**UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA**
**ESCOLA SUPERIOR DE TECNOLOGIA - EST**

**GABRIEL VICTOR LIMA VALLE**

# MONITORAMENTO E OTIMIZAÇÃO DE MONOCULTURAS EM ESTUFAS POR MEIO DA TECNOLOGIA LORAWAN

Manaus
2023

**GABRIEL VICTOR LIMA VALLE**

# MONITORAMENTO E OTIMIZAÇÃO DE MONOCULTURAS EM ESTUFAS POR MEIO DA TECNOLOGIA LORAWAN

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

Orientador: Edgard Luciano Oliveira da Silva, Dr.

Manaus

2023

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

*Reitor:*
***André Luiz Nunes Zogahib***
*Vice-Reitor:*
***Kátia do Nascimento Coureiro***
*Diretor da Escola Superior de Tecnologia:*
***Jucimar Maia da Silva Júnior***
*Coordenador do Curso de Engenharia Elétrica:*
***Israel Gondres Torné***

*Banca Avaliadora composta por:* *Data da Defesa: 28/09/2023.*
***Prof. Edgard Luciano Oliveira da Silva, Dr*** *(Orientador)*
***Prof. Fábio de Sousa Cardoso, Dr***
***Prof. Angilberto Muniz Ferreira Sobrinho, Dr***

## CIP - Catalogação na Publicação

Valle, Gabriel Victor Lima

Monitoramento e Otimização de Monoculturas em Estufas por meio da Tecnologia LoRaWAN / Gabriel Victor Lima Valle; [orientado por] Prof. Edgard Luciano Oliveira da Silva, Dr - Manaus: 2023.

85 f. p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2023.

1. Internet das Coisas. 2. Sistemas de Telecomunicações. I. da Silva, Edgard Luciano Oliveira.

**GABRIEL VICTOR LIMA VALLE**

# MONITORAMENTO E OTIMIZAÇÃO DE MONOCULTURAS EM ESTUFAS POR MEIO DA TECNOLOGIA LORAWAN

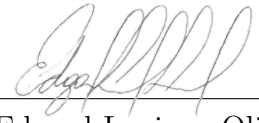
Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Nota obtida: 9,9(Nove pontos e nove décimos)

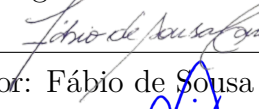
Aprovado em 28/09/2023

Área de concentração: Engenharia de Telecomunicações

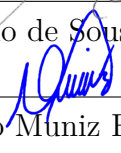
BANCA EXAMINADORA



Orientador: Edgard Luciano Oliveira da Silva, Dr.



Avaliador: Fábio de Sousa Cardoso, Dr.



Avaliador: Angilberto Muniz Ferreira Sobrinho, Dr.

Manaus

2023

**Dedicatória**

Dedico este trabalho à comunidade acadêmica e à sociedade, com o propósito de fomentar os alicerces do conhecimento em prol do bem-estar coletivo e impulsionar o avanço em direção a um futuro promissor, visando a glória de Deus, pois dEle, por Ele e para Ele são todas as coisas.

## AGRADECIMENTOS

À Deus, pois aprouve a Ele em seu conselho, mediante sua soberania e providência, me sustentar e guiar, colocando as pessoas certas em meu caminho, sem as quais não seria possível realizar este trabalho.

Agradeço à minha família por sempre me incentivar, e especialmente, aos meus pais, Paulo e Bianca, que não mediram esforços para investir em minha educação e me proporcionar o melhor que podiam.

Agradeço aos meus colegas de trabalho Iury Batalha e Yuri Barbosa, por terem me orientado e ajudado a descobrir e escolher o tema deste trabalho.

Agradeço ao meu colega de trabalho Paulo Santos, que altruisticamente se propôs a me ajudar com o hardware do meu dispositivo de sensoriamento.

Agradeço aos meus colegas egressos, David Freitas e Danilo de Souza Frazão, que me emprestaram equipamento, tempo e disposição para me ajudar a confeccionar o sistema deste trabalho. Além disso, quero destacar a influência significativa de seus empenhos e trabalhos na área de LoRa, os quais serviram como modelos e fontes de inspiração fundamentais para a elaboração deste ensaio. Seus conhecimentos e conquistas foram uma bússola essencial nesta jornada acadêmica, e por isso, meu singelo agradecimento.

Agradeço ao professor Edgard Luciano, que se prontificou a ser meu orientador e me dar novos horizontes, que sem dúvidas foi indispensável para eu chegar até aqui. Sua orientação e apoio foram essenciais para meu percurso até este ponto, não apenas me impulsionou, mas também desempenhou um papel fundamental na minha jornada acadêmica.

## RESUMO

Na criação e manutenção de estufas e floriculturas, condições ideais são cruciais para o crescimento das culturas. A Agricultura 4.0, combinando tecnologias avançadas com práticas agrícolas, apresenta desafios no monitoramento dessas condições. Luminosidade, temperatura, umidade do solo e concentração de $CO_2$ são fundamentais, mas monitorá-las é desafiador. A solução proposta é a união do microcontrolador *ESP32* com a rede *LoRaWAN*. O *ESP32*, com sensores específicos, coleta dados em tempo real sobre os fatores ambientais. A rede *LoRaWAN* permite a transmissão eficiente dos dados a um nó central conectado à internet, possibilitando o acesso remoto aos dados. Isso viabiliza um monitoramento contínuo e preciso, alertando agricultores sobre desvios críticos e permitindo estudos. A abordagem é acessível economicamente, tornando-a vantajosa para estabelecimentos menores. Em suma, a combinação do *ESP32* com *LoRaWAN* oferece uma solução de baixo custo, confiável e eficaz para otimizar o crescimento de culturas, minimizando perdas e maximizando a eficiência dos recursos, alinhando-se com os princípios da Agricultura 4.0.



**Palavras chave**: *LoRaWAN*, *ESP32*, Sensores, Estufas, Agricultura 4.0.

## ABSTRACT

*In the creation and maintenance of greenhouses and flower farms, ideal conditions are crucial for crop growth. Agriculture 4.0, combining advanced technologies with agricultural practices, presents challenges in monitoring these conditions. Light, temperature, soil moisture, and $CO_2$ concentration are essential factors, but monitoring them is challenging. The proposed solution is the integration of the ESP32 microcontroller with the LoRaWAN network. The ESP32, equipped with specific sensors, collects real-time data on environmental factors. The LoRaWAN network enables efficient transmission of data to a central node connected to the internet, allowing remote access to the data. This enables continuous and precise monitoring, alerting farmers to critical deviations and facilitating studies. The approach is economically feasible, making it advantageous for smaller establishments. In essence, the combination of ESP32 with LoRaWAN offers a cost-effective, reliable, and efficient solution to optimize crop growth, minimizing losses and maximizing resource efficiency, aligning with the principles of Agriculture 4.0.*



***Keywords**: LoRaWAN, ESP32, Sensors, Greenhouses, Agriculture 4.0.*

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas, Siglas e Símbolos

| | |
|---|---|
| ADC | Analog to Digital Converter |
| APT | Advanced Packaging Tool |
| CS | Chip Select |
| CSS | Chirp Spread Spectrum |
| DAC | Digital to Analog Converter |
| DHT | Digital Temperature and Humidity Sensor |
| ESP32 | Espressif System Platform 32 |
| GPIO | General Purpose Input/Output |
| HTTP | Hypertext Transfer Protocol |
| I2C | Inter-Integrated Circuit |
| IP | Internet Protocol |
| IoT | Internet of Things |
| JSON | JavaScript Object Notation |
| LDR | Light Dependent Resistor |
| LoRa | Long Range Wide Area |
| LoRaWAN | Long Range Wide Area Network |
| MISO | Master In Slave Out |
| MOSI | Master Out Slave In |
| MQTT | Message Queuing Telemetry Transport |
| NDIR | Non-Dispersive Infrared |
| PWM | Pulse Width Modulation |
| RTD | Resistence Temperature Detector |
| SBC | Single Board Computer |
| SCL | Serial Clock |
| SDA | Serial Data |
| SPI | Serial Peripheral Interface |
| TTN | The Things Network |
| WiFi | Wireless Fidelity |

**SUMÁRIO**

# INTRODUÇÃO

No processo de criação e manutenção de estufas e floriculturas, são necessárias aferições e estudos a respeito das condições em que as culturas que se pretende ou já estão sendo cultivadas se encontram. Dentre os fatores essenciais que favorecem o crescimento dessas culturas, quatro se demonstram determinísticos e críticos: a luminosidade, temperatura, umidade do solo e concentração de gás carbônico no ambiente. Cada espécime de planta existente possui sua própria quantidade adequada de cada componente dessas para realizar suas funções vitais e sobreviver, de forma que a falta ou excesso de cada uma delas pode prejudicar seu crescimento e acabar matando a planta. Contudo, realizar essas medições, estudar o local e até mesmo conhecer a necessidade de cada vegetal pode ser um trabalho complexo que por muitas vezes, estabelecimentos dessa natureza de médio e pequeno porte não possuem ferramentas para tal análise, o que acarreta perdas de recursos biológicos e monetários. Dessa forma, uma solução precisa, confiável e de baixo custo se faz necessária para auxiliar nesta tarefa.

Diante da necessidade de um sistema confiável e de baixo custo para o monitoramento das condições ambientais em estufas e floriculturas, há diversas opções disponíveis no mercado. No entanto, uma das alternativas mais viáveis e econômicas é a combinação de um microcontrolador ESP32 com uma infraestrutura de LoraWAN.

O microcontrolador ESP32 oferece uma solução flexível pois é uma plataforma eletrônica de código aberto que permite a criação de projetos customizados de acordo com as necessidades específicas do projetista ou aplicação. Neste caso, com o uso de sensores conectados a um ESP32, é possível coletar dados sobre luminosidade, temperatura, umidade do solo e concentração de gás carbônico no ambiente de cultivo de forma precisa e em tempo real. Esses sensores podem ser escolhidos de acordo com as necessidades específicas de cada espécie vegetal, evitando perdas significativas de recursos biológicos e financeiros. Por sua vez, a infraestrutura de LoRaWAN (Long Range Wide Area Network) é uma rede de comunicação de longo alcance, especialmente projetada para dispositivos IoT (Internet of Things) de baixo consumo de energia. Ela permite a transmissão de dados em grandes distâncias, possibilitando a cobertura de áreas extensas, como estufas e floriculturas. Além disso, o protocolo LoRaWAN é conhecido por ser de baixo consumo de energia, o que é crucial para dispositivos alimentados por bateria, como os sensores do ESP32.

Ao combinar o ESP32 com sensores a uma infraestrutura LoRaWAN, os dados coletados pelos sensores podem ser transmitidos de forma wireless para um nó central. Esse nó central, por sua vez, pode ser conectado à internet, permitindo o acesso remoto aos dados coletados através de uma plataforma online ou aplicativo dedicado. Com essa solução, é possível realizar medições precisas e contínuas das condições ambientais nas estufas e floriculturas sem a necessidade de um monitoramento manual constante. Os agricultores e responsáveis pelo cultivo podem receber alertas em tempo real caso algum dos fatores

críticos esteja for a dos parâmetros adequados para determinada espécie vegetal, bem como podem ser feitos estudos com dados computados ao longo de um período e como a variação desses fatores afeta cada espécie. Isso possibilita uma intervenção rápida e eficiente, considerações e conhecimentos aprofundados sobre cada espécie e a diminuição da perda de recursos.

Além disso, uma das vantagens significativas dessa solução é a sua viabilidade econômica. A utilização do ESP32 como microcontrolador e nó é acessível, com uma gama de opções de modelos e preços. De igual forma, a infraestrutura LoRaWAN é um protocolo de comunicação de baixo custo, permitindo uma cobertura eficiente em áreas extensas sem a necessidade de grandes investimentos em infraestruturas de rede.

Portanto, neste trabalho será realizada a confecção deste sistema que permitirá a medição dos parâmetros essencias na criação de culturas e serão apresentados os recursos de hardware e software que permitaram chegar no resultado esperado.

# 1 REFERENCIAL TEÓRICO

Para a elaboração da possível solução para a problemática identificada no contexto das estufas é necessário perscrutar as etapas de forma sistemática e dividi-las em blocos para ter uma noção exata das fases do processo. Primeiramente, começa-se entendendo o funcionamento das estufas agrícolas e em seguida os processos por meio dos quais as plantas e vegetais crescem.

Posteriormente, analisa-se a maneira de como a platorma ESP32 pode ser de grande utilidade para a medição dos parâmetros estudados na primeira etapa, bem como a justificativa para a utilização dessa tecnologia em vez das demais encontradas no mercado.

Como o intuito da obtenção dos dados medidos é poder monitora-los, estuda-los e extrair informações úteis para otimização, é necessário transmiti-los para uma central de dados através de uma rede de telecomunicações, que neste caso, trata-se da tecnologia LoRaWAN, que permite a transmissão dessas informações por meio de frequências isentas de taxas.

Uma vez com uma rede que permite a transmissão das informações coletadas, essas medições são enviadas para uma central e então podem ser tratadas. O intuito é a construção de uma base de dados que capacite a tomada de decisões a curto e longo prazo que aprimorarão o trabalho nas estufas, evitando perdas biológicas e financeiras.

## 1.1 ESTUFAS E AS NECESSIDADES DAS PLANTAS

No contexto da agricultura familiar e floricultura, o termo estufa representa o local com cobertura e estrutura adequada para o cultivo de uma ou mais espécies de hortaliças e plantas, retornando lucro para o propietário ao investir tempo e dinheiro para a colheita. As estufas são uma alternativa que foi descoberta para contornar a problemática do clima em determinadas regiões, uma vez que cada vegetal possui particularidades quanto aos níveis de nutrientes que necessitam para a sobrevivência (HANAN, 1998). Essas estruturas utilizam como fonte de energia primária o sol, porém, também dependem do fator clima. Isso se deve ao fato de que cada parte do globo apresenta sua própria variação e condição climática, fazendo com que certos lugares não precisem tanto de aquecimento, enquanto outros é extramente necessário. Em suma, as estufas recriam as condições adequadas para o crescimento das plantas que não são possíveis de forma natural no ambiente que se pretende cultivá-las.

Dessa maneira, a forma de como a cobertura será construida, bem como o tipo de estrutura utilizada, além dos fatores de aquecimento e resfriamento são considerados no momento de projetar a estufa. Muitos desses fatores atuam como contrapontos do ambiente em que se pretende instala-la, como por exemplo, a força e direção dos ventos nas diferentes estações do ano, o nível de irradiação solar, temperatura, nível pluviométrico de chuva e neve, entre outros (HANAN, 1998).

O princípio do funcionamentro das estufas é relativamente simples de ser entendido, embora muitos fatores sejam considerados para sua construção. Primeiramente a luz do sol penetra a estufa por meio do material utilizado para sua cobertura(telas, vidros ou plástico). Uma vez dentro da estufa, as plantas, o chão e outros objetos absorvem esta luz convertendo-a em energia térmica. A quantidade de energia convertida depende do conteúdo interno da estufa, bem como seu tamanho e exposição ao sol. Em seguida, o calor absorvido e convertido é irradiado para o ambiente por meio dos objetos que o absorveram de forma gradual ao longo do tempo, fazendo com que o ar se aqueça e por consequência, mantém a temperatura adequada para a vegetação mesmo após o pôr do sol. Este calor perdura pois uma vez que a luz solar entra na estufa, é absorvida e irradiada, o comprimento de onda da mesma varia e não retorna pelo material que uma vez passara, fazendo com que o calor fique armazenado. Este calor permanece dentro, subindo, enquanto o ar frio é empurrado para baixo criando um clima artificial.

Este clima artificial criado varia de acordo com a necessidade do local, como discutido anteriormente, porém, além deste, ele também depende do tipo de plantação que será cultivada no interior da estufa. Quanto aos fatores relacionados ao cultivo, podem-se destacar quatro fatores principais que determinam o crescimento e sobrevivência da planta: a quantidade de luminosidade disponível, temperatura ambiente, umidade do solo e concentração de gás carbônico no ar (MOURãO, 2007).

Cada espécie de planta possui uma quantidade específica desses quatro elementos que favorecem seu crescimento. Quando esses fatores estão desregulados, seu crescimento é prejudicado e em muitos casos, acaba morrendo (MAUSETH, 2017). Por exemplo, plantas como os cactos são especialistas em sobreviver com escassez de água, mas murcham quando há alta concentração de água em suas raízes. De forma semelhante, plantas como as violetas ou os lírios-da-paz precisam receber iluminação indireta, pois suas folhas podem queimar à exposição solar direta.

Além disso, muitas estufas são usadas para o cultivo de diversos tipos de monoculturas, e muitas possuem multiclimas, o que significa que possuem padrões de temperatura e luminosidade diferentes em determinadas partes do estabelecimento devido a forma de sua construção e ambiente. Dessa forma, monitorar e analaisar essas variáveis são de grande importância para o melhor proveito da estufa, de acordo com a necessidade.

## 1.2 A PLATAFORMA ESP32

O ESP32 é um microcontrolador desenvolvido pela Espressif Systems, uma empresa chinesa especializada em soluções de conectividade sem fio. O ESP32 é conhecido por sua versatilidade e poder de processamento, sendo amplamente utilizado em uma variedade de aplicações, desde projetos de IoT (Internet das Coisas) até sistemas embarcados complexos.

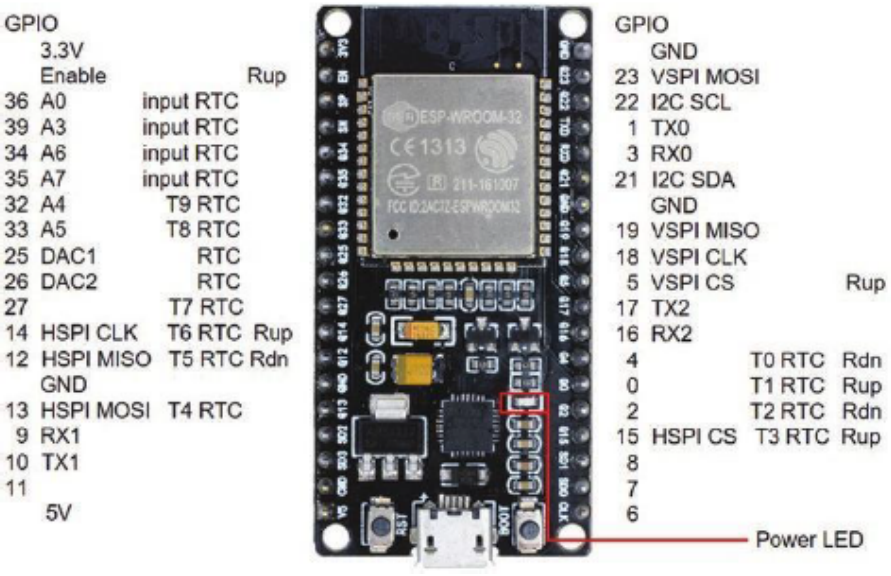
A arquitetura do ESP32 é baseada em um processador Xtensa LX6 de 32 bits, projetado pela Tensilica (agora uma divisão da Cadence Design Systems). Este processador dual-core opera a uma frequência de clock de até 240 MHz, oferecendo um desempenho robusto para uma variedade de tarefas computacionais. Além disso, a placa ESP32 conta com muitos recursos-chave tais como conectividade Wi-Fi e Bluetooth, pinos GPIOs, conversores ADC e DAC, periféricos(UART, SPI, I2C, PWM), segurança e criptografia (CAMERON, 2021).

A programação com ESP32 é geralmente realizada usando a plataforma de desenvolvimento Arduino IDE através das linguagens de programação C/C++ e as ferramentas de desenvolvimento fornecidas pela Espressif. A comunidade de desenvolvedores é vasta e ativa, com uma ampla variedade de bibliotecas e recursos disponíveis para ajudar no desenvolvimento de projetos. Devido à sua versatilidade, o ESP32 é amplamente utilizado em várias aplicações, incluindo sistemas de IoT, automação residencial e monitoramento ambiental.

O ESP32 conta, geralmente, com 25 pinos GPIO que desempenham muitas funções. Neil (CAMERON, 2021) faz um breve resumo das funcionalidades de cada um:

> Os pinos são referenciados pelos números GPIO (General Purpose Input/Output) do microcontrolador ESP32. Existem seis pinos ADC (conversor analógico-digital) com resolução de 12 bits (GPIO 32, GPIO 33, GPIO 34, GPIO 35, GPIO 36 e GPIO 39) e dois pinos DAC (conversor digital-analógico) com resolução de 8 bits (GPIO 25 e GPIO 26). Os pinos de comunicação para I2C são GPIO 21 (SDA) e GPIO 22 (SCL), e para SPI são GPIO 23 (MOSI), GPIO 19 (MISO), GPIO 18 (CLK) e GPIO 5 (CS). Todos os pinos GPIO, exceto os pinos de entrada apenas (GPIO 34, GPIO 35, GPIO 36 e GPIO 39), são pinos PWM (Modulação por Largura de Pulso); e todos os pinos GPIO têm funcionalidade de interrupção. Existem nove pinos capacitivos de toque (GPIO 2, GPIO 4, GPIO 12, GPIO 13, GPIO 14, GPIO 15, GPIO 27, GPIO 32 e GPIO 33). O LED embutido está no GPIO 2, e o LED está ativo em nível ALTO. Vários pinos estão disponíveis para o relógio em tempo real para acionar o microcontrolador ESP32 a partir do modo de suspensão. Resistores pull-up internos estão conectados aos pinos GPIO 0, 5, 14 e 15, com resistores pull-down nos pinos GPIO 2, 4 e 12.

Figura 1 – Pinagem comum de um ESP32.



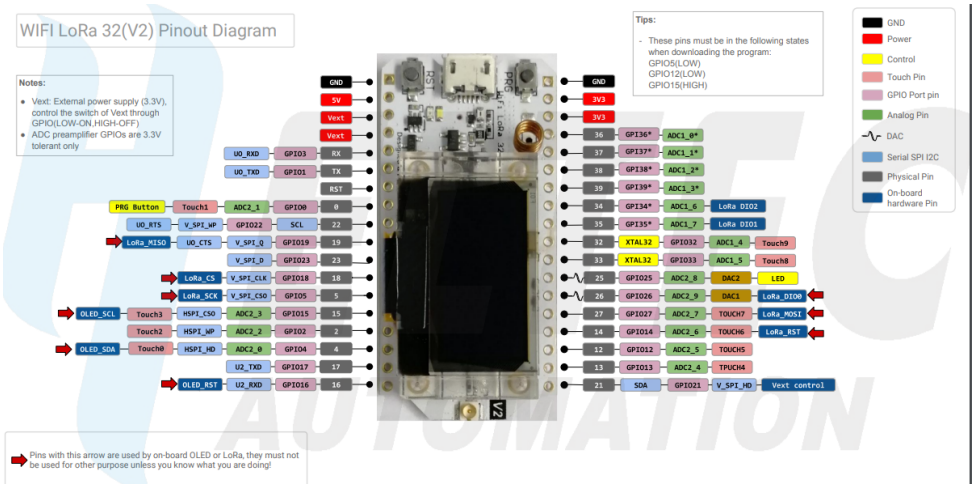

Fonte: (CAMERON, 2021).

