Universidade Federal de Ouro Preto - UFOP - Escola de Minas - Colegiado do curso de Engenharia de Controle e Automação - CECAU

Aline Maria Milagres

Sistema de Monitoramento de Parâmetros Ambientais para Biotérios: Fase II

Monografia de Graduação em Engenharia de Controle e Automação

Ouro Preto, 2022

Aline Maria Milagres

Sistema de Monitoramento de Parâmetros Ambientais para Biotérios: Fase II

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Alan Kardek Rêgo Segundo, Dr. Sc.

Ouro Preto, 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M637s Milagres, Aline Maria.
Sistema de monitoramento de parâmetros ambientais para biotérios [manuscrito]: fase II. / Aline Maria Milagres. - 2022.
50 f.: il.: color..

Orientador: Prof. Dr. Alan Kardek Rêgo Segundo.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. Biotérios. 2. Redes de computadores - Conexão. 3. Semicondutores - ESP32. 4. Sites da Web. 5. Banco de dados. I. Segundo, Alan Kardek Rêgo. II. Universidade Federal de Ouro Preto. III. Título.

CDU 658.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Aline Maria Milagres

Sistema de Monitoramento de Parâmetros Ambientais para Biotérios: Fase II

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 23 de agosto de 2022

Membros da banca

Dr. Alan Kardek Rêgo Segundo - Orientador (Universidade Federal de Ouro Preto)
Dra. Karla Boaventura Pimenta Palmieri (Universidade Federal de Ouro Preto)
Dra. Adrielle de Carvalho Santana (Universidade Federal de Ouro Preto)

Alan Kardek Rêgo Segundo, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 02/09/2022

Documento assinado eletronicamente por **Alan Kardek Rego Segundo**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 02/09/2022, às 09:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0386096** e o código CRC **A220668D**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Agradeço aos meus pais por sempre acreditarem em mim e sempre me incentivarem a buscar mais conhecimento apesar das dificuldades. Aos meus amigos por me acompanharem nessa jornada e pelo companheirismo mútuo durante o processo.

Agradeço ao meu orientador por acreditar em mim e no potencial desse projeto, e também à UFOP por me proporcionar tantas oportunidades que foram cruciais para meu desenvolvimento acadêmico e pessoal.

Por último mas não menos importante, agradeço aos meus amigos Gabriel, Samuel e Julia, que além de bons amigos no dia a dia, foram cruciais para que este projeto pudesse ser finalizado com excelência.

"O sucesso tem sido sempre um grande mentiroso." (Friedrich Nietzsche)

Resumo

Um biotério é um local com ambiente controlado onde são armazenados animais utilizados em experimentos científicos ou pesquisas. Na fase I do projeto, foi construído um sistema de aquisição de dados para um biotério feito por um módulo transmissor e receptor de baixo custo e plataforma aberta, e um sistema supervisório para acompanhamento de dados. Dando sequência à pesquisa, este trabalho consiste em uma adaptação do sistema com base no módulo *Wi-Fi* ESP32, para tornar possível o monitoramento em tempo real das variáveis por meio de qualquer dispositivo com acesso à internet. Essas variáveis são: ruído, luminosidade, temperatura, umidade e amônia. O monitoramento constante dos biotérios é extremamente necessário pois é crucial que o ambiente esteja confortável e seguro para que os animais presentes nele não sofram com estresse, além de manter a segurança dos funcionários que ali trabalham. Além de adaptação do tipo de conexão, uma aplicação *Web* foi criada para acesso remoto aos dados, contendo uma página *Web* comunicando com um banco de dados.

Palavras-chaves: Biotério. Conexão. ESP32. Página *Web*. Banco de Dados.

Abstract

A vivarium is a place with a controlled environment where animals used in scientific experiments and research are stored. On the first phase of this project, it was built a data acquisition system for a vivarium, consisting on a low-cost and open source transmitter and receiver module, and a supervisory system for data tracking. Following this up, this current project is an adaptation based on the ESP32 Wi-Fi module, which will monitor variables in real time from any device connected to internet. These variables are: noise, luminosity, temperature, humidity and ammonia. A constant monitoring of a vivarium is extremely necessary because it is essential that the environment is comfortable and safe so that the animals present there do not suffer from stress, in addition to keeping the workers safe. Besides the adaptation of the connection type, it was created a Web application to remote data access, with a Web Page communicating with a data base.

Key-words: Vivarium. Connection. ESP32. Web Page. Data Base.

Lista de ilustrações

Lista de abreviaturas e siglas

ESP32	ESP-WROOM-32 módulo *Wi-Fi*
dB	Decibéis
IDE	Integrated Development Environment
GPIO	General Purpose Input/Output
ADC	Analog to Digital Converter
LDR	Light Dependent Resistor
HTML	Hipertext Markup Language
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
SQL	Standard Query Language

Sumário

1 Introdução

As instalações controladas chamadas de Biotério são ambientes capazes de produzir e manter animais destinados a servir como reagentes biológicos em pesquisas, ensino, controle de qualidades etc (CARDOSO, 2001). Os Biotérios precisam sempre estar em conformidade com as regras de biossegurança definidas para o tipo de função realizada no mesmo.

As condições de ambiente e alojamento dos biotérios precisam ser minuciosamente controlados para garantir o bem estar e a saúde dos animais que serão utilizados nas pesquisas. Ter um ambiente controlado é crucial para que não haja interferências nos resultados, e também para que o bem estar dos profissionais que ali trabalham seja garantido. No estudo feito por Santos et al. (2010), foi constatada a importância do estabelecimento, por parte do laboratório ou biotéro, de um conjunto de valores de referência biológicos dos animais. Tendo isso em mente, faz-se necessário a utilização de um sistema que consiga monitorar em tempo real todas as condições ambientais que são cruciais para que esse controle seja alcançado, seguindo normas e padrões preestabelecidos.

O Centro de Ciência Animal da UFOP, local de estudo deste trabalho, possui um biotério destinado a pesquisas com roedores. As principais variáveis a serem analisadas, neste caso, são: a temperatura, o ruído, a taxa de amônia, a umidade e a luminosidade. Na fase I do projeto, foi construído dois módulos para cada sala do biotério, sendo um emissor e um receptor. O módulo emissor lê os parâmetros ambientais e transmite esses dados para o módulo receptor, o qual está conectado a um computador no escritório do biotério, onde se encontra instalado o sistema supervisório (LISBOA et al., 2019). Como continuação, a ideia principal deste trabalho final de curso é adaptar o sistema para que a comunicação seja feita via *Wi-Fi* por meio do módulo ESP32, e desenvolver uma página *Web* para que o monitoramento seja feito por meio de qualquer aparelho com conexão à internet.

1.1 Objetivos gerais

Este trabalho tem como objetivo principal realizar a conexão com a rede local para permitir o monitoramento das variáveis temperatura, ruído, umidade, amônia e luminosidade no biotério da UFOP de forma remota e por meio de qualquer dispositivo conectado à internet. Além disso, pretende-se criar uma página *Web* com interface intuitiva para acompanhamento dos dados. Os dados obtidos por meio dos sensores são armazenados

em um banco de dados no servidor da UFOP e podem ser acessados por meio de uma página de histórico no *Website*. Entretanto, os dados lidos pelos sensores não foram tratados neste trabalho, visto que isso já foi feito em trabalhos anteriores.

1.1.1 Objetivos específicos

- Utilizar o ESP-WROOM-32 para fazer a comunicação entre os sensores e a página *Web*;
- Desenvolver a interface *front-end* do site;
- Desenvolver a interface *back-end* do site;
- Desenvolver a comunicação entre a página *Web* e o banco de dados da UFOP para que os dados sejam armazenados de forma correta.

1.2 Justificativa do trabalho

Atualmente, o sistema apresenta as leituras das variáveis ambientais (temperatura, umidade, amônia, luminosidade e ruído) em um programa supervisório desenvolvido no *Visual Studio* em forma de gráficos dinâmicos e tabelas. Isso permite que os responsáveis pelo laboratório façam a sua supervisão de uma maneira clara e intuitiva, permitindo rápidas correções de parâmetros que não se adequam à norma de segurança para biotérios.

Na etapa anterior, o objetivo principal foi fazer correções e calibrações pendentes nos sensores e no sistema como um todo, assim como correção de erros de comunicação entre o sistema receptor e o sistema emissor, e também dar continuidade à aplicação *Web* para compartilhamento de dados do sistema supervisório. Este projeto está em constante desenvolvimento e aprimoramento e, apesar de ainda existirem falhas de comunicação devido ao alcance do sinal de transmissão, os resultados da etapa anterior se mostraram eficientes quanto à comunicação entre o emissor e o receptor, e a aplicação *Web* criada atendeu a todos os requisitos necessários.

Na etapa atual, viu-se a necessidade de fazer uma adaptação no sistema para que ele não precise mais realizar a comunicação entre um receptor e um emissor. Isso faz com que o sistema seja resumido a apenas um circuito contendo os sensores necessário para a leitura das variáveis ambientais e um módulo *Wi-Fi* ESP32 para fazer a transmissão de dados via internet.

Como trata-se de um ambiente controlado, uma alteração em algum parâmertro pode ser prejudicial não só para os resultados dos estudos, mas tabém para os animais ali presentes e para os trabalhadores responsáveis pelo laboratório. Se uma variável extrapolar os valores preestabelecidos pela norma durante um certo período de tempo, é imprescindível

que ela seja corrigida à tempo a fim de evitar tais problemas. Então, é essencial que seja possível acessar esses dados em tempo real por meio de dispositivos que não estejam presentes apenas no laboratório.

Portanto, a criação de uma página *Web* onde é possível acessar estes dados em tempo real e o histórico dos mesmos em qualquer dispositivo conectado à internet é de extrema importância para ampliar a versatilidade e acessibilidade, melhorando ainda mais a interação do usuário com o sistema.

1.3 Estrutura do trabalho

Este trabalho está estruturado da seguinte maneira:

- No Capítulo 2, é apresentada uma revisão bibliográfica a fim de compreender melhor a aplicação do ESP32 em sistemas de monitoramento e em outros projetos semelhantes;
- No Capítulo 3, são descritas as etapas de desenvolvimento deste trabalho. São apresentadas as adaptações feitas em relação ao projeto anterior, como a troca da comunicação via rádio para a comunicação via *Wi-Fi*, usando uma *protoboard* auxiliar. Além disso, é apresentado o desenvolvimento *Web* (*front-end* e *back-end*), onde são mostradas as etapas para a construção da página de *login*, a página de exibição dos dados em tempo real e a página de histórico. Na sequência, são mostradas as etapas de realização dos testes de conexão do ESP32 com a página *Web* e com os sensores, e também a conexão com o banco de dados;
- No Capítulo 4, são apresentados os resultados pertinentes ao capítulo citado anteriormente, contendo a aplicação *Web* pronta, bem como os resultados das conexões do ESP32 e dos testes nos sensores;
- No Capítulo 5, são apresentadas as conclusões feitas por parte da discente perante o desenvolvimento deste projeto de conclusão de curso, além de sugestões para trabalhos futuros, tendo em vista a melhoria contínua do mesmo.

