UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL

UNIDADE EM GUAÍBA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MATHEUS ARAUJO DE LIMA

RTS Embarcado com LoRaWAN para monitoramento de dados de qualidade do solo na agricultura

Porto Alegre
2023

MATHEUS ARAUJO DE LIMA

RTS Embarcado com LoRaWAN para monitoramento de dados de qualidade do solo na agricultura

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Prof. Dr. Celso Maciel da Costa
Orientador

Porto Alegre
2023

UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL
Reitor:
Vice-Reitor:
Pró-Reitor de Graduação:
Coordenador do curso de Engenharia de Computação:

Catalogação de Publicação na Fonte

L732r Lima, Matheus de Araujo.
RTS Embarcado com LoRaWAN para monitoramento de dados de qualidade do solo na agricultura. / Matheus de Araujo Lima – Guaíba, 2023.

75f., il.

Orientador: Prof. Dr. Celso Maciel da Costa.

Trabalho de Conclusão de Curso (Graduação) – Universidade Estadual do Rio Grande do Sul; Curso de Bacharelado em Engenharia da Computação, Unidade em Guaíba, 2023.

1. RTOS. 2. IOT. 3. LoRa 4. Qualidade do solo. 5. Sistemas embarcados. I. Costa, Celso Maciel da. II. Título.

Ficha catalográfica elaborada pela bibliotecária Carina Lima CRB10/1905

AGRADECIMENTOS

Aos meus amados pais, que foram os pilares do meu crescimento e educação, agradeço por seu amor incondicional, sacrifícios incansáveis e apoio constante ao longo desta jornada.

A minha avó Nilda pelo apoio nesta reta final e a minha falecida avó Teresa que sempre acreditou em mim e infelizmente não estará aqui no final.

A toda minha família pelo apoio e por acreditarem em mim.

Aos colegas e amigos que fiz na UERGS e que espero levar para a vida.

Aos funcionários e professores desta instituição, cujo trabalho dedicado e esforços incansáveis contribuíram significativamente para a minha jornada acadêmica.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

IOT	Internet of things(Internet das coisas)
BR	Brasil
UERGS	Universidade Estadual do Rio Grande do Sul
BD	Banco de Dados

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

RESUMO

O presente trabalho apresenta a implementação de um sistema embarcado distribuído para a coleta de dados referentes à qualidade do solo, destinados à agricultura, visando o aperfeiçoamento do processo de tomada de decisões. A coleta de dados é realizada por uma malha de estações de coleta, que são distribuídas pela área de interesse. Cada estação de coleta é composta por um microcontrolador do modelo esp32, um módulo LoRa e32 da eByte usado para comunicação e um ou mais sensores de temperatura, pertinentes à aplicação. As estações de coleta operam realizando leituras dos sensores e transmitindo os dados entre si, seguindo rotas pré definidas, até alcançar uma estação conectada à internet. Esta última encaminha, então, os dados para o servidor. A estratégia de dividir as estações entre nodos coletores e gateways (nodos com conexão a Internet), responsáveis pelo envio ao servidor, viabiliza a utilização da malha de coleta em regiões com baixa infraestrutura, já que apenas uma porção da malha precisa ter acesso à rede de internet. Os dados enviados ao servidor podem ser visualizados em uma interface gráfica, que oferece métricas de acompanhamento, ou exportados para análise por ferramentas externas. As estações foram desenvolvidas utilizando FreeRTOS, um sistema operacional de tempo real. Suas funcionalidades são organizadas como tarefas, executando periodicamente em paralelo e fazendo uso de interrupções para comunicação. O sistema foi projetado e implementado de maneira a facilitar sua adaptação para diversas aplicações que envolvam leituras de sensores em áreas de interesse. A interface gráfica mostrou-se amigável, proporcionando facilidade de uso. Os dados coletados pelos sensores são exibidos sob a forma de gráficos e tabelas, fornecendo elementos para a tomada de decisões relacionadas à área agrícola monitorada.

Palavras-Chave: RTOS, IOT, LoRa, qualidade do solo, sistemas embarcados, sistemas distribuídos

ABSTRACT

The present work introduces the implementation of a distributed embedded system for collecting data related to soil quality in agriculture, aiming to enhance the decision-making process. Data collection is performed by a network of collection stations distributed throughout the area of interest. Each collection station comprises an ESP32 microcontroller, an eByte LoRa E32 module for communication, and one or more temperature sensors relevant to the application. The collection stations operate by reading sensors and transmitting data among themselves, following predefined routes until reaching an internet-connected station. This latter station then forwards the data to the server. The strategy of dividing stations into collector nodes and gateways (nodes with internet connection), responsible for sending data to the server, enables the use of the collection network in regions with limited infrastructure, as only a portion of the network needs internet access. The data sent to the server can be viewed on a graphical interface, providing tracking metrics, or exported for analysis using external tools. The stations were developed using FreeRTOS, a real-time operating system. Their functionalities are organized as tasks, executing periodically in parallel and utilizing interruptions for communication. The system is designed and implemented to facilitate adaptation to various applications involving sensor readings in areas of interest. The graphical interface proved to be user-friendly, offering ease of use. The sensor-collected data is presented in the form of graphs and tables, providing insights for decision-making related to the monitored agricultural area.

Keywords: RTOS, IOT, LoRa, soil quality, embedded systems, distributed systems.

1 INTRODUÇÃO

A agricultura desempenha um papel crucial na economia brasileira, fornecendo alimentos e matérias-primas essenciais. A crescente demanda por aumento na produção agrícola tem impulsionado o desenvolvimento de novas tecnologias, incluindo sistemas embarcados e distribuídos, para otimizar processos e aprimorar a tomada de decisões. [1][2]

Diante da escassez de recursos naturais e da necessidade de preservação ambiental, a busca por maneiras eficientes de usar água e insumos na produção alimentar tornou-se uma preocupação central. A implementação de tecnologias que permitam o uso sustentável desses recursos é vital para garantir a sustentabilidade dos processos agrícolas.

Esta monografia tem como objetivo conceber e implementar um sistema embarcado distribuído para a coleta de dados sobre a qualidade do solo, visando melhorar a produção agrícola. A qualidade do solo desempenha um papel crucial na produtividade e saúde das culturas, tornando o monitoramento constante uma prática essencial para otimizar o uso de insumos nas terras cultiváveis.[10]

O sistema proposto consiste em uma rede de dispositivos embarcados implantados ao longo da área agrícola. Esses dispositivos, conhecidos como nós ou dispositivos finais, podem ser equipados com sensores de alta precisão para coletar dados relevantes, como umidade, temperatura e pH do solo. Os dados coletados são enviados para um dispositivo central, o gateway, que processa as informações e as envia para um servidor na nuvem. Essa abordagem fornece informações cruciais para auxiliar os profissionais do agronegócio na tomada de decisões e na execução de suas funções.

A utilização desse sistema oferece diversas vantagens, incluindo coleta em tempo real, cobertura eficiente de grandes áreas com baixo custo, acesso a dados históricos e monitoramento contínuo. Os dados coletados capacitam agricultores e profissionais da área a compreender melhor a situação e o comportamento do solo, facilitando e otimizando os processos agrícolas.

2 OBJETIVOS

No cenário da agricultura atual o desenvolvimento de um sistema embarcado distribuído para o monitoramento da qualidade de solo é de alta relevância. Com o aumento constante da demanda por alimentos devido ao crescimento populacional, torna-se necessário implementar o uso de tecnologias que otimizem os processos agrícolas e garantam um uso eficiente dos insumos necessários[3][4]. O desenvolvimento do sistema proposto irá possibilitar o monitoramento contínuo e em tempo real das condições do solo em largas extensões de terra, assim proporcionando aos agricultores informações úteis para embasar e validar as tomadas de decisões.

Além disso, a otimização do uso de recursos e a implementação de práticas agrícolas mais eficientes contribuem para a sustentabilidade do agronegócio, minimizando impactos ambientais negativos.

2.1 Objetivo Geral

Este trabalho tem como objetivo conceber, implementar e validar um sistema embarcado distribuído para o monitoramento de dados referentes à qualidade do solo em áreas rurais, visando auxiliar no planejamento agrícola e possibilitando o uso otimizado dos recursos necessários ao setor.

O sistema consiste de uma rede de coleta de dados, composta por dispositivos ligados a sensores relacionados a métricas importantes na determinação da qualidade do solo. Cada dispositivo deve operar de forma independente e utilizar a comunicação com seus pares a fim de enviar os dados coletados até um gateway que por sua vez realizará o envio ao servidor na nuvem. O gateway é um dispositivo capaz de receber os dados coletados pelos diversos nós da rede e transmiti-los para o servidor remoto. Após o envio ao servidor os dados estarão disponíveis para visualização em uma interface gráfica.

O Sistema como um todo deve ter capacidade de operar em regiões com baixo acesso a conexão com a internet, requerendo uma infraestrutura mínima para sua implementação.

2.2 Objetivos Específicos

- **Revisão bibliográfica**: Realizar uma revisão bibliográfica sobre sistemas embarcados, IoT, redes de sensores, comunicação LoRa e qualidade do solo para embasar o desenvolvimento do sistema.
- **Malha de coleta de dados**: Projetar e implementar uma rede distribuída de estações de coleta de dados. Cada estação sendo formada por um microprocessador, sensores de interesse e LoRa, tecnologia usada para a transferência de dados entre as estações e com capacidade de cobrir áreas extensas e necessitando de baixa infraestrutura.
- **Sistema de tempo real**: Desenvolver um sistema de tempo real capaz de lidar com a comunicação, coleta de dados e demais funcionalidades das estações de coleta utilizando FreeRTOS.
- **Sistema Distribuído**: Desenvolver um sistema distribuído que permita a coleta e monitoramento dos dados de forma eficiente utilizando a comunicação entre as estações de coleta e o servidor.
- **Adaptabilidade**: Projetar o sistema e seus componentes de tal forma que possa ser utilizado em diversas aplicações sem necessidade de grandes alterações.

No cumprimento destes objetivos o trabalho colabora com a agricultura moderna, fornecendo uma ferramenta para que os agricultores e profissionais do agronegócio possam melhorar os processos, reduzir os custos e promover a sustentabilidade das atividades agrícolas.

3 MATERIAIS E MÉTODOS

Para alcançar os objetivos propostos por este trabalho, bem como testar as funcionalidades implementadas , foram desenvolvidos protótipos de hardware e software.

Nesta seção serão detalhados os materiais, métodos e técnicas utilizadas na elaboração deste projeto

3.1 ESP-32

O ESP-32 [5][6] é uma plataforma de desenvolvimento de projetos de microcontrolador de código aberto, inicialmente desenvolvida pela Espressif Systems, que vem cada vez mais se tornando uma das opções mais populares para projetos de Internet das Coisas (IoT). Conta com um poderoso processador dual-core que opera com uma frequência de até 240 MHz, chegando até 600 DMIPS(Dhrystone milhões de instruções por segundo) por segundo. O ESP-32 oferece alto desempenho computacional, sendo utilizado em tarefas variadas. Possui suporte a interfaces e periféricos variados, como GPIOs, UART, SPI e I2C, oferecendo flexibilidade e versatilidade no desenvolvimento de aplicações.

Figura 3.1.1 Placa esp32 usada no projeto (Matheus A. de Lima,2023)

Além disso, o ESP32 tem compatibilidade com uma grande variedade de ambientes de desenvolvimento, como a Arduino-framework e o ESP-IDF (Espressif IoT Development Framework). Essa compatibilidade torna mais simples o processo de desenvolvimento de projetos, tornando o ESP32 acessível tanto para iniciantes quanto para desenvolvedores experientes. O ESP32 se destaca como uma plataforma flexível e poderosa para projetos de IoT e eletrônica embarcada.

Neste projeto utilizamos a ESP32-WROOM-32 com núcleo ESP32-D0WDQ6.

3.2 RA-02

É um microchip transceptor de comunicação sem fio que utiliza a tecnologia LoRa[7] para realizar a transmissão de dados em longas distâncias, tornando-o ideal para aplicações que exigem alcance estendido e baixo consumo de energia. Opera na frequência de comunicação de 433 MHz.

O RA-02 utiliza a modulação LoRa para alcançar transmissões confiáveis, mesmo em ambientes com interferências, possibilitando a cobertura de áreas extensas. O RA-32 utiliza como base o chip transceptor de rádio SX1278, suporta comunicação bidirecional, e permite o envio e recebimento de dados entre dispositivos. É uma escolha atual para aplicações de IoT, rastreamento de ativos, e monitoramento remoto de sensores.

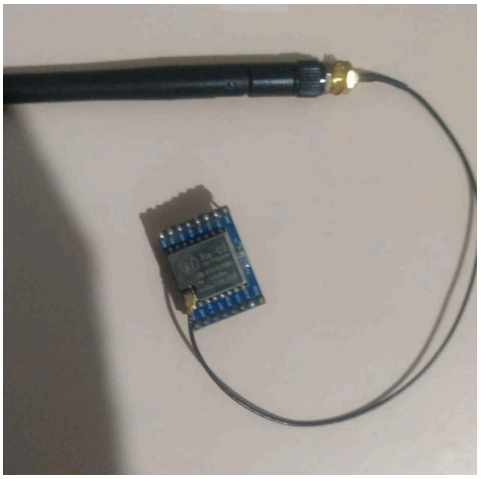

Figura 3.2.1 Placa ra-02 usada no projeto (Matheus A. de Lima,2023)

3.3 PlatformIO

PlatformIO [8] é uma plataforma de desenvolvimento de código aberto que visa simplificar e facilitar o processo de desenvolvimento de sistemas embarcados.

Possui suporte para diversas plataformas de hardware, entre elas, Arduino, ESP8266, ESP32, Raspberry Pi e outras.

Oferece um ambiente unificado e amigável para escrever e compilar código. Sua integração com editores de texto populares, como Visual Studio Code e Sublime Text, proporciona uma experiência de desenvolvimento que permite que os desenvolvedores se concentrem no desenvolvimento de projetos, ao invés de se preocuparem com a configuração do ambiente de desenvolvimento. Oferece recursos de gerenciamento de bibliotecas e dependências. O PlatformIO é uma ferramenta muito usada na comunidade de desenvolvimento de sistemas embarcados, tornando o processo de programação mais ágil e simples.

3.4 Arduino-framework

Originalmente desenvolvido para as placas Arduino, esse framework [9] suporta também placas com base Espressif(ESP8266,ESP-32 e etc). Conta com ampla gama de bibliotecas e uma comunidade global altamente ativa. Projetado para ser amigável e acessível, simplifica o processo de desenvolvimento para microcontroladores. O Arduino Framework oferece muitos recursos e funcionalidades, que vão desde o controle de luzes e motores até a coleta de dados de sensores.

3.5 FreeRTOS

O FreeRTOS (Real-Time Operating System) [10] consiste em um sistema operacional de tempo real de código aberto desenvolvido para sistemas embarcados e microcontroladores. Fornece uma solução eficiente e confiável para gerenciar e agendar tarefas em aplicativos de tempo real. É amplamente utilizado em muitos setores, incluindo automotivo, aeroespacial, médico, industrial e Internet das Coisas (IoT), devido à sua portabilidade, escalabilidade e baixo consumo de recursos.

Possui um pequeno kernel, que requer apenas alguns kilobytes de memória, tornando-se uma opção adequada para dispositivos com recursos limitados, como microcontroladores. Oferece suporte ao agendamento proativo de tarefas, permitindo que aplicativos críticos em tempo real sejam priorizados e executados em horários específicos do dia, garantindo um funcionamento previsível do sistema. O FreeRTOS também fornece sincronização de tarefas e semáforos, gerenciamento de

memória, bem como uma arquitetura bem definida de tarefas e interrupções. É altamente customizável, assim permitindo que os desenvolvedores escolham os recursos que precisam para suas aplicações, garantindo um sistema otimizado.

Por ser de código aberto, o FreeRTOS possui uma grande comunidade, o que significa que a plataforma constantemente recebe atualizações, sendo aprimorada e suportada por diversos desenvolvedores ao redor do mundo. Essa comunidade fornece documentação detalhada , suporte técnico, e uma extensa coleção de exemplos, tornando o FreeRTOS uma ótima escolha para sistemas de tempo real em diversos aplicativos.

