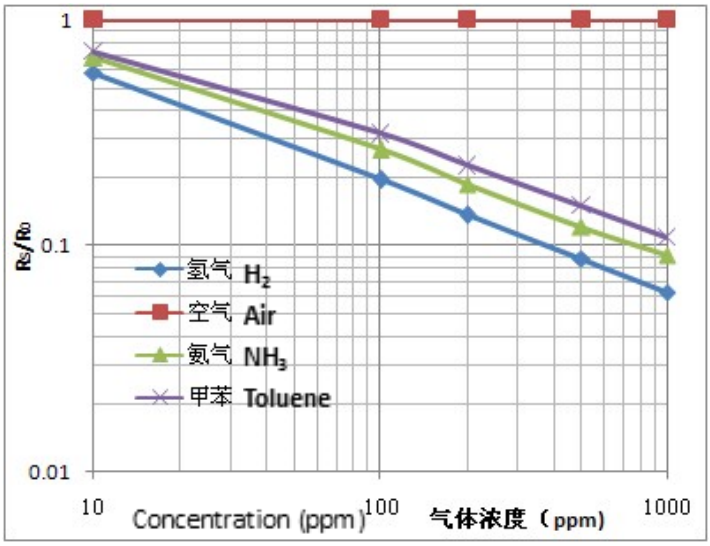


Figura 10 - Gráfico da relação entre resistência e concentração
Fonte: WINSEN, 2015.

Assim como para o LDR, nesse caso também foi necessário estabelecer uma fórmula matemática para expressar a variável desejada, nesse caso a concentração em ppm em função do valor da resistência encontrada. No Apêndice D, constam os dados utilizados para a realização dos cálculos. Também para esse gráfico, foi adequada a aplicação do

método de mínimos quadrados, obtendo-se então a seguinte fórmula após a reversão da linearização:

$$y = 2{,}048679292.\, x^{-0{,}447408181}$$

Onde:

y: concentração (ppm)
x: R_S / R_0 (Ω)

A fórmula apresenta R^2 de 0.9982, que representa uma boa aproximação entre a fórmula utilizada e os dados lidos no gráfico fornecido pelo fabricante.

Para o caso do sensor de gás, utilizou-se uma porta digital para a leitura, que informa em sinal booleano a presença de gás, e sua regulagem é feita por meio de um resistor variável presente na placa do sensor, ou *shield*, sendo utilizado o método acima mencionado para a calibração da utilização da porta digital. Ao atingir a concentração desejada, lida na porta analógica para a emissão do sinal digital, o resistor variável, potenciômetro, deve ser acionado até que haja a mudança de estado na leitura para o desejado. Para realizar os testes do projeto o objetivo do sinal digital foi configurado para 20 ppm de Amônia.

5.1.5. Relés

No projeto, os relés são utilizados para realizar o acionamento dos dispositivos, como por exemplo: ventiladores, exautores, lâmpadas, desumidificadores, entre outros, que poderão ser utilizados pelo produtor para realizar o controle das diferentes variáveis envolvidas no aviário. Um ventilador pode ser acionado para reduzir a temperatura e remover a umidade, enquanto uma resistência pode ser acionada para elevar a temperatura.

Os relés utilizados, levam em conta que o produtor possa ter equipamentos de 127 V e 220 V, então optou-se por componentes que trabalham no range de 120 V até 250 V, maximizando as opções de utilização de equipamentos e a região em que estiver a produção, pois em algumas regiões do país predomina a utilização de redes de 220 V e em outras, as redes de 127 V. Abaixo na Figura 11, pode ser observada uma foto dos 4 relés utilizados no projeto.

Figura 11 - À esquerda os relés utilizados e detalhe das informações à direita do RELÉ RAS-1210
Fonte: Elaboração própria.

Para evitar danificar o microcontrolador com sobretensões, diodos foram adicionados ao projeto, sendo ligados em paralelo aos contatos da bobina de acionamento do relé.

5.1.6. Transistor

Neste projeto foi necessário utilizar relés para realizar o acionamento de atuadores, a corrente necessária para acionar os relés é superior a corrente disponível nas portas de entrada e saída do microcontrolador ESP-8266EX, por esse motivo a utilização de transistor para acionamento do relé foi uma ótima solução.

A corrente máxima disponível nas portas do ESP-8266EX de 12 mA, sendo insuficiente para realizar o acionamento. Como a corrente no coletor do BC-548 é superior a corrente de acionamento do relé, o transistor BC-548 atende a necessidade de corrente para o acionamento do relé no projeto.

A voltagem de operação das portas do ESP-8266EX é de 3,3 V e a corrente é de 12 mA, sendo superiores a 0,9 V e 5 mA do acionamento da base do transistor, provocando assim a saturação do transistor e fazendo-o trabalhar como uma chave eletrônica. Abaixo na Figura 12, uma foto dos quatro transistores utilizados no projeto.

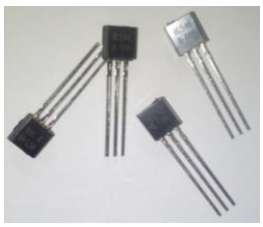


Figura 12 - Transistores NPN BC-548
Fonte: Elaboração própria.

5.1.7. Relógio de Tempo Real

Nesse projeto o RTC DS1307 foi incluído para possibilitar o registro das informações referentes ao funcionamento do equipamento ao longo do tempo, possibilitando verificar o comportamento das variáveis monitoradas. Abaixo na Figura 13, podemos ver a foto de um dispositivo igual ao utilizado no projeto.

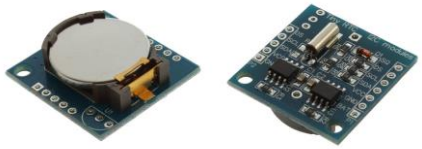


Figura 13 - RTC DS1307
Fonte: <https://www.filipeflop.com/blog/relogio-rtc-ds1307-arduino/>.

5.1.8. Alarme ou buzina

Para o projeto foi escolhido o *buzzer* do tipo ativo, pela facilidade de utilização e por demandar menos componentes eletrônicos na placa do projeto. O módulo adquirido já continha em seu circuito o transistor necessário para acionamento do *buzzer*. Seu baixo custo também foi levado em conta, assim como o espaço utilizado por ele dentro da caixa. No projeto, sua função é sinalizar por meio de um alarme intermitente, a certa distância, que há presença de gás Amônia acima do limite determinado na área de produção. Na Figura 14, uma foto do *buzzer* utilizado no projeto.

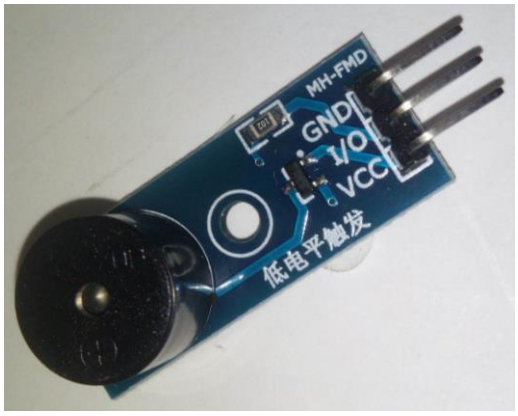


Figura 14 - Buzzer, ou buzina
Fonte: Elaboração própria.

5.2. Esquema elétrico

O esquema elétrico foi elaborado tendo em conta algumas premissas; que o dispositivo tivesse alimentação bivolt (110/220 V), e que os atuadores teriam ligações independentes entre si, possibilitando a mescla de dispositivos 110 V e 220 V, com isso o dispositivo pode ser utilizado na ampla maioria de lugares com energia elétrica disponível e diferentes tipos de dispositivos, mesmo que com tensões diferentes, podem ser controlados com um único dispositivo controlador.

Para a alimentação, foram utilizadas 3 tensões em corrente contínua diferentes, 3,3 e 5 V para a placa controladora e 12 V para a placa de atuadores. Sendo todas obtidas de fontes do tipo bivolt (110/220 V) amplamente encontradas no mercado, do tipo visto na Figura 15.

Figura 15 - Fontes de 3,3V, 5V e 12V utilizadas
Fonte: Elaboração própria.

O funcionamento do dispositivo se dá tendo como principal componente o microcontrolador Wemos D1 Mini, com o chip ESP-8266EX, que é alimentado em 5 V diretamente da fonte em seu pino 5 V. Sua porta 3,3 V é utilizada em série com um LED vermelho, LED5 e seu resistor R10, que tem função de limitar a corrente até o LED, e que servem para indicar que o microcontrolador está devidamente alimentado. Em sua porta A0 está conectada o LDR, capturando os sinais analógicos e convertendo-os em informações sobre a luminosidade. Na porta D0 encontra-se a conexão com o sensor MQ-135, recebendo o sinal digital indicando a presença de gás amônia no ambiente. Em D5, D6, D7 e D8, são enviados os sinais digitais para acionamento dos transistores que controlam os relés, enviando sinais de 3,3 V para a base dos transistores, deixando-os saturados. As portas D1 e D2 realizam a comunicação com o RTC, recebendo via sinais digitais os dados acerca do tempo atual. Já em D3, são recebidos os dados digitais referentes as medições de temperatura e umidade relativa do ar, fornecidas pelo sensor

DHT22. A porta D4, fornece saída digital para o acionamento da buzina, ou *buzzer*. No pino GND é feita a ligação ao negativo da fonte.

O resistor R5 tem o papel de divisor de tensão, rebaixando o sinal recebido na porta A0 do microcontrolador. Os resistores R1, R2, R3 e R4 fazem a proteção dos transistores. Os resistores R6, R7, R8 e R9, são de proteção dos LEDs 1, 2, 3 e 4, LEDs verdes, que sinalizam o acionamento dos 4 relés para os atuadores. Q1, Q2, Q3 e Q4, são os transistores que acionam os relés K1, K2, K3 e K4, respectivamente. Os diodos D1, D2, D3 e D4 são ligados em paralelo as bobinas dos relés, para proteção de sobrecarga nos transistores. A fonte de 3,3V é responsável pela alimentação do LDR, DHT22 e *buzzer*, A fonte de 5V é responsável pela alimentação do microcontrolador Wemos D1 Mini, MQ-135 e RTC DS1307. A fonte de 12V fornece alimentação com exclusividade para o acionamento das bobinas internas dos relés. Os relés K1, K2, K3 e K4, receberão as ligações dos dispositivos a serem utilizados como atuadores nas portas normalmente abertas. Os três negativos das três fontes de energia são ligados entre si.

Abaixo, na Figura 16, é apresentado o esquema elétrico utilizado. A representação do esquema foi elaborada utilizando o software gratuito e aberto Fritzing, que pode ser obtida pelo endereço eletrônico <https://fritzing.org/>.

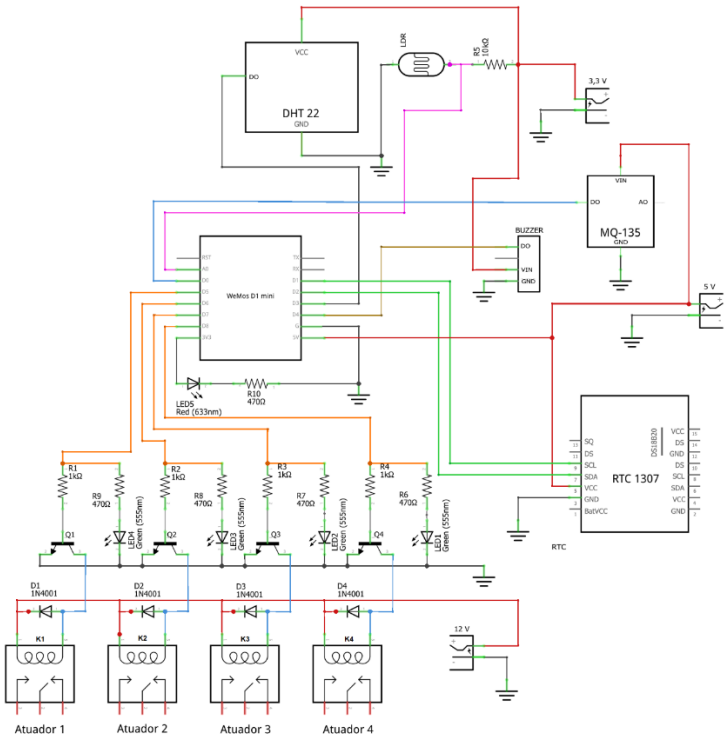


Figura 16 - Esquema elétrico para o sistema de monitoramento e controle proposto
Fonte: Elaboração própria.

5.3. Prototipagem

Após a etapa da confecção do esquema elétrico, pôs-se em prática a etapa de prototipagem, para averiguar se o esquema elétrico funcionava como o esperado. Nessa etapa foi utilizada uma placa de prototipagem, ou *protoboard*. O esquema elétrico pôde então ser dividido em duas partes físicas, uma parte para os sensores e controles e outra parte contendo apenas os circuitos para os atuadores, dessa forma foi possível um melhor arranjo dos componentes que posteriormente possibilitou que todas as peças fossem acondicionadas de forma satisfatória, dentro da caixa escolhida para o projeto. Na Figura 17, podemos ver um desenho do protótipo utilizado para testagem da interação entre os diferentes componentes feito em bancada. O desenho da prototipagem, assim como o esquema elétrico, foi elaborado com o software Fritzing.

Essa etapa se mostrou muito importante, pois nela foi possível identificar alguns erros de ligação no projeto inicial, que posteriormente foram corrigidos no esquema elétrico.

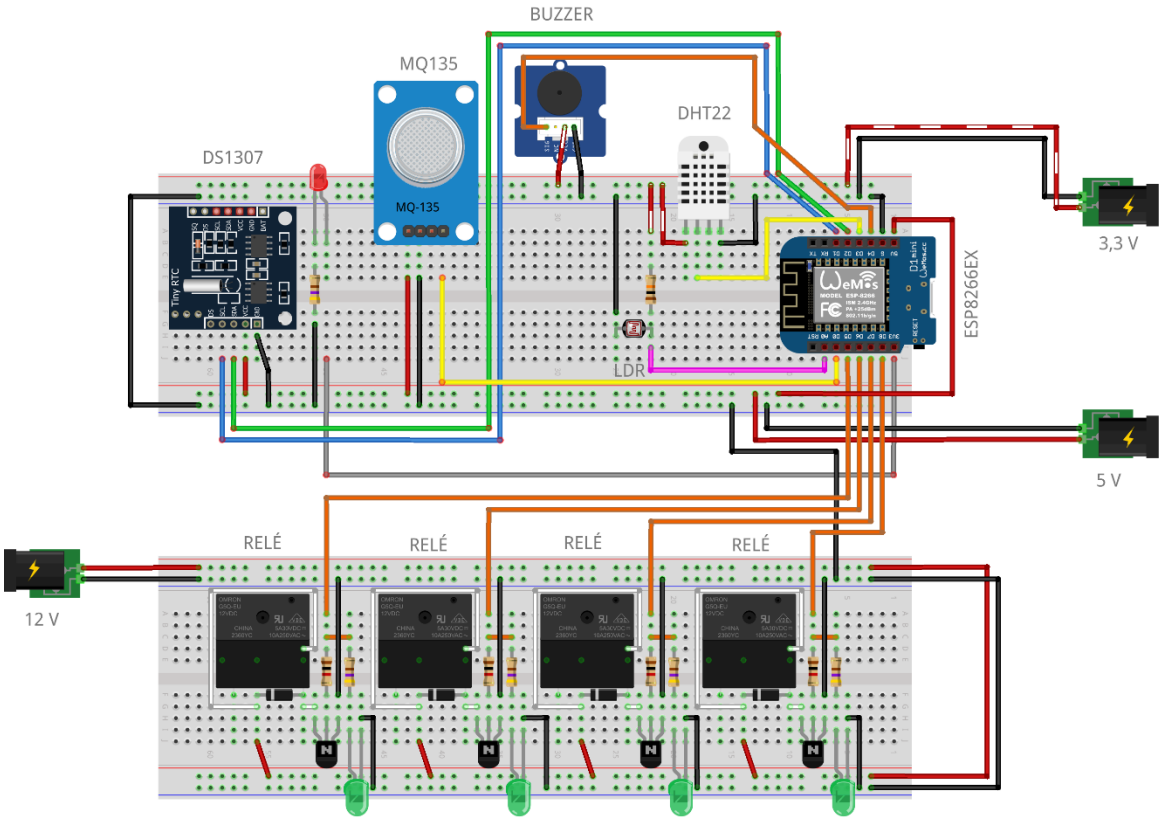


Figura 17 - Prototipagem
Fonte: Elaboração própria.

5.4. Montagem do Hardware

Após a etapa de prototipagem, foi colocada em prática a etapa de construção das placas de controle e de atuadores, soldando os componentes nas placas perfuradas de fenolite. Na Figura 18, temos uma foto do protótipo para testes da placa de controle dos atuadores. Já na Figura 19, o detalhe durante os testes da placa de atuadores.

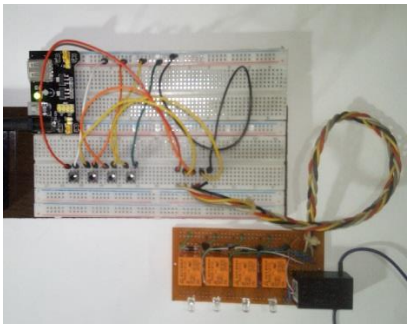


Figura 18 - Testes da placa de atuadores em bancada
Fonte: Elaboração própria.

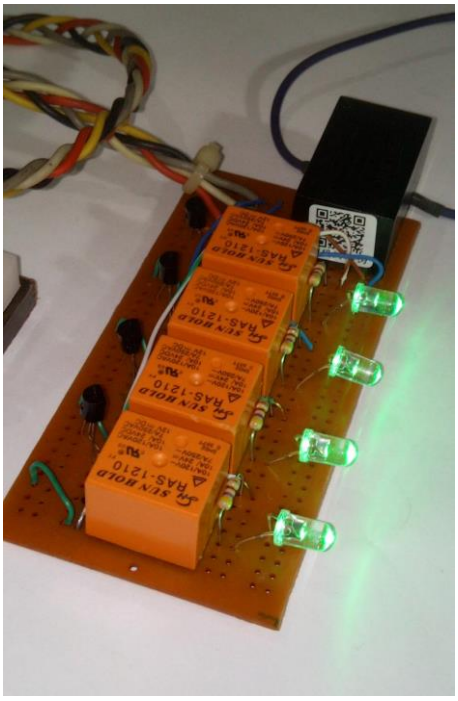


Figura 19 - Detalhe dos testes da placa de atuadores
Fonte: Elaboração própria.

Abaixo na Figura 20, em detalhe, a placa de controle após a soldagem dos componentes e fios.