2 Revisão Bibliográfica

Para a realização desse projeto, uma série de pesquisas foram feitas a fim de compreender melhor a aplicação dos dispositivos da familia ESP em sistemas de monitoramento em diversos projetos diferentes. Nesta seção, são apresentados dois trabalhos feitos usando o ESP8266, e um trabalho usando o ESP32.

2.1 Sistema IoT para Monitoramento de Temperatura e Umidade Ambientes e Acionamento Remoto de Cargas

No artigo elaborado por Lima, Alves e Juca (2018), é apresentado um sistema de propósito geral e de baixo custo para monitoramento de temperatura e umidade e acionamento de carga de forma remota, que foca no gerenciamento de ambientes domésticos. Assim como a ideia do projeto do Biotério, a aplicação desse trabalho consiste no uso do módulo *Wi-Fi* e da plataforma de desenvolvimento do Arduino, além da utilização de um banco de dados para armazenar as informações de temperatura e umidade captadas pelo sensor DHT11 e o estado de uma carga (representada por um LED).

Com o *firmware* da placa ESP8266 NodeMCU adicionado, foi montado um circuito em uma *protoboard*, fazendo a conexão entre o módulo, o DHT11 e o LED. Com o sistema todo montado, os dados obtidos pelo sensor e o estado do LED foram armazenados em um banco de dados, e os mesmos foram exibidos na aplicação *Web* desenvolvida. Algo interessante implementado pelos autores foi a adição de um botão na página *Web* para a realização da sincronização da placa NodeMCU com o banco de dados.

Com este trabalho, conclui-se que a utilização de um sistema de propósito geral e de baixo custo, combinado com as plataformas de banco de dados disponíveis, oferecem recursos robustos para aplicações que podem ser consideradas complexas, caso sejam limitadas a componentes sem acesso à internet. Isso mostra a eficácia de projetos desse tipo, onde seu desenvolvimento se mostra intuitivo, rápido e com ótimos resultados.

Pode-se perceber, então, a importância da adaptação de sistemas de monitoramento para que seja possível o acesso a dados por meio de qualquer dispositivo, sem ter a necessidade da criação de um sistema supervisório em cada.

2.2 Monitagro – Monitoramento de Atividades Agropecuárias Utilizando Sensores e Internet das Coisas

Nesse artigo desenvolvido por Beirão et al. (2019), é apresentado um sistema de monitoramento de fatores naturais que influenciam um cultivo agrícola de qualidade, tais como luminosidade, temperatura, umidade, recursos hídricos do solo, umidade do ar, pressão atmosférica e minerais, em que as variáveis são capturadas e monitoradas por meio de uma aplicação *Web*.

De maneira análoga à anterior, foi realizada toda a conexão com a ajuda de uma *protoboard*, onde o módulo ESP8266 recebe e envia os dados lidos pelos sensores, e uma placa *Raspberry* recebe esses valores para realizar o processamento e armazenamento dos dados. Após fazer uma comparação entre os dados recebidos e os dados gravados para que não haja tráfego de informações desnecessárias, ela faz a transmissão desses dados para a nuvem por meio de requisições *HTTP*, armazenando os mesmos em um banco de dados.

Os resultados dos testes se mostraram promissores. Foi possível realizar uma leitura consistente dos dados e também fazer uma análise minuciosa dos mesmos para se obter melhores condições de cultivo.

Analisando esse trabalho, foi possível compreender melhor como funciona a aplicação do módulo ESP8266 NodeMCU em sistemas práticos, e ficou mais claro como as placas da família ESP podem facilitar os trabalhos de monitoramento em diversas áreas.

2.3 Proposta de automação de baixo custo para aviários utilizando o microcontrolador ESP32

Na proposta desenvolvida por Leite et al. (2020), é descrito o processo de desenvolvimento de um protótipo de aviário automatizado utilizando o ESP32, cujo objetivo principal é ajudar proprietários de aviários a diminuir a mortalidade de aves. Esse objetivo se assemelha ao projeto descrito nesse trabalho, cujo um dos propósitos é impedir a mortalidade precoce dos animais presentes nos laboratórios.

O projeto contou com testes em *protoboard* e montagem esquemática em *softwares* de simulação, onde foi possível realizar os testes dos circuitos de iluminação, umidificação e refrigeração. Além do monitoramento ambiental, algo interessante implelentado por Leite et al. (2020) foi a adição de *coolers* para resfriamento do ambiente de acordo com uma temperatura máxima definida previamente.

Após a montagem do circuito de teste e programação do mesmo, foi realizada a conexão do ESP32 à rede para que os dados pudessem ser acessados e armazenados. Com isso, foi montado o protótipo físico, o qual mostrou resultados satisfatórios quanto ao

controle das variáveis iluminação, umidificação e temperatura, além de mostrar a eficácia de se utilizar um microcontrolador de baixo custo, como o ESP32, para realizar conexões em redes locais para coleta e envio de dados.

3 Desenvolvimento

Durante o desenvolvimento deste trabalho, foram seguidas algumas etapas cruciais para a montagem e configuração dos parâmetros necessários para uma boa leitura e envio de dados.

3.1 Desenvolvimento *Web* (*front-end* e *back-end*)

Para a realização do desenvolvimento da parte *Web*, foi imprescindível o entendimento dos conceitos de *front-end* e *back-end*. O desenvolvimento do *front-end* faz com que a interface apresentada para o usuário seja intuitiva e entregue uma boa experiência para quem está acessando sua plataforma, como botões, menus e gráficos, permitindo que o usuário interaja com o sistema, realizando o envio de informações obtidas delas para o servidor (SOUZA et al., 2019). As linguagens utilizadas para seu desenvolvimento são o *HTML* (linguagem de marcação), *CSS* (linguagem de estilo) e o *JavaScript* (linguagem de *script*), entre outros.

Já o desenvolvimento do *back-end*, como o nome já diz, não interfere na parte visual da aplicação, e sim no que ocorre por trás do que é mostrado para o usuário, ou seja, o *back-end* consiste no servidor que é responsável pelo processamento de dados enviados pelo *front-end*, bem como seu armazenamento e retorno de dados (SOUZA et al., 2019). As linguagens utilizadas para seu desenvolvimento são o *PHP*, C, *Java*, *Ruby*, entre outras.

Durante o desenvolvimento deste projeto, as linguagens escolhidas para o *front-end* foram o *HTML*, o *CSS* e o *JavaScript* (na parte dos gráficos), e para o *back-end* foi utilizado o *PHP* e *JavaScript*. Além dessas linguagens, também foi utilizado o servidor do *MySql* como banco de dados.

3.1.1 *HTML*

A linguagem *HTML* (*Hipertext Markup Language*, ou seja, linguagem para marcação de hipertexto) é uma linguagem usada para criar páginas *Web* utilizando *tags* e atributos onde serão definidos o conteúdo mostrado para o usuário que está acessando o site. Segundo Silva (2008), essa marcação de hipertexto consiste em todo o conteúdo inserido em um documento para *Web* com a característica de interligar outros documentos da *Web*.

3.1.2 *CSS*

O *CSS* (*Cascading Style Sheets*, ou seja, folhas de estilo em cascata), como o próprio nome diz, é uma linguagem de estilo usada para descrever a apresentação de uma página *Web* escrita em *HTML*, dando movimentos e interações diversas. A folha de estilo em cascata (*CSS*) é um mecanismo simples para adicionar estilos, como fonte, cores, espaçamentos etc. (SILVA, 2007).

3.1.3 *PHP*

A linguagem de programação *PHP* (*Hypertext Preprocessor*, ou seja, preprocessador de hipertexto) é usada para a comunicação com o servidor (*back-end*). Ela é utilizada para criação de *scripts* do lado do servidor e é praticamente invisível para o usuário final (CONVERSE; PARK, 2003). O uso do *PHP* facilita a criação de sites dinâmicos com agilidade de execução, permitindo o gerenciamento de arquivos presentes no servidor, a coleta de dados em um formulário, modificar base de dados etc.

3.2 O módulo *Wi-Fi* ESP32

O módulo *Wi-Fi* ESP-WROOM-32 utilizado neste projeto é uma plataforma de baixo custo e *open source* da família do ESP32 que oferece diversas aplicações, seja para projetos simples com alguns sensores, seja para projetos mais complexos com uma demanda de processamento maior, segundo Systems (2016). Com esse módulo, é possível realizar uma integração *Bluethooth* e *Wi-Fi*, porém, neste trabalho, o foco será apenas no *Wi-Fi*, em que é possível realizar uma grande variedade de conexões para atender os requisitos do projeto que foram previamente estabelecidos. O módulo ESP-WROOM-32 é apresentado na figura 1.

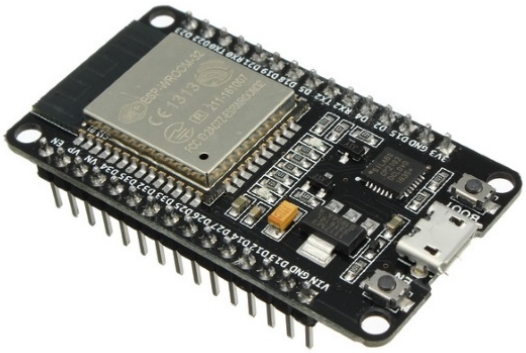

Figura 1 – ESP-WROOM-32: Módulo Wi-Fi ESP-WROOM-32.
Fonte: Filipeflop (2022).

3.3 Pinagem e programação do ESP32

Os pinos digitais do módulo ESP-WROOM-32 recebem a sigla *GPIO*, que representa o conjunto de pinos utilizadas para comunicação das entradas e saídas digitais. Já os pinos analógicos recebem a sigla *ADC*, e eles são o conjunto de pinos responsáveis pelas entradas analógicas dos sensores. Ambos pinos são de extrema importância para a conclusão deste trabalho, visto que os sensores que serão utilizados necessitam das duas entradas para leitura correta de seus dados. Na figura 2 é possível observar a relação dos pinos na placa do ESP32, o que permite escolher de forma correta qual pino conectar em qual sensor, dependendo de sua necessidade.