3.6 LoRa e LoRaWan

LoRa [7] [11] [12] é uma tecnologia de comunicação sem fio desenvolvida para aplicações de longo alcance e baixo consumo de energia, muito utilizada em projetos de Internet das Coisas (IoT). Uma característica fundamental do LoRa é sua capacidade de transmitir dados por distâncias consideravelmente longas, mesmo em áreas urbanas densamente povoadas e ambientes rurais remotos. Operando em frequências não licenciadas, como 33 MHz, 868 MHz e 915 MHz, essa tecnologia oferece ampla cobertura e permite a comunicação entre dispositivos em uma grande área.

Trata-se de um protocolo de Chirp Spread Spectrum (CSS), isto é, uma técnica baseada nos conceitos de chirp (pulso de radar de alta intensidade comprimido) e espectro espalhado. No CSS o sinal é espalhado por uma ampla faixa de frequências durante a transmissão, o sinal de chirp é caracterizado por sua frequência que varia de forma linear ao longo do tempo de transmissão.

O processo de comunicação LoRa consiste dos seguintes passos:

Sincronização: Para decodificar corretamente, o receptor LoRa deve estar em sincronia com o transmissor. Os horários de início e término do chirp recebido são ajustados para os parâmetros de transmissão conhecidos.

Detecção de chirp: Detecta-se a presença de chirps no sinal recebido. Identificando o momento em que estes são enviados.

Estimativa do Parâmetro de Frequência: Ao identificar os chirps, o algoritmo estima os parâmetros de frequência.

Demodulação de Máxima Verossimilhança: Com parâmetros de frequência estimados e usando o princípio de máxima verossimilhança, o algoritmo usa o chirp recebido para determinar quais sequências de bits seriam as mais prováveis de terem sido geradas.

Correção de erro: Após a demodulação, os dados decodificados podem conter erros. Dependendo da implementação e da taxa de erro desejada, técnicas de correção de erro, como codificação de correção de erro direta (FEC), podem ser aplicadas.

Recuperação de dados: Após obter a sequência correta de bits, os dados são recuperados e disponibilizados para processamento pelas camadas superiores do protocolo.

Uma grande vantagem desta tecnologia é o seu baixo consumo de energia. Isso permite que os dispositivos conectados usem baterias como fonte de alimentação, o que é essencial para aplicativos IoT que tendem a ser instalados em locais de difícil acesso. Essa eficiência energética o torna uma ótima escolha para projetos que exigem comunicação constante e baixo consumo de energia.

Uma outra vantagem da utilização da comunicação LoRa é a baixa necessidade de infraestrutura prévia. os dispositivos de uma rede LoRa apenas necessitam de um circuito transceptor de rádio com suporte ao protocolo LoRa e uma bateria simples. isso faz com que a tecnologia se torne uma ótima escolha para implementações em zonas sem infraestrutura.

Outras opções de comunicação como a tecnologia BLE (Bluetooth Low Energy) [13][14] e às comunicações sem fio padrão como Wi-Fi, apesar de serem consideravelmente mais rápidas e serem boas opções para diversas aplicações, se tornam limitadas quando o assunto é alcance e baixo consumo de energia. A tecnologia BLE é projetada para comunicação de curta distância e é muito utilizada

em dispositivos vestíveis como smartwatches, tem um alcance na ordem máxima das dezenas de metros e um consumo considerável de energia. Já o Wi-Fi apesar de muito conhecido e utilizado, requer uma infraestrutura com roteadores para ser utilizado, além de ter um alcance limitado e um grande consumo de energia. De acordo com testes e comparações realizados por Octavio José Salcedo-Parra e Nelson Giovanni Agudelo-Cristancho em 2021 [13] , a tecnologia LoRa quando empregada em cenário urbano, em condições similares com BLE ou WiFi obtém um alcance efetivo melhor, mostrando ser capaz de conseguir um alcance até 14 vezes maior do que a tecnologia BLE e 7 vezes maior do que a tecnologia WiFi.

Vale notar os testes de precisão da comunicação realizados por Fasih Ullah Khan, Muhammad Awais, Muhammad Babar Rasheed , Bilal Masood, and Yazeed Ghad [14] i. Estes testes comparam a precisão e perda de pacotes em ambiente interno tendo sido realizados em 2021, o trabalho demonstra de forma clara que a precisão do Wifi é maior do que a da tecnologia LoRa enquanto a tecnologia BLE se mostra a mais ineficiente em todos os cenários comparados.

LoRaWAN é o protocolo de comunicação de rede baseado em LoRa mais utilizado, descrito em detalhes em [15]. Permite que dispositivos IoT se comuniquem com gateways de rede a uma distância de alguns quilômetros em áreas urbanas e pode ter um alcance superior a 10 km em zonas rurais

A topologia de rede comumente usada neste protocolo pode ser definida como uma estrela hierárquica com três níveis:

1. Nível de dispositivos finais (End-Devices): os dispositivos finais são os dispositivos IoT que se comunicam com a rede, enviando e recebendo dados.
2. Nível de gateway: o gateway é o dispositivo que atua como ponte entre os dispositivos finais e a rede central. Ele recebe os dados dos dispositivos finais e os encaminha para a rede central, e também recebe os dados da rede central e os encaminha para os dispositivos finais.
3. Nível de rede central: a rede central é responsável por gerenciar os gateways e o tráfego de dados na rede LoRaWAN. Ela também fornece serviços de segurança e gerenciamento de dispositivos.

Pode-se considerar um possível servidor de aplicação como um quarto nível que se comunica com o terceiro nível e é responsável por armazenar e processar os dados.

3.7 E32-900T20D

O E32-900T20D é um módulo de comunicação sem fio LoRa (Long Range), projetado para operar a 900 MHz. Este módulo utiliza como base o circuito de rádio SX1276, que oferece excelente desempenho, tanto em alcance quanto em capacidade de recepção. Graças à tecnologia LoRa, o E32-900T20D é ideal para aplicações que requerem transmissão de dados a longa distância e baixo consumo de energia.

Figura 3.7.1 Placa E32-900T20D usada no projeto (Matheus A. de Lima,2023)

Suporta transferência bidirecional de dados, que permite a troca de dados entre dispositivos. Isso torna o módulo adequado para várias aplicações de Internet das Coisas (IoT), como monitoramento remoto, telemetria, automação e controle, onde a transferência de informações é essencial para a operação adequada do sistema.

Além disso, conta com diversas funcionalidades extras construídas na camada física como possibilidade de broadcast e single cast, divisão da comunicação em canais em modo de operação de baixo consumo de energia entre outras.

Combinando todas essas funções, torna-se uma solução confiável e robusta para projetos que requerem comunicação sem fio de longa distância, como sistemas de telemetria rural, monitoramento remoto de sensores ou comunicação e controle de dispositivos distribuídos em uma rede de Internet das Coisas. Sua eficiência e facilidade de uso o tornam uma escolha adequada para transmissão de dados em longas distâncias.

2.7.1 FEC forward error correction

É uma tecnologia de correção de erro implementada nos chips da família E32. O FEC consiste em adicionar informações redundantes nos dados a serem enviados com o objetivo de que o destinatário possa detectar ou, dependendo do caso e do grau de redundância, até mesmo corrigir possíveis erros na transmissão. Esse tipo de método é especialmente útil para transmissão em canais sujeitos a interferências, ruídos ou perda de dados, segundo Faber, M. J., Zwaag, K. M. vd, Dos Santos, W. G. V., Rocha, H. R. O., Segatto, M. E. V., & Silva, J. A. L [16] "*O uso de FEC é extremamente importante no LoRaWAN, especificamente em distâncias e áreas onde os valores de RSSI estão próximos das sensibilidades do receptor*".

Existem diversos algoritmos FEC, bem como parâmetros para os mesmos, sendo possível variar o grau de redundância. É importante notar que um grau maior se traduz em um “payload” maior e, consequentemente, uma transmissão mais lenta.

O algoritmo utilizado nos módulos da linha E32 da eByte, bem como seus parâmetros e detalhes de implementação não são revelados na documentação disponível. O FEC pode ser desativado nesses módulos, caso seja necessário. Ao desativá-lo, a velocidade de transmissão aumenta ao custo de uma redução na confiabilidade e no alcance útil da comunicação É necessário utilizar a mesma configuração no emissor e no receptor.

2.7.2 Configuração do módulo

A comunicação LoRa e o módulo utilizados possuem diversos parâmetros configuráveis Para que seja possível a troca de mensagens entre dois dispositivos é necessário que estes parâmetros sejam iguais em ambos.

Alguns destes parâmetros são responsáveis pela forma como os dados são transmitidos, tais como, como a “taxa de transferência de dados no ar” e o “poder de transmissão”. O primeiro define a taxa com que os dados serão enviados e o segundo determina quanto de potência o módulo irá utilizar. Também é necessário ajustar o modo de paridade usado em ambos os dispositivos, pois havendo divergência os dados serão interpretados de forma errônea .

A biblioteca utilizada neste projeto fornece uma estrutura de dados para configuração destes parâmetros, o que acaba simplificando este processo. Um exemplo de configuração pode ser visto abaixo:

```
ResponseStructContainer c;
c = e32Serial.getConfiguration();
Configuration configuration = *(Configuration *)c.data;
/* seta as configuraçoes */
configuration.SPED.airDataRate = AIR_DATA_RATE;
configuration.SPED.uartBaudRate = UART_BPS_9600;
configuration.ADDH = self.ADDH;
configuration.ADDL = self.ADDL;
configuration.CHAN = self.CHAN;
configuration.OPTION.fec = FEC_1_ON;
configuration.OPTION.ioDriveMode = IO_D_MODE_PUSH_PULLS_PULL_UPS;
configuration.OPTION.transmissionPower = POWER_20;
configuration.OPTION.fixedTransmission = FT_FIXED_TRANSMISSION;
configuration.OPTION.wirelessWakeupTime = WAKE_UP_250;
configuration.SPED.uartParity = MODE_00_8N1;

ResponseStatus rs = e32Serial.setConfiguration(configuration,
WRITE_CFG_PWR_DWN_SAVE);
```

Algoritmo 3.7.1 Configuração do e32

Neste exemplo os parâmetros são acessados, alterados e salvos para os valores desejados em poucas linhas.

Os parâmetros dentro de “SPED” são airDataRate e uartBaudRate. O primeiro, é responsável pela taxa de transferência de dados no envio e recebimento e deve ser o mesmo entre emissor e receptor para que a comunicação ocorra.Taxas menores significam uma transmissão mais lenta e um alcance mais alto. O segundo parâmetro uartBaudRate define a taxa de comunicação entre o módulo e o

microcontrolador. Este parâmetro pode ser diferente entre emissor e receptor, uma vez que não está relacionado a transmissão de dados. Ambos estes parâmetros recebem valores enumerados na biblioteca.

Os parâmetros ADDH e ADDL representam o endereço do transceptor e são utilizados para a transmissão endereçada. Ambos possuem um byte de tamanho e compõem o endereço do módulo na rede, seus valores variam de 00 a FF e o valor padrão é 00.

O parâmetro CHAN representa o canal de comunicação do módulo e deve ser o mesmo entre emissor e receptor. Quando uma mensagem é transmitida são consideradas a frequência e o canal configurados para determinar a frequência final, seguindo a seguinte equação: frequência*M + CHAN*1M ou seja para uma frequência de 915 teria-se 915M + CHAN*1M, caso M fosse 4 a frequência final seria 919 Mhz.

O parámetro fec pode receber os valores FEC_1_ON e FEC_0_OFF e define se a correção em hardware está ativa ou inativa. Este parâmetro deve ser o mesmo entre emissor e receptor uma vez que seu valor altera como o envio e recebimento são realizados.

O parâmetro ioDriveMode é usado para o resistor de pull-up interno do módulo. Ele também aumenta a adaptabilidade do nível, no caso de dreno aberto. No entanto, em alguns casos, pode ser necessário um resistor de pull-up externo.

O parâmetro transmissionPower está relacionado ao poder de transmissão do módulo e deve ser verificado na documentação, de acordo com o número de série do chip que está sendo utilizado.

O parâmetro fixedTransmission determina o modo de transmissão a ser usado e seu valores possíveis são: FT_TRANSPARENT_TRANSMISSION (transparente) e FT_FIXED_TRANSMISSION (fixada). No modo de transmissão fixo, os três primeiros bytes de cada quadro de dados do usuário podem ser usados como endereço alto/baixo e canal. O módulo altera seu endereço e canal ao transmitir. Retornará às configurações originais após concluir o processo.

O parâmetro wirelessWakeupTime determina o delay entre o recebimento da mensagem e o início da transmissão ao microcontrolador. Este parâmetro precisa ser o mesmo entre as duas partes da comunicação.

uartParity determina a paridade usada na comunicação entre o módulo e o microcontrolador. Este parâmetro não interfere na comunicação e pode ser diferente entre as partes.

3.8 NodeJs

Trata-se de um framework de código aberto que permite a execução de programas escritos em javascript fora do navegador[17]. Construído sobre a engine V8 do Google é conhecido por ser leve, prático e muito utilizado na construção de API's Rest e microsserviços.

Popular entre os desenvolvedores de software, conta com uma grande comunidade que constrói e mantém diversos pacotes e bibliotecas através do "Node Package Manager(npm)", para os mais diversos fins.

3.9 Rest

Sigla de "Representational State Transfer", trata-se de um modelo de arquitetura amplamente utilizado para o desenvolvimento de sistemas de software na web. Fornece princípios e diretrizes para projetar sistemas baseados em web. Tem sua base na ideia de que os recursos devem ser tratados como objetos acessíveis a partir de urls e, as operações em tais recursos por sua vez devem utilizar os métodos HTTP normais: GET (consulta), POST (cadastro), PUT (atualização), DELETE (exclusão)[18].

Sistemas Rest são "stateless" , ou seja, não possuem estado. Cada requisição contém todos os dados necessários para ser concluída, eliminando a necessidade de manter dados de estado no servidor. Essa diretiva torna o sistema mais fácil de escalar e manter. Além disso, promove acesso de forma uniforme e previsível dos recursos, sendo uma escolha muito usada atualmente.

3.10 MySQL

É um sistema de gerenciamento de banco de dados relacional de código aberto originalmente desenvolvido pela Oracle. Muito conhecido e utilizado , o mysql é um opção racional para o desenvolvimento de variadas aplicações[19].

3.11 Sequelize

Trata-se de uma biblioteca de mapeamento objeto-relacional (ORM) para Node.js que simplifica muito a interação com bancos de dados relacionais, tais como MySQL, PostgreSQL, SQLite entre outros. Permite que os desenvolvedores escrevam consultas e operações de banco de dados em JavaScript, facilitando o desenvolvimento.Além disto, o Sequelise oferece recursos avançados, como migração de banco de dados, junções de tabelas e geração automática de consultas SQL[20].

3.12 Grafana

Grafana é uma ferramenta de código aberto para análise de dados multi propósito, isto é, pode ser usada de diferentes formas, tais como, para monitorar erros, logs ou mudanças em banco de dados. A ferramenta é gratuita e conta com diversas funcionalidades para construção e gerenciamento de interfaces para monitoramento de dados interativos e em tempo real podendo ser conectada a bases de dados ou a arquivos de texto[21].

4 REVISÃO BIBLIOGRÁFICA

Na presente seção, será apresentado um panorama do estudos e pesquisas preexistentes relacionados ao tema central deste Trabalho de Conclusão de Curso (TCC): o desenvolvimento e implementação de um sistema embarcado distribuído para a coleta de dados de sensores sobre a qualidade do solo no agronegócio usando LoRa como comunicação.

No campo específico da integração de sistemas embarcados na agricultura, é importante compreender como as abordagens e descobertas anteriores moldaram o entendimento atual do tema. Serão examinados não apenas os avanços tecnológicos, mas também as metodologias adotadas para a coleta de dados no contexto agrícola.

A contribuição de Achmad Fauzi Rachmani [22], em 2018, é notável na concepção de um sistema comparável ao proposto neste estudo. Utilizando o microcontrolador Arduino UNO, foi desenvolvida uma solução para monitorar uma plantação de carambolas na Indonésia. Seu trabalho oferece uma descrição minuciosa do sistema implementado, composto por unidades coletoras, um gateway, um servidor central organizados em uma topologia estrela, além de uma interface gráfica para visualização dos dados. O detalhamento fornecido por Rachmani é essencial, pois esclarece a estrutura do sistema, seus componentes e a integração eficaz entre eles. Além da implementação específica, Rachmani apresenta alternativas tecnológicas, como o uso de Zigbee. A exploração dessas alternativas demonstra uma compreensão abrangente do panorama tecnológico disponível para projetos semelhantes.