### 1.2.1 Heltec WiFi LoRa 32(V2)

O Heltec WiFi LoRa 32 V2 é uma variante do popular módulo ESP32, produzido pela empresa Heltec Automation, que combina conectividade Wi-Fi e LoRa em um único dispositivo. Este módulo é uma escolha versátil para projetos que exigem comunicação sem fio de longo alcance e acesso à internet. Em termos de recursos e funcionalidades, o Heltec WiFi LoRa 32 V2 possui um poderoso processador dual-core ESP32, que oferece um desempenho excepcional para tarefas de processamento de dados e conectividade. Além disso, sua capacidade de comunicação LoRa permite a transmissão de dados a longas distâncias, tornando-o ideal para aplicações como monitoramento ambiental, rastreamento de ativos e agricultura inteligente(OLIVEIRA et al., 2022).

Uma das principais vantagens econômicas deste módulo é a sua capacidade de combinar múltiplas tecnologias de comunicação em um único dispositivo, o que reduz a necessidade de componentes adicionais e simplifica o projeto, economizando tempo e recursos. Além disso, a pinagem do Heltec WiFi LoRa 32 V2 é projetada para ser compatível com a plataforma Arduino, tornando-o acessível para uma ampla comunidade de desenvolvedores e oferecendo uma curva de aprendizado suave para entusiastas.

Figura 2 – Pinagem do Heltec WiFi LoRa 32(V2).




Fonte:(WIFI. . . , 2020).

### 1.2.2 Sensores

Sensores são dispositivos ou instrumentos que têm a capacidade de detectar, medir ou capturar informações específicas do ambiente físico ou de outros sistemas e converter essas informações em sinais elétricos, digitais ou analógicos que podem ser processados, exibidos, registrados ou utilizados para controle. Eles são utilizados em uma ampla variedade de aplicações em diferentes campos, incluindo eletrônica, automação industrial, automobilismo, medicina, meio ambiente, robótica e muito mais.

Os sensores são projetados para detectar uma variedade de propriedades e fenômenos, tais como temperatura, luminosidade, umidade, entre muitos outros. Eles desempenham um papel crítico na coleta de dados em tempo real, automação de processos, controle de sistemas e uma série de outras aplicações em ambientes acadêmicos e industriais. Além disso, cada tipo de sensor é projetado para operar em uma ou várias plataformas, tornando-os bastante versáteis.

### 1.2.3 Sensores de Temperatura

Sensores de temperatura são dispositivos que permitem medir a temperatura e a umidade relativa do ar em um ambiente. Eles são muito utilizados em projetos de automação residencial, sistemas de controle de climatização, estações meteorológicas caseiras e outros aplicativos que requerem monitoramento ambiental básico.

O princípio de funcionamento de um sensor de temperatura baseia-se na propriedade física de certos materiais de alterar sua resistência elétrica em resposta a variações na temperatura. Este fenômeno é conhecido como coeficiente de temperatura da resistência e é explorado em sensores de resistência de platina (RTDs) e termistores. Em um RTD, um fio de platina é usado, cuja resistência aumenta linearmente com o aumento da tempe-

ratura, enquanto em termistores, a resistência diminui exponencialmente com o aumento da temperatura. A mudança na resistência elétrica é medida por um circuito elétrico, e a temperatura é calculada com base em relações matemáticas específicas para o tipo de sensor utilizado(BROOKER, 2009).

Os sensores de temperatura mais utilizados no mercado são os DHT11 e DHT22. Estes sensores possuem três terminais encapsulados por um revestimento de plástico.

Os sensores DHT11 e DHT22 diferem em sua resolução e processamento de dados, sendo o primeiro mais básico em relação ao segundo. O DHT11 possui precisão de detecção de temperatura de +/– 2 graus celsius e uma resolução de umidade de +/– 5% de umidade relativa. Além disso, sua faixa de trabalho de 20 a 90% de umidade relativa e 0 a 50 graus C. Já o DHT22 possui precisão de +/– 0,2 graus celsius resolução de temperatura e capacidade de detecção de +/– 1% umidade relativa, além de um processamento de dados mais rápido. O DHT22 tem uma faixa de detecção mais ampla do que o DHT11: –40 a 80 graus celsius e 0 a 100% de umidade relativa. Ambos possuem pinagem idêntica (HUGHES, 2016).

### 1.2.4 Sensores de Luminosidade

Sensores para detectar luz vêm em uma variedade de estilos e tipos. Alguns, como sensores infravermelhos, podem detectar calor, enquanto outros alguns respondem à temperatura emitida por calor, outros respondem à luz visível.

Um sensor de luz pode empregar um elemento resistivo que altera sua resistência intrínseca em resposta à quantidade de luz que incide sobre ele. Esse tipo de sensor é conhecido com fotocélula.

Uma fotocélula, também conhecida como resistor dependente da luz (LDR), é um componente no qual a resistência muda em função da quantidade de luz que incide sobre ele. Embora esses dispositivos não sejam tão rápidos pelos padrões de fotodiodo ou fototransistor, eles ainda são rápidos o suficiente para transportar áudio em um feixe de luz modulado em amplitude. Eles são úteis como detectores de nível de luz ambiente, links de dados ópticos codificados por pulso de baixa velocidade simples, sensores de beacon para um robô para que ele possa encontrar uma estação de recarga e rastreadores de posição solar para um conjunto de células solares (HUGHES, 2016).

### 1.2.5 Sensores de Umidade do Solo

A maioria dos sensores de umidade do solo consiste em um sensor de dois pinos que funciona como uma sonda de condutividade. A sonda está configurada para atuar como um componente em um divisor de tensão simples, e a tensão que aparece através dela será função da condutividade do solo entre as sondas(BROOKER, 2009).

### 1.2.6 Sensores de Gases

Sensores de gás (também conhecidos como detectores de gás) são dispositivos eletrônicos que detectam e identificam diferentes tipos de gases. Eles são comumente usadospara detectar gases tóxicos ou explosivos e medir a concentração de gás, geralmente em fábricas para identificar vazamentos de gás ou em residências para detectar fumaça e monóxido de carbono. Os sensores de gás variam muito em tamanho (portátil e fixo), alcance e capacidade de detecção. Eles geralmente fazem parte de um sistema incorporado maior, como sistemas de segurança e materiais perigosos, e normalmente são conectados a um alarme ou interface audível. Como os sensores de gás estão constantemente interagindo com o ar e outros gases, eles precisam ser calibrados com mais frequência do que muitos outros tipos de sensores.

Dependendo dos ambientes e funções pretendidos, a composição física e o processo de detecção podem variar notavelmente entre os sensores. Um dos sensores de gás mais comumente usados para identificação de tóxicos e detecção de fumaça é o sensor de gás à base de óxido metálico. Este tipo de sensor emprega um quimiresistor que entra em contato e reage com os gases alvo. Os sensores de gás de óxido metálico aumentam sua resistência elétrica à medida que entram em contato com gases como monóxido de carbono, hidrogênio, metano e butano. A maioria dos sistemas de detecção de fumaça baseados em casa são sensores baseados em óxido(AWANG, 2014).

## 1.3 A TECNOLOGIA LORA

O termo LoRa é uma sigla que denota Long Range, que em inglês significa algo como grande área de atuação. Isso se dá pois a tecnologia LoRa trata-se de um espectro de frequência cuja modulação foi patenteada pela empresa Semtech e baseada na modulação CSS (Chirp Spread Spectrum). A propagação do sinal desta tecnologia se dá em faixas de frequências isentas de taxas e legais, o que facilita a implementação e uso por qualquer interessado no seu estudo, ao mesmo que tempo que é acessível para consumidores de baixo poder aquisitivo. Como o nome indica, a tecnologia LoRa fornece longo alcance e baixo consumo de energia a uma baixa taxa de dados e uma transmissão de dados segura. O LoRa pode ser usado com redes públicas, privadas ou híbridas para alcançar um alcance maior do que as redes celulares. A tecnologia LoRa pode se integrar facilmente às redes existentes e permite aplicativos de Internet das Coisas (IoT) de baixo custo e operados por bateria (FRAZÃO; SILVA, 2021).

Um gateway (porta de entradas) dessa tecnologia pode cobrir, sozinho, a área de cidades inteiras ou até mesmo centenas de quilômetros. O que determina sua cobertura são os obstáculos que o sinal encontra no caminho da propagação, como construções, edifícios, árvores, entre outros. Alguns fatores ambientais e técnicos também podem interferir, como as tempestades e interferências de sinal, respectivamente.

### 1.3.1 O Protocolo LoRaWAN

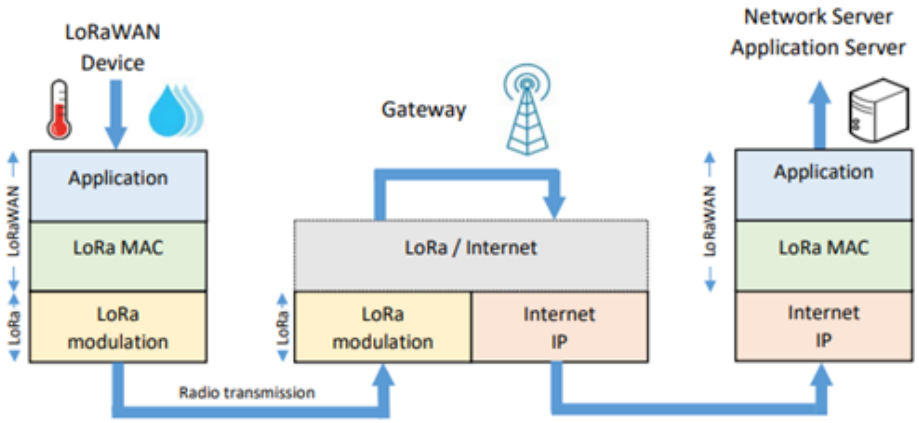
As redes LoRa são consideradas como LPWANs, que significam Low-Power Wide Area Networks. Isso se dá porque os nós dessas redes podem funcionar a base de baterias que podem durar cerca de 10 anos. Esses nós operam com pequenos pacotes de informação que são transmitidos à longas distâncias e com frequência baixa, como por exemplo, a cada vinte minutos (SENEVIRATNE, 2019).

O termo LoRaWAN significa Long Range Wide Area Network e é o nome dado para a o protocolo de comunicação e sistema de arquitetura da rede da tecnologia LoRa. As grandes vantagens desse protocolo podem ser expressas na otimização da bateria dos nós, capacidade de transmissão de dados, qualidade de transmissão, segurança dos dados e as inúmeras aplicações possíveis para essa tecnologia.

A arquitetura do protocolo LoRaWAN consiste em quatro pontos: endnodes, gateways, um servidor de rede e servidores de aplicativos. Em uma rede LoRaWAN, os dados transmitidos por um endnode geralmente são recebidos por vários gateways. Assim que os dados forem recebidos, cada gateway encaminhará o pacote recebido para o servidor de rede por meio de celular, Ethernet, Wi-Fi ou satélite. O software em execução no gateway é responsável por encaminhar qualquer pacote de dados de entrada para o servidor de rede. Este software é conhecido como *Packet Forwarder*(FRAZÀO; MARTINS; SILVA, 2022).

O servidor de rede envia e recebe mensagens LoRaWAN para dispositivos e se comunica com servidores de aplicativos upstream. O servidor de aplicativos é o destino dos dados de aplicativos do dispositivo enviados como carga útil nas mensagens LoRaWAN (SENEVIRATNE, 2019).

Figura 3 – Estrutura da rede LoRaWAN.




Fonte: (MONTAGNY, 2021).

### 1.3.2 EndNodes

Os Endnodes (LoRaWAN Devices) são sistemas eletrônicos embarcados pertencentes a estrutura LoRaWAN e IoT que se caracterizam por ter baixo consumo de energia, tamanho e custo. Eles têm um transceptor de rádio LoRa para se comunicar com os gateways, contudo não endereça especificamente um gateway. É possível que existam mais de um gateway na área de atuação e todos recebem as mensagens e as processam. Existem centenas de fabricantes de dispositivos finais LoRaWAN.

Os EndNodes referem-se a dispositivos equipados com sensores para coletar e monitorar dados. Eles geralmente vêm na forma de microcontroladores de baixa potência que podem ser implantados por anos sem qualquer necessidade de manutenção. Eles podem ser implantados em qualquer lugar, desde sistemas de monitoramento de segurança ou proteção, máquinas de venda automática, até rastreamento de animais de estimação.

Um EndNode pode ser construído combinando várias opções de hardware e software . Alguns EndNodes são fisicamente fixos enquanto outros são móveis e transmitem dados enquanto se movem. Por exemplo, um EndNode LoRa com um módulo GPS fixado em um veículo pode enviar geolocalização periodicamente para o gateway LoRa enquanto se move dentro da cobertura da rede. Um EndNode LoRa pode receber dados através do downlink do gateway LoRa. Normalmente, um EndNode LoRa consiste em um rádio LoRa, microcontrolador, sensor, unidade de fonte de alimentação (PSU) e uma antena (SENEVIRATNE, 2019).

### 1.3.3 Gateways

Os Gateways LoRaWAN são os dispositivos que fazem a comunicação entre os EndNods e a rede. Para receber informações dos EndNodes, os gateways são equipados com um concentrador LoRa e podem, em essência, ser considerados como uma espécie de roteador. Existem dois tipos principais de Gateways LoRaWAN que são diferenciados por seu tipo de software: Os que possuem apenas o firmware, que executa apenas o software de encaminhamento de pacotes para o servidor de rede e os que possuem sistema operacional, em que o software de encaminhamento de pacotes é executado em segundo plano. Embora isso consuma mais energia, os administradores de gateway podem utilizar o dispositivo de gateway para outros fins.

É importante distinguir que os EndNodes LoRaWAN se comunicam com os gateways LoRaWAN através de LoRaWAN de baixa potência, enquanto os gateways se comunicam com o servidor de rede por meio de protocolos de comunicação de largura de banda mais alta, como WiFi, Ethernet ou celular (A..., 2021). Os gateways LoRa devem ter mais poder de processamento do que os EndNodes. Um gateway LoRa de canal único pode ser construído com o Raspberry Pi, ESP32 ou uma plataforma de hardware semelhante.

Esses gateways de canal único não são compatíveis com LoRaWAN e, apesar de simples, eles podem ser usados para começar a explorar as redes de rádio LoRa.

Contudo, esses gateways recebem pacotes LoRa em uma única frequência e, portanto, podem oferecer uma cobertura ruim. Suponha que EndNode envie dados para um gateway LoRa de canal único usando oito frequências diferentes em momentos diferentes. Mas o gateway LoRa de canal único pode receber dados com apenas uma frequência e todas as outras frequências com dados são ignoradas. Um concentrador multicanal consiste em um chip de banda base digital (SX1301) e um ou mais front-ends de RF para transceptores de modo multi-PHY modulador/demodulador digital I e Q (SX1257). O chip de banda base é capaz de receber dados em várias frequências ao mesmo tempo (SENEVIRATNE, 2019).

Um gateway LoRa de canal único consiste em um transceptor de rádio, microprocessador, fonte de alimentação e antena. Para gateways LoRa multicanal, o transceptor de rádio é conhecido como concentrador (MONTAGNY, 2021). O padrão de frequência para a transmissão do sinal na rede LoRaWAN é definido pela LoRa Alliance . Ela define perfis de frequência regionais para operar LoRaWAN para diferentes regiões regulatórias em todo o mundo. A tabela abaixo mostra as frequências utilizadas de acordo com cada região:

Figura 4 – Frequências de operação das redes LoRaWAN por região.

| Country | Band/Channels | Channel Plan |
|---|---|---|
| United States | 902–928 MHz | US902-928, AU915-928 |
| United Kingdom | 433.05–434.79 MHz | EU433 |
| | 863–873 MHz | EU863-870 |
| | 918–921 MHz | Other |
| Canada | 902–928 MHz | US902-928, AU915-928 |
| Australia | 915–928 MHz | AU915-928, AS923 |
| India | 865–867 MHz | IN765-867 |
| France | 433.05–434.79 MHz | EU433 |
| | 863–870 MHz | EU863-870 |
| Sri Lanka | 433.05–434.79 MHz | EU433 |

Fonte: (SENEVIRATNE, 2019).

O Gateway geralmente opera com 8 ou mais canais simultaneamente e encaminha os dados recebidos para um servidor de rede LoRaWAN. O software em execução no Gateway LoRa, responsável pela recepção e transmissão, é o *Packet Forwarder*. Implementações comuns são o Semtech Packet Forwarder e o Semtech Basic Station Packet Forwarder.

Existem muitos gateways prontos disponíveis no mercado, porém, por possuirem boas infraestruturas e serem vendidos para grandes projetos, o preço acaba sendo igualmente alto. Contudo, é possível configurar um gateway caseiro ao instalar o Semtech Packet Forwarder em um micro-computador Raspberry PI acoplado a um módulo RAK2243.

Figura 5 – Raspberry PI 3B+.

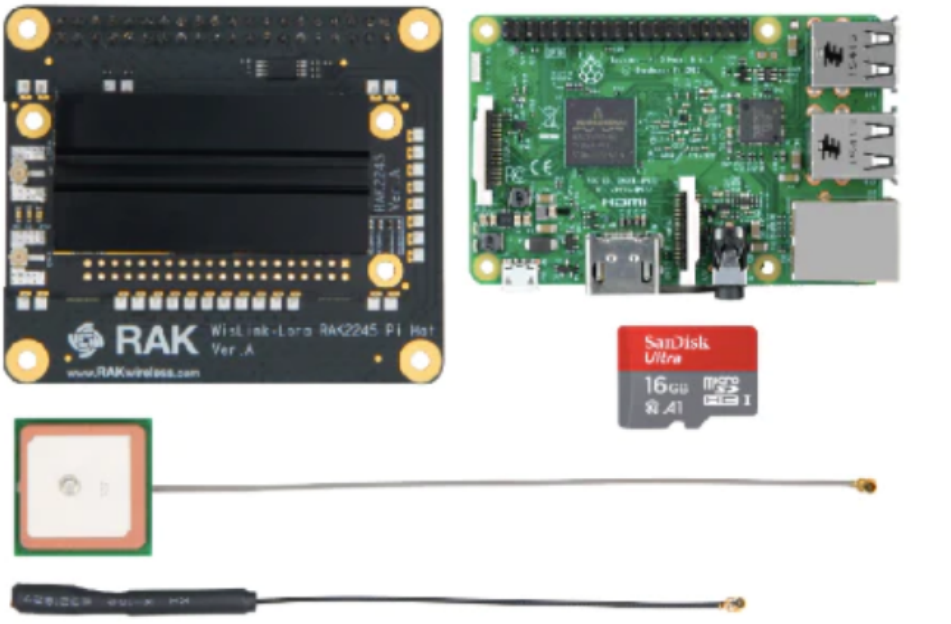
Fonte: (RASPBERRY. . . , 2021).

O Raspberry Pi 3B+ é um computador de placa única (SBC) desenvolvido pela Raspberry Pi Foundation. É conhecido por sua versatilidade e capacidade de executar uma

variedade de aplicativos e sistemas operacionais. Ele oferece recursos como processador quad-core, conectividade Wi-Fi e Bluetooth integradas, tornando-o ideal para projetos de IoT e automação residencial.

Já o RAK2243 é um concentrador LoRaWAN de alto desempenho produzido pela RAK Wireless. Ele é projetado para fornecer conectividade LoRa de longo alcance e alta confiabilidade para dispositivos IoT. O RAK2243 oferece uma interface SPI que permite a comunicação com um SBC, como o Raspberry Pi.

Figura 6 – Módulo RAK2243 para Raspberry PI.




Fonte: (RAK2245. . . , 2020).

Ao combiná-los, o Raspberry Pi 3B+ e o RAK2243 podem ser usados para criar uma solução gateway completa. O Raspberry Pi atua como o cérebro do sistema, executando aplicativos e gerenciando a conectividade Wi-Fi e hospedando softwares paralelos, enquanto o RAK2243 lida com a comunicação LoRaWAN, permitindo que os dispositivos IoT transmitam dados a longas distâncias com baixo consumo de energia.

### 1.3.4 Servidores de Rede e de Aplicação

O Servidores de Rede são os componentes da rede LoRaWAN responsáveis por receberem as mensagens transmitidas pelos gateways e remover os pacotes duplicados (vários gateways podem receber a mesma mensagem e transmiti-los ao mesmo servidor). Assim que eles recebem, eles autenticam a mensagem graças a uma chave AES de 128 bits chamada NwkSKey (Network Session Key).

Se o processo de autenticação falhar, o servidor descarta a mensagem LoRaWAN. Se o processo de autenticação for bem-sucedido, o servidor transfere a mensagem para o

Servidor de Aplicação.

A chave de sessão de rede (NwkSKey) é usada para autenticação entre o EndNode LoRaWAN e o servidor de rede. Para realizar esta autenticação, um campo MIC (Message Integrity Control) é adicionado ao quadro. O MIC depende dos dados transmitidos criptografados e da NwkSKey. Durante a recepção, o mesmo cálculo é realizado. Se NwkSKey for o mesmo no EndNode e no servidor, então o MIC transmitido deve ser o mesmo que o gerado durante a recepção (MONTAGNY, 2021).

O Servidor de Aplicação recebe mensagens criptografadas de servidor de rede. A criptografia e descriptografia são feitas graças a uma chave AES de 128 bits chamada AppSKey (Application Session Key). A chave de sessão do aplicativo (AppSKey) é usada para criptografar os dados do usuário EndNode LoRaWAN. Esta é uma criptografia simétrica, portanto, AppSKey no EndNode deve ser igual ao armazenado no servidor de aplicação. Não há como o gateway e o servidor de rede entenderem o valor real dos dados do usuário, o canal é confidencial (MONTAGNY, 2021).

Por fim, esses dados são transmitidos no que se chama de Servidor de IoT, que nada mais é que o lugar onde o usuário acessa essas informações de forma que façam sentido . Nesta etapa ocorrem três serviços principais:

1. Conexão com o servidor de aplicação para coleta de dados. Na maioria das vezes, isso é feito com o protocolo HTTP ou MQTT. Durante o desenvolvimento, às vezes se usa as versões não seguras desses protocolos. Contudo, usaria-se as versões seguras de HTTPS e MQTTS durante a entrega do sistema;
2. Um Banco de dados onde são armazenados dados;
3. Um Dashboard acessível para o usuário por meio de uma página da Web ou um aplicativo móvel.

