UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Gilberto Alves de Oliveira Júnior

Implementação de Algoritmo de Adaptação de Taxa de Dados para Rede LoRaWAN

Florianópolis
2023

Gilberto Alves de Oliveira Júnior

Implementação de Algoritmo de Adaptação de Taxa de Dados para Rede LoRaWAN

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Carlos Montez, Dr.
Supervisor: Sandro Kirchner, Eng.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Oliveira Júnior , Gilberto Alves
Implementação de Algoritmo de Adaptação de Taxa de Dados para Rede LoRaWAN / Gilberto Alves Oliveira Júnior ; orientador, Carlos Barros Montez, coorientador, Sandro Kirchner , 2023.
68 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Internet das Coisas. 3. LoraWAN. 4. Taxa de dados adaptativa . 5. Gateway. I. Montez, Carlos Barros. II. Kirchner , Sandro . III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Gilberto Alves de Oliveira Júnior

Implementação de Algoritmo de Adaptação de Taxa de Dados para Rede LoRaWAN

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 11 de Julho de 2023.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Carlos Montez, Dr.
Orientador
UFSC/CTC/DAS

Sandro Kirchner, Eng.
Supervisor
Khomp

Prof. Rodrigo Lange, Dr.
Avaliador
Instituição IFRS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

AGRADECIMENTOS

Quero registrar a minha mais profunda e calorosa gratidão aos meus familiares que foram e sempre serão o meu porto seguro. Em especial a minha mãe Sandra e ao meu pai Gilberto, por todo incondicional amor, apoio e suporte em todos os momentos difíceis da minha vida. À minha madrinha Angela que tem uma simpatia infinita e sempre fez de tudo pra me deixar bem. Aos meus irmãos, Thiago e André que sempre incentivaram e acreditaram no potencial do caçulinha deles. Não sei como seria passar por tudo que passei nesses últimos anos sem vocês. Gostaria muito que minhas avós, Noemi e Bernadete, bem como meu padrinho Amauri, pudessem estar aqui para vibrar a minha conclusão no curso assim como vibraram o meu ingresso.

Agradeço ao meu orientador Carlos Montez, que me orientou ao longo do estágio e agora na conclusão do curso, me ajudando a me tornar um acadêmico melhor e profissional mais capacitado. Registro também minha gratidão a todo o departamento de automação e sistemas, que nunca deixaram de se preocupar com o bem estar dos alunos.

Deixo minha gratidão também ao Sandro Kirchner pelas oportunidades e por acreditar no meu potencial, e a todo o time de IoT da Khomp por sempre estarem contribuindo pro meu crescimento e aprendizado.

Ao longo dessa jornada fiz muitos amigos, e guardo muitas memórias boas. Obrigado pelo suporte, risadas e pelos sucos que tomamos e ainda tomaremos juntos.

Em resumo agradeço a todos que me ajudaram a enfrentar os desafios e dificuldades que passei, e participaram da construção da pessoa na qual eu estou me tornando.

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 02 de Julho de 2023.

Na condição de representante da Khomp na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

gov.br
Documento assinado digitalmente
SANDRO ALBERTON KIRCHNER
Data: 19/07/2023 18:25:40-0300
Verifique em https://validar.iti.gov.br

Sandro Kirchner, Eng.
Khomp

RESUMO

A Internet das Coisas está cada vez mais presente em nossas vidas, e inúmeros desafios a acompanham, como por exemplo ter um baixo consumo e longo alcance. Nesse contexto, surge o protocolo LoRaWAN, que por operar em frequência baixa consegue atingir longas distâncias, e ainda ser resistente a ruídos por causa de sua modulação. E nesse protocolo existem orientações e técnicas que visam otimizar o consumo dos dispositivos, como o algoritmo de taxa de dados adaptiva - o ADR. E buscando atender as demandas LoRaWAN, esse projeto visa implementar essa inteligência no network server dos gateways Khomp de telemetria, e posteriormente mostrar como é vantajoso ter o ADR implementado ativado em redes LoRaWAN. Esta monografia aborda os conceitos e explicações de LoRaWAN, e também as definições do protocolo. Aborda os fundamentos do algoritmo de ADR, bem como sua fundamentação matemática. Este é um projeto *full stack*, ou seja, foi implementado os algoritmos de ADR e também a interface de usuário, que permite gerenciar as taxas de dados e ativar e desativar o ADR.

Palavras-chave: Taxa de dados adaptativa. LoRaWAN. Internet das Coisas, *gateway*.

ABSTRACT

The Internet of Things is increasingly present in our lives, and numerous challenges accompany it such as having a low power consumption and long range. In this context, LoRaWAN protocol emerges, which by operating at low frequency can reach long distances, and distances, and still be resistant to noise because of its modulation. And in this protocol there are guidelines and techniques that aim to optimize the consumption of devices, such as the adaptive adaptive data rate algorithm - ADR. And seeking to meet the demands of LoRaWAN, this project aims to implement this intelligence in the network server of the Khomp Telemetry gateways, and then show how advantageous it is to have the ADR activated in LoRaWAN networks. in LoRaWAN networks. This monograph covers the concepts and explanations of LoRaWAN, and also the definitions of the protocol. It covers the fundamentals of the ADR algorithm, algorithm as well as its mathematical foundation. This is a full stack project, that is, it was implemented the ADR algorithms and also the user interface, which allows you to manage the data rates and to enable and disable ADR

Keywords: Adaptive Data Rate. LoRaWAN. Internet of Things. Gateway.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

P&D	*Pesquisa e Desenvolvimento*
IoT	*Internet das Coisas (do inglês Internet of Things)*
DR	*Taxa de Dados (do inglês Data Rate)*
ADR	*Taxa de Dados Adaptative (do inglês Adaptive Data Rate)*
LoRa	*Longo Alcance (do inglês Longe Range)*
WAN	*Rede de Longa Distância (do inglês Wide Area Network)*
QA	*Garantia de Qualidade (do inglês Quality Assurance)*
CSS	*Espectro de Dispersão por Chirp (do inglês Chirp Spread Spectrum*)
TOA	*Tempo no Ar (do inglês Time on Air*)
RSSI	*Indicador de Força do Sinal Recebido (do inglês Received signal strength indicator)*
SNR	*Relação sinal-ruído (do inglês Signal over Noise Ratio)*
SF	*Fator de Espalhamento (do inglês Spreading Factor)*
BW	*Largura de banda (do inglês Bandwidth)*
MAC	*Controle de Acesso ao Meio (do inglês Media Access Control)*
CID	*Identificador do Comando (do inglês Command Identifier)*
TDM	*Multiplexação por Divisão de Tempo (do inglês Time Division Multiplexing)*
FDM	*Multiplexação por divisão de frequência (do inglês Frequency Division Multiplexing)*
CHIRP	*Pulso de Radar de Alta Intensidade Comprimido Compressed (do inglês High Intensity Radar Pulse)*
EIRP	*Potência Isotrópica Radiada Equivalente (do inglês Effective Isotropic Radiated Power)*
OSI	*Sistemas Abertos de Interconexão(do inglês Open Systems Interconnection)*
RFU	*Reservado para Uso Futuro (do inglês Reserved for Future Use)*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A Internet das Coisas (*IoT - Internet of Things*) refere-se à interconexão de dispositivos físicos através da internet, permitindo a troca de dados e comunicação entre eles. Existem tecnologias e protocolos específicos, cada qual com as suas especificidades, vantagens ou desvantagens. É importante ressaltar que não existe uma solução perfeita, que se adéque a todos os cenários.

Dentre a diversidade de tecnologias sem fio disponíveis, destaca-se o LoRa (Long Range), uma tecnologia de comunicação sem fio desenvolvida especialmente para atender às necessidades das aplicações de Internet das Coisas (IoT). Essa tecnologia é notável por sua capacidade de alcançar longas distâncias e por apresentar um baixo consumo de energia, o que a torna especialmente adequada para cenários em que os dispositivos requerem comunicação em áreas extensas ou de difícil acesso.

LoRa é uma tecnologia de rádio proprietária e patenteada, relacionada a Camada Física do Modelo *OSI*, por outro lado *LoRaWAN (Long Range Wide Area Network)* é o protocolo que orquestra a infraestrutura de redes LoRa, permitindo que os dispositivos se conectem e recebam comandos. O LoRaWAN é projetado para ser escalável, seguro e eficiente em termos de consumo de energia, tornando-se uma opção viável para implantações de IoT em larga escala, como em *smart cities* e parques industriais.

Já o *ADR (Adaptive Data Rate)* é uma funcionalidade do LoRaWAN que permite otimizar a taxa de dados de transmissão entre os dispositivos e a rede. Com o ADR a rede pode ajustar automaticamente a taxa de dados dos dispositivos com base na qualidade do sinal e nas condições do canal de comunicação, resultando em economia de energia e aumento da vida útil da bateria, enquanto ainda mantêm uma comunicação confiável.

Dito isso, é evidente que o ADR é um mecanismo importante para as redes LoRaWAN. Visando um melhor desempenho de seus produtos, a empresa Khomp (referência nacional no ramo de Internet das Coisas) espera implementar em seus *gateway*s LoRa o Adaptive Data Rate, para se manter atualizada no mercado. Isso foi feito inserindo rotinas dentro da aplicação do *gateway*, seguindo as orientações propostas pelas padronizações LoRa Alliance, e pela Semtech, empresa responsável pela tecnologia LoRa.

A solução então foi estudar os conceitos e como funciona a dinâmica de troca de mensagens, abrir detalhadamente os pacotes LoRa e estabelecer a lógica para troca de taxa de dados (*data rate*). Além disso, é importante que o ADR possa ser ativado e desativado, em alto nível, na interface de usuário. A abordagem proposta foi oferecer ao usuário, através de página web, a possibilidade de configurar os parâmetros LoRaWAN.

Esse projeto faz com que os *gateway*s de telemetria produzidos e desenvolvidos pela Khomp estejam mais próximos de estarem de acordo com todas as funcionalidades LoRaWAN. Através deste projeto será possível mensurar ganhos físicos, como por exemplo a economia de pilhas dos dispositivos conectados ao *gateway*.

1.2 MOTIVAÇÃO

LoRa possui vários parâmetros que precisam ser ajustados, tais como potência de rádio e fator de espalhamento. O ajuste automático e adequado desses parâmetros é fundamental porque se materializam na taxa de transmissão obtida pelos dispositivos. Portanto, a proposta deste projeto foi implementação de leitura e envio de comandos *MAC*, necessários para troca da taxa de dados e implementação do ADR. Além disso, foi implementado também o algoritmo de Taxa Adaptativa de Dados, seguindo o algoritmo proposto pela SemTech e pela LoRa Alliance. Este algoritmo é usado por outras redes LoRaWAN, como por exemplo a TTN (The Things Network) (NETWORK, s.d.) e já foi amplamente testado. Por fim, foi feito também a máquina de estados contendo o fluxo do algoritmo e da troca de mensagens LoRaWAN.

1.3 OBJETIVOS

1.3.1 Objetivos Gerais

Estudar e implementar um algoritmo de ADR eficiente e escalável para otimizar a taxa de dados de transmissão em uma rede LoRaWAN, levando em consideração as condições dos sinais de comunicação e as características dos dispositivos IoT.

1.3.2 Objetivos Específicos

Para colaborar com a organização e aumentar as chances de atingir o objetivo geral do projeto, foram propostos os seguintes objetivos específicos:

- Desenvolver e implementar um algoritmo de ADR eficiente e escalável para otimizar a taxa de dados de transmissão em uma rede LoRaWAN, levando em consideração as condições do canal de comunicação.
- Realizar uma revisão e análise aprofundada do algoritmo de adaptação de taxa de dados existentes, levando em consideração as especificações e recomendações da *LoRa Alliance*.
- Implementar um modelo matemático e um algoritmo de adaptação de taxa de dados que leve em consideração a qualidade do canal, a interferência e outros parâmetros relevantes.

- Realizar e testes em laboratório para avaliar o desempenho e a eficácia do algoritmo de ADR em diferentes cenários de canal e carga de dispositivos.
- Avaliar e otimizar os parâmetros do algoritmo de ADR com base nos resultados dos testes, considerando métricas como taxa de sucesso de transmissão, latência e consumo de energia.
- Documentar todo o processo de implementação, incluindo a descrição do algoritmo, as metodologias de teste, os resultados obtidos e as conclusões.
- Realizar uma avaliação final do projeto, analisando o impacto do algoritmo de ADR na otimização do desempenho da rede LoRaWAN e identificando possíveis melhorias ou futuras pesquisas relacionadas.

Esses objetivos gerais e específicos fornecem uma estrutura abrangente para o projeto de implementação do algoritmo de ADR em uma rede LoRaWAN. Eles abordam desde a pesquisa inicial e a análise até a implementação prática, os testes e a validação do algoritmo, garantindo a eficiência e o desempenho adequados da rede.

1.4 ESTRUTURA DO TRABALHO

Esta monografia está dividida da seguinte forma.

O Capítulo 2 trata sobre o *gateway* Khomp de Telemetria, dispositivos finais LoRaWAN, um pouco da a história e estrutura da Khomp.

No Capítulo 3 é explicada toda teoria e importância do IoT, estrutura e funcionamento de redes LoRaWAN, estrutura dos pactes e algoritmo sugerido pela Semtech para o funcionamento do ADR.

O Capítulo 4 trata sobre o desenvolvimento, desde as tecnologias utilizadas até a escrita do código e execução de testes.

O Capítulo 5 é uma análise de melhorias relacionadas a LoRaWAN que podem ser feitas no *gateway* em um futuro próximo.

No Capítulo 6 são discutidos os resultados obtidos após a implementação do ADR.

No Capítulo 7 são feitas considerações finais e sugestão de trabalhos para o futuro.

2 EMPRESA E PRODUTO

Este capítulo apresenta brevemente sobre a história da empresa Khomp, fala sobre o *gateway* de telemetria, produto este que receberá a funcionalidade da taxa adaptativa de dados, e também traz alguns dos dispositivos mais importantes da linha LoRaWAN da empresa.

2.1 EMPRESA

A Khomp (KHOMP. . . , s.d.) é uma empresa brasileira, com matriz em Florianópolis, e que atua no mercado de tecnologia da informação e comunicação (TIC) desde 1996. Possuindo uma equipe técnica especialista em processamento digital de sinais, a empresa tem se destacado no mercado ao desenvolver projetos personalizados em hardware e software. Essa expertise tem permitido a criação de produtos de alto desempenho, atendendo às demandas mais exigentes. Com soluções inovadoras e avançadas, a empresa se consolida como uma das líderes no setor.

A empresa oferece uma ampla variedade de produtos, incluindo *gateways* de voz, interfaces de telefonia IP, centrais telefônicas, soluções para *contact center*, soluções completas para internet das coisas (IoT), entre outros. A Khomp é conhecida por sua alta qualidade de produtos e serviços, que são projetados para atender às necessidades de seus clientes.

Além disso, a instituição também tem um forte compromisso com a inovação e investe fortemente em pesquisa e desenvolvimento para criar soluções tecnológicas avançadas que atendam às demandas do mercado. A empresa tem uma equipe de profissionais altamente capacitados e qualificados que trabalham constantemente para melhorar e atualizar seus produtos.

2.2 PRODUTOS

2.2.1 *Gateway*

Gateway é o termo em inglês para “portão” ou “porta de entrada”. No contexto de IoT, ele funciona como uma *interface* de conexão entre tecnologias variadas, permitindo que equipamentos que utilizam diferentes padrões de comunicação conversem e troquem dados entre si.

O projeto aqui descrito, foi inteiramente desenvolvido para o *Gateway* Khomp de telemetria que é desenvolvido para integração de soluções de monitoramento IoT. Ele recebe e transmite dados coletados por até dois sensores conectados diretamente ao gabinete ou por meio de *Endpoints* com tecnologias sem fio, provendo escalabilidade à solução IoT. Os módulos para comunicação sem fio *gateway* do com Endpoints possi-

bilitam ampliar a quantidade de sensores de leitura e ampliar a área de monitoramento, com a instalação de sensores em locais distantes do *gateway*.

A arquitetura do gateway consiste em um Linux embarcado que faz a gerência dos drivers, módulos e programas. A aplicação principal, escrita em Python, é parte responsável por subir um servidor HTTP, e através desse servidor o usuário pode interagir com o gateway e realizar todas as configurações necessárias, também se conecta a *broker* MQTT, orquestrando outro meio de comunicação e integração com o usuário. Outro elemento importante se trata do Go-Connect, escrito em Go Lang, é responsável por toda gerência de conectividade, priorização de *interface* e responsável por checar o status da conexão e realizar eventuais trocas de *interface*.

Existem dois tipos de módulos sem fio, para se conectarem com suas respectivas tecnologias: LoRa e Zigbee. Em outras palavras *gateways* desenvolvidos pela Khomp são portais que conectam as redes IoT (LoRa ou ZigBee) à internet comum. Ou seja, os sensores (endpoints) realizam as leituras, enviam os dados para o *gateway* que por sua vez processa os dados e envia a um servidor, através de conexão cabeada ou tecnologia mobile 3g/4g.

A Khomp vende também o *gateway* de telemetria X *outdoor*, que é uma modificação puramente física da versão padrão, adaptada para ambientes externos. É um gateway com uma antena mais robusta e resistente às intempéries.

2.2.2 *Endpoints*

A Khomp possui solução LoRaWAN completa, desde *Gateway* até dispositivos finais, e tem sensores e dispositivos para as mais diversas áreas como indústria, saúde e agronegócio. O portfólio de dispositivos cresce anualmente, de acordo com as demandas comerciais ou por necessidade de atualizar o *hardware*.