A pesquisa conduzida por Huang-Chen Lee e Kai-Hsiang Ke [23], em 2018, é uma valiosa contribuição, uma vez que explora o uso de redes do tipo malha e LoRa para a coleta de dados. Este estudo oferece uma definição clara e uma avaliação abrangente do emprego dessas redes, apresentando vantagens e desafios de forma precisa e aprofundada.

Ao detalhar as características das redes de malha utilizando LoRa, Lee e Ke fornecem uma base sólida para compreensão das tecnologias em questão. Além disso, destacam não apenas os benefícios, mas também os desafios associados a

essas abordagens. Esta pesquisa é essencial para informar decisões no desenvolvimento de sistemas de coleta de dados, oferece dados valiosos sobre os detalhes dessas tecnologias, permitindo uma análise crítica e fundamentada.

Dessa forma, a pesquisa de Lee e Ke expande o conhecimento sobre o uso de topologia em malha com LoRa na coleta de dados, e fornece uma análise dos aspectos positivos e desafios associados a essas tecnologias específicas.

O trabalho conjunto de Bouras, C., Gkamas, A., Kokkinos, V., & Papachristos, N[24] em 2019 forneceram uma comparação entre o uso de WiFi e LoRa para coleta e envio de dados, apesar de o sistema proposto por eles se tratar de coleta de geolocalização para rastreamento e monitoramento da temperatura a comparação realizada entre as duas tecnologias é valiosa para justificar e compreender a tecnologia que mais se adequa a cada cenário.

Montagny, Sylvain[25], da Universidade de Savoie-Mont-Blanc, apresentou um sistema semelhante documentado em seu webinar "All you need to know about LoRaWAN". Neste webinar, ele explora o desenvolvimento de uma rede LoRaWAN para monitoramento de colmeias de abelhas, utilizando a topologia típica em estrela associada a essa tecnologia. O trabalho de Montagny, Sylvain, apresenta uma abordagem didática, fornecendo explicações detalhadas de vários conceitos fundamentais e relevantes para essa aplicação específica. Sua documentação apresenta de forma clara os princípios subjacentes e as práticas envolvidas na implementação bem-sucedida de redes LoRa para monitoramento agrícola

Wang, Y., Wang, Y., Qi, X., & Xu, L[26] em 2009 forneceram o detalhamento de um sistema similar usando topologia estrela e zigbee para sua implementação. É válido notar a contribuição deste trabalho para os temas relacionados uma vez que fornece com detalhes a implementação de uma arquitetura para coleta de dados, apesar de utilizarem uma tecnologia diferente.

No seu trabalho de conclusão de curso pela Universidade Federal do Rio Grande do Sul em 2022, Matté, Junior [27], apresentou uma análise detalhada sobre o emprego de redes LoRaWAN em topologia estrela para a coleta de dados geográficos e geológicos. A pesquisa conduzida por Matté, Junior, apresenta uma descrição da implementação prática e os benefícios do uso de redes LoRa para coleta de dados de sensores.

Este trabalho posiciona-se como uma contribuição significativa, descrevendo em detalhes o desenvolvimento em implementação de um sistema de coleta e

monitoramento de dados para regiões de interesse, como foco na agricultura. Uma das características distintivas desta implementação reside na escolha de uma topologia de rede híbrida, combinando elementos de redes em malha e em estrela. Esta decisão foi motivada pelas necessidades específicas do ambiente agrícola, visando oferecer uma aplicação de baixo custo e que necessite ao mínimo de infraestrutura externa ao projeto. A combinação de características dessas duas topologias oferece uma abordagem mais adaptável e resiliente, capaz de lidar com as complexidades e desafios inerentes às regiões rurais.

Outra inovação está na implementação da segmentação por canais. Essa adaptação foi incorporada para lidar com interferências potenciais e otimizar a eficiência da transmissão de dados. A segmentação dinâmica por canais mitiga possíveis conflitos de pacotes e assegura uma transmissão mais confiável, especialmente em ambientes com múltiplos dispositivos se comunicando.

Essas diferenciações expandem o conhecimento já existente do tema e oferecem novos caminhos de desenvolvimento para problemas similares, além disto este trabalho oferece uma descrição clara do processo de desenvolvimento do projeto e das tecnologias utilizadas.

5 PROTOTIPAGEM E TESTES

Foram realizados testes com o objetivo de determinar qual circuito transceptor seria utilizado no projeto. Os testes foram realizados com condições climáticas limpas em zona urbana com relevo e obstáculos variáveis.

Os testes também tinham o objetivo de determinar a influência de fatores físicos e geográficos no alcance da transmissão, por exemplo, qual tipo de obstáculo causaria maior perda de alcance.

5.1 Especificações dos testes

Os testes utilizaram os dispositivos que constavam com um transceptor RA-02 ou E32-900T20D conectado a uma esp32. Utilizou-se o mesmo transceptor em dois dispositivos, um destes ficou parado enviando um sinal a uma taxa constante(emissor) e o outro foi deslocado e teve sua taxa de recebimento aferida em pontos fixados do mapa.

Foi utilizado um programa que envia uma mensagem a cada 5 segundos com um número múltiplo de 5 indo de 0 até 60, totalizando assim 12 mensagens por ciclo.O receptor por sua vez recebe essas mensagens e as contabiliza, calculando quantas destas são perdidas em um ciclo de um minuto.

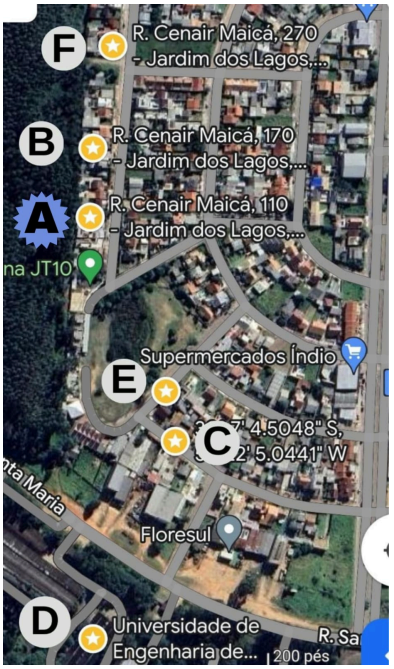

Figura 5.1.1: Mapa dos pontos de coleta utilizados nos testes (Matheus A. de Lima,2023)

A imagem acima mostra os pontos de interesse do teste. O ponto “A” é a localização fixa do dispositivo emissor e os demais pontos são os locais de medição da taxa de recebimento do sinal. As distâncias do emissor até os pontos de coleta pode ser vistas na seguinte tabela:

Tabela 5.1.1 Pontos de coleta

Ponto	Distância do emissor
B	70m
C	222m
D	400m
E	158m
F	199m

É importante notar que o emissor encontrava-se dentro de uma casa e que a linha entre emissor e receptor contava com obstáculos comuns ao meio urbano, tais como casas, vegetação e postes.

Vale também ressaltar a diferença de relevo entre os pontos de coleta. Os pontos “C”, “D” e “E” estavam localizados em posições mais altas que o emissor, enquanto os pontos “B” e “F” encontravam-se mais baixos que o ponto de emissão, tendo uma vantagem física no recebimento do sinal. Além disso, as barreiras entre o emissor e os pontos mais baixos eram menos presentes, enquanto o ponto “E” contava com obstáculos formados em sua maior parte por vegetação (uma praça) e os demais contavam como obstáculos massivos como construções, muros e prédios.

5.2 Testes com RA-02

O dispositivo testado consistia de uma esp32 conectada ao módulo ra-02 que por sua vez usava uma antena simples para envio do sinal.

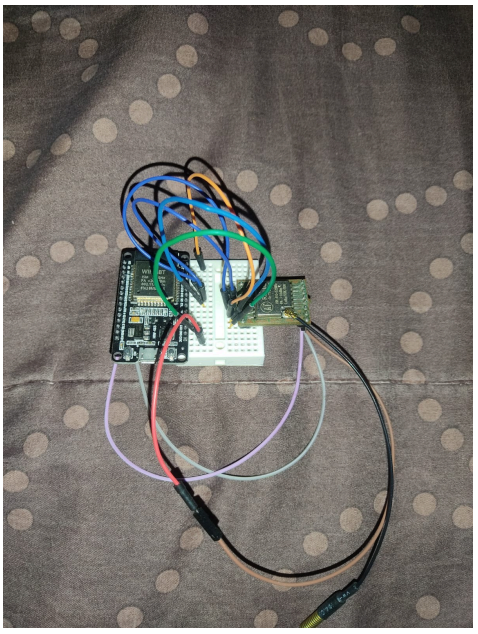

Figura 5.2.1: Protótipo com RA-02 (Matheus A. de Lima,2023)

Na configuração do ra-02 o desenvolvedor é capaz de ajustar o fator de espalhamento usado pelo módulo. O fator de espalhamento é uma configuração importante no protocolo LoRa que determina a taxa de transmissão de dados, o tempo de transmissão e a sensibilidade do receptor.

Quanto maior o fator de espalhamento, menor a taxa de transmissão, mas maior a resistência ao ruído e a interferências. Isso permite que a transmissão alcance maiores distâncias, mas em contrapartida, leva mais tempo para transmitir os dados. Em resumo, o fator de espalhamento afeta o alcance do sinal de LoRa, permitindo que ele chegue a distâncias maiores em detrimento da velocidade de transmissão.

A figura 5.2.2 ilustra a organização dos dispositivos utilizados durante os testes realizados.

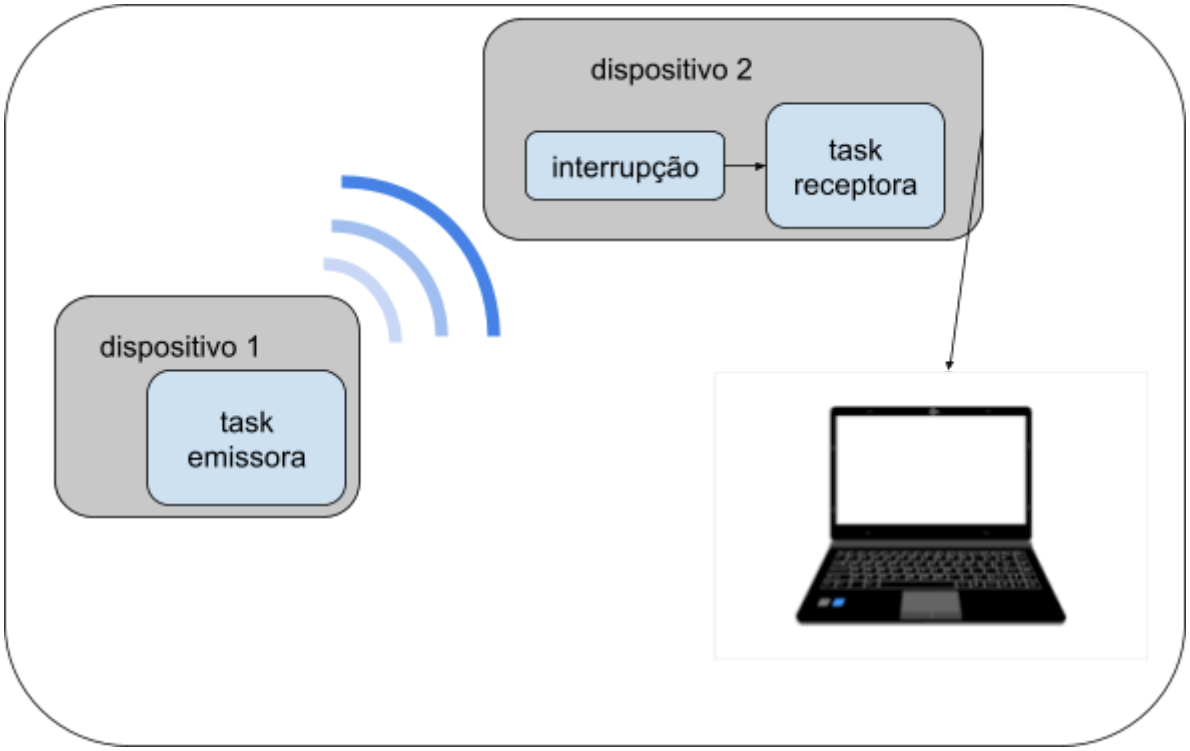

Figura 5.2.2: Ilustração das tarefas (Matheus A. de Lima,2023)

O ra-02 conta com um biblioteca para programação em C/C++ em conjunto com o framework arduino que foi desenvolvida inicialmente por Sandeep Mistry. Esta biblioteca pode ser usada em qualquer placa compatível com o mesmo, como por exemplo, placas da família esp32, esp8266 ou arduino. A biblioteca se chama arduino-LoRa, pode ser encontrada facilmente no github e é distribuída sob a licença MIT. Também é possível utilizar diretamente o protocolo SPI.

A biblioteca é bastante amigável e tem sua configuração e utilização muito bem documentadas. O setup do código para esp32 usando freeRTOS fica assim:

```
void setup()
{
 // Inicializar a porta serial
 Serial.begin(115200);
 Serial.println("Inicializado...");
 semaforoValorRecebido = xSemaphoreCreateMutex();
 // Inicializar o módulo LoRa RA-02
 LoRa.setPins(SS, RST, DI0);
 LoRa.setSpreadingFactor(12);

 if (!LoRa.begin(915E6))
 {
   Serial.println("Erro ao inicializar o módulo LoRa RA-02!");
 }
 // Configurar a função de callback para receber dados
 LoRa.onReceive(onReceive);
 // Habilitar a recepção de pacotes LoRa
 LoRa.receive();
}
```

Algoritmo 5.2.1 Setup do ra-02 (Matheus A. de Lima,2023)

Com a função “LoRa.setPins” configura-se os pinos do módulo ra-02 sendo eles: o pino de comunicação (ss), o pino de reset (rst) e o pino de interrupção (DI0). a função “LoRa.setSpreadingFactor” define o fator de espelhamento que será utilizada, quanto maior o fator mais lenta a comunicação e maior o alcance. Usando “LoRa.begin(915E6)” define-se a taxa de comunicação e inicia-se o módulo. Esta biblioteca conta com tratamento interno de interrupções através da função “LoRa.onReceive” que recebe como entrada uma função de callback que será então chamada toda vez que o módulo receber uma mensagem.

A função que trata o recebimento de mensagens neste testes se chama “onReceive” e sua implementação é a seguinte:

```
void onReceive(int packetSize)
{
 int valor;
 int diferenca = 0;
 if (packetSize == 0)
   return;
```

```
// Ler a mensagem recebida
String mensagem = "";
while (LoRa.available())
{
  mensagem += (char)LoRa.read();
}

// Exibir a mensagem recebida
Serial.println("Mensagem recebida: " + mensagem);

valor = get_valor_recebido(mensagem);

if (xSemaphoreTake(semaforoValorRecebido, (TickType_t)10) == pdTRUE)
{
  if(valor == 0){
    if(last_num != 60){
      exibe_perdas(perdas_ciclo);;
    }
    perdas_ciclo = 0;
    Serial.println("Inicializando ciclo...");
  }
  diferenca = valor - last_num;
  if (diferenca > 5)
  {
    perdas_ciclo += (diferenca - 5) / 5;
  }
  if(valor == 60){
    exibe_perdas(perdas_ciclo);
    perdas_ciclo = 0;
  }
  last_num = valor;
  xSemaphoreGive(semaforoValorRecebido);
}
}
```

Algoritmo 5.2.2 Função onReceive (Matheus A. de Lima, 2023)

É uma função simples que conforme recebe mensagens vai contabilizando quantas foram recebidas corretamente e quantas foram perdidas para posteriormente exibir estes dados.

A função começa usando parâmetro packetSize para verificar se existe algo a ser recebido, após isso usa uma laço while com a função “lora.available” para concatenar todos os dados da mensagem. Ao finalizar o recebimento da mensagem, extrai o número presente no corpo do texto com a função get_valor_recebido.