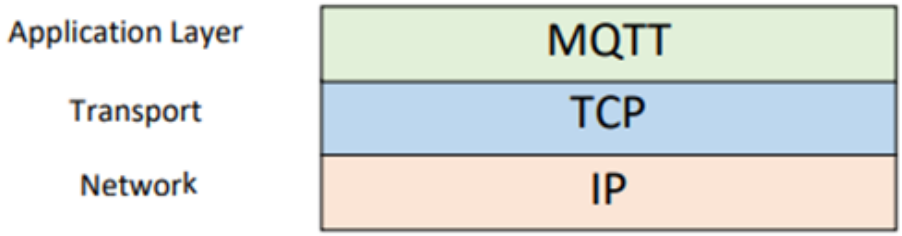
### 1.3.5 Protocolo de comunicação MQTT

O MQTT é um protocolo simples para a Internet das Coisas. Em vez da arquitetura cliente-servidor clássica que trabalha com solicitações e respostas, o MQTT é baseado em um modelo de editor-assinante. Essa diferença é fundamental pois evita ter que solicitar dados que não possuem previsão de chegada. Os dados serão transmitidos diretamente ao assinante assim que forem recebidos no broker (servidor central). Para receber os dados que pertencem a um topic, um assinante deve primeiro se inscrever a esse topic. Um broker MQTT é uma entidade de software central na arquitetura MQTT.

Ele age como um corretor de imóveis que primeiro verifica os antecedentes das partes envolvidas e, depois de garantir que as regras relevantes sejam aplicadas, ele inicia uma transação (MONTAGNY, 2021).

MQTT é um protocolo baseado em TCP, representado pelo diagrama abaixo:

Figura 7 – Protocolos usados para comunicação via MQTT.




Fonte: (MONTAGNY, 2021).

O MQTT é popular em IoT e IIoT (Internet das Coisas Industrial) porque pode ser usado em várias plataformas enquanto consome largura de banda mínima. Os dispositivos IoT geralmente têm recursos limitados e como as implementações do MQTT são mais leves e eficientes do que outras arquiteturas de comunicação (por exemplo, uma API RESTful baseada em HTTP). O MQTT geralmente é uma escolha inteligente para IoT.

## 1.4 SOLUÇÕES VIÁVEIS

Quando se trata de servidores de rede para o protocolo LoRaWAN, é necessário ter em mente o objetivo final de aplicação, que impacta diretamente na escolha dos servidores de aplicação e interface, que tratarão de processar e apresentar os dados de forma legível. Além disto, apesar de ser possível criar uma rede privada LoRaWAN, tal projeto requere muita infraestrutura e recursos para ser implementado. Felizmente, por se tratar de um protocolo aberto e *open source* administrado pela LoRa Alliance, empresa sem fins lucrativos, diversas empresas e comunidades filiadas desenvolveram plataformas que contam com os recursos necessários para implementar sistemas LoRa, como a Chirpstack e TTN (The Things Network).

Além disso, existem muitas possibilidades de bancos de dados como PostgreSQL, MongoDB e InfluxDB. Cada uma dessas opções possui características próprias de integração e armazenamento de dados, que podem ser associados a interfaces gráficas que organizam os dados obtidos de forma legível, como Power BI, Tableau e Grafana.

Para este trabalho, a escolha para a plataforma LoRa, banco de dados, e interface gráfica foram a Chirpstack, InfluxDB e Grafana respectivamente. Essa escolha se deu pela facilidade de conversação entre esses softwares, além da alta acessibilidade deles em plataformas de baixo custo, como a Raspberry PI.

Tendo em vista a natureza dos dados obtidos e o propósito de suas coletas, o banco de dados InfluxDB se mostrou o mais adequado pois seu armazenamento já é dimensionado considerando o tempo exato de chegada de cada informação.

Como a Chirpstack e o InfluxDB, por padrão, não estão diretamente integrados para armazenar os dados, uma ferramenta de fluxo de trabalho desempenha um papel fundamental para o direcionamento desse fluxo. Para tal, foi utilizado o Node-RED.

### 1.4.1 Chirpstack

De acordo com a comunidade oficial da Chirpstack (CHIRPSTACK, 2020), a plataforma é definida como:

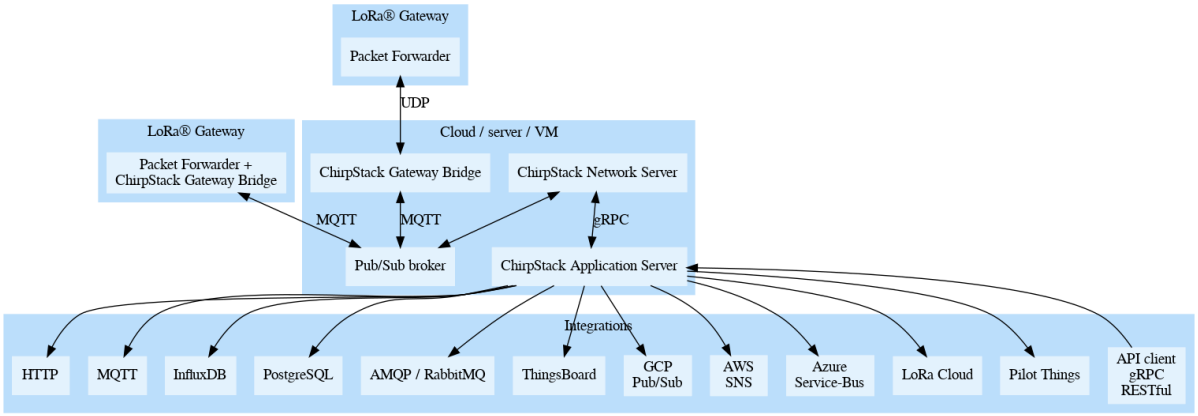
> ChirpStack é um Servidor de Rede LoRaWAN de código aberto que pode ser usado para configurar redes LoRaWAN. O ChirpStack oferece uma interface web para a gestão de gateways, dispositivos e inquilinos, bem como para configurar integrações de dados com os principais provedores de nuvem, bancos de dados e serviços comumente utilizados para lidar com dados de dispositivos. O ChirpStack fornece uma API baseada em gRPC que pode ser usada para integrar ou estender o ChirpStack. Além disso, fornece componentes de código aberto para redes LoRaWAN. Juntos, eles formam uma solução pronta para uso, incluindo uma interface web amigável para gerenciamento de dispositivos e APIs para integração. A arquitetura modular possibilita a integração em infraestruturas existentes. Todos os componentes são licenciados sob a licença MIT e podem ser utilizados para fins comerciais. Os seguintes componentes são disponibilizados: ChirpStack Gateway Bridge, ChirpStack Network Server e ChirpStack Application Server.

Figura 8 – Chirpstack Logo.


Fonte: (CHIRPSTACK, 2020).

Uma implantação típica do ChirpStack possui a seguinte arquitetura:

Figura 9 – Arquitetura da Chirpstack.




Fonte: (CHIRPSTACK. . . , 2020).

A ChirpStack Gateway Bridge fica entre o *Packet Forwarder* e o broker MQTT. Ela converte o formato do *Packet Forwarder* (como o protocolo Semtech Packet Forwarder) em um formato de dados usado pelos componentes ChirpStack. Além disso, ela oferece integrações com várias plataformas de nuvem, como o GCP Cloud IoT Core e o Azure IoT Hub.

O ChirpStack Network Server é um Servidor de Rede LoRaWAN, responsável por gerenciar o estado da rede. Ele possui conhecimento das ativações de dispositivos na rede e é capaz de lidar com solicitações de associação quando os dispositivos desejam ingressar na rede.

Quando os dados são recebidos por vários gateways, o ChirpStack Network Server elimina duplicações nesses dados e os encaminha como uma única carga útil para o Servidor de Aplicativos ChirpStack. Quando um servidor de aplicativos precisa enviar dados de volta a um dispositivo, o ChirpStack Network Server mantém esses itens em fila até que possa enviá-los para um dos gateways.

O ChirpStack Application Server é um Servidor de Aplicativos LoRaWAN, compatível com o ChirpStack Network Server. Ele fornece uma interface web e APIs para o gerenciamento de usuários, organizações, aplicativos, gateways e dispositivos. Nesta etapa, os dados de uplink recebidos podem ser encaminhados para uma ou várias integrações configuradas.

### 1.4.2 Node-RED

O Node-RED é uma plataforma de desenvolvimento de código aberto que permite a criação de fluxos de trabalho automatizados de maneira visual e interativa. Ele foi inicialmente desenvolvido pela IBM Emerging Technology Services e posteriormente doado à Fundação JS Foundation, ganhando popularidade na comunidade de desenvolvedores como uma ferramenta versátil para conectar dispositivos, serviços e APIs de forma eficiente(NODE-RED. . . , 2017).

Figura 10 – Node-RED Logo.


Fonte: (NODE-RED. . . , 2017).

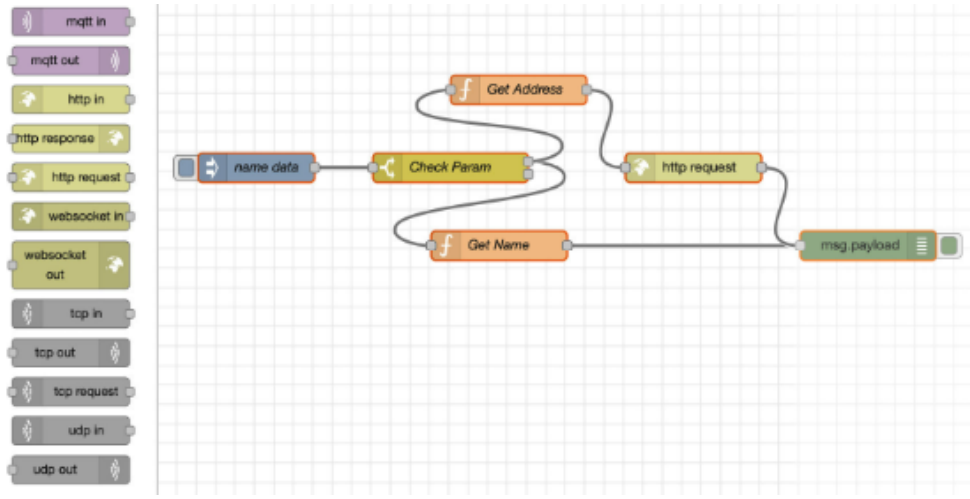
A principal característica distintiva do Node-RED é a sua abordagem baseada em programação visual. Em vez de escrever código tradicional, os desenvolvedores criam fluxos arrastando e conectando nodes (nós) em uma interface gráfica. Cada node representa uma ação ou funcionalidade específica e pode ser configurado através de uma interface intuitiva.

Taiji (HAGINO, 2021) descreve a importância desta ferramenta:

> O Node-RED é uma ferramenta de programação baseada em fluxo (FBP - Flow-based programming), adequada para criar aplicações de controle de dados para aplicações web e IoT. Seu ambiente de desenvolvimento e ambiente de execução são aplicativos baseados em navegador feitos com Node.js, o que torna o desenvolvimento o mais simples possível. É necessário usar o editor de fluxo para criar aplicações Node-RED para IoT, serviços web e muito mais.

Os nodes são os blocos de construção fundamentais dos fluxos no Node-RED. Cada node executa uma função específica, como leitura de dados de sensores, transformação de dados, chamada de APIs externas ou execução de ações. O Node-RED é fornecido com uma ampla variedade de nodes integrados, e os desenvolvedores também podem criar seus próprios nodes personalizados para atender às suas necessidades.

Figura 11 – Exemplo de um fluxo de trabalho em Node-RED.




Fonte:(HAGINO, 2021).

Os fluxos (flows) são a representação visual dos processos automatizados no Node-RED. Eles consistem em nodes conectados por linhas que indicam a sequência das ações. Os desenvolvedores podem criar fluxos complexos conectando nodes de diferentes tipos, permitindo a transformação e encaminhamento de dados de maneira eficiente(HAGINO, 2021).

O Node-RED possui uma biblioteca extensível de nodes pré-construídos que abrangem uma ampla gama de funcionalidades. Isso inclui nodes para integração de bancos de dados, serviços de nuvem, dispositivos IoT, redes sociais e muito mais. A biblioteca ajuda a economizar tempo, reutilizando nodes já desenvolvidos em seus fluxos.

### 1.4.3 InfluxDB

O InfluxDB, um banco de dados de séries temporais de código aberto, é projetado para armazenar, consultar e visualizar dados que variam ao longo do tempo, com otimização notável para cenários onde a linha do tempo é um componente central da análise de dados. Desenvolvido pelo InfluxData, o sistema é amplamente utilizado em diversas aplicações que lidam com dados temporais, incluindo monitoramento de infraestrutura, Internet das Coisas (IoT) e análise financeira, especialmente em contextos onde a variação temporal desempenha um papel fundamental.

Figura 12 – InfluxDB Logo.


Fonte: (INFLUXDB, 2018).

Fundamentado em um modelo de dados baseado em séries temporais, o InfluxDB organiza os dados em pontos de dados associados a timestamps. Cada ponto de dados possui medições, campos, tags e um timestamp. As medições capturam o que está sendo registrado, enquanto os campos armazenam os valores numéricos dessas medições e as tags fornecem metadados para categorização e filtragem dos dados. Essa estrutura permite a representação coerente de informações evolutivas(KEY. . . , 2019).

Um exemplo dessa estrutura de banco de dados pode ser observado abaixo:

Tabela 1 – Dados de um censo.

| **time** | **location** | **scientist** | **butterflies** | **honeybees** |
|---|---|---|---|---|
| 2015-08-18T00:00:00Z | 1 | langstroth | 12 | 23 |
| 2015-08-18T00:00:00Z | 1 | perpetua | 1 | 30 |
| 2015-08-18T00:06:00Z | 1 | langstroth | 11 | 28 |

Fonte: (KEY. . . , 2019).

Além disso, o InfluxDB oferece recursos de gerenciamento de dados avançados, como políticas de retenção de dados. Essas políticas permitem definir o tempo pelo qual os dados serão mantidos no banco de dados antes de serem automaticamente expirados. Isso é crucial para garantir um uso eficiente do espaço em disco, especialmente em ambientes com volumes consideráveis de dados. O banco de dados adota uma abordagem de armazenamento em tabelas particionadas chamadas ”shards”, agrupando pontos de dados em intervalos temporais específicos para facilitar recuperações ágeis.

Para otimizar a performance das consultas, o InfluxDB emprega técnicas de compactação e indexação exclusivamente adaptadas para dados de séries temporais. Esses dados são compactados em blocos de tamanho fixo para reduzir a ocupação de disco, enquanto os índices são otimizados para agilizar consultas baseadas em tempo e tags. No contexto de análise temporal, o InfluxDB oferece a linguagem de consulta InfluxQL, projetada para simplificar a análise de séries temporais, incluindo operações de agregação, filtragem e agrupamento, para facilitar a exploração detalhada dos dados ao longo do tempo.

### 1.4.4 Grafana

O Grafana, uma plataforma de visualização e análise de dados de código aberto, desempenha um papel essencial na capacidade das organizações de compreender seus sistemas e dados. Desde sua introdução em 2013, essa ferramenta se tornou um recurso inestimável ao permitir que os usuários criem painéis altamente personalizados para monitoramento e análise de dados em tempo real. Sua interface intuitiva e flexibilidade tornam-no um aliado valioso para equipes de operações e desenvolvedores, permitindo uma análise mais informada para a tomada de decisões críticas(GRAFANA. . . , 2022).

Figura 13 – Grafana Logo.

Fonte: (GRAFANA. . . , 2022).

Um dos pontos mais fortes do Grafana reside em sua capacidade de se conectar a uma ampla variedade de fontes de dados. Essa flexibilidade é um diferencial notável, visto que o Grafana suporta conexões nativas a várias fontes, como Prometheus, InfluxDB, MySQL, AWS CloudWatch e muitas outras. Essa riqueza de opções permite aos usuários consolidar dados provenientes de diferentes sistemas e serviços, proporcionando uma visão unificada e abrangente do ambiente operacional.

Figura 14 – Exemplos de gráficos gerados pela Grafana.


Fonte: (APP. . . , 2020).

Ao criar painéis personalizados, o Grafana oferece a liberdade de escolher entre diferentes linguagens de consulta, como PromQL e SQL, e criar expressões para transformação e agregação de dados. Além disso, o Grafana capacita os usuários a configurar alertas detalhados e notificações com base em métricas específicas, garantindo uma abordagem proativa para identificação e resolução de problemas. A combinação de sua extensibilidade, capacidade de integração com diversas fontes de dados e recursos avançados de visualização faz da Grafana uma ferramenta requisitada para a análise eficaz de informações em diversos ambientes, incluindo a área de IoT(GRAFANA. . . , 2022).

# 2 METODOLODIA DO DESENVOLVIMENTO DO SISTEMA

Neste ensaio foi realizada uma Pesquisa Aplicada, que tem como finalidade a realização de uma pesquisa descritiva e de campo a respeito das variáveis que compõe o desenvolvimento de culturas em estufas, buscando soluções práticas por meio da coleta de dados em ambiente real escolhido pelo pesquisador. Os procedimentos técnicos que serão empregados são os de pesquisa experimental e de campo. Como método de abordagem, será utilizado o quantitativo e a elaboração adotará o método monográfico. A coleta de dados será feita por sensoriamento de objetos de estudo e os dados serão interpretados de forma prescritiva e preditiva para obtenção de melhorias.

O processo pelo qual a solução para a problemática será desenolvida se dará em cinco etapas.

Primeiramente, foram escolhidos os dispositivos necessários para a implementação do equipamento sensorial, que, a partir deles, o protótipo EndNode que realiza as medições das variáveis discutidas no referencial teórico foi confeccionado. Dessa forma, foi feito um circuito elétrico conectando os sensores à plataforma ESP32 ao passo que se desenvolveu o código fonte em linguagem C responsável por comandar e acionar o protótipo.

Já na segunda etapa, foi projetado o módulo de gateway que realiza a conexão do aparelho EndNode com a rede por meio do protocolo LoRaWAN. Foi utilizado um micro-computador Raspberry PI com adaptador LoRa para o projeto deste gateway.

A terceira etapa consistiu em estabelecer a conexão entre o módulo EndNode, o gateway e o servidor por meio da tecnologia LoRaWAN. Nesta etapa foi considerada a plataforma mais adequada para hospedar o servidor, bem como o desenvolvimento do servidor LoRA com a definição dos parâmetros a serem utilizados para a recepção dos dados enviados pelo gateway.

Na quarta etapa, foi preparada o sistema de integração que possibilitou o encaminhamento, tratamento e visualização dos dados obtidos pelo equipamento sensorial. Para isso, foi definida a conexão do procotolo MQTT do servidor de rede com o banco de dados escolhido através de uma ferramenta de fluxo de trabalho. Após a configuração do banco de dados, foi realizada a integração deste com a plataforma de visualização gráfica, bem como a criação da dashboard responsavél por mostrar os dados ao usuário final de forma intuitiva.

Finalmente, na quinta etapa foram realizadas medições em ambiente real, coletando e observando os dados em amostras de mudas para testagem da confiabilidade do sistema.

## 2.1 Dispositivos e Softwares utilizados

Dessa forma, os equipamentos, dispositivos e softwares escolhidos e empregados no projeto foram os seguites:

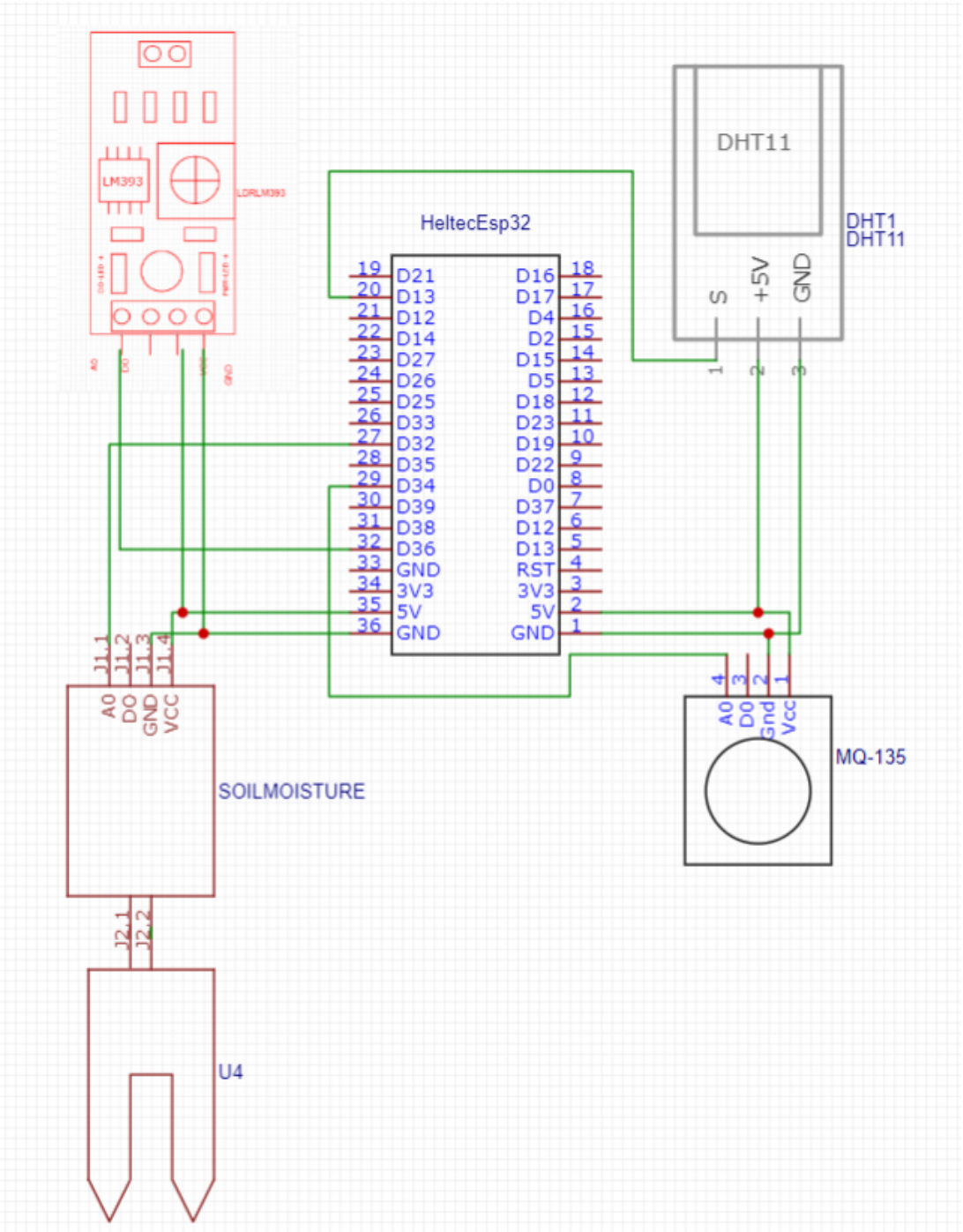
- Placa de Desenvolvimento Heltec ESP32 V.2;
- Sensor de Temperatura e Umidade DHT11;
- Sensor de Luminosidade LDR com LM393;
- Sensor de gases MQ-135;
- Sensor de Umidade do Solo genérico;
- Antenas operando na faixa de 915MHz;
- Micro-computador Raspberry PI 3B+;
- Software de emulação de terminal PuTTY;
- Módulo LoRaWAN RAK2245 para gateway;
- Cartão Micro-SD de 32gb de memória;
- Fonte 5V/2A;
- Cabos USB e Micro USB;
- Jumpers fêmea/fêmea;
- Placa perfurada;
- Software Arduino IDE;
- Software Lora-Gateway Semtech Packet Forwarder;
- Plataforma Lora Chirpstack;
- Software FBP Node-RED;
- Software de Banco de Dados InfluxDB;
- Software de Interface Gráfica Grafana;

## 2.2 O Dispositivo EndNode

A parte inicial do sistema consistiu, primeiramente, preparar um dispositvo que fosse capaz de obter as variáveis requisitadas de forma confiável e continua. Para tal, foi selecionado o dispositivo Heltec ESP32 V.2. Como mencionado anteriormente, o Heltec ESP32 V2 é uma placa de desenvolvimento IoT com o chip ESP32, oferecendo conectividade Wi-Fi e Bluetooth, um display OLED integrado, pinos GPIO para conexão de sensores e compatível com a Arduino IDE. É uma opção versátil e conveniente para projetos envolvendo LoRa.

