PLANETA IFMG 2024
seminário de iniciação científica
ISSN 2558-6052

COMUNICAÇÃO VIA REDE LORA ENTRE GERADORES DISTRIBUÍDOS CONECTADOS A REDE MONOFÁSICA E AO CONTROLADOR CENTRAL, PARA UM CONTROLE SECUNDÁRIO EM UMA EM MICRORREDE TRIFÁSICA DE BAIXA TENSÃO.

Elberth Oliveira Lopes[(1)], Wanderson F. de Souza[(2)]

[(1)] Graduando em Engenharia de Controle e Automação - Instituto Federal de Minas Gerais (IFMG) - Campus Ibirité.
[(2)] Wanderson F. Souza - Professor orientador - IFMG - Campus Ibirité

RESUMO

Este projeto faz parte de um projeto maior, que visa desenvolver um sistema de comunicação entre geradores distribuídos conectados à rede elétrica, utilizando o protocolo LoRaWAN para comunicação com um controlador central via rede sem fio. Este trabalho se concentra na comunicação LoRa para o desenvolvimento de um controle terciário de uma microrrede monofásica de baixa tensão devido às vantagens atrativas que a LoRa pode oferecer com soluções de Internet das Coisas. Inicialmente, o estudo introduz o leitor ao protocolo LoRa, com um breve levantamento de informações sobre essa tecnologia e suas principais características. Em seguida, foram elaboradas baterias de testes, onde houve uma coleta de dados para análise e identificação dos melhores parâmetros LoRa (*spreading factor, coding rate* e TxPower) para operação. Os equipamentos utilizados para os testes com dispositivos diferentes (configuração híbrida) foram: um LoRa SX1276 conectado ao Arduino Mega 2560 em um nó, e um ESP32 equipado com LoRa SX1276 da Heltec no outro nó. Também foram utilizados dois Heltec ESP32 LoRa SX1276, um em cada nó, para testes com dispositivos semelhantes. O objetivo era utilizar ambas as configurações para montar uma rede com mais dispositivos, tendo o Arduino como facilitador para conexão com o controlador central. No entanto, observou-se que a configuração com dispositivos diferentes apresentou muitos desafios na fase de implementação para comunicação dos dispositivos, além de mais oscilações, como perda de pacotes e mensagens corrompidas, em comparação à configuração com dispositivos semelhantes.

Palavras-chave: Comunicação LoRa. ESP32. *Gateways*. LoRaWAN. Microrrede.

1 INTRODUÇÃO

Este projeto foca na aplicação do protocolo LoRaWAN para comunicação em sistemas de geração distribuída conectados à rede elétrica. O objetivo é estabelecer uma comunicação confiável entre geradores e um controlador central usando LoRaWAN como canal de transmissão utilizando as vantagens de comunicação dessa tecnologia de internet das coisas. De

PLANETA IFMG 2024
seminário de iniciação científica
ISSN 2558-6052

acordo com a (Semtech, 2024), LoRa é uma solução eficiente que consegue atender a demandas que não podem ser resolvidas adequadamente por tecnologias convencionais, como redes celulares, Wi-Fi e Bluetooth. Embora o projeto inicialmente incluísse uma análise mais ampla de controle e monitoramento de microrredes, contratempos na entrega de materiais levaram a uma mudança de foco. Atualmente, o projeto concentra-se exclusivamente na comunicação via LoRa, considerada crucial para a integração eficaz dos componentes do sistema.

A pesquisa avalia parâmetros críticos da comunicação, como taxa de transmissão, alcance, latência e perda de pacotes, visando otimizar o desempenho da rede. Utiliza-se uma infraestrutura baseada em dispositivos de baixo custo, como *Raspberry* Pi e módulos ESP32 com chip SX1276. Além disso, será realizada uma análise comparativa entre LoRa e outras tecnologias de comunicação sem fio, como *Zigbee, Bluetooth* e redes celulares (3G/4G), para avaliar a eficiência em aplicações de controle e monitoramento de microrredes.

2 Levantamento de Informações sobre a Tecnologia LoRa

O LoRa, de acordo com The *Things Network* (TTN, 2022), é uma técnica de comunicação sem fio derivada da tecnologia *Chirp Spread Spectrum* (CSS) que codifica informações em onda de rádio usando pulsos *Chirp* - semelhante à forma como golfinhos e morcegos se comunicam. A transmissão modulada LoRa é robusta contra distúrbios e pode ser recebida em grandes distâncias.

É o dispositivo ideal para aplicações que necessitam transportar um pequeno bloco de dados com pequenas taxas de bits. Opera em banda de sub-gigahertz de licenças livres, por exemplo, (915MHz, 818 MHz). Pode ser utilizado com frequência na faixa de GHz para atingir taxas de dados mais altas, ao custo da distância.

2.1 LoRaWAN

O LoRaWAN é um protocolo de camada *Media Access Control* (MAC) construído sobre a modulação LoRa. Segundo (TTN, 2022), é a camada de *software* que define como os dispositivos usam o *hardware* LoRa. Ele é adequado para transmitir cargas úteis de tamanho pequeno (como dados do sensor) em longas distâncias. De acordo com (Arduino, 2024), A modulação LoRa fornece uma faixa de comunicação significativamente maior com larguras de banda baixas do que outras tecnologias de transmissão de dados sem fio concorrentes, por

exemplo, WiFi, *Bluetooth* ou *ZigBee*. De acordo com *The Things Network* (s.d.), essas são as principais características LoRaWAN:

- **Bateria:** Possui um nível muito baixo de consumo de energia podendo durar até 10 anos com uma única bateria do tipo moeda.
- **Longa distância:** Os *gateways* LoRaWAN podem transmitir e receber sinais a uma distância de mais de 10 quilômetros em áreas rurais e até 3 quilômetros em áreas urbanas densas.
- **Geolocalização:** Uma rede LoRaWAN pode determinar a localização de dispositivos finais usando triangulação sem a necessidade de GPS. Um dispositivo final LoRa pode ser localizado se pelo menos três *gateways* captarem seu sinal.
- **Alta capacidade:** Os servidores de rede LoRaWAN lidam com milhões de mensagens de milhares de gateways.
- **Implantações públicas e privadas:** É fácil implantar redes LoRaWAN públicas e privadas usando o mesmo *hardware* (*gateways*, dispositivos finais, antenas) e software (encaminhadores de pacotes UDP, *software Basic Station*, pilhas LoRaWAN para dispositivos finais).
- **Segurança de ponta a ponta:** LoRaWAN garante comunicação segura entre o dispositivo final e o servidor de aplicativos usando criptografia AES-128.
- **Atualizações de *firmware* pelo ar:** Você pode atualizar remotamente o firmware (aplicativos e a pilha LoRaWAN) para um único dispositivo final ou grupo de dispositivos finais.
- ***Roaming:*** Os dispositivos finais LoRaWAN podem realizar transferências contínuas de uma rede para outra.
- **Baixo custo:** Infraestrutura mínima, nós finais de baixo custo e software de código aberto.
- **Programa de certificação:** O programa de certificação LoRa Alliance certifica dispositivos finais e fornece aos usuários finais a confiança de que os dispositivos são confiáveis e compatíveis com a especificação LoRaWAN.
- **Ecossistema:** LoRaWAN possui um ecossistema muito grande de fabricantes de dispositivos, fabricantes de *gateways*, fabricantes de antenas, provedores de serviços de rede e desenvolvedores de aplicativos.

2.2 Arquitetura

As redes LoRaWAN, de acordo com (TTN, 2022), são aplicadas em topologia **estrela das estrelas.** De maneira típica é composta por dispositivos finais, *gateways*, servidor de rede, servidores de aplicativos e *Join Server*. A seguir, uma breve descrição de cada um desses componentes.

- ***Gateways*** **-** recebem mensagens de dispositivos finais e as encaminham para o servidor de rede.
- **Servidor de rede** - um *software* executado em um servidor que gerencia toda a rede.
- **Servidores de aplicativos -** um *software* executado em um servidor responsável pelo processamento seguro dos dados do aplicativo.
- ***Join Server*** **-** um *software* executado em um servidor que processa mensagens de solicitação de ingresso enviadas por dispositivos finais.

Os dispositivos finais se comunicam com *gateways* próximos e cada *gateway* é conectado ao servidor de rede. As redes LoRaWAN usam um protocolo baseado em ALOHA, portanto, os dispositivos finais não precisam emparelhar com *gateways* específicos.

As mensagens enviadas dos dispositivos finais trafegam por todos os *gateways* dentro do alcance. Essas mensagens são recebidas pelo *network center*. Caso ele receba várias cópias da mesma mensagem ele **manterá uma única cópia devido ao seu processo de desduplicação.** Os *gateways* LoRaWAN podem ser categorizados em *gateways* internos (picocell) e externos (macrocell).

Gateways **Internos (Picocell):** São mais acessíveis e são melhores em oferecer cobertura em locais com muitas paredes e andares. Dependendo do ambiente físico interno, alguns *gateways* internos podem receber mensagens de sensores localizados a vários quilômetros de distância.

Gateways **Externos (macrocell):** Adequados para fornecer cobertura em áreas rurais e urbanas. Esses *gateways* podem ser montados em torres de celular, telhados de prédios muito altos, tubos de metal (mastros) etc. A sensibilidade do receptor do *gateway* externo é maior que o *gateway* interno.

***Join server*:** É responsável por auxiliar na ativação segura do dispositivo, armazenamento de chave raiz e geração de chaves de sessão.

2.3 Parâmetros Regionais

De acordo com (TTN, 2022), o LoRa opera em espectros de rádio livres, logo não é necessário pagar uma licença para sua utilização. Para isso utiliza baixas frequências de rádio com alcance maior, porém podem haver algumas restrições que são específicas do país.

Os parâmetros regionais incluem parâmetros da camada física, como planos de frequência (planos de canal), frequências de canais obrigatórias e taxas de dados para mensagens de solicitação de junção. Os Parâmetros Regionais também incluem parâmetros da camada LoRaWAN, como o tamanho máximo da carga útil.

De acordo com a (LORA ALLIANCE, 2021), no documento LoRaWAN® *Regional Parameters* (RP002-1.0.4) os parâmetros de comunicação LoRa para a região Brasil são:

- **Banda de Frequência padrão:** 915-928 MHz.
- ***Channel Plan*:** AU915-928.
- **Bandas e canais:** 902-907.5 MHz; 902-907.5 MHz; 902 -907.5 MHz.
- **Ciclo de trabalho:** É a quantidade máxima de tempo que um dispositivo pode gastar se comunicando, no caso do Brasil não tem limite.
- **Equação:** *Time-On-Air* = número de segundos por dia X ciclo de trabalho.

Frequências adicionais:

upstream: 64 (915.2 to 927.8 [+ by 0.2]) + 8 (915.9 to 927.1 [+ by 1.6]).

downstream: 8 (923.3 to 927.5 [+ by 0.6].

Taxas de dados:

É a taxa de dados é o número de bits que são transmitidos por unidade de tempo. Essa taxa de transmissão no LoRa depende do fator de espalhamento, largura de banda e taxa de codificação. Fatores de espelhamento mais altos causam taxas de bits mais baixas e fatores de espelhamento mais baixos causam taxas de bits mais altas.

Máximo EIRP/ERP:

+30 dBm. A potência isotrópica irradiada efetiva (EIRP) é a potência total irradiada por uma antena isotrópica em uma única direção. O ganho da antena é expresso em dBi para antenas isotrópicas.

Tabela 1 - Planos de frequência.

Uplink/ Downlink	**Canais**	**Faixa**	**Alcance da frequência**	**Largura de banda (BW)**	**Taxa de dados**
Uplink	64	0 - 63	915.2- 927.8 em incrementos de 200 kHz	125kHz	DR0 - DR5
Uplink	8	64 - 71	915.9-927.1 em incrementos de 1.6 kHz	500KHz	DR6 ou DR7LR-939 FHSS 1.523 MHz BW em DR7
Downlink	8	0 - 8	923.3-927.5 em incrementos de 600 kHz	500KHz	DR8-DR13

Fonte: The Things Network

Taxa de dados/*Data Rate*:

- [DR0 - DR6] e [DR8 - DR13]: são usados para modulação LoRa.
- DR6 = DR12
- DR0 – DR13: todas as taxas de dados são implementadas no dispositivo final.
- [DR0 – DR4] e [DR8 – DR13] – É o conjunto mínimo de taxas de dados necessário para obter a certificação LoRaWAN.
- [DR0 a DR13] - Todas as taxas de dados implementadas

Ao usar a ativação pelo ar (OTAA), o dispositivo final deve transmitir a mensagem de solicitação de junção em um canal selecionado aleatoriamente da seguinte maneira:

- 64 canais (cada um com largura de banda de 125 kHz) definidos usando DR2
- 8 canais (cada um com largura de banda de 500 kHz) definidos usando DR6

2.4 Tipos de mensagem

2.4.1 Mensagens de *uplink* e *downlink*

As mensagens LoRa podem ser divididas em mensagens de *uplink* e *downlink* com base na direção em que viajam. de acordo com (TTN, 2022), quando elas partem do dispositivo final para o *Network* e dispositivos seguintes ela recebe o nome de mensagem de *uplink*. A mensagem de *downlink* ocorre de maneira oposta sendo começando pelo Network Server e indo até os dispositivos finais.