Dependendo da diferença entre o valor recebido e o último valor contabiliza as perdas e caso um ciclo tenha terminado, exibe as métricas do mesmo.

A função implementada do lado do emissor chama-se “lora_send_task” e a sua implementação é a seguinte:

```
void lora_send_task(void *pvParameters)
{
 int num_count = 0;
 Serial.println("lora_send_task");
 // Inicializar o módulo LoRa RA-02
 LoRa.setPins(SS, RST, DI0);
 LoRa.setSpreadingFactor(12);

 if (!LoRa.begin(915E6))
 {
  Serial.println("Erro ao inicializar o módulo LoRa RA-02!");
  while (1)
    ;
 }

 // Loop infinito de envio de dados
 while (1)
 {
  // Enviar uma mensagem
  String mensagem = "Test LoRa: " + String(num_count);
  LoRa.beginPacket();
  LoRa.print(mensagem);
  LoRa.endPacket();
  Serial.println("Enviando msg");
  num_count += 5;
  if(num_count > 60){
    num_count = 0;
  }
```

```
  // Esperar um tempo antes de enviar a próxima mensagem
  vTaskDelay(pdMS_TO_TICKS(5000));
}
}
```

Algoritmo 5.2.3 Função lora_send_task (Matheus A. de Lima,2023)

É uma implementação simples e amigável que conta com uma montagem de mensagem e um contador incremental. A configuração do módulo se dá da mesma forma que no receptor e o envio acontece através das funções “LoRa.beginPacket”, “LoRa.print" e “LoRa.endPacket” responsaveis por abrir a comunicação, enviar os dados e encerrar a comunicação respectivamente.

O emissor envia a uma taxa constante atraveś de um task periódica criada no “setup” da seguinte forma:

```
xTaskCreate(lora_send_task, "lora_send_task", 4096, NULL, 1, NULL);
```

Algoritmo 5.2.4 Criação da tarefa lora_send_task (Matheus A. de Lima,2023)

Os parâmetros de criação da task são, em ordem: a função a ser rodada, o nome da task, o tamanho da pilha em palavras, parâmetros a serem passados para a função definida no primeiro parâmetro e por último o handle da task que pode ser utilizado para interagir com a mesma de forma externa.

Esta task roda a função mostrada acima lora_send_task para realizar envios de forma periódica e utiliza o comando vTaskDelay para executar de forma periódica a cada cinco segundos.

Na realização dos testes com o RA-02 o fator de espalhamento foi alterado entre os valores de 7 e 12 a fim de aferir sua influência no alcance da comunicação.

Os resultados obtidos foram os seguintes:

Tabela 5.2.1 Resultados com o RA-02

Fator	Ponto B (70m)	Ponto C (222m)	Ponto D (400m)	Ponto E (158m)	Ponto F (199m)
6	recebimento sem perdas	-	-	-	-
8	recebimento sem perdas	-	-	58% perdas por ciclo e interferência	Recebimento inconsistente e interferência
10	Sem perdas	-	-	25% perdas por ciclo	50% perdas por ciclo e interferência
12	Sem perdas	Recebimento inconsistente	Recebimento ocasional e com presença de interferência	Sem perdas	Recebimento com perdas ocasionais

O recebimento é considerado inconsistente quando não ocorre de maneira que seja possível calcular com precisão o percentual de perdas, como quando, ao realizar 10 ciclos, apenas duas mensagens são recebidas no total. Apesar de haver recebimento, o número de perdas é tão significativo que o resultado é declarado inconsistente.

Por outro lado, o recebimento com perdas ocasionais acontece quando o percentual de perda é próximo a zero, mas ainda ocorrem perdas esporádicas, como, por exemplo, ao esperar dez ciclos e registrar apenas uma perda.

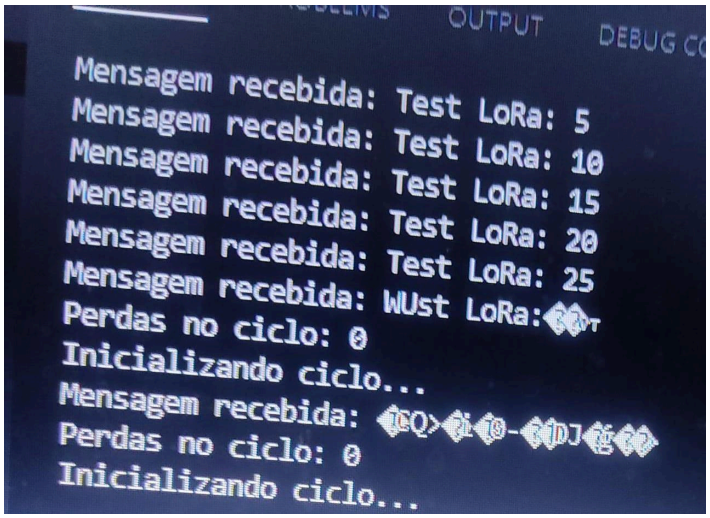

Figura 5.2.3: Exemplo de interferência (Matheus A. de Lima,2023)

É possível notar o ganho de alcance com o aumento do fator de espalhamento, bem como a influência que os obstáculos têm uma vez que linhas de visão mais abertas obtiveram resultados melhores. Interessante notar que ao atingir distâncias próximas do limite de transmissão a presença de interferência se torna mais constante, juntamente com as perdas.

Com base nos testes realizados, foi possível observar os obstáculos potenciais e sua influência no envio e recebimento do sinal.

Entre os obstáculos analisados, as linhas de transmissão com grande presença de edificações, como casas e construções, apresentaram o pior desempenho, resultando em uma redução significativa no alcance e na qualidade do sinal LoRa.

Além disso, as linhas que enfrentaram uma desvantagem em relação à altura do emissor, com o receptor posicionado em locais mais elevados, também enfrentaram dificuldades para estabelecer uma comunicação confiável.

Por outro lado, os obstáculos naturais, como árvores e vegetação, tiveram uma influência relativamente menor no alcance do sinal em comparação com os outros obstáculos mencionados.

Portanto, os resultados destacaram a importância de considerar e contornar esses obstáculos para garantir uma comunicação eficiente e confiável em sistemas LoRa.

5.3 Testes com E32-900T20D

O módulo E32-900T20D é um transceptor de rádio de longo alcance (LoRa) fabricado pela Ebyte. Ele é projetado para comunicações sem fio de baixa potência e longo alcance, usando a tecnologia LoRa, opera na faixa de frequência de 900 MHz e possui uma potência de transmissão ajustável de até 20 dBm, permitindo uma comunicação eficiente em distâncias maiores.

Utilizando a modulação LoRa, oferece alta sensibilidade e resistência a interferências, o que garante uma comunicação confiável mesmo em ambientes desafiadores.

O módulo utiliza uma interface serial UART para comunicação com dispositivos externos como microcontroladores, além disso o módulo é conhecido pelo seu baixo consumo de energia e facilidade de uso.

Figura 5.3.1: Dispositivo com E32-900T20D (Matheus A. de Lima,2023)

Diferente do ra-02 onde o fator de espalhamento podia ser ajustado para reduzir a taxa de transmissão e ganhar alcance, no E32-900T20D o parâmetro que é configurável é a taxa de transmissão.

O fator de espalhamento e a taxa de transmissão são inversamente proporcionais ou seja quanto maior o fator de espalhamento menor será a taxa de transmissão e por consequência maior será o alcance.

Por exemplo, em um sistema LoRa, um fator de espalhamento de 7 pode proporcionar uma taxa de transmissão de cerca de 11 kbps, enquanto um fator de espalhamento de 12 pode reduzir a taxa de transmissão para cerca de 293 bps. Vale notar que essa relação não é completamente linear e depende de outros fatores, como o hardware e os protocolos utilizados.

A taxa de transmissão do E32-900T20D pode assumir os seguintes valores: 0.3kbps, 1.2kbps, 2.4kbps (padrão), 4.8k, 9.6kbps e 19.2kbps.Foram realizados teste com os seguintes valores de 0.3, 2.4, 9.6 e 19.2.

A biblioteca utilizada anteriormente não cobre esta placa, para o e32 deve-se utilizar a biblioteca “LoRa_E32_Series_Library” desenvolvida inicialmente por Renzo Mischianti [12][28][29] ou utilizar o protocolo UART diretamente. A biblioteca é

bastante completa e implementa uma interface de configuração humanizada para facilitar o uso do hardware, bem como diversas funções para lidar com envio de mensagens e dados.

Utilizando essa biblioteca a inicialização do hardware para utilização de suas funcionalidades se dá da seguinte forma:

```
#define FREQUENCY_915

// Definir o pinout do módulo LoRa RA-02
byte RX_PIN = 17;  // Pino RX do módulo E32-900T20D conectado ao pino TX da ESP32
byte TX_PIN = 16;  // Pino TX do módulo E32-900T20D conectado ao pino RX da ESP32
byte M0_PIN = 4;   // Pino M0 do módulo E32-900T20D conectado a um pino digital da ESP32
byte M1_PIN = 5;   // Pino M1 do módulo E32-900T20D conectado a um pino digital da ESP32
byte AUX_PIN = 15; // Pino AUX do módulo E32-900T20D conectado a um pino digital da
ESP32

HardwareSerial mySerial(1); // RX, TX

LoRa_E32 e32Serial(TX_PIN, RX_PIN, &mySerial, AUX_PIN, M0_PIN, M1_PIN,
UART_BPS_RATE_9600, SERIAL_8N1); // Criação do objeto SoftwareSerial
```

Algoritmo 5.3.1 Definições do e32 (Matheus A. de Lima,2023)

Como pode-se notar é necessário definir a frequência de transmissão utilizada (primeira linha), também é necessário definir a pinagem e declarar um objeto do tipo “HardwareSerial”. O HardwareSerial é utilizado para nas implementações internas da biblioteca e recebe por parâmetro o número do par de pinos uart (rx e tx) que serão utilizados.

A configuração das opções da placa pode ser feita durante o “setup” da seguinte forma:

```
e32Serial.begin();

 ResponseStructContainer c;
 c = e32Serial.getConfiguration();

 Configuration configuration = *(Configuration *)c.data;
            .........

 configuration.SPED.airDataRate = AIR_DATA_RATE_000_03;
 ResponseStatus rs = e32Serial.setConfiguration(configuration,
WRITE_CFG_PWR_DWN_LOSE);
```

```
Serial.println(rs.code);
printParameters(configuration);
c.close();
e32Serial.setMode(MODE_2_POWER_SAVING);
```

Algoritmo 5.3.2 Inicialização do e32 (Matheus A. de Lima,2023)

Existem diversos parâmetros a serem configurados neste hardware e uma descrição breve dos mesmo foi apresentada na seção 2.7.2. Informações mais detalhadas podem ser encontradas na documentação da biblioteca.

Esta biblioteca não trabalha com interrupções de forma nativa, sendo assim, para utilizar recebimento via interrupção é necessário atrelar uma interrupção ao pino aux e definir uma task para tratar a interrupção, isso pode ser feito como abaixo:

```
pinMode(AUX_PIN, INPUT_PULLUP);
attachInterrupt(AUX_PIN, onReceive, RISING);

xTaskCreate(receiveLoop, "receiveLoop", 2048, NULL, 1, &receiveTaskHandle);
```

Algoritmo 5.3.3 Definição de interrupção (Matheus A. de Lima,2023)

Define-se o pino auxiliar como input e um função “onReceive” é cadastrada como rotina de tratamento dessa interrupção, o tratamento dos dados ocorre na task “receiveLoop” que será acordada pela interrupção. É necessário separar o tratamento da interrupção do recebimento da mensagem, para evitar estouro durante a interrupção. Como o hardware tem um buffer de entrada não há risco de perda de dados.

A implementação da função “onReceive” é a seguinte:

```
void IRAM_ATTR onReceive()
{
 xTaskResumeFromISR(receiveTaskHandle);
}
```

Algoritmo 5.3.4 Definição da função IRAM_ATTR (Matheus A. de Lima,2023)

Uma função pequena que se resume em colocar a task de tratamento dos dados recebidos de volta na fila de ready,Essa implementação é necessária, pois ao utilizar funções com tempos de execução maiores diretamente no tratamento de interrupções corre-se o risco de gerar erros no escalonamento do freeRTOS e provocar o reset da placa.

A função da task de tratamento dos dados recebidos “receiveLoop” é a seguinte:

```
void receiveLoop(void* pvParameters)
{
 ResponseContainer rs;
 int valor;
 int diferenca = 0;
 String message;

 while(1){
  vTaskSuspend(NULL);
  if (e32Serial.available() > 1)
  {
    rs = e32Serial.receiveMessage();

    message = rs.data; // First ever get the data
    Serial.println(message);
    if (message != "")
    {
      valor = get_valor_recebido(message);

      if (xSemaphoreTake(semaforoValorRecebido, (TickType_t)10) == pdTRUE)
      {

        if (valor == 0)
        {
          if (last_num != 60)
          {
            exibe_perdas(perdas_ciclo);
            ;
          }
          perdas_ciclo = 0;
          Serial.println("Inicializando ciclo...");
        }
        diferenca = valor - last_num;
        if (diferenca > 5)
        {
          perdas_ciclo += (diferenca - 5) / 5;
        }
```

```
        if (valor == 60)
        {
          exibe_perdas(perdas_ciclo);
          perdas_ciclo = 0;
        }
        last_num = valor;
        xSemaphoreGive(semaforoValorRecebido);
      }
    }
  }
}
}
```

Algoritmo 5.3.5 Definição da função receiveLoop(Matheus A. de Lima,2023)

Uma implementação bastante similar com a do ra-02 com apenas algumas mudanças devido ao uso de outra biblioteca. Importante notar a linha com o comando “vTaskSuspend(NULL)”, essa linha é responsável por retirar a task executante da fila de ready, ou seja essa task recebe um dado e sai da fila de ready até que seja colocada novamente nela através da interrupção

Importante notar que o pino auxiliar é utilizado para diversas funções neste hardware, além do recebimento de mensagens, ou seja, ao receber uma interrupção é necessário verificar se existem dados a serem recebidos. Esta verificação se dá com a seguinte comparação:

```
e32Serial.available() > 1
```

Algoritmo 5.3.6 Verificação de disponibilidade (Matheus A. de Lima,2023)

O teste foi realizado nos mesmo pontos que o teste anterior com o módulo ra-02 e o resultado foi o seguinte:

Tabela 5.3.1 Resultados com o E32

Taxa(kbps)	Ponto B (70m)	Ponto C (222m)	Ponto D (400m)	Ponto E (158m)	Ponto F (199m)
0.3	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas
2.4	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas
9.6	Sem perdas	-	-	média de 41% de perdas	Sem perdas
19.2	Sem perdas	-	-	média de 58% de perdas	média de 41% de perdas

Como o módulo se comportou bem com as duas taxas de transmissão mais baixas foi realizada uma nova rodada de testes com pontos mais longínquos para tentar determinar o limite do hardware. Os pontos podem ser vistos abaixo:

Figura 5.3.2: Locais da segunda rodada de testes (Matheus A. de Lima,2023)

O objetivo da escolha desses pontos foi permitir uma estimativa da distância máxima em uma rua(pontos x e y) e do poder de penetração do sinal em obstáculos massivos como construções(pontos w e z). O resultado deste teste foi o seguinte:

Tabela 5.3.2 Resultados extras do E32

Taxa (kbps)	Ponto X (346m)	Ponto Y (700m)	Ponto W (90m)	Ponto Z (178m)
0.3	recebimento sem perdas	recebimento sem perdas	recebimento sem perdas	perdas ocasionais
2.4	média de 41% de perdas	Sem recebimento	perdas ocasionais	sem recebimento

Nota-se que dos impactos encontrados em uma área urbana o que mais influencia no alcance são as construções. Essa afirmação se torna clara quando compara-se a distância entre os pontos “x” e “y” com a distância entre “w” e “z”, apesar de “w” e “z” estarem mais próximos um do outro o espaço entre ele é formado por construções(casas na maior parte) o que age como uma barreira drasticamente reduzindo o alcance do sinal.

5.4 Conclusão dos testes realizados

Com estes testes foi possível concluir que a segunda configuração de hardware, com o módulo E32-900T20D, se mostrou superior à configuração inicial com ra-02. Pode-se apontar que esta diferença pode ser causada pela mudança nas antenas, uma vez que a configuração original usava antenas bem menores e menos potentes.