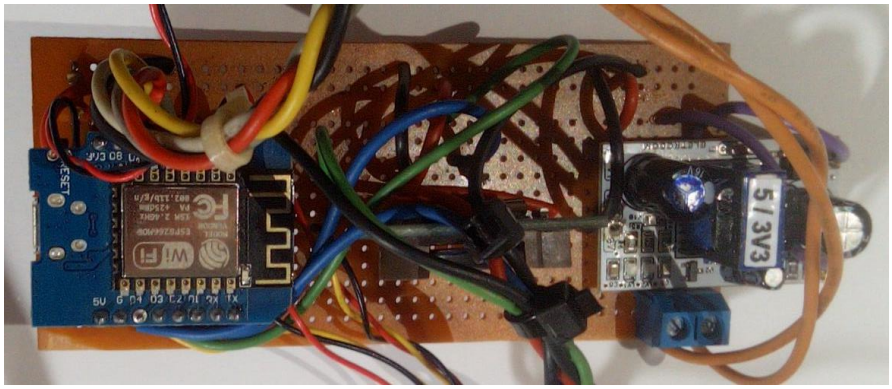


Figura 20 - Detalhe da confecção da placa de controle
Fonte: Elaboração própria.

Tendo em vista o design de montagem escolhido, o equipamento deverá ser posicionado na altura da cabeça das aves, para que os sensores capturem os dados com menos perturbações ambientais possíveis. Em detalhe na Figura 21, o posicionamento dos sensores na tampa do dispositivo, na imagem é possível visualizar no canto superior esquerdo o sensor de gás MQ-135, no canto superior direito o sensor de temperatura e umidade DHT22, no canto inferior esquerdo o sensor de luminosidade LDR 5528 e o dispositivo de alarme, buzzer ativo no canto inferior direito, posicionamentos esses para a vista da esquerda, a interna ao dispositivo.

Figura 21 - Detalhe do posicionamento dos sensores e do alarme na tampa do dispositivo, vista interna à esquerda e externa à direita
Fonte: Elaboração própria.

5.5. Placa de circuito impresso

O software Fritzing, utilizado para a elaboração do esquema elétrico e prototipagem também permite a elaboração de projetos de confecção de placas de circuito impresso, ou mais comumente conhecida como PCB (Printed Circuit Board). O intuito da utilização de PCBs é a redução de custos na produção de produtos eletrônicos e miniaturização dos projetos. Para esse projeto, que tem como finalidade ser apenas um protótipo, não foi encomendada a PCB, mas seu projeto foi elaborado no software e as imagens necessárias para adquirir as PCBs encontram-se no Apêndice B deste trabalho, abaixo na Figura 22, é possível ver uma ilustração de como essa placa poderia ser montada.

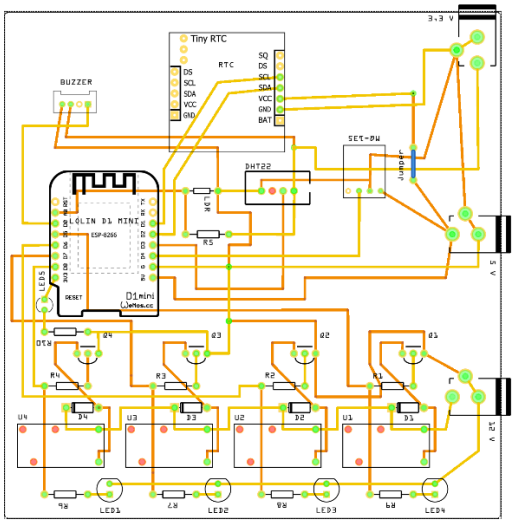


Figura 22 - Projeto de PCB para a proposta
Fonte: Elaboração própria.

5.6. Programação (Software)

5.6.1. Funcionamento do Programa

O software de modo geral, realiza a coleta das variáveis de interesse, temperatura, umidade, luminosidade e presença de amônia, através de sensores, armazenando os dados na memória interna do microcontrolador. Em seguida disponibiliza esses dados por meio de rede Wi-Fi, via protocolo HTTP, e envia os dados também por meio de protocolo MQTT pela internet, para o serviço no site adafruit.io. No microcontrolador esses dados são

processados e de acordo com um critério lógico, são acionados atuadores, por meio de relés, a fim de realizar as correções pertinentes no ambiente de produção. Por exemplo, aumentando a temperatura do ambiente de produção por meio do acionamento de uma resistência, para o caso do ar estar demasiadamente frio, ou ligar a iluminação para que as aves possam dispor de luz por um período maior do que o período de iluminação natural.

Neste projeto há características funcionais que o categorizam como um IoT, um dispositivo pequeno, portátil, que se conecta diretamente a internet quando disponível e transmite dados que podem ser coletados por outros. Pode receber comandos a distância e atuar diretamente sobre o sistema de produção.

5.6.2. Protocolo HTTP (Hypertext Transfer Protocol)

O principal acesso as opções de software no protótipo é realizado por meio do protocolo HTTP, através dos endereços de IP fornecidos pela rede Wi-Fi local, ou pelo endereço padrão do ponto de acesso do microcontrolador ESP-8266EX, por padrão no endereço 192.168.4.1. Ao acessar o endereço por meio de qualquer navegador, informações importantes são disponibilizadas:

- Data e hora atual do sistema, que foram lidas do RTC no protótipo
- Temperatura atual, última leitura do sensor DHT22
- Umidade relativa atual, última leitura do sensor DHT22
- Luminosidade, última leitura do sensor LDR
- Presença de Amônia, última leitura do sensor MQ135
- Modo de operação, se naquele momento está definido para o modo manual ou automático
- *Buzzer*, se os sinais sonoros estão habilitados
- Botão atualizar, para recarregar a página atual
- Botão ver dados, redireciona para o link de download dos dados contidos no arquivo dados.txt na memória do dispositivo
- Botões de atuadores de 1 até 4, possibilitam o desligamento ou religação instantânea das portas dos atuadores

- Botão para alternar modo, possibilita a troca do modo de operação automático para manual e vice-versa
- Botão ligar/desligar buzina, permite habilitar ou desabilitar o funcionamento da buzina
- Botão deletar dados, permite a deleção do arquivo dados.txt, de modo a preservar a memória do dispositivo para novas gravações após realizar o download dos dados pregressos
- Botão formatar SPIFFS, permite a formatação da memória do dispositivo, para o caso de ocorrência de erros de leitura e escrita
- Botão enviar por serial, permite o envio dos dados do arquivo dados.txt diretamente por meio da porta serial conectada por meio de um cabo USB
- Botão reiniciar, permite reiniciar o dispositivo remotamente, por meio da rede, evitando a necessidade de corte de energia

A opção modo de operação foi adicionada ao software, possibilitando que as rotinas que controlam os atuadores sejam momentaneamente desabilitadas e que o estado das portas dos atuadores se mantenham os mesmos. O estado das portas pode ser alterado por meio dos botões de atuadores na página, apenas localmente por acesso direto no protocolo HTTP. A possibilidade de mudança no estado por protocolo MQTT é possível, mas não foi realizada, para que não fosse alterado por meio de acessos não autorizados. Essa opção foi implementada para que caso seja necessário realizar alguma manutenção, troca de equipamento ou desligamento por qualquer motivo de algum equipamento ligado ao protótipo, isso possa ser feito. Inclusive preservando a segurança de quem for fazer alguma manutenção em equipamentos ligados ao protótipo, como ventiladores ou resistências.

A possibilidade de desligar a buzina foi incluída, pois se de alguma forma os sons de leitura estiverem incomodando os animais ou as pessoas que ali se encontram, pode ser desabilitado facilmente.

5.6.3. Protocolo MQTT

No projeto, o microcontrolador, por meio de rede Wi-Fi, quando disponível no local de utilização, faz a transmissão dos dados para o serviço MQTT do site io.adafruit.com, já

previamente configurado para fornecer assinatura, para que outros dispositivos possam fazer a consulta acerca da situação do ambiente de produção. O objetivo é possibilitar ao produtor receber os dados em seu celular, para acompanhar a situação em qualquer lugar, por meio de internet móvel celular ou conexão por Wi-Fi. A consulta também pode ser realizada por qualquer computador conectado à internet, sem a necessidade de instalar qualquer aplicativo, apenas acessando o site da Adafruit.

Há na internet outros sites que disponibilizam o serviço de MQTT, porém o da Adafruit foi escolhido, por ser de ampla utilização, fácil implantação e também por seu plano gratuito atender as necessidades do projeto.

5.6.4. Algoritmo

O algoritmo foi programado utilizando o programa Arduino IDE na sua versão 1.8.19, e as bibliotecas necessárias foram todas adicionadas do próprio repositório de bibliotecas no Arduino IDE sendo: Adafruit MQTT Library (Adafruit 2.4.2), Adafruit FONA Library (Adafruit 1.3.10), WiFi101 (Arduino 0.16.1), SimpleDHT (Winlin 1.0.15) e RTC (Manjunath CV 1.0.2).

Abaixo é apresentado um resumo na forma de tópicos das etapas de execução do algoritmo, e o código inteiro do projeto em C++ completamente comentado, pode ser consultado no Apêndice A.

Funções e variáveis globais:

- Inicialização das bibliotecas de Wi-Fi
- Inicialização das bibliotecas para programação via Wi-Fi (OTA – Over the Air)
- Inicialização das bibliotecas para execução do servidor http na porta 80
- Inicialização das bibliotecas para os informes ao site io.adafruit.com
- Entrada dos parâmetros por meio de variáveis para a conexão com o serviço Adafruit
- Definição das portas de saída do microcontrolador a serem utilizadas com os 4 atuadores, relés.
- Inicialização da biblioteca para leitura do sensor de temperatura e criação do objeto que receberá a leitura
- Definição da porta de leitura analógica em A0 para o sensor de luminosidade (LDR)
- Definição da porta digital de saída para leitura do sensor de gás MQ135
- Definição da porta digital de saída para o sinal de buzina (*buzzer*)
- Inicialização das bibliotecas para leitura do relógio de tempo real (RTC) e criação do objeto que receberá a leitura

- Inicialização da biblioteca que fará a leitura e gravação de dados na memória não volátil
- Entrada das variáveis globais de trabalho
- Criação de uma função para reiniciar a placa
- Criação de uma função para criar o arquivo dados.txt, que armazenará os dados lidos ao longo do tempo
- Criação de uma função que permite deletar o arquivo dados.txt
- Criação de uma função que anexa uma linha com os valores lidos nos sensores ao final do arquivo dados.txt
- Criação de uma função para alternar o estado das portas digitais dos atuadores, entre ligados e desligados, registrando os eventos
- Criação de uma função para limitar o acionamento de algum atuador para menos que 3 minutos
- Criação de uma função para apenas verificar o estado de uma porta do atuador e atualizar a variável global correspondente
- Criação de uma função para realizar a leitura da temperatura e umidade por meio do sensor DHT22
- Criação de uma função para realizar a leitura da luminosidade no sensor LDR
- Criação de uma função para realizar a calibragem da porta digital do sensor de gás, enviando dados da leitura analógica do sensor via porta serial
- Criação de uma função para acionar a buzina, levando em conta o número de apitos e o tempo de acionamento de cada apito em milissegundos
- Criação de uma função para executar o toque de alarme
- Criação de uma função para realizar a leitura do sensor de gás MQ135
- Criação de uma função para realizar a consulta do dia e horário atual do RTC
- Criação de uma função para atualizar a data e horário atual no RTC
- Criação de uma função para testar os diferentes dispositivos do equipamento, sensores, atuadores e informações a respeito de data e horário
- Criação de uma função para conexão ao serviço do site Adafruit
- Criação de uma função para controle do tempo
- Criação de uma função para transmitir os dados ao serviço do site Adafruit
- Criação de uma função para formatar a memória do dispositivo
- Criação de uma função para ler o conteúdo do arquivo de dados e armazenar em uma variável para uso em HTML
- Criação de uma função para iniciar o sistema de leitura e gravação na memória do microcontrolador
- Criação de uma função para fechar o sistema de leitura e gravação
- Criação de uma função para enviar via porta serial todo o conteúdo de dados armazenado no microcontrolador
- Criação de uma função para armazenar as leituras realizadas no arquivo dados.txt
- Criação de uma função para acionar o atuador correspondente para o caso de detecção de amônia
- Criação de uma função para acionar o atuador correspondente para o caso de mudanças na luminosidade, informando o limite em Lux, o intervalo de hora para a luz permanecer acesa e o intervalo em que a luz permanece apagada
- Criação de uma função para acionar o atuador correspondente para o caso da temperatura estar elevada
- Criação de uma função para acionar o atuador correspondente para o caso da temperatura estar baixa

- Criação de uma função para acionar o atuador correspondente para o caso da umidade estar elevada
- Criação de uma função para tratar solicitação na porta serial e executar tarefa correspondente
- Criação de variáveis globais para construção do conteúdo HTML
- Criação de uma função para transmitir página HTML de menu principal quando requisitado por um dispositivo externo
- Criação de uma função para transmitir página HTML para realizar o download dos dados armazenados no microcontrolador
- Criação de uma função para transmitir página HTML para possibilitar a realização da formatação
- Criação de uma função para transmitir página HTML para possibilitar a deleção do arquivo de dados
- Criação de uma função para transmitir página HTML para informar o término da transmissão de dados por meio da porta serial
- Criação de uma função para transmitir página HTML para informar a mudança no estado do Atuador 1
- Criação de uma função para transmitir página HTML para informar a mudança no estado do Atuador 2
- Criação de uma função para transmitir página HTML para informar a mudança no estado do Atuador 3
- Criação de uma função para transmitir página HTML para informar a mudança no estado do Atuador 4
- Criação de uma função para transmitir página HTML para possibilitar o reinício do microcontrolador
- Criação de uma função para alternar entre modo automático e manual para os atuadores
- Criação de uma função para possibilitar o desligamento da buzina
- Criação de uma função para possibilitar a programação do microcontrolador por meio de rede Wi-Fi (OTA – Over the Air)
- Criação de uma função para realizar a conexão com a rede Wi-Fi local

Função de configuração “setup” () {

- Configuração da porta serial
- Abre sistema de leitura e gravação e verifica se há o arquivo dados.txt
- Configuração dos pinos de entrada e saída do microcontrolador
- Configuração do RTC
- Inicialização do Wi-Fi e envio das informações de conexão pela porta serial
- Inicialização das configurações para ponto de acesso Wi-Fi
- Inicializando rotinas para programação via Wi-Fi (OTA – Over the Air)
- Inicialização do servidor web
- Desligamento dos atuadores
- Atualiza situação dos atuadores nas variáveis globais
- Realiza as primeiras leituras dos sensores

}

Função de repetição “loop” () {

- Inicializando escuta do servidor web

- Inicializando escuta de programação via Wi-Fi
- Inicializando função de controle de tempo
- Atualiza variáveis de estado dos atuadores
- Se o modo teste estiver habilitado, executa testes ininterruptamente
- Se estiver em modo calibração, envia os dados da porta analógica pela serial com os valores do sensor de gás
- Caso contrário:
  - Verifica e incrementa o tempo passado
  - Realiza a leitura de temperatura e umidade
  - Realiza a leitura da luminosidade
  - Realiza a leitura do sensor de amônia
  - Emite um pequeno sinal sonoro, indicando a realização da leitura naquele instante
  - Realiza conexão com o servidor do site io.adafruit.com
  - Faz a transmissão das leituras para o site io.adafruit.com
  - Grava as leituras na memória do microcontrolador
  - Modifica o estado do atuador para amônia caso necessário
  - Modifica o estado do atuador para luminosidade caso necessário
  - Modifica o estado do atuador para temperatura caso necessário
  - Modifica o estado do atuador para umidade caso necessário
- Realiza a escuta para a entrada de comandos via porta serial

}

5.6.5. Licença de uso de software

O código-fonte desenvolvido para esse projeto, utiliza como base de distribuição a licença de código livre GPL versão 3.0 (Licença Pública Geral).

Os arquivos referentes a esse projeto, incluindo o código-fonte, podem ser baixados no seguinte endereço eletrônico: <https://github.com/rafaelsantoro/tcc_controle_de_aviario/>, ou solicitados por e-mail ao autor, no endereço, rafaelsantoro@outlook.com, na possível indisponibilidade do primeiro.

5.7. Teste de funcionamento

Os testes do protótipo foram conduzidos em uma residência no município de Avaré, entre os dias 26/05/2022 a 30/05/2022, em duas etapas, na primeira, em que o dispositivo ficou localizado dentro de uma chocadeira comercial adaptada, dentro da residência, e outra em que o dispositivo ficou exposto a luminosidade natural no ambiente externo à residência.

No ambiente interno, considerou-se um sistema fechado, análogo aos galpões totalmente fechados para criação, quanto ao fator de luminosidade, assim como para a temperatura. Já no ambiente externo, considerou-se um sistema aberto, exposto as variações naturais de luminosidade promovidas pelo sol e suas variações em função do posicionamento e formação de nuvens.

O protótipo foi instalado dentro de uma chocadeira comercial, dispositivo com a finalidade de chocar ovos, que em seu interior possui resistências, lâmpadas, ventiladores e motores, além de um termostato eletrônico para o controle de temperatura e realizar periodicamente a movimentação dos ovos incubados. Todos os equipamentos da chocadeira foram desativados, a fim de que o controle da resistência e lâmpada ficassem a cargo do protótipo. A tampa da chocadeira foi substituída, por outra em que foi acoplado nela dois ventiladores, um como ventilador, com fluxo de ar para dentro do corpo da chocadeira e outro como exaustor, com fluxo de ar para fora do corpo da chocadeira. Um segundo equipamento, também equipado com o microcontrolador ESP-8266EX, diferente do protótipo foi instalado no lado de fora, utilizando o mesmo sensor de temperatura e luminosidade, a fim de captar a temperatura, umidade e luminosidade que variavam no ambiente em que a chocadeira estava posicionada, registrando todos os valores.

Para realizar os testes, alguns parâmetros precisaram ser definidos, o tempo de coleta de informações foi de 3 minutos, a idade das supostas aves, foi de 21 dias, sendo os valores de temperatura ideias na faixa de 24 a 26ºC, e Umidade Relativa entre 50 e 60%. Para o valor de luminosidade, foi definido como 25 Lux. O sensor de Amônia foi regulado para considerar a detecção acima de 20ppm.

Num primeiro momento, os testes foram realizados com a chocadeira do lado interno da residência, a fim de testar o controle de temperatura, umidade, luminosidade controlada por tempo e reação a presença de amônia, em um segundo momento o protótipo ficou posicionado no lado externo da residência, para que o acionamento da lâmpada pudesse ser testado em relação as condições de luminosidade.

A presença de amônia foi testada uma única vez, tendo em conta os riscos envolvidos na manipulação desse agente químico. A amônia é facilmente encontrada em casas que vendem produtos para limpeza, e está presente em produtos como o limpa-pedra. Pedaços de papel foram embebidos com o produto limpa-pedra e colocados dentro da chocadeira, próximo ao protótipo.

6. RESULTADOS

6.1. Custos

Os custos envolvendo o projeto se limitaram exclusivamente a aquisição de componentes para a construção do dispositivo. O serviço de MQTT escolhido (io.adafruit.com), na modalidade utilizada é gratuito, com algumas limitações. Outro custo foi o frete de aquisição dos componentes, mas como esse valor varia de acordo com a loja e o endereço de entrega, foi suprimido. Em grandes capitais como São Paulo, os componentes utilizados podem facilmente ser encontrados no centro da cidade, ficando este custo muito reduzido ou anulado. Para esse caso, em que a cidade onde foi realizado, Avaré-SP, os custos com frete se aproximaram de R$ 200,00. Componentes como os fios utilizados foram retirados de componentes sucateados, como fontes de alimentação de computadores, e uma tomada, aproveitada de um cabo de energia com defeito. Abaixo é possível ver na Tabela 5, a descrição dos itens utilizados e seus custos.