Estes são os principais dispositivos LoRaWAN da empresa, e usados ao longo deste projeto para validação e testes do ADR:

- **Endpoint** de temperatura e umidade: Possui sensores internos de temperatura e umidade, e é extremamente versátio por permitir que seja conectado uma extensão (dando outras funções ao *Endpoint*) e até outros dois sensores em suas duas portas 1-wire.
- **Endpoint** contador de Pulsos: Usado medição de vasão de fluídos, como hidrômetros e gasômetros;
- **Endpoint** de vibração: Este é o mais recente lançamento no segmento LoRaWAN da Khomp. É um sensor de vibração para máquinas e motores elétricos.

2.3 FLUXO DE OPERAÇÃO DA IOT DA KHOMP

A Khomp possui a Pesquisa e Desenvolvimento para a área de IoT. Os clientes da área, em sua maioria, são empresas que desenvolvem soluções utilizando os produtos desenvolvidos, fabricados e vendidos pela Khomp. Os clientes prestam serviços, oferecendo telemetria em tempo real, gráficos e plataformas de monitoramneto. Estes tipos de clientes são também conhecidos como integradores, porque eles integram os produtos da Khomp com processamentos na nuvem, *dashboard* e criam o seu negócio. O termo integrador será utilizado nessa monografia em outros momentos para expressar essa categoria de clientes, que são os mais expressivos da empresa.

Para ilustrar um exemplo, a seguir é descrito um cenário hipotético. A Khomp vende um *Gateway* e 3 *Endpoint*'s para o integrador de soluções Agrícolas AgroPop (nome fictício). Por sua vez, o AgroPoP está aplicando a solução em um Aviário, e vai utilizar um dispositivo *Endpoint* para coletar a temperatura e umidade do mesmo, monitorando assim a qualidade de vida das galinhas e dos ovos que elas produzem. Outro *Endpoint*, com um sensor magnético foi acoplado para saber quantas vezes a porta do aviário foi aberta. Outro *Endpoint* foi usado pra realizar a medição da temperatura da água que as galinhas tomam. Os *Endpoint* vão enviar seus respectivos dados sensoriais ao *gateway*, que por sua vez vai enviar via MQTT para os serviços em Cloud desenvolvido pela Agropop, que por sua vez vai entregar Dashboards de monitoramento à gerência do aviário.

Além disso, a empresa AgroPoP está solicitando um dispositivo com uma funcionalidade nova. Nesse caso, a Khomp faz uma pesquisa de mercado para averiguar a viabilidade desse novo dispositivo. Caso seja viável, o time de *endpoints* projeta e desenvolve esse produto, e o time de *gateway*, sabendo das especificações desse novo sensor, realizará a integração para que funcione perfeitamente os dois produtos em conjunto.

3 REVISÃO DA LITERATURA

Este capítulo aborda todos os conhecimentos necessários para a execução e entendimento deste projeto.

3.1 INTERNET DAS COISAS

A internet se popularizou na década de 90 (MUSEUM, s.d.). Naquela época, o acesso era restrito a computadores e conexão cabeada, e as condições da época (velocidade e estabilidade) dificultavam o acesso. Com o passar dos anos, a internet foi se desenvolvendo e o seu uso ficando cada vez mais fácil e popular. A partir da evolução das tecnologias sem fio em paralelo com o aumento do processamento computacional, a internet passou a estar presente em objetos que antes tinham apenas uma função específica. Por exemplo, os celulares passaram a ter acesso a internet, trazendo conforto de resolver burocracias em questão de cliques ou o lazer de navegar em uma rede social - tudo isso em um dispositivo que cabe no nosso bolso. Posteriormente, os televisores passaram a se conectar à internet, facilitando assim a forma como assistimos séries, filmes e noticiários.

Dando continuidade para a evolução de trazer a internet para os objetos – a IoT, sigla em inglês para Internet das Coisas – é, como visto em (BHAT, 2018), uma rede de dispositivos que são incorporados com sensores ou atuadores, e possuem conectividade à Internet. Esses dispositivos coletam e trocam dados, permitindo que haja interação entre si e com o ambiente em que está inserido em tempo real.

Como não existe uma definição universal sobre Internet das Coisas, Karen Rose, Scott Eldridge e Lyman Chapin descrevem o termo Internet of Things de uma forma mais abrangente da seguinte maneira: "The term Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. There is, however, no single, universal definition"(ROSE; ELDRIDGE; CHAPIN, 2015, p. 5).

Uma coisa é certa, independente da definição adotada, a Internet das Coisas é uma tecnologia promissora e que veio pra ficar, e se estima que em 2030 tenha mais de 30 bilhões de sensores e atuadores conectados a Internet das Coisas (INSIGHTS, s.d.). E diante desse cenário, onde uma tecnologia muito nova, e que vem crescendo cada vez mais, inúmeros desafios vem com ela. Desde atender a demandas comerciais, até melhorias em questões de eficiência ou necessitar de menos manutenções presenciais, e principalmente encontrar descarte adequado para o lixo gerado por dispositivos IoT, e também torná-los mais econômicos do ponto de vista energético. Estima-se que até 78 milhões de pilhas e baterias que alimentam dispositivos IoT podem ser descartadas diariamente em 2025 (CORDIS, 2021).

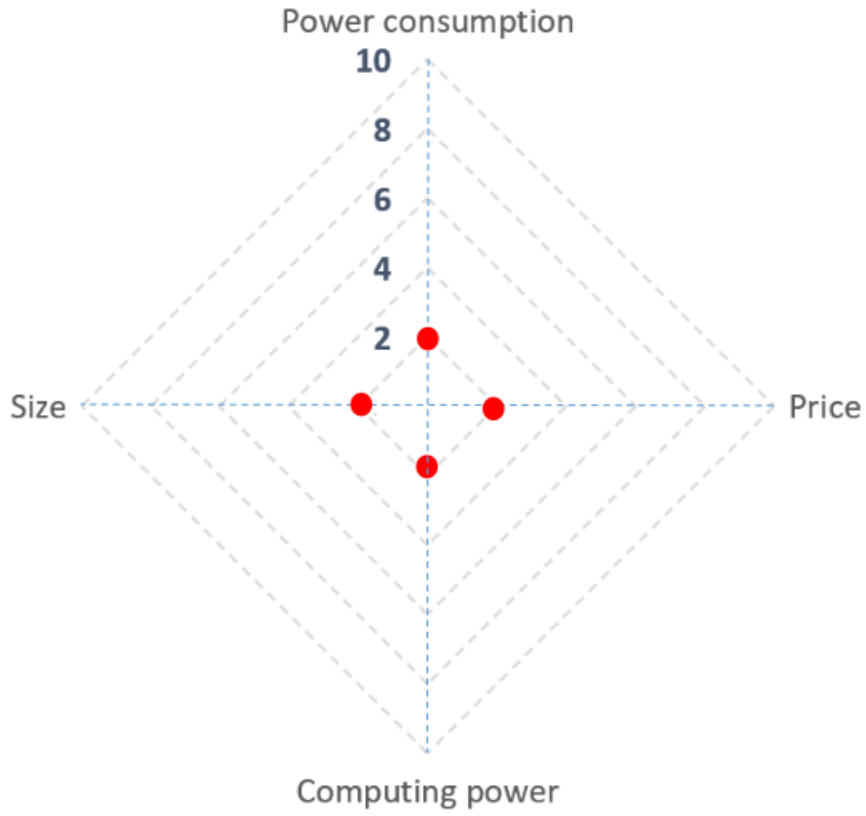
A Internet das Coisas está transformando a maneira como vivemos e trabalhamos, criando novas oportunidades para empresas e cidades (ATZORI; IERA; MORABITO, 2010). Por exemplo, na agricultura, os sensores de IoT podem ajudar os agricultores a otimizar o crescimento das espécies cultivadas e aumentar os rendimentos das safras, fornecendo dados em tempo real sobre a umidade do solo, temperatura e outros fatores ambientais. Nas manufaturas, a IoT pode ajudar as fábricas a reduzir o tempo de inatividade e melhorar a eficiência, permitindo a manutenção preditiva e o monitoramento remoto dos equipamentos.

Essa nova e expansiva etapa da internet também teve um impacto significativo na saúde. Como exemplo, pode-se citar: dispositivos capazes de monitorar a temperatura e umidade de unidades hospitalares; dispositivos vestíveis e sensores que auxiliam no monitoramento de sinais vitais (GUBBI *et al.*, 2013) dos pacientes e emitem alertas aos profissionais médicos sobre possíveis problemas de saúde antes o quadro se agrave; dispositivos domésticos inteligentes, como termostatos e sistemas de segurança, que tornam nossas casas mais seguras, confortáveis e com maior eficiência energética.

Para que ocorra a internet dentro das coisas, é necessário que um dispositivo com a capacidade sensorial ou atuadora possua um módulo de comunicação sem fio. Esse módulo pode ser wi-fi ou de telefonia móvel (como 3g ou 4g) (GUBBI *et al.*, 2013), e nesse caso o dispositivo é capaz de se comunicar diretamente com a nuvem, e são conhecidos como dipositivos *stand-alone* (autônomos ou independente). Existe também o cenário onde esse módulo é de uma outra tecnologia sem fio, mais barata, com um volume menor de dados, própria para IoT, com baixo consumo de energia e que alcança distâncias maiores. Nesse caso, o dispositivo se comunica com um equipamento intermediário, chamado de *gateway*, e ele é responsável por enviar os dados à nuvem. O *gateway* é o foco de trabalho do presente projeto.

Os sistemas eletrônicos podem ser qualificados de acordo com seu consumo, poder de computação, tamanho e preço. No caso específico dos dispositivos para Internet das Coisas, eles possuem baixo consumo e poder de processamento, tamanho pequeno e preços baixos (MONTAGNY, 2022). A Figura 1 ilustra como devem ser os dispositivos IoT.

Figura 1 – Características dispositivos IoT.



Fonte: LoRa - LoRaWAN and Internet of Things for beginners

3.2 FREQUÊNCIAS DE RÁDIO ISM

ISM (Industrial, Scientific, and Medical) refere-se a uma faixa de frequência de rádio licenciada internacionalmente e reservada para comunicações de baixa potência, bem como para aplicações não críticas que não interferem com serviços licenciados (TELECO, s.d.).

As faixas de frequência ISM podem variar em diferentes regiões do mundo, mas as mais comumente utilizadas são 315MHz, 433 MHz, 868MHz, 915 MHz, 2.4 GHz. Essas faixas de frequência são abertas e não requerem licenciamento específico para seu uso, o que permite que fabricantes e desenvolvedores utilizem essas frequências para aplicações de comunicação sem fio, como redes de sensores, dispositivos IoT, comunicação de curto alcance e outros sistemas de rádio de baixa potência.

A vantagem de usar a faixa de frequência ISM é que ela fornece um espaço disponível para comunicações sem fio sem a necessidade de obter licenças individuais, tornando-a acessível e de baixo custo para diversas aplicações. No entanto, uma das desvantagens é que como as faixas de frequência ISM são abertas e amplamente

utilizadas, pode haver interferência de outros dispositivos e redes operando na mesma faixa, o que pode afetar o desempenho e a confiabilidade das comunicações.

O Brasil opera utilizando a faixa 915MHz a 928MHz, normatizada pela Anatel (ATO. . . , 2017), mesma faixa referenciada para a Australia, e por isso a nomenclatura é AU915-928. E é por isso, neste projeto e monografia, várias vezes será referenciado os parâmetros regionais de AU915-928. Em algumas bibliografias ou provedores de serviço LoRaWAN pode-se encontrar LA915-928 (de *Latin America*), mas em questões físicas de modulação LA e AU são a mesma coisa, apenas alguns parâmetros da camada *MAC* são diferentes.

3.3 LORA

LoRa (Long Range) é uma tecnologia de comunicação sem fio de longo alcance, projetada especificamente para aplicações de Internet das Coisas (IoT) (RAZA *et al.*, 2017). Ela foi desenvolvida pela 2015, e posteriormente comprada pela *Semtech*. Oferece uma solução de conectividade de baixa potência, baixo custo e longo alcance, adequada para uma ampla gama de aplicações de IoT (MONTAGNY, 2022).

As frequências LoRa são FDM (Frequemcy Division Multiplexing), que é a banda (ex. 915MHz) (MONTAGNY, 2022) particionada em outros canais, permitindo mais opções. Além disso, a comunicação possuí TDM (Time division Multiplexing) (MONTAGNY, 2022), o que significa que os dispositivos usam intervalos de tempo (ou eventos) para transmitir dados, deixando assim o canal livre por mais tempo. Ou seja, não é uma comunicação contínua (RAO; PAPANDREOU-SUPPAPPOLA, 2007).

O LoRa utiliza uma técnica de modulação chamada Chirp Spread Spectrum (CSS) (MONTAGNY, 2022), que permite a transmissão de dados em uma ampla faixa de frequência, com baixa taxa de dados e baixo consumo de energia. Pelo fato de ser uma frequência baixa, o sinal LoRa atinge grandes distâncias, e graças a essa modulação conhecida como CSS o sinal se torna resistente a ruídos. (REYNDERS; MEERT; POLLIN, 2016).

Para se ter uma ideia da distância que a transmissão LoRa pode chegar, em 2020 um recorde foi batido (que antes era de 766km). Um experimento realizado com um balão com um sensor, voando a uma altitude de 38km, disparou um pacote LoRaWAN que foi recebido por um gateway a 832km de distância do balão (LORAWAN. . . , s.d.).

3.3.1 *Chirp Spread Spectrum*

Chirp Spread Spectrum (CSS) é uma técnica de modulação utilizada na tecnologia LoRa (*Long Range*) para permitir a comunicação sem fio de longo alcance e baixo consumo de energia (REYNDERS; MEERT; POLLIN, 2016). Essa técnica é

baseada na propagação de sinais que se espalham em frequência ao longo do tempo, criando um sinal que varia linearmente em frequência (MONTAGNY, 2022), conhecida como *CHIRP*, que pode ser ascendente (do menor para o maior valor de frequência) ou descendente (do maior para o menor valor de frequência). Essa variação linear de frequência é o que torna um sinal robusto e resistente e é também o que fazem os bits.

Uma das vantagens do CSS é que ele oferece uma maior imunidade a interferências, especialmente em ambientes com ruídos e obstáculos físicos. A técnica de *Chirp Spread Spectrum* permite que os sinais sejam recebidos mesmo em presença de ruídos e interferências de outros dispositivos, como dispositivos Wi-Fi, Bluetooth ou outros sistemas de rádio. Isso possibilita uma comunicação mais confiável e estável em ambientes com interferências eletromagnéticas. Além disso, o CSS possui uma boa eficiência energética, permitindo que os dispositivos LoRa transmitam dados utilizando níveis de potência muito baixos, o que resulta em uma maior duração das baterias. Essa característica é especialmente importante em aplicações de Internet das Coisas (IoT), onde os dispositivos geralmente são alimentados por baterias de longa duração ou fontes de energia limitadas.

3.3.2 Fator de Espalhamento e Largura de Banda

Em LoRa, o *Spreading Factor* (fator de espalhamento) é parâmetro que determina a taxa de transmissão e a robustez da comunicação (MONTAGNY, 2022). Ele é um dos principais elementos configuráveis na tecnologia LoRa e desempenha um papel fundamental no desempenho e alcance da comunicação.

Um *CHIRP*, ou símbolo, é o ciclo da modulação pra descrever N *bits*, onde N é o número do SF. Ou seja, quanto maior o *Spreading Factor* maior é a quantidade de informações que um CHIRP carrega, e maior é o tempo pra modular esse chirp. O problema é que o aumento de tempo é exponencial, como se pode constatar na equação a seguir.

$$T_{chirp} = \frac{2^{SF}}{Bandwidth} \tag{1}$$

Em um cenário onde a largura de banda (*Bandwidth*) é 125KHz, temos a seguinte relação:

Tabela 1 – Relação *Spreading Factor* e tempo período de um Chirp

Spreading Factor	Período de um chirp(ms)	Taxa de Bits (bits/s)
SF7	1.024	5470
SF8	2.048	3125
SF9	4.096	1760
SF10	8.192	980
SF11	16.384	440
SF12	32.768	250

Fonte: *LoRa - LoRaWAN and Internet of Things for beginners*

A partir da Tabela, percebe-se que quanto maior o *Spreading Factor*, menor é a velocidade pra se modular uma mensagem. Como consequência disso, o rádio precisa ficar ligado mais tempo, consumindo mais energia. O aumento do Spreading factor infere também no aumento de sensibilidade do receptor, pois como um símbolo contém mais bits, o receptor consegue demodular o sinal mesmo em condições onde o ruído está alto (MONTAGNY, 2022);

A escolha de um Fator de Espalhamento depende das necessidades específicas da aplicação. Em cenários em que a longa distância de comunicação é essencial, como em redes de IoT em áreas rurais ou em smart cities, é comum utilizar *Spreading Factor*s mais altos para garantir um alcance maior. Por outro lado, em aplicações que requerem uma alta taxa de transferência de dados, como monitoramento em tempo real, *Spreading Factor*s mais baixos podem ser preferíveis.

A combinação entre um um SF com um BW forma o que conhecemos em LoRa como Data Rate, e essa é a principal variável controlada em um algoritmo de taxa de dados adaptativa. O funcionamento desse algoritmo será detalhado e explicado na Seção 3.5.

O *gateway* possui oito canais de comunicação, e cada canal é capaz de ouvir em 7 taxa de dados diferentes simultaneamente (DR0 ao DR 6). Por isso é também importante que os dispositivos finais sejam configurados para taxas diferentes das padronizadas pela fábrica, pois assim diminuem as chances de encontrarem o canal ocupado para aquela frequência.