É possível notar claramente o impacto dos diferentes tipos de obstáculos que são encontrados em áreas urbanas. A presença de vegetação por exemplo se mostrou uma barreira de baixo impacto uma vez que a perda de sinal em rotas com presença de vegetação foi baixa. Ao contrário, por sua vez, das rotas que tinham construções no meio do caminho como, por exemplo, os testes com os pontos “W” e

“Z”, Os materiais geralmente usados em construções de alvenaria são opacos às ondas de rádio, o que pode resultar em uma atenuação considerável do sinal e reduzir drasticamente o alcance. Além disso, diferenças de altura entre os dispositivos da rede afetam o alcance da comunicação. Quando esses dispositivos estão em elevações com uma diferença notável pode ocorrer perda de visibilidade direta entre eles, assim ocasionando sombreamento resultando em um alcance reduzido.

Apesar do E32-900T20D e o ra-02 compartilharem o mesmo chip de comunicação via rádio(SEM TECH's SX1276), o módulo da Ebyte traz uma série de melhorias como possibilidade de trabalhar em modo de economia de energia, comunicação ponto a ponto na camada física e uma entrada de antena sma-k padrão facilmente encontrada por diversos fornecedores, o que permite uma conexão fácil e simples com diversos modelos de antena. O ra-02 é uma implementação mais simples sem características extras.

Além do ganho em alcance vale destacar que durante os testes com E32-900T20D a presença de interferência foi praticamente nula tendo sido detectada apenas 5 vezes durante todos os testes mesmo quando próximo do alcance final, ou seja a comunicação com a segunda configuração de hardware é mais confiável e menos suscetível a interferências que modifiquem a mensagem. Tal diferença pode ser creditada a implementação de FEC no hardware utilizado, por sua vez o hardware ra-02 não apresenta nenhum tipo de controle ou detecção de erros durante a transmissão.

Considerando-se os pontos levantados e a análise dos testes realizados, o E32-900T20D se mostrou a melhor opção para a aplicação e foi a o modelo de chip transceptor escolhido. Além disso, tendo-se o objetivo de obter o maior alcance possível, a taxa de transferência de dados escolhida para o projeto foi de 0.3kbytes/s, e a fim de reduzir as perdas e interferências a implementação utiliza fec implementado pelo transceptor da eByte.

6 VISÃO GERAL DO SISTEMA

Após a identificação e análise dos desafios aqui expostos, assim como das tecnologias exploradas, testadas e validadas, foi possível definir a especificação para o sistema a ser implementado. A estrutura delineada consiste em três componentes fundamentais, cada um desempenhando um papel crucial na resolução dos problemas destacados, sendos estes:

1. **Rede de coleta de dados**

A infraestrutura da rede de coleta de dados constitui uma parte essencial do sistema, empregando dispositivos inteligentes compostos por microcontroladores do tipo ESP-32 conectados a transceptores LoRa, especificamente o modelo E32-900T20D, e aos sensores pertinentes. Essa rede opera com dois tipos distintos de dispositivos: os nodos finais (end nodes) e os gateways. Os nodos desempenham o papel crucial de coletar dados dos sensores e encaminhá-los adiante, enquanto os gateways assumem a responsabilidade de transmitir esses dados para o servidor e receber comandos do mesmo.

A missão primordial dessa rede é coletar e replicar os dados essenciais sobre a qualidade do solo no âmbito do agronegócio. Sensores integrados nos nodos capturam informações. Esses dados formam a base para uma compreensão abrangente, permitindo a implementação de práticas agrícolas mais eficientes e sustentáveis.

A comunicação entre os dispositivos é realizada por meio de ondas curtas, utilizando o LoRa, garantindo assim, a transmissão sem perdas até que os dados alcancem um gateway. Para assegurar a integridade desses dados durante a transmissão, são implementadas técnicas como checagem de paridade e FEC (Forward Error Correction), a fim de que a informação coletada seja entregue de maneira confiável e precisa ao servidor. Essa abordagem integrada da rede de coleta de dados constitui um elemento fundamental na obtenção e transmissão eficaz dos dados.

2. **Servidor**

O servidor consiste em uma API seguindo o padrão REST, desempenhando o papel de mediador e processador dos dados coletados na primeira camada da rede.

Suas funções abrangem atuar como um gateway para o banco de dados, recebendo e processando os dados antes de serem registrados, validando as informações e fornecendo dados relevantes aos sistemas conectados. Essa infraestrutura desempenha um papel crítico na integração e organização eficiente das informações coletadas pela rede, garantindo acesso rápido e seguro quando necessário.

Além disso, o servidor possibilita a comunicação entre a rede de coleta de dados e a interface gráfica, assegurando uma transmissão eficaz e segura dos dados para análise e visualização. Sua atuação como gateway, processador e sua integração com a interface gráfica é vital para a operação fluida e eficiente de todo o sistema, proporcionando uma gestão centralizada e segura dos dados.

3. Interface gráfica

A interface gráfica desempenha um papel crucial sendo a terceira camada do sistema, consumindo os dados coletados na primeira camada por meio de uma conexão direta com o servidor. Esta interface visual é projetada para oferecer uma representação acessível e compreensível dos dados, sendo a principal ferramenta de visualização e exportação dessas informações.

A interface gráfica fornece aos agricultores e profissionais do agronegócio uma visão das condições do solo, em tempo real. Ele apresenta gráficos e métricas de maneira visualmente atraente, possibilitando uma análise significativa dos dados coletados.

Além disso, a interface gráfica desempenha um papel essencial na tomada de decisões informada e eficiente, oferecendo uma plataforma centralizada para a interpretação rápida e precisa das informações agrícolas.

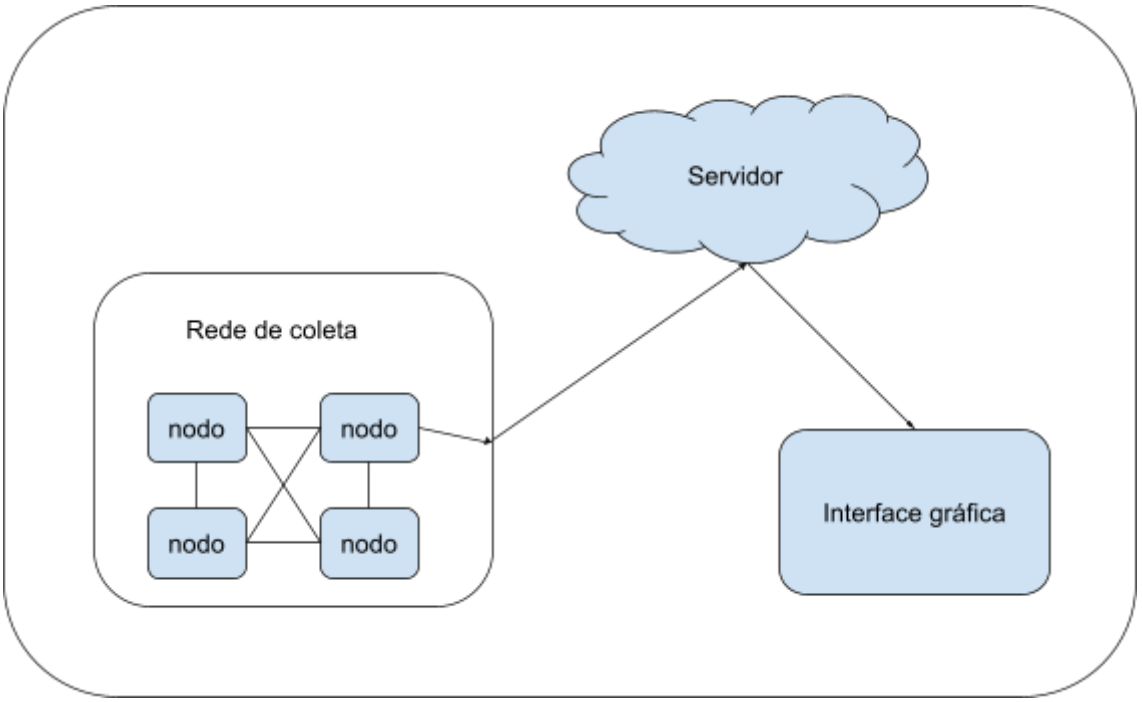

Figura 6.1: visão geral do sistema (Matheus A. de Lima, 2023)

Estas três partes, entrelaçadas e complementares, constituem um sistema integrado que atende aos objetivos específicos de cada componente, e também opera de maneira sinérgica para atingir os objetivos propostos para o sistema.

A rede de coleta de dados abastece o servidor com informações, sendo que este processa e armazena esses dados. Por sua vez, a Interface gráfica oferece uma interface acessível para interpretação rápida e embasada. Essa abordagem integrada visa otimizar a gestão agrícola, fomentando práticas mais sustentáveis e eficazes no âmbito do agronegócio.

7 IMPLEMENTAÇÃO

Neste capítulo serão apresentados os detalhes e passos da implementação do sistema proposto neste trabalho.

7.1 Tipos de dispositivos

Seguindo o padrão da arquitetura LoRaWan este sistema utiliza dois tipos de dispositivos sendo eles:

1. Dispositivo Gateway
2. Dispositivo EndNode

Os dispositivos do tipo gateway são responsáveis por se conectar com a internet e realizar as funções de comunicação com o servidor. Entre estas funções pode-se destacar o envio das leituras recebidas pelos endnode's, sincronização de horário da rede e envio de comando de configuração remota da rede.

Os dispositivos do tipo endnode são responsáveis por realizar a leitura periódica dos sensores neles conectados bem como enviá-las utilizando o sistema de roteamento e divisão de canais para que sejam processadas por um gateway e enviadas ao servidor remoto.

Destaca-se que a depender das configurações de roteamento um endnode pode receber leituras de outro endnode e enviá-las, seguindo sua rotina padrão de envio.

7.2 Verificação de integridade da mensagem

O sistema utiliza bits de paridade para garantir a integridade da mensagem. antes do envio ser realizado o sistemas calcula a paridade byte a byte e envia o resultado deste cálculo junto com a mensagem, o destinatário refaz o cálculo e confere se os valores batem com os recebidos antes de confirmar o recebimento.

```
// retorna 1 se for ímpar e zero se for par
unsigned int getParity(unsigned char text){
```

```
text ^= text >> 4;
text &= 0xf;
return ((0x6996 >> text) & 1);

}
```

Algoritmo 7.2.1 Verificação de paridade (Matheus A. de Lima,2023)

Para o cálculo do bit de paridade de um byte é usado o código acima, que trata-se de uma adaptação do código apresentado por Sean Eron Anderson da universidade de Stanford [30]. A grande vantagem está no baixo número de operações necessárias para se obter o resultado.

```
void calcMsgParity(defaultMsg *msg){
  int index_parity;
  //pode ir de 0 a 32
  for (int i = 0; i < (msg->sizeOfMessage); i++)
  {
    index_parity = i/8;
    if(getParity(msg->message[i]) == 1){
    msg->parity[index_parity] |= 1UL << (i-index_parity*8);
  }
}
```

Algoritmo 7.2.2 Definição da função calcMsgParity (Matheus A. de Lima,2023)

O emissor calcula os bits de paridade do conteúdo a ser enviado e os armazena em parte da mensagem.

```
bool checkMsgParity(defaultMsg msg){
  defaultMsg aux;
  aux = msg;
  calcMsgParity(&aux);
  if(aux.parity[0] == msg.parity[0] &&
     aux.parity[1] == msg.parity[1] &&
     aux.parity[2] == msg.parity[2])
  {
    return true;
  }else{
    return false;
```

```
  }
}
```

Algoritmo 7.2.3 Definição da função checkMsgParity (Matheus A. de Lima,2023)

O receptor por sua vez recebe a mensagem refaz o cálculo do conteúdo e o compara com os valores recebidos para garantir a integridade.

Ao se utilizar desta técnica é possível reduzir drasticamente os riscos de processar uma mensagem que teve seu conteúdo alterado, seja por interferência, choque de pacotes ou outro motivo.

A combinação de checagem por paridade via software e FEC via hardware faz com que o sistema se torne extremamente confiável à alterações de informações durante a comunicação.

7.3 Definição da topologia

Para definir a topologia da rede LoRawan e a disposição dos dispositivos é necessário aferir a distância segura de comunicação, isto é, definir o alcance de comunicação máximo onde não ocorrem interferências constantes ou perdas significativas. É necessário definir uma distância segura para assegurar que a necessidade de reenvio das mensagens seja mínima assim diminuindo o risco de perda de informações ou congestionamento da rede.

Nos testes realizados foi aferido um alcance seguro por volta de 700 metros em zona urbana utilizando uma taxa de transmissão de 0,3 Mbits, nesta configuração foi possível atingir esse alcance sem a presença de perdas ou interferência o que caracteriza uma distância segura uma vez que está abaixo do limite possível . Vale ressaltar que o cenário dos testes é bastante hostil à comunicação e causa uma perda significativa de alcance.

A distância segura é usada para definir o espaçamento entre os dispositivos na rede. É necessário realizar testes de alcance com o hardware no local de implantação para otimizar ao máximo possível o layout de disposição dos dispositivos pois um mesmo terreno pode apresentar obstáculos como relevo e

vegetação de forma localizada, assim requerendo que em uma dada parte os dispositivos sejam posicionados a uma distância menor.

Considerando que existem dois tipos de dispositivos na solução apresentada: dispositivos finais(microcontroladores com sensores) e gateways(dispositivos com acesso a internet dedicados a enviar dados ao servidor remoto). É necessário definir uma distribuição dos mesmos de modo que cubra-se a maior área possível com a menor necessidade possível de sinal de internet, isto é, torna-se necessário definir uma topologia que conte com poucos pontos de acesso a internet(gateways) e cubra uma longa área de extensão utilizando a comunicação por ondas de rádios como intermediário

Uma possibilidade é a topologia em estrela, “Em uma rede de computadores organizada por meio de uma topologia estrela,toda a informação gerada pelas estações de trabalho deve passar por um nó central.” (Macedo, Ricardo Tombesi e col, REDES DE COMPUTADORES, 2018). No caso da solução aqui abordada teria-se um nó central e todos os dispositivos finais deveriam estar ao alcance do mesmo se comunicando diretamente com ele.

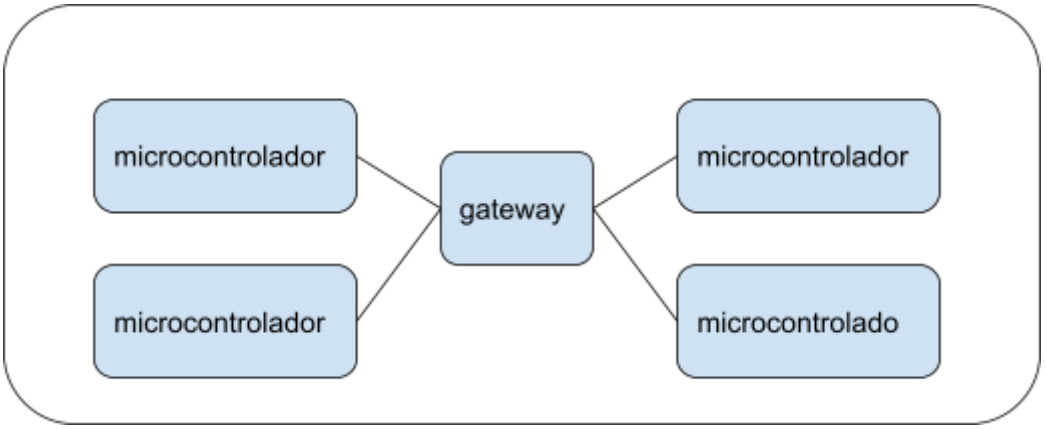

Figura 7.3.1: primeira opção de topologia (Matheus A. de Lima, 2023)

A necessidade de se comunicar diretamente como o gateway se torna um fator limitante das implementações práticas do sistema uma vez é necessário manter todos os dispositivos finais a uma distância segura do gateway assim tornando as possibilidades de disposição limitadas. Essa limitação acaba tornando difícil a implementação de uma rede de coleta de dados extensa, uma vez que o melhor

alcance possível só será alcançado com o gateway no centro geográfico da área de coleta, nos cenários onde este ponto central não tem conectividade a área de coleta total é altamente reduzida.