Tabela 5 - Custos por peça utilizada no projeto

Descrição	Quantidade	Valor Unitário (R$)	Total (R$)
Mini Fonte 3,3V / 5V	1	28,90	28,90
Módulo Buzzer Ativo	1	12,00	12,00
WeMos D1 Mini (ESP-8266EX)	1	30,00	30,00
Placa Fenolite perfurada 5x10 cm	2	11,45	22,90
Caixa Plástica Patola PB-112	1	32,90	32,90
Mini Fonte 12V	1	45,49	45,49
Sensor de Gás MQ-135	1	23,99	23,99
Sensor DHT-22	1	38,10	38,10
LED verde de alto brilho 5mm	4	0,60	2,40
LED vermelho de alto brilho 3mm	1	0,40	0,40
Suporte para LED cromado 3mm	1	0,55	0,55
LDR 5528	1	0,90	0,90
RTC DS1307	1	15,90	15,90

Relé Sun Hold RAS-1210	4	10,80	43,20
Transistor BC-548	4	0,32	1,28
Resistor 1KOhm 1/4W	4	0,24	0,96
Resistor 10KOhm 1/4W	1	0.10	0.10
Resistor 470Ohm 1/4W	5	0,16	0,80
Diodo 1N4001	4	0,15	0,60
Barramento 12 vias 60A	1	22,50	22,50
Barramento 12 vias 20A	1	13,50	13,50
Porta fusível	1	2,90	2,90
Fios coloridos	-	0,00	0,00
Cabo para tomada	1	0,00	0,00
Total:			340,27

Fonte: Elaboração própria.

As ferramentas utilizadas para construção do dispositivo, foram ferramentas simples e muito comuns, como ferro de solda, sugador de solda, furadeira, estilete, chave Philips, algumas brocas e pistola de cola quente para prender alguns sensores. Os insumos foram: tubos de cola quente, solda de estanho, fluxo de solda e abraçadeiras de nylon.

6.2. Montagem

Após a soldagem de todos os componentes e cabos necessários para a comunicação entre os diferentes dispositivos, pode-se ver na Figura 23, a interligação entre todos, sendo todos esses componentes e fios acondicionados posteriormente dentro da caixa.

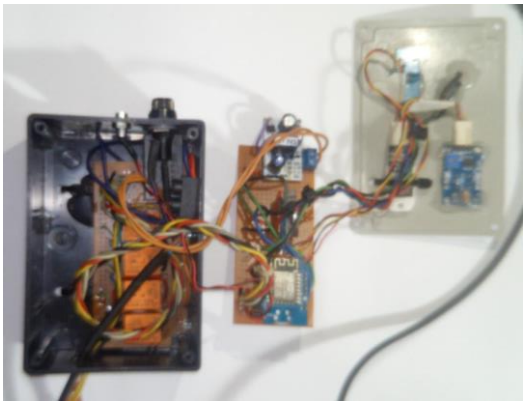


Figura 23 - Detalhe das ligações entre os componentes
Fonte: Elaboração própria.

Depois de acondicionados todos os componentes e fechado o dispositivo, é apresentado na Figura 24, o resultado da montagem do hardware, à esquerda o dispositivo devidamente fechado e à direita o barramento de conexão dos dispositivos a serem controlados pelos atuadores.

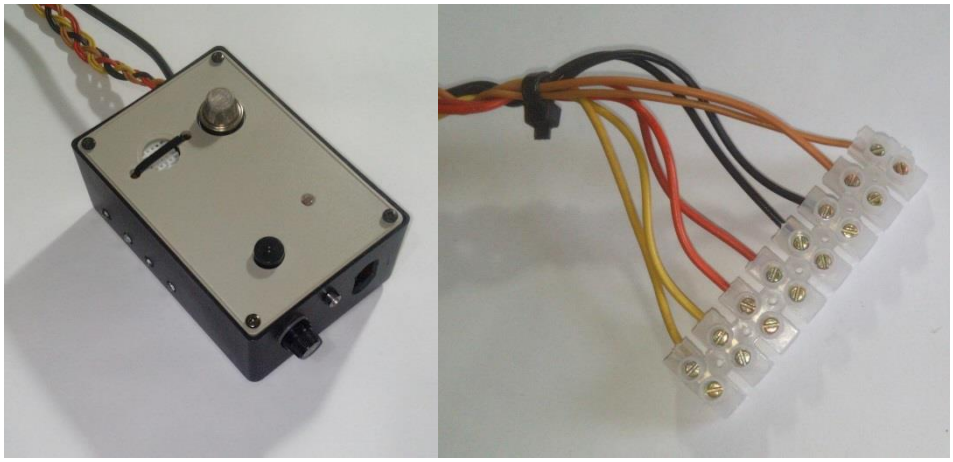


Figura 24 - Resultado final da montagem do hardware
Fonte: Elaboração própria.

6.3. Funcionamento básico

Antes da realização dos testes, o protótipo foi ligado e suas medidas de temperatura foram comparadas com uma placa externa, equipada com um sensor igual, o DHT22, porém as medidas entre as duas eram muito discrepantes, em torno de 6°C a mais, medidas no protótipo. Ao avaliar a caixa que acondicionava os sensores, foi percebido que as duas fontes de energia no interior do protótipo e também o sensor de gás MQ135 produziam calor, que nas cercanias do sensor DHT22 produziam aquecimento e adulteravam a medida do sensor

do protótipo. A primeira opção a ser tentada, foi a realização de diversos furos na caixa do protótipo, para que o ar quente, por meio de convecção deixasse o interior da caixa sem que o calor pudesse afetar o sensor, porém essa medida ainda não foi suficiente, sendo a diferença entre os sensores reduzida para 2°C, o que ainda poderia prejudicar os resultados comparativos entre o ambiente e os valores medidos dentro do ambiente de teste. Então o sensor DHT22 foi removido da caixa plástica que acondicionava o projeto e foi afixado ao lado de fora, e nos novos testes a temperatura se igualou ao sensor da placa de medição de ambiente. Na Figura 25, é possível ver como ficou o protótipo após os furos e a exposição do sensor DHT22.

Figura 25 - Imagem do protótipo após realização de furos e exposição do sensor DHT22
Fonte: Elaboração própria.

6.4. Amônia

O teste de funcionamento do sensor de amônia foi realizado próximo às 19:00 do dia 27/05. Após a inserção do papel embebido em amoníaco e passados alguns minutos, o sensor detectou a presença de amônia e ligou o exaustor. Após três minutos o exaustor desligou. Abaixo pode ser verificado na Tabela 6, a sequência de eventos e as leituras antes e logo após o teste com amônia.

Tabela 6 - Dados na proximidade temporal no momento da realização do teste de presença de Amônia

Data	Hora	Amônia	Exaustor Ligado	Exaustor Desligado
27/05/2022	18:58:10	Ausente		
27/05/2022	19:01:10	Ausente		
27/05/2022	19:04:09	Ausente		
27/05/2022	19:07:08	Presente	X	
27/05/2022	19:10:10	Ausente		X
27/05/2022	19:13:09	Ausente		
27/05/2022	19:16:08	Ausente		
27/05/2022	19:19:07	Ausente		

Fonte: Elaboração própria.

6.5. Temperatura e Umidade

Entre os dias 26 e 27/05, foram realizados os testes com a temperatura e umidade. Nesses dias a temperatura ambiente estava abaixo da meta estipulada, que era de 24 a 26ºC, sendo assim o protótipo deveria elevar a temperatura dentro da chocadeira. Como pode ser visto na Figura 26, em um primeiro momento, entre as 22:00 e 06:00 houve uma queda na temperatura ambiente, fora da chocadeira, levando a resistência a atuar no ambiente interno por muitas vezes e predominantemente se manteve ligada, mantendo a temperatura dentro da chocadeira em torno de 24ºC, dentro da meta. A partir das 06:00, iniciou-se o aquecimento do ambiente interno devido ao nascer do sol, nesse momento a resistência atuou por menos vezes, mas ainda mantendo a temperatura relativamente estável dentro da chocadeira até as 13:52, quando a temperatura do ambiente se elevou de tal modo que a própria chocadeira com o isolamento térmico de seus materiais, manteve-se estável chegando a quase 25ºC.

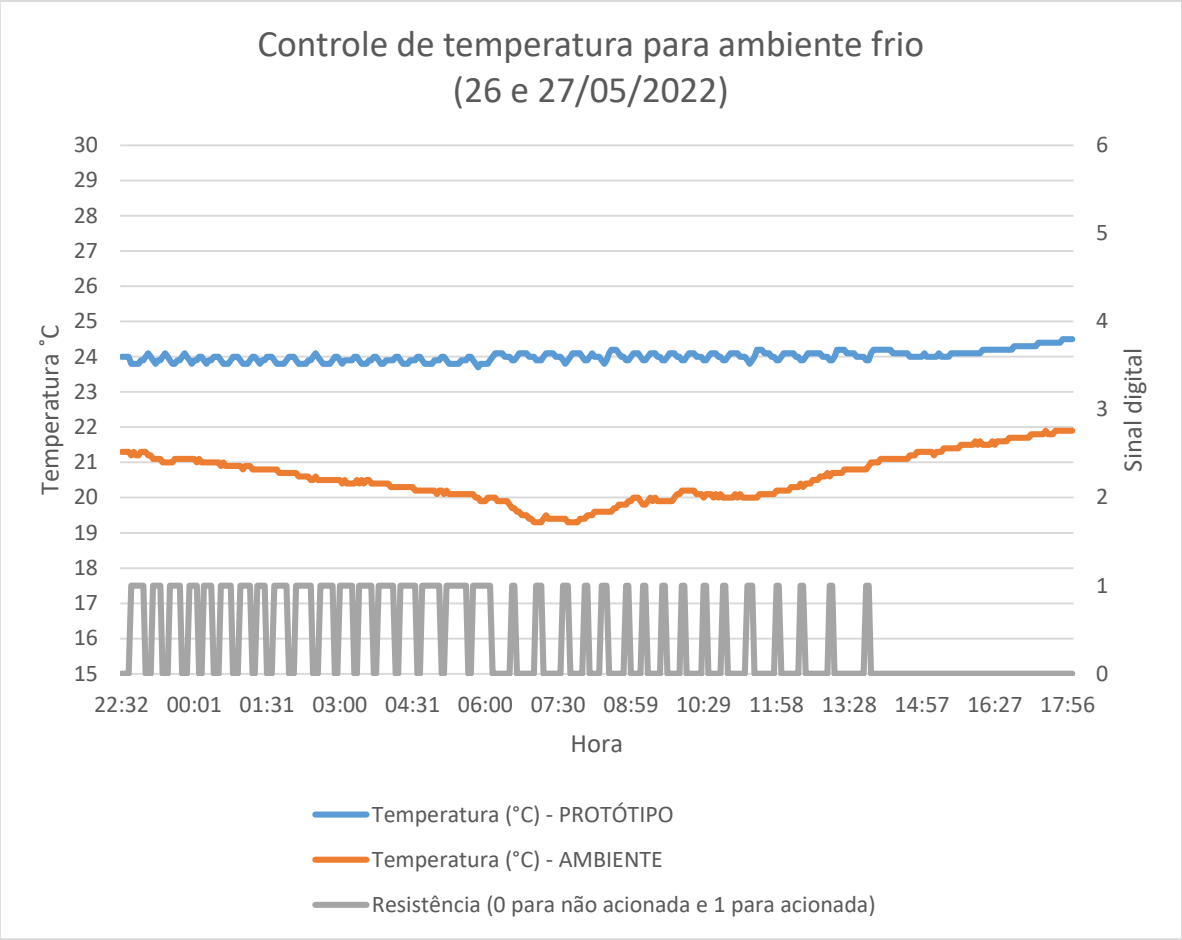


Figura 26 - Atuação da resistência em função da temperatura medido pelo protótipo
Fonte: Elaboração própria.

No mesmo intervalo de tempo em que foi testada a temperatura, foram também registradas as umidades relativas dentro e fora da chocadeira, observando na Figura 27, é possível notar que devido ao controle de temperatura realizado, a umidade relativa dentro da chocadeira se manteve dentro dos parâmetros pré-estabelecidos, entre 50 e 60%, já a umidade relativa do ambiente teve picos de 62% e suas variações foram mais abruptas.

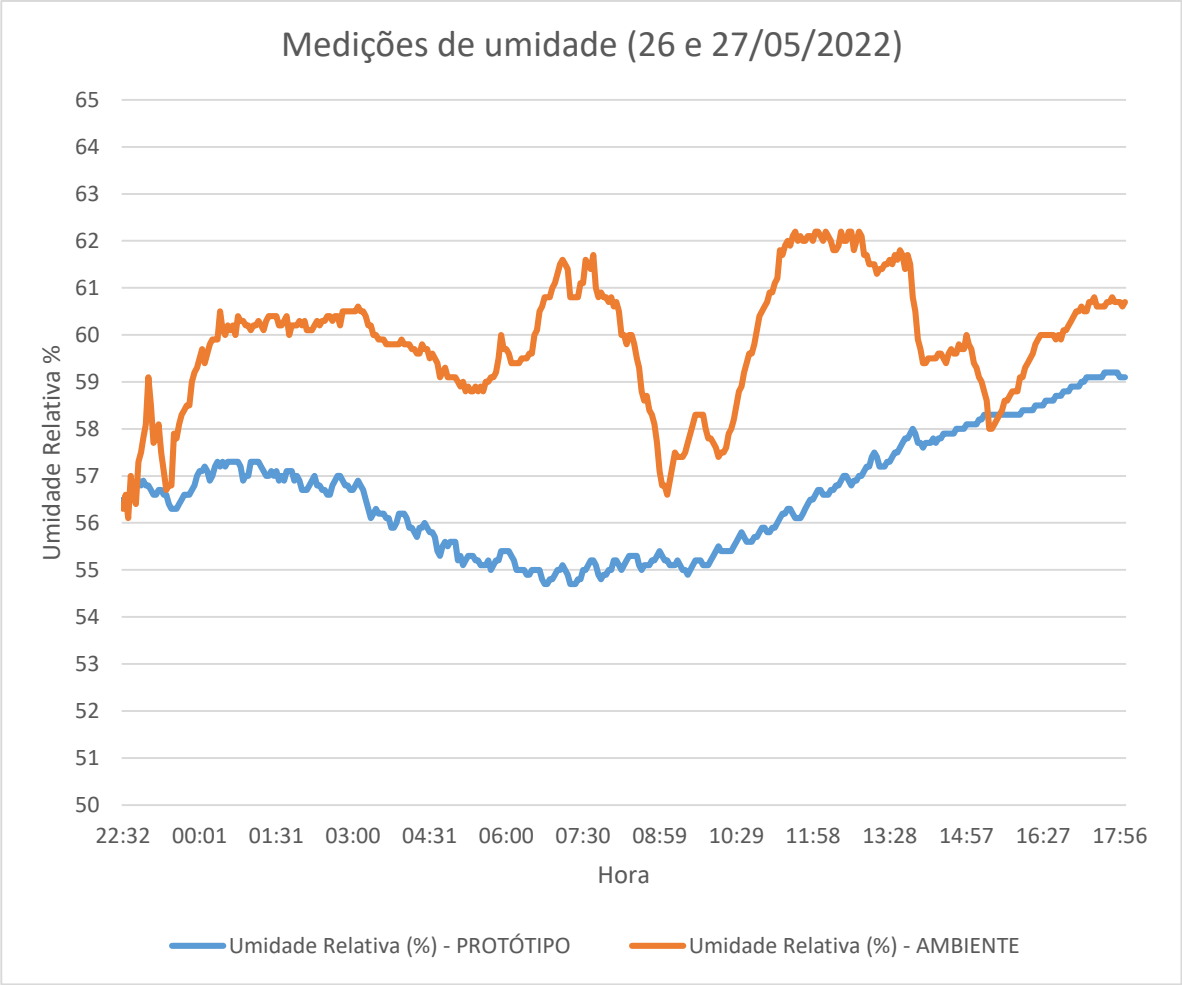


Figura 27 - Registros de umidade relativa dentro da chocadeira e de ambiente
Fonte: Elaboração própria.

Durante os primeiros testes realizados, não houve aumento de temperatura, e o ventilador não foi acionado nenhuma vez devido à baixa temperatura ambiental predominante nessa época. Para realizar o teste de atuação do ventilador, o protótipo foi aproximado de uma lâmpada de LED acesa, o que elevou a temperatura em suas cercanias e provocou a atuação do ventilador. Na Figura 28, é possível verificar que, de acordo com a elevação de temperatura acima de 26ºC, o ventilador era então acionado, decaindo a temperatura dentro da chocadeira em pouco mais de 1ºC, desligando em seguida, mantendo a temperatura nas proximidades do objetivo, de 24 a 26ºC.

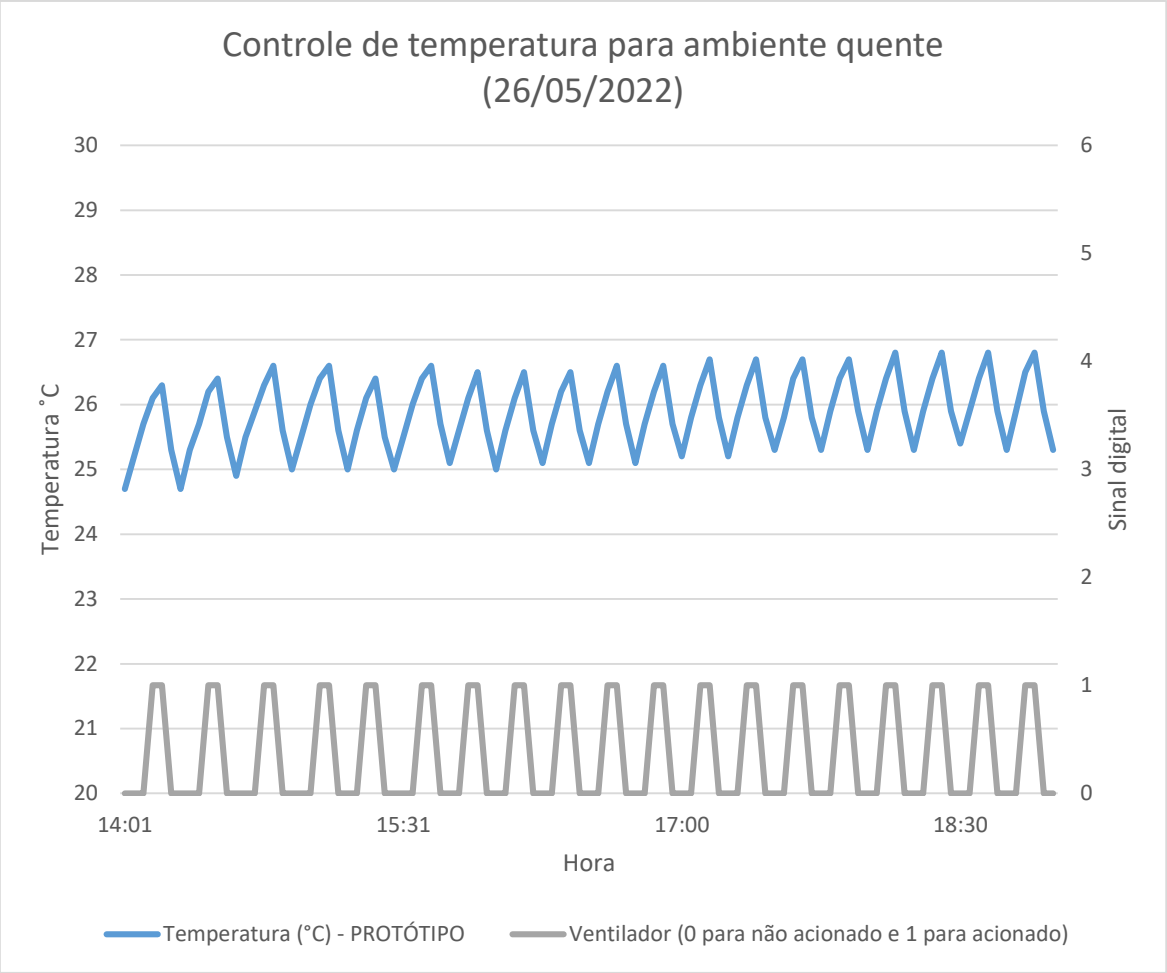


Figura 28 - Atuação do ventilador em função da temperatura medido pelo protótipo
Fonte: Elaboração própria.