A partir dos sensores DHT11, LDR LM393, MQ-135 e de Umidade do Solo, foi elaborado o circuito de medição abaixo:

Figura 15 – Esquema de conexões do EndNode.




Fonte: Própria.

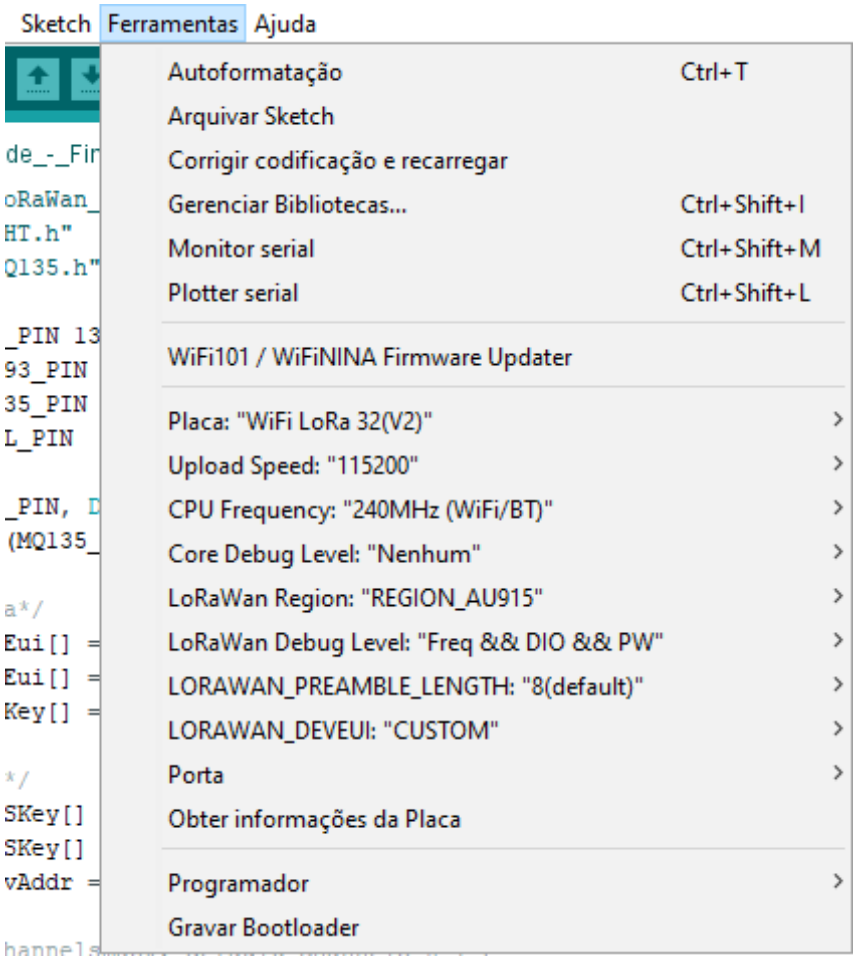
De posse do esquema de conexões, foi feita a soldagem das componentes numa placa perfurada, que posteriormente foi conectada a placa ESP32 por meio de jumpers fêmea/fêmea.

Como o hardware devidamente conectado, fez-se o código responsável por operar o EndNode. A princípio, foi feito um código de teste para fazer a testagem dos sensores e tratamento dos dados, para aferir se os resultados obtidos eram coerentes. Uma vez constatado seu funcionamento, este foi adicionado ao código principal, responsável pela transmissão dos dados para o gateway.

Devido o grande avanço e utilização das plataformas ESP32, o ambiente de desenvolvimento integrado Arduino IDE já conta com um código de envio de pacotes padrão personalizado para a placa Heltec. Ele define parâmetros importantes, como as chaves de segurança devEui e appEui, e configurações específicas para a rede LoRaWAN. Esses parâmetros são cruciais para autenticar o dispositivo e estabelecer a comunicação com a rede. Este código pode ser visualizado no apêndice B, referente aos *scripts* utilizados no projeto, ao final deste trabalho.

Além disso, é importante configurar as condições de operação da placa utilizada no software da Arduino IDE com as especificações abaixo antes de compilar e enviar o código:

Figura 16 – Definição dos parâmetros da placa na plataforma Arduino IDE.




Fonte: Própria.

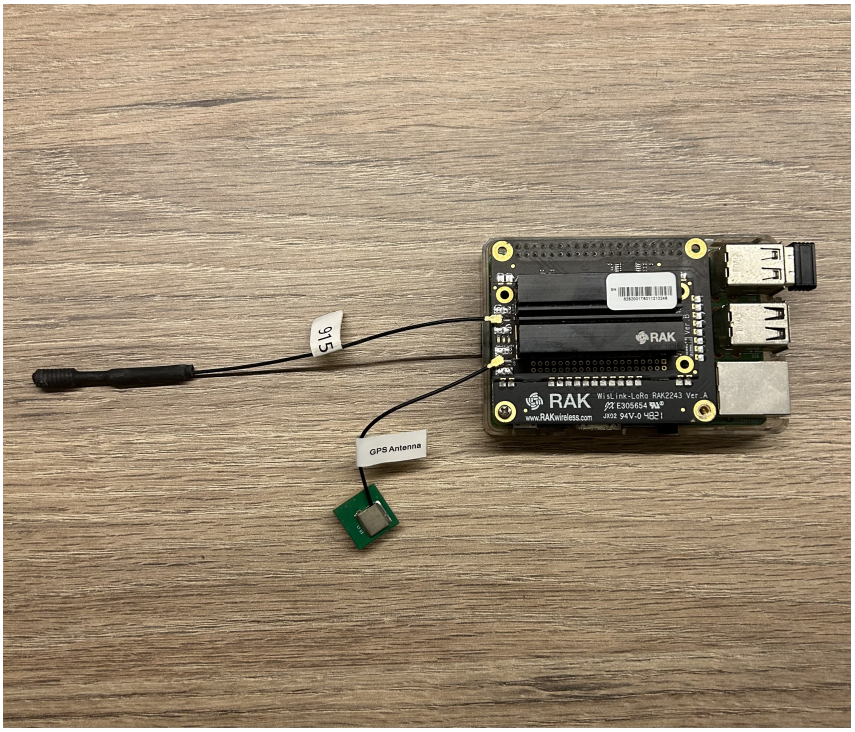
Dessa forma, os sensores conectados ao dispositivo capturam os dados brutos, que são então processados e convertidos para grandezas pertinentes a cada tipo de medição, como temperatura em graus Celsius e umidade em porcentagem. Essas conversões são essenciais para tornar os dados compreensíveis e úteis. Os valores tratados são armazenados em um payload de dados que representa as informações coletadas.

O payload, contendo os dados ambientais tratados, é transmitido periodicamente através da rede LoRaWAN. Isso permite que os dados sejam enviados para um servidor centralizado, onde podem ser acessados e analisados remotamente. Essa abordagem facilita o monitoramento ambiental eficaz e a tomada de decisões baseadas em dados em tempo real.

## 2.3 Gateway Semtech Packet Forwarder

Para a escolha do gateway, foi utilizado a junção do micro-computador Raspberry PI 3B+ com o módulo RAK2245. O RAK2243 é um concentrador LoRaWAN de alta qualidade que se integra perfeitamente com o Raspberry PI. Ele desempenha um papel crucial na recepção e transmissão eficientes de dados LoRa, garantindo uma comunicação confiável entre os dispositivos finais e a infraestrutura da rede. Essa combinação oferece desempenho e facilidade de desenvolvimento, tornando mais acessível a criação de aplicações IoT LoRa.

Figura 17 – Raspberry PI 3B+ e Módulo RAK2243 montados.



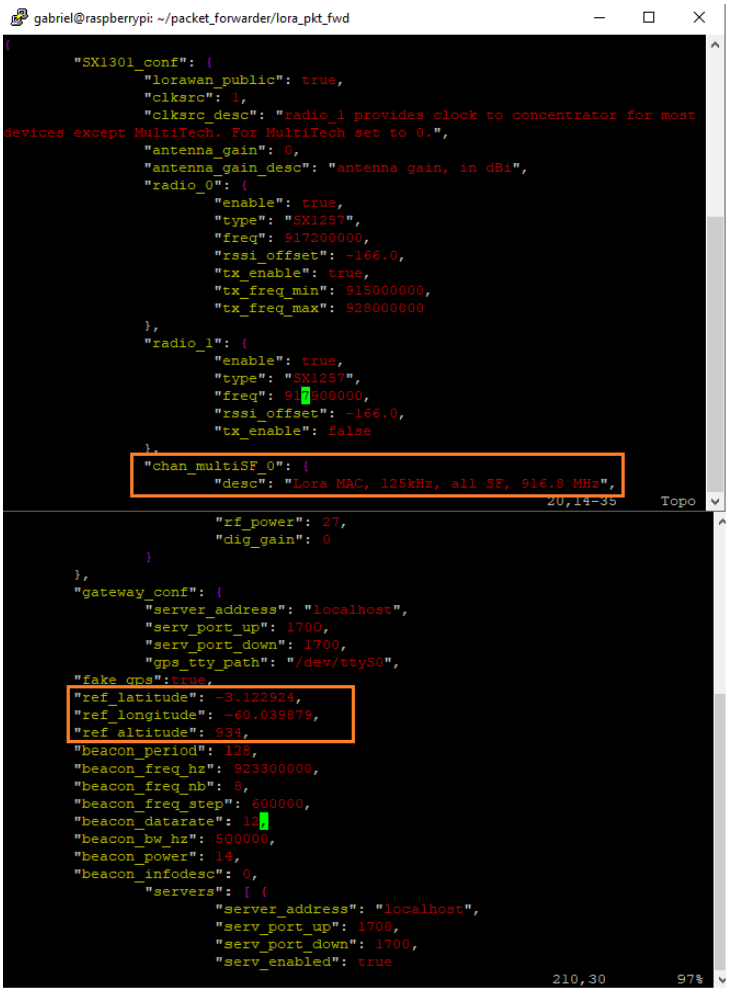

Fonte: Própria.

Além disso, essa combinação é totalmente compatível com o Semtech Packet Forwarder. Ele é um software de código aberto que atua como uma peça intermediária entre dispositivos finais LoRa e os gateways LoRaWAN, e um componente fundamental em redes LoRaWAN. Sua principal função é receber, encapsular e encaminhar os pacotes de dados entre os dispositivos finais e o servidor, permitindo a comunicação eficiente entre todos os dispositivos da rede LoRaWAN.

A instalação da pilha de software Semtech Packet Forwarder pode ser conferida no apêndice A, que tratará sobre o processo de instalação dos softwares hospedados no gateway.

Para que haja a devida comunicação entre os dispositivos, é preciso fazer configurações personalizadas em alguns arquivos-chave durante a instalação deste software. Esses arquivos referem-se às configurações globais e locais do gateway em questão.

Figura 18 – Configurações globais do Gateway.

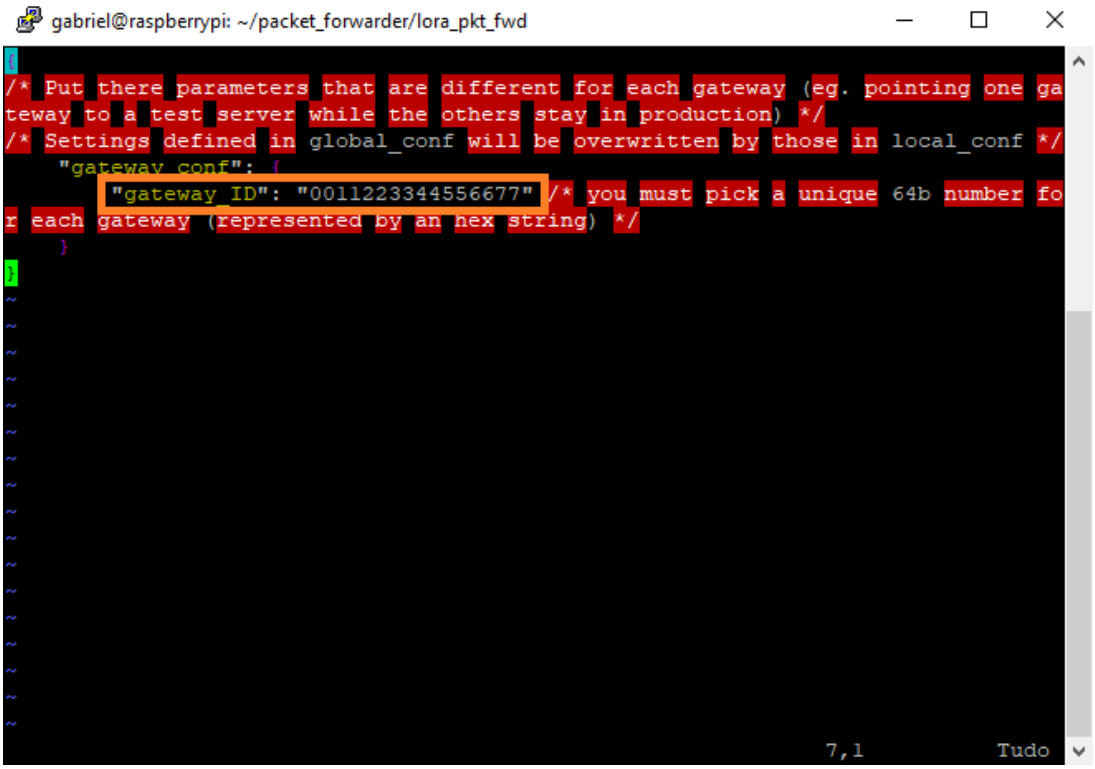



Fonte: Própria.

As configurações globais referem-se a parâmetros como a sub-banda que será utilizada, considerando a faixa de frequência AU915, que é faixa designada na região brasileira. Neste projeto em questão, foi utilizada a sub-banda 2, que engloba os canais 8-15, com frequências de 916.8 a 918.2 Mhz, separadas para região. Além disso, é nele que é configurada as coordenadas de localização geográfica do gateway.

Já na configuração local, é definido o *Gateway ID*, que é necessário para fazer a autenticação do mesmo no servidor de rede.

Figura 19 – Configurações Locais do Gateway.




Fonte: Própria.

## 2.4 Implementação da plataforma Chirpstack

Após a configuração correta do gateway, partiu-se para a instalação da plataforma Chirpstack, onde é concentrado o recebimento dos pacotes encaminhados pelo gateway. O processo de instalação, bem como a pilha de software são fornecidos pela própria comunidade oficial da Chirpstack e são disponibilizados de forma geral mas intuitiva, de forma que, para atender às especificações de cada região e projeto, basta efetuar poucas alterações. O processo de instalação é descrito no apêndice A ao final deste trabalho.

Segundo a documentação presente na comunidade oficial, o software da Chirpstack pode ser hospedado em um dispositivo dedicado ou pode ser instalado no próprio dispositivo usado para o gateway.

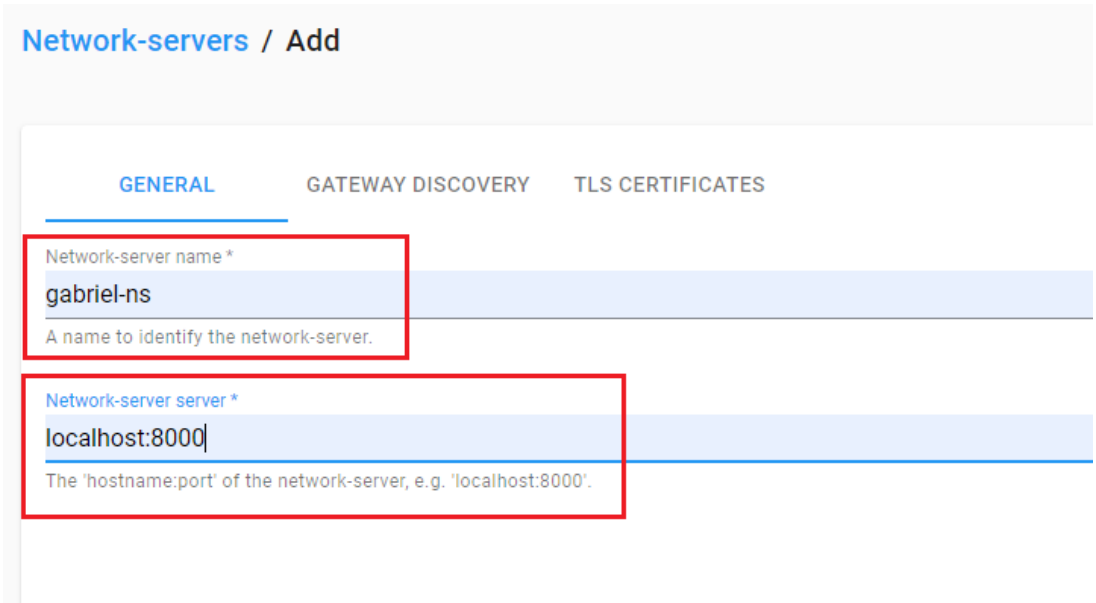
Tendo em vista a simplicidade e economia da solução proposta, optou-se por fazer a instalação no próprio Raspberry PI utilizado, visto que este conta com banstante memória disponível devido ao cartão SD de 32gb de memória, além de contar com a capacidade de processamento suficiente.

É de extrema importância ressaltar e atentar para alguns arquivos de configurações nesta etapa. É necessário conferir as variáveis de definação da região de operação e de sub-bandas utilizadas no arquivo de configuração do servidor de rede, de forma que sejam as mesmas configuradas no gateway, ao passo que não haverá a comunicação entre ambos caso contrário. Além disso, é preciso garantir que as senhas de autenticação escolhidas na instalação da pilha de software sejam adicionadas aos arquivos de configuração do servidor de rede e de aplicação.

Após a instalação e configuração inicial, os serviços do Chirpstack foram verificados para garantir que estavam funcionando corretamente. Isso envolveu a verificação do status dos componentes principais do Chirpstack após a reinicialização.

A interface de configuração do Chirpstack foi acessada por meio de um navegador da web através do endereço padrão *localhost:8080*, permitindo a personalização das configurações de rede. O Network Server foi configurado para atender às necessidades específicas do projeto, incluindo a definição de perfis de gateway e serviços. Para tal, foram configurados o Servidor de Rede, Perfil para Gateway, Perfil de Serviço, Perfil dos Dispositivos e por fim, a seção de Aplicação, onde o dispostivo EndNode é adicionado.

Figura 20 – Configuração do Servidor de Rede.



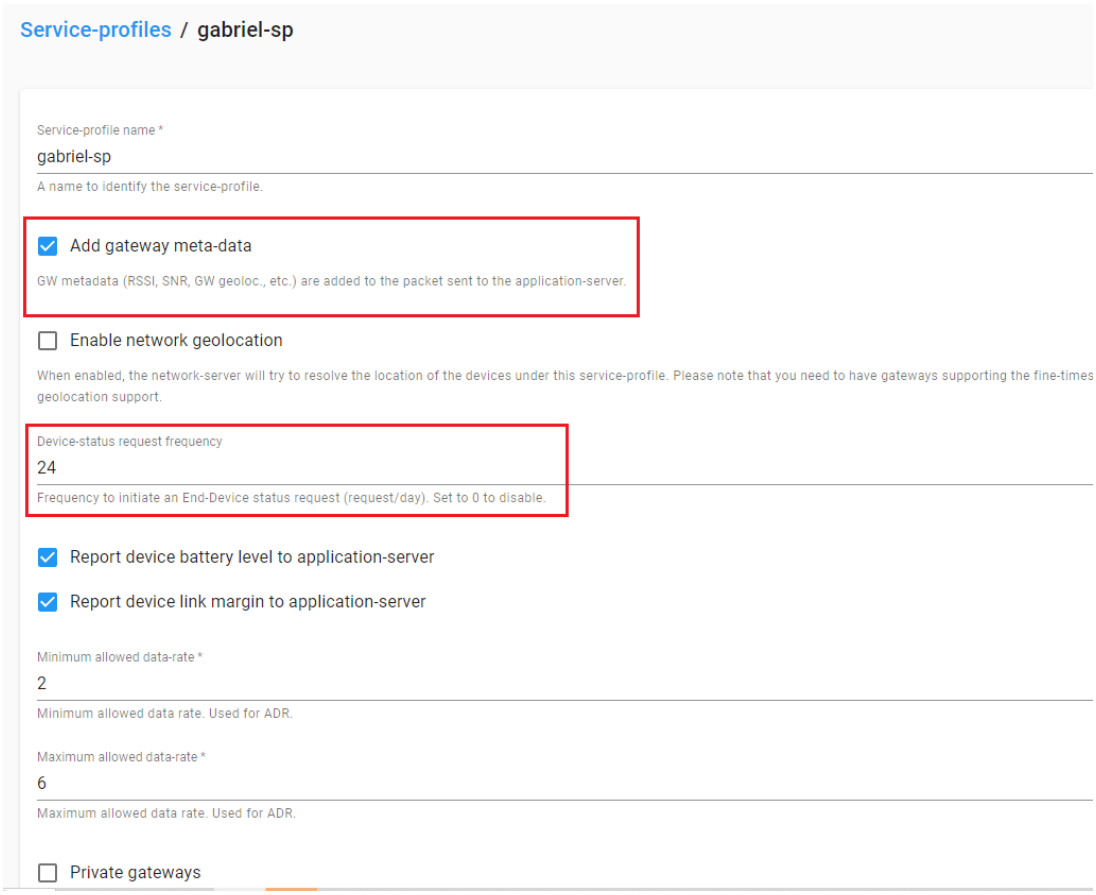

Fonte: Própria.