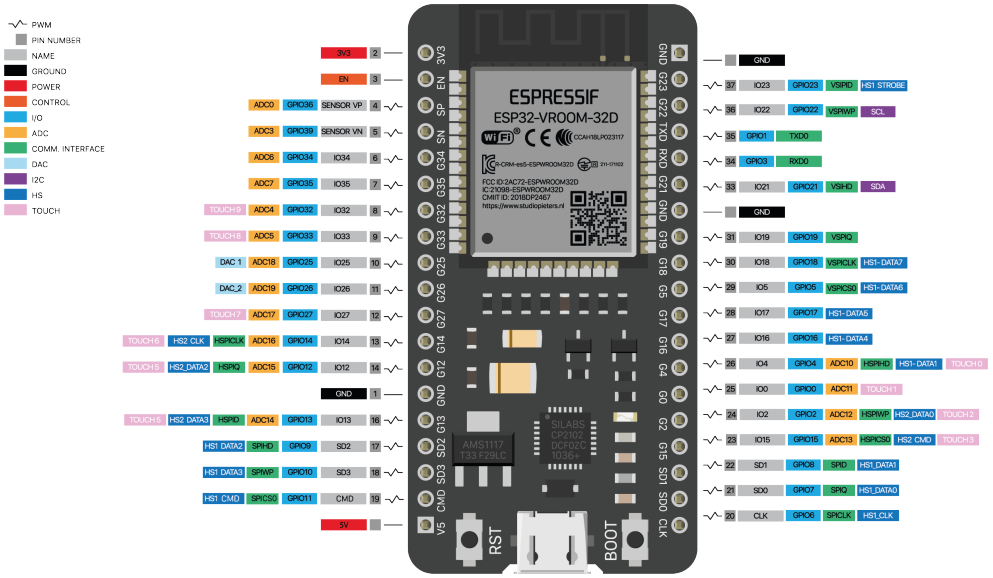

Figura 2 – ESP-WROOM-32: Pinagem.
Fonte: Pieters (2022).

A linguagem de programação utilizada pelo módulo ESP-WROOM-32 é a mesma utilizada pela *IDE* do Arduino, ou seja, uma junção das linguagens C e C++. Essa é uma das maiores vantagens em se utilizar esse módulo para adaptação de projetos feitos no Arduino por não haver a necessidade de se aprender mais uma linguagem de programação para usá-lo.

3.4 O Sistema de monitoramento e adaptações necessárias

No sistema projetado por Lisboa et al. (2019), foram definidas faixas de valores para cada parâmetro ambiental, de acordo com pesquisas. Também foram feitas calibrações nos sensores de ruído, luminosidade, temperatura e umidade, e amônia.

Para calibração do sensor de ruído, foi utilizado um amplificador operacional, onde o sinal de saída do microfine foi ampliado 237 vezes. Após testes do sistema eletrônico do sensor de ruído com o amplificador operacional, foi feita a conversão do valor lido, utilizando um decibelímetro como referência para sua calibração. A figura 3 mostra a curva de calibração para este sensor.

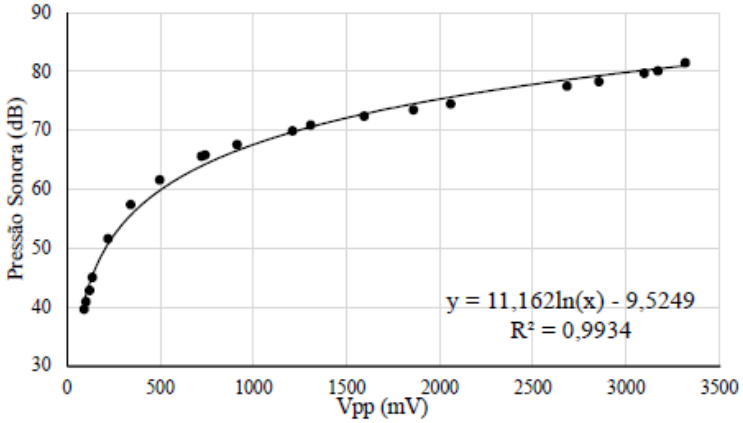

Figura 3 – Curva de calibração do sensor de intensidade sonora.
Fonte: Lisboa et al. (2019)

Na calibração do sensor de luminosidade, foi utilizado um luxímero posicionado ao lado dele com o mesmo angulo sobre a fonte de luz, e foram comparados os valores lidos pelo luxímero em relação à resistência do sensor. A figura 4 mostra a sua curva de calibração. Com isso, foi calculado o valor da luminância para que o valor final fosse mostrado em *lux* a partir da resistência do sensor de luminosidade.

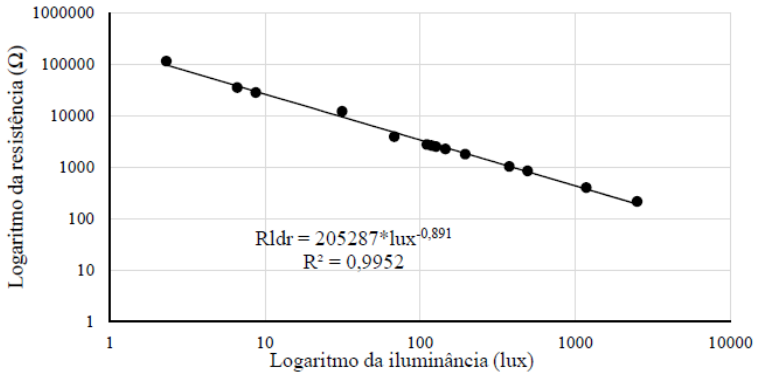

Figura 4 – Curva de calibração do sensor de luminosidade.
Fonte: Lisboa et al. (2019)

Para o sensor de umidade e temperatuda, não foi necessário realizar sua calibração, visto que o mesmo já vem calibrado de fábrica. Foi necessário, apenas, utilizar a biblioteca "DHT.h"disponível na IDE do Arduino.

A calibração do sensor de amônia foi feita utilizando um exaustor, onde foram colocados três emissores na mesma altura que um sensor de detecção digital de amônia, que foi utilizado como referência. Foi injetado com uma seringa uma quantidade gradual de amônia no aquário pra realização da calibração, cuja curva é mostrada na figura 5.

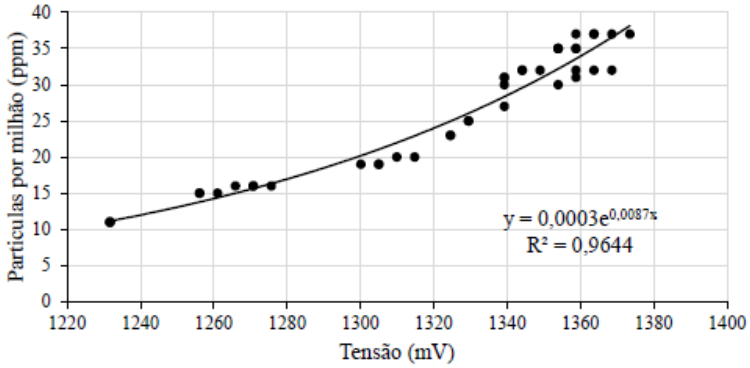

Figura 5 – Curva de calibração do MQ-135
Fonte: Lisboa et al. (2019)

Além dos testes nos sensores, foi feita a transmissão via rádio do módulo NRF24101 com amplificador e antena externa, além de um sistema supervisório para acompanhamento dos dados. O sistema possui um módulo emissor para cada sala do biotério e um módulo receptor, os quais são mostrados nas figuras 6 e 7, respectivamente. Na etapa atual, foi substituída essa transmissão via rádio por uma via *Wi-Fi*.

Figura 6 – Emissor.
Fonte: Elaborado pelo autor.

Figura 7 – Receptor.
Fonte: Elaborado pelo autor.

Após a revisão de literatura e avaliação da disponibilidade dos dispositivos pelo laboratório da UFOP, foi iniciada a montagem e os testes usando uma *protoboard* auxiliar. Os materiais utilizados nessa etapa foram:

- ESP-WROOM-32 (módulo *Wi-Fi*);
- LDR (sensor de luminosidade);
- DHT11 (sensor de temperatura e umidade);
- MQ135 (sensor de gás);
- KY-038 (sensor de ruído);
- LEDs;
- *Jumpers* Macho/Fêmea.

A montagem para testes nos sensores é mostrada na figura 8.

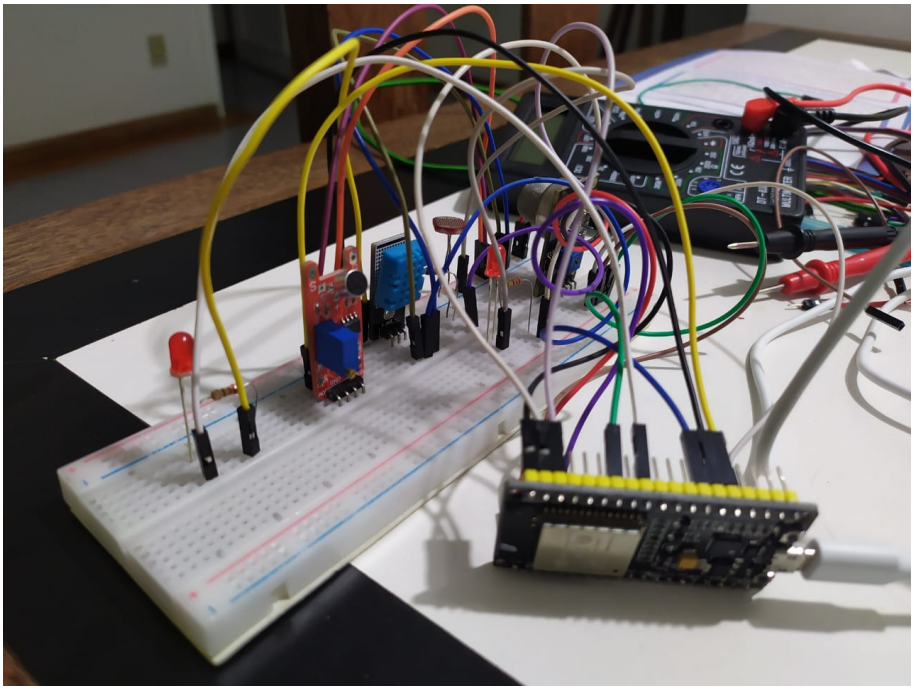

Figura 8 – Montagem do circuito na protoboard.
Fonte: Elaborado pelo autor.