6.6. Luminosidade

Duas modalidades de utilização foram preparadas nesse protótipo, uma levando em conta apenas o horário de ativação de lâmpadas, para ser utilizadas em galpões completamente fechados e com iluminação completamente artificial e a outra modalidade, em que o sensor de luz captando a luminosidade natural, pode acionar as lâmpadas com base no valor de Lux transmitido pelo sensor. O teste da primeira modalidade foi feito entre o dia 26 e 27/05, a programação foi feita de modo a manter a lâmpada acesa entre as 6:00 e as 20:00. Como visto na Figura 29, a lâmpada foi acionada as 6:00 e desligada as 20:00 como previsto.

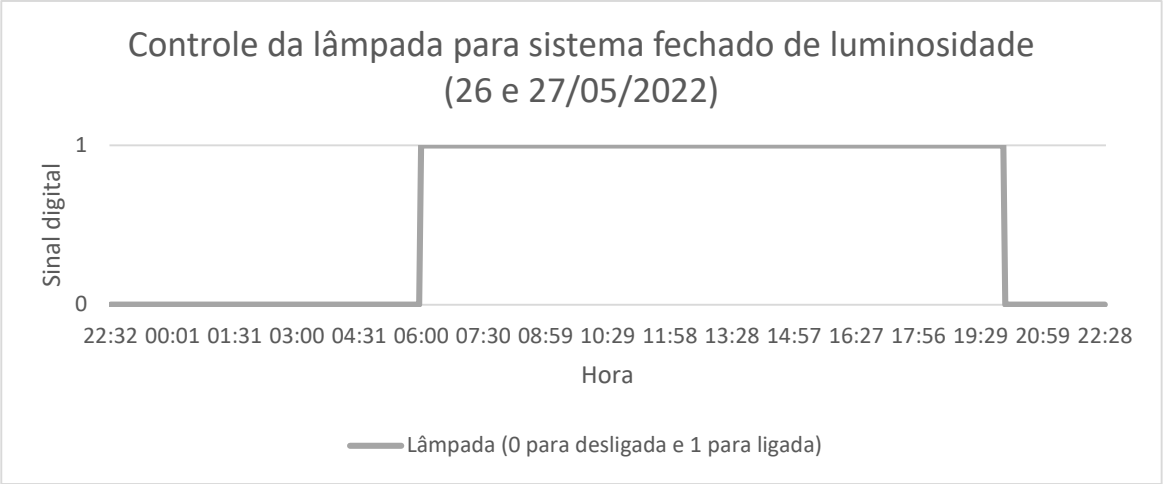


Figura 29 - Atuação da lâmpada em função do horário estabelecido no protótipo
Fonte: Elaboração própria.

Na segunda modalidade, o limite de Lux utilizado foi de 25, o horário de desligamento da luz definido em 20:00 e foi realizado no lado externo da residência, em ambiente aberto com cobertura apenas por telhado. No início da avaliação o sensor ambiente e o sensor do protótipo marcavam 100 Lux, pois a luz solar ainda era abundante. Por volta das 17:00 como é esperado, a luminosidade solar decaiu, nesse ponto o protótipo lia menos que 25 Lux no LDR, então a lâmpada foi acionada. Devido ao posicionamento ligeiramente diferente entre o protótipo e a placa de medidas de ambiente, a leitura de luminosidade apresentou uma ligeira diferença, pois sobre o protótipo incidia um pouco de luminosidade da lâmpada e sobre a placa de medidas não. Quando o horário atingiu 20:00, limite do monitoramento de luz estabelecido no protótipo, a lâmpada foi desligada e as leituras do protótipo e da placa de medidas de ambiente ficaram semelhantes, na escuridão, como já era esperado e o protótipo a manteve desligada até as 6:00, quando a religou automaticamente, pois a luminosidade solar incidente ainda não tinha atingido o limiar de 25 Lux. Por volta de 7:40, a luminosidade solar atingiu 25 Lux, desligando a lâmpada, porém, o dia estava nublado, com muitas nuvens pairando no céu, o que pode ter provocado as diversas oscilações de luminosidade, ocasionando vários acionamentos da lâmpada até as 8:30, quando as medições do protótipo atingiram níveis acima de 25 Lux, não acionando mais a lâmpada até o final da avaliação. Todas essas variações podem ser vistas na Figura 30.

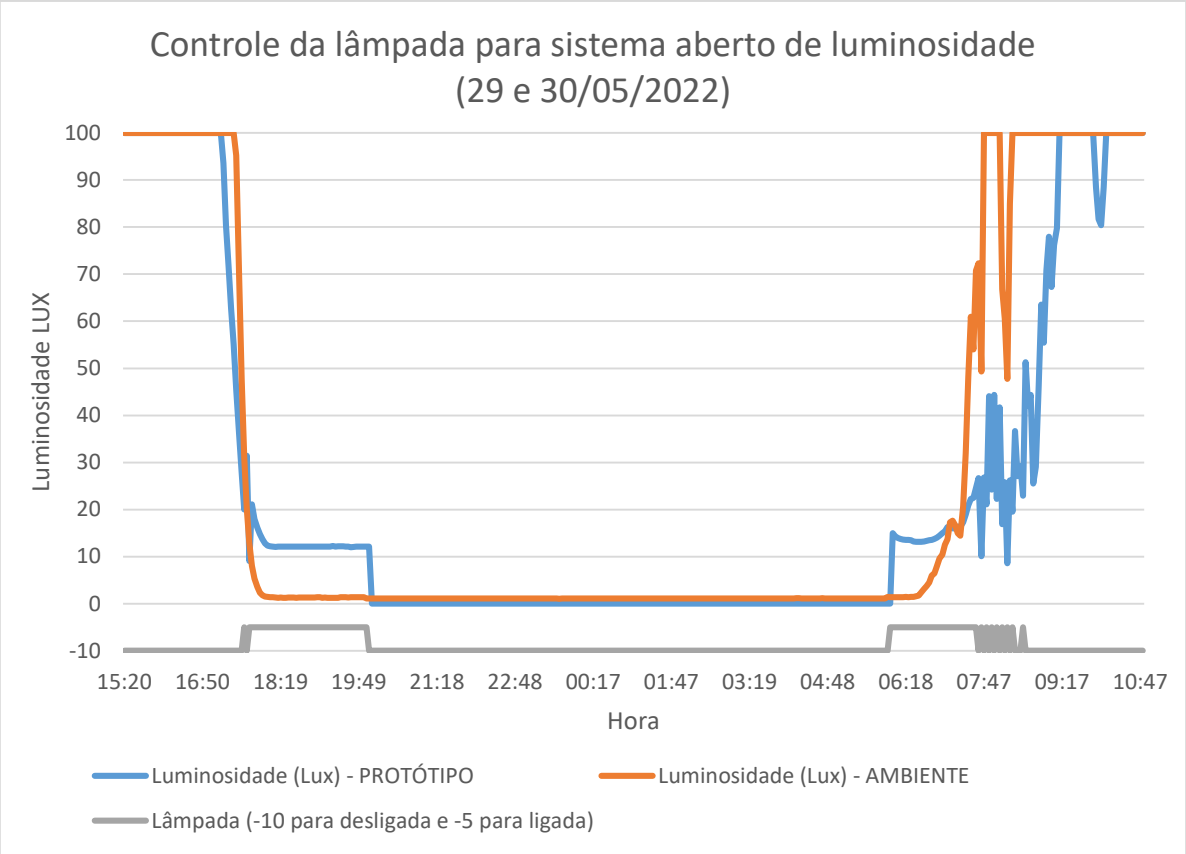


Figura 30 - Atuação da lâmpada em função da luminosidade e horário estabelecido no protótipo
Fonte: Elaboração própria.

6.7. Protocolo HTTP

As configurações, opções e estado atual dos sensores puderam ser verificados por uma página web pelo protocolo HTTP. Uma captura de tela com as informações e botões disponíveis pode ser vista na Figura 31.

Figura 31 - Captura de tela do acesso HTTP
Fonte: Elaboração própria.

Os dados registrados pelo microcontrolador em sua memória, puderam facilmente ser baixados diretamente na página HTTP, o arquivo pode ser aberto diretamente no navegador e facilmente convertido para o formato CSV, podendo ser utilizado em programas editores de planilhas. Uma captura de tela com uma amostra dos dados coletados pode ser vista na Figura 32.

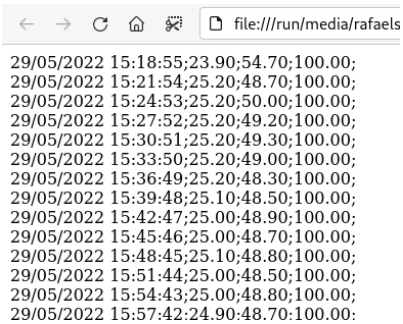


Figura 32 - Captura de tela com os dados baixados diretamente do microcontrolador
Fonte: Elaboração própria.

6.8. Protocolo MQTT

Durante a execução dos testes, os dados puderam ser acompanhados praticamente em tempo real, por meio do protocolo MQTT, com diferença de apenas alguns segundos entre o envio da informação pela internet e a atualização no serviço de MQTT da Adafruit. Os dados puderam ser acompanhados pelo proprietário da conta, no caso o autor e por qualquer pessoa que tivesse o link de compartilhamento previamente autorizado pelo autor no site do serviço, por meio do link: <https://io.adafruit.com/rafaelsantoro/dashboards/tcc-controlador-de-aviario?kiosk=true>. O acompanhamento dos dados pode ser visto em uma captura de tela, na Figura 33.

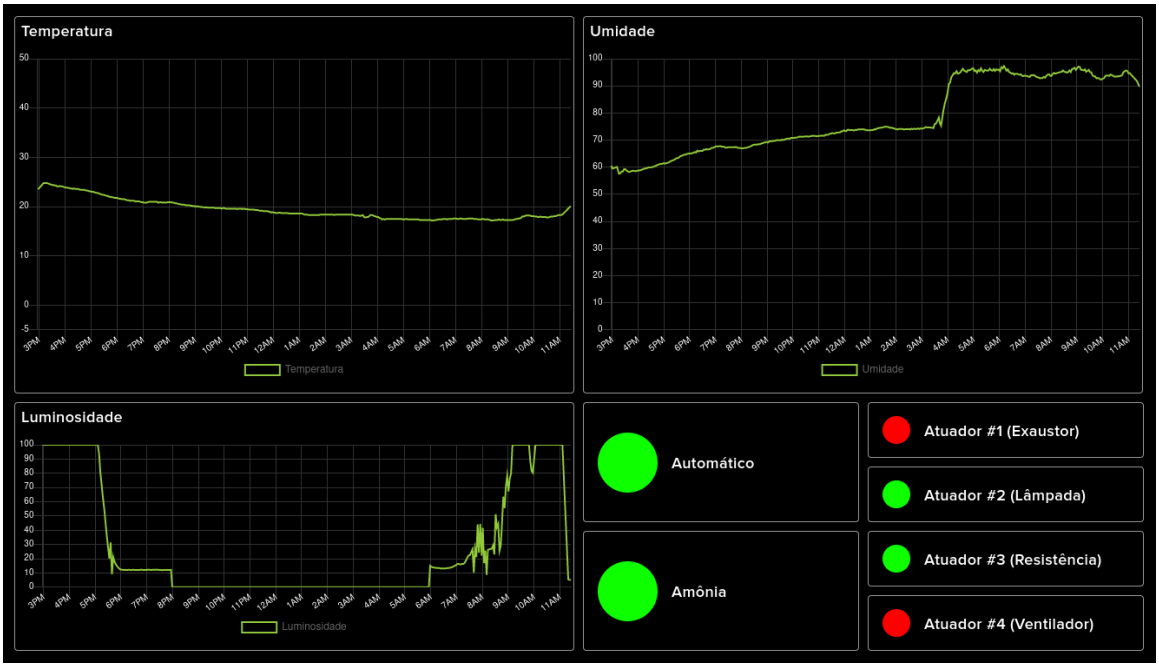


Figura 33 - Captura de tela do site io.adafruit.com
Fonte: Elaboração própria.

Ações também podem ser configuradas no serviço da Adafruit, uma ação de notificação por e-mail no caso de detecção do nível de amônia foi configurada e o e-mail chegou apenas alguns minutos após a realização do teste. Nota-se que o horário no e-mail diverge do momento de realização do teste em alguns minutos, isso se deve a disparidade entre o horário do RTC do protótipo e do relógio do computador que recebeu o e-mail. Uma captura de tela do e-mail recebido pode ser vista na Figura 34. O serviço da Adafruit permite além de envio de e-mails com critérios configuráveis, a integração com outros serviços que podem ser utilizados para envio para as redes sociais ou mensagens telefônicas.

Figura 34 - Captura de tela do e-mail de aviso recebido do site io.adafruit.com
Fonte: Elaboração própria.

Para além do computador, foi configurada a conexão ao serviço MQTT da Adafruit, por um aplicativo de celular, o IoTMQTTPanel, disponível gratuitamente na Play Store, loja de aplicativos do sistema Android. Como pode ser visto na Figura 35, foi possível receber dados por meio da internet em um aplicativo em um dispositivo móvel, que pode ser verificado em qualquer local do globo com cobertura de internet disponível.

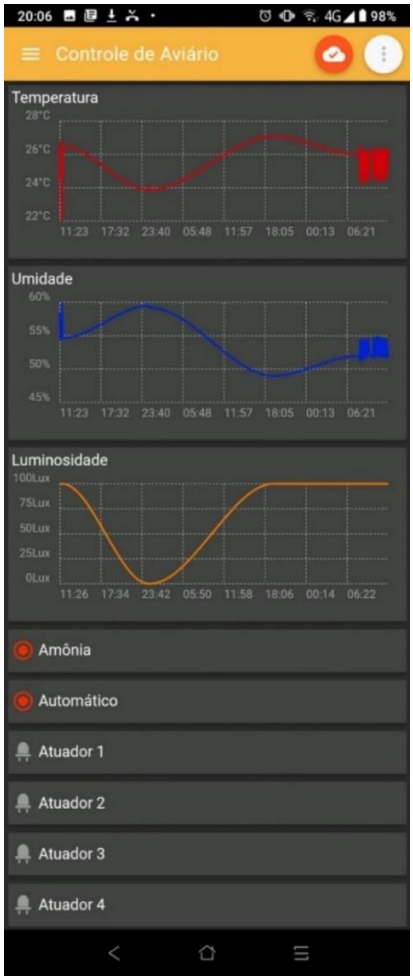


Figura 35 - Captura de tela do software IoTMQTTPanel, em sistema Android
Fonte: Elaboração própria.

7. CONSIDERAÇÕES FINAIS

Foi possível o desenvolvimento de um equipamento funcional, que monitora, registra e controla as variáveis de temperatura, umidade, luminosidade e presença de gás amônia, que funcionou de forma contínua.

O nível de controle alcançado por meio dos atuadores e medidas dos sensores foi satisfatório, mantendo as variáveis dentro dos limites estabelecidos.

Os dados produzidos pelo protótipo puderam ser transmitidos pela internet de modo satisfatório e foram acessados por diferentes dispositivos de distintas maneiras. Os dados também ficaram armazenados e puderam ser baixados diretamente da placa, o que proporcionou o embasamento para a análise dos resultados deste trabalho.

O protótipo produzido possui custo relativamente baixo, o que permite que pequenos produtores façam uso, com a utilização de linguagem amplamente difundida, ou para que profissionais qualificados possam oferecer serviços por meio da utilização do equipamento a custos acessíveis.

O documento produzido, possui linguagem de fácil entendimento, acompanhada da maneira mais próxima possível dos métodos de construção, procedimentos de programação e análise, além de muitas figuras que facilitam futuras reproduções desse projeto por outras pessoas interessadas, sejam curiosos, estudantes ou pesquisadores.

8. IMPLICAÇÕES

Sugere-se para trabalhos futuros:

A avaliação da área coberta pelos sensores do equipamento.

Verificar se é possível a substituição das fontes de 3,3V e 5V por conversores DC/DC.

Avaliar a utilização de multiplexadores, a fim de possibilitar o incremento no número de sensores, ou dispositivos que possam ser controlados pelo equipamento.

Desenvolver dentro do próprio protocolo HTTP, meios de alterar os parâmetros por meio de um formulário em um navegador, como o nível de amônia, luminosidade e as temperaturas críticas.

Avaliar a possibilidade de utilização de mais de um equipamento no mesmo ambiente, possibilitando a avaliação de diferentes locais no mesmo ambiente de produção, podendo até mesmo ser realizado o controle por meio de inteligência artificial.

Acondicionar o equipamento em uma caixa plástica maior, para contornar o problema do aquecimento.

Estudar ou testar a aplicação deste mesmo equipamento para outros nichos comerciais, como na cunicultura.

REFERÊNCIAS

ABPA – Associação Brasileira de Proteína Animal. **Relatório Anual 2020**. São Paulo. 2021.

ABREU, P. G.; ABREU, V. M. N. **Manejo no Calor.** Ageitec - Agência Embrapa de Informação Tecnológica. Disponível em: < https://www.agencia.cnptia.embrapa.br/gestor/frango_de_corte/arvore/CONT000fc6ggago02wx5eo0a2ndxyernkww7.html> Acesso em: 17/01/2022.

ABREU, P. G.; ABREU, V. M. N. **Conforto Térmico para Aves.** Concórdia. Embrapa Suínos e Aves, 2004. (Embrapa Suínos e Aves. Comunicado Técnico, 365).

AGEVOLUTION, **Aviário inteligente de baixo custo controla ambiência de aves,** 2020**.** Disponível em: < https://agevolution.canalrural.com.br/aviario-inteligente-de-baixo-custo-controla-ambiencia-de-aves/> Acesso em: 17/03/2022.