2.4.2 Tipos de mensagem MAC e seus usos

As mensagens MAC em sua maioria são semelhantes, mas possuem diferenças na quantidade e tipos de mensagens de acordo com a versão em utilização do LoRaWAN. de acordo com (TTN, 2022), os tipos de mensagem que compõem o MAC são pedidos de adesão, Join-aceitar, solicitação de reingresso, mensagem de dado. A seguir, uma breve descrição de cada um desses itens:

- **Pedido de adesão:** A mensagem de solicitação de associação não é criptografada.
- ***Join-request*:** Possui criptografia, que varia de acordo com versão
- **Solicitação de reingresso:** A rede responde com uma mensagem de aceitação de ingresso.
- **Mensagem de dados:** Verificar quantos tipos de mensagens de dados a versão do LoRaWAN em utilização possui.

2.4.3 Envios de comando MAC e dados específicos do aplicativo

De acordo com (TTN, 2022), uma mensagem de dados em uma rede LoRaWAN pode transmitir simultaneamente comandos MAC e dados de aplicativos, mas em campos distintos, respeitando as restrições de estrutura e alocação para comandos e dados

Uma mensagem de dados pode conter qualquer sequência de comandos MAC. Uma mensagem de dados pode transportar comandos MAC e dados de aplicativos simultaneamente em campos separados. Os comandos MAC podem ser enviados no campo de opções de quadro (FOpts) ou no campo de carga útil do quadro (FRMPayload) de uma mensagem de dados, mas não em ambos simultaneamente. Os dados do aplicativo podem ser enviados no campo de carga

útil do quadro (FRMPayload) de uma mensagem de dados. O campo FRMPayload NÃO PODE conter comandos MAC e dados de aplicativos simultaneamente.

2.4.4 Enviando comandos MAC e dados específicos do aplicativo no campo FRMPayload

Conforme descrito por (TTN, 2022), o campo (FRMPayload) em uma mensagem LoRaWAN possui características específicas quanto ao seu uso para comandos MAC ou dados de aplicativos. Quando o (FRMPayload) tem qualquer um desses elementos, o campo FPort precisa estar presente, com atenção especial aos parâmetros definidos nele. Além disso, para garantir a segurança dos dados transmitidos, o campo (FRMPayload) deve ser criptografado antes que o Código de Integridade da Mensagem (MIC) seja calculado, assegurando a confidencialidade da mensagem.

Envios de comando MAC no Campo FOpts:

Os comandos MAC podem ser carregados no campo FOpts de uma mensagem de dados para envio. O comprimento total dos comandos MAC NÃO DEVE exceder 15 bytes. Podem ser criptografados ou não dependendo da versão.

Calculando o código de integridade da mensagem (MIC):

O *Message Integrity Code* (MIC) garante a integridade e a autenticidade de uma mensagem. de acordo com (TTN, 2022), o código de integridade da mensagem é calculado em todos os campos da mensagem e, em seguida, adicionado à própria mensagem. Os campos para calcular o MIC variam de acordo com a versão da rede LoRaWAN.

2.5 Segurança

De acordo com (TTN, 2022), o LoRaWAN utiliza diversas chaves de segurança, sendo elas NwkSKey, AppSKey e AppKey. Todas as chaves possuem um comprimento de 128 bits e utilizam o padrão AES-128. A seguir abordaremos os itens sobre as chaves de sessão e suas utilizações:

- **Chaves de Sessão:** Quando um dispositivo ingressa na rede (isso é chamado de associação ou ativação), uma chave de sessão de aplicativo AppSKey e uma chave de

sessão de rede NwkSKey são geradas. O NwkSKey é compartilhado com a rede, enquanto o AppSKey é mantido privado.

- **NwkSKey:** é utilizada para interação entre o nó e servidor de rede. Possui o objetivo verificar a integridade por uma verificação MIC. Essa MIC é semelhante a soma de verificação, entretanto impede a adulteração intencional da mensagem. Para isso, LoRaWAN usa AES-CMAC. No *back-end* do *The Things Network*, essa validação também é usada para mapear um endereço de dispositivo não exclusivo (DevAddr) para um único DevEUI e AppEUI.
- **AppSKey:** é uma chave utilizada para criptografar e descriptografar toda a informação. A informação fica criptografada entre o nó e o componente *Handler/Application Server* da *The Things Network*.

Essas duas chaves são exclusivas por dispositivo e aplicativo. Se forem ativadas dinamicamente por (OTAA) são geradas novas chaves, mas caso forem ativadas de forma estática as chaves permanecerão as mesmas até que o usuário faça uma alteração.

Chave do aplicativo

A AppKey é conhecida pelo dispositivo e aplicativo. De acordo com (TTN, 2022), os dispositivos ativados dinamicamente (OTAA) também utilizam essa chave para derivar duas chaves de sessão para ativação. A ativação terá um padrão ou poderá ser personalizada por dispositivo. A seguir, uma breve descrição dessas medidas de segurança.

Contadores de quadro: Na utilização no LoRAWAN trabalhamos com protocolo de rádio, por esse motivo um indivíduo pode vir capturar e armazenar uma mensagem, entretanto não poderá ser interpretada sem AppSKey e muito menos alterada sem a chave NwkSKey.

De acordo com (TTN, 2022), a segurança em redes LoRaWAN é mantida por meio de chaves específicas, como a AppKey, e pelo uso de contadores de quadro (FCntUp e FCntDown), que garantem a integridade e protegem contra ataques de repetição, especialmente em dispositivos OTAA e ABP.