A seguir, na figura 9, é possível observar a montagem do circuito no *software* Fritzing [1], onde são mostradas de forma mais clara as conexões realizadas entre o ESP32 e os sensores, para que as leituras e envio de dados pudessem ser feitos de forma correta. Os LEDs representam alarmes que podem ser ativados se alguma das variáveis saírem da faixa recomendada.

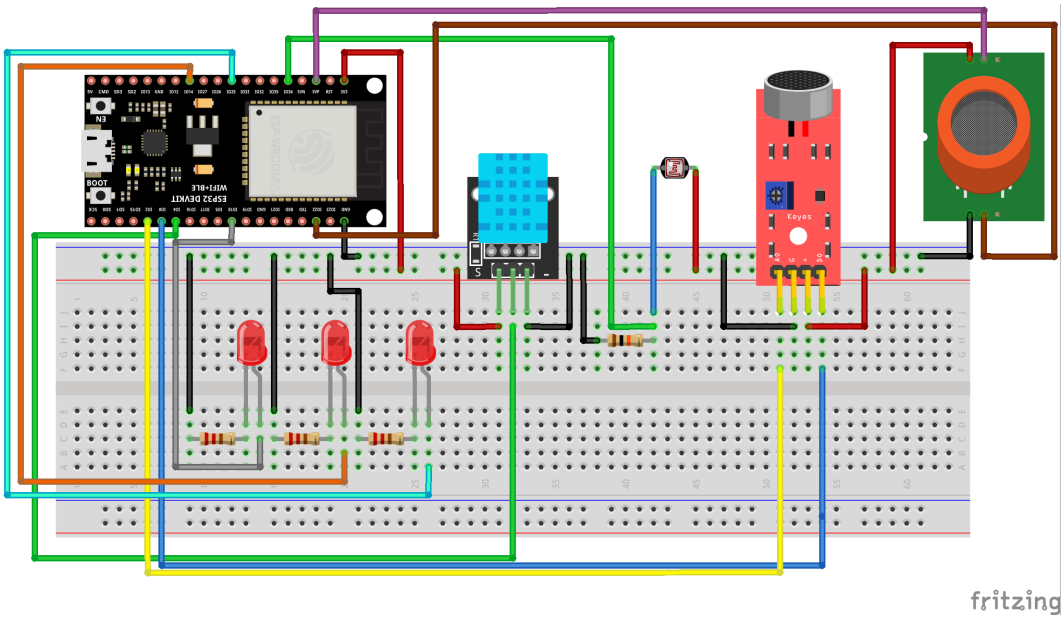

Figura 9 – Esquemático do circuito.
Fonte: Elaborado pelo autor.

[1] Disponível em: Fritzing.

3.5 Bibliotecas utilizadas

As bibliotecas utilizadas neste trabalho foram as seguintes:

- WiFi.h [2]: essa biblioteca permite a conexão do módulo ESP32 à rede *Wi-Fi*;
- WebServer.h[3]: essa biblioteca ajuda na configuração do servidor e nas solicitações *HTTP* recebidas;
- HTTPClient.h[4]: é essa biblioteca que faz as requisições *HTTP* no ESP32;
- DHT.h[5]: essa biblioteca permite o uso do sensor de temperatura e umidade na plataforma do Arduino;
- Adafruit-Sensor.h[6]: essa biblioteca é exigida por sensores como o DHT11.

3.6 Teste nos sensores

Após a montagem na *protoboard*, foram selecionadas as portas respectivas às conexões nos pinos do ESP32 para realizar as leituras. Para o sensor de amônia, foi usado um código auxiliar para calibração, como mostra o anexo A (HOSSEINI, 2019), e o sensor de ruído foi calibrado manualmente. Como o objetivo pricipal deste trabalho é a aplicação *Web*, os valores lidos pelos sensores não foram tratados, conforme foi apresentado na etapa feita por Lisboa et al. (2019). Foram apenas realizadas calibrações afim de entender melhor o funcionamento dos mesmos. A seguir, são apresentados os testes para os sensores de temperatura e umidade e para o sensor de luminosidade.

A figura 10 mostra o teste do LDR com a luz do ambiente acesa, onde o LED encontra-se desligado, como esperado.

[2] Disponível em: Wi-Fi.h
[3] Disponível em: WebServer.h
[4] Disponível em: HTTPClient.h
[5] Disponível em: DHT.h
[6] Disponível em: Adafruit-Sensor.h

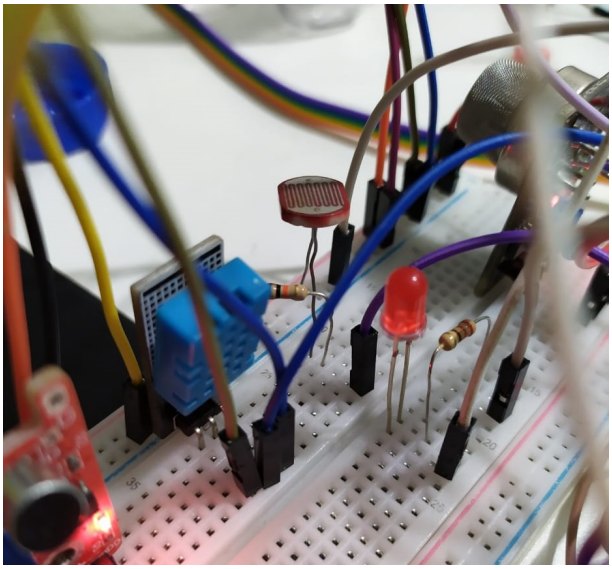

Figura 10 – Teste LDR com a luz acesa.
Fonte: Elaborado pelo autor.

Em seguida, com a luz desligada, o LED encontra-se ligado, como mostra a figura 11.

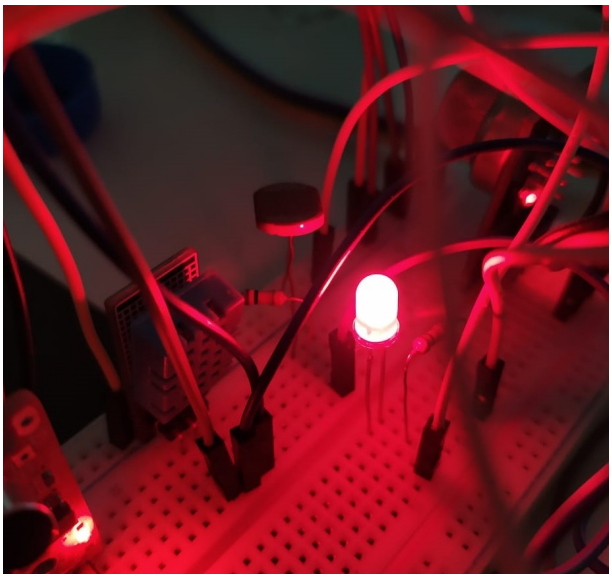

Figura 11 – Teste LDR com a luz apagada.
Fonte: Elaborado pelo autor.

Com isso, ao conectar o ESP32 ao servidor local, é possível testar a sua conexão com a rede e mostrar os dados do LDR e do DHT11 em uma página *Web* de teste.

As figuras a seguir mostram os testes feitos com os sensores DHT11 e LDR já devidamente conectados ao ESP32. A figura 12 mostra o teste com a luz acesa, e a figura 13 mostra o teste com a luz apagada. Como se trata apenas do teste da conexão e nos sensores, o valor da resistência do LDR lido e exibido pelo ESP32 não foi convertido.

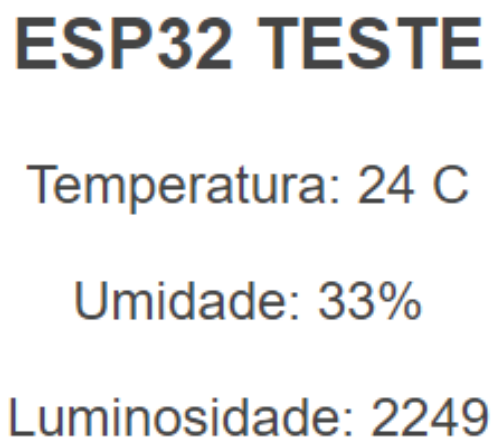

Figura 12 – Página *Web* teste com a luz acesa.
Fonte: Elaborado pelo autor.

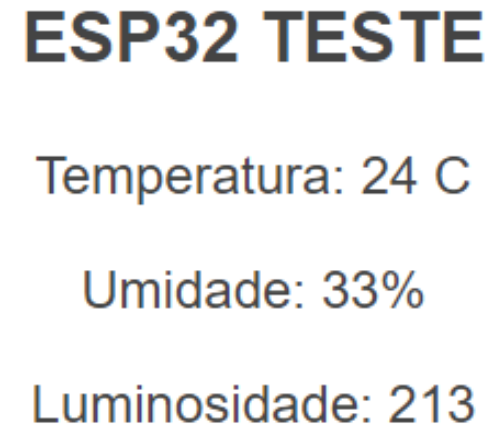

Figura 13 – Página *Web* teste com a luz apagada.
Fonte: Elaborado pelo autor.

3.6.1 Calibração MQ135

Segundo Hosseini (2019), o sensor MQ135 é um sensor de gás que é capaz de detectar vários tipos de gases, podendo identificar mais de um tipo de uma vez. Para este trabalho, ele é utilizado para detectar a presença de amônia no ambiente do biotério. Esse sensor consegue medir a presença de gases no ambiente realizando uma reação química em seus eletrodos quentes e medindo a corrente resultante (HOSSEINI, 2019).

Devido à necessidade de esquentar os eletrodos, foi preciso deixá-lo conectado à fonte de corrente por um período de 24 horas para realizar a sua calibração.

Como mencionado anteriormente, foi utilizado o código no anexo A para fazer a calibração do sensor. Essa calibração é baseada na taxa de resistência, que inclui a

resistência do sensor em uma concentração de gás (R0) calculada, e a resistência interna do sensor que muda de acordo com a concentração do gás no ar (Rs) (HOSSEINI, 2019).

Na figura 14 é possível observar a tensão no sensor e a concentração de gás obtida, a qual foi utilizada no código de leitura da taxa de amônia no ambiente.

