UNIVERSIDADE DO ESTADO DO AMAZONAS

ESCOLA SUPERIOR DE TECNOLOGIA – EST

MATHEUS DE OLIVEIRA ASSUNÇÃO

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO BASEADO EM *INTERNET OF THINGS* PARA MONITORAMENTO DE FOCOS DE INCÊNDIO NA FLORESTA AMAZÔNICA

Manaus

2022

MATHEUS DE OLIVEIRA ASSUNÇÃO

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO BASEADO EM *INTERNET OF THINGS* PARA MONITORAMENTO DE FOCOS DE INCÊNDIO NA FLORESTA AMAZÔNICA

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro em Eletrônica.

Orientador: Fábio de Sousa Cardoso, Dr.

_____

Manaus

2022

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
André Luiz Nunes Zogahib
Vice-Reitor:
Kátia do Nascimento Couceiro
Diretor da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Eletrônica:
Bruno da Gama Monteiro

Banca Avaliadora composta por: Data da defesa: <17/10/2022>.
Prof. Fábio de Sousa Cardoso, Dr. *(Orientador)*
Prof. Israel Gondres Torné, Dr.
Prof. Rubens de Andrade Fernandes, Me.

CIP – Catalogação na Publicação

Assunção, Matheus de Oliveira

Desenvolvimento de um protótipo de dispositivo baseado em *internet of things* para monitoramento de focos de incêndio na floresta amazônica / Matheus de Oliveira Assunção; [orientado por] Fábio de Sousa Cardoso. – Manaus: 2022.
50 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica). Universidade do Estado do Amazonas, 2022.

1. Sistemas Embarcados. 2. Lora. 3. ESP32. I. Cardoso, Fábio de Sousa.

MATHEUS DE OLIVEIRA ASSUNÇÃO

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO BASEADO EM *INTERNET OF THINGS* PARA MONITORAMENTO DE FOCOS DE INCÊNDIO NA FLORESTA AMAZÔNICA

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro em Eletrônica.

Nota obtida: 9,4 (NOVE VÍRGULA QUATRO)

Aprovada em 17 / 10 / 2022.

Área de concentração: Sistemas Embarcados

BANCA EXAMINADORA

Orientador: Fábio de Sousa Cardoso, Dr.

Avaliador: Israel Gondres Torné, Dr.

Avaliador: Rubens de Andrade Fernandes, Me.

Manaus – 2022

AGRADECIMENTOS

Primeiramente agradeço a Deus pela vida, pela sua graça e misericórdia. Agradeço a meus pais, Maria Edineide e Aser Assunção, por todo esforço e investimento nos meus estudos, e se cheguei até aqui foi porque estive nos ombros desses gigantes. Agradeço a minha esposa, Pâmella Assunção, por estar ao meu lado sempre me apoiando e ajudando durante essa etapa. Agradeço meus irmãos, Acsa Júlia e Marcos Kalleb por serem grandes amigos que sempre me apoiaram e estiveram ao meu lado. Agradeço meus familiares em geral. Agradeço meus amigos do Hub – Tecnologia e inovação que foram grandes amigos nessa caminhada, e em especial ao meu professor orientador Fábio Cardoso, Gabriel, Rubens, Arlley, Wesley, Pantoja, Samuel, Lennon e etc.

“O temor do Senhor é o princípio da sabedoria, e o conhecimento do Santo é entendimento”

Provérbios 9:10

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de sistema de monitoramento focos de incêndio em regiões de florestas com base nos parâmetros de temperatura e leitura de partículas de dióxido de carbono, os testes do protótipo de sistema serão feitos na região de floresta próximo ao HUB tecnologia e inovação localizado na Escola Superior de Tecnologia da UEA. Com auxílio de sensores esses dados serão coletados por um ESP32 que possui um módulo Lora embutido, esses dados serão enviados por Lora a um módulo concentrador, onde o mesmo irá disponibilizar os dados por WiFi a um broker MQTT, que em seguida será salvo em um banco de dados SQLite, e os dados podem ser visualizados através do aplicativo SQLite Browser. Inicialmente, será feita uma apresentação onde serão abordados dados de queimada disponibilizados pelas autoridades competentes. Em seguida, será apresentada uma fundamentação teórica sobre a queimada na floresta, internet das coisas, ESP32, modulação Lora e o protocolo MQTT, do inglês *Message Queuing Telemetry Transport*. Posteriormente, são apresentadas as etapas e os materiais necessários para o desenvolvimento do dispositivo de monitoramento, seguidos da descrição detalhada do experimento, ferramentas utilizadas, e implementação do que foi supracitado. Como resultado dos experimentos realizados ao longo do trabalho, os parâmetros monitorados serão exibidos através do *software* SQLite browser, que é um visualizador do arquivo de banco de dados. Por fim, os dados obtidos mostraram que o protótipo de sistema é capaz de monitorar os parâmetros de temperatura e gás nos arredores da Escola Superior de Tecnologia.

Palavras-chaves: Focos de incêndio, ESP32, Lora.

ABSTRACT

This work presents the development of a prototype system for monitoring fire outbreaks in forest regions based on temperature parameters and reading of carbon dioxide particles The system prototype tests will be carried out in the forest region close to the HUB – Tecnologia e Inovação located at the Escola Superior de Tecnologia da UEA . With the help of sensors, these data will be collected by an ESP32 that has a built-in Lora module, these data will be sent by Lora to a concentrator module, where it will make the data available via WiFi to an MQTT broker, which will then be saved in a SQLite database, and the data can be viewed through the SQLite Browser application. Initially, a presentation will be made in which fire data provided by the competent authorities will be discussed. Then, a theoretical foundation will be presented about burning in the forest, internet of things, ESP32, Lora modulation and the MQTT protocol, Message Queuing Telemetry Transport. Subsequently, the steps and materials necessary for the development of the monitoring device are presented, followed by a detailed description of the experiment, tools used, and implementation of the aforementioned. As a result of the experiments carried out throughout the work, the monitored parameters will be displayed through the SQLite browser software, which is a database file viewer. Finally, the data obtained showed that the prototype system is capable of monitoring the temperature and gas parameters in the surroundings of the Escola Superior de Tecnologia.

Keywords: Fire outbreaks, ESP32, Lora.

SUMÁRIO

INTRODUÇÃO

O tema deste trabalho é protótipo de sistema para monitoramento de focos de incêndio na floresta amazônica. Sabe-se que o aquecimento global tem se tornado uma preocupação ainda maior a cada ano. Eventos climáticos extremos estão acontecendo com uma frequência cada vez maior, levando as autoridades mundiais a buscarem uma solução conjunta para esse problema.

Atualmente, sabe-se que a floresta amazônica possui um papel muito importante não só na manutenção dos ciclos da chuva em regiões brasileiras, mas também ajuda a regular o clima global e contribui com a imensa biodiversidade local. Entretanto, a destruição das florestas provenientes de queimadas e o desmatamento pode fazer, segundo os cientistas, que o planeta sofra grandes riscos caso a temperatura ultrapasse 2º C acima dos níveis pré-industriais (CAMPOS; HIGUCHI, 2009).

Existem vários fatores que podem causar ocorrência de incêndios na Amazônia. Dentre eles, estão as queimadas intencionais que geralmente são promovidas na estação seca, quando lavouras são mais fáceis de queimar e as florestas estão mais vulneráveis (NEPSTAD; MOREIRA; ALENCAR, 1999).

O monitoramento das queimadas nas florestas brasileiras é feito por um conjunto de satélites que possuem sensores óticos operando na faixa termal-média. Segundo o Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA), mesmo com um número considerável de satélites, algumas queimadas não conseguem ser detectadas. Existem condições que impedem ou prejudicam muito a detecção das queimadas como é o caso de: frentes de fogo com menos de 30 metros; fogo apenas no chão de uma floresta densa, sem afetar a copa das árvores; nuvens cobrindo a região; queimadas de pequena duração; e fogo em uma encosta de montanha.

Os problemas do monitoramento por satélite, além do seu alto custo, fizeram com que esta pesquisa fosse apresentada como alternativa a solução atual. Este projeto de pesquisa tem como objetivo desenvolver um protótipo de monitoramento de focos de incêndio, usando sensores de fumaça e temperatura conectados a uma plataforma de *hardware* para desenvolvimento, da fabricante Heltec, que possui um microcontrolador ESP32 e um módulo LoRa.

Este trabalho tem como hipótese desenvolver um protótipo de monitoramento de focos de incêndio em tempo real, capaz de realizar este monitoramento de forma descentralizada, tornando possível o uso do sistema por particulares donos de grandes terras. Este protótipo

emprega os sensores de gás e fumaça MQ-9 e o sensor de temperatura BMP-280 ambos associados ao microcontrlador ESP32 e utilizam o módulo LoRa para transmitir os dados. O objetivo deste trabalho é desenvolver este protótipo para monitorar os parâmetros supracitados e criar uma base de dados que podem ser visualizados com o software SQLite Browser.

A justificativa para implementação deste projeto se dá por propor um sistema de monitoramento em tempo real, além da descentralização do monitoramento, tornando possível o uso do protótipo por particulares.

Será feita uma revisão teórica dos seguintes assuntos: Queimadas na floresta amazônica, *internet of things*, Lora, microcontrolador ESP32, barramento I2C, sensores, *message queuing telemetry transport* (MQTT) e banco de dados sqlite.

Este trabalho está dividido em quatro capítulos, os quais são: referencial teórico, materiais e métodos, implementação e análise dos resultados obtidos.

O primeiro capítulo aborda o referencial teórico, onde são abordados os assuntos supracitados que estão relacionados ao projeto.

No segundo capítulo são mostrados os materiais utilizados para a implementação do protótipo. Em seguida, será apresentado método proposto para o desenvolvimento do protótipo, se forma sequencial, começando na pesquisa e finalizando com a visualização na interface.

No terceiro capítulo é apresentado a descrição detalhada do sistema implementado, onde primeiramente, serão apresentados os módulos que constituem o sistema, os quais são: placa de coleta de dados, placa concentradora, *backend*, banco de dados e visualização através do software.

No quarto capítulo, serão apresentados os resultados obtidos com a implementação do trabalho proposto. Serão apresentadas as análises dos resultados baseadas no referencial teórico e no aprendizado adquirido durante o desenvolvimento do projeto.

Para finalizar, a conclusão, será retomado o que foi proposto como hipótese em comparação com os resultados obtidos, e será sugerido novas implementações em trabalhos futuros.

1 REFERENCIAL TEÓRICO

Neste capítulo, serão abordados os aspectos teóricos dos assuntos relacionados ao projeto. Inicialmente, será feita uma contextualização sobre as queimadas na região da floresta amazônica. Em seguida, serão abordados conceitos introdutórios sobre internet das coisas, tecnologia de comunicação LoRa, microcontrolador ESP32, sensores, protocolo mqtt e banco de dados SQlite.

1.1 QUEIMADAS NA FLORESTA AMAZÔNICA

A Amazônia é a floresta mais extensa do planeta, presente em vários países da américa do sul com uma área total de aproximadamente 6,3 milhões de Km². Apenas o território brasileiro possui 5,5 milhões de Km² abrangendo todos os estados do norte do país e parte dos estados do Mato Grosso e Maranhão (SANTOS *et al*, 2017).

O governo passou a monitorar o desmatamento na Amazônia desde 1985 e os números apresentados são de que aproximadamente 10% da floresta foi destruída. Esse desmatamento geralmente acontece após a extração de árvores de grande valor levando as pessoas a atear fogo na vegetação remanescente. Embora, também, ocorram queimadas para preparar áreas para agricultura, pecuária ou especulação de terras. Geralmente, os picos dessas queimadas ocorrem na estação seca, entre os meses de julho e outubro (HUMAN RIGHTS WATCH, 2020).

O fogo na Amazônia brasileira emite grandes quantidades de gases que causam o efeito estufa. Essas queimadas emitem gases não só da biomassa que queima, mas também da parte que não queima. Quando há uma queimada, ocorre uma grande liberação de gás carbônico, mas ele não é o único, gases-traço como metano, monóxido de carbono e nitroso de oxigênio também são liberados. A grande quantidade de gases de efeito estufa liberados pelas queimadas tem tanto um impacto presente, como um potencial gigantesco de impacto futuro (FEARNSIDE, 2002).

1.2 INTERNET OF THINGS

Segundo Ashton (2009), os dados presentes nos computadores e na internet são quase totalmente dependentes dos seres humanos, logo, se os computadores tivessem a capacidade de coletar dados sem intervenção humana, haveria uma redução drástica dos desperdícios ocasionados por desgaste ou mal uso de um equipamento. Portanto, a internet das coisas seria capaz de fazer com que os computadores coletassem informações do ambiente ou de um objeto através de uma rede sem fio, criando uma grande rede de dados.

1.3 LORA

LoRa é uma tecnologia de rádio frequência, de propriedade da Semtech, que permite uma comunicação em longas distâncias, dependo da área de propagação do sinal podendo superar 10 km, apresentando um baixo consumo de energia elétrica. A tecnologia LoRa utiliza frequências sub-gigahertz (abaixo de 1 GHz), regulamentada em bandas diferentes de acordo com cada região do planeta. No Brasil, a faixa de frequência regulamentada encontra-se entre 915MHz a 928MHz (BERTOLETI, 2019).