AMARIEI C., **Arduino Development Cookbook**, Ed. Packt Publishing, Birmingham – Reino Unido, abril de 2015.

AOSONG. **Temperature and humidity module AM2302 Product Manual.** 2019. Disponível em: <http://akizukidenshi.com/download/ds/aosong/AM2302.pdf> Acesso em: 02/02/2022.

ASCOM, **A Internet das Coisas chega à avicultura.** 2018**.** Disponível em: <https://www.aviculturaindustrial.com.br/imprensa/a-internet-das-coisas-chega-a-avicultura/20180416-145048-w683>. Acesso em: 24/01/2022.

ATMEL. **8-bit AVR Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller**. Disponível em: < https://www.microchip.com/wwwproducts/productds/ATmega8U2> Acesso em: 02/11/2021.

BAÊTA F. C., **Sistemas de ventilação natural e artificial na criação de aves.** Departamento de Engenharia Agrícola da Universidade Federal de Viçosa-MG. Simpósio Internacional sobre Ambiência e Sistemas de Produção Avícola, Concórdia-SC, 1998.

BRAGA N. C., **Tudo Sobre Relés**, Instituto NCB Newton C. Braga, 2009. Disponível em: <https://www.newtoncbraga.com.br/index.php/como-funciona/597-como-funcionam-os-reles.html?start=4> Acesso em: 31/03/2022.

BRASIL. Ministério do Trabalho e Previdência. **Norma Regulatória N. 15 (NR-15) Atividades e Operações Insalubres**. Brasília. DF. 2022.

CFMV – Conselho Federal de Medicina Veterinária. **Programa de Luz na Avicultura de Postura.** Revista CFMV. N. 52, Brasília. DF. 2011. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/42481/1/Paginas-de-CFMV-52.pdf> Acesso em: 15/03/2022.

COBB-VANTRESS. **Manual de manejo de frangos de corte**. Guapiaçu-SP, Cobb Vantress, p. 20, 2009.

EMBRAPA. **Manejo Ambiental na Avicultura**. Embrapa Suínos e Aves. Documentos. N. 149, Concórdia-SC, 2011.

EXPRESSIF SYSTEMS. **ESP-8266EX *Datasheet***. 2020. Disponível em: < https://espressif .com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf> Acesso em: 02/11/2021.

FAIRCHILD, **BC546/BC547/BC548/BC549/BC550 NPN Epitaxial Silicon Transistor**, *Datasheet*, 2014. Disponível em: < https://www.mouser.com/datasheet/2/149/BC547-190204.pdf> Acesso em: 31/03/2022.

GNU. **Um guia rápido para a GPLv3**, 2022. Disponível em: < https://www.gnu.org/licenses/quick-guide-gplv3.pt-br.html> Acesso em: 01/06/2022.

HAN H., ZHOU Y, LIU Q., WANG G., FENG J., ZHANG M.; **Effects of Ammonia on Gut Microbiota and Growth Performance of Broiler Chickens.** Animals, v. 11, n. 6, p. 1716 2021.

LIBERA G. P. D., OLIVEIRA M. E., TECH R. B., **DESENVOLVIMENTO DE UM SISTEMA PARA CONTROLE DE TEMPERATURA, UMIDADE E ILUMINAÇÃO EM AVIÁRIOS**, 26° SIICUSP - Simpósio Internacional de Iniciação Científica e Tecnológica da USP, São Paulo-SP, 2018.

LIDA OPTICAL & ELETRONIC CO. **CdS Photoconductive Cells GL5528.** *Datasheet*. China. Disponível em: < https://datasheetspdf.com/pdf-file/1346414/ETC/GL5528/1> Acesso em: 02/11/2021.

LOTT B.; DONALD J., **Amônia**. Avicultura Industrial. 2003. Disponível em: <https://www. aviculturaindustrial.com.br/imprensa/amonia/20030711-113203-0098 > Acesso em: 15/03/2022.

MAXIM INTEGRATED, **DS1307 64 x 8, Serial, I2C Real-Time Clock**, *Datasheet*. Sunnyvale, Califórnia, USA, 2015.

MONITOR – Repórter Brasil. **A Indústria do Frango no Brasil**. Número 2. Junho/2016.

MONK S., **Internet das Coisas – Uma Introdução com o Photon**. Bookman, Porto Alegre, 2018.

PRO-SIGNAL. **Buzzer.** *Datasheet*. 2016. Disponível em: < https://components101.com/sites/ default/files/component_datasheet/Buzzer%20Datasheet.pdf> Acesso em: 02/02/2022.

PROTTO. **Estrutura e lógica de programação Arduino**, 2020. Disponível em: <https://protto.com.br/2020/10/26/estrutura-e-logica-de-programacao-arduino/> Acesso em: 29/04/2022.

RED HAT. **IDE - Ambiente de Desenvolvimento Integrado.** 2019. Disponível em: <https://www.redhat.com/pt-br/topics/middleware/what-is-ide> Acesso em: 20/01/2022.

RORIZ B. C., **LESÕES NO COXIM PLANTAR DE FRANGOS DE CORTE EM DIFERENTES SISTEMAS DE PRODUÇÃO**. UFGD. Dourados-MS. 2016.

ROSS, **Manual de Manejo – Frango de Corte**, Aviagem, 2018. Disponível em: < http://pt.aviagen.com/assets/Tech_Center/BB_Foreign_Language_Docs/Portuguese/Ross-BroilerHandbook2018-PT.pdf> Acesso em: 29/03/2022.

SAINT-EXUPÉRY, A**. O pequeno príncipe**. 31.ed., Ed. Agir, Rio de Janeiro, 1987.

SANTOS R. J.; JÚNIOR L. R., **PROJETO, CONSTRUÇÃO E INSTALAÇÃO DE UM DISPOSITIVO ELETRÔNICO PARA REGISTRO E MONITORAMENTO DA TEMPERATURA E UMIDADE EM UMA GRANJA DE COELHOS**. Encontro de Desenvolvimento de Processos Agroindustriais. Universidade de Uberaba, Departamento de Engenharia Elétrica, Uberaba-MG, 2020.

SCOLARI T. M. G., **Cuidados com o manejo das aves no verão.** EMBRAPA. 2008. Disponível em: < https://www.embrapa.br/busca-de-noticias/-/noticia/18023076/cuidados-com-o-manejo-das-aves-no-verao>. Acesso em: 17/01/2022.

SHAMIEH C.; McCOMB G., **Eletrônica para LEIGOS**, Ed. Alta Books, Rio de Janeiro, 2012.

SNS. **TECHNICAL DATA MQ-135 GAS SENSOR**. *Datasheet*. Disponível em: < https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf > Acesso em: 16/11/2021.

SOARES R.; FERREIRA I., **Cuidados essenciais no controle de temperatura dos aviários em dias de calor**. Avicultura Industrial. N. 9, 2020.

SOARES R. F. R. D., **Comparação entre protocolos de camada de aplicação para IoT**. UFRPE – Universidade Federal Rural de Pernambuco, Recife, 2019.

STEVAN, S. L. Jr.; FARINELLI, F. A. **DOMÓTICA – Automação residencial e casas inteligentes com Arduino e ESP8266.** São Paulo: Érica, 2019.

SUHANKU D., **A "linguagem do Arduino" é C ou C++?** 2019. Disponível em: < https://www.dobitaobyte.com.br/a-linguagem-do-arduino-e-c-ou-c/> Acesso em: 27/04/2022.

SUN HOLD, **RAS Series**, *Datasheet*. Disponível em: < https://www.sunhold.com/upload/prd1/118-3.pdf> Acesso em 31/03/2022.

WEAVER W. D. Jr.; MEIJERHOF R., **The effect of different levels of relative humidity and air movement on litter conditions, ammonia levels, growth and carcass quality for broiler chickens.** Poultry Science, v.70, n.4, p.746-755, 1991.

WEMOS. **Documentation LOLIN D1 mini v3.1.0.** 2021. Disponível em: <https://www.wemos.cc/en/latest/d1/d1_mini_3.1.0.html> Acesso em: 02/01/2021.

WINSEN. **MQ135 Semiconductor Sensor for Air Quality.** *Datasheet*. China. 2015. Disponível em: < https://www.winsen-sensor.com/d/files/PDF/Semiconductor%20Gas%20Sensor/MQ135%20(Ver1.4)%20-%20Manual.pdf> Acesso em: 16/11/1984.