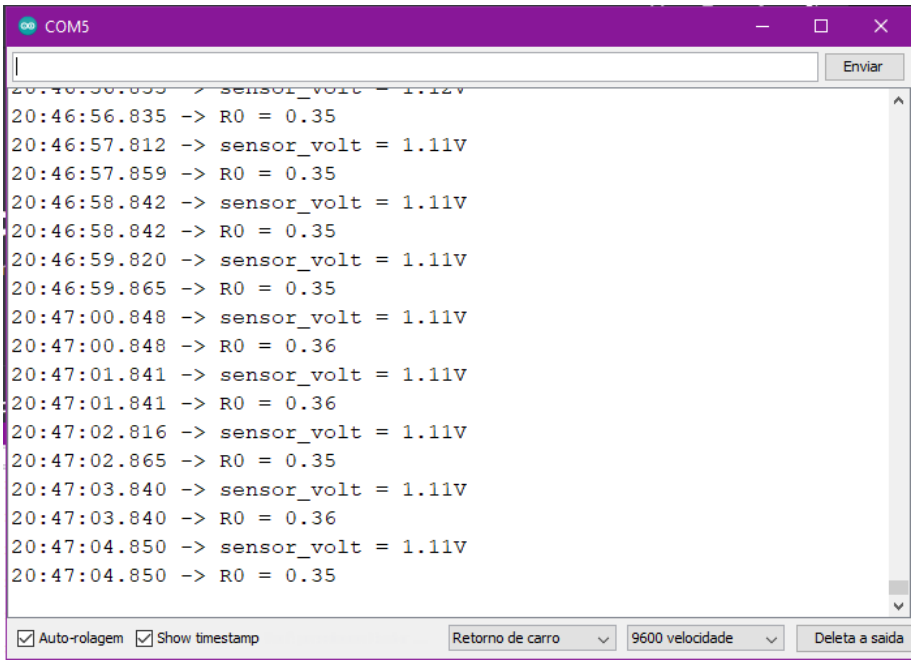

Figura 14 – Calibração do MQ135.
Fonte: Elaborado pelo autor.

3.6.2 Calibração KY-038

O sensor de ruído KY-038 possui um potenciômetro embutido em seu sistema para a calibração da sensibilidade do microfone. Para que o microcontrolador perceba a presença de som, a tensão é variada de acordo com a intensidade, ficando menor à medida que o som aumenta (MORAIS, 2018).

Segundo David (2021), a calibração funciona da seguinte forma: inicialmente, gira-se o potenciômetro no sentido anti-horário até que o LED localizado abaixo dele se apague, conforme mostra a figura 15; quando o *LED* se apaga, gira-se o potenciômetro no sentido horário até o *LED* começar a acender, sem que ele acenda totalmente. O que fará ele acender com sua força total é o ruído no ambiente, então, quando o *LED* acender com o barulho, o sensor estará calibrado, como pode-se observar na figura 16.

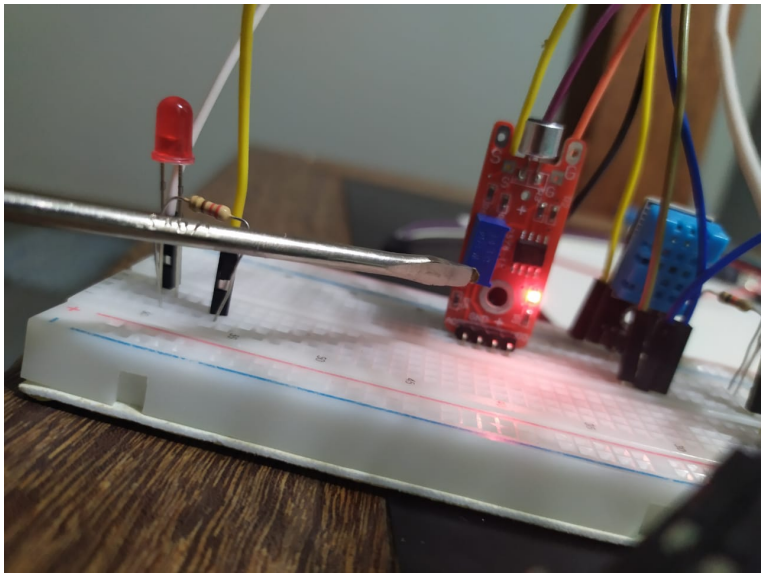

Figura 15 – Calibração KY-038.
Fonte: Elaborado pelo autor.

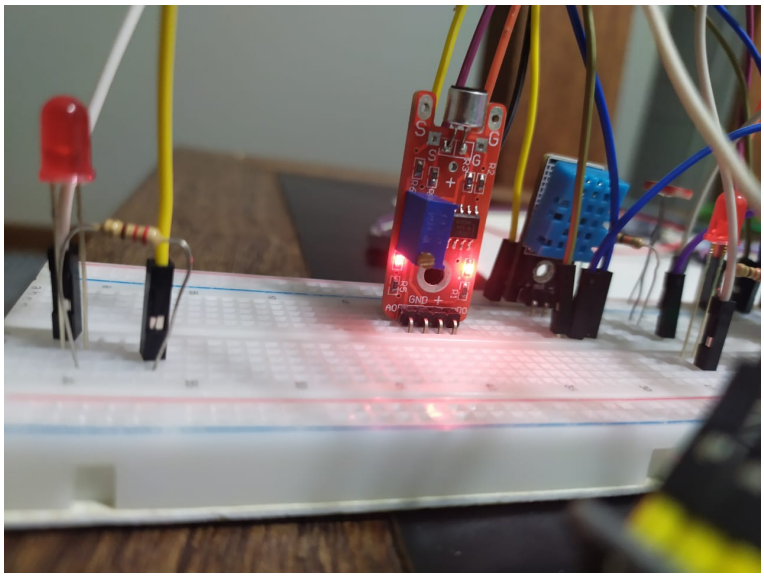

Figura 16 – KY-038 calibrado.
Fonte: Elaborado pelo autor.

Analisando o *plotter serial* da *IDE* do Arduíno, pode-se perceber claramente a presença e a ausência de som durante os testes de funcionamento do sensor KY-038. São visíveis na figura 17 os ruídos discrepantes ao longo do tempo que representam a presença de som no qual, nesse teste em específico, se tratavam de batidas de palmas.

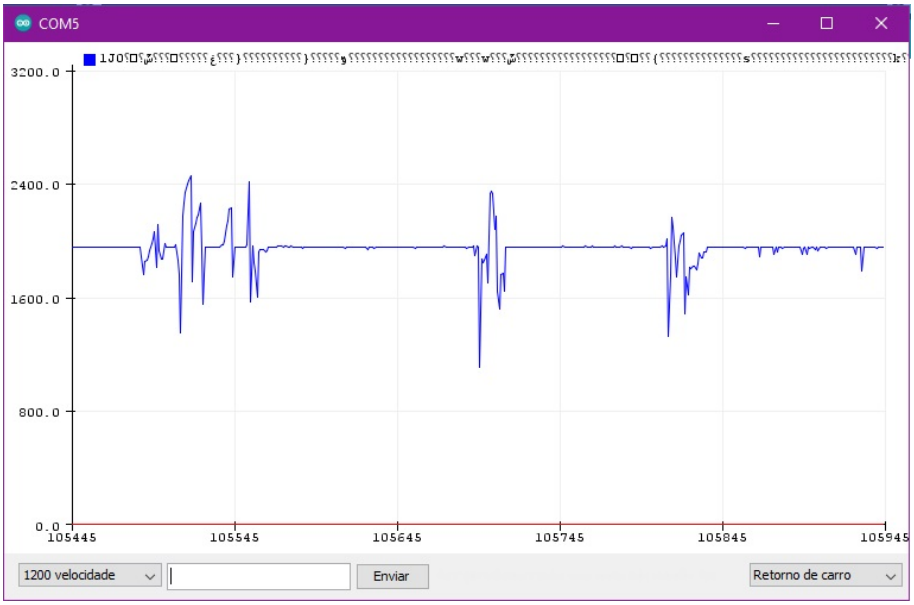

Figura 17 – Teste KY-038.
Fonte: Elaborado pelo autor.

No monitor *serial* da figura 18, é possível verificar o valor lido analogicamente pelo sensor para identificar a itensidade do som presente no ambiente. Na figura 19, pode-se identificar a presença de som abaixo de um valor analógico predeterminado.

Uma futura avaliação deverá ser feita junto aos técnicos do laboratório para avaliar a melhor posição em que os sensores devem ficar no biotério. A distância entre o dispositivo que possui os sensores e os animais vai influenciar nos resultados, então uma análise mais profunda deve ser feita quando o protótipo físico estiver pronto.

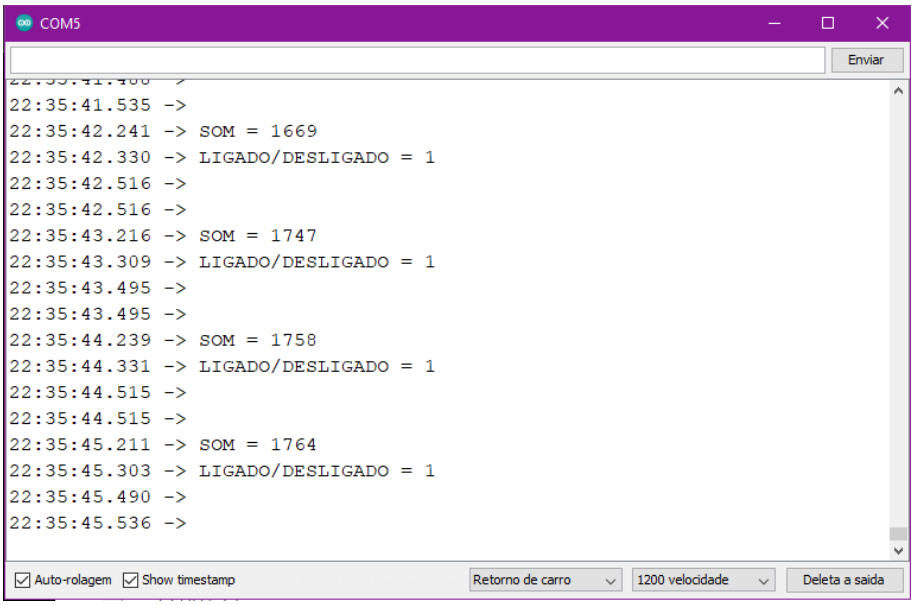

Figura 18 – Som ligado.
Fonte: Elaborado pelo autor.