A arquitetura da rede LoRa é dividida em três partes, primeiramente, existem os módulos responsáveis por captar informações dos sensores e transmitir os dados via LoRa, esses módulos são chamados de dispositivos finais. Logo em seguida, há o *gateway* que é responsável por receber as mensagens via LoRa, enviada pelos dispositivos finais, e encaminhá-las para internet. Por último, existe a camada de aplicação que, dentre outras funções, é responsável pela visualização dos dados na internet (SILVA NETO, 2018).

1.4 MICROCONTROLADOR ESP32

Microcontroladores são equipamentos programáveis desenvolvidos para atuar em tarefas definidas no seu código fonte, suas principais características são o seu pequeno porte e seu baixo custo (SANTOS; LARA JUNIOR, 2019).

O ESP32 é um módulo microcontrolador com Wi-Fi e Bluetooth, conta com um poderoso poder de processamento capaz de executar tarefas que vão desde sensoriamento de baixa potência até tarefas mais exigentes como codificação de voz e *streaming* de música. As especificações do ESP32 mostram que sua CPU é composta de dois núcleos, com frequência de *clock* ajustável de 80 a 240 MHz, além de possuir um grande conjunto de periféricos como ethernet, SPI, UART, I2S e I2C (ESPRESSIF SYSTEMS, 2021).

1.4.1 Módulo ESP32 LoRa da Heltec

A evolução tecnológica das plataformas de *hardware* para desenvolvimento e sua popularização, abriu grandes possibilidades tanto aos entusiastas da área quanto aos profissionais. O ESP32 foi um sucesso no mundo *Internet of Things* (IoT) devido ao seu poder de processamento e uma grande quantidade de periféricos. Para complementar um módulo que já era potente, a Heltec criou um módulo híbrido de ESP32 e LoRa, conhecido como WiFi LoRa 32 (BERTOLETI, 2019).

Esse módulo, além de possuir o ESP32 como unidade central de processamento, é dotado de conectividade Wi-Fi, Bluetooth e LoRa, e possui um *display* OLED de 0,96". Assim como ilustrado na figura 1.

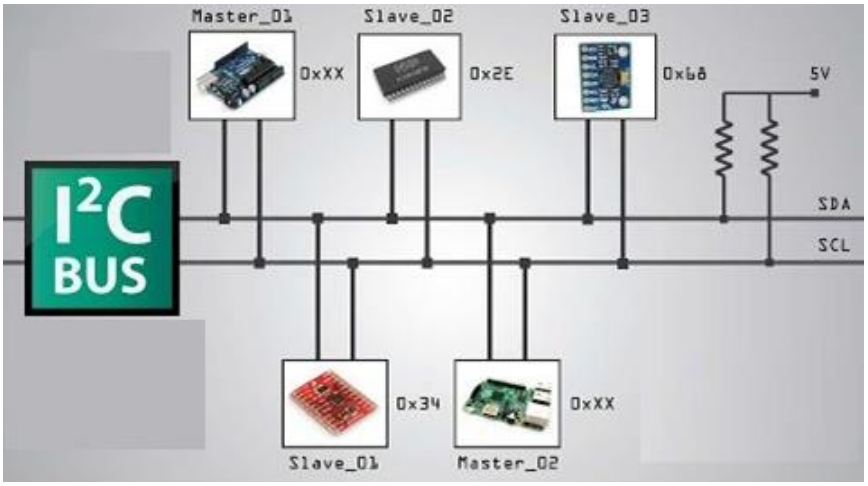
Figura 1 – ESP32 LoRa da Heltec.

Fonte: (BERTOLETI, 2019, p. 30).

1.5 BARRAMENTO I2C

I2C é um barramento de comunicação serial que utiliza apenas dois fios, seu nome vem da sigla Inter-Integrated Circuit, este protocolo é muito utilizado para conectar dispositivos periféricos que não operam em alta velocidade à microcontroladores. A figura 2 ilustra a conexão no barramento I2C.

Figura 2 – Barramento I2C



Fonte: (How To Mechatronics, 2022, p. 1)

O barramento de comunicação I2C é muito utilizado em projetos eletrônicos pois é facilmente implementado em projetos que possuem uma comunicação ente mestre e escravo, além de que esse barramento suporta entre 112 à 128 dispositivos ao usar 7 bits de endereçamentos e podendo suportar entre 1008 e 1024 dispositivos caso seja utilizado um endereçamento de 10 bits. Isso é possível pois, cada dispositivo tem identificador predefinido ou endereço de dispositivo exclusivo para que o mestre possa escolher qual dispositivo se comunicará. Os dois fios ou linhas desse barramento são chamados de Serial, um deles é o relógio que sincroniza a transferência dos dados que serão enviados no barramento I2C e é gerado pelo mestre, este fio é chamado de SCL e existe os fios de dados que é a informação a ser enviada, este é chamado de SDA. Os dados são transferidos em sequências de 8 bits, após o barramento de dados ser colocado em nível lógico baixo a transmissão começa, primeiramente, vem a sequência de 8 bits de endereço do escravo, para qual os dados estão sendo enviados. Após cada sequência de 8 bits é seguido por um bit de reconhecimento chamado “Acknowledge”, em seguido é enviado uma outra sequência de endereçamento,

porém agora é dos registros internos do escravo. Após isso, seguem-se as sequências de dados a serem enviados para o escravo.

1.6 SENSORES

Sensor é o termo utilizado para descrever dispositivos sensíveis à alguma forma de energia ambiente, essa energia pode ser luminosa, térmica, cinética e outras. Essas informações se relacionam com a grandeza que precisa ser medida como: pressão, velocidade, temperatura e outras. Os sensores podem ser subdivididos em analógicos e digitais. Sensores analógicos podem assumir qualquer valor, dentro da sua faixa de operação, ao longo do tempo, enquanto que sensores digitais podem apenas assumir dois valores que podem ser interpretados como zero e um (WENDLING, 2010). A figura 3 ilustra os sensores mais utilizados em internet das coisas.

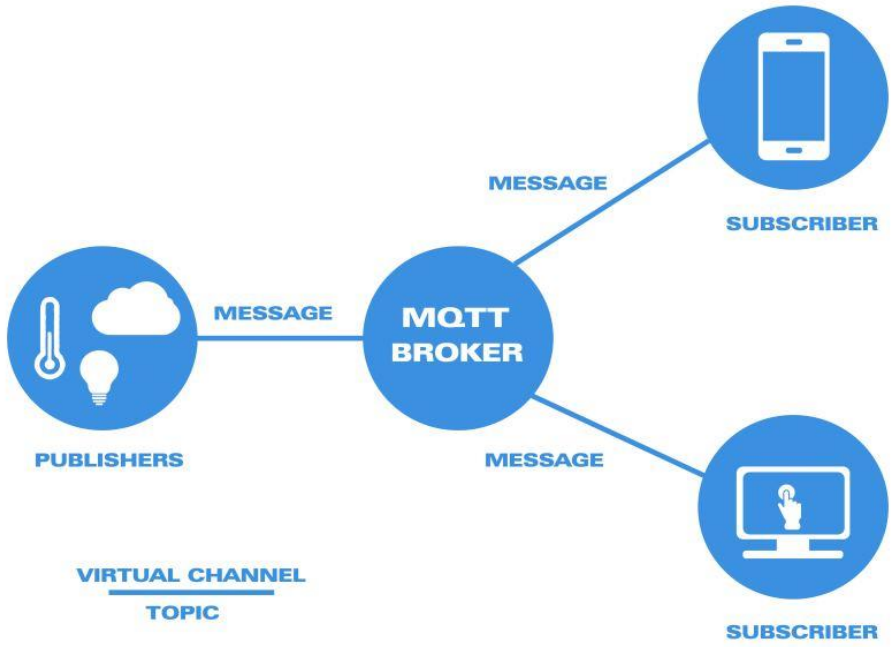
Figura 3 – Ilustração de sensores.

Fonte: (shutterstock, 2022, p.1)

1.7 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT)

O protocolo MQTT foi desenvolvido em 1999 pela IBM com o objetivo de ser mais eficiente que os existentes comparando com largura de banda e energia, o *Message Queuing Telemetry Transport* (MQTT) é um protocolo (padrão ISO/IEC PRF 20922) de mensagens leves baseado em publicação e subscrição (*Publish and Subscriber*) que roda sob o protocolo TCP/IP, como indica a figura 3 com um exemplo de topologia MQTT.

Figura 4 – Esquema de funcionamento do MQTT.



Fonte: (NOVUS, 2022, p.1).

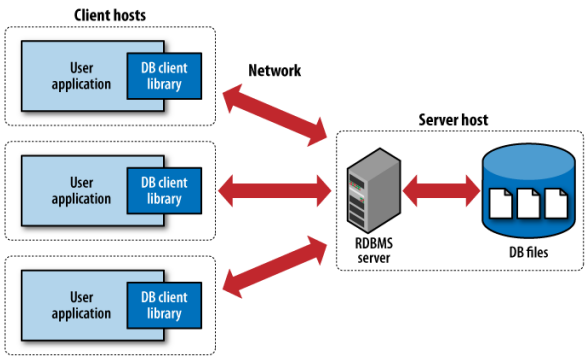
O protocolo MQTT define dois tipos de entidades na rede: um *message broker* e inúmeros clientes. O broker é um servidor que recebe todas as mensagens dos clientes e, em seguida, redireciona essas mensagens para os clientes de destino, ou seja, para os clientes que assinaram o tópico ao qual a mensagem pertence. Um cliente pode ser um microcontrolador que faz a leitura de um sensor ou uma aplicação que faz a leitura desse sensor. O modelo de publicação e assinatura possui um padrão de trocas de mensagens. Neste padrão, um cliente se conecta ao broker, que é o gerenciador da rede, e envia uma mensagem juntamente com o tópico da mesma ao broker, caso uma aplicação necessite dessa informação, esta vai assinar o mesmo tópico para ter acesso a essa publicação (IBM, 2022).

1.8 BANCO DE DADOS

Banco de dados são dados estruturados, armazenados eletronicamente em um computador. Normalmente existe um sistema gerenciador de banco de dados (SGDB) que faz o gerenciamento desse banco. Os dados no tipo mais comuns de banco de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes, dessa forma, os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados.

O SQL foi desenvolvido pela IBM em meados da década de 70, com a Oracle como a principal contribuinte, o que levou à implementação do padrão SQL ANSI. O SQL é uma linguagem de programação utilizada por quase todos os bancos de dados relacionais para consultar, manipular e definir dados e fornecer controle de acesso (ORACLE, 2022). A figura 5 aborda o funcionamento de um banco de dados.

Figura 5 – Esquema de funcionamento de um banco de dados.



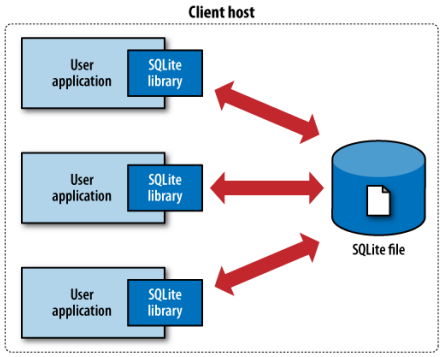
Fonte : (DevOpedia, 2022, p.1)

1.8.1 **Sqlite**

O SQLite é um banco de dados relacional que, diferentemente de outras ferramentas do tipo, não armazena informações em um servidor. Essa independência acontece porque ele consegue colocar os seus arquivos dentro de si próprio. Essa base de dados é de código aberto e gratuita, sendo muito utilizada em aplicações mobile, com foco no sistema Android.

Ao mesmo tempo em que dispensa um servidor, o SQLite também não demanda nenhum tipo de configuração. Assim, seu uso como base de dados para diferentes tipos de aplicações se torna, além de fácil, mais fluído, dinâmico e leve. (SQLite, 2022). A figura 6 aborda o esquema de funcionamento de um banco de dados sqlite.

Figura 6 – Esquema de funcionamento do banco de dados SQLite.