Espectro de Propagação

A técnica utilizada pelo LoRa é o 'CHIRP' Pulso de Radar Comprimido de Alta Intensidade, de acordo com a (TTN, 2022), o requisito de design de fundo não é utilizado para

ocultar o sinal, mas por outros motivos, sendo alguns deles processamento, imunidade a interferência, compartilhamento de canal e resistência a reflexões de rádio (entre outros). Essas medidas são utilizadas como segurança contra condições operacionais não para resistência a vigilância.

2.6 Classes de Dispositivos

Os dispositivos LoRaWAN implementam três tipos de classes de dispositivos, sendo respectivamente A, B e C. De acordo com (TTN, 2022), todos os dispositivos possuem a classe A, enquanto as classes B e C são extensões da A. Todas as classes suportam comunicação bidirecional (*Uplink* e *Downlink*). A utilização das classes varia de acordo com a aplicação que será desenvolvida devido às características de cada uma delas. Abaixo estão as principais características e situações em que cada um dos tipos é mais adequado

Obs: Os dispositivos finais não podem enviar mensagens de *uplink* e receber mensagens de *downlink* ao mesmo tempo.

Características Classe A: Possui menor consumo de energia por ficar a maior parte do tempo em modo de hibernação; mantém longos intervalos de *uplinks*; têm alta latência de *downlink* (para receber um *downlink*, o dispositivo final deve enviar um *uplink*).

Características Classe B: Dispositivos finais de classe B têm latência menor do que os dispositivos finais de classe A, pois podem ser acessados em horários pré-configurados e não precisam enviar um *uplink* para receber um *downlink.* A duração da bateria é menor na Classe B do que na Classe A, porque o dispositivo passa mais tempo no modo ativo, durante *beacons* e *slots* de *ping*.

Obs: Os *beacons* são sincronizados com o tempo transmitidos pelo *gateway*, os dispositivos abrem janelas de recepção periodicamente.

Características Classe C: Os dispositivos da Classe C estendem a Classe A mantendo as janelas de recepção abertas (contínua), a menos que estejam transmitindo. Isso permite uma

comunicação de baixa latência, mas consome muitas vezes mais energia do que os dispositivos Classe A. Por este motivo são conectados à rede elétrica.

2.7 Ativação de dispositivo

A ativação de um dispositivo final ocorre após o seu registro na rede, isso é o que o possibilitará enviar e receber mensagens. A ativação pode ser realizada pelo método **OTAA** ou **ABP.** A seguir algumas explicações sobre o tema obtido de (TTN, 2022):

Ativação pelo ar / *Over-The-Air* (OTAA): É o método de ativação mais seguro e recomendado para dispositivos finais. Os dispositivos executam um procedimento de junção com a rede, durante o qual um endereço de dispositivo dinâmico é atribuído e as chaves de segurança são negociadas com o dispositivo.

Ativação por personalização (ABP): requer a codificação do endereço do dispositivo, bem como as chaves de segurança no dispositivo. O ABP é menos seguro que o OTAA e também tem a desvantagem de que os dispositivos não podem trocar de provedor de rede sem alterar manualmente as chaves no dispositivo.

2.8 Fatores de propagação

Fatores de propagação LoRa são características do ambiente que afetam a qualidade e o alcance do sinal LoRa. Com base nas informações (TTN, 2022), esses fatores determinam como o sinal se propaga e se comporta:

- Obstáculos: Edifícios, árvores, montanhas e outros obstáculos refletem, absorvem ou difratam o sinal, impactando sua intensidade e alcance.
- Condições climáticas: Chuva, neve, neblina e até mesmo folhas podem enfraquecer o sinal.
- Frequência de operação: O sinal LoRa se propaga de forma diferente dependendo da frequência escolhida.

  Impactos dos fatores de propagação:
- Alcance: Obstáculos e condições climáticas podem reduzir o alcance do sinal, limitando a distância entre os dispositivos.

- Qualidade do sinal: Fatores de propagação podem introduzir ruído e interferência no sinal, impactando a qualidade da comunicação.
- Consumo de energia: Se o sinal tem que viajar por um ambiente mais desafiador, os dispositivos podem precisar usar mais energia para transmitir e receber dados.

Entender os fatores de propagação é crucial para o planejamento e otimização de redes LoRa. Considerar esses fatores permite:

- Selecionar a melhor frequência de operação: A escolha da frequência pode minimizar o impacto de certos obstáculos.
- Posicionar os dispositivos estrategicamente para evitar áreas com muitos obstáculos.
- Ajustes como potência de saída e taxa de transmissão podem ajudar a compensar as condições desafiadoras.

Ao entender e gerenciar os fatores de propagação LoRa, você pode garantir uma comunicação confiável e eficiente em uma variedade de ambientes.

3 Desenvolvimento

Para os testes a seguir, foram enviadas 20 mensagens, nas quais foram observados o recebimento correto, a ordenação e a quantidade de confirmações de mensagens recebidas (ACKs). Para realização dos testes será mantida uma configuração equilibrada sendo **SF (7)**, **CR (5)** e **TXpower (14).** Realizando alterações apenas no parâmetro que está sendo analisado no momento.

Observações gerais

- No campo de “Mensagens Corrompidas” das tabelas, foram classificadas apenas as mensagens cujo ID não foi encontrado no receptor, seja por algum tipo de ruído ou mesmo após o estouro de tentativas sem obter nenhuma mensagem.
- Para as tabelas no documento a seguir a palavra “**Mensagens**” foi abreviada para a sigla “**MSGs**” para melhor visualização das tabelas de dados coletados.