A tabela 2 mostra a relação entre fator de espalhamento, largura de banda e a taxa de bits, além de mostrar também, que existem diferenças entre quando alterada a direção das mensagens, onde *uplink* são so dados que o dispositivo final envia ao *gateway*, enquanto *downlink* são mensagens de comandos que o *gateway* envia ao dispositivo final. As definições e explicações de *Uplink* e *Downlink* estão presentes nas seções 3.4.3 e 3.4.4 respectivamente.

Tabela 2 – Relação entre *Data Rates*, *Spreading Factors* e *Bandwidth*

DR	SF	BW	Direção	Taxa de Bits (bits/s)
0	12	125kHz	*uplink*	250
1	11	125kHz	*uplink*	440
2	10	125kHz	*uplink*	980
3	09	125kHz	*uplink*	1760
4	08	125kHz	*uplink*	3125
5	07	125kHz	*uplink*	5470
6	08	500kHz	*uplink*	12500
7	RFU	RFU	RFU	RFU
8	12	500kHz	*downlink*	980
9	11	500kHz	*downlink*	1760
10	10	500kHz	*downlink*	3900
11	9	500kHz	*downlink*	7000
12	8	500kHz	*downlink*	12500
13	7	500kHz	*downlink*	21900
14 e 15	RFU	RFU	RFU	RFU

Fonte: *Regional Parameters* AU915-928 *Semtech* (2020)

A Tabela 2 mostra a relação entre o respectivo DR e sua combinação de SF e BW. O RFU presente em duas linhas, significa *Reserved for Future Use*, ou reservado para uso futuro. Ou seja, são taxas de dados ainda não implementadas e nem definidas pelo protocolo. O algoritmo aplicado no projeto descrito nesta monografia controla apenas o DR de *Uplink*.

O receptor da mensagem é capaz de identificar qual é a taxa de dados (e consequentemente qual o SF e o BW) da mensagem recebida, sem a necessidade de ser informado em algum *byte* da mensagem. Isso se dá porque a taxa de dados é expressada de forma física, é como aquela mensagem foi modulada, e o receptor tem a capacidade demodular e identificar qual foi a taxa de dados da mensagem recebida.

3.3.3 Potência do Rádio

A potência de rádio TX (transmissão) refere-se à potência de sinal utilizada para transmitir dados em uma rede de comunicação sem fio, como no caso das redes LoRa. Essa potência é medida em unidade como dBm.

Em redes LoRaWAN, a potência de rádio TX é um parâmetro configurável que determina a força do sinal transmitido pelos dispositivos finais. Uma potência de rádio mais alta resulta em um sinal mais forte, o que pode levar a um alcance de transmissão maior. Por outro lado, uma potência de rádio mais baixa resulta em um sinal mais fraco e um alcance de transmissão reduzido e também um menor consumo de energia. É possível ajustar a potência de rádio dos dispositivos finais (reduzir ou aumentar) conforme as variáveis de qualidade de sinal, com o objetivo de poupar energia ou de

não gerar ruído desnecessário. A forma como isso é feito será explicado também na Seção 3.5, visto quê a potência do rádio é também controlada pelos algoritmos de ADR.

Cada região possui as suas próprias normas e limitações sobre a potência equivalente isotropicamente irradiada (EIRP) (LORA ALLIANCE, 2018). Na região latino Americana, que possui os mesmos parâmetros regionais que a Australia (AU915-918), e tem definido como 30dBm como máximo para EIRP - sendo normalmente essa é a potência padrão nos dispositivos finais. A Tabela 3 mostra a relação entre o valor enviado de *TXpower* e a potência aplicada pelo dispositivo final.

Tabela 3 – Tabela da potência do rádio para a região AU915-928

TXpower	Configuração EIRP
0	$EIRP_{max}$
1:14	$EIRP_{max} - 2 * TX_{power}$

Fonte: Parametros Regionais LoRaWAN (*Semtech*)

O *TXpower* é o valor enviado ao dispositivo. Portanto, para diminuir a potência de rádio efetiva, aumenta-se o *TXpower*. Na Seção 3.4.6 é explicado melhor como se diminui ou aumenta o rádio por meio de comandos enviados aos dispositivos finais.

3.3.4 Sinal

O sinal pode ser mensurado por duas variáveis, RSSI e SNR. Esses são dois parâmetros importantes na comunicação sem fio, incluindo redes LoRaWAN. Eles são utilizados para avaliar a qualidade do sinal de rádio. São variáveis de sensibilidade do *gateway*, ou seja, é a força (ou a força e o ruído) que um determinado *gateway* recebeu o pacote em seu rádio. Se um mesmo pacote for recebido por dois ou mais *gateways*, eles provavelmente terão valores diferentes de RSSI e SNR. Em redes que possuem em sua topologia múltiplos *gateways*, uma mensagem pode ser recebida também em diferentes pontos, e quando isso acontece a rede decide que o pacote com os melhores valores de RSSI e SNR é a mensagem válida.

Tabela 4 – Representação entre qualidade e sinal medido

Qualidade	RSSI (dBm)	SNR (DB)
Excelente	-45	13
Boa	-84	2
Impraticável	-110	-11

Fonte: Arquivo Pessoal

Qualidade excelente significa que o dispositivo está bem perto, e é altamente sugerido realizar ajustes na taxa de dados e na potência de rádio. Qualidade boa signi-

fica que poucos ajustes são necessários. Finalmente, qualidade impraticável significa que a comunicação está comprometida, e um dos dois lados pode deixar de receber pacotes.

Como RSSI e SNR são medidas independentes, ou seja não existe uma proporcionalidade, é comum existir casos onde um dos valores está bom e o outro está ruim.

3.4 LORAWAN

O LoRaWAN é uma arquitetura de rede de estrela em que os dispositivos finais, também conhecidos como nós ou sensores, se comunicam com os gateways LoRaWAN que estão conectados a uma infraestrutura de rede de back-end (LORA. . . , s.d.). Essa infraestrutura de back-end inclui servidores de rede LoRaWAN, de aplicativos e serviços em nuvem que permitem a coleta, processamento e análise de dados provenientes dos dispositivos IoT.

Uma das principais características do LoRaWAN é seu foco em eficiência energética, permitindo que os dispositivos IoT operem com baterias de longa duração. Tendo em vista esses objetivos, a forma como ocorre a troca de dados, o tamanho dos payloads, desligamento do rádio - tudo isso é gerenciado pelo protocolo de modo a atingi-los.

O protocolo é dividido em três camadas, a camada física, *MAC* e de aplicação (MONTAGNY, 2022). A camada física diz respeito a modulação para transferência de dados. A camada *MAC*, responsável pela gerência, autenticação, criptografia e acessos. É nela que o projeto descrito nesta monografia foi executado. E por último, a camada de aplicação é onde os dados são interpretados e processados.

3.4.1 Infraestrutura Padrão LoRaWAN

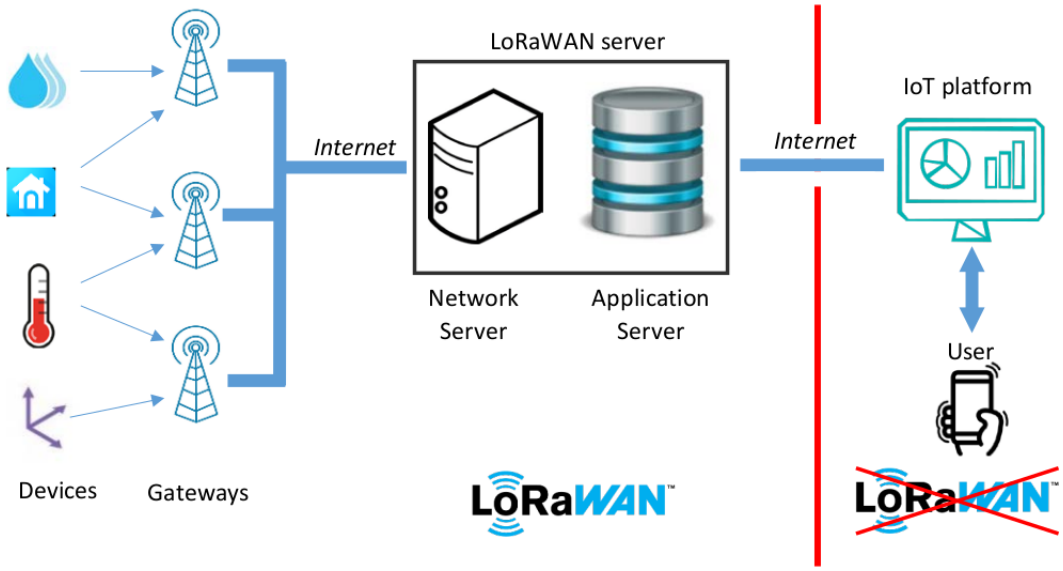
A infraestrutura LoRaWAN é composta por diferentes elementos que trabalham em conjunto para permitir a comunicação eficiente e escalável entre os dispositivos finais e os servidores de aplicação (MONTAGNY, 2022). Esses elementos incluem:

- **Dispositivos Finais**: São os sensores de baixo consumo de energia e baixo custo que são implantados e enviam dados via LoRa.
- **Gateways**: Os gateways são responsáveis por receber as transmissões dos dispositivos finais e encaminhá-las para os servidores de rede. Eles atuam como pontes entre os dispositivos finais e a infraestrutura de rede.
- **Servidores de Rede**: Os servidores de rede LoRaWAN são responsáveis por receber as mensagens dos gateways, decodificar os pacotes, encaminhar os dados para os servidores de aplicação e coordenar a operação da rede. Esses

servidores gerenciam a autenticação dos dispositivos finais, o roteamento dos dados e a manutenção do estado da rede.

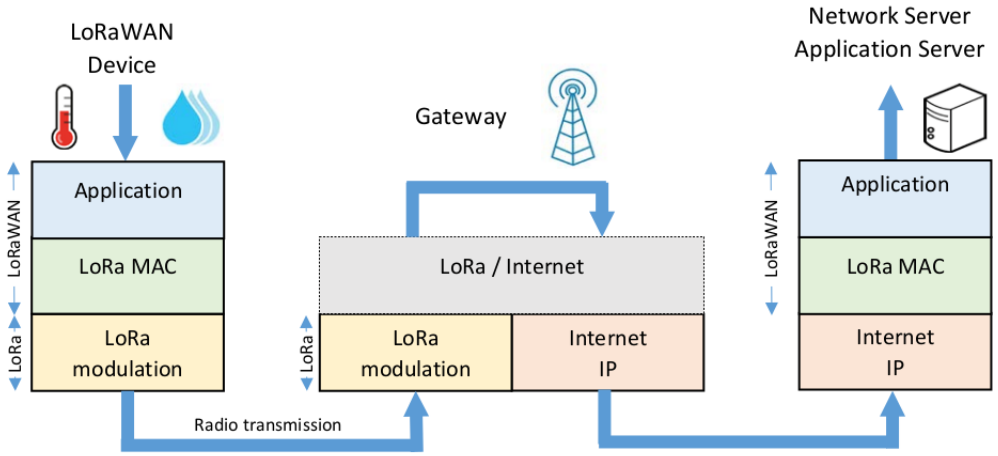
- **Servidores de Aplicação**: Os servidores de aplicação são responsáveis por receber os dados dos dispositivos finais enviados pelos servidores de rede, processar esses dados de acordo com as necessidades específicas da aplicação e fornecer respostas ou comandos de volta aos dispositivos finais, se necessário.

Figura 2 – Topologia padrão LoRaWAN



Fonte: LoRa - LoRaWAN and Internet of Things for beginners

Figura 3 – Fluxo de dados e transições LoRaWAN



Fonte: LoRa - LoRaWAN and Internet of Things for beginners

3.4.2 Network Server

O *network server* atua como um intermediário entre os dispositivos finais e os servidores de aplicação. Ele é responsável por receber e processar os *frames* de dados enviados pelos dispositivos finais, aplicar políticas de autenticação, encriptação e encaminhar os dados relevantes para os servidores de aplicação apropriados (MONTAGNY, 2022). São eles os responsáveis por realizarem a gerência de *gateways* e dispositivos finais. Em uma rede podem ter múltiplos *gateways* pertencentes a sua topologia, com o objetivo de garantir uma maior cobertura ou redundância de recebimento de sinal.

As principais funções de um *network server* LoRaWAN incluem:

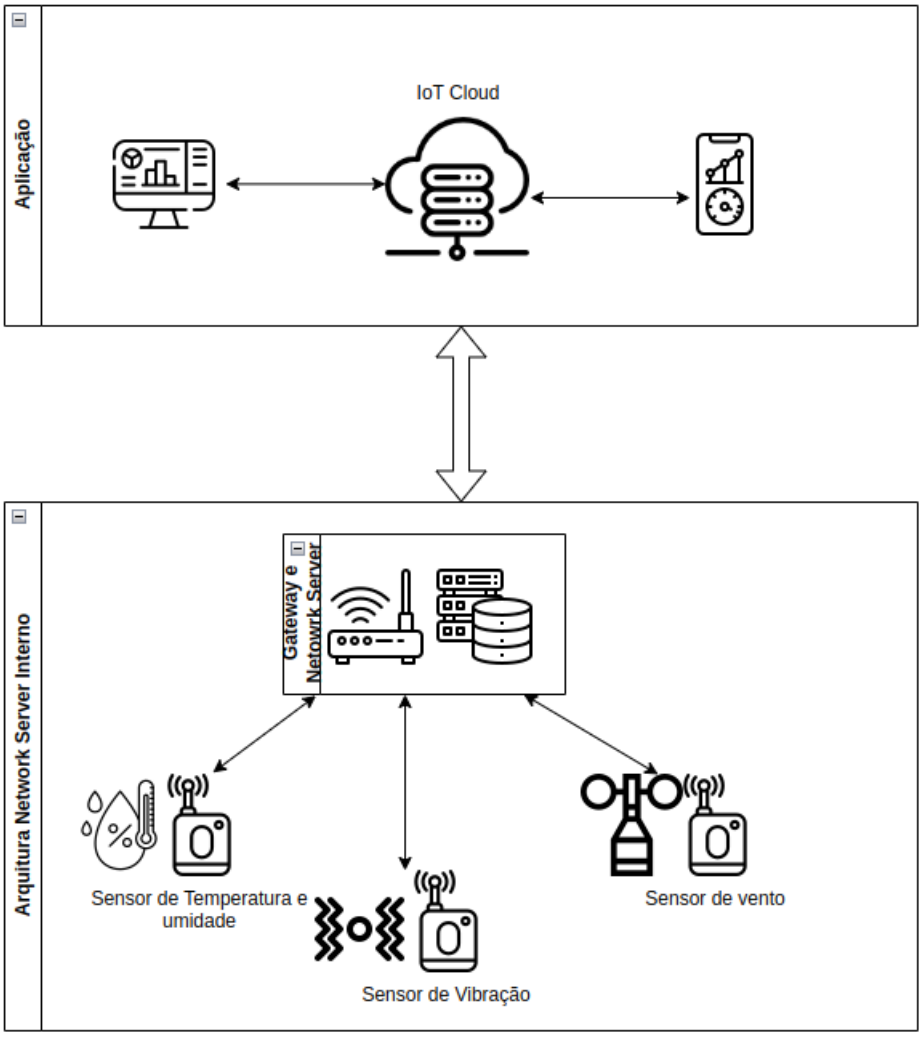
- **Autenticação e autorização**: O *network server* autentica os dispositivos finais que desejam ingressar na rede LoRaWAN e verifica se eles estão autorizados a se conectar. Isso é feito por meio de troca de chaves de segurança e autenticação de identidade.
- **Gerenciamento de sessão**: O *network server* gerencia as sessões de comunicação entre os dispositivos finais e a rede LoRaWAN. Isso envolve o estabelecimento de parâmetros de comunicação, como taxa de dados (Data Rate), potência de transmissão (Tx Power) e frequência de operação.
- **Roteamento e encaminhamento**: O *network server* encaminha os dados recebidos dos dispositivos finais para os servidores de aplicação apropriados com base nas configurações e políticas definidas. Ele também roteia os comandos e mensagens de controle para os dispositivos finais correspondentes.
- **Gerenciamento de *downlink***: O *network server* gerencia os *downlink*s, ou seja, as mensagens enviadas da rede LoRaWAN para os dispositivos finais.
- Implementação de políticas de rede: O *network server* aplica políticas de rede definidas.
- Monitoramento e diagnóstico: O *network server* monitora o desempenho da rede LoRaWAN, registra informações de atividade e fornece diagnósticos para otimização e solução de problemas.

Existem várias soluções de *network server* disponíveis no mercado, tanto comerciais quanto de código aberto, que podem ser usadas para implantar e gerenciar redes LoRaWAN. Alguns exemplos populares incluem *ChirpStack* (CHIRPSTACK. . . , s.d.), e públicos como *The Things Network (TTN)* (THE. . . , s.d.) entre outros.