Fonte : (DevOpedia, 2022, p.1)

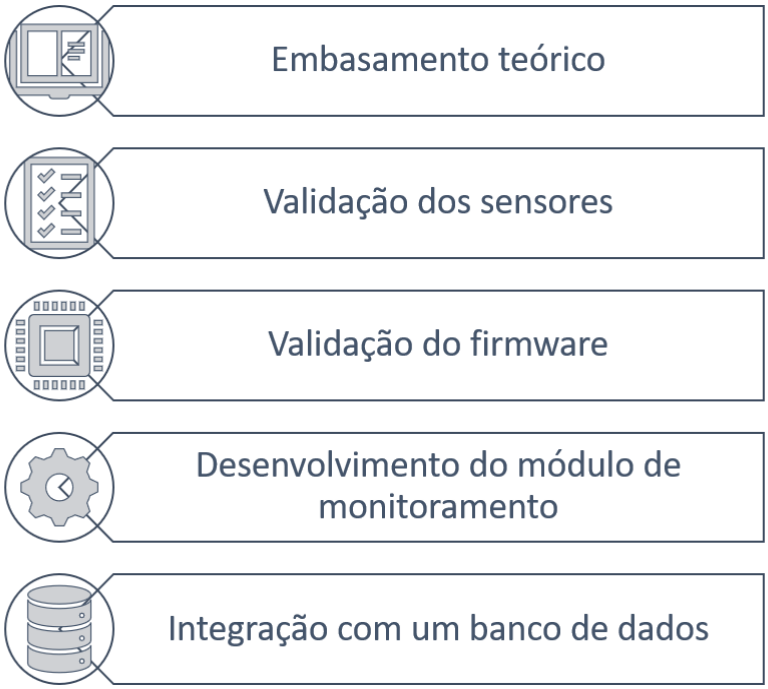
2 MATERIAIS E MÉTODOS

Neste capítulo serão apresentados os materiais utilizados para o desenvolvimento deste protótipo e será abordado o método utilizado durante cada etapa da implementação do projeto.

2.1 MÉTODO PROPOSTO

O método proposto consiste nas seguintes etapas: embasamento teórico para sustentação da pesquisa, validação dos sensores junto a um kit de desenvolvimento, validação do firmware junto a um kit de desenvolvimento, construção do módulo de monitoramento de focos de incêndio e integração a um banco de dados através de um *backend* que utiliza o protocolo MQTT e, por fim, testes de validação final do protótipo de sistema como um todo. As etapas estão exemplificadas na figura 7.

Figura 7 – Diagrama de atividades.



Fonte: Autoria Própria.

Na primeira etapa do desenvolvimento do protótipo, foi realizada uma pesquisa acerca dos problemas regionais do Amazonas e do Brasil, onde chegou-se ao tema das queimadas, após a definição do tema, seguiu-se para a definição de todas as tecnologias que serão utilizadas para o desenvolvimento do protótipo, como por exemplo, o microcontrolador que será utilizado, bem como os sensores que vão fazer parte do protótipo. A escolha do

microcontrolador se deu pelo fato de o ESP32 possuir uma ampla gama de periféricos, alto poder de processamento e bastante conteúdo na internet, a partir disso, foi definido que o kit de desenvolvimento a ser usado no projeto seria o módulo ESP32 LORA HELTEC, um módulo que alia o poder de processamento do ESP32 com a comunicação em longas distâncias do LORA. Em seguida, foram realizadas pesquisas acerca dos sensores a serem utilizados no projeto, onde levou-se em consideração o custo-benefício e as especificações técnicas necessárias para inseri-lo no sistema.

Na segunda etapa, foi desenvolvido um *firmware* de validação dos sensores e das funcionalidades de todo o sistema. Primeiramente, foi desenvolvido um firmware para a leitura I2C do sensor de temperatura, para averiguar se a medição do sensor estava marcando próximo ao de um aparelho de teste universal, após a validação do mesmo, foi feita a validação com o sensor de fumaça, onde o mesmo, em condições normais, mantinha um valor estável. Após ser feita queimada de um material próximo do sensor, a leitura ADC aumenta mostrando que o mesmo está sendo sensibilizado pelo gás monóxido de carbono (CO). Em seguida foram realizados testes com o protocolo MQTT, para o envio da leitura dos sensores, para que haja troca de mensagens, é necessário que o módulo esteja conectado a alguma rede e essa rede possua um broker MQTT, após isso, é necessário se conectar ao broker e enviar e receber mensagens por assinatura e subscrição.

Na terceira etapa, foi desenvolvido o *firmware* que será utilizado no protótipo final, o mesmo foi testado no kit de desenvolvimento associado aos sensores por meio de um protoboard, para facilitar a conexão dos sensores ao microcontrolador. Houve a necessidade de usar o *protoboard* apenas na placa que irá coletar os dados dos sensores, enquanto, na outra placa que envia para o banco de dados não houve a necessidade de usar *protoboard*. O *firmware* foi desenvolvido utilizando o ambiente de desenvolvimento IDE Arduino 1.8.13. Foram desenvolvidos dois scripts, um para a placa que irá coletar dados dos sensores chamada de coletora e o outro para a placa que irá receber essa informação, chamada de concentradora. Para a placa coletora, foram empregadas as bibliotecas de Wire.h, SPI.h, Heltec.h e Adafruit_BMP280.h, as duas últimas possuem funções para o Lora, display e para o sensor de temperatura. Para a placa concentradora, foram empregadas as bibliotecas de WiFi.h, PubSubClient.h e heltec.h, onde a biblioteca de WiFi é para fazer a conexão na rede e a PubSubClient.h é a biblioteca do MQTT. Foi desenvolvido, em seguida, um script em python usando o software de desenvolvimento Visual Studio Code 1.62.3 para receber os dados publicados por MQTT e salvá-los em um banco de dados, para isso, foram utilizadas as bibliotecas paho-mqtt para receber os dados via MQTT e a biblioteca peewee para salvar os

dados em um arquivo SQLite. Na arquitetura do sistema proposto, a placa concentradora será o Publisher, enquanto que o computador executará um script python Subscriber, além de ser o broker do sistema usando o aplicativo Mosquitto.

Na quarta etapa, foi desenvolvido o protótipo de *hardware* da placa coletora de dados, esta placa possui dois reguladores de tensão, sendo um regulador de 3.3V para alimentação do ESP32 e outro para alimentação do sensor de gás com valor de 1.5V, conforme a especificação do fabricante. Após a definição dos circuitos de alimentação e dos demais componentes foi feita a elaboração do esquemático e *layout* da placa de circuito impresso, utilizando a ferramenta EasyEDA versão 6.4.25. Em seguida foi realizada a montagem da placa do sistema de monitoramento, utilizando uma placa de circuitos universal.

2.2 MATERIAIS UTILIZADOS

Para implementação do projeto, foram utilizadas as seguintes ferramentas:

- Interface de desenvolvimento IDE Arduino versão 1.8.13;
- Interface de desenvolvimento Visual Studio Code 1.62.3;
- EasyEDA versão 6.4.25
- Software SQLiteBrowser;
- Estação de solda Hikari, modelo HK-937.

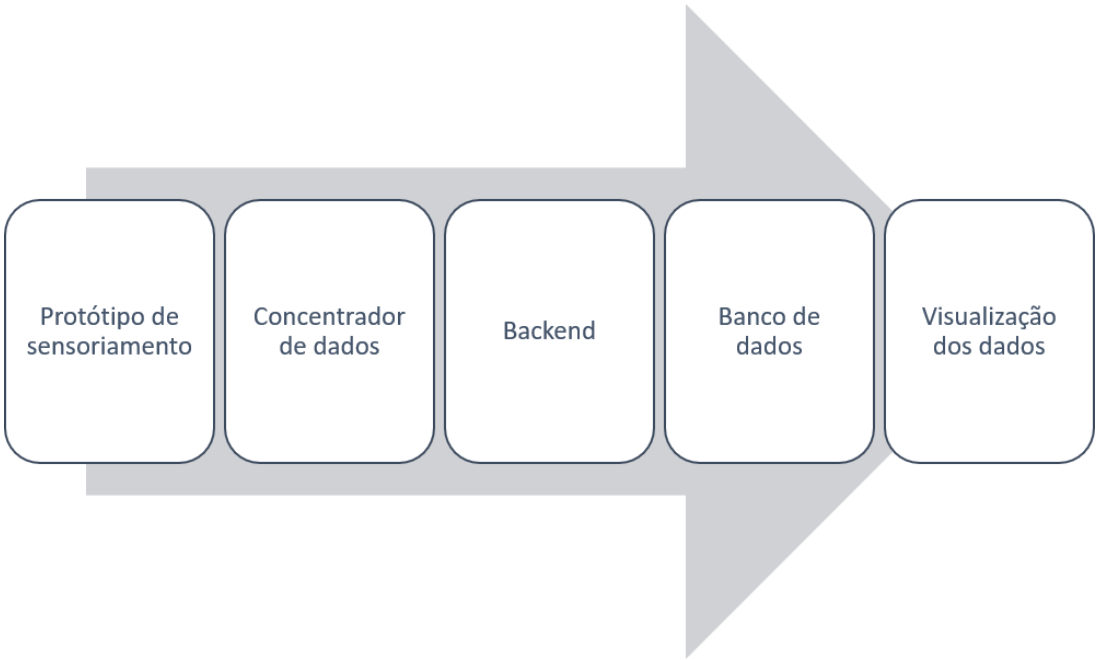
Para a construção desse protótipo de *hardware*, foram utilizados os seguintes componentes:

- 1 sensor de temperatura, modelo BMP280;
- 1 sensor de gás e fumaça, modelo MQ9;
- 1 módulo ESP32;
- 2 circuitos Step Down;
- 1 bateria de 12 volts.

3 IMPLEMENTAÇÃO DO PROJETO

Este capítulo apresenta o procedimento detalhado para o desenvolvimento do protótipo de sistema, que consiste em um Módulo de Monitoramento de focos de incêndio na floresta amazônica dotado de sensores capazes de monitorar parâmetros térmicos e de gás em ambientes de floresta. Os dados coletados serão processados por um microcontrolador ESP32, e será enviado via módulo Lora a um concentrador de dados que utiliza um ESP32, em que o mesmo enviará as mensagens através do protocolo MQTT. Após a mensagem ser enviada via MQTT, um código que simula um backend fará o tratamento dos dados e irá salvá-los em um banco de dados SQLite. A visualização dos dados será feita pelo software SQLite Browser. A figura a 8 ilustra a sequência do sistema proposto, desde o protótipo de sensoriamento até a visualização dos dados.

Figura 8 – Sequência dos módulos do sistema.



Fonte: Autoria Própria

Neste capítulo serão descritas as ferramentas e tecnologias utilizadas para implementação de cada módulo que compõem esse protótipo de sistema. Para facilitar o entendimento do sistema completo, será apresentado os seguintes tópicos:

i. Preparação do ambiente de desenvolvimento;
ii. Sensor de Temperatura;
iii. Sensor de fumaça;

iv. Testes no banco de dados SQLite;
v. Desenvolvimento do módulo de monitoramento de focos de incêndio;
vi. Implementação do firmware, integração com o MQTT e banco de dados; e
vii. Validação do protótipo de sistema.

3.1 PREPARAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

O microcontrolador ESP32 é um dos microcontroladores mais conhecidos da área de sistemas embarcados, devido sua alta capacidade de processamento, capacidade de uso de vários periféricos, conectividade como WiFi e Bluetooth presentes nos seus kits de desenvolvimento. Devido ao sucesso do ESP32, foram surgindo módulos híbridos associando o ESP a alguma outra tecnologia, como por exemplo o módulo da Heltec, que associa o ESP32 a um módulo LoRa para comunicações em longas distâncias. O projeto foi idealizado para realizar medições em longas distâncias, sem que fosse necessária uma infraestrutura de rede complexa ou que demandasse um custo adicional à sua manutenção, por isso, o sistema de comunicação utilizando a tecnologia LoRa foi escolhido e visando a praticidade do desenvolvimento o ESP32 LoRa da Heltec foi outra escolha realizada. Essa placa possui os pinos de entrada e saída externalizados, e dispõe de uma conexão USB e um display OLED. A conexão USB permite uma comunicação serial com o kit da Heltec, possibilitando a programação do módulo através da uart, além de alimentar a placa através do regulador de tensão presente na mesma.

Os sensores utilizados no projeto utilizam o barramento Inter-Integrated Circuit (I2C) sendo uma comunicação serial que utiliza 2 pinos onde um é pino de clock e o outro é o de dados, possuindo um protocolo pré-definido para troca de mensagens, o sensor de temperatura utiliza este barramento que é compartilhado com o Display OLED e o LoRa, também é utilizado a entrada Analog Digital Converter (ADC), por parte do sensor de gás, onde este barramento é responsável por converter grandezas analógicas em grandezas digitais. Deste modo, é apresentado na tabela 1, o sensor, a natureza do pino e o número do pino.

Tabela 1 - Tabela dos pinos utilizados no ESP32 LoRa Heltec.