APÊNDICES

Apêndice A – Algoritmo utilizado no microcontrolador em linguagem C++

```
//Programa para utilização em ESP8266 para controle de variáveis em pequenos aviários//
//Elaborado sob licença GPL v3.0//
//Os interessados podem utilizar o software para qualquer finalidade,//
//mudar o software de acordo com suas necessidades, compartilhar o software//
//com seus amigos e vizinhos, entre outros e compartilhar as mudanças que você fez.//
//Peço a gentileza, que cite o criador desse código quando fizer uso deste e mantenha o link//
//do projeto original no GitHub: https://github.com/rafaelsantoro/tcc_controle_de_aviario nos
comentários//
//do seu projeto. No mais faça bom uso e tenha um ótimo dia.//
//Este código teve por objetivo permitir o funcionamento de um protótipo elaborado como traba-
lho de//
//conclusão de curso para o curso de Engenharia de Biossistemas pelo acadêmico Rafael R. San-
toro//
//no campus do IFSP-Avaré, Brasil em 2022.//

//Inicializando Wi-Fi
#include <ESP8266WiFi.h>               //Biblioteca para controle do WI-FI
#define WLAN_SSID  "wwwwwwww"          //Variável com o SSID da rede
#define WLAN_PASS  "xxxxxxxx"          //Variável com a senha de rede
#define AP_SSID    "yyyyyyyy"          //Constante com o nome de SSID para acesso via Access
Point (direto no aparelho)
#define AP_PASS    "zzzzzzzz"          //Constante com a senha para acesso via Access Point
(direto no aparelho)

//Inicializando OTA
#include <ESP8266mDNS.h>              //Biblioteca para publicação de IP via DNS
#include <WiFiUdp.h>                  //Biblioteca para protocolo UDP, auxilia no DNS
#include <ArduinoOTA.h>               //Biblioteca para programação do microcontrolador via
WI-FI (Over the air)

//Inicialização para formulário e WEB Server
#include <WiFiClient.h>               //Biblioteca auxiliar de conexão WI-FI para o serviço
Adafruit
#include <ESP8266WebServer.h>         //Biblioteca com o servidor WEB para o microcontrolador
ESP8266WebServer server(80);          //Cria o objeto server do servidor web na porta 80

//Inicializando io.adafruit.com (Protocolo MQTT)
#include "Adafruit_MQTT.h"                                    //Biblioteca necessária para comu-
nicação com o serviço Adafruit
#include "Adafruit_MQTT_Client.h"                             //Biblioteca necessária para comu-
nicação com o serviço Adafruit
#define AIO_SERVER      "io.adafruit.com"                     //Constante com o endereço DNS do
servidor da Adafruit
#define AIO_SERVERPORT  1883                                  //Porta de comunicação com a Ada-
fruit
#define AIO_USERNAME  "xxxxxxxxxxxxx"                         //Usuário do serviço Adafruit
#define AIO_KEY  "aio yyyyyyyyyyyyyyyyyyyyyyyyyyyy"           //Chave para publicação do serviço
Adafruit
WiFiClient client;                                            //Cria objeto client do tipo Wifi-
Client para uso com Adafruit
//Parâmetros a serem utilizados na conexão com a Adafruit
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

  //Configurando os feeds do MQTT
  Adafruit_MQTT_Publish modo = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/modo");
//Publicação para Modo Automático
  Adafruit_MQTT_Publish atuador1 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/atua-
dor1");            //Publicação para atuador1
  Adafruit_MQTT_Publish atuador2 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/atua-
dor2");            //Publicação para atuador2
  Adafruit_MQTT_Publish atuador3 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/atua-
dor3");            //Publicação para atuador3
  Adafruit_MQTT_Publish atuador4 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/atua-
dor4");            //Publicação para atuador4
  Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/tempe-
ratura");      //Publicação da Temperatura
  Adafruit_MQTT_Publish umidade = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/umidade");
//Publicação da Umidade
  Adafruit_MQTT_Publish luminosidade = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/lumi-
nosidade");    //Publicação da Luminosidade
```

```
  Adafruit_MQTT_Publish amonia = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/amonia");
//Publicação do estado de NH3

//Inicializando ATUADORES
#define ATUADOR_1 13     //Constante de conexão da porta para o atuador1 (Relé)
#define ATUADOR_2 12     //Constante de conexão da porta para o atuador2 (Relé)
#define ATUADOR_3 14     //Constante de conexão da porta para o atuador3 (Relé)
#define ATUADOR_4 15     //Constante de conexão da porta para o atuador4 (Relé)

//Inicializando DHT
#include <SimpleDHT.h>         //Inclusão da biblioteca SimpleDHT, para leitura do sensor DHT22
#define DHTPORT D3             //Constante de conexão da porta para o sensor DHT22
SimpleDHT22 DHT22(DHTPORT);    //Criação do objeto DHT22 na porta DHTPORT para leitura

//Inicializando LDR
#define LDRPORT A0             //Constante de conexão da porta analógica para o sensor LDR

//Inicializando MQ135
#define MQ135PORT 16           //Constante de conexão da porta para o sensor MQ135

//Inicializando o BUZZER
#define BUZZER 2               //Constante de conexão da porta para o buzzer, apito.

//Inicializando o RTC (DS1307)
#include <Wire.h>              //Biblioteca auxiliar de funcionamento para o DS1307
#include <RTC.h>               //Biblioteca para leitura e escrita dos dados do DS1307
static DS1307 RTC;             //Criação do objeto de trabalho com o DS1307

//Inicializando o SPIFFS (Leitura e gravação de arquivos)
#include <FS.h>                //Biblioteca para trabalho com arquivos no SPIFFS (Armazenamento
não volátil)

//Variáveis de trabalho
bool dht_erro = false;                    //Variável para tratamento de erro na leitura do
sensor DHT22
bool modo_teste = false;                  //Variável para sinalizar que o sistema está em modo
de teste
bool calibracao = false;                  //Variável para verificar se modo de calibração está
ativo
float leitura_temperatura = 0;            //Variável para armazenar o valor de temperatura
lido
float leitura_umidade = 0;                //Variável para armazenar o valor de umidade rela-
tiva lido
float leitura_luz = 0;                    //Variável para armazenar o valor de luminosidade
lido
String leitura_gas = "";                  //Variável para armazenar texto resultado da leitura
do sensor de gás MQ135
String leitura_gas_anterior = "";         //Variável para mudança no estado da Amônia
int dht_leit_ant = 60;                    //VAriável para controle do intervalo de leituras do
DHT22
String buf = "";                          //Variável para armazenar dados à serem lidos por
SPIFFS
static unsigned long ult_tempo = 0;       //Variável auxiliar na rotina de controle de tempo
com a classe millis();
int tempo_passado_s = 0;                  //Variável de controle de tempo determinado
bool atu1 = false;                        //Variável para armazenar o estado do atuador 1
bool atu2 = false;                        //Variável para armazenar o estado do atuador 2
bool atu3 = false;                        //Variável para armazenar o estado do atuador 3
bool atu4 = false;                        //Variável para armazenar o estado do atuador 4
bool automatico = true;                   //Variável que define o modo de operação dos atuado-
res para manual ou automático
bool buzina = false;                      //Variável para habilitar e desabilitar o buzzer

//Função para reinício da placa remotamente
void(* reinicia) (void) = 0;              //Função de acionamento da porta 0 que provoca o
reinício do microcontrolador

//Função para criação do arquivo "dados.txt"
void cria_arquivo(void){                                //Cria função
  File wFile;                                           //Variável para o arquivo
  if(SPIFFS.exists("/dados.txt")){                      //Verifica se o arquivo existe
    Serial.println("Arquivo já existe!");               //Informa que já existe
  }
  else {                                                //Caso contrário cria um novo arquivo
    Serial.println("Criando arquivo ...");              //Informa via serial
    wFile = SPIFFS.open("/dados.txt","w+");             //Abre arquivo para escrita
```

```
    if(!wFile) {                                        //Caso ocorra um erro
      Serial.println("Erro ao criar arquivo.");         //Informa via serial
    }
    else {                                              //Caso não ocorra nenhum erro
      Serial.println("Arquivo criado com sucesso.");    //Informa via serial
    }
  }
  wFile.close();                                        //Fecha o arquivo
}

//Função para deletar o arquivo "dados.txt"
void deleta_dados(void) {                                           //Cria a função
  if(SPIFFS.remove("/dados.txt")) {                                 //Deleta o arquivo
    Serial.println("Arquivo deletado.");                            //Informa via serial
  }
  else {                                                            //Caso ocorra erro
    Serial.println("Erro ao remover o arquivo.");                   //Informa via serial
  }
}

//Função que anexa uma string ao final do arquivo "/dados.txt"
void anexa_dados(String msg) {                                      //Cria função que recebe
string
  File rFile = SPIFFS.open("/dados.txt","a+");                      //Abre o arquivo para adi-
ção (append)
  if(!rFile){                                                       //Caso ocorra erro
    Serial.println("Erro ao escrever no final do arquivo.");        //Informa via serial
  } else {                                                          //Caso não ocorra erro
    rFile.println(msg);                                             //Acrescenta na linha
  }
  rFile.close();                                                    //Fecha arquivo
}

//Função de controle dos atuadores
void Atuador(int atu) {                                         //Cria função atuador
  switch (atu) {                                                //Função de tratamento para nú-
mero do atuador solicitado
    case 1:                                                     //Caso atuador 1
    digitalWrite(ATUADOR_1,!digitalRead(ATUADOR_1));            //Inverte o sinal da porta do
Atuador 1, entre HIGH e LOW
    if (digitalRead(ATUADOR_1) == HIGH) { Serial.println("Atuador 1 - Ligado"); anexa_da-
dos("\nAtuador 1 - Ligado;\n"); }
    else { Serial.println("Atuador 1 - Desligado"); anexa_dados("\nAtuador 1 - Desligado;\n");
}
    break;                                                      //Sai da função switch
    case 2:                                                     //Idem para atuador 2
    digitalWrite(ATUADOR_2,!digitalRead(ATUADOR_2));
    if (digitalRead(ATUADOR_2) == HIGH) { Serial.println("Atuador 2 - Ligado"); anexa_da-
dos("\nAtuador 2 - Ligado;\n"); }
    else { Serial.println("Atuador 2 - Desligado"); anexa_dados("\nAtuador 2 - Desligado;\n");
}
    break;
    case 3:                                                     //Idem para atuador 3
    digitalWrite(ATUADOR_3,!digitalRead(ATUADOR_3));
    if (digitalRead(ATUADOR_3) == HIGH) { Serial.println("Atuador 3 - Ligado"); anexa_da-
dos("\nAtuador 3 - Ligado;\n"); }
    else { Serial.println("Atuador 3 - Desligado"); anexa_dados("\nAtuador 3 - Desligado;\n");
}
    break;
    case 4:                                                     //Idem para atuador 4
    digitalWrite(ATUADOR_4,!digitalRead(ATUADOR_4));
    if (digitalRead(ATUADOR_4) == HIGH) { Serial.println("Atuador 4 - Ligado"); anexa_da-
dos("\nAtuador 4 - Ligado;\n"); }
    else { Serial.println("Atuador 4 - Desligado"); anexa_dados("\nAtuador 4 - Desligado;\n");
}
    break;
  }
}

//Função para limitar o acionamento de algum atuador para menos de 3 minutos
void tempo_atuador(int atuador, int tempo){                         //Cria a função para que
determinado atuador possa ser temporizado em menos de 3 minutos
  if (atuador == 1 && tempo_passado_s >= tempo && atu1 == true) { Atuador (atuador); }  //Ve-
rifica se o atuador selecionado está ativado e se já alcançou o tempo determinado, desligando-
o
  if (atuador == 2 && tempo_passado_s >= tempo && atu2 == true) { Atuador (atuador); }  //Idem
para o atuador 2
```

```
  if (atuador == 3 && tempo_passado_s >= tempo && atu3 == true) { Atuador (atuador); }  //Idem
para o atuador 3
  if (atuador == 4 && tempo_passado_s >= tempo && atu4 == true) { Atuador (atuador); }  //Idem
para o atuador 4
}                                                                    //Final da função moni-
tora_atuador

//Função para atualizar variáveis de atuadores
void verif_atuadores () {                                    //Cria a função de verificar
atuadores
  if (digitalRead(ATUADOR_1) == HIGH) { atu1 = true; }       //Caso a porta do atuador 1 es-
teja acionada, em HIGH, atualiza a variável atu1
  else { atu1 = false; }                                     //Caso contrário atualiza a va-
riável para false
  if (digitalRead(ATUADOR_2) == HIGH) { atu2 = true; }       //Idem para atuador 2
  else { atu2 = false; }
  if (digitalRead(ATUADOR_3) == HIGH) { atu3 = true; }       //Idem para atuador 3
  else { atu3 = false; }
  if (digitalRead(ATUADOR_4) == HIGH) { atu4 = true; }       //Idem para atuador 4
  else { atu4 = false; }
}

//Função para leitura de temperatura e umidade
void Temp_Umid() {
//Cria a função de leitura de Temp. e Umidade
  dht_leit_ant = RTC.getMinutes();
//Atualiza variável para leitura a cada 1 minuto
  float temperatura = 0;
//Variável local para armazenar temperatura
  float umidade = 0;
//Variável local para armazenar umidade
  int err = SimpleDHTErrSuccess;
//Variável com estado de leitura do DHT22
  if ((err = DHT22.read2(&temperatura, &umidade, NULL)) != SimpleDHTErrSuccess) {
//Caso tenha algum erro na leitura
    Serial.print("Falha ao ler sensor DHT22, err="); Serial.print(SimpleDHTErrCode(err));
//Envia mensagem e código de erro via porta serial
    Serial.print(","); Serial.println(SimpleDHTErrDuration(err));
//Envia duração do erro na porta serial
    Buzzer(3, 150);
//Em caso de erro no DHT22 um sinal sonoro de 3 bips longos é acionado
    dht_erro = true;
//Atualiza a variável de erro no DHT para true
    leitura_temperatura = 0;
//Zera a variável global para temperatura
    leitura_umidade = 0;
//Zera a variável global para umidade
  }
//Fim do tratamento de erro
  else {
//Caso a leitura seja bem sucedida
    dht_erro = false;
//Atualiza a variável de erro no DHT para false
    leitura_temperatura = temperatura - 3;
//Atualiza variável global de temperatura
    leitura_umidade = umidade;
//Atualiza variável global de umidade
  }
//Final do tratamento das leituras
}

//Função para leitura da luminosidade
void Luz () {                                                //Cria a função de leitura
  float vLDR = (3.3 / 1024) * analogRead(LDRPORT);           //Retorna o valor de voltagem
para vLDR, da porta LDRPORT
  float ResistLDR = ((10000 * 3.3) / vLDR)-10000;            //Retorna o valor de resistân-
cia para ResistLDR utilizando cálculo de divisão de tensão
  float luminosidade = 2242931 * pow(ResistLDR,-1.3011);     //Calcula o valor em Lux de
luminosidade que incide sobre o LDR 5528
  if (luminosidade >= 100) { luminosidade = 100; }           //Levando em conta o limite do
sensor, se o valor em LUX for igual ou maior que 100, será considerado como 100 LUX
  else if (luminosidade < 0.5) { luminosidade = 0; }         //Levando em conta o limite do
sensor, se o valor em LUX for menor que 1, será considerado como ZERO
  leitura_luz = luminosidade;                                //Atualiza variável global de
luminosidade
}
```

```
//Função para calibragem da porta digital do sensor MQ-135
//Para realizar a calibragem da porta digital do sensor MQ-135 usa-se momentâneamente a porta
analógica, devendo-se desconectar
//o LDR da porta A0 e ligar a porta de dados analógicos do MQ-135 em seu lugar, em seguida
deve-se posicionar o resitor
//do sensor MQ-135 até que o sinal desejado fique com a concentração desejada. O sensor neste
momento deve estar exposto
//a amônia na concentração informada na porta analógica, podendo-se então fazer a comparação.
void Calib_Gas () {
//Criação da função de calibração
  float vLDR = (3.3 / 1024) * analogRead(LDRPORT);
//Retorna o valor de voltagem para o sensor de gás na porta A0 do LDR
  float ResistLDR = ((22000 * 3.3) / vLDR)-22000;
//Retorna o valor de resistância para a porta A0
  float amonia = 2242931 * pow(ResistLDR,-1.3011);
//Calcula o valor em ppm de amonia que incide sobre o MQ-135
  if (amonia >= 300) { amonia = 300; }
//Levando em conta o limite do sensor, se o valor de Amônia for igual ou maior que 300, será
considerado como 300 ppm
  else if (amonia < 10) { amonia = 10; }
//Levando em conta o limite do sensor, se o valor em Amônia for menor que 10, será considerado
como 10 ppm ou AUSENTE
  Gas ();
//Faz a leitura do sensor de gás
  Serial.println("AMÔNIA DIGITAL: " + leitura_gas + " // AMÔNIA ANALÓGICO: " + String(amo-
nia));     //Envia dados para comparação via porta serial
}

//Função para acionamento do Buzzer
void Buzzer (int vez,int tempo) {          //Cria a função de acionamento do buzzer, apito.
  if (buzina == true) {                    //Executa se a buzina estiver habilitada
    for (int i = 0; i < vez; i++) {        //Laço que faz tocar pelo número de vezes solicitado
      digitalWrite(BUZZER, LOW);           //Porta do buzzer em estado de nível baixo
      delay(tempo);                        //Para o microcontrolador pelo tempo solicitado
      digitalWrite(BUZZER, HIGH);          //Muda o estado da porta para alto
      delay(tempo);                        //Para o microcontrolador pelo tempo solicitado,
mantendo o apito
    }
  }
}

//Função para toque de alarme de presença de amônia
void Alarme () {                          //Cria uma função de alarme
  Buzzer (10, 200);                       //Ativa o buzzer por 10 vez e 200 ms de intervalo
}

//Função para leitura do sensor MQ-135
void Gas () {                             //Cria função para leitura de gás no sensor MQ 135
  int gas = digitalRead(MQ135PORT);       //Lê porta digital do sensor
  String saida = "";                      //Zera a string de saida da função
  if (gas ==1) {saida = "Ausente";}       //Caso o sensor retorne 1 não há presença de gás
  else {saida = "Presente";}              //Caso contrário altera variável para Presente
  leitura_gas = saida;                    //Atualiza variável global de leitura de gás
}

//Função para leitura do dia e hora
String Data_Hora() {                        //Criação da função
  String saida = "";                        //Variável de saída
  switch (RTC.getWeek())                    //Seleção para o dia da semana
  {
    case 1:                                 //Caso retorne 1
      saida = "DOM-";  break;               //Variável saída será domingo
    case 2:                                 //Caso retorne 2
      saida = "SEG-";  break;               //Variável saída será segunda
    case 3:                                 //e assim sucessivamente
      saida = "TER-";  break;
    case 4:
      saida = "QUA-";  break;
    case 5:
      saida = "QUI-";  break;
    case 6:
      saida = "SEX-";  break;
    case 7:
      saida = "SAB-";  break;
  }
```

```
  saida = saida + RTC.getHours() + ":" + RTC.getMinutes() + ":" + RTC.getSeconds() + "-";
//Adiciona o tempo na variável de saída
  saida = saida + RTC.getDay() + "/" + RTC.getMonth() + "/" + RTC.getYear();
//Adiciona a data na variável de saída
  return saida;
//Retorna string com dia, data e hora.
}

//Função para a atualização do dia, data e hora no DS 1307
void Atua_Data_Hora (String tempo) {                    //Cria a função de atualização
                                                        //Uma variável para cada campo, como
descrito abaixo, a partir de substring
  int semana = (tempo.substring(0,2)).toInt();          //01-DOM, 02-SEG, 03-TER, 04-QUA, 05-
QUI, 06-SEX, 07-SAB
  int hora = (tempo.substring(2,4)).toInt();            //01 ATÉ 23
  int minuto = (tempo.substring(4,6)).toInt();          //01 ATÉ 59
  int segundo = (tempo.substring(6,8)).toInt();         //00 ATÉ 59
  int dia = (tempo.substring(8,10)).toInt();            //01 ATÉ 31
  int mes = (tempo.substring(10,12)).toInt();           //01 ATÉ 12
  int ano = (tempo.substring(12,14)).toInt();           //00 ATÉ 99 (2000-2099)

  RTC.setWeek(semana);                                  //Armazena o dia da semana no DS 1307
  RTC.setTime(hora,minuto,segundo);                     //Armazena a hora minuto e segundo
  RTC.setDate(dia,mes,ano);                             //Armazena o dia mês e ano
}

//Função para teste dos diferentes dispositivos da placa
void Modo_Teste () {                                                        //Cria a função
de teste

  Serial.println("-----------------------------------------------------");     //Delimitador na
saída serial do início do ciclo
  Serial.println(Data_Hora());                                              //Envia na porta
serial o tempo atual no DS 1307
  Serial.println("UMIDADE: " + String(leitura_umidade) + " %");             //Envia a umi-
dade na porta serial
  delay(1000);                                                              //Aguarda 1 se-
gundo
  Serial.println("AMÔNIA: " + leitura_gas);                                 //Envia a lei-
tura de gás
  delay(1000);                                                              //Aguarda 1 se-
gundo
  Serial.println("LUMINOSIDADE: " + String(leitura_luz) + " Lux");          //Envia a lei-
tura de lumnosidade
  delay(1000);                                                              //Aguarda 1 se-
gundo
  Serial.println("TEMPERATURA: " + String(leitura_temperatura) + " *C");    //Envia a lei-
tura de temperatura
  delay(1000);                                                              //Aguarda 1 se-
gundo
  Buzzer(1,50);             //Toca apito para testar o buzzer
  Atuador(1);               //Alterna sinal do atuador 1
  delay(3000);              //Aguarda 3 segundos
  Atuador(2);               //Alterna sinal do atuador 2
  delay(3000);              //Aguarda 3 segundos
  Atuador(3);               //Alterna sinal do atuador 3
  delay(3000);              //Aguarda 3 segundos
  Atuador(4);               //Alterna sinal do atuador 4
  delay(3000);              //Aguarda 3 segundos
}

//Função de conexão MQTT
void MQTT_connect() {                                   //Cria a função de conexão MQTT
  int8_t ret;                                           //Variável auxiliar para erros de
conexão
  if (mqtt.connected()) {                               //Se já estiver conectado sai da
função
    return;                                             //Sai da função
  }
  Serial.print("Conectando MQTT... ");                  //Envia mensagem na porta serial de
conexão MQTT
  uint8_t retries = 10;                                 //Variável auxiliar para contar ten-
tativas de conexão
  while ((ret = mqtt.connect()) != 0) {                 //Executa enquanto não houver cone-
xão MQTT
       Serial.println(mqtt.connectErrorString(ret));    //Se ocorrer erro, envia erro via
serial
```

```
      Serial.println("Reconectando em 3s...");          //Envia mensagem de reconexão na
porta serial
      mqtt.disconnect();                               //Desconecta o MQTT
      for (int i = 1; i < 30; i++) {                   //Executa um loop para que o micro-
controlador não fique parado muito tempo
      delay(100);                                      //Aguarda 100 milisegundos
      }
      retries--;                                       //Subtrai 1 da variável auxiliar de
contagem de conexões
      if (retries == 0) {                              //Executa se não houver mais tenta-
tivas
        break;                                         //Encerra o loop while
      }
  }
  yield();
}
//Função para controle de tempos
void tempo (int intervalo) {                           //Cria a função tempo
  if ((millis() - ult_tempo) >= 1000) {                //Se a variável de sistema millis
completar 1 segundo passado
    tempo_passado_s ++;                                //Acresce 1 a variável tempo_pas-
sado s
    ult_tempo = millis();                              //Atualiza a variável ult_tempo
  }
  if (tempo_passado_s >= intervalo) { tempo_passado_s = 0; }     //Se tempo_passado_s for
maior que intervalo atualiza
}

//Função de transmissão MQTT
void Transmite_MQTT () {                                                    //Cria a fun-
ção de transmissão

    verif_atuadores ();                                                     //Verifica o
estado dos atuadores para enviar o estado atualizado

    int i_amonia = 0;                                                       //Variável in-
teira local para status do gás
    if (leitura_gas == "Ausente") { i_amonia = 0; } else { i_amonia = 1; }      //Trata lei-
tura afim  de transformar string em inteiro

    if (dht_erro == false) {                                                //Transmite a
temperatura e umidade somente se a leitura não tiver retornado erro
      if (! temperatura.publish(leitura_temperatura)) {                     //Se ocorrer
erro na transmissão da temperatura
        Serial.println("Falha na publicação da temperatura"); }             //Informa erro
via serial
      else {                                                                //Executa caso
não tenha erro
       Serial.println("Temperatura publicada! (" + String(leitura_temperatura, 2) + " *C)"); }
//Informa via serial que publicou com sucesso
      if (! umidade.publish(leitura_umidade)) {
//Se ocorrer erro na transmissão da umidade
        Serial.println("Falha na publicação da umidade"); }
//Informa erro via serial
      else {
//Executa caso não tenha erro
        Serial.println("Umidade publicada! (" + String(leitura_umidade, 2) + " %)"); }
//Informa via serial que publicou com sucesso
    }
    if (! luminosidade.publish(leitura_luz)) {
//Se ocorrer erro na transmissão da luminosidade
      Serial.println("Falha na publicação da luminosidade"); }
//Informa erro via serial
    else {
//Executa caso não tenha erro
      Serial.println("Luminosidade publicada! (" + String(leitura_luz, 2) + " Lux)"); }
//Informa via serial que publicou com sucesso
    if (! amonia.publish(i_amonia)) {
//Se ocorrer erro na transmissão da Amônia
      Serial.println("Falha na publicação de NH3"); }
//Informa erro via serial
    else {
//Executa caso não tenha erro
      if ( i_amonia == 0 ) { Serial.println("NH3 publicado! (Ausente)"); }
//Informa via serial que publicou com sucesso (Amônia Ausente)
      else { Serial.println("NH3 publicado! (Presente)"); }
//Informa via serial que publicou com sucesso (Amônia Presente)
```

```
    }

    int operacao = 0;
//Cria variável local para converter modo
    if (automatico == true) { operacao = 1; } else { operacao = 0; }
//Faz a conversão de booleano para inteiro
    if (! modo.publish(operacao)) {
//Se ocorrer erro na transmissão do modo de operação
      Serial.println("Falha na publicação do modo de operação"); }
//Informa erro via serial
    else {
//Executa caso não tenha erro
      if ( operacao == 0 ) { Serial.println("Modo de operação publicado! (Manual)"); }
//Informa via serial que publicou com sucesso (Manual)
      else { Serial.println("Modo de operação publicado! (Automático)"); } }
//Informa via serial que publicou com sucesso (Automático)

    int conv_atu1 = 0;
//Cria variável local para converter booleano para inteiro
    if (atu1 == true) { conv_atu1 = 1; } else { conv_atu1 = 0; }
//Faz a conversão de booleano para inteiro
    if (! atuador1.publish(conv_atu1)) {
//Se ocorrer erro na transmissão do Atuador
      Serial.println("Falha na publicação do Atuador #1"); }
//Informa erro via serial
    else {
//Executa caso não tenha erro
      if (conv_atu1 == 1) {Serial.println("Atuador #1 publicado! (Ligado)"); }
//Informa via serial que publicou com sucesso (Ligado)
      else { Serial.println("Atuador #1 publicado! (Desligado)"); } }
//Informa via serial que publicou com sucesso (Desligado)

    int conv_atu2 = 0;
//Idem para Atuador #1
    if (atu2 == true) { conv_atu2 = 1; } else { conv_atu2 = 0; }
    if (! atuador2.publish(conv_atu2)) {
      Serial.println("Falha na publicação do Atuador #2"); }
    else {
      if (conv_atu2 == 1) {Serial.println("Atuador #2 publicado! (Ligado)"); }
      else { Serial.println("Atuador #2 publicado! (Desligado)"); } }

    int conv_atu3 = 0;
//Idem para Atuador #1
    if (atu3 == true) { conv_atu3 = 1; } else { conv_atu3 = 0; }
    if (! atuador3.publish(conv_atu3)) {
      Serial.println("Falha na publicação do Atuador #3"); }
    else {
      if (conv_atu3 == 1) {Serial.println("Atuador #3 publicado! (Ligado)"); }
      else { Serial.println("Atuador #3 publicado! (Desligado)"); } }

    int conv_atu4 = 0;
//Idem para Atuador #1
    if (atu4 == true) { conv_atu4 = 1; } else { conv_atu4 = 0; }
    if (! atuador4.publish(conv_atu4)) {
      Serial.println("Falha na publicação do Atuador #4"); }
    else {
      if (conv_atu4 == 1) {Serial.println("Atuador #4 publicado! (Ligado)"); }
      else { Serial.println("Atuador #4 publicado! (Desligado)"); } }
}

//Função para formatar o SPIFFS
void formata(void){                          //Cria função
  Serial.print("Formatando SPIFFS...");      //Informa formatação via serial
  SPIFFS.format();                           //Apaga o conteúdo do SPIFFS
  Serial.println(" OK");                     //Informa finalização
}

//Função que lê arquivo "/dados.txt"
void le_dados_html(void) {                                  //Cria função
  buf = "";                                                 //Zera variável de buffer
  File rFile = SPIFFS.open("/dados.txt","r");               //Faz a abertura do arquivo
  Serial.println("Lendo arquivo...");                       //Informa a leitura do arquivo
  server.sendContent ("<p>Início dos dados.</p><p>");
  while(rFile.available()) {                                //Enquanto não acabar a leitura do ar-
quivo
    buf = rFile.readString();
    buf += "<br>";
```

```
    server.sendContent (buf);
    buf = "";


//    server.sendContent (rFile.readString());
//    server.sendContent ("<br>");
  }
  server.sendContent ("</p><p>Fim dos dados.</p>");
  rFile.close();                                        //Fecha arquivo
}

//Função que inicia sistema de arquivos SPIFFS
void abre_FS(void){                                                 //Cria função
  if(!SPIFFS.begin()){                                              //Tenta abrir o sistema de
arquivos SPIFFS
    Serial.println("Erro ao abrir o sistema de arquivos.");       //Informa que houve um erro
  } else {                                                          //Caso não ocorra erro
    Serial.println("Sistema de arquivos aberto com sucesso.");    //Informa via serial
  }
}
//Função que fecha sistema de arquivos SPIFFS
void fecha_FS(void){                                //Cria função
  SPIFFS.end();                                     //Fecha sistema de arquivos
}

//Função que envia dados pela porta serial para ser lido no monitor serial
void enviaSerial(void){                                             //Cria função
  File rFile = SPIFFS.open("/dados.txt","r");                       //Faz a abertura do arquivo
  Serial.println("***Lendo arquivo...***");                         //Informa a leitura do início do
arquivo
  while(rFile.available()) {                                        //Enquanto não acabar a leitura do
arquivo
  Serial.println(rFile.readString());                               //Envia a linha via serial
  }
  rFile.close();                                                    //Fecha arquivo
  Serial.println("***Leitura Finalizada!***");                      //Informa final da leitura do ar-
quivo
}
//Função que armazena as leituras no arquivo "dados.txt"
void gravaLEITURAS(void){                                           //Cria a função
      //Envia uma linha ao final do arquivo de dados.txt
      anexa_dados(Data_Hora() + ";" + String(leitura_temperatura) + ";" + String(leitura_umi-
dade) + ";" + String(leitura_luz) + ";" + leitura_gas + ";");
}

//Função para acionamento de atuador por amônia
void trata_amonia(int atuador){                                     //Cria a função
  if ( leitura_gas_anterior != leitura_gas ) {                      //Mudança no estado da Amônia
    Atuador(atuador);                                               //Aciona atuador
    leitura_gas_anterior = leitura_gas;                             //Atualiza a variável
  }
  if (leitura_gas == "Presente") { Alarme(); }                      //Soa alarme caso ainda haja amô-
nia
}

//Função para acionamento de atuador por luz
void trata_luz(int atuador, float lux, int hora_inic, int hora_term, int hora_in_apa, int
hora_ter_apa){     //Cria a função
  bool trata_luz = false;                                                   //Variável local para es-
tado do relé de luz
  if (atuador == 1) {trata_luz = atu1; }                                    //Verifica se está utili-
zando o atuador 1
  if (atuador == 2) {trata_luz = atu2; }                                    //Verifica se está utili-
zando o atuador 2
  if (atuador == 3) {trata_luz = atu3; }                                    //Verifica se está utili-
zando o atuador 3
  if (atuador == 4) {trata_luz = atu4; }                                    //Verifica se está utili-
zando o atuador 4
  if (RTC.getHours() >= hora_inic && RTC.getHours() < hora_term) {  //No intervalo de tempo
selecionado mantém a luz acesa
        if (trata_luz == false) { Atuador(atuador); }
  }
  else if ((RTC.getHours() >= hora_in_apa && RTC.getHours() > hora_ter_apa) || (RTC.getHours()
<= hora_in_apa && RTC.getHours() < hora_ter_apa)) {      //No intervalo de tempo selecionado
mantém a luz apagada
        if (trata_luz == true) { Atuador(atuador); }
  }
```

```
  else {
    if (leitura_luz < lux) {                                    //Alterna o atuador se a leitura
de luz for inferior e se atuador estiver desligado
      if (trata_luz == false) { Atuador(atuador); }             //Alterna o estado do atuador
    }
    if (leitura_luz >= lux) {                                   //Alterna o atuador se a leitura
de luz for superior e se atuador estiver ligado
      if (trata_luz == true) { Atuador(atuador); }              //Alterna o estado do atuador
    }
  }
}

//Função para acionamento de atuador por temperatura quente
void trata_temp(int atuador, float temp){       //Cria a função
  bool trata_temp = false;                      //Variável local para estado do relé de tempe-
ratura
  if (atuador == 1) {trata_temp = atu1; }       //Verifica se está utilizando o atuador 1
  if (atuador == 2) {trata_temp = atu2; }       //Verifica se está utilizando o atuador 2
  if (atuador == 3) {trata_temp = atu3; }       //Verifica se está utilizando o atuador 3
  if (atuador == 4) {trata_temp = atu4; }       //Verifica se está utilizando o atuador 4
    if (leitura_temperatura > temp && trata_temp == false) {                //Alterna o atua-
dor se leitura maior que temperatura desejada e atuador estiver desligado
      Atuador(atuador);                                                     //Alterna o estado
do atuador
    }
    if (leitura_temperatura < temp && trata_temp == true) {                 //Alterna o atua-
dor se leitura menor que temperatura desejada e atuador estiver ligado
      Atuador(atuador);                                                     //Alterna o estado
do atuador
    }
}

//Função para acionamento de atuador por temperatura fria
void trata_temp_fria(int atuador, float temp_fria){     //Cria a função
  bool trata_temp_fria = false;                         //Variável local para estado do relé
de temperatura
  if (atuador == 1) {trata_temp_fria = atu1; }          //Verifica se está utilizando o atua-
dor 1
  if (atuador == 2) {trata_temp_fria = atu2; }          //Verifica se está utilizando o atua-
dor 2
  if (atuador == 3) {trata_temp_fria = atu3; }          //Verifica se está utilizando o atua-
dor 3
  if (atuador == 4) {trata_temp_fria = atu4; }          //Verifica se está utilizando o atua-
dor 4
    if (leitura_temperatura < temp_fria && trata_temp_fria == false) {        //Alterna o atu-
ador se leitura maior que temperatura desejada e atuador estiver desligado
      Atuador(atuador);                             //Alterna o estado do atuador
    }
    if (leitura_temperatura >= temp_fria && trata_temp_fria == true) {         //Alterna o
atuador se leitura menor que temperatura desejada e atuador estiver ligado
      Atuador(atuador);                             //Alterna o estado do atuador
    }
}

//Função para acionamento de atuador por umidade alta
void trata_umid(int atuador, float umid){       //Cria a função
  bool trata_umid = false;                      //Variável local para estado do relé de umi-
dade
  if (atuador == 1) {trata_umid = atu1; }       //Verifica se está utilizando o atuador 1
  if (atuador == 2) {trata_umid = atu2; }       //Verifica se está utilizando o atuador 2
  if (atuador == 3) {trata_umid = atu3; }       //Verifica se está utilizando o atuador 3
  if (atuador == 4) {trata_umid = atu4; }       //Verifica se está utilizando o atuador 4
    if (leitura_umidade > umid && trata_umid == false) {                    //Alterna o atua-
dor se leitura maior que umidade desejada e atuador estiver desligado
      Atuador(atuador);                                                     //Alterna o estado
do atuador
    }
    if (leitura_umidade <= umid && trata_umid == true) {                    //Alterna o atua-
dor se leitura menor que umidade desejada e atuador estiver ligado
      Atuador(atuador);                                                     //Alterna o estado
do atuador
    }
}

//Função para tratar entradas na porta serial
String trata_serial (String texto) {            //Cria função
    if ((texto.substring(0,5)) == "tempo") {    //Caso o texto enviado se inicie por tempo
```

```
    Atua_Data_Hora(texto.substring(5,19));        //Envia a string para a função de atualizar
o tempo em DS 1307
    Serial.println(Data_Hora());                  //Informa nova hora via serial
    }
    else if (texto == "reinicia") {               //Caso o texto seja reinicia
    reinicia();                                   //Reinicia o microcontrolador
    }

    else if (texto == "modoteste") {              //Caso o texto seja modoteste
    modo_teste = !modo_teste;                     //Muda a variável afim de ligar o modo
teste, ou desligá-lo
    }
    else if (texto == "calibracao") {             //Caso o texto seja calibracao
    calibracao = !calibracao;                     //Muda a variável afim de ligar o modo de
calibração ou desligá-lo
    }

    else if (texto == "enviaserial") {            //Caso o texto seja enviaserial
      enviaSerial();                              //Envia dados do arquivo dados.txt via porta
serial
    }

    else if (texto == "atuador1") {               //Caso o texto seja atuador1
      Atuador(1);                                 //Alterna o Atuador 1
    }
    else if (texto == "atuador2") {               //Idem para atuador 2
      Atuador(2);
    }
    else if (texto == "atuador3") {               //Idem para atuador 3
      Atuador(3);
    }
    else if (texto == "atuador4") {               //Idem para atuador 4
      Atuador(4);
    }
}
//Variáveis globais de construção HTML
//Variável do início dos documentos html
String htmlinicio = "<!DOCTYPE HTML><meta charset ='utf-8'/><meta name='viewport' con-
tent='width=device-width, initial-scale=1'><html>";
//Variável do final dos documentos html
String htmlfim = "</html>";

//Função HTTP
//Cria Função
void pagina_http () {

String operacao = "";     //Variável para conversão do formato do modo de operação de booleana
para string
if (automatico == true) { operacao = "Automático"; } else { operacao = "Manual"; } //Conversão
do formato

String modo_buzina = "";     //Variável para conversão do formato do modo de buzina de boole-
ana para string
if (buzina == true) { modo_buzina = "Ligada"; } else { modo_buzina = "Desligada"; } //Conver-
são do formato

//HTML da pagina principal
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}"); //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Situa&ccedil;&atilde;o Atual - MICROCONTROLADOR</h2>Data e
hora do sistema: " + Data_Hora());
         server.sendContent ("<p>Temperatura atual: " + String(leitura_temperatura) +
"&#176;C<br>");     //Anexa no HTML a temperatura
```

```
        server.sendContent ("Umidade relativa atual: " + String(leitura_umidade) + "%<br>");
//Anexa no HTML a umidade
        server.sendContent ("Luminosidade: " + String(leitura_luz) + " Lux<br>");
//Anexa no HTML a luminosidade
        server.sendContent ("Presen&ccedil;a de Am&ocirc;nia (NH3): " + leitura_gas +
"<br>");           //Anexa no HTML a leitura de amonia
        server.sendContent ("Modo de operação: " + operacao + "<br>");
//Anexa no HTML o modo de operação
        server.sendContent ("Buzzer (buzina): " + modo_buzina + "<br>");
//Anexa no HTML o modo de operação
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b3\" va-
lue=\"Atualizar\" onclick=\"location.href='/'\"></p>");                //Anexa botão Atualizar
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b4\" value=\"Ver
DADOS\" onclick=\"location.href='/dados'\"></p>");           //Anexa botão Ver DADOS
        server.sendContent ("<h3>Atuadores</h3>");
        server.sendContent ("<input class='button' type=\"button\" name=\"b9\" value=\"Alter-
nar atuador #1\" onclick=\"location.href='/atuador1'\"> ");             //Anexa botão
para alternar atuador 1
        server.sendContent ("Estado: ");
        if (atu1 == true) { server.sendContent ("Ligado<br>"); } else { server.sendContent
("Desligado<br>"); }
        server.sendContent ("<input class='button' type=\"button\" name=\"b10\" value=\"Al-
ternar atuador #2\" onclick=\"location.href='/atuador2'\"> ");            //Anexa botão
para alternar atuador 2
        server.sendContent ("Estado: ");
        if (atu2 == true) { server.sendContent ("Ligado<br>"); } else { server.sendContent
("Desligado<br>"); }
        server.sendContent ("<input class='button' type=\"button\" name=\"b11\" value=\"Al-
ternar atuador #3\" onclick=\"location.href='/atuador3'\"> ");            //Anexa botão
para alternar atuador 3
        server.sendContent ("Estado: ");
        if (atu3 == true) { server.sendContent ("Ligado<br>"); } else { server.sendContent
("Desligado<br>"); }
        server.sendContent ("<input class='button' type=\"button\" name=\"b12\" value=\"Al-
ternar atuador #4\" onclick=\"location.href='/atuador4'\"> ");            //Anexa botão
para alternar atuador 4
        server.sendContent ("Estado: ");
        if (atu4 == true) { server.sendContent ("Ligado<br>"); } else { server.sendContent
("Desligado<br>"); }
        server.sendContent ("<h3>Avan&ccedil;ado</h3>");
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b13\" va-
lue=\"Alternar Modo\" onclick=\"location.href='/modo'\"></p>");                    //Anexa
botão que permite mudar o modo para manual
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b14\" va-
lue=\"Ligar/Desligar Buzina\" onclick=\"location.href='/buzina'\"></p>");          //Anexa
botão que permite desligar o buzzer
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b5\" value=\"De-
letar DADOS\" onclick=\"location.href='/delete'\"></p>");                   //Anexa botão que
permite deletar dados
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b6\" va-
lue=\"Formatar SPIFFS\" onclick=\"location.href='/format'\"></p>");                 //Anexa
botão para formatar SPIFFS
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b7\" value=\"En-
viar por Serial\" onclick=\"location.href='/enviadados'\"></p>");           //Anexa botão En-
viar por Serial
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b9\" value=\"Re-
iniciar\" onclick=\"location.href='/reinicia'\"></p>");                     //Anexa botão re-
iniciar microcontrolador
        server.sendContent (htmlfim);
}

//Função para exibir a página HTML de dados
void pagina_dados () {      //Cria a função
        server.sendContent (htmlinicio);
        server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
        server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
        server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
        server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
        server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
        server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
```

```
        server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
        server.sendContent ("<h2>Dados</h2>");         //Envia o cabeçalho para a página
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
        server.sendContent ("<a download=\"dados.html\" href=\"data:text/html;charset=UTF-8,"
+ htmlinicio);                                                    //Início da tag de download
           File rFile = SPIFFS.open("/dados.txt","r");         //Faz a leitura do arquivo
para o link de download
           while(rFile.available()) {
             String linha = rFile.readStringUntil('\n');
             server.sendContent (linha + "</br>");             //Insere os dados dentro do
código HTML para gerar o arquivo dados.txt
           }
           rFile.close();
        server.sendContent (htmlfim + "\">Download dos DADOS</a><br>");     //Final da tag de
donwload
        le_dados_html();
        server.sendContent ("</p>");    //Finaliza parágrafo
        server.sendContent (htmlfim);   //Encerra o html
}

//Função para exibir HTML que formata SPIFFS
void pagina_format () {     //Cria a função
formata();                                                  //Formata o SPIFFS
cria_arquivo();                                             //Cria o arquivo de dados
        server.sendContent (htmlinicio);
        server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
        server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
        server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
        server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
        server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
        server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
        server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
        server.sendContent ("<h3>SPIFFS formatado! Arquivo recriado!</h3>");
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
        server.sendContent (htmlfim);   //Encerra o html
}

//Função para exibir HTML de deletar o arquivo de dados
void pagina_delete () {      //Cria a função
deleta_dados();                                             //Deleta "/dados.txt" do SPIFFS
cria_arquivo();                                             //Cria arquivo "/dados.txt" caso ele
não exista
        server.sendContent (htmlinicio);
        server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
        server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
        server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
        server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
        server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
        server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
        server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
        server.sendContent ("<h3>Arquivo Deletado!</h3>");
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
        server.sendContent (htmlfim);   //Encerra o html
}
```

```
//Função para exibir HTML com solicitação de envio pela serial
void pagina_enviadados () {
enviaSerial();
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Dados enviados!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
         server.sendContent (htmlfim);   //Encerra o html
}

//Função para exibir HTML com alternância do atuador 1
void pagina_atuador1 () {
Atuador(1);
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Atuador 1 alternado!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
         server.sendContent (htmlfim);   //Encerra o html
}

//Função para exibir HTML com alternância do atuador 2
void pagina_atuador2 () {
Atuador(2);
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Atuador 2 alternado!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
         server.sendContent (htmlfim);   //Encerra o html
```

```
}

//Função para exibir HTML com alternância do atuador 3
void pagina_atuador3 () {
Atuador(3);
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Atuador 3 alternado!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
         server.sendContent (htmlfim);   //Encerra o html
}

//Função para exibir HTML com alternância do atuador 4
void pagina_atuador4 () {
Atuador(4);
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Atuador 4 alternado!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
         server.sendContent (htmlfim);   //Encerra o html
}

//Função para alternar modo entre manual e automático
void pagina_alternar_modo () {
         automatico = !automatico;
         server.sendContent (htmlinicio);
         server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
         server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
         server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
         server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
         server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
         server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
         server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
         server.sendContent ("<h2>Modo alternado!</h2>");
         server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
```