Figura 21 – Configuração do Perfil para Gateway.


Fonte: Própria.

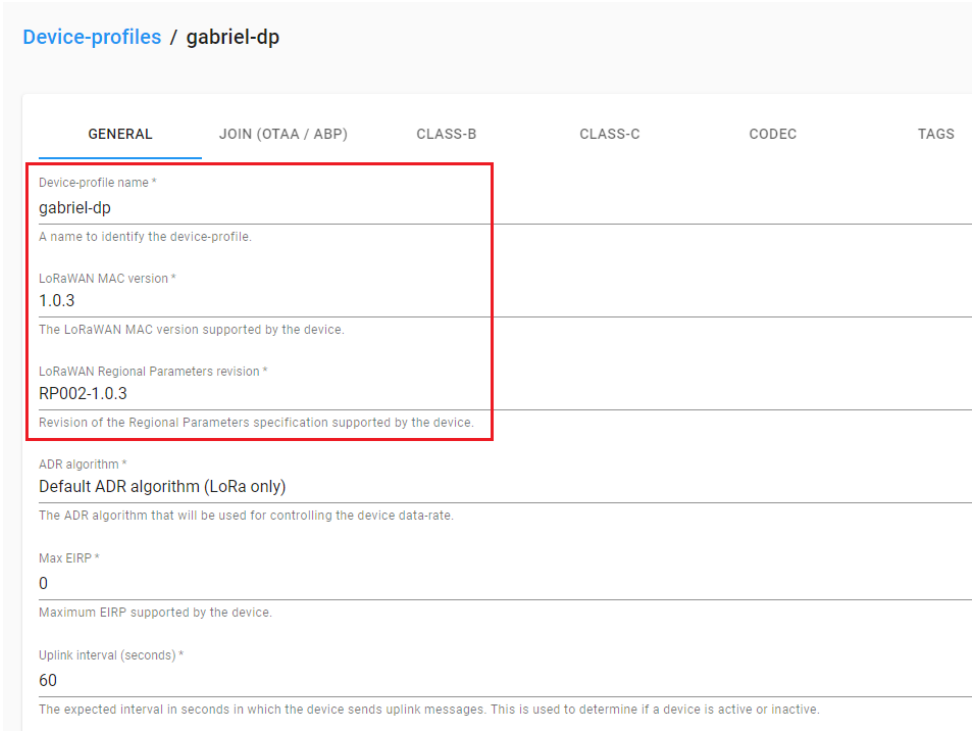
Na configuração de Gateway, são selecionados os canais da sub-banda utilizada, como mencionado nas configurações globais do Semtech Packet Forwarder.

Figura 22 – Configuração do Perfil de Serviço.



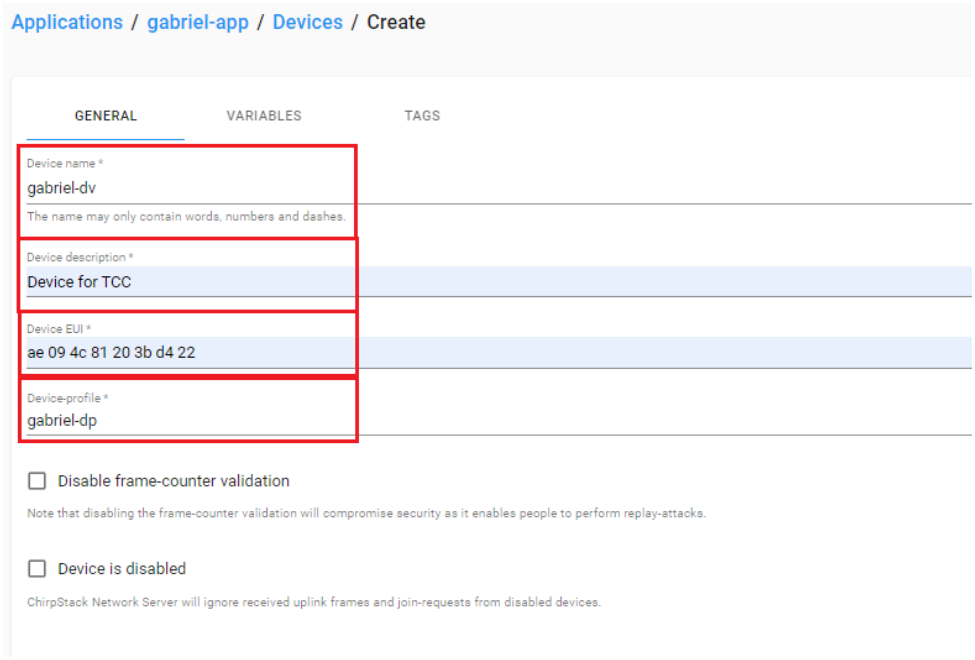

Fonte: Própria.

Figura 23 – Configuração do Perfil de Dispositivo.




Fonte: Própria.

Figura 24 – Configuração do Perfil de Aplicação.




Fonte: Própria.

Na etapa de configuração do perfil de dispositivos há uma particularidade que é a condificação e decodificação do pacote de dados recebido, que é chamada de Codec.

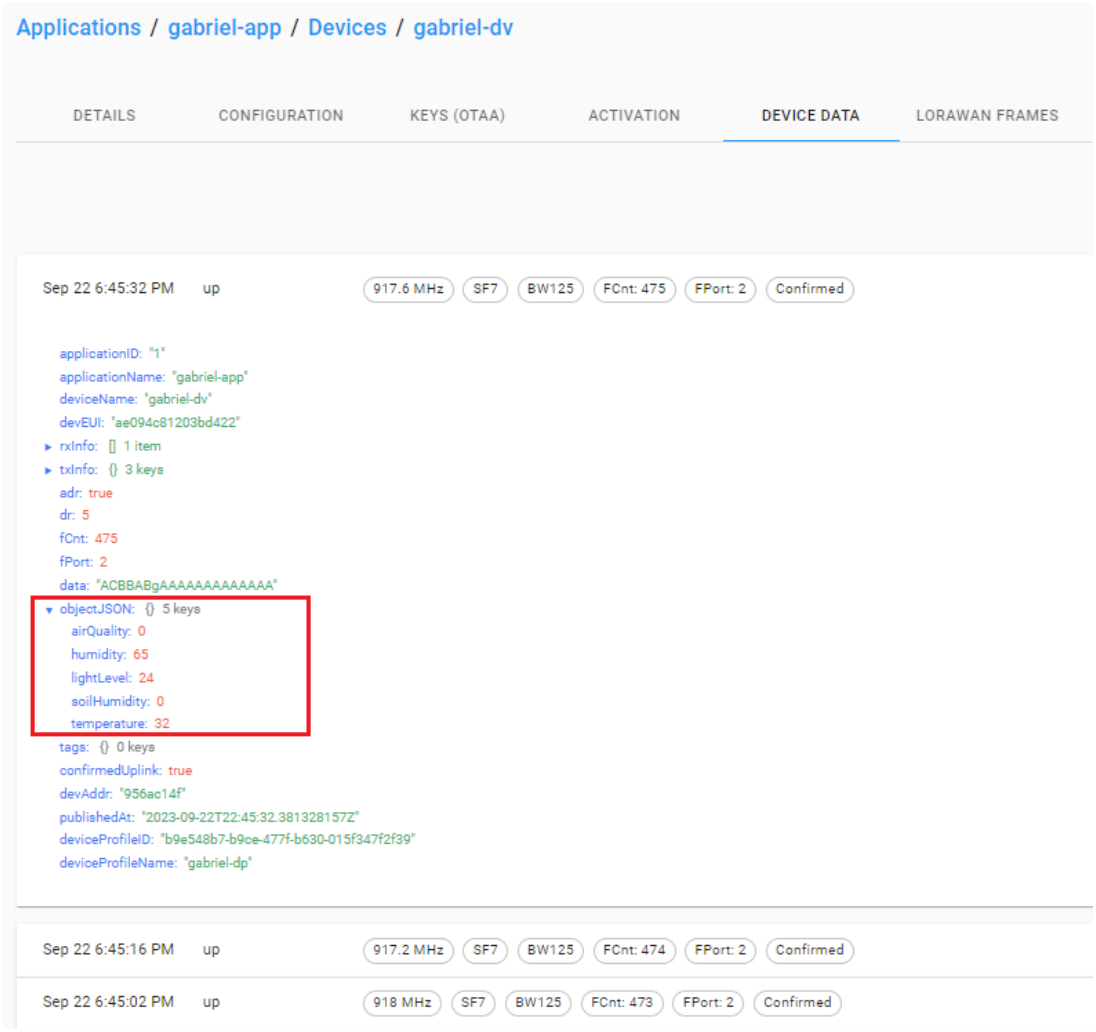
Esta etapa não pode ser negligenciada pois é ela que transforma o pacote de bits recebidos para um objeto no formato JSON, que é um formato de arquivo muito utilizado nas ferramentas de processamento de dados, permitindo uma flexibilidade e facilidade de extração e manipulação dos dados.

Figura 25 – Seção de codificação dos pacotes de dados.


Fonte: Própria.

Uma vez configurado todos os perfis e customizações de autenticação na plataforma Chirpstack, é esperado que nesta etapa já seja possível reconhecer o gateway que opera o software *Packet Forwarder* bem como visualizar os pacotes de dados enviados pelo dispositivo EndNode em formato de objeto JSON. Dessa forma, cadastra-se o dispositivo na seção *Gateway* com o mesmo *Gateway ID* estabelecido anteriormente, inicia-se o software do gateway e alimenta-se o hardware do dispositivo sensorial para que passem a operar em estado ativo. Não havendo empecilhos, é possível verificar o pacote de dados recebidos na seção *Device Data*, dentro do tópico *Devices* na aba *Applications* da plataforma Chirpstack, como abaixo:

Figura 26 – Interface de recebimento de pacotes na plataforma Chirpstack.




Fonte: Própria.

## 2.5 Ecossistema Node-RED-InfluxDB-Grafana

Após a configuração do dispositivo final, gateway e servidor LoRaWAN, tem-se um sistema que faz a medição e transmissão de dados à distância com rápida frequência, porém, brutos, desorganizados e ilegíveis à primeira vista. Como o intuito deste sistema é obter e visualizar dados de forma inteligível e que se possam fazer análises, é preciso organizar e processar os dados obtidos para que sejam observáveis de forma clara e intuitiva.

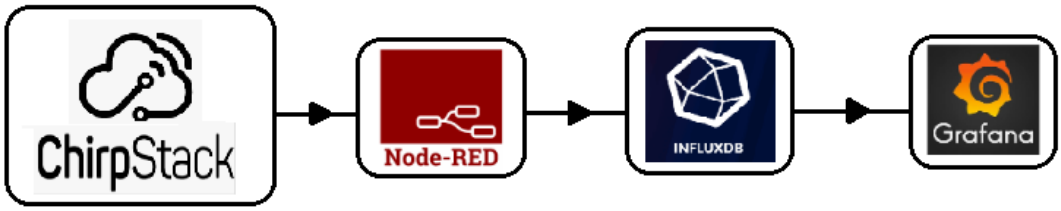
Para atingir este objetivo, foi selecionada a combinação das plataformas Node-RED, InfluxDB e Grafana. A Node-RED é uma plataforma de automação que simplifica o processamento de dados através de fluxos de trabalho. Neste caso, a Node-RED faz a conexão e processamento dos dados da Chirpstack com o InfluxDB. InfluxDB é o sistema de gerenciamento de banco de dados baseado em séries temporais que armazena os dados processados no fluxo Node-RED.

Já a Grafana é uma ferramenta de visualização que cria *dashboards* de fácil visualização para representar graficamente os dados importados do InfluxDB de maneira inteligível, facilitando o monitoramento e a tomada de decisões em tempo real.

Assim como a plataforma da Chirpstack e tendo em vista a natureza dos dados, foi possível e portanto escolhida a instalação das três plataformas no próprio Raspberry PI 3B+, que, nesta etapa, opera todos os softwares necessários de forma completamente satisfatória. O processo de instalação e configuração de parâmetros destes softwares pode ser conferida no apêndice A deste trabalho.

Dessa maneira, a sequência de processos se estabeleceu como a seguir:

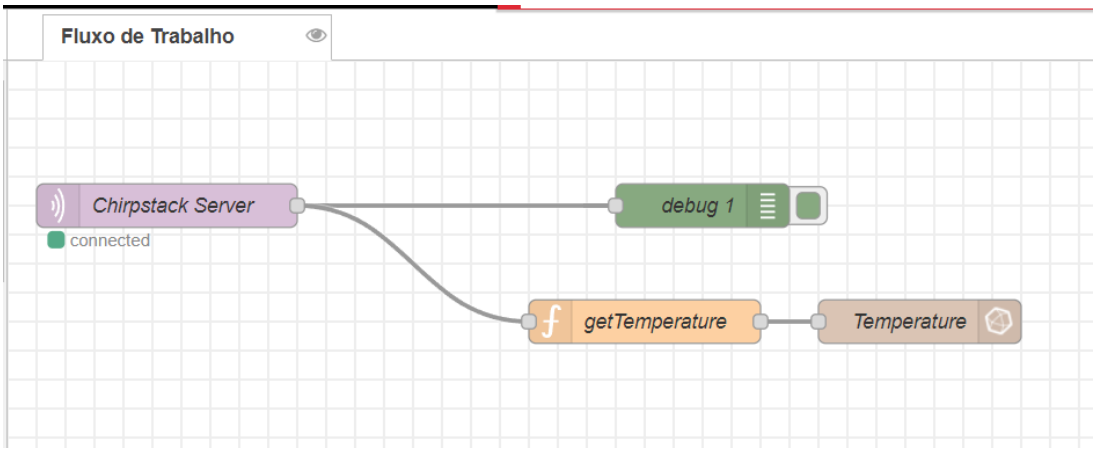
Figura 27 – Sequência de processamento dos dados.




Fonte: Própria.

Como exemplificado, iniciou-se a configuração da plataforma Node-RED. O intuito nesta primeira etapa é transportar os dados brutos recebidos da Chirpstack e armazená-los no banco de dados InfluxDB de forma ordenada. Para isso, criou-se um fluxo de trabalho abaixo:

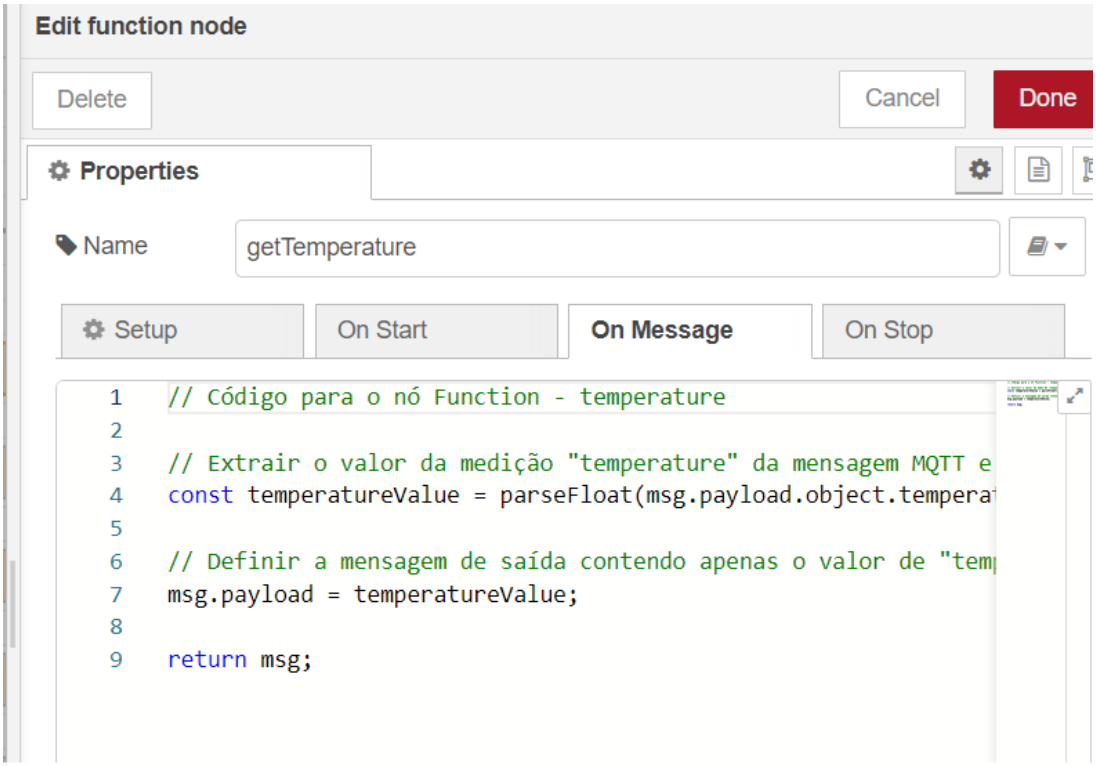
Figura 28 – Esboço do Fluxo de Trabalho utilizado.




Fonte: Própria.

O nó denominado de *Chirpstack Server* é um nó *MQTT in* que representa o tópico MQTT do servidor Chirpstack de onde os dados estão sendo recebidos. Neste nó é onde o objeto JSON obtido pela função Codec do servidor Chirpstack é transmitido para o Node-RED. O nó denominado *Debug* é um nó de depuração que opera como um verificador, que neste caso, mostra se o objeto JSON interpretado pelo Node-RED é o mesmo que se observa na plataforma Chirpstack, para efeitos de controle. Da mesma forma, outros nós são acoplados ao nó *MQTT in*, denominados de *function nodes*, que operam códigos *JavaScript* cuja função é manipular e transformar dados. Esses nós são necessários pois o objeto JSON oriundo do servidor de aplicação não está no formato compatível ao armazenamento no banco de dados InfluxDB.

Figura 29 – Exemplo de código de um *function node*.



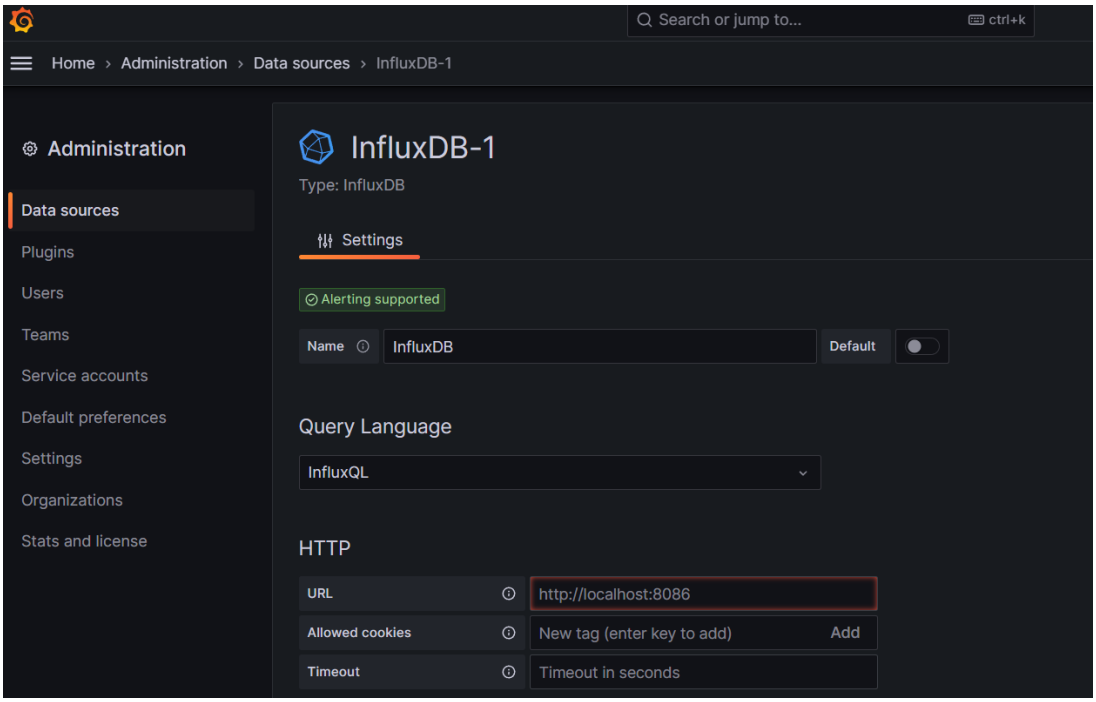

Fonte: Própria.

Dessa forma, é preciso primeiro manipular o objeto e extrair as variáveis pertinentes de forma legível ao InfluxDB. Sendo assim, cada nó executa uma função que extrai uma variável das cinco variáveis pertencentes ao objeto original e as envia aos nós *InfluxDB out*. Por sua vez, esses nós são responsáveis por alimentar o banco de dados de acordo com as informações recebidas. Cada nó *InfluxDB out* possui o campo *measurement*, que representam as tabelas onde os dados são armanezados em séries temporais. Apesar de cada tabela possibilitar a inserção de muitas *tags*, foi escolhida a abordagem de dedicar uma tabela para cada variável pois esse método permite uma configuração mais eficiente e facilitada na plataforma gráfica, que será descrita a seguir.

Uma vez que os dados foram tratados e corretamente armazenados no banco de dados, pôde-se iniciar a configuração da plataforma gráfica responsável por mostrar os dados, caracterizando a etapa final do sistema. A plataforma Grafana foi a escolhida para tal função, pois possui um sistema de integração pré-configurado para bancos de dados como MongoDB, MySQL, InfluxDB e muitos outros.

Portanto, no contexto deste projeto, foi selecionada a integração com o banco de dados InfluxDB utilizado na etapa anterior. Na seção de configuração, é necessário preencher apenas alguns campos simples a respeito do InfluxDB que foram configurados no processo de sua instalação. Esses campos são as credenciais de usuário e senha, nome do banco de dados criado e o endereço IP onde se encontra hospedado.

Figura 30 – Integração do InfluxDB à Grafana.




Fonte: Própria.