Sensor	**Natureza do pino**	**Pino**
Sensor de Temperatura	I2C - SDA	4
	I2C - SCL	15
Sensor de Gás	ADC	13

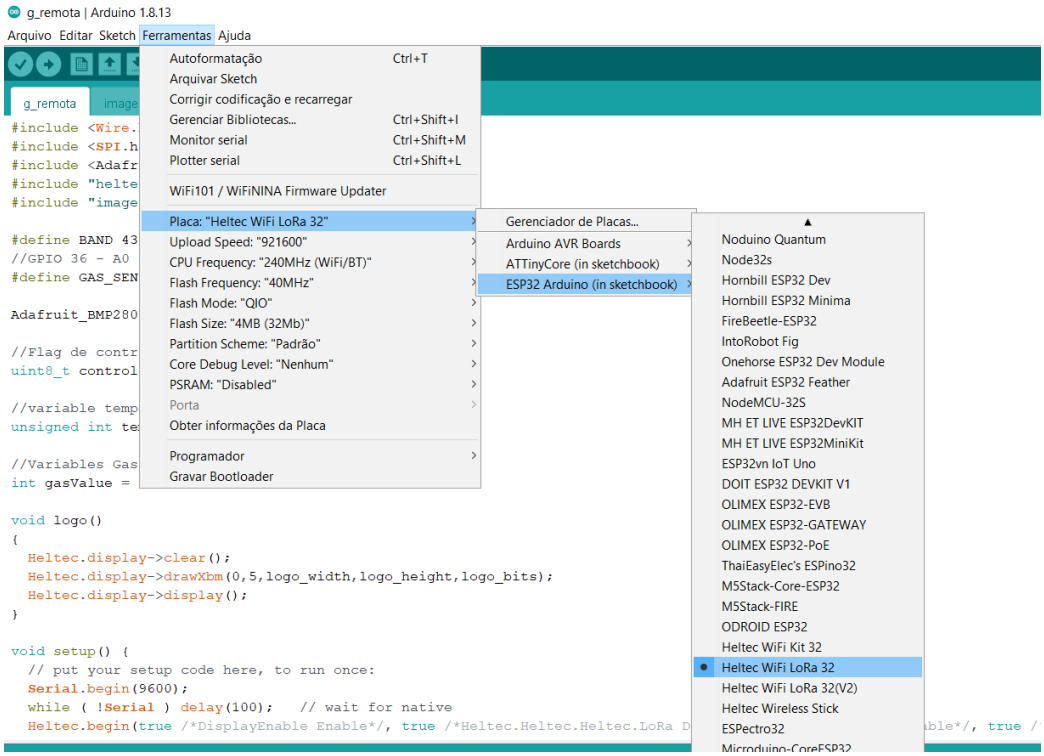
Fonte: Autoria Própria

Para iniciar o desenvolvimento, foi necessário configurar o ambiente de desenvolvimento do Arduino (1.8.13), adicionando manualmente as configurações do SoC, *System On a Chip,* ESP32 na IDE. Isso possibilitou que os códigos fossem embarcados no módulo em questão pois o procedimento a seguir faz o download de arquivos relacionados ao ESP como mapeamento de memória, uso periférico e etc. Foram adotadas as seguintes etapas:

1) Acessar o repositório localizado no endereço https://github.com/espressif/arduino-esp32 e clicar em "*Clone or Download*" e, em seguida, "*download* zip";
2) Um arquivo será salvo em seu diretório *Downloads.* Após abrir o diretório e procurar o arquivo baixado com o nome *"Arduino-esp32-master"*, deve-se descompactar, abrir e copiar seu conteúdo;
3) Em seguida, abrir a pasta Arduino localizada no diretório Documentos; e
4) Dentro dessa pasta, colar o conteúdo copiado.

Após este procedimento, o *software* deve ser reiniciado e ser realizado a escolha da placa "WiFi Lora Heltec" clicando em ferramentas e, em seguida, em "Placa", conforme ilustrado na figura 9.

Figura 9 – Configuração do ambiente de desenvolvimento.



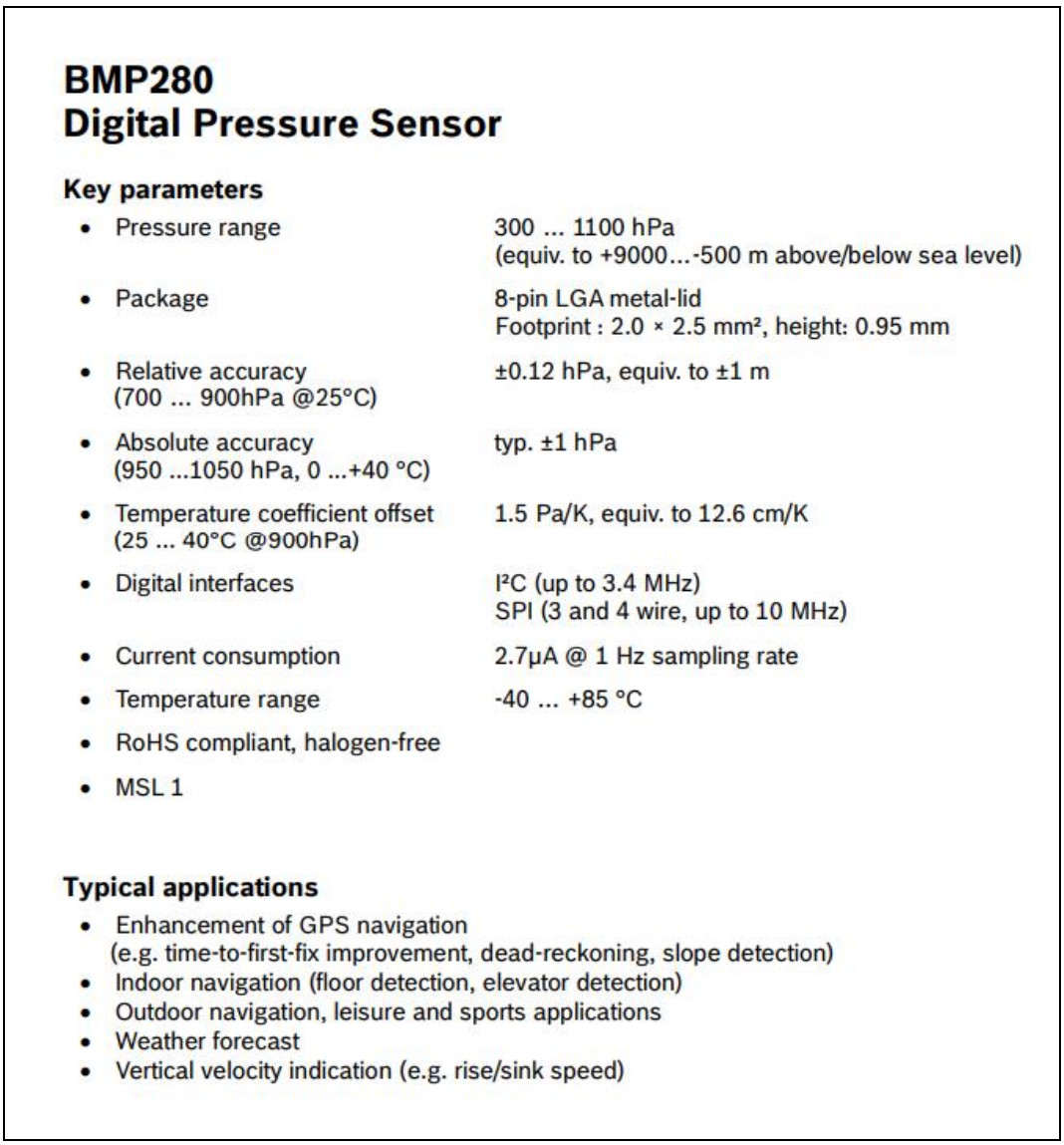
Fonte: Autoria Própria.

Foi necessário também, incluir algumas bibliotecas específicas módulo ESP32 LoRa. Essas bibliotecas gerenciam o uso do display embarcado no módulo e do próprio LoRa. Após isso, o ambiente de desenvolvimento estava pronto para implementar os códigos a serem embarcados no microcontrolador para realização de testes.

3.2 SENSOR DE TEMPERATURA E SENSOR DE GÁS

O sensor de temperatura utilizado neste protótipo de sistema, foi sensor da Bosch BMP280 utilizado para fazer aferições de temperatura e pressão. Devido as suas características como pequenas dimensões e baixo consumo de energia tornam possível a sua implementação em dispositivos alimentados por bateria. O sensor BMP 280 é um sensor extremamente robusto, possuindo uma faixa de operação de -40°C à 80°C com um consumo médio de corrente (a taxa de 1 Hz) de 3,4uA e possuindo I2C e SPI como sua interface de comunicação serial. A figura 10 apresenta a folha de dados do sensor de temperatura.

Figura 10 – Dados do sensor BMP280.



BMP280
Digital Pressure Sensor

Key parameters

• Pressure range	300 ... 1100 hPa (equiv. to +9000...-500 m above/below sea level)
• Package	8-pin LGA metal-lid Footprint : 2.0 × 2.5 mm², height: 0.95 mm
• Relative accuracy (700 ... 900hPa @25°C)	±0.12 hPa, equiv. to ±1 m
• Absolute accuracy (950 ...1050 hPa, 0 ...+40 °C)	typ. ±1 hPa
• Temperature coefficient offset (25 ... 40°C @900hPa)	1.5 Pa/K, equiv. to 12.6 cm/K
• Digital interfaces	I²C (up to 3.4 MHz) SPI (3 and 4 wire, up to 10 MHz)
• Current consumption	2.7µA @ 1 Hz sampling rate
• Temperature range	-40 ... +85 °C
• RoHS compliant, halogen-free	
• MSL 1	

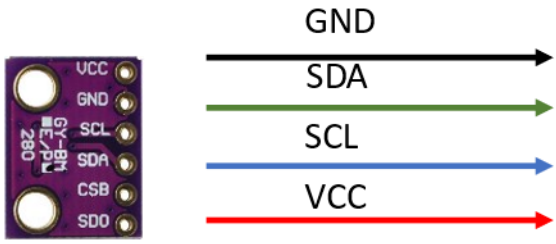
Typical applications

- Enhancement of GPS navigation
  (e.g. time-to-first-fix improvement, dead-reckoning, slope detection)
- Indoor navigation (floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Vertical velocity indication (e.g. rise/sink speed)

Fonte: (Bosch, 2021, p.2)

O sensor BMP280 se comunica através do barramento I2C, e para ocorrer a comunicação com este sensor é necessário a conexão de quatro fios para ligação: a primeira ligação é a que está na cor preta (GND) para se ter um pino de referência em comum, o segundo é o que está na cor verde (SDA) onde o mesmo é o pino de dados sendo responsável pelo pela transmissão de dados, o terceiro é o que está na cor azul (SCL) onde o mesmo é o relógio sendo responsável pela sincronização dos dados a serem enviados e por último, em vermelho, a alimentação (VCC) que energiza o sensor. A figura 11 orienta acerca da conexão nos pinos do sensor.

Figura 11 – Conexões do sensor BMP280.



Fonte: Autoria própria.

Para realizar testes no sensor, foi utilizado um *firmware* que apenas realiza a leitura dos parâmetros de temperatura e pressão e os imprime na serial. Esse *firmware* foi desenvolvido com o auxílio da biblioteca da "Adafruit", onde a mesma visa abstrair várias configurações complexas do sensor. Para utilizar essa biblioteca é necessário realizar o download do seu código fonte no repositório do github, incluí-la nas bibliotecas da IDE do Arduino e fazer a chamada de inclusão no código dessa forma <Adafruit_BMP280.h>. A figura 12 exemplifica como ficou o código de teste do sensor BMP280.

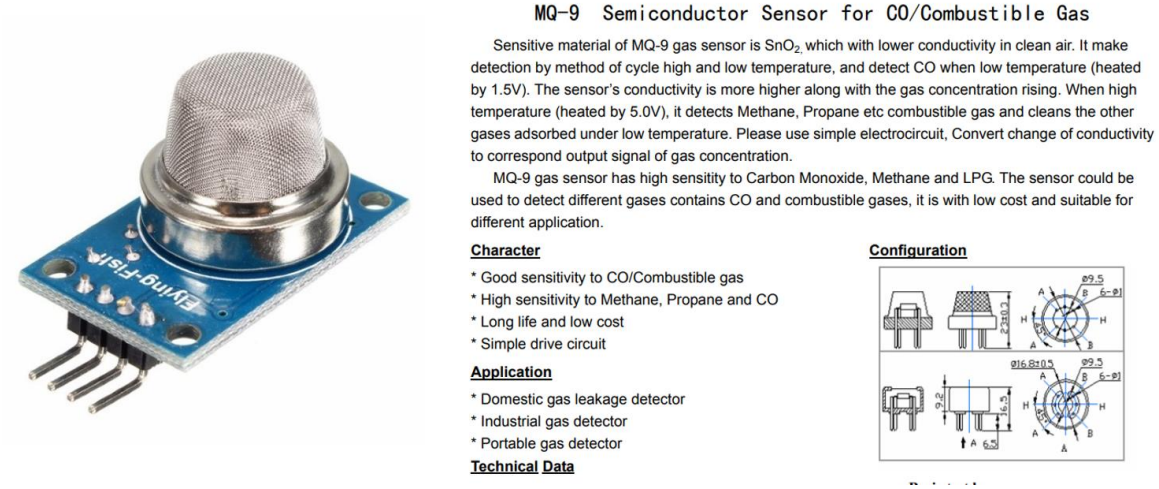
Figura 12 – Firmware de teste do sensor de temperatura.