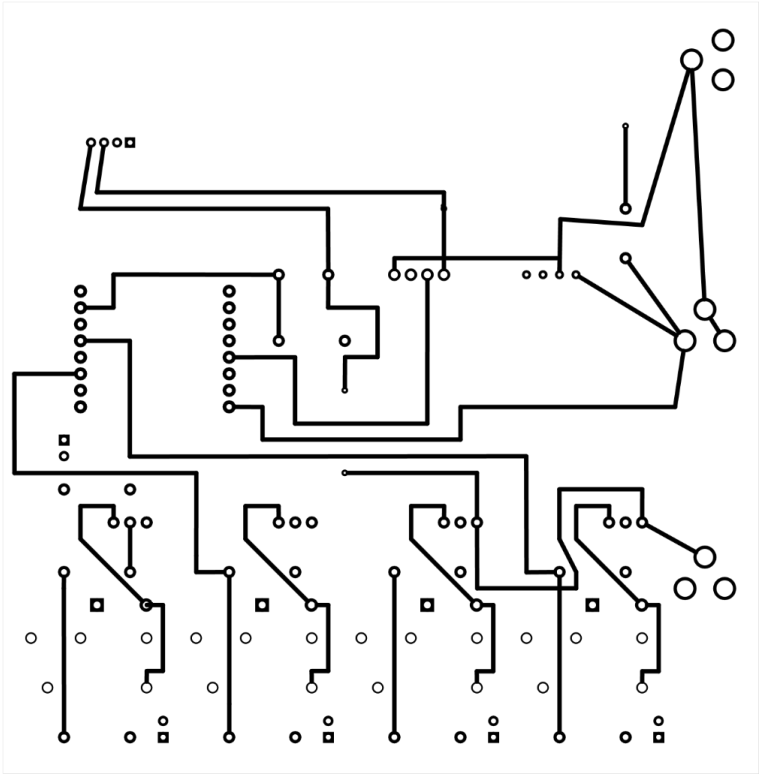
```
        server.sendContent (htmlfim);   //Encerra o html
}

//Função para alternar funcionamento buzzer
void pagina_alternar_buzzer () {
        buzina = !buzina;
        server.sendContent (htmlinicio);
        server.sendContent ("<style>body { color: #000000; font-family: Verdana, Helvetica,
sans-serif; font-size: 17px;}");  //Linhas para adição de código CSS à página
        server.sendContent ("h2 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 25px; }");
        server.sendContent ("h3 { color: #008CBA; text-shadow: 2px 2px 4px #bbbbbb; font-
size: 20px; }");
        server.sendContent (".button { font-family: Verdana, Helvetica, sans-serif; font-
size: 15px; transition-duration: 0.2s; background-color: white; color: black; border: 2px so-
lid #008CBA; padding: 8px 8px;");
        server.sendContent ("text-align: center; text-decoration: none; display: inline-
block; margin: 2px 1px; cursor: pointer; border-radius: 12px;");
        server.sendContent ("box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
0, 0, 0.19); }");
        server.sendContent (".button:hover { background-color: #008CBA; color: white;
}</style>");
        server.sendContent ("<h2>Buzina alternada!</h2>");
        server.sendContent ("<p><input class='button' type=\"button\" name=\"b2\" va-
lue=\"Voltar\" onclick=\"location.href='/'\"></p>");              //Insere o botão voltar na
página
        server.sendContent (htmlfim);   //Encerra o html
}

//Função para que reinicia o microcontrolador
void pagina_reinicia () {
reinicia();
}

//Função OTA - Over the Air
void OTA() {        //Cria a função

  //Define porta padrão de utilização
  ArduinoOTA.setPort(8266);

  //Define o nome padrão do microcontrolador para o IDE
  ArduinoOTA.setHostname("MINI");

  //Inicia o serviço de OTA
  ArduinoOTA.onStart([]() {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH) { //Se gravará sistema de arquivos ou programação
      type = "sketch";
    } else {
      type = "filesystem";
    }

    //Envia pela serial o início da atualização
    Serial.println("Começando atualização via OTA " + type);
  });
  //Executa quando termina a atualização
  ArduinoOTA.onEnd([]() {
    Serial.println("\nFim");        //Informa via serial o fim da atualização
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {     //Executa durante
atualização
    Serial.printf("Progresso: %u%%\r", (progress / (total / 100)));         //Informa via se-
rial durante a atualização
  });
  ArduinoOTA.onError([](ota_error_t error) {    //Caso haja erro na atualização
    Serial.printf("Erro[%u]: ", error);         //Informa o número do erro
    if (error == OTA_AUTH_ERROR) {              //Caso seja erro de autenticação
      Serial.println("Auth Falhou");
    } else if (error == OTA_BEGIN_ERROR) {      //Caso erro no início
      Serial.println("Begin Falhou");
    } else if (error == OTA_CONNECT_ERROR) {    //Caso erro na conexão
      Serial.println("Connect Falhou");
    } else if (error == OTA_RECEIVE_ERROR) {    //Caso erro de recepção
      Serial.println("Receive Falhou");
    } else if (error == OTA_END_ERROR) {        //Caso erro no fim
      Serial.println("End Falhou");
    }
```

```
  });
}

//Função para realizar a conexão com a rede de Wi-Fi local
void conecta_WIFI () {
    if (WiFi.status() != WL_CONNECTED) {                //Se não estiver conectado a rede Wi-
Fi executa
        int vezes = 0;                                  //Variável para permitir sair do loop
while após algum tempo
        WiFi.begin(WLAN_SSID, WLAN_PASS);               //Tenta realizar a conexão
        Serial.print("Tentando conexão Wi-Fi ...");     //Informa via serial a tentativa de
conexão
        while (WiFi.status() != WL_CONNECTED) {         //Executa enquanto aguarda a conexão
          delay(10);                                    //Faz um delay de 10 milisegundos
          vezes++;                                      //Incrementa a variável de controle
          if (vezes >= 1500) {                          //Executa se o tempo ultrapassar 15
segundos na tentativa de conexão
            Serial.print("Sem sucesso.");               //Informa via serial que não houve su-
cesso na tentativa
            break;                                      //Fecha o loop while
          }
        }
    }
        Serial.println();                               //Adiciona linha em branco na saída
serial
        if (WiFi.status() == WL_CONNECTED) {            //Executa caso consiga efetuar a cone-
xão Wi-Fi com êxito
          Serial.println();                             //Pula uma linha na saída serial
          Serial.print("Endereço IP da rede ");         //Adiciona texto na saída serial
          Serial.print(WLAN_SSID);                      //Informa o nome da rede a qual conse-
guiu se conectar via serial
          Serial.print(" : ");                          //Adiciona texto na saída serial
          Serial.println(WiFi.localIP());               //Adiciona o endereço IP local na sa-
ída serial
        }
}

//Laço de configuração
void setup() {

  //Configurando porta serial
  Serial.begin(115200);           //Inicia serial na velocidade 115200 bps
  delay(500);                     //Executa uma pausa de 500 ms no controlador

  //Abrindo sistema de arquivos e verificando ou criando arquivo "dados.txt"
  abre_FS();
  cria_arquivo();

  //Configurando pinos de E/S
  pinMode(DHTPORT, INPUT_PULLUP);           //Pino do DHT22 como entrada digital e resistor
interno
  pinMode(MQ135PORT, INPUT);        //Pino do MQ135 como entrada digital
  pinMode(ATUADOR_1, OUTPUT);       //Pino para atuador 1 como saída digital
  pinMode(ATUADOR_2, OUTPUT);       //Pino para atuador 2 como saída digital
  pinMode(ATUADOR_3, OUTPUT);       //Pino para atuador 3 como saída digital
  pinMode(ATUADOR_4, OUTPUT);       //Pino para atuador 4 como saída digital
  pinMode(LDRPORT, INPUT);          //Pino do LDR como entrada analógica
  pinMode(BUZZER, OUTPUT);          //Pino do Alarme como saída digital
  digitalWrite(BUZZER, HIGH);       //Desliga sinal do buzzer se estiver ativo
  Buzzer(3,50);                           //Emite som para avisar que chegou neste ponto

  //Configurando RTC
  RTC.begin();                            //Inicializa o RTC
  RTC.setHourMode(CLOCK_H24);             //Configura o RTC para modo de horas 24

  //Configurando Wi-Fi
  WiFi.mode(WIFI_AP_STA);             //Configurando o WI-FI para modo Estação e Ponto de
Acesso Simultâneos
  //Setando ips quando necessário
  //IPAddress local_IP(10,0,0,99);
  //IPAddress gateway(10,0,0,1);
  //IPAddress subnet(255,255,255,0);
  //WiFi.softAPConfig(local_IP, gateway, subnet);
  conecta_WIFI();

  WiFi.softAP(AP_SSID, AP_PASS);                                    //Inicializa o Wi-Fi no modo
de ponto de acesso
```