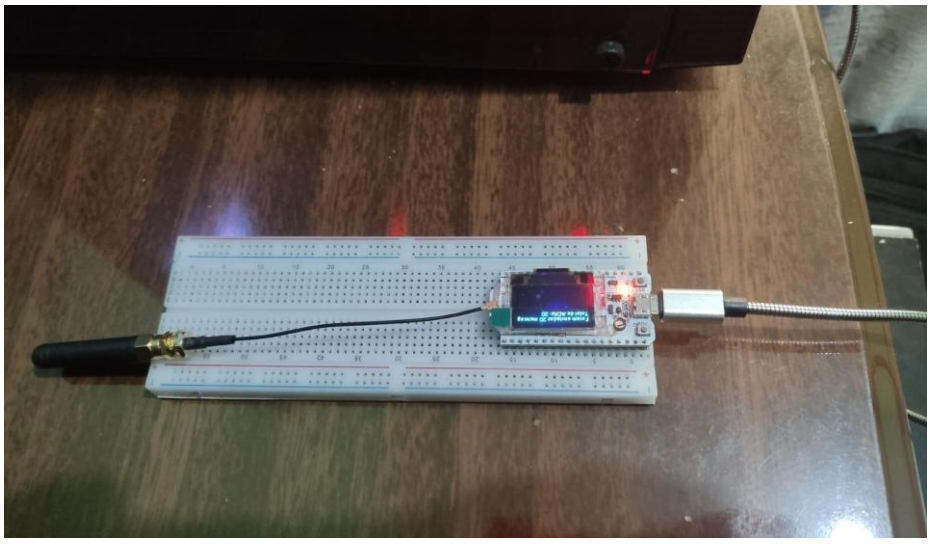
Durante o desenvolvimento do projeto, diversas análises de dados foram realizadas para avaliar o desempenho da comunicação via LoRaWAN. Para consultar os detalhes completos dessas análises, acesse o *link* no **Apêndice A**.

3.1 Bateria de Testes Com Dispositivos Diferentes (Teste Híbrido)

3.1.1 Descrição Sobre os Equipamentos e Testes

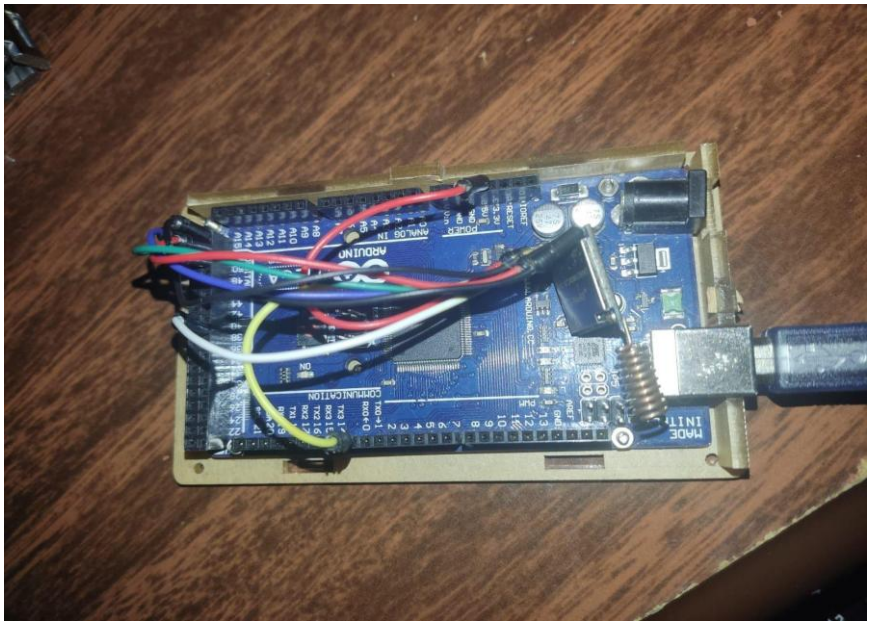
Foram utilizados dois dispositivos sendo um deles o **Heltec Wifi LoRa esp32 (V2)** com ***chip*** **LoRa SX1276**, e um **Arduino Mega 2560** equipado com um módulo LoRa conectado na configuração SPI. Implementamos a comunicação onde foram enviadas 20 mensagens, nas quais foram observados os seguintes parâmetros: recebimento correto, a ordenação e a quantidade de confirmações de mensagens recebidas (ACKs). Os testes foram realizados em um quarto onde os dispositivos estavam posicionados próximos um do outro a uma distância de aproximadamente 60 cm.

Wifi LoRa Esp32 (V2)



Quadro 1- Heltec ESP32 WIFI LoRa (V2) SX1276 utilizado na configuração de dispositivos diferentes.
Fonte: Elaborado pelo autor.

Arduino Mega 2560



Quadro 2- Módulo LoRa SX1276 conectado ao Arduino uno na configuração de dispositivos diferentes.
Fonte: Elaborado pelo autor.

Testes de *Spreading Factor* (SF): O parâmetro de *spreading factor* (fator de espalhamento) é essencial para otimizar nosso projeto de comunicação sem fio. O SF controla a taxa de *chirp*, afetando a velocidade e o alcance da transmissão de dados.

Tabela 2 - Testes de *Spreading Factor*.

SF	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompidas	MSGs Estouro de tentativas
7	19	20	1	0.05	1	0
8	12	20	8	0.4	0	0
9	8	13	5	0.6	0	0
10	4	3	0	0.8	0	1
11	1	1	0	1	0	0
12	1	0	0	0.95	1	1

Fonte: Elaborado pelo autor.

Observações:

- No SF (7), houve uma confirmação de mensagem muito rápida na mensagem 17. Sendo a primeira um ACK, que não obteve leitura pelo receptor, gerando assim uma perda de mensagem.

- No SF (8), apesar de receber apenas 12 mensagens, a sua confirmação de ACKs foi alta. Isso ocorre devido a várias confirmações logo nas primeiras mensagens, que não obtiveram leituras do receptor gerando a perda da mensagem.

- No SF (9), apesar de 8 mensagens recebidas, outras não obtiveram a leitura devido à velocidade com que a mensagem chegou ao receptor, mesmo o transmissor recebendo os ACKs.

Testes de *Coding Rate (CR)*: O *Coding Rate* é responsável por adicionar bits extras aos dados para corrigir erros. Quando aumentamos esse parâmetro, a confiabilidade da comunicação também aumenta, mas a velocidade de transmissão é reduzida. O contrário ocorre ao utilizarmos valores menores para esse parâmetro.

Tabela 3 - Testes de *Coding Rate* (CR).

CR	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompidas	MSGs Estouro de tentativas
5	19	19	1	0.05	0	0
6	16	16	4	0.2	0	4
7	17	17	3	0.15	0	3
8	15	15	4	0.25	1	1

Fonte: Elaborado pelo autor.

Observações:

- No CR (8), o número de ACKs confirmados e não computados não totaliza 20, pois uma mensagem atingiu o limite de 25 reenvios e não recebeu nenhum ACK.

Teste de configuração de parâmetros LoRa TXpower: Este parâmetro é responsável pela forma que o sinal de rádio será transmitido, impactando diretamente no alcance da comunicação e consumo de energia do dispositivo.