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>

Adafruit_BMP280 bmp; // I2C


void setup() {
  Serial.begin(9600);
  while ( !Serial ) delay(100);   // wait for native usb
  Serial.println(F("BMP280 test"));
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
    while (1) delay(10);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
}

void loop() {
    Serial.print(F("Temperature = "));
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

    Serial.print(F("Pressure = "));
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");
```

Fonte: Autoria própria.

Além do sensor de temperatura, foi utilizado o sensor de gás MQ-9, no qual é realizado uma leitura analógica para coletar os valores do sensor. Observando o *datasheet* do mesmo, percebe-se que é necessário energizar o sensor com 1.5V para que ele seja sensibilizado por monóxido de carbono (CO) que é o gás liberado durante uma queimada. O *datasheet* também informa que é necessário um período de pré-aquecimento do sensor, que dura cerca de 15 minutos. A figura 13 apresenta as características mecânicas do sensor de gás e a sua folha de dados.

Figura 13 – Sensor MQ9 e sua folha de dados.



MQ-9 Semiconductor Sensor for CO/Combustible Gas

Sensitive material of MQ-9 gas sensor is SnO_2, which with lower conductivity in clean air. It make detection by method of cycle high and low temperature, and detect CO when low temperature (heated by 1.5V). The sensor's conductivity is more higher along with the gas concentration rising. When high temperature (heated by 5.0V), it detects Methane, Propane etc combustible gas and cleans the other gases adsorbed under low temperature. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-9 gas sensor has high sensitity to Carbon Monoxide, Methane and LPG. The sensor could be used to detect different gases contains CO and combustible gases, it is with low cost and suitable for different application.

Character

* Good sensitivity to CO/Combustible gas
* High sensitivity to Methane, Propane and CO
* Long life and low cost
* Simple drive circuit

Application

* Domestic gas leakage detector
* Industrial gas detector
* Portable gas detector

Technical Data

Configuration

Fonte: Autoria Própria.

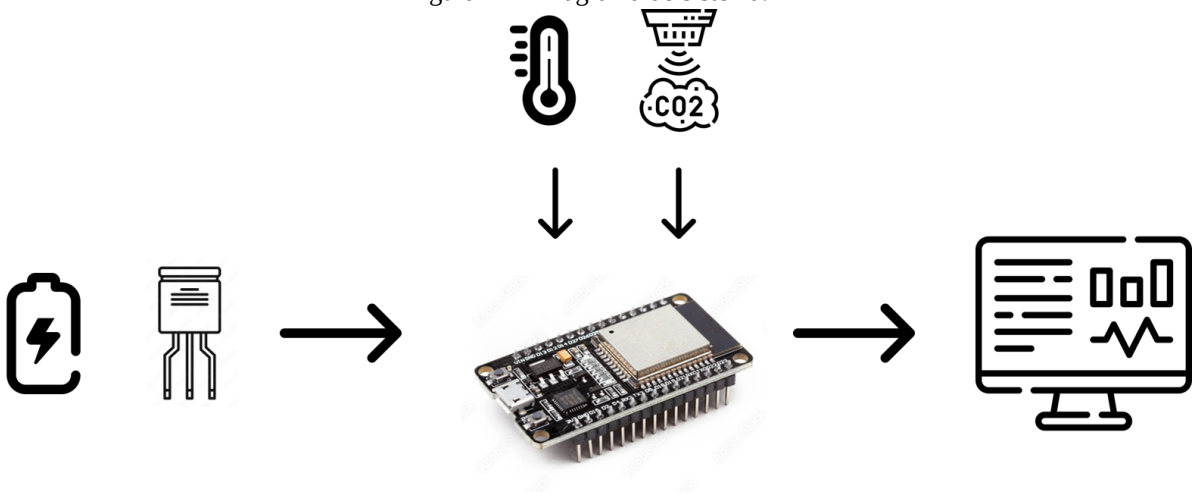
3.3 DESENVOLVIMENTO DO MÓDULO DE DETECÇÃO DE FOCOS DE INCÊNDIO

Nesta etapa, será descrito o desenvolvimento do *hardware* do dispositivo, detalhando a arquitetura do módulo, a elaboração do esquemático e *layout*, a prototipação e a montagem.

3.3.1 Elaboração da arquitetura

O protótipo de sistema desenvolvido possui um módulo de monitoramento de focos de incêndio, responsável pela aferição, processamento e envio dos dados de entrada do sistema. Esse módulo é constituído pelo circuito regulador de tensão, pinagem do ESP32-Lora e conectores dos sensores de temperatura e fumaça. Na figura 14, é apresentado um diagrama da arquitetura de *hardware*.

Figura 14 – Diagrama do sistema.



Fonte: Autoria Própria.

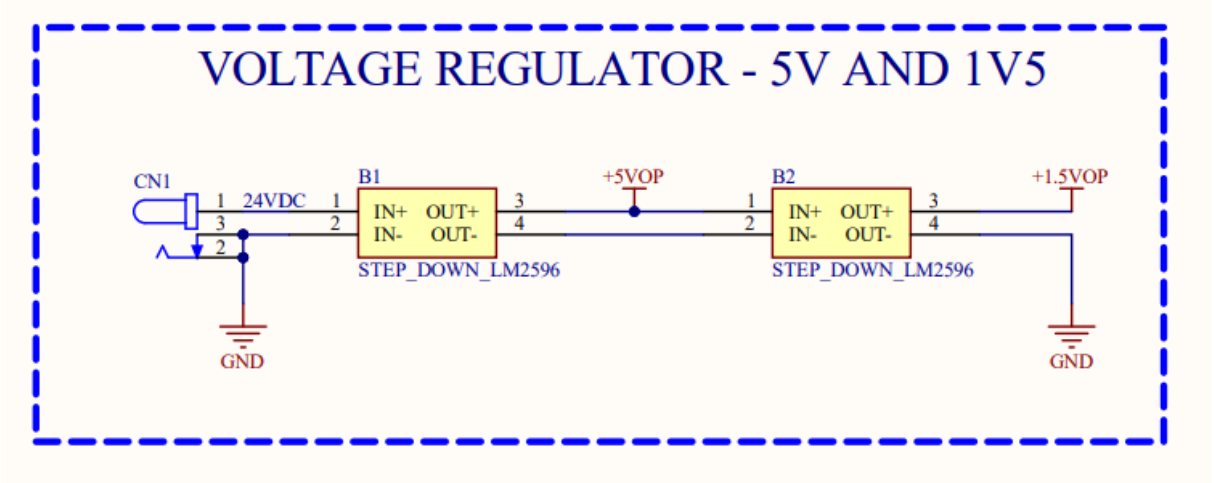
3.3.2 Elaboração do esquemático e layout

Após a definição da arquitetura, iniciou-se o desenvolvimento do esquemático, utilizando a ferramenta Altium Designer v.21.

A placa foi projetada para ser utilizada com a alimentação de uma bateria de 12 volts, e o módulo ESP32 Lora necessita de uma alimentação de 3,3 volts tornando necessário o uso de um regulador de tensão que converta 12V em 3,3V.

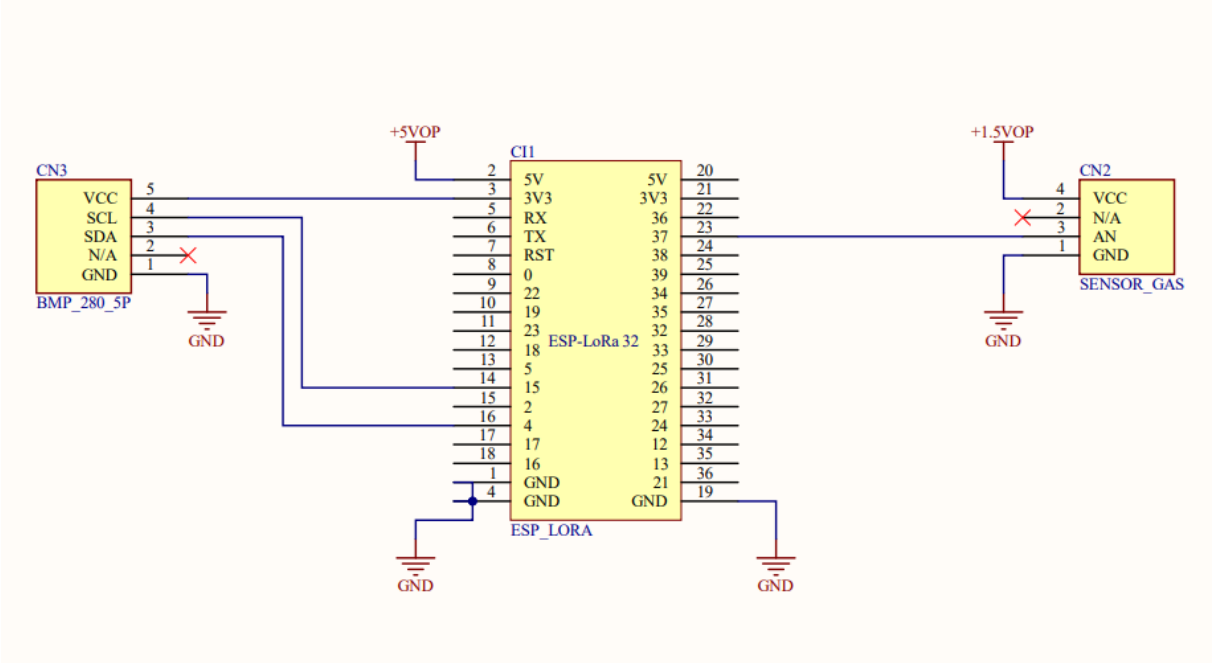
O módulo de detecção de focos de incêndio possui um sensor de gás e fumaça (MQ-9) onde sua alimentação para detectar dióxido de carbono deve ser de 1,5V, logo é necessário o uso de mais um regulador de tensão para alimentar adequadamente este sensor. Após o levantamento dessas informações, começou-se o desenvolvimento do esquemático do módulo, onde, a figura 15 contém os circuitos reguladores de tensão e a figura 16 a conexão com os sensores de temperatura BMP280 e o sensor de fumaça MQ9.

Figura 15 – Regulador de tensão.



Fonte: Autoria Própria.

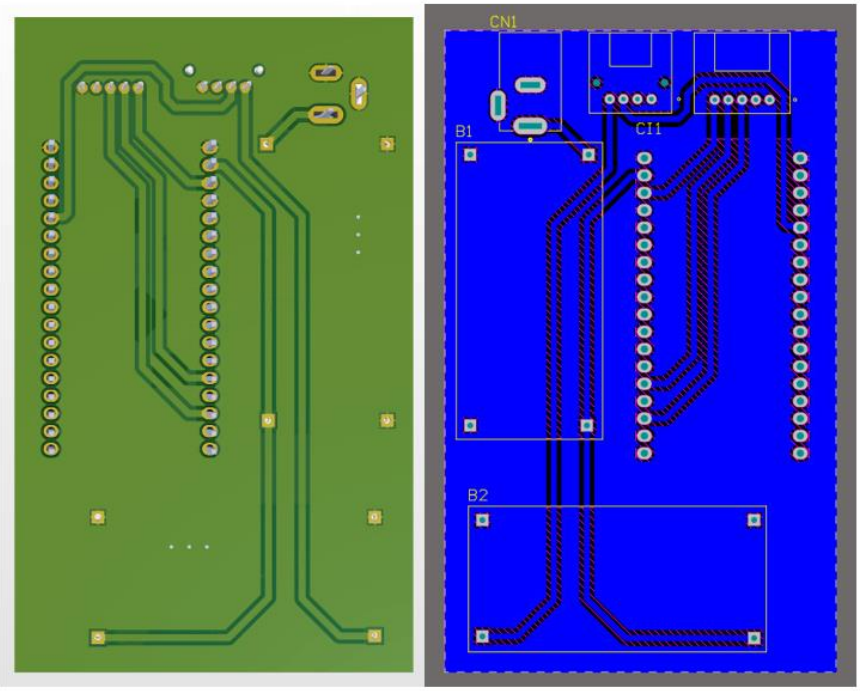
Figura 16 – Conexão do ESP Lora nos sensores



Fonte: Autoria Própria.

A partir do desenvolvimento do esquemático do *hardware* foi realizado o desenvolvimento do *layout* utilizando a regras de configuração do roteamento, dimensionamento das bordas da PCI, criação dos arquivos de fabricação e exportação dos mesmos. O *layout* do módulo é apresentado na figura 17.

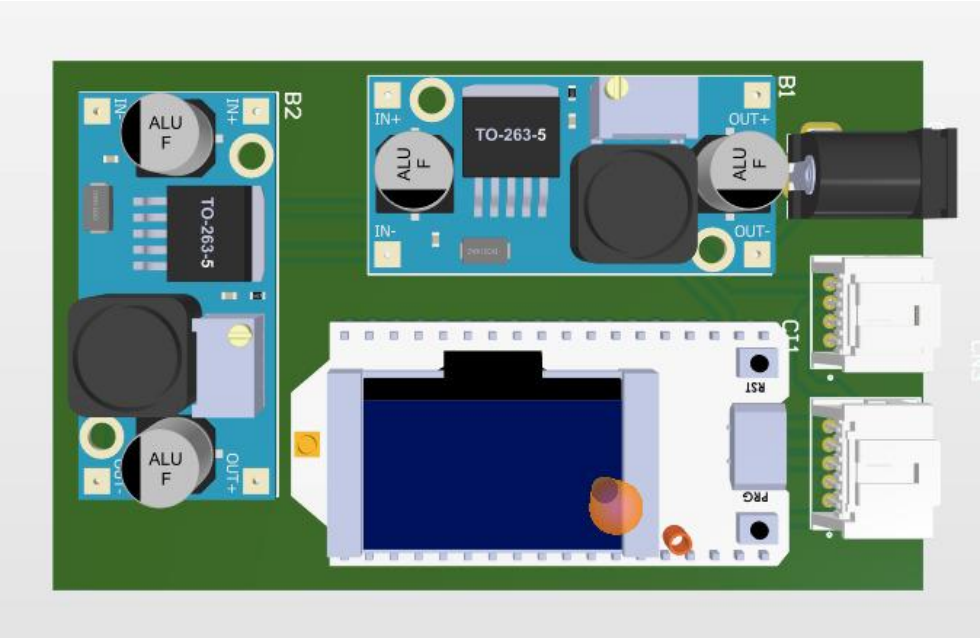
Figura 17 – *Layout* do módulo de detecção de focos de incêndio.



Fonte: Autoria Própria.

Após o desenvolvimento do esquemático e ser realizado o roteamento do protótipo, foi desenvolvido um esquema 3D para facilitar a visualização do dispositivo final, conforme apresenta a figura 18.

Figura 18 – Esquema em 3D

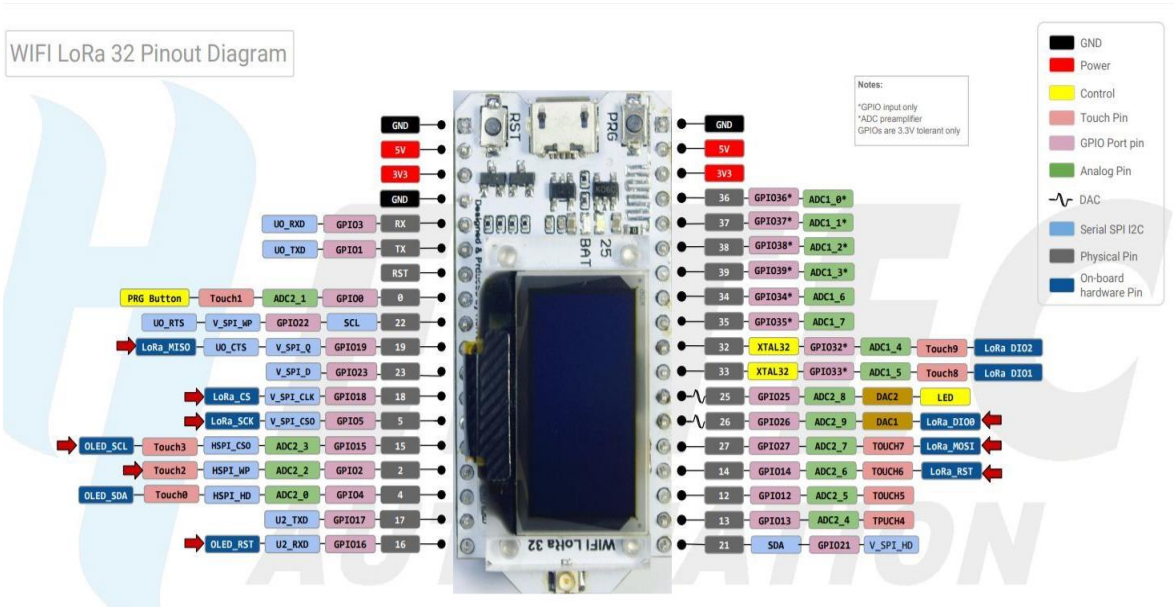


Fonte: Autoria Própria

3.4 IMPLEMENTAÇÃO DO FIRMWARE DO SISTEMA

O sistema possui um *firmware* para coletar os dados dos sensores, onde o mesmo realizará uma leitura através do barramento I2C para obter os dados de temperatura, em seguida realizar uma leitura analógica para realizar a leitura de monóxido de carbono, esses dados poderão ser visualizados através de um display OLED disponível no módulo ESP Lora e em seguida serão enviados via Lora a um outro dispositivo que será o concentrador, este também é um ESP32 Lora. Para realizar a leitura do BMP280 que é o sensor de temperatura, através do barramento i2c, foi utilizado a biblioteca "Adafruit_BMP280.h" onde a mesma configura o barramento i2c e realiza a leitura do valor da temperatura em graus celsius. Para a implementação comunicação por Lora, foi utilizada a biblioteca do próprio fabricante do módulo a "heltec.h". Porém, para que o dado lido no sensor de temperatura fosse visualizado foram necessárias algumas mudanças no mapeamento dos pinos para que ambos estivessem configurados no pino correto do barramento i2c, a figuram 19 ilustra o *pinout* do ESP32 Lora, também conhecido como Wifi Lora 32.

Figura 19 – Diagrama dos pinos do módulo Wifi Lora 32.



Fonte: (Instructables, 2022, p.1)

Conforme a figura 19 é possível visualizar que há 2 pares de pinos que podem ser utilizados no barramento i2c, 21/22 e 4/15, dependendo apenas de um cabeçalho que realiza o mapeamento dos pinos. No entanto, o display OLED está conectado está conectado nos pinos 4 e 15 foi necessário realizar essa mudança dentro do diretório

“\Documents\Arduino\hardware\espressif\esp32\variants\heltec_wifi_lora_32" no arquivo “pins_arduino.h”. Conforme ilustrado na figura 20 foi alterado o valor de SDA, que é o pino da linha de dados e o SCL que é o pino da linha de *clock*.

Figura 20 - Mudança nos pinos do barramento i2c.