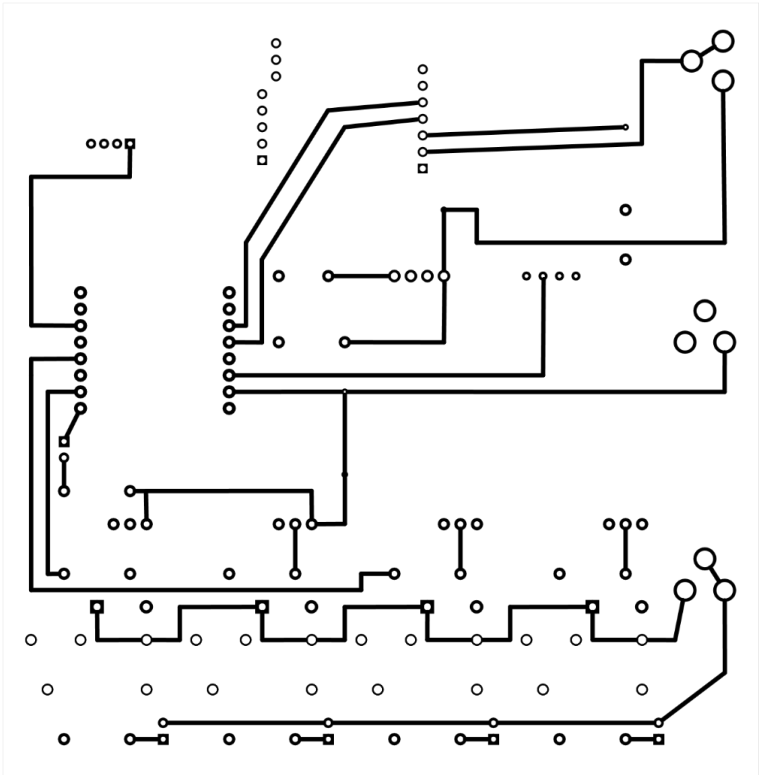
```
  Serial.println();
  Serial.print("Endereço IP da rede ");
  Serial.print(AP_SSID);
  Serial.print(" : ");
  Serial.print(WiFi.softAPIP());
  Serial.println();
  Serial.print("Senha da rede ");
  Serial.print(AP_SSID);                                          //Informa a senha de rede
para Ponto de Acesso
  Serial.print(" : ");
  Serial.print(AP_PASS);
  uint8_t macAddr[6];                                             //Cria variável com MAC
Address do microcontrolador
  WiFi.softAPmacAddress(macAddr);                                 //Captura endereço de MAC
  Serial.println();
  //Envia via serial o endereço de MAC
  Serial.printf("MAC address = %02x:%02x:%02x:%02x:%02x:%02x\n", macAddr[0], macAddr[1], macA-
ddr[2], macAddr[3], macAddr[4], macAddr[5]);


  //Inicializando OTA
  OTA ();                           //Inicializa rotinas para envio via OTA
  ArduinoOTA.begin();               //Inicializa o serviço de envio via OTA

  //Inicializa servidor WEB
  server.on("/", pagina_http);                              //Encaminha para página principal se
requisição for igual a "/"
  server.on("/dados", pagina_dados);                        //Encaminha para página de dados se
requisição for igual a "/dados"
  server.on("/format", pagina_format);                      //Encaminha para página de formata-
ção se requisição for igual a "/format"
  server.on("/delete", pagina_delete);                      //Encaminha para página de deleção
se requisição for igual a "/delete"
  server.on("/enviadados", pagina_enviadados);              //Encaminha para página envio de da-
dos via serial se requisição for igual a "/enviadados"
  server.on("/atuador1", pagina_atuador1);                  //Encaminha para página Atuador 1 se
requisição for igual a "/atuador1"
  server.on("/atuador2", pagina_atuador2);                  //Encaminha para página Atuador 2 se
requisição for igual a "/atuador2"
  server.on("/atuador3", pagina_atuador3);                  //Encaminha para página Atuador 3 se
requisição for igual a "/atuador3"
  server.on("/atuador4", pagina_atuador4);                  //Encaminha para página Atuador 4 se
requisição for igual a "/atuador4"
  server.on("/modo", pagina_alternar_modo);                 //Encaminha para página Alternar
Modo se requisição for igual a "/modo"
  server.on("/buzina", pagina_alternar_buzzer);             //Encaminha para página Alternar Bu-
zina se requisição for igual a "/buzina"
  server.on("/reinicia", pagina_reinicia);                  //Reinicia o microcontrolador se re-
quisição for igual a "/reinicia"
  server.begin();                                           //Inicializa o servidor HTTP

  //Desligando atuadores quando o dispositivo é ligado, assegurando desligá-las mesmo que haja
memória de porta
  digitalWrite(ATUADOR_1,LOW);          //Desliga atuador 1
  digitalWrite(ATUADOR_2,LOW);          //Desliga atuador 2
  digitalWrite(ATUADOR_3,LOW);          //Desliga atuador 3
  digitalWrite(ATUADOR_4,LOW);          //Desliga atuador 4

  Temp_Umid();                                              //Faz leitura e atualiza variável de
temperatura e umidade
  Luz();                                                    //Faz leitura e atualiza variável de lu-
minosidade
  Gas();                                                    //Faz leitura e atualiza variável de
presença de gás
  leitura_gas_anterior = leitura_gas;       //Atualiza variável de leitura de gás
}

//Laço de repetição
void loop() {

  //Escutando para respostas no protocolo HTTP
  server.handleClient();

  //Escutando para OTA
  ArduinoOTA.handle();
```

```
  //Aciona rotina de atualização dos controles de tempo definindo intervalo de leitura para
180 segundos
  tempo(180);

  //Aciona rotina para atualização do estado dos atuadores
  verif_atuadores();

  //Se estiver em modo teste, executa apenas a rotina de teste
  if (modo_teste == true) { Modo_Teste(); }
  //Se estiver em modo de calibração, executa apenas a rotina de calibração
  else if (calibracao == true) { Calib_Gas(); }
  //Rotinas de trabalho padrão do sistema
  else {
    if (tempo_passado_s == 0) {           //Executa se o tempo passado for 0 no intervalo da
função tempo
    conecta_WIFI();                       //Realiza uma tentativa de conexão Wi-Fi local caso
ainda não esteja conectado
    tempo_passado_s++;                    //Incrementa variável de controle do tempo passado
    Temp_Umid();                          //Faz leitura e atualiza variável de temperatura e
umidade
    Luz();                                //Faz leitura e atualiza variável de luminosidade
    Gas();                                //Faz leitura e atualiza variável de presença de gás
    Buzzer(1,15);                         //Emite pequeno sinal sonoro para saber que as lei-
turas estão sendo realizadas normalmente
    gravaLEITURAS();                      //Grava as leituras realizadas na memória

      if (automatico == true) {           //Executa apenas se o modo automático estiver ligado
      trata_amonia(1);                    //Verifica a presença de gás e realiza o tratamento
      trata_luz(2, 25, 18, 20, 20, 6);    //Verifica o nível de luz e realiza o tratamento
      trata_temp_fria(3, 24);             //Verifica a temperatura e realiza o tratamento
      trata_temp(4, 26);                  //Verifica a temperatura e realiza o tratamento
      }

    if (WiFi.status() == WL_CONNECTED) {
    MQTT_connect();                   //Conecta ao serviço de MQTT
    Transmite_MQTT();                 //Executa a rotina de envio e recepção de informações do
MQTT
    }

    server.close();                   //Fecha o servidor HTTP
    server.begin();                   //Reinicia o servidor HTTP
    }
    tempo_atuador(3, 60);   //Limita o tempo de acionamento da resistência para 1 minuto, para
não superaquecer
    tempo_atuador(4, 45);   //Limita o tempo de acionamento do ventilador para 45 segundos,
para não remover todo o ar quente
  }

  Serial.setTimeout(50);                          //Define tempo de escuta da porta serial
para entrada
  if (Serial.available() > 0) {                   //Se houver algo na entrada da porta se-
rial
    trata_serial(Serial.readStringUntil('\n'));   //Excuta rotina de tratamento da porta se-
rial com a informação de entrada
  }

  yield();

}   //Final do loop
```

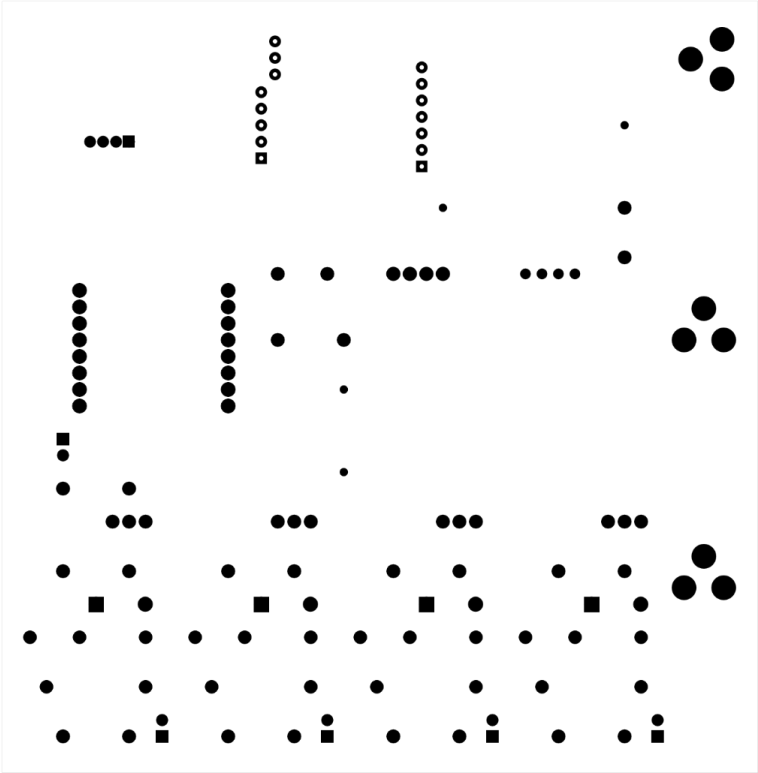
Apêndice B – Imagens para produção da placa eletrônica de circuito impresso



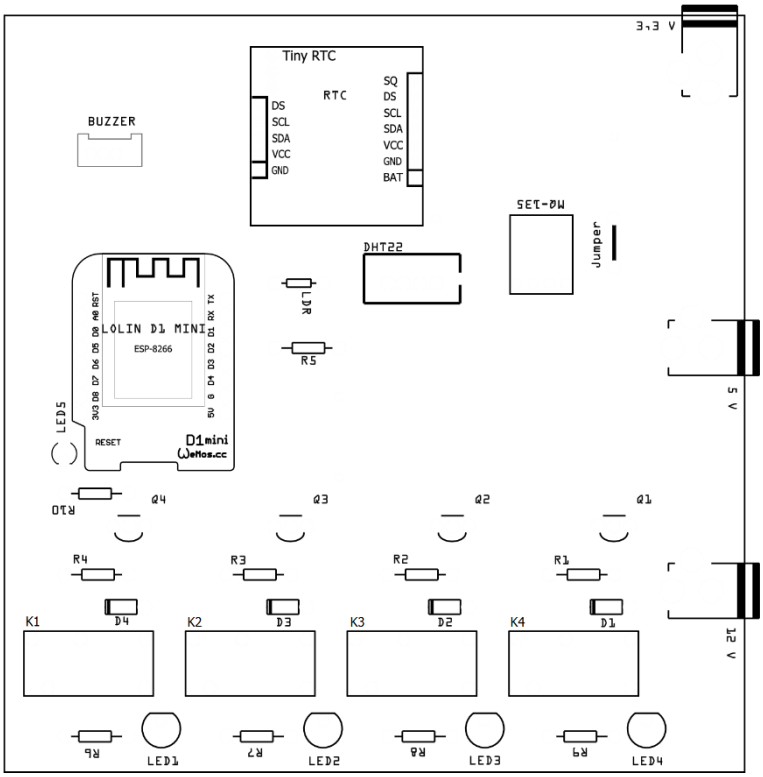
Trilhas de cobre inferiores - Fonte: Elaboração própria.



Trilhas de cobre superiores - Fonte: Elaboração própria.



Mapa para furação - Fonte: Elaboração própria.



Pintura do posicionamento de componentes - Fonte: Elaboração própria.

Apêndice C – Dados utilizados para a aplicação de mínimos quadrados no LDR

Dados fornecidos pelo fabricante do sensor LDR através do gráfico no *datasheet*.
Fonte: Adaptado de LIDA OPTICAL & ELETRONIC CO.

Ohm	Lux
2000	100
3000	70
4000	60
5000	40
6000	25
7000	20
8000	18
9000	17
10000	15
20000	5
30000	3
40000	2
50000	1,5
60000	1,3
100000	1

Apêndice D – Dados utilizados para a aplicação de mínimos quadrados no MQ-135

Dados fornecidos pelo fabricante do sensor MQ-135 através do gráfico no *datasheet*.
Fonte: Adaptado de WINSEN, 2015.

ppm	Rs/R0
10	0,7
20	0,52
30	0,44
40	0,39
50	0,36
60	0,34
70	0,32
80	0,29
90	0,28
100	0,27
200	0,19
300	0,16
400	0,14
500	0,13
600	0,12
700	0,11
800	0,099
900	0,094
1000	0,091