Tabela 3 - Teste de configuração de parâmetros LoRa TXpower.

TX_Power	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompida	MSGs Estouro de tentativas
2 dBm	20	2	0	0	0	18
3 dBm	20	1	0	0	0	19
4 dBm	18	3	1	0.1	1	16
5 dBm	20	4	0	0	0	16
6 dBm	20	6	0	0	0	14
7dBm	20	5	0	0	0	15
8 dBm	20	4	0	0	0	16
9 dBm	19	3	0	0.05	1	15
10 dBm	20	5	0	0	0	15
11 dBm	20	16	0	0	0	4
12 dBm	20	10	0	0	0	10
13 dBm	20	17	0	0.15	0	3
14 dBm	19	8	1	0.05	1	12
15 dBm	20	12	0	0	0	8
16 dBm	20	11	0	0.45	0	9
17 dBm	20	15	0	0	0	5
18 dBm	17	11	2	0.15	2	8
19 dBm	20	3	0	0	0	17
20 dBm	20	7	0	0	0	13

Fonte: Elaborado pelo autor.

Observação: Durante a realização dos testes que possui 20 itens, esquecemos de registrar os dados referente a linha de 1 dBm. Portanto, optamos por remover esses dados para não influenciar nas análises.

3.1.2 Resultado da Bateria de Teste Híbrido nos parâmetros LoRa

As análises a seguir têm como base os gráficos gerados a partir do banco de dados de uma bateria de testes nos três principais parâmetros LoRa: *Spreading Factor* (SF), *Coding Rate* (CR) e TX *Power* (tabelas 2, 3 e 4). Os dados completos da pesquisa estão disponíveis no Apêndice A. Foram importantes para a classificação de cores realizada abaixo, utilizando as seguintes variáveis: **Quantidade de mensagens recebidas, ACKs confirmados, ACKs não computados e mensagens que chegaram por estouro de tentativas.**

Legenda sobre o desempenho dos parâmetros:

- Itens marcados em verde obtiveram os melhores resultados
- Itens marcados em azul obtiveram resultados satisfatórios
- Itens marcados em vermelho obtiveram os piores resultados
- Itens marcados em amarelo obtiveram resultados mediano

Teste de configuração de parâmetros LoRa *Spread Factory* (SF): Valores mais altos são ideais para comunicações a curta distância, para as longas os menores são recomendados

SF = 12,
SF = 11,
SF = 10
SF = 9
SF = 8,
SF = 7
SF = 6

Teste de configuração de parâmetros LoRa Coding Rate (CR): Valores mais altos possuem menor taxa de bits e menor taxa de erros e o contrário se aplica para valores de CR menores. O mínimo utilizado é 4/5 e o máximo é 4/8.

CR = 5

CR = 6

CR = 7

CR = 8

Teste de configuração de parâmetros LoRa TXpower: O valor mínimo que pode se utilizar é 2 dBm indicado para testes e utilização em ambientes controlados. O valor máximo que pode ser utilizado é 20 dBm indicado para trabalhos em campo.

TXpower = 2

TXpower = 3

TXpower = 4

TXpower = 5

TXpower = 6

TXpower= 7

TXpower = 8

TXpower = 9

TXpower = 10

TXpower = 11

TXpower = 12

TXpower= 13

TXpower = 14

TXpower = 15

TXpower = 16

TXpower = 17

TXpower = 18

TXpower = 19

TXpower = 20

O código abaixo foi elaborado pelo autor**.**

Código transmissor com reenvios (Arduino com LoRa).

```
#include <SPI.h>
#include <LoRa.h>

#define SS 53 // cs pin
#define RST 9 // reset pin
#define DI0 38 // Di00
#define BAND 915E6

int messageCounter = 0; // Contador para mensagens enviadas
int ackCounter = 0; // Contador para ACKs recebidos
bool continueSending = true; // Variável para controlar a continuação do envio

void setup() {
 Serial.begin(115200); // Inicia a comunicação serial
 while (!Serial); // Espera a porta serial estar pronta

 Serial.println("LoRa Sender"); // Imprime uma mensagem de boas-vindas

 LoRa.setPins(SS, RST, DI0); // Configura os pinos para o módulo LoRa

 if (!LoRa.begin(BAND)) { // Tenta iniciar a comunicação LoRa na banda especificada
    Serial.println("Starting LoRa failed!"); // Imprime uma mensagem de erro se falhar
    while (1); // Loop infinito se a inicialização falhar
 }

 // Configurações adicionais do LoRa
 LoRa.setTxPower(14); // Define a potência de transmissão
 LoRa.setSpreadingFactor(7); // Define o Spreading Factor
 LoRa.setCodingRate4(5); // Define o Coding Rate
}

void loop() {
 if (!continueSending) {
   return; // Retorna imediatamente se continueSending for false
 }
```

```
 Serial.print("Sending packet: ");
 Serial.println(messageCounter); // Imprime o número da mensagem enviada

 // Envia uma mensagem com o contador incluído e espera por um ACK
 send_message_with_retry("Hello, world " + String(messageCounter), 25,
1000);
 messageCounter++; // Incrementa o contador após cada envio bem-sucedido

 delay(3000); // Espera 3 segundos antes do próximo envio

 // Verifica se o contador de mensagens atingiu 20
 if (messageCounter >= 20) {
   Serial.println("20 messages sent, stopping...");
   continueSending = false; // Define continueSending como false para
parar o loop
   printAckCount(); // Imprime o total de ACKs recebidos após o envio de
20 mensagens
 }
}

void send_message_with_retry(String message, int retries, int delay_ms)
{
 bool ackReceived = false;
 while (!ackReceived && retries > 0) {
    LoRa.beginPacket(); // Inicia um novo pacote LoRa
    LoRa.print(message); // Adiciona a mensagem ao pacote
    LoRa.endPacket(); // Finaliza o pacote

    unsigned long start_time = millis(); // Inicia o tempo de espera
    while (millis() - start_time < delay_ms) { // Espera pelo ACK
      int packetSize = LoRa.parsePacket(); // Verifica se há um pacote
recebido
      if (packetSize) {
        String ack = LoRa.readString(); // Lê o ACK recebido
        Serial.println("ACK received: " + ack); // Imprime o ACK recebido
        ackReceived = true; // Indica que um ACK foi recebido
        ackCounter++; // Incrementa o contador de ACKs recebidos
        break; // Sai do loop de espera
      }
    }
```

```
    if (!ackReceived) {
      Serial.println("ACK not received, retrying..."); // Imprime uma
mensagem se o ACK não for recebido
      retries--; // Decrementa o número de tentativas
    }
 }
 if (!ackReceived) {
   Serial.println("Failed after " + String(retries) + " retries."); //
Imprime uma mensagem se todas as tentativas falharem
 }
}

void printAckCount() {
 Serial.println("Total ACKs received: " + String(ackCounter)); // Imprime
o total de ACKs recebidos
}
```