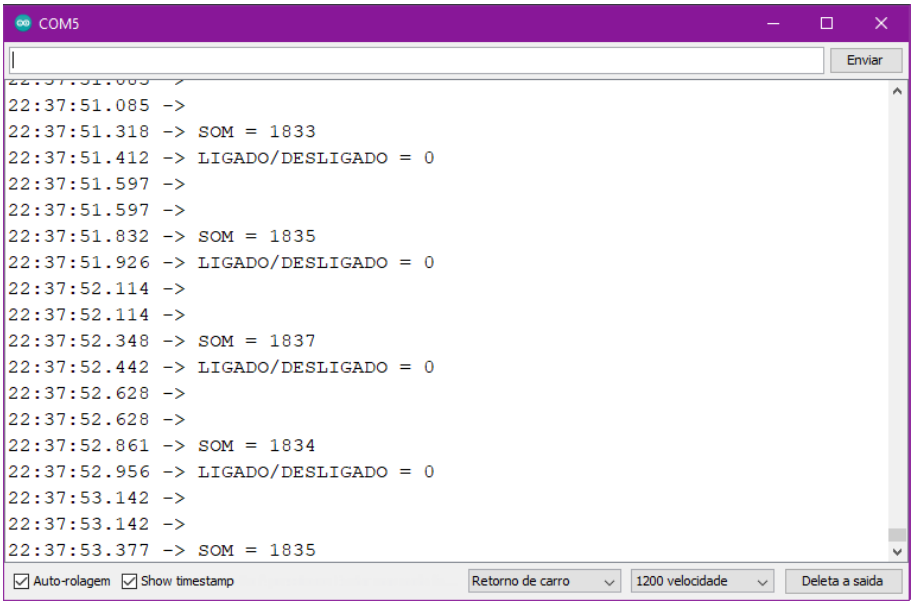

Figura 19 – Som desligado.
Fonte: Elaborado pelo autor.

Como o intuito deste trabalho é o desenvolvimento *Web* com foco na conexão com a rede local, os valores lidos pelo sensore KY-038 não foram tratados e nem calibrados usando um decibelímetro, como foi feito por Lisboa et al. (2019) para obtenção dos valores em dB (decibéis).

3.7 Página de *Login* (*front-end*)

O *front-end* da página de *login* foi elaborada com o uso do HTML e CSS para que a apresentação dos campos de entrada do nome do usuário e da senha fossem mostrados de forma confortável e intuitiva para o usuário. Essa página de *login* é mostrada na figura 20.

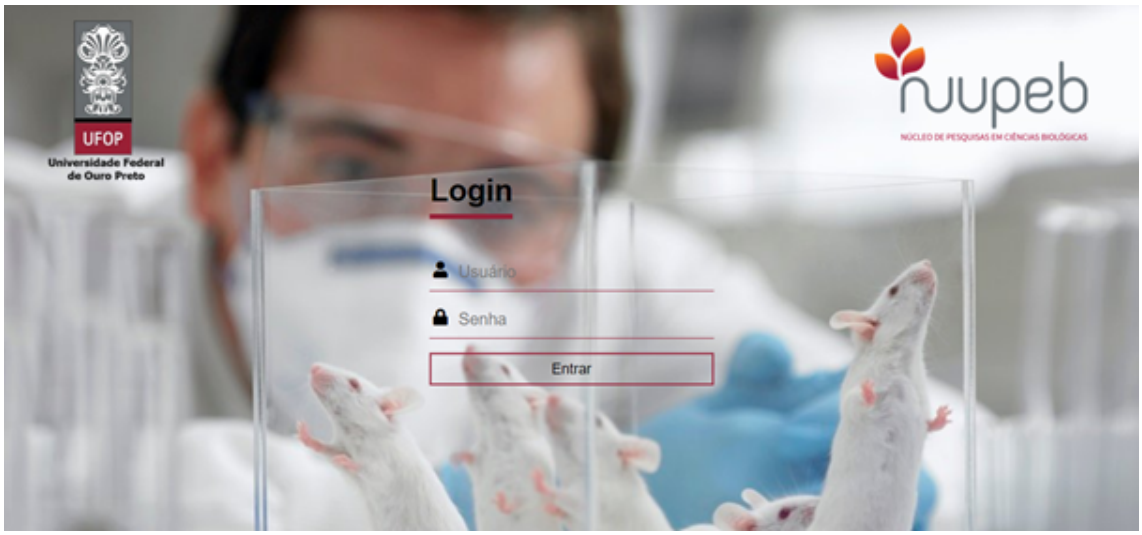

Figura 20 – Página de login.
Fonte: Elaborado pelo autor.

3.8 Página de *Login* (*back-end*)

O método *POST* é um dos métodos de requisição mais usados pelo protocolo HTTP, e ele foi projetado para que o servidor aceite os dados anexados pela requisição de armazenamento.

A página de *login* do site contém duas caixas de formulário para inserção do nome de usuário e a senha de quem estiver acessando a página que contém os dados em tempo real. Esse formulário usa o método *POST* para selecionar os dados que foram inseridos. Após essa seleção, é feito uma *Query*, que é responsável pela solicitação de informações feita ao banco de dados, ou seja, é a *Query* que verifica se as informações inseridas correspondem às que estão cadastradas previamente.

Essa verificação é crucial para que haja segurança em relação ao acesso de pessoas aos dados presentes no site, permitindo o acesso apenas de pessoas autorizadas e pré-cadastradas. Esse cadastro só é possível de ser feito tendo acesso direto ao banco de dados, então é necessário entrar em contato com alguém responsável pela administração do sistema.

3.9 Conexão do ESP32 à página *Web*

Antes de iniciar a programação para realizar a conexão do ESP32 à rede *Wi-Fi*, foi selecionada a sua placa no gerenciador de placas da plataforma do Arduino para que a *IDE* consiga identificá-la, de maneira a evitar erros, como mostra a figura 21.

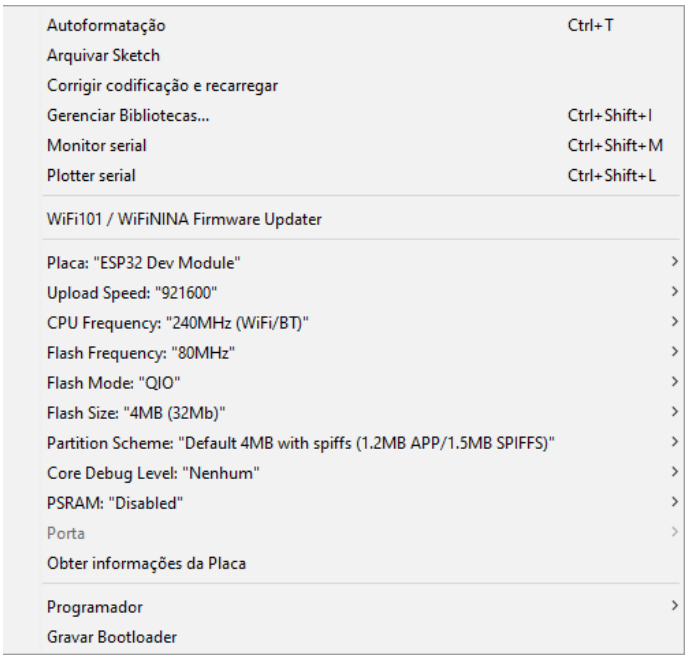

Figura 21 – Placa ESP32 selecionada.
Fonte: Elaborado pelo autor.

Com a placa selecionada, foi utilizado um código fonte para iniciar os testes de conexão, onde foram inseridos os dados da rede *Wi-Fi*, sua senha, e o *link* onde se encontra o servidor do biotério, além de uma chave de autenticação HTML. Vide apêndice A.

No código responsável por realizar essa conexão do ESP32 à rede *Wi-Fi*, foi realizada uma requisição de dados do banco de dados do site do biotério. Como nesse momento ainda se tratava de um teste, o código não retornou nada, mas foi possível ver que a conexão foi feita com sucesso. Algumas variáveis foram criadas para receberem os valores lidos pelos sensores, para que esses valores possam ser enviados para a página *Web*.

Nessa etapa, o *POST* faz a requisição dos dados que estão sendo adicionados pelas leituras dos sensores em um determinado período de tempo. Com isso, o sistema envia os dados dos sensores para a página *Web* e a mesma salva-os no banco de dados presente no servidor da UFOP. O próximo passo é mostrá-los na página.

3.10 Página para exibição de dados (*front-end*)

A figura 22 mostra a página de exibição de dados, onde existe uma opção para trocar a sala que está sendo monitorada, uma tabela que exibe a média dos valores medidos de cinco em cinco segundos nas últimas duas horas, e um gráfico que indica esse valor médio para cada variável ambiental. Além disso, possui um botão que leva para a página de histórico e um que desconecta o cliente, voltando para a página de *login*. Os dados presentes na página são os dados reais lidos pelos sensores.

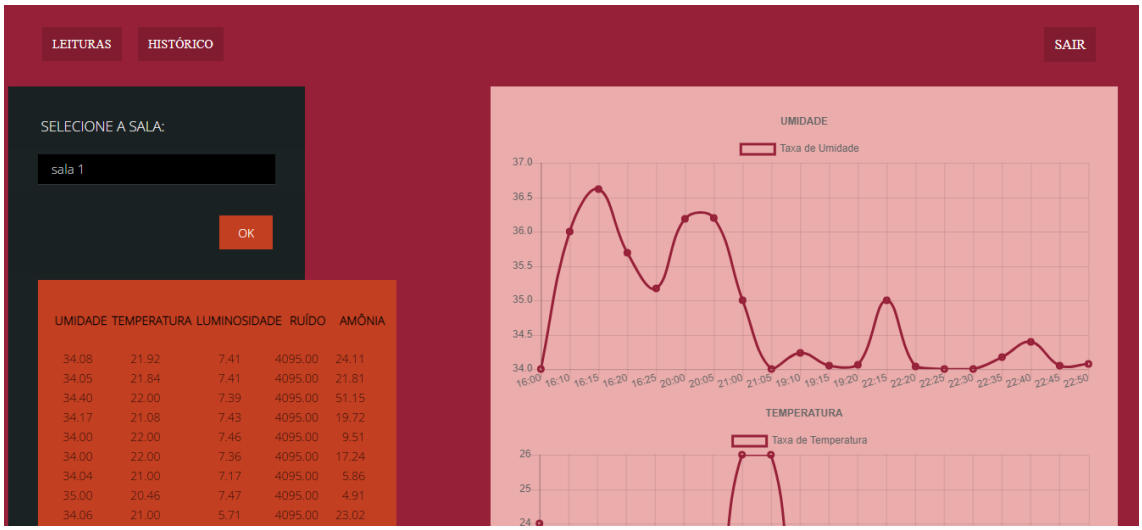

Figura 22 – Página de exibição de dados.
Fonte: Elaborado pelo autor.

3.11 Página para exibição de dados (*back-end*)

Para a exibição dos dados na página *Web*, foi necessário fazer uma conexão com o banco de dados para salvar os dados lidos. Utilizando a chave de autenticação do HTML, foi possível ler na página os dados que estão sendo enviados pelo ESP32.