Os *network server*s podem estar associadas a uma topologia diferente, onde o *gateway* é responsável por realizar as funções de gerenciamento da rede. O nome comercialmente utilizado pela Khomp para esta topologia é de *network server* interno,

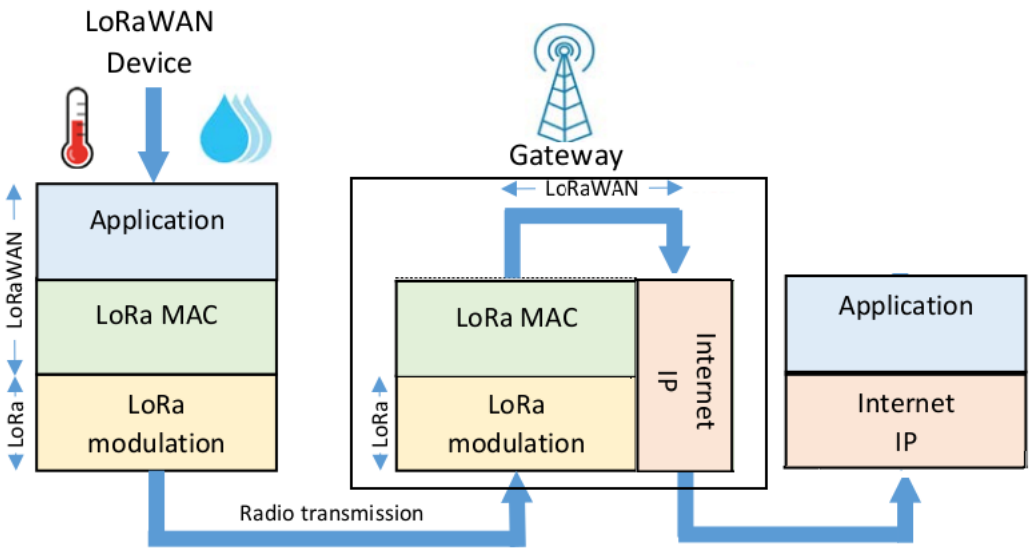
em oposição ao nome *network server* externo que é quando o mesmo fica em uma estrutura externa ao gateway. É comum encontrar essa topologia com o nome de gateway centralizado. E obviamente, existe apenas um gateway nesta topologia.

Figura 4 – Topologia *network server* interno



Fonte: Arquivo Pessoal

Figura 5 – Fluxo LoRaWAN em *network server* interno



Fonte: fluxograma da Figura 3 adaptado

Essa topologia alternativa para arquitetura LoRaWAN pode ser vantajosa em cenários de aplicações pequenas sem muitos sensores ou sem a necessidade de uma área de cobertura enorme e integrada (MONTAGNY, 2022). É muito mais rápido para o integrador, já que pouquíssima configuração é necessária para se ter LoRaWAN (MONTAGNY, 2022). Além disso, pode ser preferível que os dados não passem por uma outra estrutura, e sejam todos processados dentro do gateway, deixando para aplicação apenas o tratamento dos mesmos. Uma outra vantagem é que em caso de perda de comunicação com a internet, o gateway como *network server* também funciona como *datalogger*, armazenando todos os dados que não foram possíveis de serem enviados à nuvem, e são entregues quando a comunicação for reestabelecida.

Os gateways Khomp de Telemetria LoRaWAN possuem a topologia *network server* interno implementada, mas também permitem também que o gateway aponte para um servidor LoRaWAN externo, ficando a critério do integrador de decidir a qual topologia quer aderir. Apesar de a a maioria dos integradores usarem essa topologia alternativa, é importante salientar que nem todas as funcionalidades do protocolo LoRaWAN estão implementadas no gateway quando ele atua como *network server*, algumas nem poderão ser implementadas pela falta de processamento, mas outras como o ADR podem, e tornam o *gateway* cada vez mais completo como solução LoRaWAN.

3.4.3 *Uplink*

Uplink é a mensagem enviada na direção do dispositivo final para o Gateway ou rede LoRaWAN (MONTAGNY, 2022). O *uplink* pode ocorrer por meio de mensagens

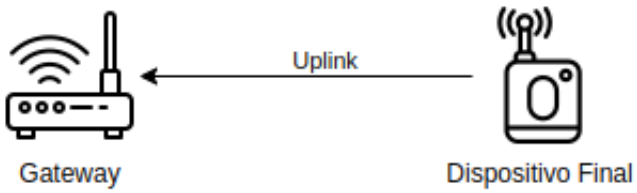
temporizadas, por exemplo, a cada 5min ou quando algum evento específico acontece, como quando um sensor detecta que uma porta foi aberta.

O *uplink* é a parte predominante da comunicação bidirecional (MONTAGNY, 2022), permitindo que os dispositivos finais enviem informações para o gateway. É através do *uplink* que os dados coletados pelos sensores ou dispositivos finais são transmitidos para a rede, possibilitando a monitoração remota, controle ou análise desses dados. E se nenhum problema tiver ocorrido, o Network Server sempre está pronto pra receber um *uplink*. Note que estar pronto pra receber um *uplink* não significa garantia de entrega, dado que pode ocorrer interferência ou o canal do *gateway* estar ocupado.

Esta mensagem pode ser com confirmação ou não (MONTAGNY, 2022). Quando for requisitado a confirmação, o *network server* precisa responder com um *downlink*. Mais informações sobre o funcionamento de *downlink*s é explicado no capítulo 3.4.4. O desenvolvedor e fabricante do dispositivo final que pode decidir em que momento o *uplink* deve solicitar confirmação ou não.

A capacidade de *uplink* é um fator importante a ser considerado, especialmente em aplicações com dispositivos finais de baixo consumo de energia ou com restrições de transmissão. A taxa de dados, potência de transmissão e outros parâmetros configuráveis podem afetar o desempenho e a confiabilidade do *uplink*. Além disso, a disponibilidade de canais de comunicação e a gestão eficiente do espectro também influenciam a capacidade de *uplink* de uma rede.

Figura 6 – Direção dos dados no *uplink*

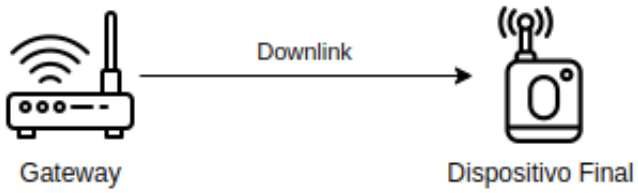


Fonte: Arquivo Pessoal

3.4.4 Downlink

O dowlink é a mensagem enviada na direção oposta a do *uplink*. Neste caso, também conhecida como mensagens de comandos, são pacotes enviados ao dispositivo final, a partir do *network server* (MONTAGNY, 2022) com o objetivo de mudar o comportamento ou o funcionamento do mesmo, solicitar alguma informação extra a respeito do dispositivo final ou simplesmente confirmar o recebimento de um *uplink*. Ele complementa o *uplink*. Juntos, o *uplink* e o *downlink* permitem a troca de dados e comandos entre os dispositivos finais e a rede.

Figura 7 – Direção dos dados no *downlink*

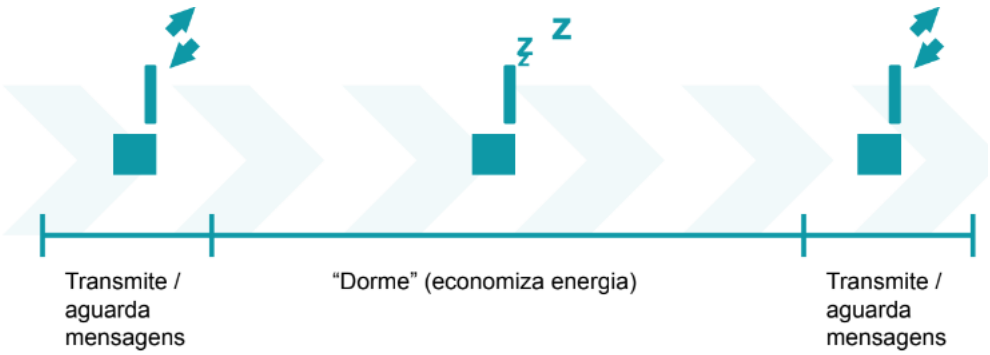


Fonte: Arquivo Pessoal

Existem três classes de Downlink - A, B e C. Cada uma dessas classes tem relação com o modo de operação e a disponibilidade de energia do dispositivo final (MONTAGNY, 2022).

Na classe A de *downlink*s, o dispositivo abre duas janelas para o recebimento de *downlink*s. Durante essa janela, o rádio do dispositivo fica ligado e preparado para o recebimento de comandos. Após isso o rádio fica desligado até o próximo *uplink*, e qualquer tipo de mensagem enviada para o dispositivo fora dessas janelas é descartada. A primeira janela se abre no tempo T (normalmente T é 1s ou 5s) após envio do *uplink*. E a segunda janela é exatamente em T mais 1s. Pelo fato do rádio estar sempre desligado, afirma-se então que a classe A tem a finalidade de poupar energia dos dispositivos, sendo assim a classe mais econômica.

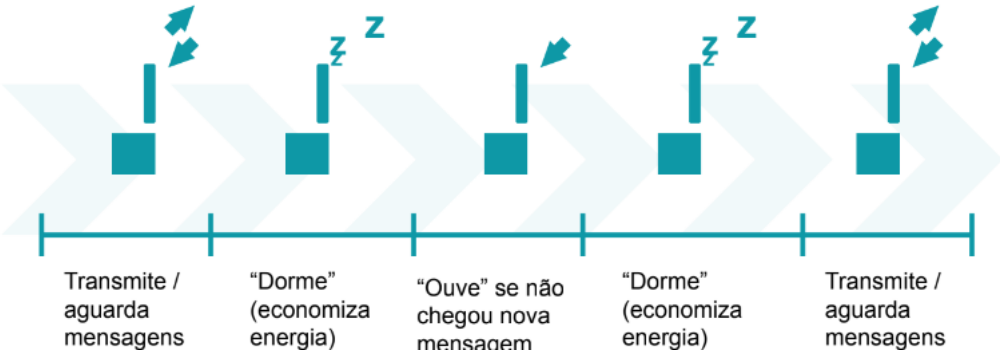
Figura 8 – Janelas de envio e recebimento Classe A



Fonte: Cartilha Khomp LoRa

Os dispositivos finais da Classe B têm um comportamento de comunicação síncrona. Além das janelas de recepção fixas da Classe A, os dispositivos finais da Classe B também sincronizam periodicamente o *gateway* através de um *beacon* enviado em intervalos regulares. A sincronização com o *beacon* permite a abertura de janelas de recepção adicionais e mais flexíveis para receber *downlink*s. É importante salientar que os dispositivos finais desenvolvidos pela Khomp não operam com classe B, o gateway com função de *network server* não é capaz de operar também.

Figura 9 – Janelas de envio e recebimento Classe B



Fonte: Apostila LoRaWAN Khomp

A classe C é a classe mais avançada em termos de capacidade de *downlink*. Os dispositivos finais da classe C, possuem o rádio sempre ligado e ouvindo. Isso permite os *downlink*s sejam enviados aos dispositivos a qualquer momento, oferecendo uma resposta praticamente em tempo real. No entanto, a operação contínua das janelas de recepção da Classe C resulta em um consumo de energia mais alto em comparação com as outras classes.

Figura 10 – Janelas de envio e recebimento Classe C

Fonte: Apostila LoRaWAN Khomp

Dispositivos em classe C são capazes de *downlink*s classe A e B, da mesma forma que classe B consegue receber classe A. Mas o oposto não é verdade.

Em seu portfólio de sensores e dispositivos finais, a Khomp conta com produtos que funcionam exclusivamente à a pilhas, e outros que são híbridos, aceitam pilhas e alimentação externa. Um padrão de fábrica definido pela empresa, que se o dispositivo está recebendo alimentação externa, ele opera em classe C. Se ele está sendo alimentado por pilhas, opera em Classe A. É possível mudar a classe de operação dos dispositivos Khomp, mesmo que não recomendado, por meio de *downlink*.

3.4.5 Pacotes LoRaWAN

A estrutura dos pacotes LoRaWAN possuem os seguintes dados (LORA ALLIANCE, 2020):

- *Header*: O cabeçalho contém informações sobre o pacote, tipo de mensagem (por exemplo, *uplink* ou *downlink*) e informações de controle adicionais do protocolo. O tamanho do cabeçalho pode variar dependendo das necessidades da rede e do tipo de mensagem.
- *Payload*: O payload é a parte do pacote que contém os dados. Isso pode incluir informações como leituras de sensores, comandos ou qualquer outra informação relevante para a aplicação específica. O tamanho do *payload* pode variar e é limitado pela taxa de dados e restrições do sistema. As literaturas também chamam de *MACPayload*.
- *MIC (Message Integrity Code)*: O *MIC* é um código de integridade da mensagem que é adicionado ao final do pacote para garantir a integridade dos dados durante a transmissão. É usado para verificar se os dados foram modificados ou corrompidos durante a transmissão, ou até mesmo impede a tentativa de clonagem do sinal.

O *header*, *Payload* e o *MIC* estão dentro de uma outra estrutura chamada *PhyPayload*.

Tabela 5 – Estrutura do *PHYpayload*

PhyPayload	MHDR	MACPayload	MIC
Tamanho(bytes)	1	7..N	4

Fonte: *LoRaWAN layer specification*

Esse é a estrutura de um *PHYPayload* (*PHY* vem de *physical*) normal, onde MHDR significa *MAC Header* (LORA ALLIANCE, 2020). Existem outras possibilidades de mensagens, como a de um *Join-Request* - Pedido de entrada na rede, enviado a partir do dispositivo final- , ou *Join-Accept* que é o envio da rede ao dispositivo, informando a aceitação de entrada.

E o *MACPayload* é também dividido em três estruturas estruturas de dados, seguindo o formato:

Tabela 6 – Estrutura do *MACPayload*

FHDR	Fport	FRMPayload

Fonte: *LoRaWAN layer specification*

O *Fport* é um campo de 1 byte que indica o número da porta lógica usada para o encaminhamento dos dados. Ele permite a categorização da mensagem. *FRMpayload*

é o conteúdo da mensagem propriamente dito, seja dados sensoriais ou de status. Ele vem criptografado. O FHDR (*Frame Header*) é a parte que contém dados sobre qual o dispositivo está enviando (LORA ALLIANCE, 2020), ou para qual endereço será enviado, a mensagem em questão. Também conta com um controlador do *frame* (FCtrl), que contém informações sobre a finalidade da mensagem, um contador (FCnt) de mensagens, e um *frame* de opções, que poderá conter mensagens de comando *MAC* enviadas por *Piggyback* (termo usado pra quando outros dados são enviados como anexo na mensagem) como por exemplo os comandos de ADR. O (*Frame Header*) possui a seguinte estrutura:

Tabela 7 – Estrutura do *Frame Header*

FHDR	DevAddr	FCtrl	FCnt	FOpts
Tamanho(bytes)	4	1	2	0..15

Fonte: *LoRaWAN layer specification*

Tanto *downlink* quanto *uplink* possuem essa mesma macro estrutura de mensagem. Muda o conteúdo, tendo identificador do que é a mensagem no *MAC Header*.

O campo do *FCtrl* é bastante importante para a execução do ADR. É preciso realizar leituras, ou deixar determinados *bits* como verdadeiro. O tamanho desse campo é 1 *byte*.

Tabela 8 – Estrutura de Bits do *FCtrl* para *downlink*

FCtrl	ADR	RFU	ACK	Fpending	FOptsLen
Bit	7	6	5	4	[3..0]

Fonte: *LoRaWAN link layer specification*

É necessário que o sétimo *bit* seja 1 (um) quando for um *downlink* de comando para trocar o DR.

Tabela 9 – Estrutura de Bits do *FCtrl* para *Uplink*

FCtrl	ADR	ADRACKReq	ACK	ClassB	FOptsLen
Bit	7	6	5	4	[3..0]

Fonte: *LoRaWAN link layer specification*

Para *uplink*, o sétimo *bit* indica se o ADR está disponível no dispositivo ou não. O sexto *bit* é uma requisição do dispositivo para troca de DR. Não é uma regra que o DR precisa mudar quando esse bit vier 1, mas pode ser usado no algoritmo de ADR como decisão de quando calcular o novo DR.

Tanto no *uplink* quanto no *downlink* possui um campo chamado *FOptslen*. Esse campo possui 4 bits, ou seja, pode ter um valor de 0 a 16 e indica o tamanho do

FOpts presente no *FHDR* - Tabela 7. Ou seja, quando houver algum comando *MAC* é necessário que seja informado o tamanho do comando no campo *FOptsLen*.

3.4.6 Comandos *MAC*

Os comandos *MAC* (*Media Access Control*) são uma parte importante do protocolo LoRaWAN e são usados para controlar e gerenciar a comunicação entre os dispositivos finais e a rede (LORA ALLIANCE, 2020). Esses comandos são enviados pelo *network server* para os dispositivos finais ou vice-versa e têm o objetivo de fornecer instruções, configurações ou solicitar informações específicas. Os comandos *MAC* podem estar no *FOpts*, presente no *frame header*, como mostrado na Tabela 7 (LORA ALLIANCE, 2020). Como dito anteriormente, comandos *MAC* são bi-direcionais, ou seja eles estão em *uplinks* ou *downlinks*. Em caso de comandos *MAC* em *downlink*s, eles também podem estar no *FRMPayload* desde que não tenha outro conteúdo para ser enviado nessa estrutura, e nesse caso o *FPort* seja igual a 0 para indicar que o *FRMPayload* se trata de um comando MAC.

Os comandos *MAC* possuem a seguinte estrutura:

Tabela 10 – Estrutura genérica de um comando *MAC*

CID	Payload

Fonte: *LoRaWAN link layer specification*

O *CID (Command Identifier)* é um *byte* que identificará qual é o comando *MAC*. O tamanho do *payload* pode variar, de acordo com o comando. Quase todos os comandos *MAC* são pares de requisição e resposta, onde quando um lado pede a requisição, o outro envia a respectiva resposta. Quando são pares de requisição e resposta, eles possuem o mesmo CID. O dispositivo final e o *gateway* não possuem comandos em comum. No contexto do ADR existem dois comandos *MAC* fundamentais, o *LinkADRReq* que é a solicitação ao dispositivo para trocar as configurações relacionadas a taxa de dados e a potência do rádio. E o *LinkADRAns* que é a resposta enviada ao *gateway*, indicando o status se foi possível ocorrer a troca solicitada (IMPLEMENTING. . . , s.d.).

Tabela 11 – Estrutura *LinkADRReq*

Payload	CID	DataRate-TXPower	CHMask	Redundancy
Tamanho(bytes)	1	1	2	1

Fonte: *LoRaWAN link layer specification*

O *CHMask*, ou máscara de canal, é pra configurar em quais canais o dispositivo pode mandar dados, e o *Redundncy* é pra configurar o número de repetições de um uplink, para minimizar a perda de pacote. O campo *DataRate-TXPower* é o campo variável para este proejeto.