Código receptor Heltec WiFi LoRa 32 (V2) com reenvio de ACK.

```
/*Potência de Transmissão
Mínimo: 0 dBm (para testes e em ambientes controlados)
Máximo: 20 dBm (para uso em campo, dependendo das regulamentações locais)
Recomendado: Comece com 14 dBm e ajuste conforme necessário.

Spreading Factor
Mínimo: 6 (para longas distâncias, menor taxa de erro)
Máximo: 12 (para curtas distâncias, maior taxa de erro)
Recomendado: Comece com 7 para um equilíbrio entre distância e taxa de
erro.

Coding Rate
Mínimo: 4/5 (maior taxa de erro, menor taxa de bits)
Máximo: 4/8 (menor taxa de erro, maior taxa de bits)
Recomendado: Comece com 5 para um equilíbrio entre confiabilidade e
eficiência de dados.*/

#include "heltec.h"
#define BAND 915E6 // Frequência

// Variável para contar o número de ACKs enviados
```

```
int ackCount = 0;

// Número de ACKs a serem enviados em sequência
int ackSequenceCount = 30;

// Intervalo de tempo entre os envíos de ACKs (em milissegundos)
unsigned long ackInterval = 100; // Exemplo: 100 milissegundos

void setup() {
 Heltec.begin(true   /*DisplayEnable   Enable*/,   true   /*Heltec.LoRa
Disable*/, true /*Serial Enable*/, true /*PABOOST Enable*/, BAND /*long
BAND*/);
 // Configuração adicional
 LoRa.setTxPower(14, PA_OUTPUT_PA_BOOST_PIN); // Potência de transmissão
(mesma do transmissor)
 LoRa.setSpreadingFactor(7); // Spreading Factor (mesmo do transmissor)
 LoRa.setCodingRate4(5); // Coding Rate (mesmo do transmissor)
}

void loop() {
 int packetSize = LoRa.parsePacket();
 if (packetSize) {
    Serial.print("Received packet '");
    while (LoRa.available()) {
      Serial.print((char)LoRa.read());
    }
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());

    // Envia múltiplos ACKs em sequência
    for (int i = 0; i < ackSequenceCount; i++) {
        // Envia um ACK de volta ao emissor
        LoRa.beginPacket();
        LoRa.print("ACK");
        LoRa.endPacket();

        // Aguarda um intervalo de tempo antes de enviar o próximo ACK
        delay(ackInterval);
    }
```

```
        // Incrementa o contador de ACKs
        ackCount++;

        // Verifica se o contador de ACKs atingiu o limite
        if (ackCount >= 10) {
            // Aqui você pode decidir o que fazer quando o limite é atingido
            // Por exemplo, parar de enviar ACKs ou reiniciar o contador
            // Para este exemplo, vamos reiniciar o contador
            ackCount = 0;
        }
    }
}
```

3.1.3 Observações sobre o código

ACKSequenceCount: É um parâmetro do código receptor responsável pela quantidade de reenvios de mensagens de confirmação em 100 ms que serão encaminhados ao transmissor. Com 30 a taxa de confirmação de mensagens é maior, mas se for utilizado um valor muito alto o receptor pode não conseguir processar a mensagem.

A bateria de testes que foi realizada tinha 20 reenvios até o estouro e passar para próxima mensagem. Os reenvios realizados pelo receptor tem 30 ACKs enviados em 500 ms, foi verificado após os testes que este valor obteve melhores resultados do que 15 ACKs testados anteriormente.

observação: parâmetro ``AckCount >=`` deve possuir o mesmo valor do ACKSequenceCount.

3.2.3 Conclusão do Teste Híbrido

Após a análise dos testes, foi possível compreender o funcionamento dos parâmetros LoRa e suas peculiaridades. A comunicação utilizando dois dispositivos diferentes apresentou um bom desempenho, entretanto, em algumas configurações, houve um aumento na taxa de erro, mensagens corrompidas e perda de mensagens.

Para mitigar esses problemas, foi implementado um sistema de reenvio e confirmação de mensagens (ACK), que, por sua vez, melhorou a estabilidade, diminuindo a taxa de erro e a perda de mensagens.2.2 - Bateria de Testes Com Dispositivos Iguais.

3.2 Bateria de Testes Com Dispositivos Iguais

Para realização dos testes será mantida uma configuração equilibrada sendo **SF (7)**, **CR (5)** e **TXpower (14).** Realizando alterações apenas no parâmetro que está sendo analisado no momento.

Observações Gerais

- No campo de "Mensagens Corrompidas" das tabelas, foram classificadas apenas as mensagens cujo ID não foi encontrado no receptor, seja por algum tipo de ruído ou mesmo após o estouro de tentativas sem obter nenhuma mensagem.
- Para as tabelas no documento a seguir a palavra "**Mensagens**" foi abreviada para a sigla "**MSGs**" para melhor visualização das tabelas de dados coletados.

Durante o desenvolvimento do projeto, diversas análises de dados foram realizadas para avaliar o desempenho da comunicação via LoRaWAN. Para consultar os detalhes completos dessas análises, acesse o *link* no **Apêndice B**.

3.2.1 Descrição sobre os Equipamentos e Testes

Para os testes a seguir, utilizamos dois dispositivos Wifi LoRa ESP32 (V2) da Helltec com *chip* LoRa SX1276, caracterizar tecnicamente melhor para realizar uma comunicação onde foram enviadas 20 mensagens, nas quais foram observados os seguintes parâmetros: recebimento correto, a ordenação e a quantidade de confirmações de mensagens recebidas (ACKs). Os testes foram realizados em um quarto onde os dispositivos estavam posicionados próximos um do outro a uma distância de aproximadamente 60 cm.

Configuração dos testes com os dois ESP32 da Heltec.

Quadro 1- Heltec ESP32 WIFI LoRa (V2) sx1276 utilizado na configuração de dispositivos iguais.
Fonte: Elaborado pelo Autor.