```
static const uint8_t LED_BUILTIN = 25;
#define BUILTIN_LED  LED_BUILTIN // backward compatibility

static const uint8_t KEY_BUILTIN = 0;

static const uint8_t TX = 1;
static const uint8_t RX = 3;

//static const uint8_t SDA = 21;
//static const uint8_t SCL = 22;

static const uint8_t SDA = 4;
static const uint8_t SCL = 15;
```

Fonte: Autoria Própria.

Após essa configuração, o display OLED funcionou utilizando os novos pinos, e em seguida foi realizado uma leitura, utilizando o mesmo barramento I2C, no sensor de temperatura BMP280. A próxima etapa será passar informação lida do sensor para o display OLED, ou seja, a estrutura do firmware estava sendo montada. Após a informação do sensor BMP280 ser mostrada através do display, era necessário enviar essa informação para um outro dispositivo Lora que seria chamado de concentrador, pois iria receber a informação. Para isso, foi necessário incluir a biblioteca do próprio módulo, chamada de “heltec.h”, onde a mesma inicia as configurações do Lora através do método de inicialização. Em seguida é realizada uma leitura ADC do sensor de gás e fumaça para ser incluída no pacote de dados que será enviado para o concentrador através do módulo Lora. A seguir será apresentado o firmware utilizado para coletar esses dados, onde a figura 21 apresentará todas as bibliotecas que foram inseridas para o desenvolvimento deste código, além de mostrar as definições, variáveis globais e a função que mostra a logo da universidade quando o sistema inicializa.

Figura 21 – Firmware da placa remota.

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>
#include "heltec.h"
#include "images.h"

#define BAND 433E6
//GPIO 36 - A0
#define GAS_SENSOR A0

Adafruit_BMP280 bmp;

//Flag de controle
uint8_t controlFlag = 0;

//variable temperatura
unsigned int temp = 0;

//Variables Gas
int gasValue = 0;

void logo()
{
  Heltec.display->clear();
  Heltec.display->drawXbm(0,5,logo_width,logo_height,logo_bits);
  Heltec.display->display();
}
```

Fonte: Autoria Própria.

A figura 22 apresenta a imagem que contém a função de setup do *firmware*, que contém as funções de configuração da “Serial”, inicialização da biblioteca heltec que contém a inicialização do módulo Lora, além da configuração do sensor BMP280. Na função ainda há a chamada da função “logo” que mostra a logo da universidade no display OLED.

Figura 22 – Firmware da placa remota.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  while ( !Serial ) delay(100);   // wait for native
  Heltec.begin(true /*DisplayEnable Enable*/, true /*Heltec.Heltec.Heltec.LoRa Disable*/,
               false /*Serial Enable*/, true /*PABOOST Enable*/, BAND /*long BAND*/);

  Heltec.display->init();
  Heltec.display->flipScreenVertically();
  Heltec.display->setFont(ArialMT_Plain_10);
  logo();
  delay(1500);
  Heltec.display->clear();

  Heltec.display->drawString(0, 0, "Heltec.LoRa Initial success!");
  Heltec.display->display();

  if (!bmp.begin()) {
    Heltec.display->clear();
    Heltec.display->drawString(0, 0, "Check wiring!");
    Heltec.display->display();

    while (1) delay(10);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

  delay(1000);
}
```

Fonte: Autoria Própria.

Em seguida, será apresentado a função “loop” do sistema que, como sua tradução irá ficar se repetindo em um ciclo de 1.5 segundos, onde é coletado os dados de temperatura e a leitura do ADC do sensor de gás, em seguida o pacote de dados é montado e enviado por Lora, como mostra a figura 23.

Figura 23 - *Firmware* da placa remota.

```
void loop() {
  Heltec.display->clear();
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);
  Heltec.display->setFont(ArialMT_Plain_10);

  if (controlFlag == 0){
    temp = bmp.readTemperature();

    Heltec.display->drawString(0, 0, "Sending temperature: ");
    Heltec.display->drawString(102, 0, String(temp));
    Heltec.display->drawString(115, 0, "*C");

    Heltec.display->display();

    LoRa.beginPacket();
    LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);
    LoRa.print("T: ");
    LoRa.print(temp);
    LoRa.endPacket();

    controlFlag = 1;
  }
  else if (controlFlag == 1){

    gasValue = (analogRead(GAS_SENSOR));

    Heltec.display->drawString(0, 0, "Sending Gas: ");
    Heltec.display->drawString(85, 0, String(gasValue));
    Heltec.display->drawString(100, 0, " adc");

    Heltec.display->display();

    LoRa.beginPacket();
    LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);

    LoRa.print("G: ");
    LoRa.print(gasValue);
    LoRa.endPacket();
    controlFlag = 0;
  }

  delay(1500);
}
```

Fonte: Autoria própria.

O *firmware* da placa remota, que coleta os dados foi apresentado na figura 23, e na figura 24, será apresentado e abordado o *firmware* da placa concentradora que vai receber essas informações e enviá-las por MQTT para ser salvo em um banco de dados. Primeiramente será exibido através da imagem abaixo, todas as bibliotecas que serão utilizadas, bem como as variáveis globais e definições, também temos as funções de inicialização do wifi (init_wifi) e inicialização do mqtt (init_mqtt).

Figura 24 – Firmware da placa concentradora.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "heltec.h"
#include "images.h"

#define BAND    433E6  //you can set band here directly,e.g. 868E6,915E6
#define PUB "teste"
#define SUB "teste_SUB"
#define ID_MQTT "TCC"

const char* SSID = "VIVOFIBRA-6D30";
const char* PASSWORD = "4E2A9711BD";
const char* BROKER_MQTT = "192.168.15.150";
int BROKER_PORT = 1883;

String rssi = "RSSI --";
String packSize = "--";
String packet ;

WiFiClient espClient;
PubSubClient MQTT(espClient);

void init_wifi(void)
{
    delay(10);
    Serial.println("------Conexao WI-FI------");
    Serial.print("Conectando-se na rede: ");
    Serial.println(SSID);
    Serial.println("Aguarde");
    reconnect_wifi();
}

void init_mqtt(void)
{
    /* informa a qual broker e porta deve ser conectado */
    MQTT.setServer(BROKER_MQTT, BROKER_PORT);
    //Funcao chamada se topico subscrito chegar
    MQTT.setCallback(mqtt_callback);
    reconnect_mqtt();
}
```

Fonte: Autoria própria.

A seguir, a figura 25 apresentará as funções de call-back do mqtt (mqtt_callback), que foi cadastrado na função de inicialização ilustrada acima, além da função de reconexão do mqtt, que é chamada quando a conexão com o mqtt é perdida.

Figura 25 – Firmware da placa concentradora.

```
void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
    String msg;

    //obtem a string do payload recebido
    for(int i = 0; i < length; i++)
    {
        char c = (char)payload[i];
        msg += c;
    }
    Serial.print("[MQTT] Mensagem recebida: ");
    Serial.println(msg);
}

void reconnect_mqtt(void)
{
    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao Broker MQTT: ");
        Serial.println(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT))
        {
            Serial.println("Conectado com sucesso ao broker MQTT!");
            MQTT.subscribe(SUB);
        }
        else
        {
            Serial.println("Falha ao reconectar no broker.");
            Serial.println("Havera nova tentatica de conexao em 2s");
            delay(2000);
        }
    }
}
```

Fonte: Autoria própria.