Por fim, feita a integração correta do banco de dados com a Grafana, foram preparados os gráficos de dados para o usuário final. Nesta etapa, pôde-se verificar que a plataforma conta com extensivas opções de visualização gráfica para as mais diversas finalidades e necessidades.

Para atender aos propósitos do projeto, optou-se por fazer gráficos de linha temporal para cada variável e um gráfico geral de atualização em tempo real. Dessa forma, criou-se uma *dashboard* e incluiu-se os gráficos propostos. Na interface de seleção dos parâmetros para os gráficos, selecionou-se os *measurements* que correspondem a cada tabela criada no InfluxDB e logo os dados já foram processados, visto que estes já são armazenados com um selo temporal.

Já na aba lateral direita, foram aplicadas regras de exibição como unidades de medida, limites mínimos e máximos de medição, título de medição, além de outros aspectos pertinentes:

Figura 31 – Seleção dos parâmetros e regras de exibição dos gráficos.


Fonte: Própria.

Dessa maneira, concluiu-se as etapas de desenvolvimento do sistema proposto para realizar as medições em espécies cultivadas em floriculturas e estufas. Para efeitos de testes, foram realizadas medições em algumas mudas de planta experimentais ao longo de um dia para futuras análises de comportamento de variáveis. Contudo, esta etapa já conclui a proposta deste ensaio, que consiste em criar esta ferramenta de medição. Futuros estudos, análises e aferições serão realizados pelos agricultores e floricultores especializados que farão o melhor uso dos dados obtidos.

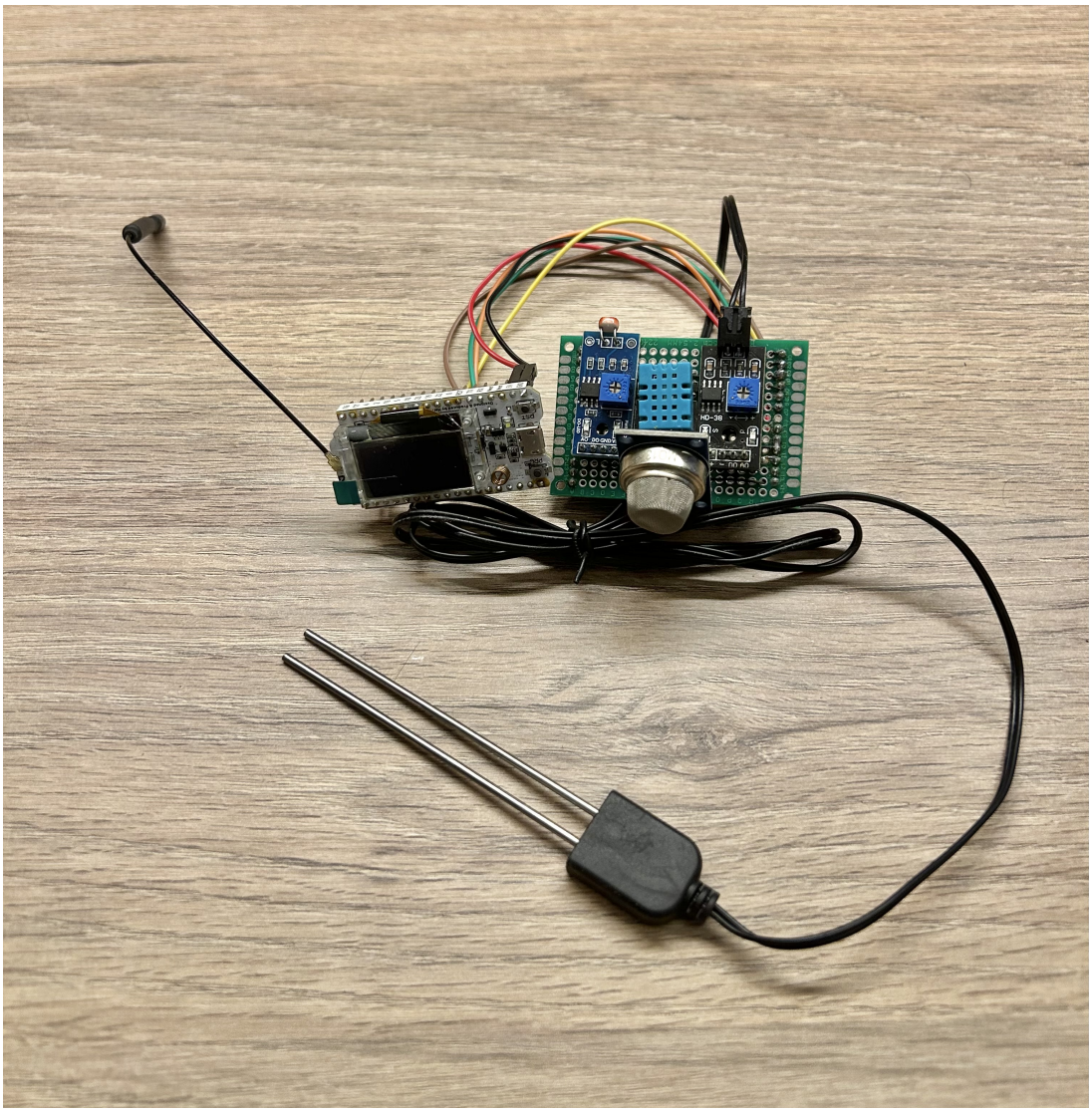
# 3 RESULTADOS

Nesta seção, portanto, serão considerados os resultados finais obtidos deste projeto, tendo em vista também algumas particularidades durante seu desenvolvimento. Serão apresentados os protótipos finais obtidos bem como as estruturas finais das plataformas utilizadas. Além disso, será observada a confibialidade do sistema através de medições feitas em duas espécies de plantas e será discutido o orçamento do projeto.

## 3.1 Protótipos Finais EndNode e Gateway

Abaixo são encontrados os modelos confeccionados para dispositivo EndNode, responsável por coletar dados através de sensores, e o dispositivo gateway, responsável por enviar esses dados para o servidor de rede, bem como realizar outras funções de hospedagem, discutidas na metodologia.

Figura 32 – Heltec ESP32 associado aos sensores como EndNode.



Fonte: Própria.

Observa-se que a conexão entre a placa de sensores e a placa ESP32 é estabelecida por meio de jumpers fêmea-fêmea. Isso ocorreu devido à decisão inicial de utilizar o microcontrolador Arduino em conjunto com o módulo LoRa Dragino, ao invés da placa ESP32, que foi escolhida posteriormente. A substituição dessas placas foi motivada pela disponibilidade de informações e recursos de suporte da comunidade para o desenvolvimento do código responsável pelo processamento e envio dos dados por meio de comunicação LoRa. Além disso, constatou-se que a placa Heltec ESP32, que já integra um módulo LoRa completo sem a necessidade de periféricos adicionais, oferece uma solução mais econômica, com uma diferença de aproximadamente R$50,00 (cinquenta reais) em relação à combinação do módulo Dragino com o Arduino. Detalhes sobre os preços serão abordados na seção de orçamento.

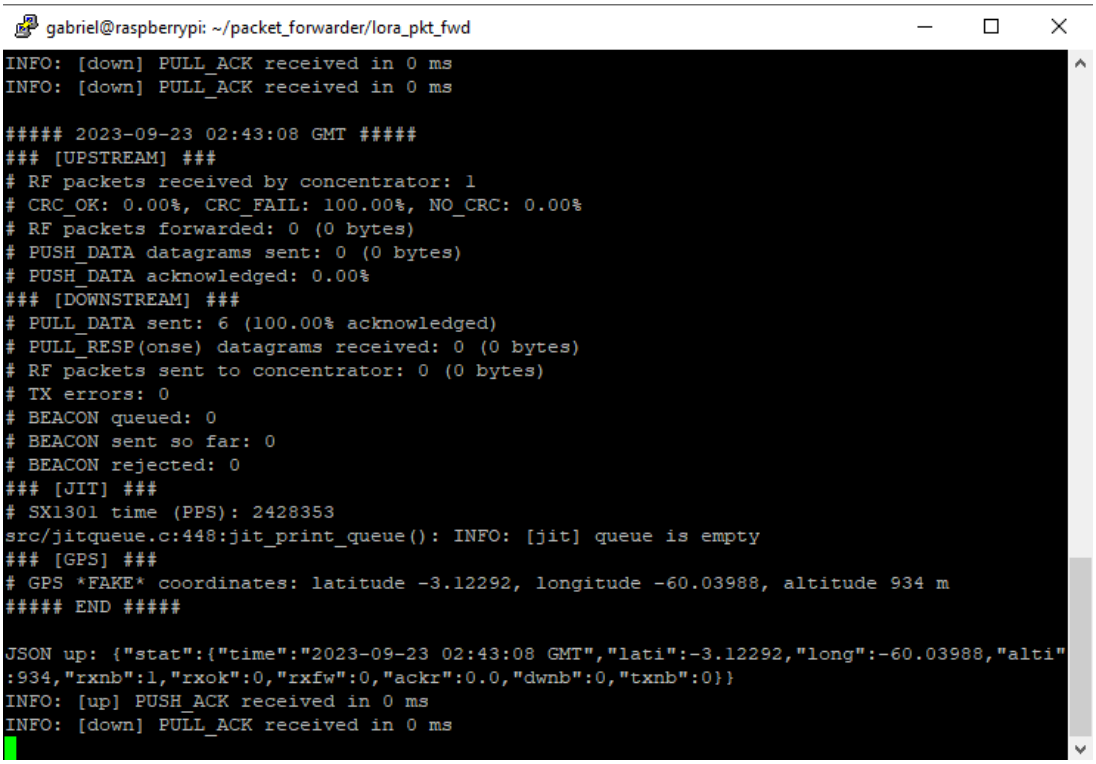
Sensores como o DHT11, LM393 LDR, MQ-135 e sensores genéricos de umidade do solo representam escolhas viáveis e eficazes em projetos LoRa de baixo custo. Esses sensores são acessíveis, de implementação simplificada e adequados para aplicações que não requerem altos níveis de precisão ou robustez. Entretanto, é importante reconhecer que, em aplicações que demandam medições mais precisas, durabilidade e confiabilidade dos equipamentos, a preferência recai sobre sensores de categoria profissional.

Sensores profissionais frequentemente possuem características que os tornam mais adequados para ambientes desafiadores ou aplicações críticas. Eles oferecem uma precisão superior, uma faixa de operação mais ampla, resistência a condições ambientais adversas e uma vida útil mais prolongada. Adicionalmente, muitos deles possuem certificações que garantem sua qualidade e confiabilidade, tornando-os ideais para aplicações onde a integridade dos dados e a segurança dos equipamentos são de suma importância.

Para exemplificar esta ideia, podemos considerar o exemplo do sensor de gás MQ-135. Embora seja uma das opções mais simples no quesito detecção de gases, no desenvolver do projeto foi constatado que o MQ-135 não foi especificamente projetado para medir concentrações de gás carbônico. Na verdade, este sensor é usado para identificar a presença de gases tóxicos no ambiente como amônica(NH3), dióxido de nitrogênio(NO2), dióxido de enxofre (SO2) e outros, como o próprio dióxido de carbono ($CO_2$). Contudo, este sensor não é capaz de fornecer concentrações precisas desses gases no ambiente. Para aplicações que exigem uma acurácia mais elevada na medição do gás carbônico, seria necessário recorrer a sensores específicos para esse fim, como sensores de infravermelho não dispersivo (NDIR) ou sensores eletroquímicos projetados para $CO_2$.

No entanto, é relevante destacar que esses sensores mais precisos tendem a ser substancialmente mais caros, o que pode resultar em um aumento significativo nos custos do projeto e, possivelmente, tornar a solução proposta financeiramente inviável. Portanto, a escolha do sensor apropriado deve ser cuidadosamente ponderada, considerando tanto a precisão requerida quanto as restrições orçamentárias do projeto.

Figura 33 – Semtech Packet Forwarder compilado no módulo Raspberry PI.



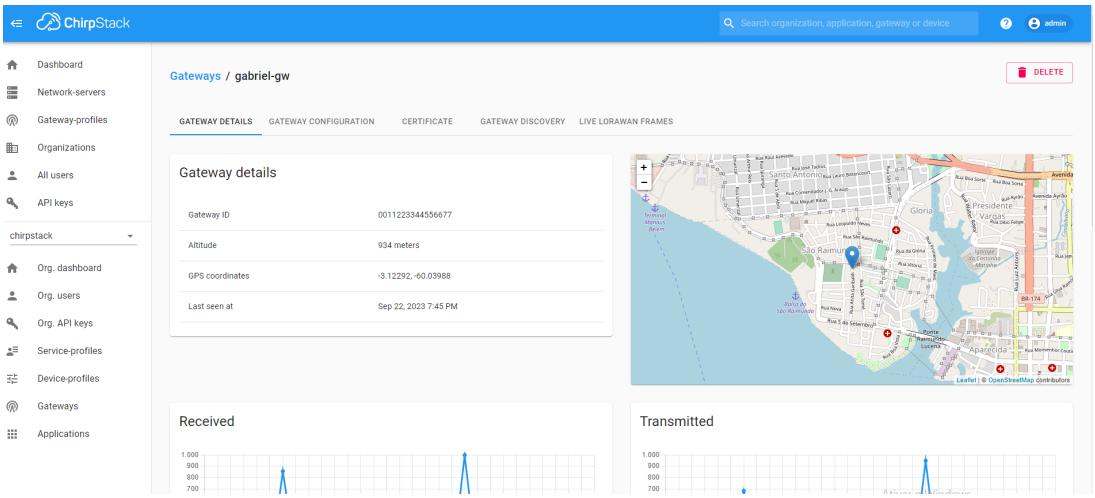

Fonte: Própria.

Quanto a solução que envolve o uso do módulo RAK2243 acoplado ao Raspberry Pi 3B+, com a implementação do software Semtech Packet Forwarder para gateway LoRa, demonstrou-se altamente eficaz e atendeu plenamente aos requisitos do projeto. Além disso, essa abordagem revelou-se economicamente viável, uma vez que a configuração pode ser montada praticamente de forma caseira, sem a necessidade de adquirir soluções prontas de fabricantes. Essa combinação de hardware e software proporcionou um desempenho notável e, ao mesmo tempo, representou uma opção acessível para a implementação do gateway, tornando-se a escolha ideal para atender às demandas do projeto.

## 3.2 Configuração Final das Plataformas

Ao todo foram configuradas quatro plataformas onde foram feitas configurações e *layouts* necessários: Chirpstack,Node-RED,InfluxDB e Grafana. Essas plataformas podem ser acessadas e, caso necessário, reconfiguradas a qualquer momento, o que torna o projeto muito versátil. Abaixo podem ser conferidas os designs e resultados finais de cada uma delas:

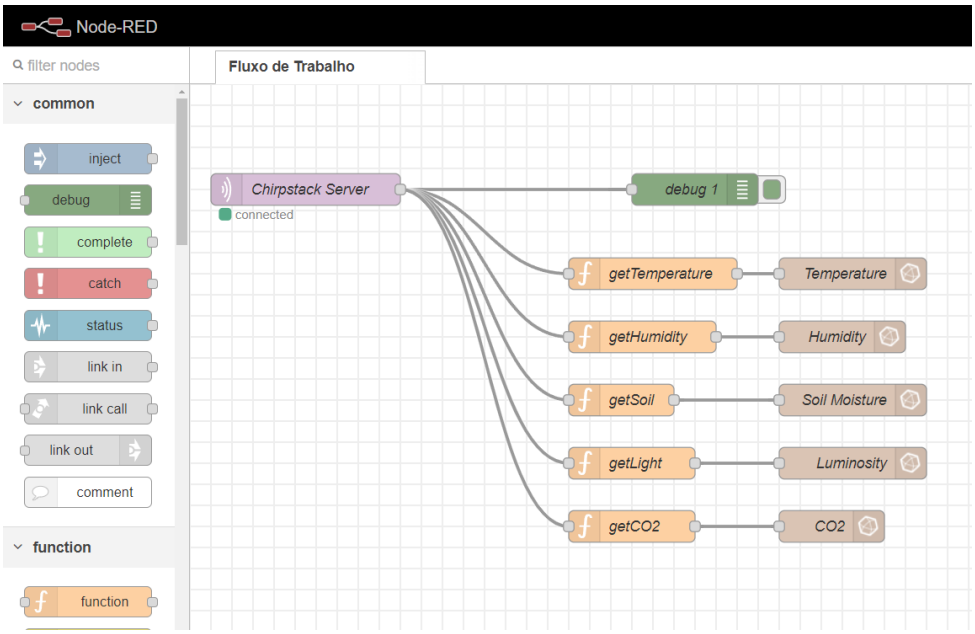
Figura 34 – Interface Final Chirpstack.




Fonte: Própria.

Como detalhado na seção de metodologia, o fluxo de trabalho utilizado foi o seguinte:

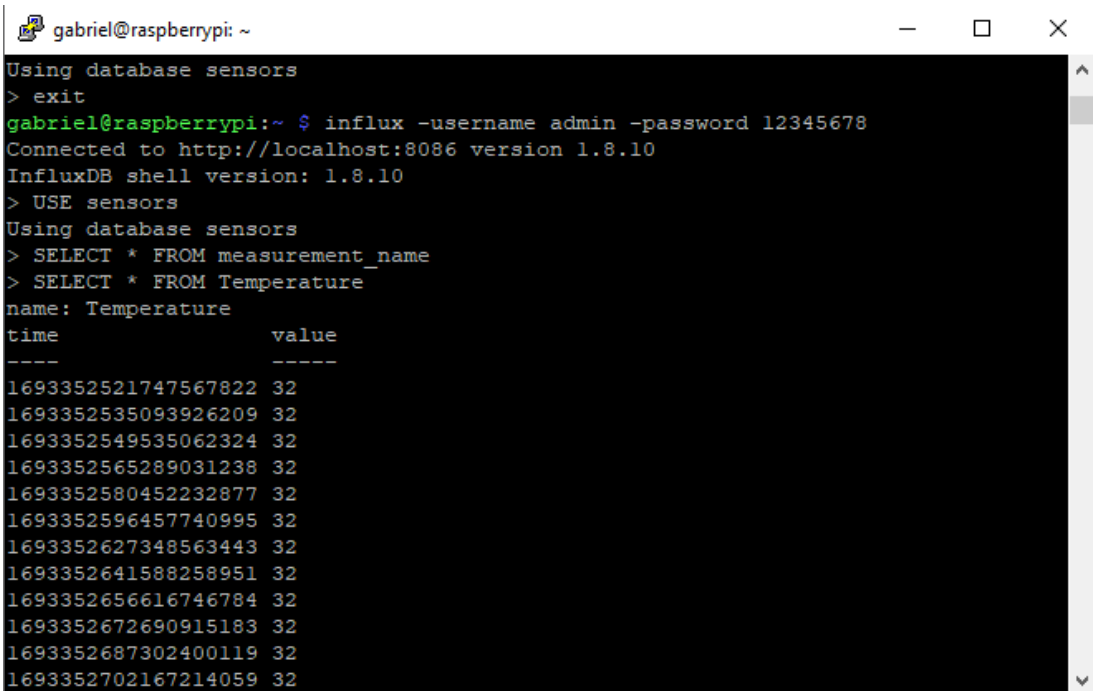
Figura 35 – Fluxo de Trabalho Final na Node-RED.




Fonte: Própria.

Após o tratamento de dados, os dados foram devidamente armazenados no banco de dados InfluxDB. Como visto no fluxo de trabalho, cada função retorna um valor de medição específico e de forma legível, que pode ser interpretada e armazenada no banco. Cada nó *InfluxDB Out* cria uma tabela própria que é intitulada com sua respectiva medição. Os dados armazenados podem ser facilmente vistos na interface do InfluxDB, acessada pelo terminal do Raspberry PI:

Figura 36 – Dados Armazenados no Banco de Dados InfluxDB.

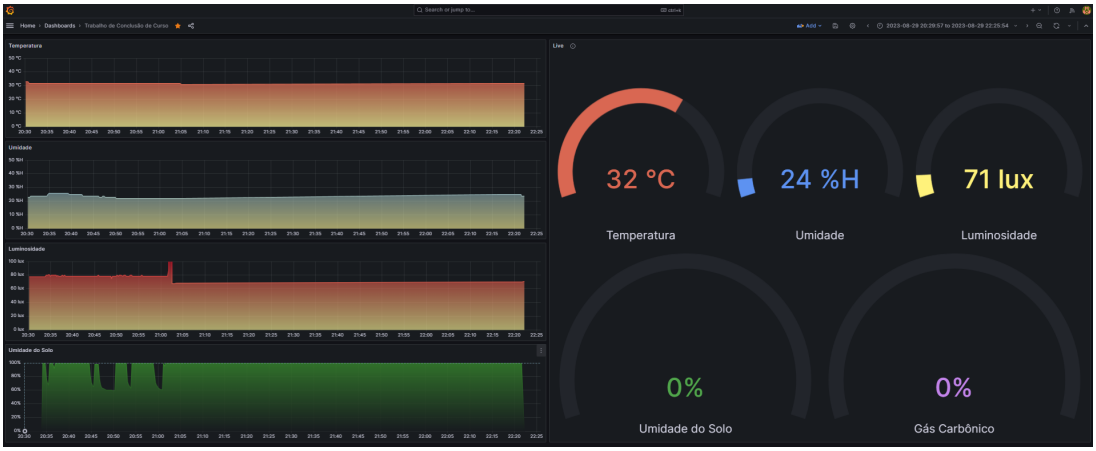



Fonte: Própria.

É importante ressaltar a vantagem de se usar este banco de dados. É possível notar que ao lado de cada medição há um marcador temporal associado. Isso é bastante útil para a criação dos gráficos de análise histórica, pois já fornece os parâmetros necessários para a confecção dos mesmos.

Por fim, tem-se a exibição dos dados na plataforma Grafana. Foram criados gráficos temporais para demonstrar a flutuação das medições ao longo tempo e um gráfico de medidas instantâneas. Todos esses gráficos foram compilados em uma única *dashboard* na plataforma, que pode ser visualizada abaixo:

Figura 37 – Painel de Visualização das Medições Instantâneas e Temporais.




Fonte: Própria.

Uma vantagem da ferramenta é a possibilidade de definir parâmetros temporais de visualização. Ela permite determinar a taxa de atualização dos dados, que, para os propósitos do projeto, foi determinada uma faixa de 5 segundos entre medições. Além disso, caso o usuário final queira, é possível também selecionar uma amostra temporal para analisar medições feitas durante o período, como horários ou dias específicos, o que pode ser interessante em alguns cenários.