Tabela 12 – Estrutura *LinkADRAns*

Payload	CID	Status
Tamanho(bytes)	1	1

Fonte: *LoRaWAN link layer specification*

Sendo que o campo status possui 5 bits não sendo utilizados no momento (RFU) e outros 3 pra bits pra confirmar o status da mudança da potência do rádio, taxa de dados e do Canal.

3.5 ADAPTIVE DATA RATE

ADR (Adaptive Data Rate) é um conceito utilizado em comunicações sem fio para otimizar a taxa de dados (data rate) de transmissão entre dispositivos. O objetivo do ADR é adaptar dinamicamente a taxa de dados de acordo com as condições de comunicação e as características do canal, a fim de maximizar a eficiência, a qualidade e a confiabilidade da transmissão (KUFAKUNESU; HANCKE; ABU-MAHFOUZ, 2020), utilizando como parâmetro o sinal e o ruído recebido nas mensagens.

No contexto do LoRaWAN, o ADR é uma funcionalidade que otimiza a eficiência da comunicação entre os dispositivos finais e o *gateway*. O objetivo do ADR é ajustar dinamicamente a taxa de dados (*data rate*) das transmissões LoRa com base na relação entre sinal e o ruído sentidos pelo receptor (SEMTECH, 2016). É implementado de forma inteligente para equilibrar a taxa de dados e a sensibilidade do receptor, levando em consideração a distância entre os dispositivos - variável não é medida nem estimada, mas se o sinal está forte se considera que o dispositivo está perto - o nível de ruído e outros fatores que afetam a qualidade da comunicação. Esse ajuste é realizado analisando os sinais recebidos pelos *uplinks*, adaptando-se às condições de canal em constante mudança.

O principal objetivo do ADR é maximizar a eficiência espectral e a vida útil da bateria dos dispositivos finais (NETWORK, s.d.). Ele alcança isso diminuindo o *ToA Time On Air* (tempo no ar), expressão utilizada para o tempo que uma mensagem leva pra ser modulada, através da diminuição do fator de espalhamento. Existem casos onde o ADR ajusta a largura de banda também, subindo de 125KHz para 500KHz (ou 250khz, dependendo da região), mas não é o caso deste projeto, pois não é usada o DR6 - SF08BW500 (porque apenas um canal é capaz de receber nessa taxa de dados).

3.5.1 Dispositivo Final informa que pode trocar a taxa de dados

A informação de se o dispositivo é capaz de trocar sua potência EIRP ou a taxa de dados vem num *bit* dentro do cabeçalho de um *uplink*, conforme mostrado na

Tabela 9. Este *bit* pode estar falso por algumas razões, entre elas a incapacidade do dispositivo de realizar esses ajustes, ou até mesmo o administrador da rede decidiu desativar manualmente (IMPLEMENTING. . . , s.d.). Por isso é importante realizar checagem se está ativo antes de iniciar o procedimento, para evitar processamentos ou envios de mensagens desnecessários.

3.5.2 Comando para alterar o DR

Desde que a rede permita, e o dispositivo esteja disponível pra trabalhar com ADR (conforme explicado na Seção 3.5.1) é possível trocar de forma arbitrária e manual (ou seja, sem nenhuma inteligência de máquina) a taxa de dados e a potência de rádio (IMPLEMENTING. . . , s.d.). Embora o mecanismo de ADR seja responsável por ajustar automaticamente a taxa de dados com base nas condições do canal, é permitido aos desenvolvedores e administradores de rede definir manualmente o data rate para comunicações específicas. Isso se dá principalmente pelo fato de que a troca de data rate seja um processo independente de qualquer algoritmo adaptativo. Sendo função do algoritmo criar o comando ao final de cada decisão.

Essa abordagem pode ser benéfica em situações em que há flutuações significativas na qualidade do sinal devido a variações físicas no ambiente. Nessas circunstâncias, o algoritmo de ADR pode ter dificuldades em estimar adequadamente a taxa de dados ou a potência de transmissão mais eficiente. Como alternativa, os usuários têm a opção de aumentar o fator de espalhamento, embora seja importante considerar as consequências, como o aumento do consumo de energia.

Para realizar a alteração de DR é necessário enviar um *downlink* contendo o comando *MAC LinkADRReq*. Utilizando como base a estrutura de dados do *LinkADRReq*, presente na Tabela 11, é possível definir o nova nova potência de rádio e a nova taxa de dados.

Tabela 13 – Estrutura *LinkADRReq*

Payload	CID	DataRate-TXPower	CHMask	Redundancy
Valor	03_h	*DRTX*	*FF*00	00_h

Fonte: *LoRaWAN link layer specification*

Como mostrado na Tabela 11, é um byte para definir o *DataRate-TXPower*, ou seja, será um caractere hexadecimal para a taxa de dados, e outro caractere para a potência de transmissão.

3.5.3 Algoritmo para mudança de Data Rate

Existem diversos algoritmos de *Adaptive Data Rate* (ADR) disponíveis para implementação, cada um com suas próprias peculiaridades e características de tomada

de decisão. No contexto do protocolo LoRaWAN, os desenvolvedores têm a liberdade de escolher e implementar o algoritmo que considerarem mais adequado para suas necessidades.

Neste trabalho, o autor optou por implementar o algoritmo criado e sugerido pela *Semtech*, referenciado em (SEMTECH, 2016). A empresa responsável pela manutenção do protocolo LoRaWAN. Essa decisão foi motivada pelo fato de que a *Semtech* é uma instituição renomada e uma referência em termos de inovação e desenvolvimento relacionados a esse protocolo. Além disso, a Khomp, já utiliza a *Semtech* como referência no desenvolvimento em outros produtos, como os seus dispositivos finais. Além disso, é importante ressaltar que outras redes LoRaWAN amplamente conhecidos no mercado, como *Chirpstack* e *The Things Network* (TTN) (NETWORK, s.d.), também utilizam esse mesmo algoritmo.

Dito isso, essa seção será inteiramente baseada no documento *LoRaWAN – simple rate adaptation recommended algorithm*.

Esse algoritmo leva em conta também o cenário onde a rede possuí mais de um *gateways*, mas como mencionado anteriormente, esse projeto se trata da implementação de ADR para *gateways*individuais que agem também como *network server*s.

Como explicado na Seção 3.3.4, a relação sinal-ruído (SNR) é uma variável mais completa do que o indicador de intensidade do sinal recebido (RSSI), justamente porquê além de levar em conta a potência do sinal, é considerado também o ruído. E tanto o ruído quanto a intensidade do sinal são importantes importantes na hora de decidir qual o fator de espalhamento e potência de transmissão são mais adequados.

A partir do momento que o ADR é ligado (no lado do dispositivo final e do *gateway*), assume-se então que os dispositivos estão em suas posições de operação, e se inicia a rotina onde é analisado as mensagens recebidas pelo *gateway* - os *uplink*. Todos os valores de SNR são armazenados em uma lista dinâmica de tamanho N. Aqui usaremos, conforme sugerido, o valor de N igual a 20. Quando essa lista é preenchida, e chegam novos valores, os valores mais antigos são descartados.

De acordo com (IMPLEMENTING. . . , s.d.), existem duas opções para decidir quando se deve iniciar os cálculos de um novo DR. Para este projeto foi optado que é a cada 20 *uplinks* recebidos pelo *gateway*.

É calculado o valor de *SNRmargin* através da seguinte equação:

$$SNR_{margin} = SNR_{max} + (20 - DR * 2.5) - margin_{db} \tag{2}$$

onde,

- SNR_{max} é o valor máximo de SNR retirado da lista.
- DR é a taxa de dados atual (de 0 até 5).
- $margin_{db}$ é uma margem escolhida pelo desenvolvedor.

Através dessa equação podemos perceber duas coisas:

- Que quanto maior for a taxa de dados, menor será o SNR_{margin}, ou seja, é condicionado a manter a taxa de dados alta, somente em casos onde o ruído está muito superior ao sinal.
- Quanto maior for o valor da constante $margin_{db}$, também menor será o SNR_{margin}. Portanto para dinâmicas mais conservadores, deve-se uma escolher $margin_{db}$ maior. Esse valor pode ser um valor geral, ou específico para cada dispositivo.

Após se obter o valor de SNR_{margin}, é calculado o N_{steps}, em que esse será o número de iterações que o algoritmo irá executar a fim de encontrar o novo valor para a taxa de dados e a potência.

$N_{steps} = int(SNR_{margin}/3)$

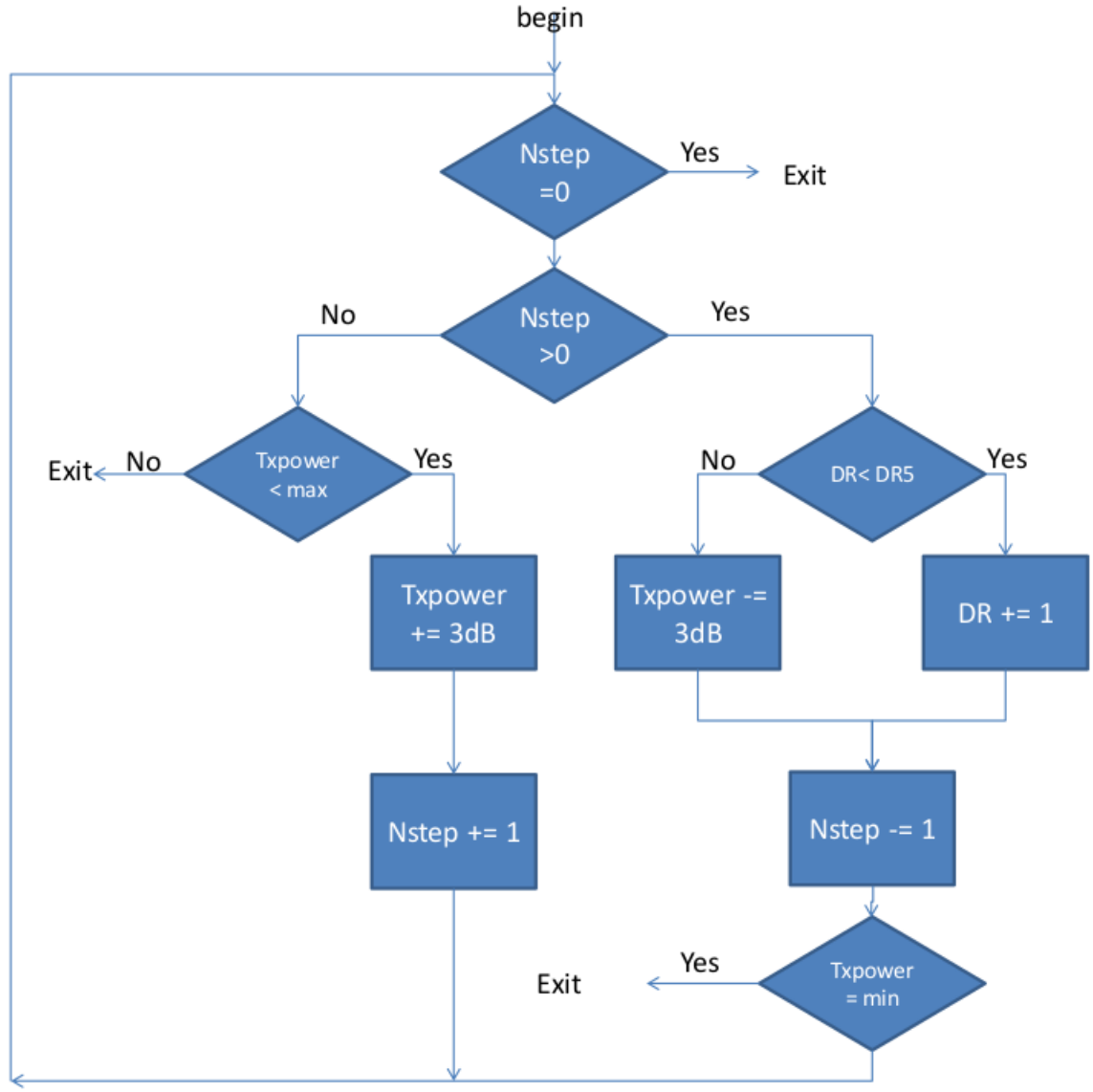
Para se ter um N_{steps} superior a 1, é necessário ter um SNR_{margin} superior a 3. Supondo que esteja em DR0, e que $margin_{db}$ seja igual a 10, temos:

$3 = SNR_{max} + 20 - 10$

Quando em DR 0, o SNR_{max} precisa ser no mínimo -7 pra subir para DR 1.

O fluxograma da Figura 11 representa como o algoritmo itera utilizando como valor base o N_{steps} previamente calculado.

Figura 11 – Fluxograma ADR



Fonte: *Semtech*

Pelo fluxograma é possível perceber quê:

- O algoritmo nunca diminuirá o *data rate*, ou seja, ou seja se for necessário mudança no data rate será sempre pra aumentar o valor
- A potencia de rádio, representada no fluxograma por Tx_{power} só será diminuída uma vez que o DR já esteja no máximo.
- Se o sinal não estiver adequado, existe um processo de aumentar o Tx_{power} até que chegue no valor padrão.

Ao definir as alterações na taxa de dados o algoritmo deve preparar um *downlink* contendo as informações novas de taxa de dados e potência de rádio.

Esse algoritmo, e todos os outros de adaptação de taxa de dados para redes LoRaWAN, possuem limitação quanto a condições físicas do local. Eles foram feitos para condições estáticas. Ele não deve ser ativado em sensores móveis. Nesse cenário diversas coisas podem acontecer, entre elas a perda de rede por parte do dispositivo. Ou também em cenários onde podem ter obstruções não previsíveis, como por exemplo uma porta de metal no caminho entre o *gateway* e o dispositivo final. Nesses contextos, o algoritmo pode acabar regulando a taxa de dados e a potência para um cenário onde o dispositivo está perto, sem obstruções entre ele e o *gateway*, e quando as condições físicas são alteradas pode ocorrer a perda da conexão sem nem mesmo ter como receber uma nova taxa de dados. Portanto, nessas condições é altamente recomendado que o ADR não esteja desligado.

3.5.4 ADR Backoff

ADR Backoff, o recuo de ADR, é o que o dispositivo inicia ao perder comunicação com o *gateway* após alguma mudança em sua taxa de dados ou potência de rádio (IMPLEMENTING. . . , s.d.). É uma implementação exclusivamente no lado do dispositivo, e ela serve pra minimizar a perda total de comunicação provocada por uma eventual alteração precipitada, ou alteração das condições físicas do ambiente. Primeiro o dispositivo coloca as configurações de rádio para o seu padrão (normalmente é a potência máxima), e depois sobe a taxa de dados gradualmente, até que a comunicação seja reestabelecida ou que chegue na taxa de dados padrão do dispositivo.

4 DESENVOLVIMENTO DO PROJETO

Este capítulo fala sobre as ferramentas e técnicas necessárias para o desenvolvimento, além de trazer também testes e o comportamento do algoritmo implementado.

4.1 FERRAMENTAS UTILIZADAS

Nesta seção serão tratadas as ferramentas utilizadas para o desenvolvimento e testes desse projeto.

4.1.1 VScode

Visual Studio Code, também conhecido como VSCode, é um editor de código-fonte gratuito e de código aberto desenvolvido pela Microsoft. Ele oferece uma ampla gama de recursos que ajudam os desenvolvedores a escrever, depurar e executar o código com eficiência.

Uma das principais vantagens do VSCode é a sua extensibilidade, permitindo que os usuários adicionem funcionalidades através de extensões criadas pela comunidade ou personalizem o editor de acordo com suas necessidades específicas.

4.1.2 Python

Python (PYTHON. . . , s.d.) é uma linguagem de programação de alto nível, interpretada e de código aberto, criada em 1991 por Guido van Rossum. É uma das linguagens de programação mais populares do mundo, sendo usada para desenvolver aplicativos de desktop, aplicativos web, análise de dados, inteligência artificial, aprendizado de máquinas, automação, entre outros.

Python possui uma sintaxe simples, deixando mais fácil a implementação de novas funcionalidades que fazem o projeto crescer. Além disso, ela oferece uma ampla variedade de bibliotecas e *frameworks* para facilitar o desenvolvimento de aplicações em diferentes áreas.

A linguagem é interpretada, o que significa que não é necessário compilar o código antes de executá-lo, tornando o processo de desenvolvimento mais rápido. Também possui um sistema de gerenciamento de memória automático, o que simplifica o processo de programação e reduz a probabilidade de erros.

Python é uma linguagem multiplataforma, o que significa que pode ser executada em diferentes sistemas operacionais, incluindo Windows, macOS e Linux (Inclusive Linux Embarcado). Também é uma linguagem extensível, permitindo a criação de módulos em C ou C++ para melhorar a velocidade de execução.

Devido à sua simplicidade, facilidade de uso e ampla variedade de bibliotecas e frameworks, Python é amplamente utilizada em muitas áreas de aplicação;

4.1.3 Javascript

JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a objetos, usada principalmente para desenvolver aplicativos web interativos. Ela é executado no navegador do usuário, permitindo a criação de conteúdo dinâmico, como animações, gráficos e formulários interativos. A linguagem é tipada dinamicamente, o que significa que não é necessário declarar o tipo de uma variável antes de usá-la. Além disso, o JavaScript é uma linguagem interpretada, o que significa que o código é executado diretamente no navegador do usuário, sem a necessidade de compilação. O JavaScript é uma das linguagens de programação mais populares do mundo, devido à sua capacidade de criar aplicativos interativos e dinâmicos em um navegador da web. Além disso, ele possui uma ampla variedade de *frameworks* e bibliotecas, que simplificam o desenvolvimento de aplicativos, como o Vue.js, usado para fazer a página web do *Gateway* Khomp de Telemetria, e ferramenta que facilitou muitas atividades descritas no presente relatório, O JavaScript é uma linguagem multiplataforma, o que significa que pode ser executada em diferentes sistemas operacionais e navegadores.