Uma solução para este problema da topologia estrela é o uso de repetidores de sinal que visam aumentar o alcance útil de comunicação. Estes dispositivos são responsáveis por receber um sinal e enviá-lo novamente assim aumentando o seu alcance total. Esta solução apesar de válida não é a mais otimizada para o problema apresentado, uma vez que os dispositivos realizam as leituras de sensores de forma periódica o tráfego de rede é baixo, sendo assim pode-se utilizar os próprios dispositivos para transmitir os sinais.

Implementando um sistema de retransmissão onde um dispositivo final envia para outro dispositivo final até que um gateway seja alcançado é possível resolver o problema de alcance possibilitando uma área de coleta maior sem a necessidade de múltiplos gateways, outra vantagem é um aumento na liberdade de posicionamento do gateway.

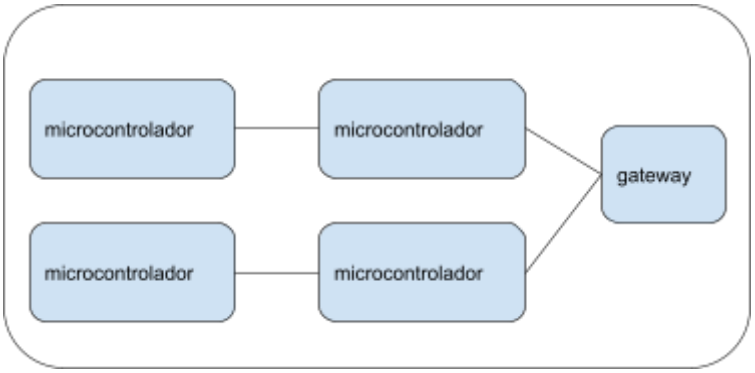

Figura 7.3.2: segunda opção de topologia (Matheus A. de Lima, 2023)

Como as leituras e envios são realizados de forma periódica a uma taxa na ordem das dezenas de minutos é possível dividir essa carga de forma a prevenir possíveis congestionamentos na rede LoRa e utilizar os dispositivos intermediários na rota como retransmissores enquanto os mesmos encontram-se ociosos.

O tipo de topologia escolhida apesar de muito semelhante com a topologia em malha ainda apresenta um nodo central(gateway) por onde a comunicação deve, de forma obrigatória. Pode-se defini-la como uma rede híbrida que apresenta tantos características de um rede em estrela quando de uma rede em malha

Existe ainda a necessidade de evitar que os dispositivos enviem mensagens ao mesmo tempo. Quando dois ou mais dispositivos realizam envios simultâneos na mesma faixa de frequência ou em faixas muito próximas existe a possibilidade de interferência mútua e choque de pacotes.Choque de pacotes é um fenômeno que ocorre quando dois dispositivos enviam dados na mesma faixa de frequência para o mesmo receptor, o sinal acaba sendo embaralhado o que resulta em dados corrompidos.

O módulo escolhido para a aplicação, E32-900T20D, conta como um sistema de canais de comunicação o que permite segmentar a comunicação em diversas faixas de frequência podendo desta forma reduzir a probabilidade de choque de pacotes, uma vez que passam a trafegar em faixas de frequência diferentes. O módulo tem uma configuração de canais que varia entre os valores 00H-45H ou seja um total de 70 canais disponíveis. É importante notar que, pelas características da comunicação LoRa em que os dados são modulados em frequência, canais muito próximos podem acabar interferindo um no outro.

Para usar a comunicação segmentada em diversos canais é preciso que os dispositivos que irão receber e retransmitir estejam no mesmo canal do emissor e que após receberem a mensagem a ser repassada ajustem o seu canal de comunicação e enviem para o próximo dispositivo na rota até o gateway. Uma vez que um dispositivo opera em apenas um canal por vez e pode trocar de canal a qualquer momento que não esteja transmitindo ou recebendo, é possível ajustar o canal de comunicação para otimizar o fluxo da rede.

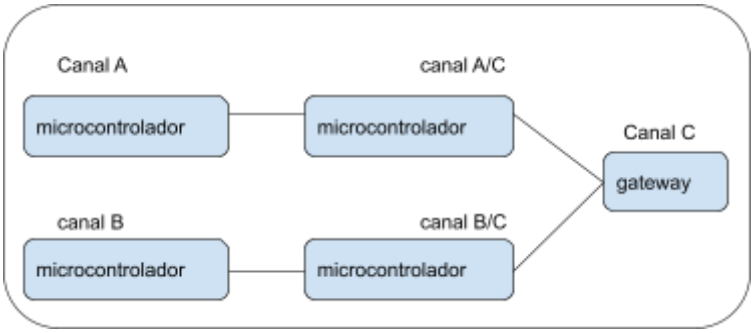

Figura 7.3.3: segunda opção de topologia com segmentação de canais (Matheus A. de Lima, 2023)

A ilustração 3 mostra um exemplo de topologia segmentada em canais, onde os dispositivos no nível mais distante operam em um único canal(canal A e B), e por

se encontrarem no nível mais distante da rota, apenas transmitem para os dispositivos de nível intermediário Estes por sua vez operam em dois canais, sendo um de recebimento e um de transmissão. Após receber uma mensagem ou realizar uma leitura, os dispositivos intermediários ajustam o seu canal para o canal do gateway ou do próximo nível intermediário na rota, caso existam múltiplos níveis intermediários.

O nível intermediário pode de acordo com a necessidade ter múltiplos pulos de retransmissão dependendo da cobertura desejada. É necessário apenas garantir que não sejam usados canais com frequências muito próximas em dispositivos que se encontram geograficamente próximos a fim de evitar interferência mútua.

Ao se aplicar estas regras de divisão na comunicação por meio da retransmissão, a topologia passa a apresentar também algumas características de um rede hierárquica, tornando assim uma rede de topologia híbrida baseada na topologia estrela e que apresenta características das topologias de malha e árvore(hierárquica).

Para este projeto a topologia usada é a apresentada na Figura 7.3.3 com retransmissão de dados em múltiplos níveis separada em canais. Optou-se por essa topologia por ela se adequar ao objetivo e atender todos os requisitos estipulados para o projeto, possibilitando cobrir grandes áreas sem necessitar de um gateway próximo e conseguindo dividir a carga de modo a minimizar congestionamentos de rede ou choque de pacotes.

7.4 Definição do roteamento

Uma vez definida a topologia é necessário então definir as rotas nesta topologia, bem como, a distribuição dos dispositivos que compõem a rede. Esta distribuição precisa se dar de tal forma que os dispositivos possam operar tirando máximo proveito do alcance possível com a comunicação LoRa, além de evitar possíveis pontos de congestionamento e manter uma distância segura de dispositivos que usem canais próximos a fim de evitar interferências.

A distribuição geográfica dos dispositivos se dá de acordo com a necessidade de cobertura e a distância segura aferida previamente. Sendo assim é preciso definir

de que forma serão configuradas as rotas após a colocação dos dispositivos. Existem duas formas possíveis de configurar o roteamento deste tipo de rede

1. Roteamento automático
2. Roteamento estático

Em um primeiro momento o roteamento automático parece ser a melhor solução, apresenta a vantagem de poder recalcular as rotas sozinho, se adaptar a mudanças na rede e otimizar as rotas ao longo do tempo. Porém, o algoritmo necessário para esta implementação é extremamente complexo e de difícil validação.

Por se tratar de uma rede via ondas de rádio com retransmissão em que os dispositivos responsáveis pela comunicação também realizam outras atividades, o caminho mais curto nem sempre será a melhor opção, uma vez que pode causar choque de pacotes e gargalos no recebimento. Sendo assim, além de calcular o caminho mais curto, é necessário levar em conta quais canais usar e como evitar congestionamento em nodos que fazem partes de múltiplas rotas. Essas características especiais tornam difícil a implementação de algoritmos de roteamento conhecidos como o Dijkstra's Shortest Path Algorithm [31] uma vez que esse não as prevê.

Dessa forma se torna interessante utilizar o roteamento estático que pode ser facilmente testado e validado e cuja implementação é muito menos custosa.Para evitar travamento e perda de dados, caso nodos intermediários sejam danificados, pode-se utilizar tabelas de roteamento onde cada nodo tem “n” opções de rotas e ao perceber que uma destas não está respondendo pode utilizar a próxima da lista.

Desta forma se dá o roteamento do sistema, cada estação possui um arquivo de rotas inicialmente carregado com “n” rotas possíveis previamente definidas. A estação tem a direção de cada uma destas rotas, seu endereço e seu canal. A primeira rota definida é tratada como rota principal e em um cenário sem erros o nodo sempre envia para ela. Para garantir que as mensagens sejam entregues de forma confiável e que reenvios desnecessários sejam evitados foi implementado um sistema de confirmação de recebimento via ack, ou seja, ao receber uma mensagem o nodo ou gateway receptor trata os dados e envia um ack ao emissor confirmando o

recebimento .Caso ocorram erros no envio, ou seja, caso ocorra o envio de dados e o ack não venha a estação incrementa um contador de controle e ao atingir o limiar configurado é realizada a reordenação das rotas. Na reordenação a rota que atingiu o limite de envios sem ack é colocado no final da lista e a segunda rota da lista passa a ser a primeira.

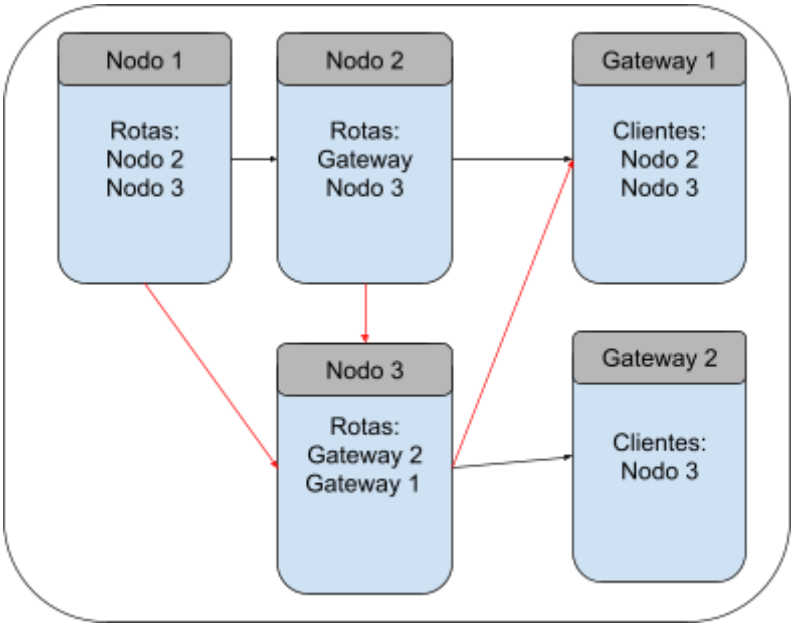

Figura 7.3.4: Exemplo de malha com tabela de rotas (Matheus A. de Lima, 2023)

Na figura 7.3.4 é apresentado um exemplo de malha hipotética e suas tabelas de rotas. As setas pretas representam a rota principal de envio dos dados e as vermelhas representam as rotas secundárias disponíveis. No caso de falhas consecutivas no envio de dados, ou seja, envios sem recebimento de ack os nodos reordenam suas rotas e começam os envios novamente. É válido notar que apesar de esse exemplo demonstrar apenas duas rotas por nodo esse número pode ser maior e deve variar de acordo com as necessidades da implantação do sistema.

7.5 Estrutura das tasks

A rede de coleta de dados é baseada no sistema operacional FreeRTOS, projetado para sistemas embarcados em tempo real. Estruturado em tasks e interrupções, o FreeRTOS permite a execução simultânea de tasks críticas, garantindo eficiência na gestão de operações concorrentes. As tasks, operando como unidades independentes, são dedicadas a tarefas específicas, otimizando a distribuição de recursos e possibilitando uma resposta rápida a eventos em tempo real, como a coleta imediata de dados dos sensores. A presença de interrupções oferece um mecanismo ágil para lidar com eventos críticos, contribuindo para a eficiência e confiabilidade do sistema em tempo real, adequando-se aos desafios dinâmicos da coleta de dados.

As tasks no nodo final desempenham papéis cruciais no processo de coleta e transmissão de dados. São elas: "vReceiveLoop", "vReadTmpSnsrLoop" e "vSendDataLoop".

A primeira, "vReceiveLoop", é implementada com base nos testes do E32-900T20D, apresentando um loop infinito de recebimento que utiliza a instrução “vTaskSuspend” do FreeRTOS para remover temporariamente a task da fila de execução ("ready"), sendo reintegrada, quando necessário pela interrupção no pino auxiliar. Esta task trata mensagens recebidas, diferenciando entre mensagens para retransmissão e mensagens de ack. No caso de retransmissão, a mensagem é adicionada à fila de transmissão, e um ack é disparado para o emissor. No caso de recebimento de um ack, a mensagem correspondente é removida da fila de envio, pois a mesma já encontra-se entregue. Além disso, os controles de horário do último ack e número de tentativas de envio são atualizadas.

A segunda task, "vReadTmpSnsrLoop", realiza leituras periódicas dos sensores conectados à placa, adicionando os dados na fila de envio para subsequente processamento e transmissão. Sua periodicidade é ajustável conforme a aplicação.

A terceira task, "vSendDataLoop", lê a fila de dados a serem enviados, transmitindo o primeiro da fila para o próximo nodo da rota até o gateway. Essa task opera de maneira periódica, com uma taxa constante. Caso a fila de envio esteja vazia, ela ajusta seu período de execução para um valor maior, evitando assim

processamento desnecessário. Além disso, essa task controla o número de tentativas mal sucedidas, trocando a rota principal por outra, caso venha a ultrapassar um limite predefinido de envios sem sucesso. Isto ocorre somente quando configuradas múltiplas rotas para o nodo. Esse controle contribui para a eficiência e confiabilidade do processo de envio de dados.

Esta implementação está ilustrada na figura 7.5.1.

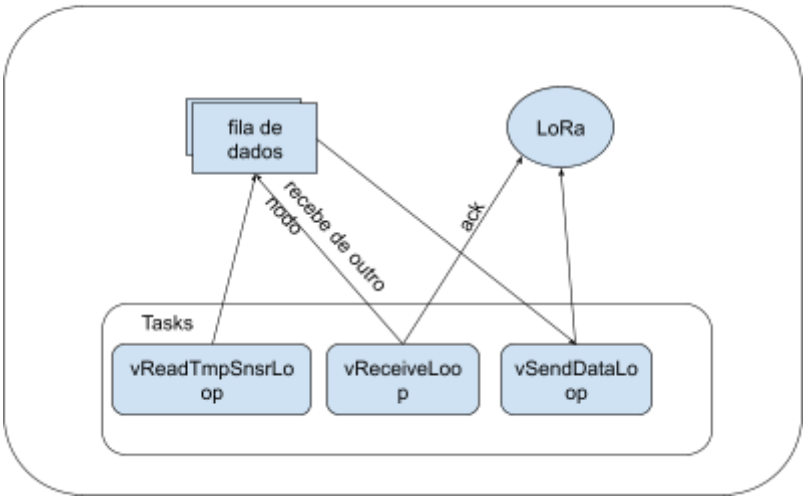

Figura 7.5.1: Modelo de tasks do nodo (Matheus A. de Lima, 2023)

O gateway tem um estrutura de tasks semelhante ao nodo final, consistindo também de três tasks, sendo elas: “vReceiveLoop”, ”vSendToServerLoop” e “vUpdateTime”.

A primeira task “vReceiveLoop” é igual a implementada no nodo final e trata o recebimento das mensagens para envio ao servidor.

A segunda task “vSendToServerLoop” assemelha-se com a task “vSendDataLoop” do nodo final, tendo como diferença o fato de que envia para o servidor através de uma chamada http do tipo POST ao invés de utilizar o protocolo LoRa como os nodos finais.

A task “vUpdateTime" consiste em uma task periódica, com período de execução grande, que serve para sincronizar o tempo da rede de coleta com o tempo do servidor.

A implementação do gateway se encontra representada na figura 7.2.2.