A representação gráfica dos resultados de medições ao longo do tempo provenientes dos sensores se mostra fundamental na pesquisa e monitoramento das variáveis. Primeiramente, os gráficos que mostram a evolução temporal das variáveis medidas fornecem uma visão completa e histórica do comportamento dessas grandezas. Isso permite a identificação de tendências, padrões sazonais, flutuações e anomalias que podem ser imperceptíveis em uma análise pontual. Essas séries temporais, são essenciais para a compreensão da dinâmica dos processos ou fenômenos estudados, além de oferecerem *insights* valiosos para previsões futuras.

Além disso, a disponibilidade de um gráfico em tempo real que apresenta os valores atuais das variáveis medidas é de extrema importância. Esse tipo de gráfico fornece uma representação instantânea e contínua do estado atual do sistema ou experimento, permitindo a detecção imediata de desvios ou comportamentos anômalos. Isso é crucial para a tomada de decisões em tempo real e a implementação de medidas corretivas quando necessário. Os gráficos em tempo real também são valiosos para a supervisão de sistemas críticos, onde a resposta rápida a eventos inesperados é crucial para a segurança e o desempenho.

## 3.3 Medições Experimentais

Uma vez concluido o desenvolvimento do sistema de medição e atestada a sua funcionalidade, é preciso submeter o projeto em ambiente real para atestar sua confiabilidade.

Para este propósito, foram escolhidas duas espécies de mudas de plantas cultivadas na residência do discente para realizar estas medições: a Palmeira Areca (*Dypsis lutescens*) e a Coroa-de-Cristo (*Euphorbia milii*). As medições foram realizadas nos seguintes intervalos de tempo: das 16h40 às 19h do dia 22 de setembro de 2023 e das 9h20 às 10h20 do dia 23 de setembro de 2023, respectivamente.

Figura 38 – Medição feita em muda de Palmeira Areca.

Fonte: Própria.

Para a primeira análise, o sistema foi instalado no vaso que abriga a muda da Palmeira Areca. O cenário e intervalo escolhidos compreendiam o final de tarde, quando a iluminação natural está diminuindo e a umidade do ar está aumentando. No entanto, houve fatores atenuantes devido ao clima consideravelmente nublado e pré-chuvoso do dia 22, o que resultou em uma significativa redução na incidência de luz e um aumento substancial na umidade relativa do ar.Embora essas condições tenham sido impostas pelo ambiente externo, não ocorreu precipitação durante o período de análise. No entanto, como parte da investigação, foi decidido aplicar irrigação ao solo em um determinado momento, com o objetivo de avaliar se essa ação se refletiria nas métricas capturadas pelo sistema e na interface gráfica. O painel obtido desta medição se encontra a seguir:

Figura 39 – Painel obtido a partir das medições em Palmeira.


Fonte: Própria.

Devido ao clima, como mencionado anteriormente, a umidade relativa do ar manteve-se consistentemente elevada, atingindo valores de até 67%. Conforme o horário da medição, os níveis de luminosidade diminuíram de 42 lux para 21 lux, o que era previsível devido ao anoitecer. Além disso, foram observados ligeiros picos de aumento na umidade do solo devido à irrigação programada, que alcançaram medições de 59% em comparação com a média de 53% que se manteve durante o período de coleta de dados, como era esperado. Por fim, é importante destacar que não foram registradas variações significativas de temperatura durante todo o período de análise, com os valores permanecendo constantes na faixa de 32ºC pois no intervalo estabelecido essa métrica tende a se estabilizar.

Para a segunda análise, o sistema foi instalado no vaso que abriga a muda da Coroa-de-Cristo. Em contraste com o cenário da primeira análise, a coleta de dados foi realizada durante a manhã do dia 23, caracterizada por um clima ensolarado e com pouquíssimas nuvens.

Figura 40 – Medição feita em muda de Coroa-de-Cristo.

Fonte: Própria.

Ao contrário do primeiro experimento, não foram realizadas intervenções, como irrigação esporádica, com o propósito de observar a variação natural do sistema. Isso proporcionou uma perspectiva do comportamento da planta em condições ambientais sem manipulações externas. Os dados obtidos foram os seguintes:

Figura 41 – Painel obtido a partir das medições em Coroa-de-Cristo.


Fonte: Própria.

É notável observar mudanças significativas nas variáveis medidas devido à variação do horário ao longo da manhã. Podemos perceber como, ao se aproximar do meio-dia, ocorrem elevações consideráveis na temperatura e na luminosidade, enquanto a umidade do ar diminui significativamente.

A temperatura inicial de 31ºC aumentou consideravelmente, atingindo 43ºC no final do período de análise, o que representa um aumento de 12 graus. Da mesma forma, a luminosidade inicial de 299 lux atingiu picos de 1247 lux. Essas mudanças estão relacionadas com o fato de que, à medida que o dia avança, a incidência de luz solar direta aumenta, causando um aumento na temperatura e na luminosidade, enquanto a umidade do ar diminui devido à evaporação mais intensa.

No entanto, é importante notar que a umidade do solo teve apenas uma ligeira diminuição, passando de 55% para 50%. Isso era esperado, indicando que a planta ainda não apresentava uma necessidade imediata de irrigação, embora essa necessidade possa surgir a depender das características individuais da planta e o comportamento dessa grandeza ao longo do tempo.

Portanto, os dados viabilizados pelo sistema e experimentos fornecem *insights* valiosos para otimizar o manejo de plantas, ajustando fatores como irrigação e iluminação com base nas necessidades reais das plantas. Além disso, o sistema abre portas para futuras pesquisas, como a criação de algoritmos de automação que ajustam automaticamente as condições do ambiente para otimizar o crescimento das plantas em tempo real e conhecimento mais aprofundado do comportamento de cada espécie de planta.

## 3.4 Orçamento do Sistema

Há muitas maneiras de produzir um sistema de medição como o descrito neste trabalho, contudo, muitas delas exigem um investimento monetário elevado, o que pode ser inviável para muitos cenários onde o orçamento é escasso ou a finalidade não demanda aparato técnico elevado. Tendo isso em vista, é importante desenvolver soluções o mais viáveis possíveis para atender a todos os tipos de necessidades, além do retorno finaceiro justificar a solução.

Tendo isso em vista, o sistema foi projetado para atender essa demanda e viabilizar uma solução de baixo custo, o tanto quanto possível. A tabela abaixo oferece um detalhamento completo do orçamento associado à construção do sistema final. Os custos relacionados aos componentes utilizados, como sensores, placas de desenvolvimento e outros acessórios, somam um valor total de R$ 1.785,76. Felizmente, todos os softwares utilizados são gratuitos.

Tabela 2 – Cálculo de Custos de Materiais no Projeto

| **Item** | **Valor Investido (R$)** |
|---|---|
| Placa Heltec ESP32 V.2 | 179 |
| Sensor de Temperatura DHT11 | 24 |
| Sensor de Luminosidade LDR com LM393 | 23,05 |
| Sensor de gases MQ-135 | 9,03 |
| Sensor de Umidade do Solo genérico | 37,72 |
| Antenas operando na faixa de 915MHz | 30 |
| Micro-computador Raspberry PI 3B+ | 548 |
| Módulo LoRaWAN RAK2245 para gateway | 875,06 |
| Cartão Micro-SD de 32gb de memória | 39,90 |
| Fonte 5V/2A | 18 |
| Cabos USB e Micro USB | 40 |
| Jumpers fêmea/fêmea | 10 |
| Placa perfurada | 10 |
| Software Arduino IDE | Gratuito |
| Software Lora-Gateway Semtech Packet Forwarder | Gratuito |
| Software de emulação de terminal PuTTY | Gratuito |
| Plataforma Lora Chirpstack | Gratuito |
| Software FBP Node-RED | Gratuito |
| Software de Banco de Dados InfluxDB | Gratuito |
| Software de Interface Gráfica Grafana | Gratuito |
| **Total** | 1785,76 |

Fonte: Própria.

É relevante observar que, embora o custo do sistema completo possa chegar aos quatro dígitos, ele já inclui uma solução de gateway integrada. Em contraste, ao optar por gateways de outras marcas amplamente utilizadas, como o The Things Gateway ou Multitech, muitas vezes, os custos individuais desses equipamentos podem facilmente ultrapassar a marca de R$ 2.000,00. Portanto, ao considerar a solução completa proposta, que inclui tanto o dispositivo de medição quanto o gateway, o custo se torna ainda mais vantajoso, tornando-a uma escolha econômica e completa para implementações de IoT robustas e acessíveis.

Uma análise interessante a ser destacada é a escolha da placa de desenvolvimento Heltec ESP32 V.2. Como mencionado anteriormente na subseção de protótipos finais, a princípio seria utilizada uma alternativa que envolve a Placa Arduino UNO R3 juntamente com o Shield LoRa Dragino para dispositivo EndNode. Contudo, esta solução foi substituida pela placa atual devido a algumas considerações feitas durante o projeto, descritas a seguir. Abaixo se encontra a comparação de preço de mercado entre as soluções idealizadas:

Tabela 3 – Comparação de Custos entre Placa Arduino e ESP32.

| **Opção 1: Placa de Desenvolvimento Heltec ESP32 V.2** | |
|---|---|
| **Item** | **Preço de Mercado (R$)** |
| Placa de Desenvolvimento Heltec ESP32 V.2 | 179 |
| **Opção 2: Placa Arduino UNO R3 + Shield LoRa Dragino** | |
| Placa Arduino UNO R3 | 88,32 |
| Shield LoRa Dragino | 133,90 |
| **Total** | **222,22** |

Fonte: Própria.

Enquanto a Placa Arduino UNO R3 tem um custo de R$ 88,32 e o Shield LoRa Dragino é avaliado em R$ 133,90, a placa Heltec ESP32 V.2 apresenta um preço de R$ 179. É importante observar que a placa Heltec ESP32 V.2 oferece não apenas uma economia financeira, mas também uma série de vantagens funcionais em relação à alternativa Arduino. A ESP32 V.2 integra recursos como conectividade Wi-Fi e Bluetooth, processamento de alto desempenho e maior capacidade de memória, o que a torna ideal para aplicações LoRa. Além disso, a integração dessas funcionalidades em uma única placa elimina a necessidade de acessórios adicionais, simplificando o desenvolvimento do projeto.

Portanto, a escolha da placa Heltec ESP32 V.2 não apenas otimiza os custos, mas também aprimora significativamente as capacidades do sistema, tornando-a uma escolha sólida e eficaz para projetos de LoRa que exigem um desempenho robusto e recursos adicionais de conectividade.

# CONCLUSÕES

A implementação bem-sucedida do sistema de monitoramento baseado no ESP32, Raspberry Pi com RAK2243 como gateway, plataforma ChirpStack e o ecossistema NodeRed-InfluxDB-Grafana demonstrou ser uma solução eficaz para a coleta e visualização de dados ambientais na criação de culturas. A integração dessas tecnologias permitiu uma coleta contínua de informações críticas, como luminosidade, temperatura e umidade do solo, proporcionando ao consumidor final uma visão precisa das condições ambientais em tempo real.

Os resultados obtidos ao aplicar esse sistema a espécies vegetais atestaram sua capacidade de viabilizar estudos sobre as necessidades específicas de cada cultura. A flexibilidade oferecida pelo dispositivo EndNode permitiu a personalização dos sensores de acordo com as particularidades de cada planta, garantindo que as condições ideais possam ser estabelecidas. Isso resulta em um crescimento mais saudável e monitoramento constante, que possibilita a redução significativa de perdas de recursos biológicos e financeiros.

Além disso, a integração do sistema com o NodeRed, InfluxDB e Grafana proporcionou uma plataforma robusta de visualização e análise de dados. Com ela, agricultores e cuidadores podem acessar remotamente informações críticas por meio de uma interface gráfica intuitiva, permitindo a tomada de decisões informadas e uma intervenção rápida quando necessário. A capacidade de conduzir análises detalhadas ao longo do tempo oferece uma visão mais profunda sobre o impacto das condições ambientais no crescimento das plantas.

Os experimentos realizados com o sistema em condições reais de cultivo de plantas forneceram resultados significativos. Ao analisar as medições da Palmeira Areca, pode-se observar como o sistema capturou as variações na umidade do solo devido à irrigação programada, bem como as mudanças na luminosidade e na umidade do ar à medida que o dia se transformava em noite. Isso demonstra a capacidade do sistema de responder a mudanças ambientais em tempo real e fornecer informações valiosas.

Da mesma forma, ao analisar as medições da Coroa-de-Cristo, pudemos observar como as condições climáticas e a variação ao longo do dia afetaram a temperatura, a luminosidade e a umidade do ar. Esses dados realçam a importância de ajustar o manejo das plantas com base nas condições ambientais em constante mudança, otimizando assim o crescimento e o desenvolvimento das culturas.

Uma consideração fundamental ao avaliar qualquer sistema de monitoramento é sua viabilidade econômica. Nesse sentido, o sistema proposto se destaca como uma solução acessível, especialmente quando comparado a alternativas que envolvem equipamentos de gateway mais caros. A escolha da placa Heltec ESP32 V.2, além de proporcionar economia financeira, também oferece vantagens funcionais que aumentam o valor geral do sistema. É importante ressaltar que o custo total do sistema, incluindo componentes e software, foi de R$ 1.785,76. Esse custo inclui tanto o dispositivo de medição quanto o gateway, tornando-o uma opção completa e econômica para implementações de IoT em larga escala. Isso significa que mesmo projetos com orçamentos limitados podem se beneficiar do nosso sistema de monitoramento.

Quanto à integração do módulo de sensor de gás, é crucial reconhecer que, embora tenhamos avaliado sua viabilidade, enfrentamos desafios significativos para obter resultados confiáveis com o sensor MQ-135. Este sensor, embora acessível, não é especializado na medição precisa de concentrações de gás carbônico ($CO_2$), o que limita sua aplicabilidade em cenários que exigem alta precisão. Para medições mais precisas de $CO_2$, seria necessário recorrer a sensores específicos, como sensores de infravermelho não dispersivo (NDIR) ou sensores eletroquímicos projetados exclusivamente para $CO_2$. No entanto, é importante destacar que esses sensores mais precisos tendem a ser consideravelmente mais caros, o que poderia aumentar substancialmente os custos do projeto e, potencialmente, tornar a solução financeiramente inviável. Assim, a decisão de não integrar o módulo de sensor de gás foi baseada na precisão necessária para a aplicação e nas restrições orçamentárias do projeto. Para melhorar o projeto, podemos explorar alternativas de sensores de gás que equilibrem precisão e custo, ampliando a gama de gases monitorados e aprimorando a utilidade do sistema em diversas situações.

Em adição, é importante destacar que diversas melhorias podem ser implementadas no projeto. Uma dessas melhorias é a expansão da rede com a adição de mais dispositivos finais, permitindo uma cobertura mais abrangente em áreas de cultivo maiores ou mais complexas. Além disso, a otimização do design dos módulos, incluindo sistemas de refrigeração e proteção dos componentes internos, pode garantir um desempenho mais confiável e a segurança dos dispositivos. Estudar e quantificar o alcance da solução em quilômetros, bem como a criação de mapas de calor, podem proporcionar insights valiosos sobre a distribuição dos dados e a eficácia da rede em diferentes cenários. Essas são apenas algumas das muitas possibilidades de aprimoramento que podem ser exploradas, demonstrando o contínuo potencial de desenvolvimento do sistema de monitoramento proposto.

Portanto, o sistema de monitoramento desenvolvido demonstrou sua eficácia em condições reais de criação de plantas, oferecendo uma solução econômica e acessível para agricultores e pesquisadores. Com sua flexibilidade, escalabilidade e potencial para melhorias contínuas, essa tecnologia tem o potencial de revolucionar a agricultura, aumentando a eficiência e reduzindo as perdas, ao mesmo tempo em que contribui para uma compreensão mais profunda do crescimento das plantas.

# REFERÊNCIAS


A Gentle Introduction to LoRaWAN Gateways & Nodes. 2021. Disponível em: <https://www.seeedstudio.com/blog/2021/04/27/a-gentle-introduction-to-lorawan-gateways-nodes/: :text=LoRaWAN

APP de análise de dados Grafana no Linux via Snap. 2020. Disponível em: ⟨https://www.edivaldobrito.com.br/app-de-analise-de-dados-grafana-no-linux-via-snap/⟩.

AWANG, Z. Gas sensors: A review. ***Sens. Transducers***, v. 168, n. 4, p. 61–75, 2014.

BROOKER, G. ***Introduction to Sensors for Ranging and Measurement***. [S.l.]: SciTech Publishing, 2009.

CAMERON, N. ***Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices***. [S.l.]: Apress, 2021. ISBN 978-1-4842-6335-8 (pbk), 978-1-4842-6336-5 (electronic).

CHIRPSTACK. 2020. Disponível em: ⟨https://www.chirpstack.io/⟩.

CHIRPSTACK Architecture. 2020. Disponível em: ⟨https://www.chirpstack.io/project/architecture/⟩.

FRAZÀO, D.; MARTINS, D.; SILVA, E. Long-range network (lora) behavior in the amazon region in a fluvial environment. In: SPRINGER. ***Brazilian Technology Symposium***. [S.l.], 2022. p. 391–398.

FRAZÃO, D.; SILVA, E. Characterization of the behavior of lora networks in a fluvial environment in the rio negro. In: SPRINGER. ***Brazilian Technology Symposium***. [S.l.], 2021. p. 363–369.

GRAFANA Logo Vector SVG PDF AI EPS CDR Free Download. 2022. Disponível em: ⟨https://logowik.com/grafana-logo-vector-svg-pdf-ai-eps-cdr-free-download-11813.html⟩.

HAGINO, T. ***Practical Node-RED Programming***. Livery Place, 35 Livery Street, Birmingham, B3 2PB, UK: Pack Publishing Ltd., 2021. ISBN 978-1-80020-159-0.

HANAN, J. J. ***GREEHOUSES: Advanced Technology for Protected Horticulture***. [S.l.: s.n.], 1998.

HUGHES, J. M. ***Arduino, A Technical Reference: A handbook for technicians, engineers, and makers***. 1st. ed. [S.l.: s.n.], 2016.

INFLUXDB. 2018. Disponível em: ⟨https://dbdb.io/db/influxdb/revisions/8⟩.

KEY Concepts - Data Elements. 2019. Disponível em: ⟨https://docs.influxdata.com/influxdb/v2/reference/key-concepts/data-elements/⟩.

MAUSETH, J. D. ***Botany: An introduction to plant biology***. 6th. ed. [S.l.: s.n.], 2017.

MONTAGNY, S. ***LoRa - LoRaWAN and Internet Of Things: A low power, long range, wireless technology***. [S.l.: s.n.], 2021.

MOURãO, I. d. M. ***Manual de Horticultura no Modo de Produção Biológico***. 4990-706 Ponte de Lima: Escola Superior Agrária de Ponte de Lima/IPVC Refóios, 2007.

NODE-RED Resources. 2017. Disponível em: ⟨https://nodered.org/about/resources/⟩.

OLIVEIRA, M. R. L. d. et al. Desenvolvimento de protótipo de aquisição de dados de localização de transportes fluviais na região amazônica utilizando rádio lora. Universidade do Estado do Amazonas, 2022. Disponível em: ⟨http://repositorioinstitucional.uea.edu.br//handle/riuea/4502⟩.

RAK2245 Pi HAT for Raspberry Pi 3B+ 16G TF Card Quick Start LoRaWAN Application Kit. 2020. Disponível em: ⟨https://uk.pi-supply.com/products/rak2245-pi-hat-raspberry-pi-3b-16g-tf-card-quick-start-lorawan-application-kit⟩.

RASPBERRY Pi 3 Model B+. 2021. Disponível em: ⟨https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/⟩.

SENEVIRATNE, P. ***Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks***. [S.l.: s.n.], 2019.

WIFI LoRa 32 V2 User Manual. 2020. Disponível em: ⟨https://resource.heltec.cn/download/WiFi_LoRa_32/WIFI_LoRa_32_V2.pdf⟩.

# APÊNDICE A - PROCESSOS DE INSTALAÇÃO DA INFRAESTRUTURA DE SOFTWARE NO GATEWAY

Nesta seção encontram-se os procedimentos de instalação e configuração dos softwares hospedados no Raspberry PI 3B+, utilizado para operar como o gateway do sistema. Abaixo segue a descrição dos comandos inseridos no terminal serial do SBC.

Começou-se então pela camada de software do servidor Chirpstack. Primeiramente tem-se a instalação dos pacotes essenciais, como o Mosquitto (servidor MQTT), Redis (sistema de armazenamento em memória), e PostgreSQL (um sistema de gerenciamento de banco de dados relacional):

```
sudo apt install mosquitto mosquitto-clients redis-server
   redis-tools postgresql
```

Em seguida, inicia-se a configuração dos bancos de dados e usuários do banco PostgreSQL. O comando abaixo é utilizado para entrar no cliente do mesmo:

```
sudo -u postgres psql
```

Uma vez na interface configura-se os usuários e senhas para o banco de dados *ChirpStack Application Server* (chirpstack_as) e *ChirpStack Network Server* (chirpstack_ns). Além disso, são criados os bancos de dados associados e extensões importantes necessárias para recursos de pesquisa e armazenamento de metadados:

```
-- set up the users and the passwords
-- (note that it is important to use single quotes and a
   semicolon at the end!)
create role chirpstack_as with login password 'dbpassword';
create role chirpstack_ns with login password 'dbpassword';

-- create the database for the servers
create database chirpstack_as with owner chirpstack_as;
create database chirpstack_ns with owner chirpstack_ns;

-- change to the ChirpStack Application Server database
\c chirpstack_as

-- enable the pq_trgm and hstore extensions
-- (this is needed to facilitate the search feature)
```

```
create extension pg_trgm;
-- (this is needed to store additional k/v meta-data)
create extension hstore;

-- exit psql
\q
```

Prosseguiu-se para a configuração do sistema de gerenciamento de pacotes APT, que lida com a instalação de pacotes relacionados à segurança, a obtenção da chave de autenticação necessária para o repositório ChirpStack e a configuração do repositório em si.

```
sudo apt install apt-transport-https dirmngr

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
    1CE2AFD36DBCCA00

sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/
   deb stable main" | sudo tee /etc/apt/sources.list.d/
   chirpstack.list

sudo apt update
```

Por fim, os três conjuntos de comandos abaixo referem-se à instalação e configuração dos componentes principais da ChirpStack, incluindo o ChirpStack Gateway Bridge, ChirpStack Network Server e ChirpStack Application Server. Cada conjunto de comandos instala o respectivo componente, inicia-o imediatamente e o configura para iniciar automaticamente durante o boot do sistema.