Com os dados armazenados no banco de dados, o próximo passo foi exibi-los na página. Para uma exibição mais compacta e interessante tanto para o gráfico quanto para a tabela, foi feito uma média dos valores que estão sendo lidos de cinco em cinco segundos nas últimas duas horas para cada sensor.

Usando uma combinação de PHP com *JavaScript*, foi possível selecionar esses dados salvos e exibi-los na página com os valores sendo atualizados a cada *"refresh"*.

3.12 Histórico

Na página de histórico mostrada na figura 23, é possível selecionar um determinado período de dias ou meses para exibição do histórico de leitura.

Figura 23 – Histórico.
Fonte: Elaborado pelo autor.

Aqui, o método *POST* é utilizado novamente, dessa vez para selecionar o intervalo escolhido pelo usuário, e uma *Query* é feita para saber se essa data existe ou não no banco de dados. Para que esses dados apareçam na página, uma variável é responsável por buscar os dados no banco de dados e retornar para outra variável, que será responsável pela escrita dos mesmos.

4 Resultados

Esse capítulo apresenta os resultados obtidos por meio da metodologia descrita no Capítulo 3, tais como os testes de conexão, armazenamento e exibição dos dados do sistema.

4.1 Conexão do ESP32

Após os testes, foi possível conectar o ESP32 à rede sem muitos problemas, como mostra a figura 24, onde é retornado um código *200* do HTTP indicando que os dados foram enviados para a página *Web* com sucesso.

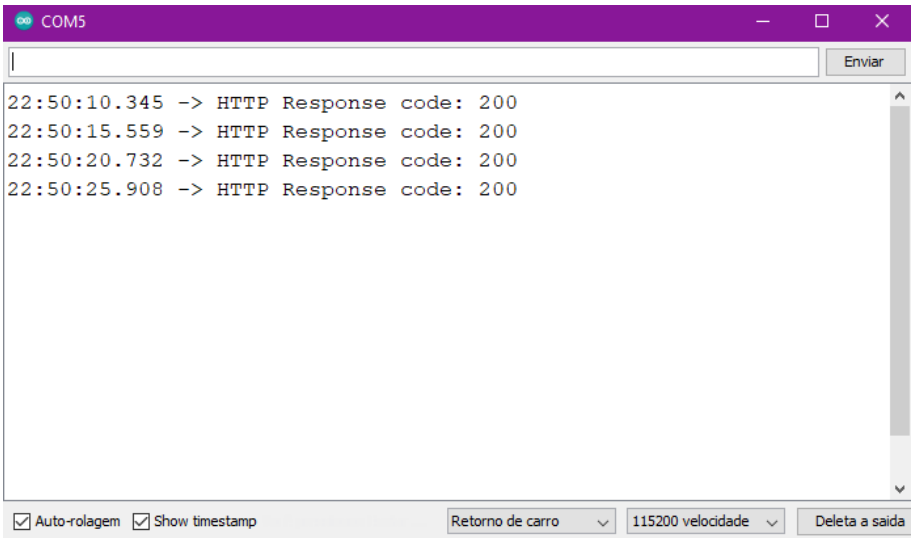

Figura 24 – Conexão realizada.
Fonte: Elaborado pelo autor.

4.2 Salvar dados no banco de dados

De acordo com a figura 25, pode-se observar que os dados enviados anteriormente foram salvos com sucesso no banco de dados presente no servidor da UFOP. Nesse servidor, foi adicionada uma tabela que armazena os dados obtidos de cada sensor, assim como o dia e a hora em que os mesmos foram enviados.

id ↓ 1	umidade	temperatura	luminosidade	ruido	amonia	data
1346	34.00	21.00	7.48	4095	9.94	2022-06-27 22:23:20
1345	34.00	21.00	7.48	4095	9.59	2022-06-27 22:23:14
1344	34.00	21.00	7.46	4095	8.92	2022-06-27 22:23:09
1343	35.00	21.00	7.41	4095	9.14	2022-06-27 22:23:04
1342	34.00	21.00	7.48	4095	8.57	2022-06-27 22:22:58
1341	34.00	21.00	7.55	4095	7.15	2022-06-27 22:22:53
1340	34.00	21.00	7.49	4095	7.27	2022-06-27 22:22:48
1339	34.00	21.00	7.01	4095	7.10	2022-06-27 22:22:42
1338	34.00	21.00	7.58	4095	6.91	2022-06-27 22:22:37
1337	34.00	21.00	7.56	4095	6.75	2022-06-27 22:22:32
1336	34.00	21.00	7.49	4095	6.39	2022-06-27 22:22:27
1335	34.00	21.00	7.57	4095	6.24	2022-06-27 22:22:22
1334	34.00	21.00	7.48	4095	6.02	2022-06-27 22:22:16
1333	34.00	21.00	7.54	4095	5.68	2022-06-27 22:22:11

Figura 25 – Dados salvos com sucesso no banco de dados.
Fonte: Elaborado pelo autor.

4.3 Mostrar dados na página *Web*

Nesta etapa, foram encontradas algumas dificuldades na forma em que os dados seriam exibidos, por se tratar de um fluxo muito alto. Inicialmente, foram exibidos todos os dados sem um controle de exibição, tornando o site muito poluído e com muita informação, e por isso foram feitas algumas mudanças.

Os dados são exibidos na página de uma maneira compacta, mostrando o resultado da média das medidas feitas de cinco em cinco segundos das últimas duas horas tanto na tabela quanto no gráfico, e nessa tabela só é exibido vinte linhas de dados. A aplicação dessas adaptações foi um sucesso e elas podem ser observadas nas figuras 26 e 27, respectivamente.

UMIDADE	TEMPERATURA	LUMINOSIDADE	RUÍDO	AMÔNIA
34.08	21.92	7.41	4095.00	24.11
34.05	21.84	7.41	4095.00	21.81
34.40	22.00	7.39	4095.00	51.15
34.17	21.08	7.43	4095.00	19.72
34.00	22.00	7.46	4095.00	9.51
34.00	22.00	7.36	4095.00	17.24
34.04	21.00	7.17	4095.00	5.86
35.00	20.46	7.47	4095.00	4.91
34.06	21.00	5.71	4095.00	23.02
34.05	21.10	4.60	4095.00	20.82
34.24	21.06	6.94	4095.00	18.36

Figura 26 – Tabela de exibição dos dados.
Fonte: Elaborado pelo autor.

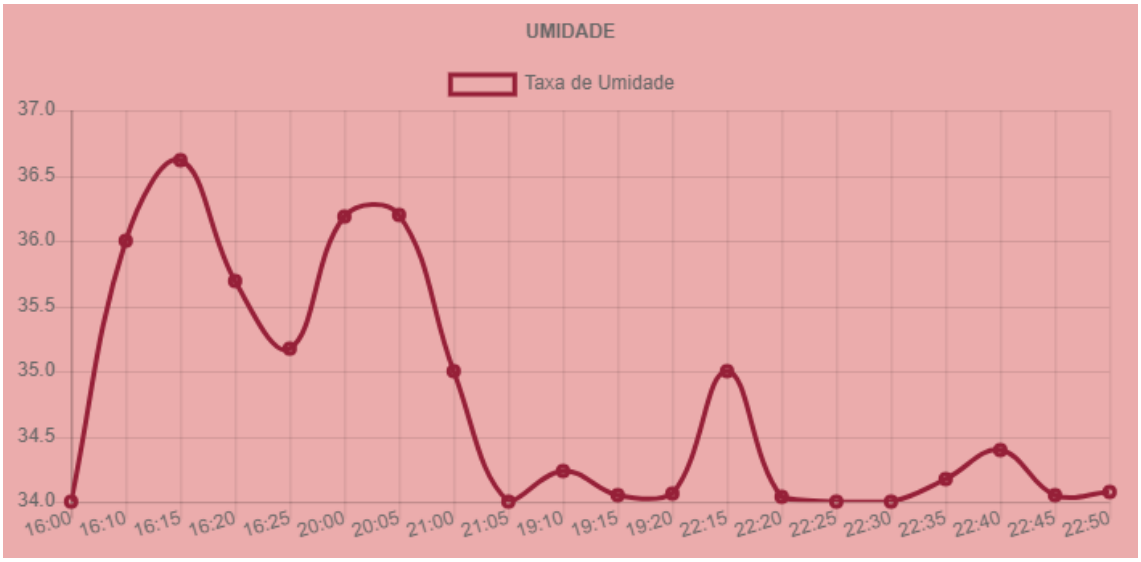

Figura 27 – Gráfico de exibição dos dados do sensor de umidade.
Fonte: Elaborado pelo autor.

Na página do histórico também foi possível realizar a seleção de um determinado período de tempo e os valores referentes a este período foram exibidos na tabela com sucesso, como mostra a figura 28:

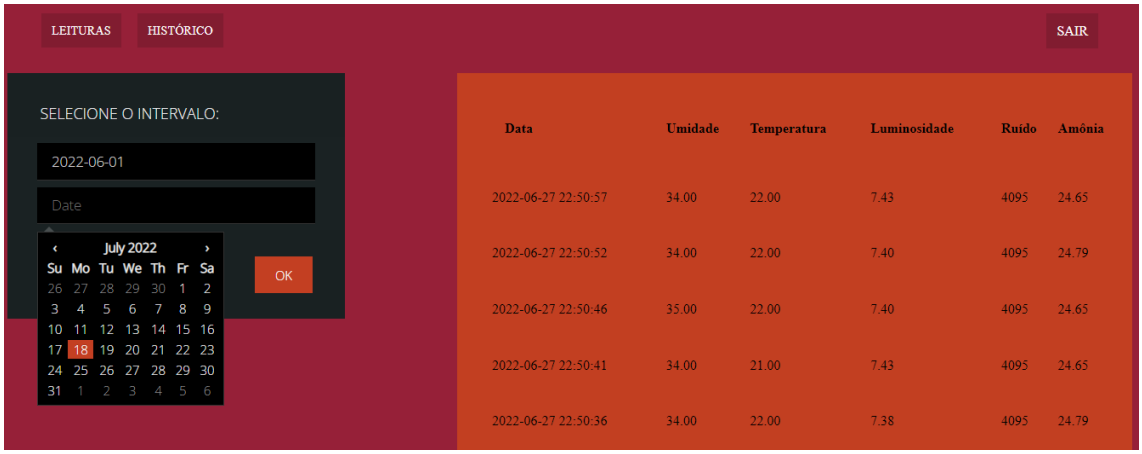

Figura 28 – Histórico.
Fonte: Elaborado pelo autor.