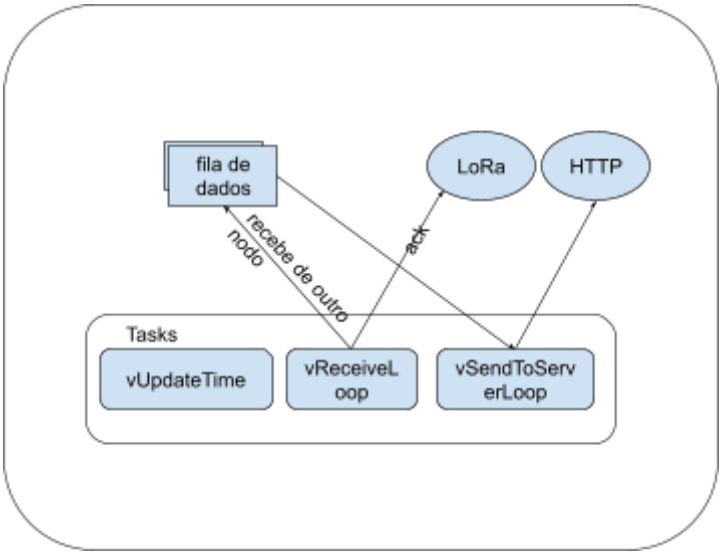

Figura 7.5.2: Modelo de tasks do gateway (Matheus A. de Lima, 2023)

7.6 Banco de dados

A seleção do MySQL como o banco de dados para o sistema em desenvolvimento foi cuidadosamente ponderada, com base em sua robustez, facilidade de uso e amplo conjunto de recursos. Essa escolha foi impulsionada pela reputação consolidada do MySQL em lidar eficientemente com grandes volumes de dados, tornando-o particularmente adequado para os requisitos específicos do projeto.

Além disso, a popularidade do MySQL e a presença de uma comunidade ativa de usuários foram consideradas vantagens significativas. A comunidade ativa, significa acesso a uma rica fonte de conhecimento e soluções compartilhadas, promovendo uma resposta rápida a desafios específicos. O conjunto de instruções simples porém extenso do MySQL fornece as ferramentas necessárias para a aplicação.

A estrutura do banco de dados foi organizada conforme a figura 7.6.1.

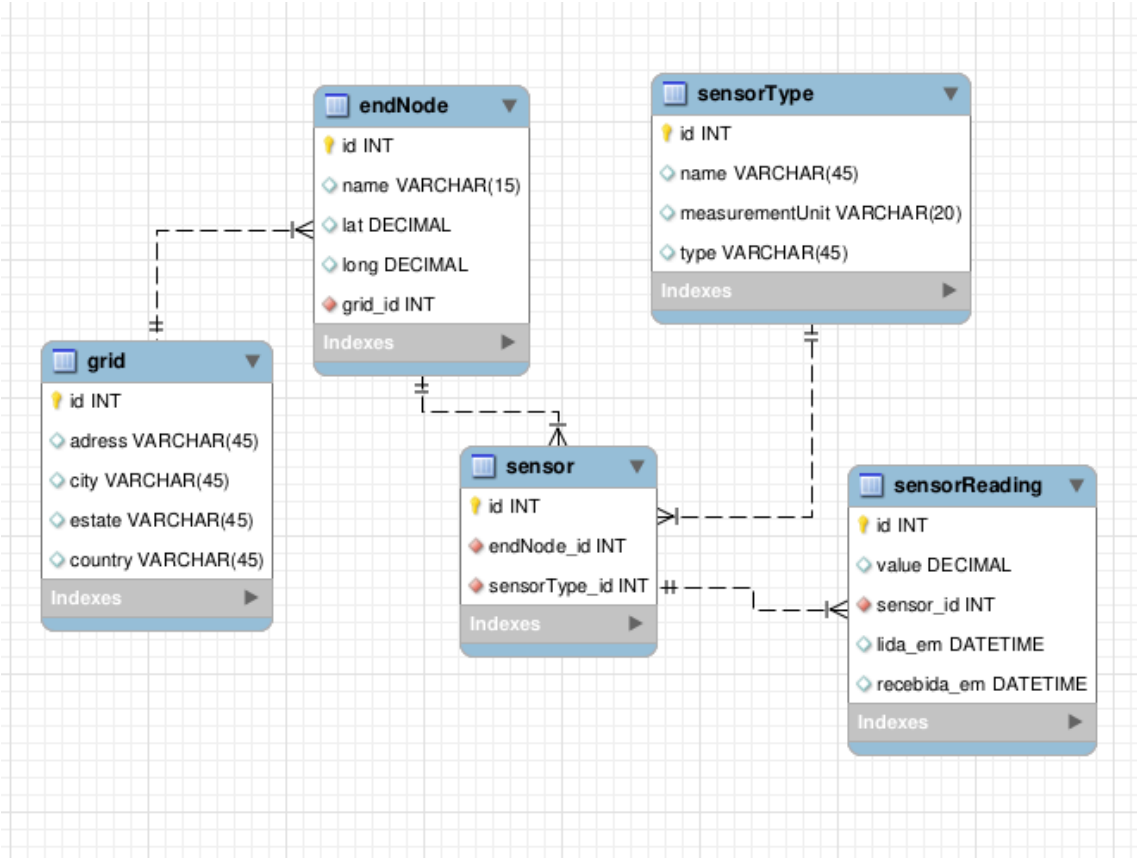

Figura 7.6.1: Modelo do banco de dados (Matheus A. de Lima, 2023)

A arquitetura do banco de dados é composta por cinco tabelas distintas, cada uma desempenhando um papel específico no armazenamento e organização dos dados. A tabela "endNode" é responsável por armazenar os registros de todos os nodos presentes na malha de coleta. Essa tabela é essencial para a manutenção de dados sobre os dispositivos de coleta, consolidando informações cruciais sobre cada nodo da rede.

A tabela "grid" possui dados relacionados à malha de coleta em si, estabelecendo relações entre os diferentes nodos que compõem uma malha específica. Cada nodo é atribuído a uma malha de coleta, permitindo a representação estruturada das relações entre nodos e malhas no sistema.

A tabela "sensorType" armazena informações sobre os tipos de sensores configurados no sistema, incluindo dados como nome, unidade de medida e tipo. Essa tabela é fundamental para a categorização e descrição detalhada dos sensores utilizados no ambiente de coleta de dados.

A tabela "sensor" estabelece a relação entre os tipos de sensores e os dispositivos de coleta. Cada dispositivo de coleta pode estar conectado a múltiplos sensores, criando uma associação flexível entre os dispositivos e os tipos de sensores utilizados na rede.

Por fim, a tabela "sensorReading" registra as leituras realizadas pelos sensores, conectando-as aos dispositivos de coleta e aos tipos específicos de sensores. Essa tabela é crucial para manter um histórico detalhado das leituras, permitindo uma análise aprofundada das informações coletadas ao longo do tempo. Em conjunto, essas tabelas formam uma estrutura coesa que suporta a complexidade e os requisitos específicos do sistema de coleta de dados.

7.7 Servidor

O servidor foi desenvolvido em Node.js seguinte o padrão rest para desenvolvimento de apis e utilizando o framework Express.js e o ORM Sequelize para manipular o banco de dados

A adoção do padrão REST (Representational State Transfer) para o desenvolvimento de APIs promove a construção de um serviço web padronizado e de fácil compreensão. Facilitando a interoperabilidade entre diferentes componentes do sistema, incluindo o servidor, dispositivos de coleta e a interface de visualização dos dados, proporcionando uma comunicação consistente e eficaz.

A escolha do framework Express.js para o desenvolvimento do servidor em Node.js é respaldada pela sua leveza e simplicidade. Express.js fornece uma camada mínima e flexível para construir aplicações web e APIs, permitindo uma configuração rápida e fácil roteamento de requisições HTTP.

Diferentemente de outros frameworks, o Express.js fornece um conjunto básico de ferramentas focando na configuração e gerenciamento de rotas, deixando a cargo do desenvolvedor os detalhes de implementação de outras partes necessárias da aplicação, como por exemplo, o banco de dados. Conta com um extenso repositório de bibliotecas e uma comunidade ativa.

A utilização do ORM (Object-Relational Mapping) Sequelize oferece uma camada de abstração para interagir com o banco de dados MySQL. Essa escolha simplifica a manipulação de dados no banco, permitindo que as operações sejam realizadas em termos de objetos e classes em vez de consultas SQL diretas,

melhorando a legibilidade do código, e tornando o sistema mais resistente a vulnerabilidades relacionadas à injeção de SQL.

Em conjunto, o uso de Node.js, Express.js e Sequelize cria uma base tecnológica coesa, oferecendo agilidade no desenvolvimento, escalabilidade e facilidade de manutenção.

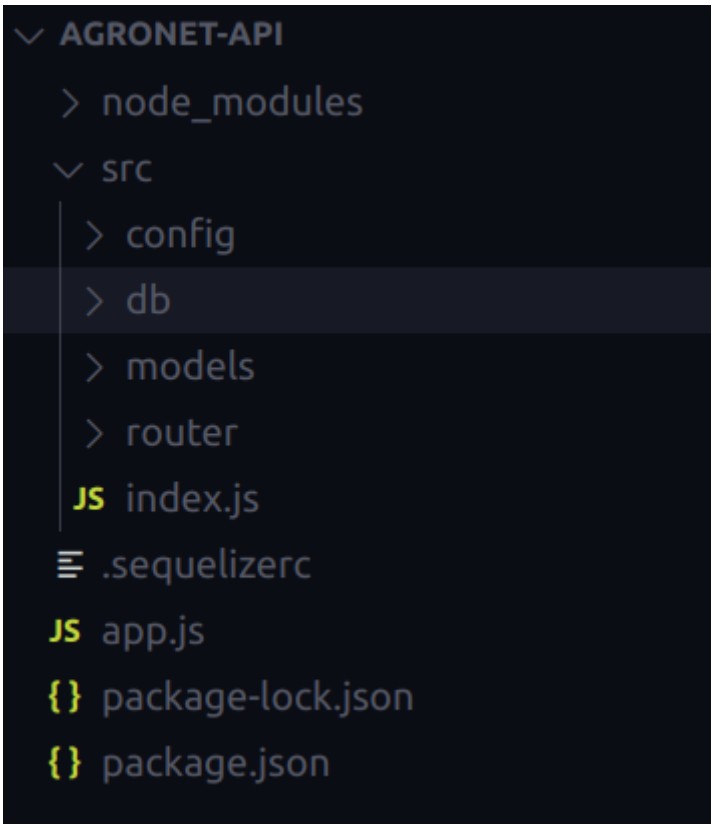

Figura 7.7.1 estrutura de pastas da API (Matheus A. de Lima, 2023)

A estrutura do servidor está dividida em duas classes de arquivos, models e routers. Arquivos do tipo model apresentam a representação abstrata dos elementos do banco de dados e as funções de manipulação direta dos mesmos utilizando sequelize. Os arquivos do tipo router representam as rotas(endereços) disponíveis para a aplicação e as ações de tratamento de cada uma.

A partir da API é possível cadastrar, editar, consultar e excluir informações referentes a aplicação e os dados coletados.

7.8 Interface gráfica

Para o desenvolvimento da interface gráfica foi utilizado o Grafana.Grafana é uma plataforma de análise e visualização de dados de código aberto, projetada para

integrar-se a uma variedade de fontes de dados e fornecer painéis interativos e gráficos visualmente atraentes. Desenvolvida em Go e JavaScript, o Grafana é altamente modular e extensível, permitindo que os usuários criem interfaces gráficas personalizadas para monitoramento e análise de dados em tempo real.

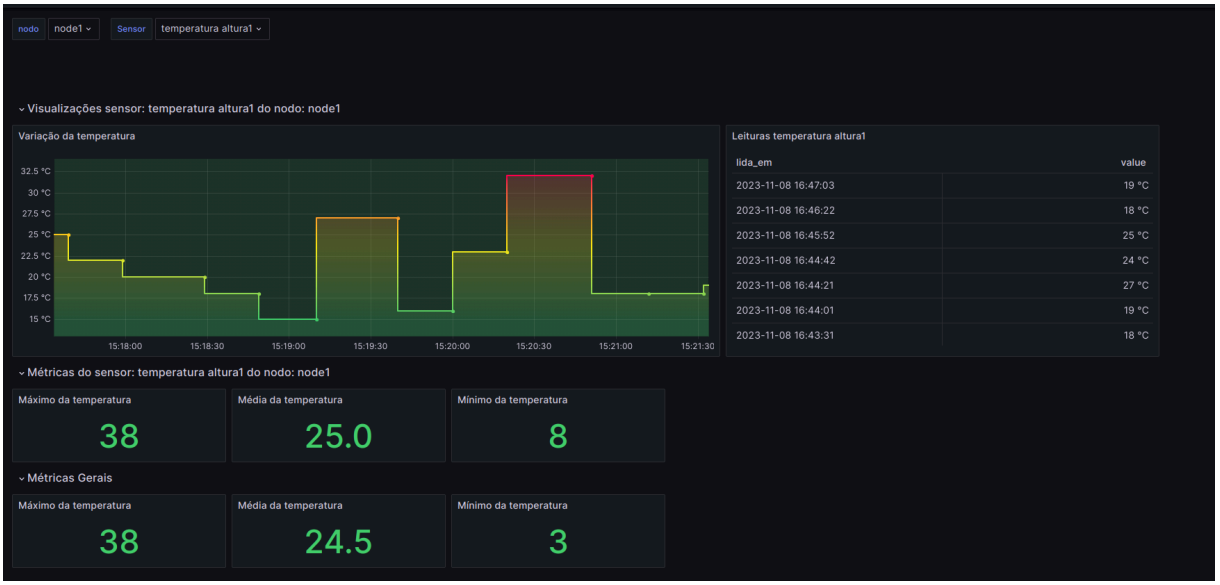

Figura 7.8.1: Visão geral da interface(Matheus A. de Lima, 2023)

O design da interface é estruturado de maneira a oferecer uma experiência intuitiva para os usuários que desejam analisar os dados. Uma característica central desta interface é a presença de uma barra de seleção, fornecendo aos usuários a capacidade de definir variáveis específicas relacionadas aos dados que estão visualizando. Essa funcionalidade permite uma personalização dinâmica, adaptando a exibição conforme as necessidades específicas do usuário ou os parâmetros de análise em questão. Na barra de seleção é possível selecionar o nodo a ser visualizado bem como o sensor deste nodo. Além disso, fornece uma seleção de período de tempo bastante abrangente e customizável.

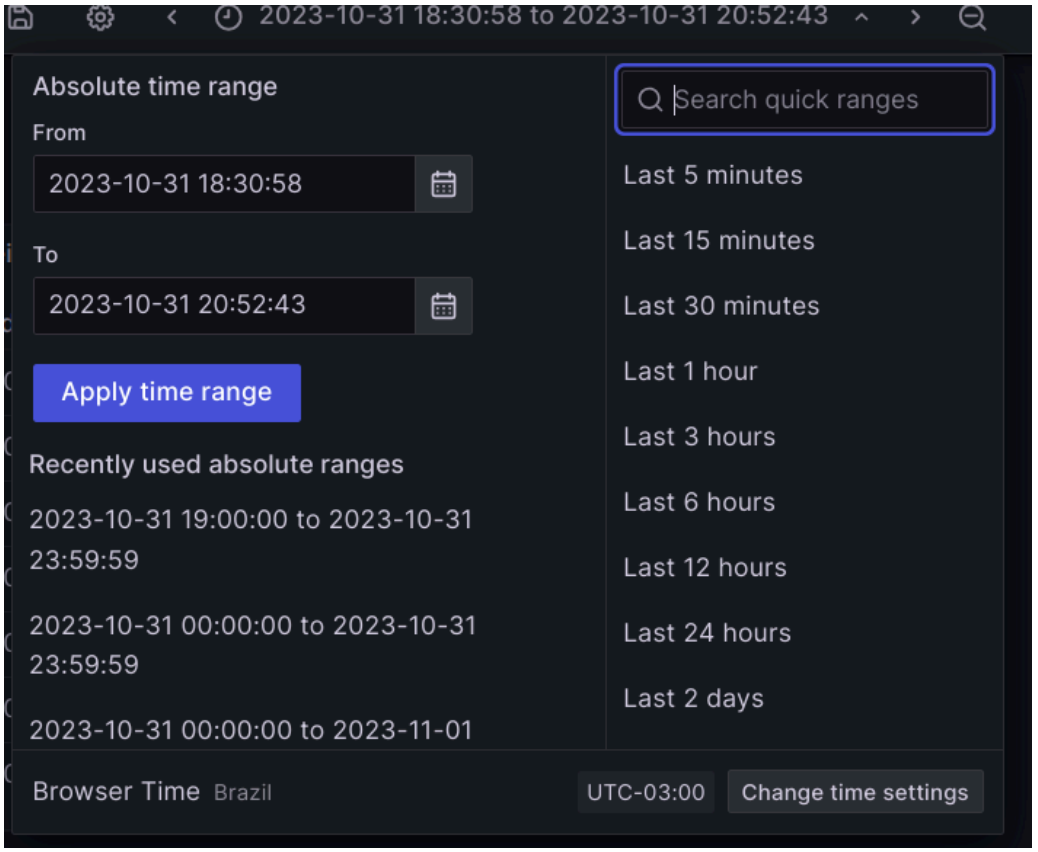

Figura 7.8.2: Seleção de tempo na interface gráfica(Matheus A. de Lima, 2023)

Dentro da seção dedicada às visualizações do sensor, são apresentados elementos visuais em tempo real. Um gráfico oferece uma representação visual das mudanças nas variáveis selecionadas, possibilitando a identificação rápida de tendências ou padrões. Complementando o gráfico, uma tabela fornece uma visão tabular dos dados, detalhando informações específicas de maneira organizada. Essa abordagem híbrida de visualização permite que os usuários escolham entre representações visuais e dados em formato tabular, proporcionando flexibilidade para diferentes preferências de análise.