4.1.4 Git e GitLab

Git (GIT. . . , s.d.) é uma ferramenta de versionamento que permite que os desenvolvedores colaborem em projetos de software, rastreiem mudanças no código-fonte e revertam para versões anteriores se necessário. Essa ferramenta é distribuída, o que significa que cada desenvolvedor tem uma cópia completa do repositório em seu próprio computador. Isso permite que eles trabalhem offline e façam alterações sem afetar o trabalho de outros desenvolvedores. É sistema que de ramificação (*branching*) e mesclagem (*merging*) para gerenciar as mudanças no código. Cada desenvolvedor pode criar sua própria ramificação para trabalhar em uma funcionalidade específica ou correção de bug, e depois mesclar suas alterações de volta na ramificação principal (branch master) quando estiverem prontas. Os usuários podem interagir com o Git por meio da linha de comando ou por meio de interfaces. Git é amplamente utilizado por desenvolvedores de software e é uma ferramenta essencial para colaboração em projetos de código aberto. GitLab é uma plataforma de desenvolvimento de software baseada na web que fornece um conjunto abrangente de ferramentas para gerenciamento de código-fonte, colaboração, integração contínua, entrega contínua (CI/CD) e rastreamento de problemas. Ele foi lançado em 2011 e é desenvolvido pela empresa GitLab Inc. Possuindo suporte integrado ao git, o gitlab é uma maneira dos desenvolvedores manterem seus projetos, modificações, e versionamento salvos em segurança na nuvem. O GitLab é utilizado também para geração de versão de testes e versões finais, utilizando a integração e a entrega contínua.

4.1.5 Kit de Desenvolvimento

Para realizar o desenvolvimento e graduais evoluções, foi usado um *Gateway* Khomp de Telemetria *LoRaWAN indoor*, com os pinos para comunicação serial soldados na placa e acesso via SSH. Dessa maneira é possível editar os códigos em python, javascript ou gerar binários de outras linguagens, e enviar diretamente para serem executados gateway. Acessando via serial ou SSH também é possível monitorar *logs*, *debugs* e analisar comportamentos do código. Tudo isso é extremamente útil pro desenvolvimento de funcionalidades, assim como o ADR.

E especialmente para o desenvolvimento do ADR, não poderia faltar o dispositivo final. Não fazemos modificações de *firmware* nele, mas enviamos comandos e precisamos monitorar o estado. Também conectamos utilizando a entrada serial para realizar as leituras.

4.2 IMPLEMENTAÇÃO DO ADR

A cada *uplink* recebido pelo gateway, uma série de procedimentos são executados. Desde funções necessárias para o funcionamento e existência do LoRaWAN dentro do gateway, até as novas funcionalidades relacionadas ao ADR. Entre os processos já existentes estão:

- É checado se a mensagem recebida é de um dispositivo pertencente ao *network server* do *Gateway*.
- São gravados os valores de SNR e RSSI sentidos pelo *gateway* ao receber essa mensagem.
- É verificado se há *downlink* para ser enviado, e se houver (ou se o *uplink* pedia confirmação) é preparado o *downlink* para ser enviado dentro das janelas de envio.
- É identificado o modelo do dispositivo final que enviou aquela mensagem.
- É realizada a conversão (*parsing*) das mensagens sensoriais recebidas.

As funções relacionadas ADR implementadas no presente projeto são executadas após a identificação dos valores de RSSI e SNR.

4.2.1 Leitura do *Payload* LoRaWAN

O processo de ler um *payload* consiste em percorrer os *bytes*, e dar o significado a eles, adotando as respectivas funções de acordo com as instruções fornecidas. Boa parte dessa leitura já é feita pelo *gateway*, mas para a implementação deste projeto foi

necessário dar atenção especial pra alguns outros campos ainda não implementados no *network server* do *gateway*, como por exemplo a leitura do bit de ADR presente no *FCtrl*, além é claro do *FOpts*. Os *payloads* vem em bytes, e é necessário converter para hexadecimal, que em *Python* isso pode ser feito utilizando funções nativas da linguagem, como o método *.hex()*. Após convertido para hexadecimal, temos um *payload* organizado em lista, onde cada posição dessa lista representa um *byte*, podendo ter valores de 00 a FF.

O *payload* de um *uplink* qualquer vem da seguinte maneira:

```
PHYPayload = [80, 8B, A9, C4, 4B, 82, 17, 00, 03, 07, 04, DE, 11, 09,
DD, 67, 69, 13, 81, 94, 84, C3, 7B, 69, 7E, D3, 85, C8]
```

Através da estrutura mostrada na Tabela 5, sabemos que *MHDR* é a posição 0 (primeiro byte), e o valor é 80, e que os quatro últimos são do *MIC*. Então, o restante é o *MAcCPayload*, que possui a estrutura mostrada na Tabela 6.

```
MACPayload = [8B, A9, C4, 4B, 82, 17, 00, 03, 07, 04, DE, 11, 09, DD,
67, 69, 13, 81, 94, 84, C3, 7B, 69]
```

O *MACPayload* começa com *frame header*, cuja estrutura é mostrada na Tabela 7, e que os 4 primeiros bytes são para o endereço de rede do dispositivo final (DevAddr). Então, o quinto byte (posição 4), de valor 82_h, é o *FCtrl*. Para interpretar esse campo, é necessário analisar os bits que formam o 82_h. Os *bits* do *FCtrl* para este projeto são *ADRACKReq*, *ADR* e *FOptslen*. Para fazer isso, em *Python*, podemos converter o hexadecimais para uma estrutura de *string* de *bits*, utilizando a seguinte sintaxe: "$:08b$".*format*(*valorHEX*)

A partir dela, temos que, por exemplo que 82_h = 1000010.

Assim, podemos separar os bits conforme a Tabela 9, e temos:

Tabela 14 – *FHDR* do *payload*

FCtrl	ADR	ADRACKReq	ACK	ClassB	FOptsLen(0..3)
Valor	1	0	0	0	0010

Fonte: Arquivo Pessoal

Com este resultado é possível ver que o ADR está ativo, e que o tamanho do *FOpts* é 2. Antes do início desse projeto, o *FOpts* não era tratado, e não tinha problema porque os dispositivos finais desenvolvidos pela Khomp não mandavam comandos MAC por essa estrutura. Mas agora, com a implementação do ADR (e eventualmente de outras funcionalidades LoRAWAN e comandos *MAC*) isso acontece, e sem o tratamento desse campo, fica comprometida a leitura de todo o restante do *payload*, inclusive os valores sensoriais essenciais para o funcionamento da aplicação. Agora se o valor do *FOptslen* for N, sendo N um valor maior que 0, será lido os últimos

N bytes do *FHDR*. Além disso, é adicionado também um deslocamento de N a tudo que vem após o *FHDR*.

Agora que as leituras para o *FCtrl* estão implementadas, é possível ler o *FHDR* completamente, sabendo o tamanho total do *frame*.

Tabela 15 – Estrutura do *Frame Header* para o MACPayload - inserir referência do *payload*

FHDR	DevAddr	FCtrl	FCnt	FOpts
Valor(hex)	8BA9C44B	82	1700	0307

Fonte: Arquivo Pessoal

O primeiro byte do *FOpts* é o CID de um comando MAC, o 03_h indica que é um *LinkADRAns* (Tabela 12, que é enviado após um envio um *LinkADRReq*, e o status indica se as mudanças foram efetuadas com sucesso ou não. O valor é 07_h, e: 07_h = 00000111

Como dito na Seção 3.4.6, os primeiros *bits* do *LinkADRAns* são RFU. Os últimos 3 indicam foi possível alterar a potencia de rádio, máscara de canal e a taxa de dados.

Essas são as leituras mais importantes implementadas para esse projeto, o restante como por exemplo o *FRMPayload* já estava implementada.

4.2.2 Envio de comandos de Troca de taxa de dados e potência TX

Para trocar a taxa de dados e a potência de transmissão, é criado um monitoramento responsável por observar sempre que existe uma solicitação de alteração de *data rate* ou da potência de transmissão, seja uma solicitação criada por meio do algoritmo de ADR ou por meio de uma requisição do usuário. Quando é identificado, se verifica se existe algum *downlink* agendado, e se não houver é criado um agendamento de um Classe A. Independente de qual que seja o *downlink*, é preciso que o *fctrl* esteja informando que é uma solicitação de alterar o ADR. Além disso, através da Tabela 11, sabemos que o *FOpts* precisa ter tamanho 5, portanto esse será o valor do *FOptsLen*. Conforme a Tabela 8, precisamos ter a seguinte estrutura:

Tabela 16 – Estrutura de Bits do *FCtrl* para *downlink*

FCtrl	ADR	RFU	ACK	Fpending	FOptsLen
Valor	1	0	1	0	0101

Fonte: *LoRaWAN link layer specification*

Convertendo, temos:
10100101 = $A5_h$ Então o *Frame Control* terá valor de $A5_h$.

Agora é preciso montar a estrutura do *FOpts*, inserindo o que serão os novos parâmetros que estão sendo configurados. Supondo que a nova taxa de dados seja *newDR* e a nova potência de transmissão seja *newTX*, onde essas são as variáveis resultantes de um cálculo do ADR ou de uma entrada do usuário, a tabela 17 é o *LinkADR Req*.

Tabela 17 – Estrutura *LinkADRReq*

Payload	CID	DataRate-TXPower	CHMask	Redundancy
Valor	03_h	*newDRnewTX*	$FF00_h$	00_h

Fonte: *LoRaWAN link layer specification*

Então, para montar essa *string* e depois converter pra hexadecimal, em Python usamos *f-strings*, que permite adicionar variáveis no meio do texto.

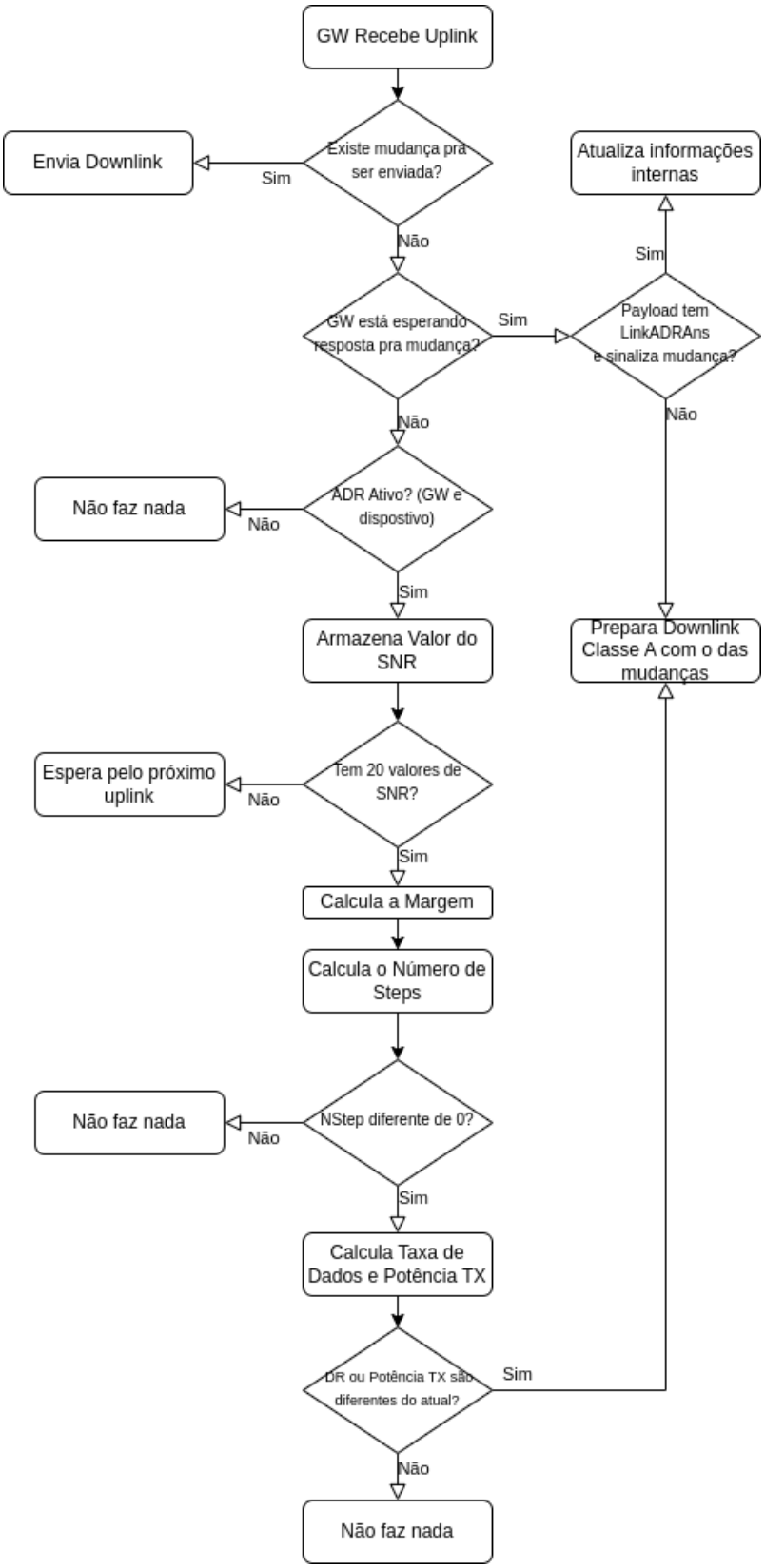
```
FOpts = f"03{newDR}{newTX}FF0000
```

Desta forma, basta incluir o *FOpts* dentro do *FHDR* na hora de montar a estrutura da mensagem.

Como nem todo *downlink* vai ter o *FOpts*, ele só será incluso quando tiver a solicitação de um comando *MAC*

4.2.3 Implementando o algoritmo na rotina no Gateway

Figura 12 – Fluxo de execuções e decisões para o cáculo do ADR



Fonte: Arquivo Pessoal

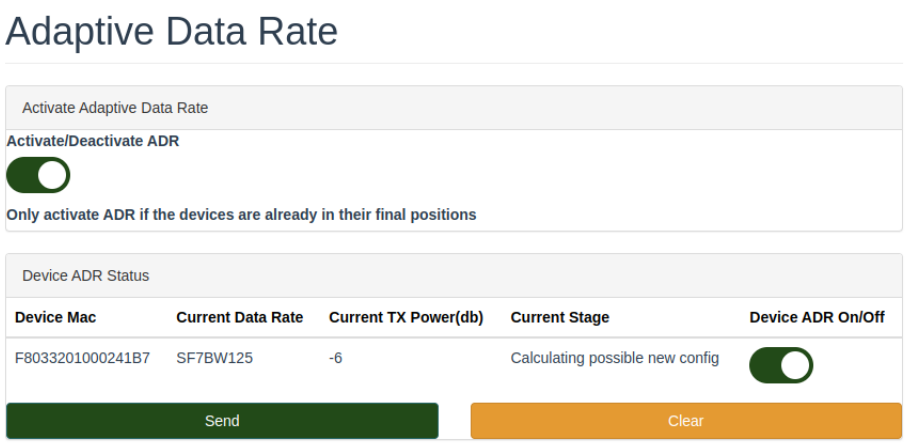
A Figura 12 ilustra como funciona a rotina de execuções do código para o cálculo do ADR. Onde a ação "Calcula taxa de dados e Potência TX"é o cálculo explicado do fluxograma da Figura 11 na Seção 3.5.3.

4.3 GERÊNCIA DA REDE

Essa etapa de implementação permite a gestão do *Adaptive Data Rate* (ADR) por meio de uma interface intuitiva. Através dessa interface (13), é possível visualizar a taxa de dados atual (ou fator de espalhamento) e a potência de rádio atual utilizada pelo dispositivo.

Além disso, o usuário tem a capacidade de realizar ações específicas, como ativar ou desativar o ADR globalmente, o que afeta todos os dispositivos registrados, ou ativar e desativar individualmente.

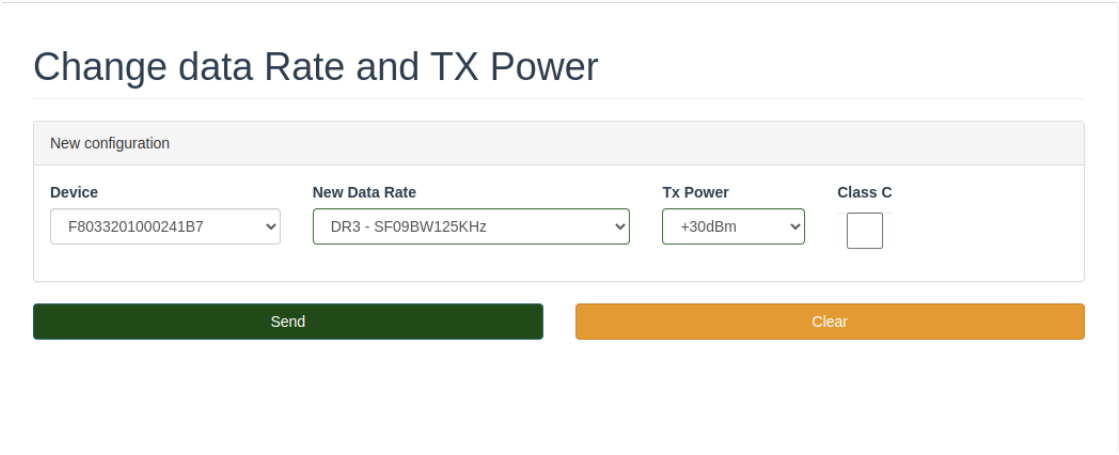
Figura 13 – Menu ADR



Fonte: Arquivo Pessoal

Além disso foi criado também o menu de configuração que permite (14) o ajuste manual da taxa de dados, conforme mencionado na Seção 3.5.2.