Para *ChirpStack Gateway Bridge*:

```
sudo apt install chirpstack-gateway-bridge

# start chirpstack-gateway-bridge
sudo systemctl start chirpstack-gateway-bridge

# start chirpstack-gateway-bridge on boot
sudo systemctl enable chirpstack-gateway-bridge
```

Para *ChirpStack Network Server*:

```
sudo apt install chirpstack-network-server

# start chirpstack-network-server
sudo systemctl start chirpstack-network-server

# start chirpstack-network-server on boot
sudo systemctl enable chirpstack-network-server
```

Para *ChirpStack Application Server*:

```
sudo apt install chirpstack-application-server

# start chirpstack-application-server
sudo systemctl start chirpstack-application-server

# start chirpstack-application-server on boot
sudo systemctl enable chirpstack-application-server
```

Após a instalação desses componentes, é necessário apenas alterar as senhas do bancos de dados nos arquivos chirpstack-network-server.toml e chirpstack-application-server.toml. Estas senhas foram estabelecidas na configuração do PostgreSQL

Após as instalações dos servidores Chirpstack, instalou-se o sistema de software Semtech Packet Forwarder para que o dispositivo operasse como um gateway LoRa.

O primeiro comando instala o Git e o pacote build-essential, que é um conjunto de ferramentas essenciais para compilar software no Debian. Essas ferramentas são necessárias para a compilação do código-fonte. Em seguida, clona-se os repositórios de código-fonte de toda a pilha de software do Packet Forwarder. Por fim, compila-se o código fonte através do comando *make -j4*:

```
sudo apt install git build-essential
git clone https://github.com/Lora-net/packet_forwarder.git
git clone https://github.com/Lora-net/lora_gateway.git
cd lora_gateway/
make -j4
cd ..
cd packet_forwarder/
```

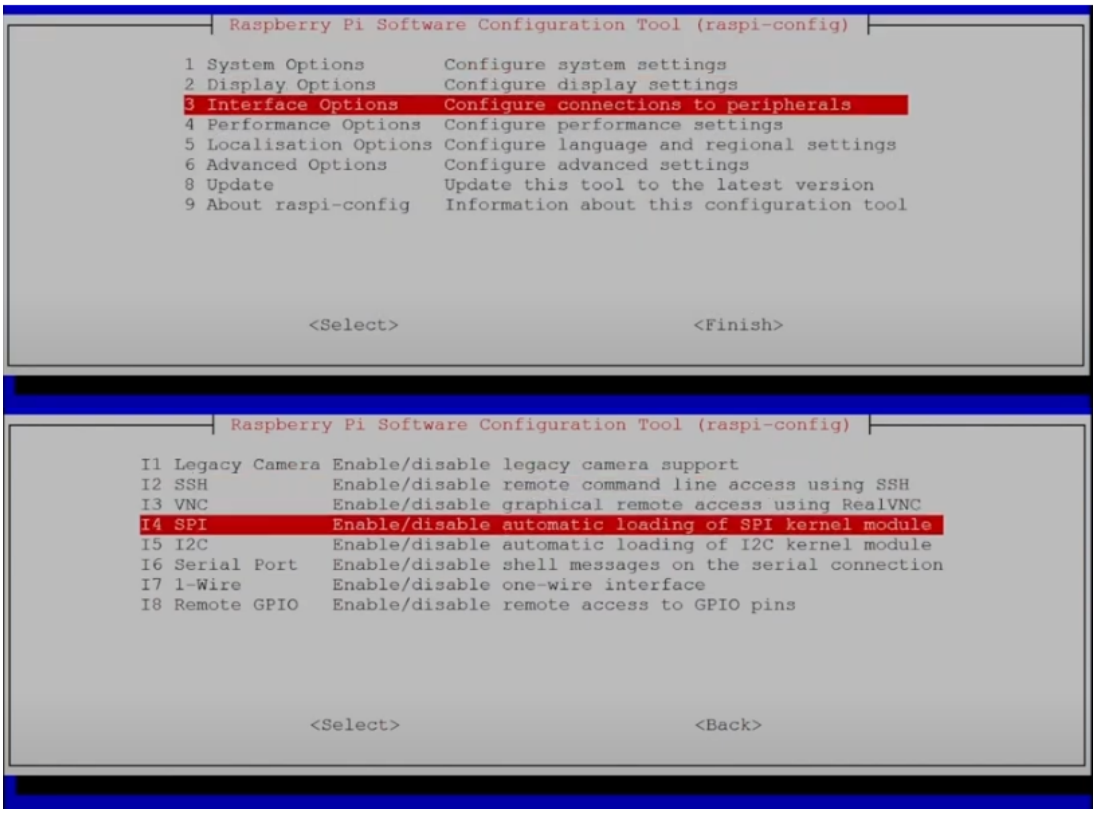
```
make -j4
```

Em seguida utiliza-se o comando *sudo raspi-config* envolve a ativação da interface SPI no Raspberry Pi, que é fundamental para estabelecer uma comunicação eficaz com dispositivos externos, como módulos LoRa. Essa configuração permite a transmissão bidirecional de dados e é crucial para o funcionamento adequado de um gateway LoRa, pois possibilita a troca de informações entre o Raspberry Pi e os dispositivos LoRa conectados.

```
sudo raspi-config
```

Com a interface aberta, basta seguir a cadeia de comandos para realizar a ativação: *Interface Options* → *SPI* → *Yes* → *Finish*.

Figura 42 – Habilitando a SPI do Raspberry PI.




Fonte: Própria.

Então, reinicializa-se o Raspberry PI para aplicar as configurações feitas, incluindo a habilitação da interface SPI:

```
sudo reboot
```

Após a reinicialização, é necessário fazer as configurações dos arquivos de configurações globais e locais chamados de global_conf.json e local_conf.json respectivamente, como descritos na seção de metodologia deste trabalho. Além disso, é necessário configurar o pino que reinicializa o modem, importante para o processo de inicialização do software. Para tal, é preciso acessar o arquivo config.txt localizado na pasta boot através do comando *sudo vim /boot/config.txt* e inserir a seguinte linha de comando ao final:

```
[all]
dtoverlay=spi0-cs,cs1_pin=4
```

É importante ressaltar que nesta etapa também configura-se os arquivos de configuração local e global, que cuidam da autenticação do gateway com o servidor. Estas configurações foram descritas na metodologia deste trabalho.

Uma vez concluidas todas estas etapas, basta inicializar o software Semtech Packet Forwarder e o dispositivo Raspberry PI já estará operando como um Gateway LoRa:

```
cd ~/packet_forwarder/lora_pkt_fwd/
./lora_pkt_fwd
```

Com o gateway em devido funcionamento, pôde-se prosseguir para a instalação dos softwares InfluxDB, Node-RED e Grafana,responsáveis pelo tratamento e visualização de dados.

Para instalar o InfluxDB em um Raspberry Pi, utilizamos o repositório oficial do InfluxDB, pois os desenvolvedores disponibilizaram pacotes específicos para diferentes sistemas operacionais compatíveis com o Raspberry Pi.

Primeiro, começamos obtendo a chave do repositório oficial e a adicionamos ao chaveiro local. Em seguida, adiciona-se o repositório correspondente ao sistema operacional utilizado e então atualiza-se a lista de pacotes disponíveis:

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo
   apt-key add -

echo "deb https://repos.influxdata.com/debian buster stable"
   | sudo tee /etc/apt/sources.list.d/influxdb.list

sudo apt update
```

Após a obtenção da chave e repositório, instalou-se o pacote InfluxDB através do comando *apt install*. Em seguida usa-se os comandos *systemctl unmask* e *systemctl enable* para garantir que o InfluxDB seja iniciado automaticamente durante o boot. Como o dispositivo já está em execução, é preciso inicializar manualmente na primeira vez. Para isso, utilizou-se o comando *systemctl start*:

```
sudo apt install influxdb

sudo systemctl unmask influxdb
sudo systemctl enable influxdb
sudo systemctl start influxdb
```

Uma vez iniciado, foi configurado o controle de acesso ao InfluxDB, criando um usuário administrador e definindo uma senha:

```
influx

CREATE USER admin WITH PASSWORD 'adminpassword' WITH ALL
   PRIVILEGES

exit
```

Por fim, edita-se o arquivo de configuração do InfluxDB para habilitar a autenticação do serviço. No arquivo de configuração, foram enseridos os comandos abaixo na seção "[HTTP]":

```
sudo nano /etc/influxdb/influxdb.conf

%commands to insert:
[HTTP]
auth-enabled = true
pprof-enabled = true
pprof-auth-enabled = true
ping-auth-enabled = true
```

Após esta etapa, a configuração do banco de dados está pronta e o InfluxDB pode começar a receber dados. Quanto a criação dos *databases* dentro do software, os nós

*InfluxDB out* na plataforma Node-RED são os responsáveis e fazem isso automaticamente, conforme foram configurados.

Dessa forma, prosseguiu-se para a instalação da plataforma Node-RED. Executou-se um script bash que atualiza o Node.js e instala o Node-RED no sistema. O script foi baixado e executado usando o comando "curl". Também foi utilizado o comando para instalar o pacote "node-red-contrib-influxdb". Este pacote é uma extensão do Node-RED que permite a integração fácil com o InfluxDB.

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/
   linux-installers/master/deb/update-nodejs-and-nodered)

npm install node-red-contrib-influxdb
```

Da mesma forma realizada na instalação do InfluxDB, os comandos *systemctl* foram utilizados para programar a iniciliazação automática e a inicialização manual do serviço Node-RED:

```
sudo systemctl enable nodered.service
sudo systemctl start nodered.service
```

Por fim, realizou-se a instalação da plataforma Grafana. Isso incluiu o download da chave de autenticação da Grafana e sua adição ao chaveiro do sistema usando os comandos *wget* e *apt-key add*. Essa medida é essencial para a verificação da autenticidade dos pacotes Grafana. Além disso, foi adicionado o repositório Grafana aos repositórios APT, possibilitando o download e a instalação de pacotes Grafana pelo sistema APT. A atualização da lista de pacotes disponíveis foi realizada com o comando *sudo apt update*, seguida da instalação do pacote Grafana por meio do comando *sudo apt install grafana*.

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-
   key add -

echo "deb https://packages.grafana.com/oss/deb stable main" |
    sudo tee -a /etc/apt/sources.list.d/grafana.list

sudo apt update
sudo apt install grafana
```

Finalmente, assim como para o InfluxDB e Node-RED, fez-se a configuração de inicialização ao *boot* e inicialização manual através dos mesmos comandos:

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Dessa forma, conclui-se a instalação de todos os recursos de softwares utilizados para o projeto.

# APÊNDICE B - *SCRIPTS* EMPREGADOS PARA O PROJETO

Nesta seção estão disponibilizados os códigos e scripts utilizados ao longo do projeto, referente às programações do Heltec WiFi LoRa 32(v2), utilizado como dispositivo final EndNode, bem como o código de decodificação dos pacotes de dados recebidos na plataforma Chirpstack e os scripts das funções de extração de variáveis na plataforma Node-RED.

Abaixo encontra-se o código para o Heltec WiFi LoRa 32(v2):

```
#include "LoRaWan_APP.h"
#include "DHT.h"
#include "MQ135.h"

#define DHT_PIN 13
#define LM393_PIN    36 // Pino onde o sensor de luminosidade esta conectado
#define MQ135_PIN    34 // Pino onde o sensor MQ135 esta conectado
#define SOIL_PIN     32 // Pino onde o sensor de umidade do solo esta conectado

DHT dht(DHT_PIN, DHT11);
MQ135 mq135(MQ135_PIN);

/* OTAA para*/
uint8_t devEui[] = { 0xae, 0x09, 0x4c, 0x81, 0x20, 0x3b, 0xd4, 0x22 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appKey[] = { 0x48, 0x37, 0xc9, 0x87, 0x14, 0xa1, 0x3c, 0x57, 0xcd, 0x3b, 0x5b, 0x94, 0x80, 0xe6, 0xad, 0xbf };

/* ABP para*/
uint8_t nwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d, 0x11, 0x18, 0x1e, 0x1e, 0xc7, 0xda,0x85 };
uint8_t appSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf, 0x55, 0xee, 0x4a, 0x77, 0x8d, 0x16, 0xef,0x67 };
uint32_t devAddr =  ( uint32_t )0x007e6ae1;

/*LoraWan channelsmask, default channels 0-7*/
uint16_t userChannelsMask[6]={ 0xFF00,0x0000,0x0000,0x0000,0x0000,0x0000 };

```

```
26 /*LoraWan region, select in arduino IDE tools*/
27 LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;
28
29 /*LoraWan Class, Class A and Class C are supported*/
30 DeviceClass_t  loraWanClass = CLASS_A;
31
32 /*the application data transmission duty cycle.  value in [ms].*/
33 uint32_t appTxDutyCycle = 15000;
34
35 /*OTAA or ABP*/
36 bool overTheAirActivation = true;
37
38 /*ADR enable*/
39 bool loraWanAdr = true;
40
41 /* Indicates if the node is sending confirmed or unconfirmed messages
       */
42 bool isTxConfirmed = true;
43
44 /* Application port */
45 uint8_t appPort = 2;
46 /*!
47 * Number of trials to transmit the frame, if the LoRaMAC layer did
      not
48 * receive an acknowledgment. The MAC performs a datarate adaptation,
49 * according to the LoRaWAN Specification V1.0.2, chapter 18.4,
      according
50 * to the following table:
51 *
52 * Transmission nb | Data Rate
53 * ----------------|-----------
54 * 1 (first)       | DR
55 * 2               | DR
56 * 3               | max(DR-1,0)
57 * 4               | max(DR-1,0)
58 * 5               | max(DR-2,0)
59 * 6               | max(DR-2,0)
60 * 7               | max(DR-3,0)
61 * 8               | max(DR-3,0)
62 *
63 * Note, that if NbTrials is set to 1 or 2, the MAC will not decrease
```

```
64 * the datarate, in case the LoRaMAC layer did not receive an
       acknowledgment
65 */
66 uint8_t confirmedNbTrials = 4;
67
68 /* Prepares the payload of the frame */
69 static void prepareTxFrame(uint8_t port)
70 {
71     float temperature = dht.readTemperature();
72     float humidity = dht.readHumidity();
73     int lightLevel = analogRead(LM393_PIN);
74     int airQuality = analogRead(MQ135_PIN);
75     int soilMoisture = analogRead(SOIL_PIN);
76
77     // Tratamento dos dados (mesmas equacoes do codigo do Arduino)
78     float ldrVoltage = lightLevel * (5.0 / 1023.0);
79     float airQualityVoltage = airQuality * (5.0 / 1023.0);
80     float soilVoltage = soilMoisture * (5.0 / 1023.0);
81
82     float ldrLux = 500.0 / ldrVoltage;
83     float mq135PPM = (airQualityVoltage / 5.0) * 1000.0;
84     float soilHumidity = map(soilMoisture, 4095, 0, 0, 100);
85     int soilHumidityInt = int(soilHumidity);
86
87     // Imprimir os valores tratados (opcional - para verificar se os
           calculos estao corretos)
88     Serial.print("Temperature: ");
89     Serial.print(temperature);
90     Serial.print(" °C, Humidity: ");
91     Serial.print(humidity);
92     Serial.print(" %, Light Level: ");
93     Serial.print(ldrLux);
94     Serial.print(" lux, Air Quality: ");
95     Serial.print(mq135PPM);
96     Serial.print(" ppm, Soil Moisture: ");
97     Serial.println(soilHumidityInt);
98
99     // Adicionar os dados tratados ao payload
100    appDataSize = 15;
101    appData[1] = int(temperature) & 0xFF;
102    appData[2] = int(humidity) & 0xFF;
```

```
103     appData[3] = (int(ldrLux) >> 8) & 0xFF; // Le os 8 bits mais
            significativos do valor de luminosidade
104     appData[4] = int(ldrLux) & 0xFF; // Le os 8 bits menos
            significativos do valor de luminosidade
105     appData[5] = int(mq135PPM); // Converte o valor de qualidade do
            ar para um únmero inteiro
106     appData[6] = soilHumidityInt & 0xFF; // Le o valor de umidade do
            solo e coloca no payload
107 }
108
109 //if true, next uplink will add MOTE_MAC_DEVICE_TIME_REQ
110
111
112 void setup() {
113   Serial.begin(115200);
114   Mcu.begin();
115   dht.begin();
116   deviceState = DEVICE_STATE_INIT;
117
118 }
119
120 void loop()
121 {
122
123   switch( deviceState )
124   {
125     case DEVICE_STATE_INIT:
126     {
127 #if(LORAWAN_DEVEUI_AUTO)
128       LoRaWAN.generateDeveuiByChipID();
129 #endif
130       LoRaWAN.init(loraWanClass,loraWanRegion);
131       break;
132     }
133     case DEVICE_STATE_JOIN:
134     {
135       LoRaWAN.join();
136       break;
137     }
138     case DEVICE_STATE_SEND:
139     {
```

```
140         prepareTxFrame( appPort );
141         LoRaWAN.send();
142         deviceState = DEVICE_STATE_CYCLE;
143         break;
144       }
145       case DEVICE_STATE_CYCLE:
146       {
147         // Schedule next packet transmission
148         txDutyCycleTime = appTxDutyCycle + randr( -APP_TX_DUTYCYCLE_RND
              , APP_TX_DUTYCYCLE_RND );
149         LoRaWAN.cycle(txDutyCycleTime);
150         deviceState = DEVICE_STATE_SLEEP;
151         break;
152       }
153       case DEVICE_STATE_SLEEP:
154       {
155         LoRaWAN.sleep(loraWanClass);
156         break;
157       }
158       default:
159       {
160         deviceState = DEVICE_STATE_INIT;
161         break;
162       }
163     }
164 }
```

O programa começa incluindo as bibliotecas necessárias e definindo os pinos para os sensores DHT, LM393, MQ135 e umidade do solo. Em seguida, são configurados parâmetros importantes para a comunicação LoRaWAN, como chaves de dispositivo, endereço, região e classe do dispositivo.

A função *prepareTxFrame* é definida para coletar dados dos sensores, processá-los e adicioná-los a um payload que é o pacote de dados a ser transmitido. O código principal está no loop, que é dividido em vários estados, incluindo inicialização, junção à rede LoRaWAN, preparação e envio de dados, agendamento de transmissões futuras e economia de energia.

Com o dispositivo final devidamente programado para envio de dados, é preciso decodificar o *payload* enviado pelo mesmo. Para isso, configura-se o código de decodificação na plataforma Chirpstack. Este código está disponibilizado abaixo:

```
1  // Decode decodes an array of bytes into an object.
2  //  - fPort contains the LoRaWAN fPort number
3  //  - bytes is an array of bytes, e.g. [225, 230, 255, 0]
4  //  - variables contains the device variables e.g. {"calibration":
       "3.5"} (both the key / value are of type string)
5  // The function must return an object, e.g. {"temperature": 22.5}
6  function Decode(fPort, bytes, variables) {
7      var decoded = {};
8
9      if (fPort === 2) {
10
11         // Decodificar os valores do payload
12         var temperature = bytes[1];
13         var humidity = bytes[2];
14         var lightLevel = (bytes[3] << 8) | bytes[4]; // Reconstruir o
               valor de 16 bits
15         var airQuality = bytes[5];
16         var soilHumidity = bytes[6];
17
18         // Adicionar os valores decodificados ao objeto "decoded"
19         decoded.temperature = temperature;
20         decoded.humidity = humidity;
21         decoded.lightLevel = lightLevel;
22         decoded.airQuality = airQuality;
23         decoded.soilHumidity = soilHumidity;
24     }
25
26     return decoded;
27 }
```

Este script é uma função JavaScript chamada Decode, que é usada para decodificar os dados recebidos do *payload*, com base no valor do fPort (número da porta) e nos bytes recebidos.

A função começa definindo um objeto vazio chamado decoded para armazenar os valores decodificados dos sensores. Ela verifica se o fPort é igual a 2, o que indica o tipo de mensagem que está sendo decodificada. Neste caso, os valores do *payload* são decodificados.

Os valores são então extraídos dos bytes recebidos, cada um referente a uma variável (temperatura, umidade, nível de luz, qualidade do ar e umidade do solo).Os valores decodi-

ficados são adicionados ao objeto *decoded* com nomes descritivos indicando suas medições, como temperatura, umidade, etc.Finalmente, o objeto decoded é retornado como resultado da função, que é o resultado ecnontrado no objeto JSON na seção *Device Data*.

O objeto JSON corretamente decodificado na plataforma Chirpstack é então processado na plataforma Node-RED para ser tratado e enviado ao banco de dados. Inicialmente, os dados fornecidos não estão no formato legível pelo InfluxDB. Portanto, foi escolhida a abordagem de extrair variável por variável para alocá-las a um *measurement* que leva o nome de sua medição.

Dessa forma, abaixo encontram-se os scripts dos *node functions* que extraem cada variável:

Função para obtenção do valor de temperatura:

```
1 // Codigo para o no Function - temperature
2
3 // Extrair o valor da medicao "temperature" da mensagem MQTT e
    converter para float
4 const temperatureValue = parseFloat(msg.payload.object.temperature);
5
6 // Definir a mensagem de saida contendo apenas o valor de "
    temperature"
7 msg.payload = temperatureValue;
8
9 return msg;
```

Função para obtenção do valor de umidade do ar:

```
1 //  Codigo para o no Function - humidity
2
3 // Extrair o valor da medicao "humidity" da mensagem MQTT e converter
     para float
4 const humidityValue = parseFloat(msg.payload.object.humidity);
5
6 // Definir a mensagem de saida contendo apenas o valor de "humidity"
7 msg.payload = humidityValue;
8
9 return msg;
```

Função para obtenção do valor de luminosidade:

```
1 // Codigo para o no Function - lightLevel
2
3 // Extrair o valor da medicao "lightLevel" da mensagem MQTT e
    converter para float
4 const lightLevelValue = parseFloat(msg.payload.object.lightLevel);
5
6 // Definir a mensagem de saida contendo apenas o valor de "lightLevel
    "
7 msg.payload = lightLevelValue;
8
9 return msg;
```

Função para obtenção do valor de umidade do solo:

```
1 // Codigo para o no Function - soilHumidity
2
3 // Extrair o valor da medicao "soilHumidity" da mensagem MQTT e
    converter para float
4 const soilHumidityValue = parseFloat(msg.payload.object.soilHumidity)
    ;
5
6 // Definir a mensagem de saida contendo apenas o valor de "
    soilHumidity"
7 msg.payload = soilHumidityValue;
8
9 return msg;
```

Função para obtenção do valor de qualidade do ar:

```
1 // Codigo para o no Function - airQuality
2 // Extrair o valor da medicao "airQuality" da mensagem MQTT e
    converter para float
3 const airQualityValue = parseFloat(msg.payload.object.airQuality);
4 // Definir a mensagem de saida contendo apenas o valor de "airQuality
    "
5 msg.payload = airQualityValue;
6 return msg;
```