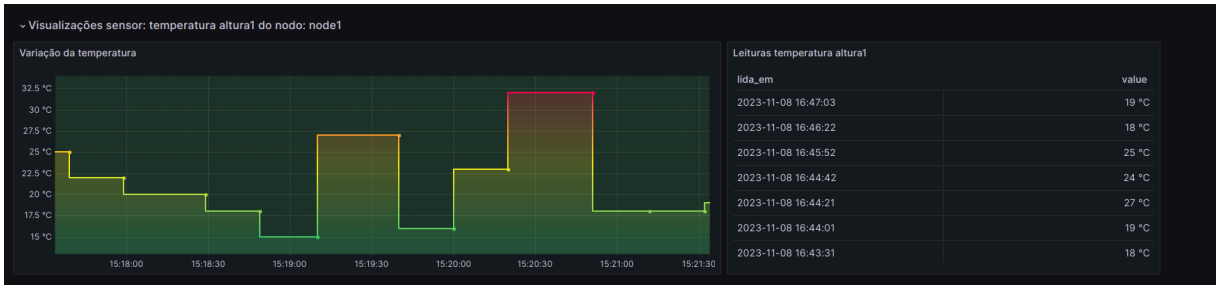

Figura 7.8.3: Seção dedicada às visualizações do sensor (Matheus A. de Lima, 2023)

Uma seção adicional destaca dados específicos da seleção atual, oferecendo algumas informações sobre as variáveis escolhidas. Essa área da interface fornece contextos específicos e detalhes importantes sobre as condições ou métricas selecionadas.

Além disso, a interface inclui uma seção dedicada a dados gerais, que oferece uma visão abrangente, incluindo dados gerais do intervalo selecionado.

Figura 7.8.4: Sessões de métricas (Matheus A. de Lima, 2023)

8 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Neste trabalho buscou-se compreender as complexidades da integração de sistemas embarcados para a coleta de dados, mas, também, apresentar uma solução eficaz para os problemas aqui apresentados.

O desenvolvimento do sistema embarcado distribuído para a coleta de dados de sensores, baseado na tecnologia LoRa, representa uma solução eficiente não apenas para o campo agrícola mas também para outros cenários que necessitam coletar dados em regiões de interesse. A escolha da tecnologia LoRa foi fundamentada na sua capacidade de oferecer comunicação de longo alcance, baixo consumo de energia e adequação a ambientes rurais.

A topologia planejada e implementada neste projeto, combinando elementos de rede malha e estrela, oferece uma solução para os desafios associados à coleta de dados em ambientes agrícolas extensos. A abordagem em malha proporciona uma cobertura ampla, permitindo a comunicação distribuída entre os dispositivos de coleta, enquanto a estrutura em estrela concentra eficientemente os dados nos gateways centralizando e otimizando os recursos de processamento e conexão com a internet. A segmentação por canais atua como uma medida estratégica para mitigar interferências entre os dispositivos, garantindo uma transmissão de dados mais confiável.

A arquitetura do sistema, composta por uma rede de coleta de dados, servidor e interface, foi projetada para atender aos objetivos específicos de cada componente, garantindo uma coleta de dados confiável e uma visualização acessível para os usuários. A integração entre essas partes cria um sistema robusto e confiável que atende aos objetivos propostos.

A utilização do FreeRTOS como sistema operacional na rede de coleta de dados cria uma estrutura eficiente, garantindo a execução simultânea de tarefas críticas. A implementação de tasks e o uso de interrupções nessa rede contribui para a eficiência e confiabilidade do sistema em tempo real.

A escolha do MySQL como banco de dados, juntamente com a infraestrutura Node.js, express.js e Sequelize para o servidor, garante uma manipulação robusta e eficiente dos dados coletados. A integração do Grafana como interface gráfica proporciona uma representação visual e acessível das condições do solo em tempo

real, que pode ser usada para facilitar a tomada de decisões pelos agricultores e profissionais do agronegócio.

O sistema proposto se mostra bastante versátil, podendo ir além de sua aplicação agrícola inicial. Desenvolvido para a coleta de dados em larga escala, sua arquitetura modular e topologia híbrida malha-estrela são capazes de fornecer a adaptabilidade necessária para se estender a diversas aplicações. Essa flexibilidade é ancorada na capacidade do sistema de facilitar uma comunicação eficiente entre dispositivos de coleta e um gateway central, independentemente do contexto específico.

8.1 Pontos fortes do sistema

O sistema apresentado revelou-se eficaz, alcançando integralmente os objetivos delineados. A escolha da topologia, uma rede híbrida malha-estrela com segmentação por canais, demonstrou ser uma solução ideal para superar os desafios inerentes à coleta de dados em ambientes agrícolas extensos. A modularidade da arquitetura permitiu uma fácil adaptação às necessidades específicas, garantindo uma comunicação eficiente entre os dispositivos de coleta e o gateway central. A eficácia do sistema é evidenciada pela precisão na coleta de dados sobre a qualidade do solo, e também pela sua versatilidade.

O sistema oferece potencial para uma ampla variedade de aplicações. A flexibilidade e modularidade da estrutura permitem personalizar o sistema para diferentes contextos, bastando modificar o período do RTOS e as funções encarregadas da coleta de dados dos sensores. A API e a interface, por sua vez, foram desenvolvidos como componentes genéricos e adaptáveis, oferecendo a capacidade de serem utilizados e customizados para outras finalidades com um esforço de desenvolvimento substancialmente reduzido.

8.2 Limitações do sistema

O sistema tem como principal limitação a taxa de coleta de dados que para um bom funcionamento deve ser da ordem de alguns minutos no mínimo, essa limitação se deve pela necessidade de múltiplos pulos para o envio até o gateway e da necessidade de transmitir com uma banda curta, sem comprometer o alcance.

Para sistemas onde dados devem ser coletados múltiplas vezes por minuto o alcance precisa ser sacrificado ou tecnologias mais rápidas devem ser utilizadas. Apesar disso existem formas de se remediar esse problema, caso seja necessário monitorar a uma taxa de varias vezes por minuto para emitir alertas, por exemplo, pode-se suprimir o envio dos dados, caso os mesmo não tenham tido alteração significativa e enviar a uma taxa menor para o servidor, caso seja detectada uma alteração significativa, realiza-se o envio fora da taxa periódica.

8.3 Trabalhos futuros

Este trabalho apresentou o desenvolvimento de um sistema distribuído para a coleta de dados de sensores em áreas de grande extensão e baixa infraestrutura, mas há ainda diversas áreas a serem melhoradas e exploradas dando continuidade ao que foi aqui desenvolvido.

Uma área importante para trabalhos que seria capaz de agregar muito no sistema como um todo envolve a otimização contínua da eficiência energética do sistema. Estratégias para reduzir o consumo de energia durante a transmissão de dados e a investigação de alternativas de energia, como a incorporação de tecnologias de geração de energia, como por exemplo o uso de energia solar, podem ser exploradas para garantir um funcionamento sustentável e de longa duração.

Outro caminho interessante para pesquisas futuras é a exploração de técnicas de análise de dados. A implementação de algoritmos de aprendizado de máquina pode aprimorar a capacidade do sistema de fornecer dados preditivos sobre a qualidade do solo, contribuindo para uma gestão agrícola mais proativa.

A versatilidade do sistema desenvolvido revela-se na sua capacidade de adaptação a diferentes contextos além da agricultura. Uma aplicação promissora é sua extensão para atender às necessidades específicas de uma equipe de biólogos da Uergs, que busca monitorar dados de temperatura relacionada aos ninhos de aves durante o período de desova. O sistema, concebido inicialmente para a coleta de dados na agricultura, destaca-se pela sua modularidade e independência do tipo de dados transportados.

A atual iniciativa de adaptação para o monitoramento de ninhos de aves ilustra como o sistema pode ser configurado para atender a requisitos específicos de diferentes domínios científicos. Nesse cenário, ajustes na configuração do sistema, como a definição do período de coleta de dados e a personalização das métricas a serem monitoradas, são efetuados para se alinhar com as necessidades dos biólogos. Essa flexibilidade destaca a aplicabilidade multifacetada da implementação realizada, e também realça seu potencial como uma ferramenta para monitoramento em ambientes biológicos e ecológicos. Essa expansão para novos domínios destaca a adaptabilidade do sistema, provando sua eficiência em áreas além da agricultura.

Essas sugestões para trabalhos futuros destacam a natureza dinâmica e a expansibilidade contínua do sistema desenvolvido, apontando para inovações e aprimoramentos, à medida que se avança em direção à soluções mais sofisticadas e abrangentes para a coleta de dados na agricultura e em outras aplicações.

REFERÊNCIAS

Castro, N. R. (2022, Outubro 5). Afinal, Quanto o Agronegócio Representa no PIB Brasileiro? CEPEA USP.

Serigati, F. (2013, Fevereiro). A agricultura puxa o PIB? Agroanalysis.

Valin, H., Sands, R. D., van der Mensbrugghe, D., Nelson, G. C., Ahammad, H., Blanc, E., … Willenbockel, D. (2013). The future of food demand: understanding differences in global economic models. Agricultural Economics, 45(1), 51–67

Khanna, A., & Kaur, S. (2019). Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. Computers and Electronics in Agriculture, 157, 218–231.

Espressif Systems. (1 de setembro de 2016). "ESP32 Overview". Consultado em 1 de abril de 2023.

Espressif Systems. (6 de março de 2017). "ESP32 Datasheet" (PDF). Consultado em 14 de março de 2023.

Shenzhen Ai-Thinker Technology Co., Ltd. (2017). Ra-02 LoRa Product Specification V1.1. Consultado em 1 de abril de 2023.

PlatformIO. Your Gateway to Embedded Software Development Excellence. Disponível em: https://docs.platformio.org/en/latest/ Acesso em: 15 de abril de 2023.

Arduino. (Data de acesso: março de 2023). Arduino Documentation. Disponível em: https://www.arduino.cc/en/Guide/HomePage

FreeRTOS. "FreeRTOS Real Time Operating System." Disponível em https://www.freertos.org/. Data de acesso: março de 2023.

Montagny, Sylvain. All you need to know about LoRaWAN, in 40 mins. YouTube, 2 de jun. de 2022 Disponível em: https://www.youtube.com/watch?v=Bsue0PzNRDU. Acesso em: 15 de novembro de 2023.

Mischianti, Renzo. LoRa E32 device for Arduino, esp32 or esp8266: settings and basic usage disponível em (https://mischianti.org/2019/10/15/lora-e32-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-part-1/) acesso em 10/05/2023

Salcedo-Parra, O. J., & Agudelo-Cristancho, N. G. (2021). RSSI performance of LoRa, BLE, and WiFi sensor nodes in an interoperable IoT system. Universidad Nacional de Colombia, Bogota - Colombia, Universidad Distrital Francisco Jose de Caldas, Bogota - Colombia.

Khan, F. U., Awais, M., Rasheed, M. B., Masood, B., & Ghadi, Y. (2021). A Comparison of Wireless Standards in IoT for Indoor Localization Using LoPy

Haxhibeqiri, J., De Poorter, E., Moerman, I., & Hoebeke, J. (2018). A Survey of LoRaWAN for IoT: From Technology to Application. IDLab, Department of Information Technology at Ghent University—IMEC, 9052 Ghent, Belgium.

Faber, M. J., Zwaag, K. M. vd, Dos Santos, W. G. V., Rocha, H. R. O., Segatto, M. E. V., & Silva, J. A. L. (2020). A Theoretical and Experimental Evaluation on the Performance of LoRa Technology. IEEE Sensors Journal

Node.js. Node.js JavaScript runtime. Disponível em: https://nodejs.org/ acesso em março de 2023

Carneiro Neto, J. A. (2020). Um mapeamento de práticas em projetos de APIs REST (Trabalho de Graduação). Universidade Federal de Pernambuco, Centro de Informática, Recife.

MySQL. (s.d.). MySQL Documentation. Recuperado de https://dev.mysql.com/doc/ acesso em março de 2023

Sequelize. (s.d.). Sequelize Documentation. Recuperado de https://sequelize.org/ acesso em março de 2023

Grafana. (s.d.). Grafana Documentation. Recuperado de https://grafana.com/docs/ acesso em março de 2023

Rachmani, A. F., & Zulkifli, F. Y. (2018). Design of IoT Monitoring System Based on LoRa Technology for Starfruit Plantation. TENCON 2018 - 2018 IEEE Region 10 Conference.

Lee, H.-C., & Ke, K.-H. (2018). Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation. IEEE Transactions on Instrumentation and Measurement, 67(9), 2177–2187.

Bouras, C., Gkamas, A., Kokkinos, V., & Papachristos, N. (2019). Using LoRa Technology for IoT Monitoring Systems. 2019 10th International Conference on Networks of the Future (NoF).

Montagny, Sylvain. All you need to know about LoRaWAN, in 40 mins. YouTube, 2 de jun. de 2022 Disponível em: https://www.youtube.com/watch?v=Bsue0PzNRDU. Acesso em: 15 de novembro de 2023.

Wang, Y., Wang, Y., Qi, X., & Xu, L. (2009). OPAIMS. Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems - CASES ’09. doi:10.1145/1629395.1629428

Matté, Junior. (2022). Aplicação da Modulação LoRa ao Monitoramento Geotécnico. UFRGS

Mischianti, Renzo. Ebyte LoRa E32 device for Arduino, esp32 or esp8266: fixed transmission – 4 disponível em (https://mischianti.org/2019/11/10/lora-e32-device-for-arduino-esp32-or-esp8266-fixed-transmission-part-4/) acesso em 10/05/2023

Mischianti, Renzo. Ebyte LoRa E32 device for Arduino, esp32 or esp8266: power saving and sending structured data – Part 5 disponível em (https://mischianti.org/2019/12/03/lora-e32-device-for-arduino-esp32-or-esp8266-power-saving-and-sending-structured-data-part-5/) acesso em 10/05/2023

Sean Eron, Anderson. Bit Twiddling Hacks, Stanford computer graphics laboratory disponível em (https://graphics.stanford.edu/~seander/bithacks.html) acesso em 25/08/2023

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

Jiannqi, L., Chunyan, A., Linchang, M., Yuandong, L., & Qingjiang, H. (2020). Splitting data and forward error correction methods for solving the interference in LoRa networks. 2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)

Celso Maciel da Costa. (2010). Sistemas Operacionais. Programação Concorrente com Pthreads. EdiPUCRS

Providello, M. V. (2020). Qualidade do solo para a cultura do milho. Universidade Estadual Paulista, Faculdade de Ciências Agrárias e Veterinárias, Campus de Jaboticabal.