Figura 14 – Menu de troca manual



Fonte: Arquivo Pessoal

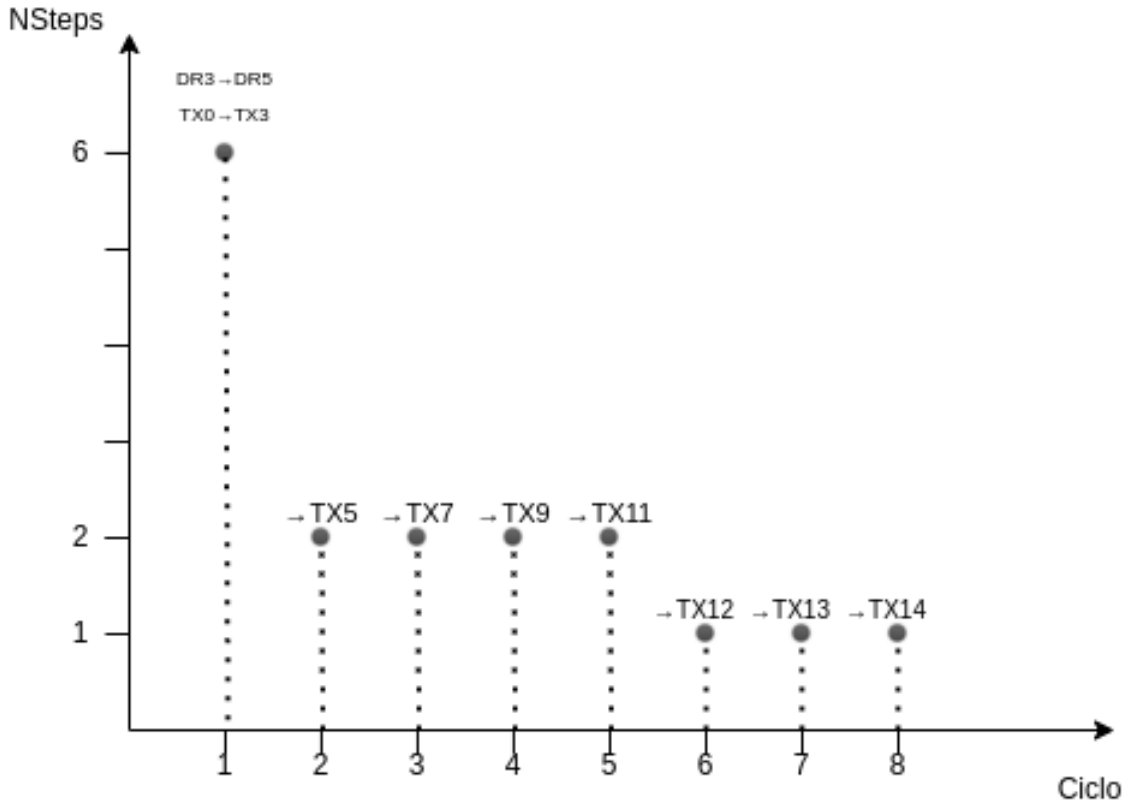
O usuário pode optar por mandar o comando como classe C, e caso o dispositivo esteja operando nesta classe, a troca será imediata.

4.4 AVALIAÇÃO DO ADR

4.4.1 Comportamento ao longo dos Ciclos

Inicialmente, o comportamento do algoritmo ao longo dos ciclos, onde um ciclo é definido pelo primeiro *uplink* até o envio do comando. O dispositivo final estava bem próximo ao *gateway* e o comportamento é ilustrado na figura 15.

Figura 15 – NSteps executados a cada ciclo



Fonte: Arquivo Pessoal

Sobre este comportamento podemos afirmar:

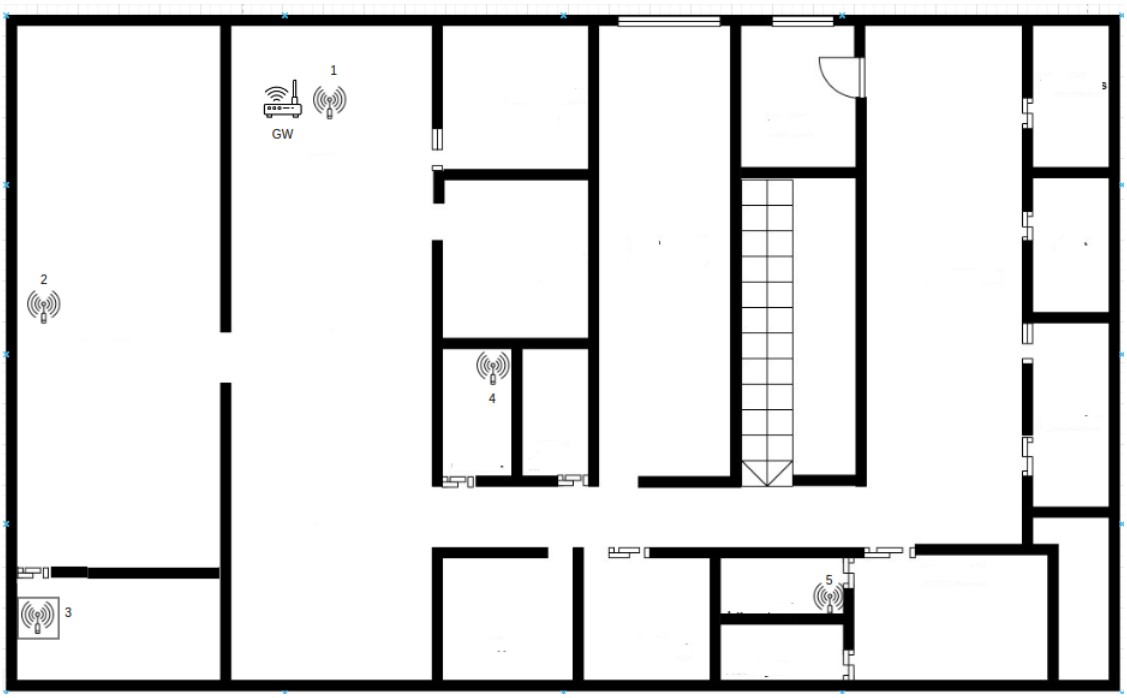
- O salto de 6 *Nsteps* no início se deve ao fato de que o SNR estava muito forte, e também porque a Equação 2 tem resultados maiores em taxas de dados mais baixas.
- A atenuação no fim se deve pelo fato de que a potência estava baixa, e consequentemente o SNR resultante também estava baixo, por isso que a dinâmica de trocas foi ficando mais devagar.

4.4.2 Cenário de Testes

Para testar o ADR, foram utilizados 8 dispositivos (de modelos e versões diferentes) cedidos pela Khomp, distribuídos estrategicamente no prédio de dois andares da empresa. Essa distribuição foi feita de forma a simular um cenário real de aplicação, com obstáculos físicos e diferentes distâncias em relação ao gateway. A taxa de dados inicial do dispositivo em sua maioria é a padrão de fábrica, e ela varia de acordo com o modelo e versão do dispositivo. Para este acelerar os testes, todos os dispositivos foram configurados para enviar mensagens a cada minuto. O objetivo dos testes foi validar a implementação do ADR em um cenário com múltiplos dispositivos, e observar

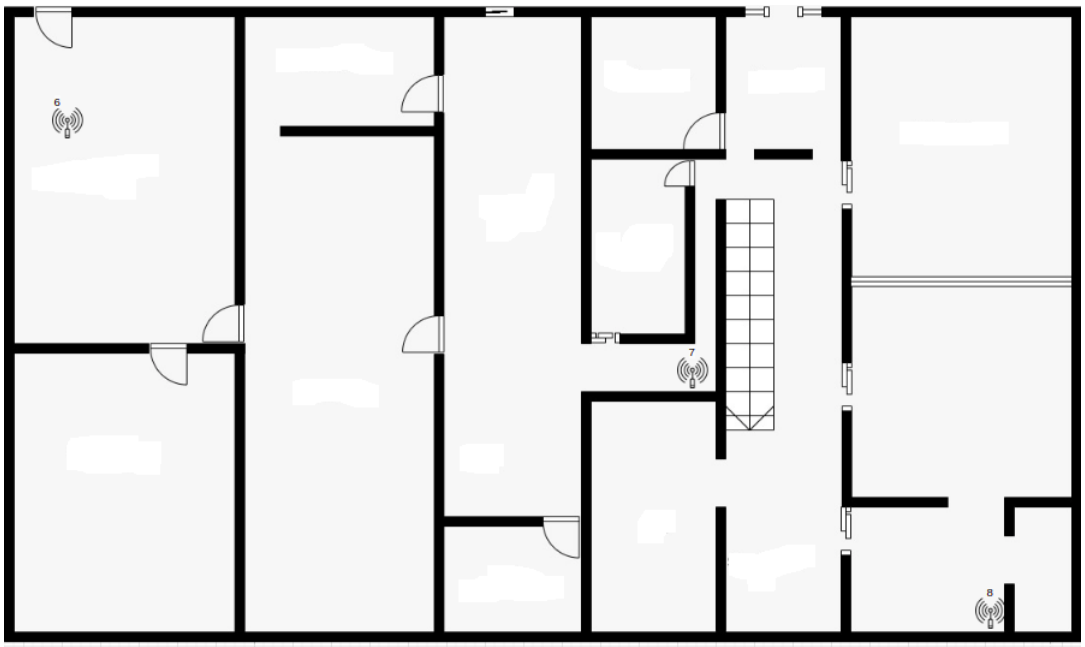
como os dispositivos convergem em relação à taxa de dados otimizada pelo algoritmo. Os dispositivos foram posicionados em locais apropriados para o funcionamento do ADR, ou seja muitas variações físicas do ambiente.

Figura 16 – Distribuição dos dispositivos no segundo andar



Fonte: Arquivo Pessoal

Figura 17 – Distribuição dos dispositivos no primeiro andar



Fonte: Arquivo Pessoal

As plantas das figuras 16 e 17 servem pra se ter uma ideia da posição dos dispositivos distribuídos em uma casa de tamanho grande, e a informação não é precisa. As portas e janelas externas ficaram fechadas durante o período de testes. O dispositivo final número 3 foi colocado dentro de uma caixa lacrada, simulando o cenário em que ele é colocado dentro de algum outro equipamento maior. Durante o período dos testes, todas as portas permaneceram fechadas.

4.4.3 Convergência dos Dispositivos

Depois de 4 horas de testes (ou seja, 12 ciclos de cálculo do ADR), a configuração com a qual cada dispositivo ficou ao final é mostrada na Tabela 18.

Tabela 18 – Teste com 8 dispositivos

Dispositivo	DR e SF - Inicial	DR e SF Final	Tx-final (dBm)
1	DR1 - SF11BW125	DR5 - SF07BW125	2
2	DR3 - SF09BW125	DR5 - SF07BW125	12
3	DR0 - SF12BW125	DR5 - SF07BW125	30
4	DR2 - SF10BW125	DR5 - SF07BW125	20
5	DR3 - SF09BW125	DR5 - SF07BW125	2
6	DR3 - SF09BW125	DR5 - SF07BW125	16
7	DR0 - SF12BW125	DR5 - SF07BW125	14
8	DR3 - SF09BW125	DR5 - SF07BW125	14

Fonte: Arquivo Pessoal

Como podemos observar, todos os dispositivos saíram do estado inicial. Mesmo os mais distantes do *gateway*, os dispositivos que estão no térreo, conseguiram atingir a maior taxa de dados. Um comportamento observado em quase todos os dispositivos, nas condições dos testadas, que ocorre um salto pra taxa de dados mais alta. Em questões matemáticas, isso ocorre porque quanto menor a taxa de dados, mais *Nsteps* para iteração tem, como visto na Equação 2. Depois disso, foi muito comum diminuir dBm por ciclo, até que o equilíbrio fosse atingido, permanecendo estável.

5 SUGESTÕES DE MELHORIAS

Este capítulo é um diagnóstico de funcionalidades que faltam para completar o projeto, ou comandos *MAC* que o autor julga serem importantes serem implementados.

5.1 TAXA DE PACOTES PERDIDOS E REENVIO DE MENSAGENS

A Semtech sugere implementar uma avaliação de quantidade de pacotes perdidos e a relação com a taxa de dados do dispositivo. O número de repetições pode ser definido no campo *Redundancy* do *LinkADRREQ*. Conforme é aumentada a taxa de dados, pode acontecer de aumentar os número de pacotes perdidos. Para evitar que a aplicação fique comprometida, o *network server* pode informar ao dispositivo que ele repita a mensagem duas ou três vezes. Vale ressaltar que em muitos casos, mesmo repetindo a mensagem, ainda há economia bateria, pois é compensado pelo aumento da taxa de dados e eventualmente da potência TX. Esse aspecto é imediatamente complementar ao *ADR*.

5.2 PERSISTÊNCIA DE DADOS RELACIONADOS AO ADR

Atualmente, todos os dados e variáveis de status do ADR são armazenados na memória volátil do gateway. No entanto, em caso de desligamento do sistema, esses dados são perdidos, e o processo de cálculo da taxa de dados é reiniciado, tendo como base apenas os dados enviados pelos dispositivos finais.

Uma solução para evitar essa perda de dados e garantir a continuidade do processo de ADR é adicionar a persistência dos dados em algum tipo de armazenamento, como um banco de dados. Dessa forma, mesmo em caso de desligamento ou reinicialização do gateway, os dados do ADR seriam preservados e o algoritmo poderia retomar o processo a partir do ponto em que parou.

Entre os dados a serem salvos, a lista de SNR (Relação Sinal-Ruído) é de extrema importância. Essa lista pode levar horas para atingir as 20 amostras necessárias para realizar os cálculos do ADR. Perder essas amostras antes de executar os cálculos adequados poderia resultar em insatisfação por parte dos clientes, que poderiam não entender por que o ADR não está funcionando corretamente.

Portanto, implementar a persistência de dados, especialmente a lista de SNR, garantiria a continuidade do processo do ADR, mesmo em situações de desligamento do gateway, evitando a perda de dados importantes e assegurando um funcionamento consistente e confiável do algoritmo.

5.3 ESTUDO DE CASO DOS DISPOSITIVOS KHOMP

A Khomp possui um extenso portfólio de sensores LoRaWAN, cada um com uma finalidade específica. Esses sensores podem ser implantados em locais com diferentes condições de ruído ou dificuldade de acesso, o que pode afetar a qualidade do sinal LoRa. Nesse contexto, uma proposta de melhoria consiste em realizar um estudo de caso para cada dispositivo vendido no portfólio da empresa, a fim de mapear as principais áreas de utilização de cada dispositivo.

Esse estudo de caso envolveria a coleta de dados sobre a localização dos dispositivos e a avaliação das condições de sinal em cada localidade. Com base nesses dados, seria possível identificar os locais mais propensos a ruídos ou com maior dificuldade de acesso, permitindo uma análise mais precisa das necessidades de cada dispositivo em termos de margens de sinal.

Com base nesse mapeamento, a Khomp poderia adotar $margin_{db}$ personalizadas para a Equação 2 de forma coerente com o uso específico de cada dispositivo. Isso permitiria ajustar os parâmetros de comunicação, como taxa de dados e potência de transmissão, de acordo com as condições reais de cada ambiente. Além disso, caso o dispositivo não seja da Khomp, poderia ser adotada uma margem genérica sem levar em conta as características do mesmo.

5.4 IDENTIFICAÇÃO AUTOMÁTICA DA CLASSE DO DISPOSITIVO

5.5 MUDAR OS PARÂMETROS DE ACORDO COM A REGIÃO

Conforme discutido ao longo desta monografia, é importante considerar os parâmetros regionais que definem as regras do LoRaWAN e do ADR. Portanto, toda a implementação feita foi para a região AU915-928. Nesse sentido, a melhoria proposta consiste em identificar a região em que o *gateway* está localizado e operando, e utilizar os parâmetros específicos daquela região.

A implementação dessa melhoria não é complexa e pode ser realizada por meio da criação de um dicionário que relacione cada região aos parâmetros correspondentes. Isso inclui considerar as potências de rádio disponíveis, as taxas de dados disponíveis, os fatores de espalhamento permitidos, entre outros parâmetros relevantes.

Dessa forma, ao adotar os parâmetros regionais adequados, é possível garantir a conformidade e compatibilidade com as regulamentações locais.

5.6 ALTERAR CONFIGURAÇÕES VIA MQTT

Atualmente, é possível ajustar diversas configurações do *gateway* de Telemetria por meio do MQTT. Com base nesse paradigma, propõe-se a criação de uma API que

permita configurar as mesmas opções disponíveis na interface mencionada na seção 4.3.

6 RESULTADOS

Com o desenvolvimento do algoritmo de taxa adaptativa de dados, agora os dispositivos podem operar em faixas ótimas para o consumo de energia. Como vimos na seção 4.4.3, mesmo distribuídos em uma área interna extensa, quase todos os dispositivos convergiram para o SF07BW125. Para mensagens de 23, 33 e 63 bytes, a Tabela 19 mostra o SF selecionado com os respectivos valores de tempo de modulação estimados.