5 Conclusão

O projeto apresentado neste trabalho de conclusão de curso consistiu no desenvolvimento de uma plataforma *Web* contendo *layouts* com informações pertinentes ao monitoramento de parâmetros ambientais presentes no biotério da UFOP. Esse desenvolvimento envolveu adaptações no sistema existente, como a substituição da transmissão via rádio por meio do módulo NRF24101, pela transmissão via *Wi-Fi* realizada pelo módulo ESP-WROOM-32.

Os resultados obtidos na reta final do projeto se mostraram eficientes quanto à conexão do módulo *Wi-Fi* e quanto ao envio de dados para a página. Também foram obtidos bons resultados quanto ao armazenamento dos dados no banco de dados da UFOP e exibição dos mesmos na página *Web*.

Por fim, este trabalho se mostrou uma excelente oportunidade de aprendizado e desenvolvimento de competências adquiridas durante a graduação, e propocionou à discente um conhecimento não só sobre leitura de parâmetros por meio de sensores, mas também sobre conexões em redes de internet locais e armazenamentos de dados em bancos de dados.

5.1 Sugestões para trabalho futuros

Como sugestão para trabalhos futuros, propõe-se criar um sistema de comunicação com o usuário, como por exemplo, um aplicativo para *iOS* ou *Android*, onde será possível enviar alertas para o celular de quem estiver monitorando caso haja alguma mudança drástica nos parâmetros que poderiam colocar em risco os animais ou as pessoas que trabalham no laboratório.

Ademais, é essencial a criação de um protótipo físico no qual serão dispostos o módulo ESP-32 e os sensores, além da instalação do mesmo nos laboratórios da UFOP após avaliação junto aos técnicos que ali trabalham. Junto a isso, é de suma importância tratar os dados dos sensores para que os mesmos possam apresentar os dados corretos na aplicação *Web* aqui descrita.

Referências

BEIRÃO, L. J. et al. Monitagro–monitoramento de atividades agropecuárias utilizando sensores e internet das coisas. *Anais da Mostra Nacional de Iniciação Científica e Tecnológica Interdisciplinar (MICTI)-e-ISSN 2316-7165*, 2019. v. 1, n. 12, 2019. Citado na página 16.

CARDOSO, T. A. de O. Considerações sobre a biossegurança em arquitetura de biotérios. *Boletim Central Panamaense Fiebre Aftosa*, 2001. v. 64, n. 67, p. 3–17, 2001. Citado na página 12.

CONVERSE, T.; PARK, J. *PHP: a bíblia*. [S.l.]: Gulf Professional Publishing, 2003. Citado na página 19.

DAVID, C. *Sound Sensor Tutorial for Arduino, ESP8266 and ESP32*. 2021. Disponível em: <https://diyi0t.com/sound-sensor-arduino-esp8266-esp32/>. Citado na página 29.

FILIPEFLOP. *Módulo WiFi ESP32 Bluetooth*. 2022. Disponível em: <https://www-.filipeflop.com/produto/modulo-wifi-esp32-bluetooth/>. Acesso em: 04 Jun. 2022. Citado na página 20.

HOSSEINI, S. *How to Calibrate And Use MQ9 Gas Sensor w/ Arduino*. 2019. Disponível em: <https://electropeak.com/learn/how-to-calibrate-and-use-mq9-gas-sensor-w-arduino/>. Acesso em: 15 Mar. 2022. Citado 3 vezes nas páginas 26, 28 e 29.

LEITE, M. A. F. et al. Proposta de automação de baixo custo para aviários utilizando o microcontrolador esp32. In: *XII Congreso de AgroInformática (CAI 2020)-JAIIO 49 (Modalidad virtual)*. [S.l.: s.n.], 2020. Citado na página 16.

LIMA, M.; ALVES, F.; JUCA, S. Sistema iot para monitoramento de temperatura e umidade ambientes e acionamento remoto de cargas. In: SBC. *Anais da IV Escola Regional de Informática do Piauí*. [S.l.], 2018. p. 232–237. Citado na página 15.

LISBOA, D. D. et al. Desenvolvimento de uma plataforma de monitoramento de ambientes para biotérios. In: *Congresso Brasileiro de Automática-CBA*. [S.l.: s.n.], 2019. v. 1, n. 1. Citado 6 vezes nas páginas 12, 21, 22, 23, 26 e 32.

MORAIS, C. A. A. e Marcos Vinicius Bueno de. Sensibilidade do sensor de intensidade sonora para detecÇÃo de granizo. *REGRAD - Revista Eletrônica de Graduação do UNIVEM - ISSN 1984-7866*, 2018. v. 11, n. 01, p. 16–27, 2018. ISSN 1984-7866. Disponível em: <https://revista.univem.edu.br/REGRAD/article/view/2539>. Citado na página 29.

PIETERS, A. *ESP32 – PinOut*. 2022. Disponível em: <https://www.studiopieters.nl-/esp32-pinout/>. Acesso em: 04 Jun. 2022. Citado na página 21.

SANTOS, M. R. V. et al. Parâmetros bioquímicos, fisiológicos e morfológicos de ratos (rattus novergicus linhagem wistar) produzidos pelo biotério central da universidade federal de sergipe. *Scientia Plena*, 2010. v. 6, n. 10, 2010. Citado na página 12.

SILVA, M. S. *Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata*. [S.l.]: Novatec Editora, 2007. Citado na página 19.

SILVA, M. S. *Criando sites com HTML: sites de alta qualidade com HTML e CSS*. [S.l.]: Novatec Editora, 2008. Citado na página 18.

SOUZA, Í. R. d. et al. Avaliação e refatoração de front–end web por meio da aplicação de padrões de projetos. 2019. Universidade Federal Rural do Semi-Árido, 2019. Citado na página 18.

SYSTEMS, E. *"ESP-WROOM-32 Datasheet"*. 2016. [Revisado em Set. 2019]. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1179101/ESPRESSIF/ESP-WROOM-32.html>. Acesso em: 04 Jun. 2022. Citado na página 19.

Apêndices

APÊNDICE A – Código fonte do ESP32

```
// Bibliotecas
#include <WiFi.h>
#include <WebServer.h>
#include <HTTPClient.h>
#include <Adafruit_Sensor.h>
#include "DHT.h"
#define DHTTYPE DHT11

WiFiClient client;

const char* ssid = "REDE_WIFI";
const char* password = "SENHA_WIFI";

const char* server = "SERVIDOR";
String apiKeyValue = "CHAVE_DE_AUTENTICAO";

//WebServer server(80);

//DHT11
uint8_t DHTPin = 4;
DHT dht(DHTPin, DHTTYPE);
float Temperature;
float Humidity;

//LDR
const int LEDluz = 18;
const int portaLDR = 34;

//MQ135
const int MQpin = 36;
const int LEDgas = 14;
const int DO = 22;

//KY-038
```

```
const int LEDsom = 18;
const int digital_som = 0;
const int analog_som = 2;

void setup() {
  Serial.begin(115200);

  //definindo as portas como input e output
  pinMode(LEDgas, OUTPUT);
  pinMode(LEDsom, OUTPUT);
  pinMode(digital_som, INPUT);
  pinMode(DO, INPUT);
  pinMode(LEDluz, OUTPUT);
  pinMode(DHTPin, INPUT);

  dht.begin();
  pinMode(portaLDR, INPUT);

  Serial.println("Conectando a ");
  Serial.println(ssid);

  //conectando    rede local
  WiFi.begin(ssid, password);

  //checa de o WiFi est  conectado    rede local
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado..!");
  Serial.print("Endere o IP: ");
  Serial.println(WiFi.localIP());

 // server.begin();
  Serial.println("Servidor HTTP iniciado");

}
```

```
void loop() {

  if(WiFi.status()==WL_CONNECTED){
    HTTPClient http;
    http.begin(server);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    //DHT11
    Temperature = dht.readTemperature();
    Humidity = dht.readHumidity();

    //LDR
    int leitura = analogRead(portaLDR);
    float sensorLDR = leitura * (3.3/1024.0);

    //MQ135
    int alarme = 0;
    float sensor_volt;
    float RS_gas;
    float taxa;
    float R0 = 0.15;
    int sensorValue = analogRead(MQpin);

    sensor_volt = ((float)sensorValue / 1024) * 5.0;
    RS_gas = (5.0 - sensor_volt) / sensor_volt;

    taxa = RS_gas / R0;

    //KY-038
    int digitalValue = digitalRead(digital_som);
    int analogValue = analogRead(analog_som);

    if (analogRead(portaLDR) < 700){
        digitalWrite(LEDluz, HIGH);
      }else{
        digitalWrite(LEDluz, LOW);
      }

    alarme = digitalRead(DO);
```

```
    if (alarme == 1) {
      digitalWrite(LEDgas, HIGH);
    }else{
      digitalWrite(LEDgas, LOW);
    }

    //Serial.println(analogValue);
    if (analogValue < 1800){
      digitalWrite(LEDsom, HIGH);
    }else{
      digitalWrite(LEDsom, LOW);
    }

    String httpRequestData = "api_key=" +
        apiKeyValue + "&umidade=" + Humidity
        + "&temperatura=" + Temperature + "&luminosidade=" + sensorLDR
        + "&ruido=" + analogValue + "&amonia=" + taxa + "";

    int httpResponseCode = http.POST(httpRequestData);

    if (httpResponseCode > 0){
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
    }else{
      Serial.print("Erro code: ");
      Serial.println(httpResponseCode);
    }
    http.end();

  }else{
    Serial.println("WiFi desconectado");
  }
  delay(5000);
}
```

Anexos

ANEXO A – Calibração MQ135

```
/*

MQ9 Calibration

modified on 19 Feb 2019
by Saeed Hosseini
Home
this code is based on https://www.elecrow.com/wiki/index.php?title=
Analog_CO/Combustible_Gas_Sensor(MQ9)
*/

const int MQpin = GPIO_NUM_36;

void setup() {
  Serial.begin(9600);
}

void loop() {
  float sensor_volt;
  float RS_air; //  Rs in clean air
  float R0;  // R0 in 1000 ppm LPG
  float sensorValue;

//Average
    for(int x = 0 ; x < 100 ; x++)
  {
    sensorValue = sensorValue + analogRead(MQpin);
  }
  sensorValue = sensorValue/100.0;
//-----------------------------------------------/

  sensor_volt = (sensorValue/1024)*5.0;
  RS_air = (5.0-sensor_volt)/sensor_volt; // Depend on RL on yor module
  R0 = RS_air/9.9; // According to MQ9 datasheet table
```

```
  Serial.print("sensor_volt = ");
  Serial.print(sensor_volt);
  Serial.println("V");

  Serial.print("R0 = ");
  Serial.println(R0);
  delay(1000);

}
```