O *firmware* da placa concentradora também possui uma função para reconectar o wifi (reconnect_wifi), uma função para mostrar a logo da UEA na inicialização do sistema (logo) e uma função para exibir no display o que chegou na função de call-back do lora (LoRaData). A figura 26 é a parte do *firmware* ora citado.

Figura 26 – Firmware da placa concentradora.

```
void reconnect_wifi()
{
    //tentiva de reconectar no wifi
    if (WiFi.status() == WL_CONNECTED)
        return;

    WiFi.begin(SSID, PASSWORD);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Conectado com sucesso na rede ");
    Serial.print(SSID);
    Serial.println("IP obtido: ");
    Serial.println(WiFi.localIP());
}

void logo(){
  Heltec.display->clear();
  Heltec.display->drawXbm(0,5,logo_width,logo_height,logo_bits);
  Heltec.display->display();
}

void LoRaData(){
  Heltec.display->clear();
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);
  Heltec.display->setFont(ArialMT_Plain_10);
  Heltec.display->drawString(0 , 15 , "Received "+ packSize + " bytes");
  Heltec.display->drawStringMaxWidth(0 , 26 , 128, packet);
  Heltec.display->drawString(0, 0, rssi);
  Heltec.display->display();
}
```

Fonte: Autoria própria.

Em seguida, será apresentada a figura 27 que contém a função de call-back do Lora (cbk) onde, esta função possui toda a tratativa do dado que está chegando na placa concentradora. Após a definição de qual informação chegou na placa concentradora e, após isso, a informação é enviada por mqtt, no formato *JavaScript Object Notation*, popularmente conhecido como JSON.

Figura 27 – Firmware da placa concentradora

```
void cbk(int packetSize) {
  packet ="";
  packSize = String(packetSize,DEC);
  for (int i = 0; i < packetSize; i++) { packet += (char) LoRa.read(); }
  rssi = "RSSI " + String(LoRa.packetRssi(), DEC) ;
  LoRaData();
  Serial.println("Variaveis recebidas");
  String msgC = "";
  String compare = "";
  compare = packet[0];
  for(int i = 3; i < packet.length(); i++)  msgC += packet[i];
  if (compare == "T"){
    String bufferTemp = "{\"Temp\":\""+msgC+"\"}";
    Serial.println(bufferTemp);
    MQTT.publish(PUB, bufferTemp.c_str());
  }
  else if (compare == "G"){
    String bufferGas = "{\"Gas\":\""+msgC+"\"}";
    Serial.println(bufferGas);
    MQTT.publish(PUB, bufferGas.c_str());
  }
}
```

Fonte: Autoria própria.

Na sequência, para finalizar o código da placa concentradora será apresentada a figura 28 que contém a função de configuração do *firmware* (setup), onde é feito a configuração do lora, inicialização do wifi e mqtt, ambas funções apresentadas nas figuras anteriores. A função de execução a cada intervalo de tempo (loop) também é apresentada na figura 28.

Figura 28 – Firmware da placa concentradora.

```
void setup() {
  //WIFI Kit series V1 not support Vext control
  Serial.begin(9600);
  init_wifi();
  Serial.println("Chamando funcao do MQTT");
  init_mqtt();

  Heltec.begin(true /*DisplayEnable Enable*/, true /*Heltec.Heltec.Heltec.LoRa Disable*/,
                 false /*Serial Enable*/, true /*PABOOST Enable*/, BAND /*long BAND*/);


  Heltec.display->init();
  Heltec.display->flipScreenVertically();
  Heltec.display->setFont(ArialMT_Plain_10);
  logo();
  delay(1500);
  Heltec.display->clear();

  Heltec.display->drawString(0, 0, "Heltec.LoRa Initial success!");
  Heltec.display->drawString(0, 10, "Wait for incoming data...");
  Heltec.display->display();
  delay(1000);
  //LoRa.onReceive(cbk);
  LoRa.receive();
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) { cbk(packetSize);  }
  MQTT.loop();
  delay(10);
}
```

Fonte: Autoria própria.

Para complementar o sistema, foi necessário o desenvolvimento de um *software* para receber as mensagens através do protocolo mqtt, que em seguida seriam armazenadas em um banco de dados SQlite. A figura 29 é o resultado final do desenvolvimento do *software* utilizado pelo sistema de monitoramento de focos de incêndio na floresta.

Figura 29 – *Software* para salvar no banco de dados.

```
def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        #print(f"Received `{msg.payload.decode()}` from `{msg.topic}` topic")
        mensagem = ""
        mensagem = json.loads(msg.payload.decode())
        for key in mensagem:
            if key == "Temp":
                GuardiaoTemp.create(sensor_temp="temperature", value_temp=int(mensagem["Temp"]), join_date_temp=datetime.utcnow().isoformat())
                dataDB_temp["Temperatura"] = mensagem["Temp"]
                dataDB_temp["Date"] = datetime.utcnow().isoformat()
                print(dataDB_temp)

            elif key == "Gas":
                GuardiaoGas.create(sensor_gas="gas", value_gas=int(mensagem["Gas"]), join_date_gas=datetime.utcnow().isoformat())
                dataDB_gas["Gas"] = mensagem["Gas"]
                dataDB_gas["Date"] = datetime.utcnow().isoformat()
                print(dataDB_gas)


    client.subscribe(topic)
    client.on_message = on_message

def run():
    client = connect_mqtt()
    subscribe(client)
    #client.loop_start()
    client.loop_forever()

if __name__ == '__main__':
    try:
        GuardiaoTemp.create_table()
    except peewee.OperationalError:
        print('A Tabela de Temperatura existe')

    try:
        GuardiaoGas.create_table()
    except peewee.OperationalError:
        print("A Tabela de Gas existe")

    run()
```

Fonte: Autoria própria.

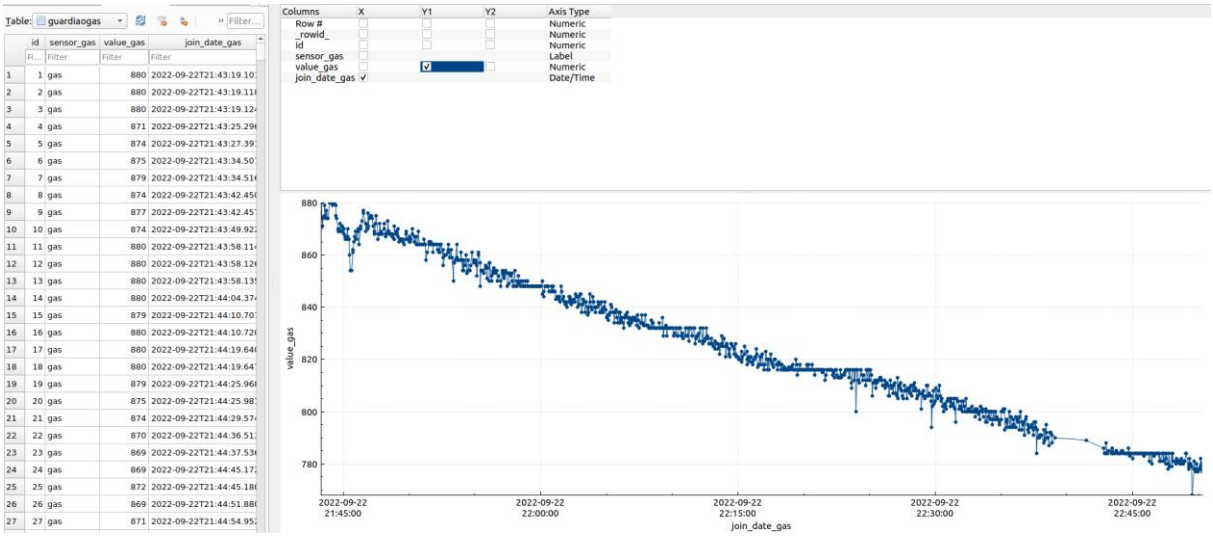
Dessa forma, após tudo ser inicializado, o sistema é capaz de atuar, coletar os dados, executando todas as suas rotinas e em seguida salvar as informações em banco de dados SQLite. Para os dados serem visualizados é necessário o uso do banco de dados SQLite Browser.

4 RESULTADOS OBTIDOS

Este capítulo mostra os resultados obtidos após a implementação de todas as etapas do projeto. Serão apresentados os resultados pertinentes ao desenvolvimento do módulo de monitoramento de focos de incêndio na floresta.

Os testes foram realizados primeiramente em ambiente de laboratório, visando validar a comunicação e integração do sistema, além da validação do protótipo em si. As imagens abaixo são os dados gerados durante o tempo de teste, onde são apresentados os dados de temperatura e dados de fumaça. A figura 30 representa o os dados do sensor de fumaça.

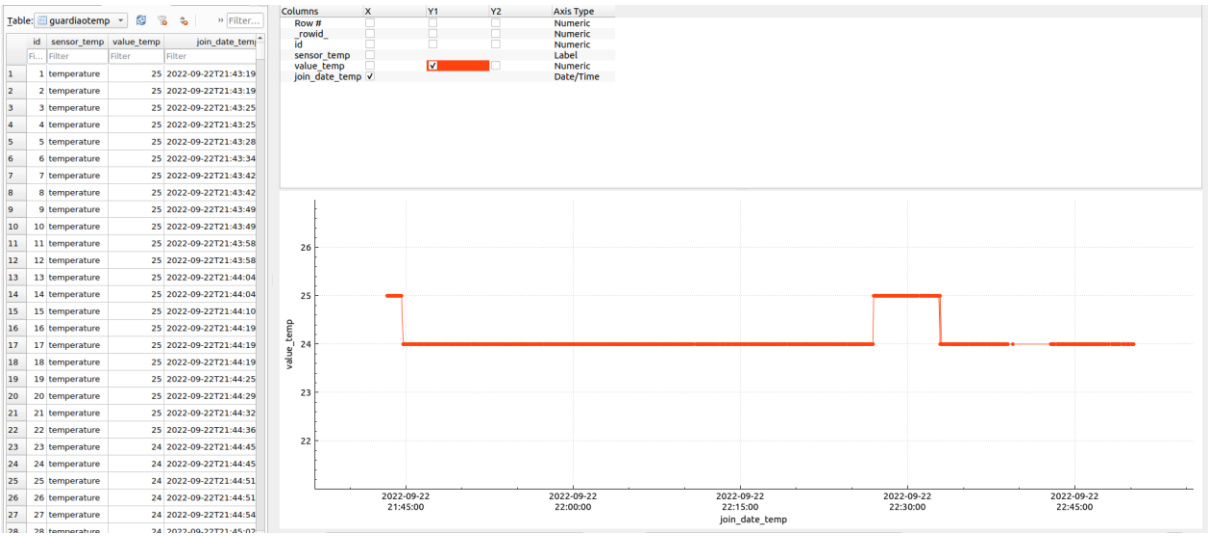
Figura 30 – Banco de dados do sensor de fumaça.



Fonte: Autoria própria.

O sensor de fumaça é uma leitura analógica e possui um certo atraso para se estabilizar, os dados acima foram obtidos dentro do laboratório após uma fumaça controlada nas dependências da universidade, onde valores que o sensor apresentava eram da ordem 1500, lembrando que a leitura ADC do ESP32 é 12 bits logo seus valores variam de 0 a 4096, durante o teste realizado no laboratório, o resultado apresentou uma curva de descida, tendo como valor inicial 850 e final 750. A figura 31 apresenta os dados obtidos do sensor de temperatura.

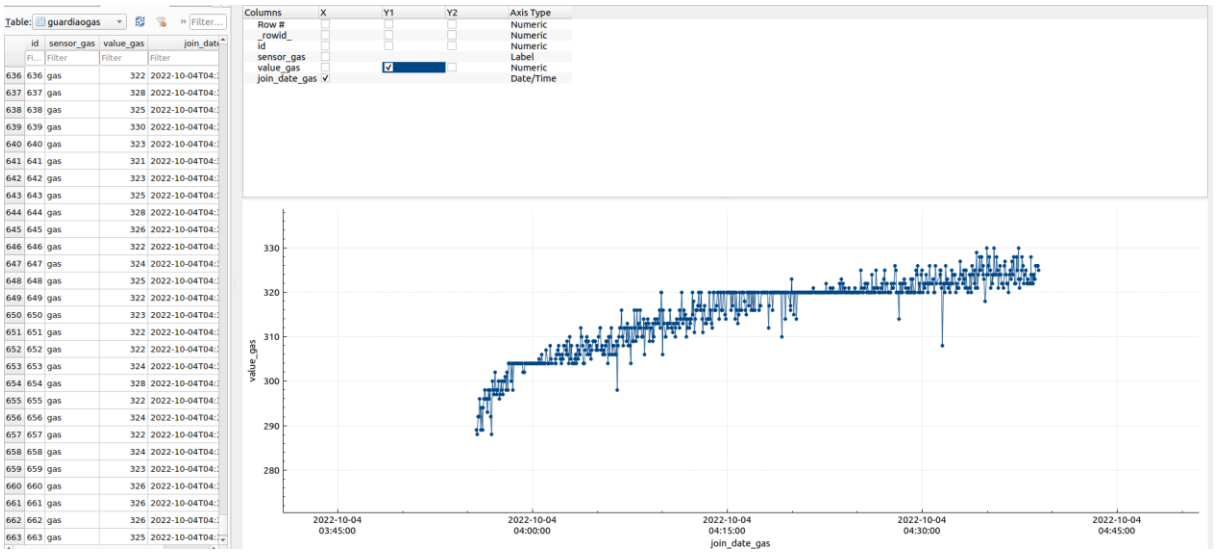
Figura 31 – Banco de dados do sensor de temperatura.