Tabela 19 – Relação entre o Fator de Espalhamento, tamanho da mensagem e o tempo de modulação

SFBW | *Tamanho* / *Tempo*	23 bytes	33 bytes	63 bytes
SF07BW125	61.7ms	71.9ms	118ms
SF08BW125	113.2ms	133.6ms	215.6ms
SF09BW125	205.8ms	246.8ms	390.1ms
SF10BW125	370.7ms	452.6ms	698.4ms
SF11BW125	823.3ms	987.1ms	1478.7ms
SF12BW125	1482.8ms	1810.4ms	2793.5ms

Fonte: Arquivo Pessoal

Para todas as estimativas e testes foi utilizado um par de pilhas alcalinas, que juntas possuem tensão de 3.08V.

6.1 ECONOMIA DE BATERIA

A tabela 19 mostra a relação entre o tempo que uma mensagem leva para ser modulada de acordo com o fator de espalhamento. Em dispositivos LoRaWAN o consumo vem principalmente do período que o rádio está ligado modulando as mensagens. Utilizando a tabela apresentada anteriormente, é possível estimar quanto será a vida útil da bateria, utilizando relação aproximações criadas pelos engenheiros da Khomp. Num cenário onde o dispositivo envia 12 mensagens por hora, com a potência de transmissão padrão, teremos a seguinte relação, para as mensagens de 33bytes:

Tabela 20 – Relação SF vida útil da bateria considerando auto descarga

SFBW	duração da bateria (em dias)
SF07BW125	547
SF08BW125	460
SF09BW125	356
SF10BW125	252
SF11BW125	143
SF12BW125	82

Fonte: Arquivo Pessoal

Uma análise do impacto positivo da redução de *EIRP* foi feita. Para isso foi necessário medir, utilizando um multímetro de bancada, qual foi a corrente quando esse parâmetro foi alterado, e assim foi possível estimar a duração da bateria se analisando apenas a redução do *EIRP*. Considerando a taxa de dados máxima (SF07BW125), esta é a relação entre a duração e a redução desta potência:

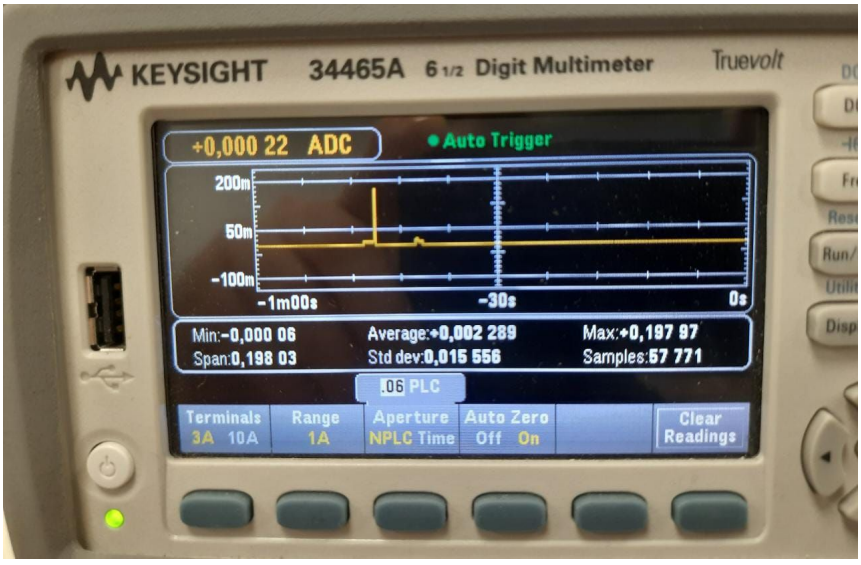
Tabela 21 – Relação entre EIRP e a duração da bateria

Potência	duração da bateria (em dias)
$EIRP_{max}$	547
2 dbm	673

Fonte: Arquivo Pessoal

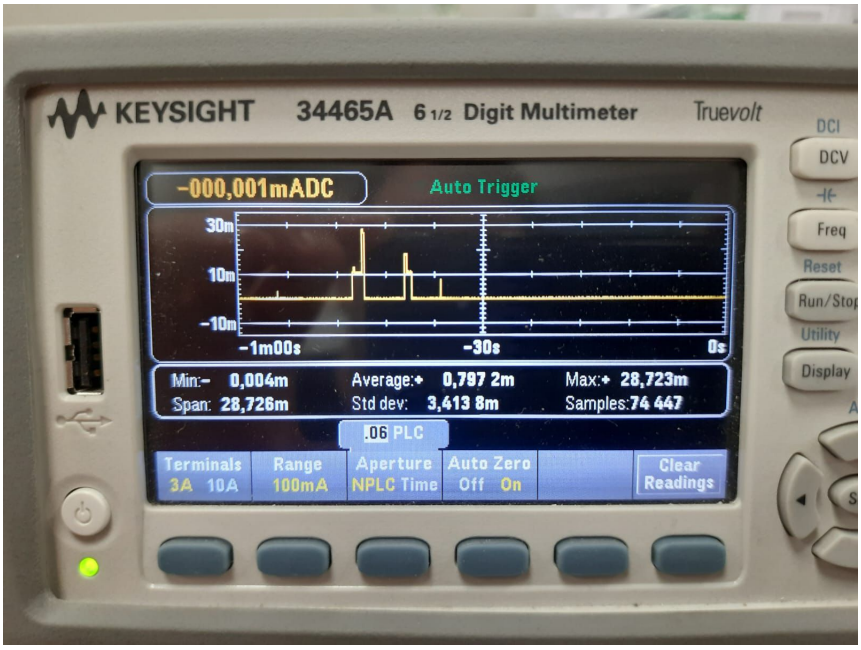
Foi possível constatar através das medições, que a potência RX, ou seja, a potência de sensibilidade, para o recebimento de *downlink*, não é alterada quando se mexe no valor de EIRP.

Figura 18 – Multímetro de bancada: Corrente de transmissão em $EIRP_{max}$



Fonte: Arquivo Pessoal

Figura 19 – Multímetro de bancada: Corrente de transmissão em 2 dbm



Fonte: Arquivo Pessoal

Foi feita apenas a análise para a taxa de dados máxima, pois a TX_{power} é alterada apenas manualmente ou após a taxa de dados chegar ao máximo.

Outro teste executado para validar o consumo de bateria, dois *endpoints* de mesmo modelo e versão, sob as mesmas condições de temperatura e umidade, pilhas novas (do tipo alcalina), um com SF07 e potência mínima e outro com SF10 e potência padrão ($EIRP_{max}$, após 4320 *uplinks* avaliou-se as tensões das pilhas de cada dispositivo:

Tabela 22 – Teste de descarga após 4320 mensagens

SFBW	Tensão Inicial	Tensão Final
SF10BW125	3.08V	2.98V
SF07BW125	3.08V	3.08V

Fonte: Arquivo Pessoal

As 4320 mensagens foram disparadas ao longo de 36h, para diminuir a influência de outros processos do dispositivo que possam consumir energia. E foi utilizado para medição da tensão a informação que o próprio dispositivo gera a respeito dessa medida.

Através desse teste de descarga, é possível observar quanto o rádio LoRa impacta na descarga das pilhas, e como é importante minimizar ao máximo esse consumo.

6.2 TEMPO DE USO DO CANAL

O tempo que o dispositivo leva para modular a mensagem, é o mesmo para o *gateway* demodular. Isso significa que quanto menor a taxa de dados, menos tempo o canal fica ocupado (para aquele DR), em um cenário com dezenas de dispositivos, todos de mesmo modelo e na mesma versão. Isso quer dizer que todos os dispositivos estão com a mesma taxa de dados, caso não tenha sido alterada. Nesse contexto, quanto maior for a taxa de dados, maiores são as chances de perda de pacote por canal ocupado. Em um intervalo de 24h com dispositivos enviando mensagens de 33 bytes a cada 5min, o tempo que cada fator de espalhamento ocupa do *gateway* é apresentado na tabela 23

Tabela 23 – Relação SF e ocupação do *gateway*

SFBW	Tempo de Canal ocupado (min)
SF07BW125	0.3
SF08BW125	0.6
SF09BW125	1.1
SF10BW125	2.2
SF11BW125	4.7
SF12BW125	8.6

Fonte: Arquivo Pessoal

É sabido que ao diminuir o fator de espalhamento, pior a sensibilidade do receptor, podendo causar mais pacotes perdidos (MONTAGNY, 2022). Mas, ao mesmo tempo, isso diminui as chances de o canal ficar ocupado, fazendo com que diminua o número de pacotes perdidos. Não é exatamente um equilíbrio, mas é algo a se levar em conta.

7 SUGESTÃO DE TRABALHOS FUTUROS E CONSIDERAÇÕES FINAIS

7.1 TRABALHOS FUTUROS

Este projeto utilizou o algoritmo sugerido pela Semtech, e que é amplamente utilizado no meio LoRaWAN. Mas existem outras propostas, e até mesmo variações do algoritmo aplicado no projeto, que podem ser implementados e testados no *gateway* Khomp de telemetria.

7.1.1 ADR+

O ADR+ (SLABICKI; PREMSANKAR; DI FRANCESCO, 2018) é uma proposta alternativa ao ADR convencional, que ao invés de utilizar o valor máximo de SNR, utiliza o valor médio. Os autores sugerem que podem ocorrer variações muito grandes de canal, e que pegar apenas o valor máximo ignoram essas variações. Isso faz com que o algoritmo se torne mais conservador do ponto de vista da dinâmica de aumento de taxa de dados, o que pode ocasionar um consumo de bateria.

No contexto do *gateway* bastam pouquíssimas alterações no código implementado nesse projeto para se ter o ADR+. Então, é fortemente sugerido que seja feito essa implementação e se observe na prática as variações, vantagens e desvantagens.

7.1.2 ADRx

O ADRx (JESUS, 2021) propõe uma $margin_{db}$ da equação 2 dinâmico, ou seja, é alterado em tempo de execução, e não como uma configuração inicial. Propõe-se que esse parâmetro seja recalculado a cada *uplink*, utilizando como referência a relação entre número de pacotes recebidos e transmitidos, encontrando o valor ótimo e individual pra cada dispositivo. Essa proposta de ADR explora um ponto pouco explorado no presente projeto, que é justamente o ajuste ótimo para $margin_{db}$. Portanto a sugestão seria avaliar utilizar o ADRx ao invés de buscar encontrar uma margem ótima para cada dispositivo.

7.2 CONSIDERAÇÕES FINAIS

A despesa para os clientes da Khomp na troca das baterias não se dá apenas pelo custo das mesmas, que considerando um cenário com dezenas de dispositivos já deixa de ser insignificante. A real despesa vem de deslocar alguém para o local para realizar essas trocas, e muitas vezes pode estar em outra cidade. Portanto, a equipe de *P&D* da empresa está sempre buscando amenizar o consumo de seus dispositivos, e isso muitas vezes se dá encontrando um equilíbrio entre o período de envio, reduzir ao mínimo necessário o tamanho das mensagens, e não ter como padrão o fator de

espalhamento alto em dispositivos onde não tem a necessidade. Acredita-se que a implementação feita neste trabalho era o que faltava por parte do *gateway* para tornar os dispositivos ainda mais eficientes e mais confiáveis com respeito à comunicação dos seus dados.

Todo o processo, no lado do *gateway*, referente à aplicação do ADR, escrita de código, implementação do algoritmo em questão foi feita por mim, tal qual a execução dos testes. É importante ressaltar que recebi ajuda do meu colega de equipe IoT e especialista em LoRaWAN. Ele é responsável pelo desenvolvimento e atualização dos *firmwares* dos dispositivos finais LoRa dentro da Khomp. Dentre o suporte prestado por ele estava a seleção de bibliografias, explicação de alguns comportamentos dos dispositivos, arquitetura LoRaWAN e elaboração de cenários de testes. Também recebi suporte de outras pessoas do time para realizar tarefas que a princípio eu não tinha familiaridade, como por exemplo manusear um multímetro de bancada. Além disso, o *gateway* em sua versão LoRaWAN já é um produto funcional consolidado no mercado, então nada além do mencionado no capítulo de desenvolvimento faz parte deste projeto.

Um dos principais desafios enfrentados durante o desenvolvimento deste projeto foi lidar com os comandos MAC, tanto para a leitura quanto para o envio de dados. Uma dificuldade adicional foi a falta de referências ou exemplos específicos de comandos para o dispositivo enviados a partir do *Gateway* Khomp de Telemetria. A parte de leitura de bytes e bits das mensagens foi especialmente desafiadora devido à ausência de uma documentação abrangente.

REFERÊNCIAS

ATO nº 14448. [*S.l.*: *s.n.*], 2017. https://www.anatel.gov.br/legislacao/atos-de-certificacao-de-produtos/2017/1139-ato-14448. Acesso em: 04/05/2023.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The Internet of Things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, 2010. DOI: 10.1016/j.comnet.2010.05.010.

BHAT, O. Introduction to IoT. **International Journal of Computer Sciences and Engineering**, IARJSET, v. 5, n. 1, 2018.

CHIRPSTACK - Open-source LoRaWAN Network Server. [*S.l.*]: ChirpStack. https://www.chirpstack.io/. Acesso em 19/06/2023.

CORDIS. **Up to 78 million batteries will be discarded daily by 2025, researchers warn**. [*S.l.*: *s.n.*], 2021. https://cordis.europa.eu/article/id/430457-up-to-78-million-batteries-will-be-discarded-daily-by-2025-researchers-warn.

GIT. About Git. [*S.l.*: *s.n.*]. https://git-scm.com/about. Accesso: 30/04/2023.

GUBBI, Jayavardhana; BUYYA, Rajkumar; MARUSIC, Slaven; PALANISWAMI, Marimuthu. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645–1660, 2013. DOI: 10.1016/j.future.2013.01.010.

IMPLEMENTING Adaptive Data Rate (ADR). [*S.l.*: *s.n.*]. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/implementing-adaptive-data-rate-adr/implementing-adaptive-data-rate/. Acesso em: 01/06/2023.

INSIGHTS, Transforma. **Current IoT Forecast Highlights**. [*S.l.*: *s.n.*]. https://transformainsights.com/research/forecast/highlights. Acesso em: 02/06/2023.

JESUS, Gabriel Germino Martins de. **Avaliação do desempenho do algoritmo adaptativo de taxa de dados de redes LoRaWAN com flexibilização da margem do enlace**. [*S.l.*: *s.n.*], 2021. [Trabalho de Conclusão de Curso].

KHOMP. Institucional. [*S.l.*: *s.n.*]. https://www.khomp.com/pt/institucional/. Acesso em: 01/06/2023.

KUFAKUNESU, Rachel; HANCKE, Gerhard P.; ABU-MAHFOUZ, Adnan M. A Survey on Adaptive Data Rate Optimization in LoRaWAN: Recent Solutions and Major Challenges. **Sensors**, v. 20, n. 18, p. 5044, 2020. DOI: 10.3390/s20185044.

LORA ALLIANCE. **LoRaWAN 1.0.4 Specifications**. [*S.l.*: *s.n.*], 2020. [S.l.]

LORA ALLIANCE. **RP002-1.0.0 LoRaWAN Regional Parameters**. [*S.l.*: *s.n.*], 2018. [S.l.]

LORA ALLIANCE. What is LoRaWAN? [*S.l.*: *s.n.*]. https://lora-alliance.org/about-lorawan/. Accesso: 15/05/2023.

LORAWAN World Record Broken Twice in Single Experiment. [*S.l.*: *s.n.*]. https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1. Acesso em: 15/06/2023.

MONTAGNY, Sylvain. **LoRa - LoRaWAN and Internet of Things for beginners**. 2. ed. Savoie: Savoie Mont Blanc University, 2022.

MUSEUM, COMPUTER HISTORY. **"Internet History: 1990s**. [*S.l.*: *s.n.*]. https://www.computerhistory.org/internethistory/1990s/. Acesso em: 30/05/2023".

NETWORK, The Things. **Adaptive Data Rate (ADR) - The Things Network Documentation**. [*S.l.*: *s.n.*], s.d. https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/. Acesso em: 01/05/2023.

PYTHON INSTITUTE. About Python. [*S.l.*: *s.n.*]. https://pythoninstitute.org/about-python. Accesso: 29/05/2023.

RAO, R.; PAPANDREOU-SUPPAPPOLA, A. Time Division Multiplexing (TDM). *In*: ENCYCLOPEDIA of Wireless and Mobile Communications. 2nd. [*S.l.*: *s.n.*], 2007. DOI: 10.1002/9780470061562.epr063.

RAZA, Usman; KULKARNI, Parag; SOORIYABANDARA, Mahesh; GAURA, Elena; KAWSAR, Fahim. Low Power Wide Area Networks: An Overview. **IEEE Communications Surveys & Tutorials**, v. 19, n. 2, p. 855–873, 2017. DOI: 10.1109/COMST.2016.2632421.

REYNDERS, Bram; MEERT, Wouter; POLLIN, Sofie. Range and coexistence analysis of long range unlicensed communication. *In*: IEEE. 2016 23rd International Conference on Telecommunications (ICT). Thessaloniki, Greece: [*s.n.*], 2016. P. 1–6. DOI: 10.1109/ICT.2016.7500415.

SEMTECH. **LoRaWAN - Simple Rate Adaptation Recommended Algorithm**. [*S.l.*: *s.n.*], 2016.

SLABICKI, Michal; PREMSANKAR, Gopika; DI FRANCESCO, Mario. Adaptive configuration of lora networks for dense IoT deployments. *In*: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. Taipei, Taiwan: [*s.n.*], 2018. P. 1–9. DOI: 10.1109/NOMS.2018.8406255.

TELECO. **Redes WiFi: Padrões 802.11 e Modulação**. [*S.l.*: *s.n.*]. Disponível em: https://www.teleco.com.br/tutoriais/tutorialredeswifi1/pagina_5.asp. Acesso em: (10/06/2023).

THE Things Network - Architecture. [*S.l.*]: The Things Network. https://www.thethingsnetwork.org/docs/lorawan/architecture/. Acesso em 19/06/2023.