Fonte: Autoria Própria.

Foram realizados testes no período noturno, para avaliar o funcionamento do dispositivo em ambiente de floresta, onde foram captados 30 minutos de dados e concluiu-se que obtivemos um padrão nos dados que começaram a ser coletados dentro do ambiente residencial e foi finalizado dentro de uma área de preservação permanente localizada no planalto, na zona oeste de Manaus. Os dados a seguir mostram essa coleta de dados, tanto de temperatura quanto de leitura do sensor de gás. A figura 32 ilustra os novos dados do sensor de fumaça.

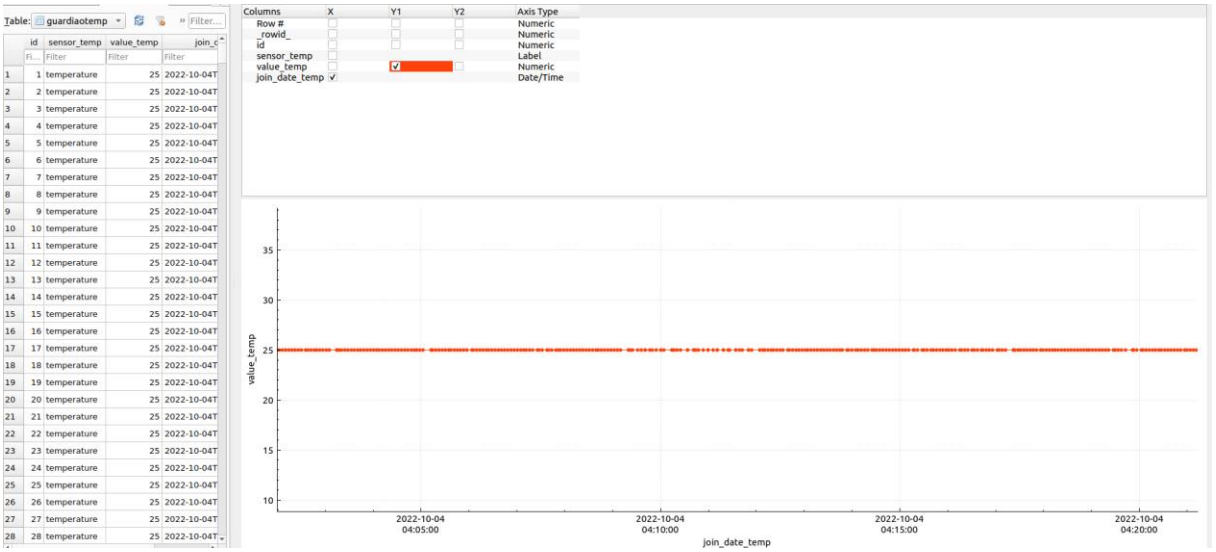
Figura 32 – Banco de dados do sensor de fumaça.



Fonte: Autoria Própria.

O gráfico do sensor de fumaça mostra um certo aumento na leitura devido ao início do teste, onde os dados começaram a ser coletado em ambiente residencial e finalizados dentro de uma área de preservação permanente, porém próximo a uma avenida onde circulam vários ônibus, por isso o aumento no valor lido verificado no sensor de gás que em seguida se estabilizou. A temperatura permaneceu estável durante o período do teste conforme é apresentado na figura 33.

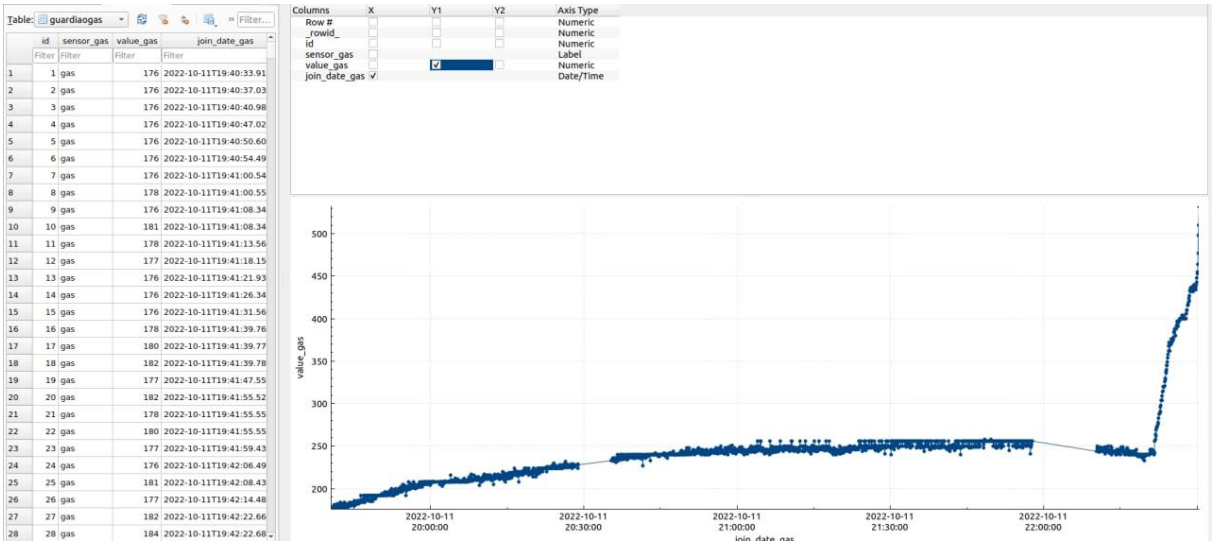
Figura 33 – Banco de dados do sensor de temperatura.



Fonte: Autoria Própria.

A figura 34 e a figura 35, apresentam os resultados do teste de 2 horas que foi realizado nas dependências da EST, os dados foram coletados dentro do ambiente de floresta, em uma distância de aproximadamente 150 metros da placa concentradora de dados. Para finalizar o teste, foi gerada uma pequena fumaça proveniente de uma folha de papel A4, sendo suficiente para estimular sensor de fumaça, porém não houve alteração nos dados do sensor de temperatura, conforme podemos verificar nas imagens abaixo.

Figura 34 - Banco de dados do sensor de fumaça.



Fonte: Autoria Própria.

Figura 35 – Banco de dados do sensor de temperatura.

Fonte: Autoria Própria.

CONCLUSÃO

Portanto, este trabalho de conclusão de curso aborda o desenvolvimento de um protótipo de dispositivo de baixo consumo baseado em *internet of things* (IoT) para detecção de focos de incêndios em regiões de floresta. Para atingir tal objetivo, foram realizadas revisões bibliográficas sobre os assuntos referentes as queimadas na floresta amazônica, *internet of things*, lora, microcontrolador ESP32, com ênfase no microcontrolador ESP32 lora da Heltec, barramento I2C, sensores, *message queuing telemetry transport* (MQTT), banco de dados SQLite.

A implementação da arquitetura do projeto foi realizada de acordo com o planejamento e de acordo com as etapas descritas nos materiais e métodos.

A introdução deste trabalho trouxe a hipótese que é possível desenvolver um protótipo de sistema que colete dados em tempo real e de forma descentralizada, e com base nos resultados obtidos, conclui-se que é possível realizar o monitoramento e salvá-los em um banco de dados. Para ser realizada a visualização dos dados, é necessário utilizar o *software* SQLite Browser para visualização de dados.

Para trabalhos futuros sugere-se um estudo sobre o carregamento da bateria sem a necessidade de intervenção para que o dispositivo permaneça no local de instalação tornando-o mais autônomo possível. Outra sugestão, é a inserção de vários dispositivos de coleta para avaliar uma rede de sensoriamento e integrar esses dados a um modelo de aprendizado de máquina para identificar o padrão do início de um incêndio.

REFERÊNCIAS

ASHTON, K. **That 'Internet of Things' Thing,** 2009. Disponível em: <https://www.rfidjournal.com/that-internet-of-things-thing>. Acesso em 04 out. 2021.

BERTOLETI, P. **Projetos com ESP32 e LORA**. Instituto Newton C. Braga, São Paulo, 2019. Disponível em: <https://www.embarcados.com.br/livro-projetos-com-esp32-e-lora-pedro-bertoleti/>. Acesso em: 04 out. 2021.

BOSCH. **BMP280 Digital Pressure Sensor**. [S.l.], 2021. Disponível em: <https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/>. Acesso em: 02 fev. 2022.

CAMPOS, M. T.; HIGUCHI, F. G. **A Floresta Amazônica e seu papel nas mudanças climáticas**. Manaus, 2009. Disponível em: <https://www.terrabrasilis.org.br/ecotecadigital/pdf/a-floresta-amazonica-e-seu-papel-nas-mudancas-climaticas.pdf>. Acesso em: 01 out. 2021.

DEVOPEDIA. **SQLite**, 2022. Disponível em: <https://devopedia.org/sqlite>. Acesso em 01 ago. 2022.

ESPRESSIF SYSTEMS. **ESP32 Series Datasheet**, 2021. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 05 out. 2021.

FEARNSIDE, P. M. **Queimadas**, 2002. Disponível em: <https://www.scielo.br/j/ea/a/XSwmhQjJmfndLx5RYZpvjxh/?lang=pt&format=pdf>. Acesso em: 03 out. 2021.

HOW TO MECHATRONICS. **How I2C Communication Works?**. [S.l], 2022. Disponível em: <https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/>. Acesso em: 02 fev. 2022.

HUMAN RIGHTS WATCH. **"O ar é insuportável"**, 2020. Disponível em: <https://www.hrw.org/pt/report/2020/08/26/376135>. Acesso em: 03 out. 2021.

IBM. ***MQTT messaging***, 2022. Disponível em: < https://www.ibm.com/docs/en/mapms/1_cloud?topic=reference-mqtt-messaging>. Acesso em: 02 fev. 2022.

INSTRUCTABLES. **Introduction ESP32 Lora OLED Display**, 2021. Disponível em: <https://www.instructables.com/Introduction-ESP32-Lora-OLED-Display/>. Acesso em 15 jan. 2022.

NEPSTAD, D. C.; MOREIRA, A.G.; ALENCAR, A. A**. Floresta em chamas: Origens, impactos e prevenção de fogo na Amazônia**. Instituto De Pesquisa Ambiental Da Amazônia, Brasília, 1999. Disponível em: <https://ipam.org.br/wp-content/uploads/2005/03/floresta_em_chamas_origens_impactos_e_pr.pdf>. Acesso em: 01 out. 2021.

NOVUS. **MQTT**, 2022. Disponível em: <https://www.novus.com.br/site/default.asp?Idioma=55&TroncoID=053663&SecaoID=0&SubsecaoID=0&Template=../artigosnoticias/user_exibir.asp&ID=618088>. Acesso em: 04 ago. 2022.

ORACLE. **O Que É um Banco de Dados?**, 2022. Disponível em: < https://www.oracle.com/br/database/what-is-database/#:~:text=Um%20banco%20de%20dados%20%C3%A9,banco%20de%20dados%20(DBMS)>. Acesso em: 02 fev. 2022.

SILVA NETO, E. **Gateways LoRa: Soluções open-source hardware**, 2018. Disponível em: <https://www.embarcados.com.br/gateways-lora-open-source-hardware/>. Acesso em: 05 out. 2021.

SANTOS, J. W.; LARA JUNIOR, R. C. **Sistema de automatização residencial de custo controlado pelo microcontrolador ESP32 e monitorado via *smartphone***, 2019. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/12133/1/PG_COAUT_2019_1_02.pdf>. Acesso em: 05 out. 2021.

SANTOS, T. O. *et al*. **Os impactos do desmatamento e queimadas de origem antrópica sobre o clima da Amazônia Brasileira: Um estudo em revisão**, 2017. Disponível em: <https://queimadas.dgi.inpe.br/~rqueimadas/material3os/2017_Santos_etal_ImpactosQueimadasOrigemAntropica_RGA_DE3os.pdf>. Acesso em: 02 out. 2021.

SHUTTERSTOCK. **Sensors**, 2022. Disponível em: <https://www.shutterstock.com/pt/image-vector/sensors-icon-set-contains-editable-icons-2026144163>. Acesso em: 05 ago. 2022.

SQLITE. ***What Is SQLite?***, 2022. Disponível em: < https://www.sqlite.org/index.html>. Acesso em: 02 fev. 2022.

WENDLING, M. **Sensores**, 2010. Disponível em: <https://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>. Acesso em: 07 out. 2021.