Testes de *Spreading Factor*: O parâmetro de *spreading factor* (fator de espalhamento) é essencial para otimizar nosso projeto de comunicação sem fio. O *spreading factor* controla a taxa de *chirp*, afetando a velocidade e o alcance da transmissão de dados.

O objetivo deste teste é verificar o comportamento da comunicação LoRa dos dispositivos e identificar em quais configurações obtivemos os melhores resultados. Isso permitirá implementar a organização dos dispositivos em campo de maneira mais dinâmica, garantindo que o alcance e a velocidade de resposta sejam adequados para uma aplicação dedicada.

Tabela 5 - Testes de *Spreading Factor*.

SF	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompidas	MSGs Estouro de tentativas
6	0	0	0	1	20	0
7	20	20	0	0	0	0
8	20	20	0	0	0	0
9	20	20	0	0	0	0
10	20	20	0	0	0	0
11	20	20	0	0	0	0

12	20	20	0	0	0	0

Fonte: Elaborado pelo autor.

Testes de *Coding Rate* (CR): O *Coding Rate* é responsável por adicionar bits extras aos dados para corrigir erros. Quando aumentamos esse parâmetro, a confiabilidade da comunicação também aumenta, mas a velocidade de transmissão é reduzida. O contrário ocorre ao utilizarmos valores menores para esse parâmetro.

Tabela 6 - Testes de *Coding Rate* (CR).

CR	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompida	MSGs Estouro de tentativas
5	20	20	0	0	0	0
6	20	20	0	0	0	0
7	20	20	0	0	0	0
8	20	20	0	0	0	0

Fonte: Elaborado pelo autor.

Teste de *TX_Power:* A potência de transmissão (Tx Power) determina a força com que o sinal de rádio é transmitido. Esse parâmetro pode ser ajustado em diferentes níveis, desde o mínimo até o máximo, com impacto direto no alcance da comunicação e no consumo de energia do dispositivo.

Tabela 7 - Teste de configuração de parâmetros LoRa TXpower.

TX_Power	MSGs recebidas	ACKs confirmados	ACKs não computados	Erro em %	MSGs corrompida	MSGs Estouro de tentativas
2 dBm	20	20	0	0	0	0
3 dBm	20	20	0	0	0	0
4 dBm	20	20	0	0	0	0
5 dBm	20	20	0	0	0	0
6 dBm	20	20	0	0	0	0
7dBm	20	20	0	0	0	0
8 dBm	20	20	0	0	0	0
9 dBm	20	20	0	0	0	0

10 dBm	20	20	0	0	0	0
11 dBm	20	20	0	0	0	0
12 dBm	20	20	0	0	0	0
13 dBm	20	20	0	0	0	0
14 dBm	20	20	0	0	0	0
15 dBm	20	20	0	0	0	0
16 dBm	20	20	0	0	0	0
17 dBm	20	20	0	0	0	0
18 dBm	20	20	0	0	0	0
19 dBm	20	20	0	0	0	0
20 dBm	20	20	0	0	0	0

Fonte: Elaborado pelo autor.

Observação: Durante a realização dos testes que possui 20 itens, esquecemos de registrar os dados referente a linha de 1 dBm. Portanto, optamos por remover esses dados para não influenciar nas análises.

3.2.2 Resultado da Bateria de testes nos parâmetros LoRa.

As análises a seguir têm como base os gráficos gerados a partir do banco de dados de uma bateria de testes nos três principais parâmetros LoRa: *Spreading Factor* (SF), *Coding Rate* (CR) e TX Power (tabelas 5, 6 e 7). Os dados completos da pesquisa estão disponíveis no Apêndice B. Foram importantes para a classificação de cores realizada abaixo, utilizando as seguintes variáveis: **Quantidade de mensagens recebidas, ACKs confirmados, ACKs não computados e mensagens que chegaram por estouro de tentativas.**

Legenda sobre o desempenho dos parâmetros:

- Itens marcados em verde obtiveram os melhores resultados
- Itens marcados em azul obtiveram resultados satisfatórios
- Itens marcados em vermelho obtiveram os piores resultados
- Itens marcados em amarelo obtiveram resultados mediano

Teste de configuração de parâmetros LoRa *Spread Factory* (SF): Valores mais altos são ideais para comunicações a curta distância, para as longas os menores são recomendados.

SF = 12,

SF = 11,

SF = 10

SF = 9

SF = 8,

SF = 7

SF = 6

Teste de configuração de parâmetros LoRa Coding Rate (CR): Valores mais altos possuem menor taxa de bits e menor taxa de erros e o contrário se aplica para valores de CR menores. O mínimo utilizado é 4/5 e o máximo é 4/8.

CR = 5

CR = 6

CR = 7

CR = 8

Teste de configuração de parâmetros LoRa TXpower: Este parâmetro é responsável pela forma que o sinal de rádio será transmitido, impactando diretamente no alcance da comunicação e consumo de energia do dispositivo rádio. O valor mínimo que pode se utilizar é 2dBm indicado para testes e utilização em ambientes controlados. O valor máximo que pode ser utilizado é 20dBm mais indicado para trabalhos em campo. Qual a referência destes números? Precisaremos de colocar referências nestas citações.

TX_power = 2

TX_power = 3

TX_power = 4

TX_power = 5

TX_power = 6

TX_power= 7

TX_power = 8

TX_power = 9

TX_power = 10

TX_power = 11

TX_power = 12

TX_power = 13

TX_power = 14

TX_power = 15

TX_power = 16

TX_power = 17

TX_power = 18

TX_power = 19

TX_power = 20

Os códigos abaixo foram elaborados pelo autor**.**

Código transmissor com reenvios (Heltec Esp32).

```
#include "heltec.h"

// Parâmetros da transmissão
int messageCount = 0;  // Contador de mensagens enviadas
int ackCount = 0; // Contador de ACKs recebidos
const int maxMessages = 20; // Número máximo de mensagens a serem enviadas
int messageID = 0; // Identificador único para cada mensagem
const int maxRetries = 10; // Número máximo de tentativas de reenvio

void setup() {
  // Inicialização da comunicação serial para depuração
  Serial.begin(115200);
  while (!Serial); // Aguarda a inicialização da comunicação serial

  // Inicialização do módulo Heltec com a configuração do LoRa
  Heltec.begin(true, true, true, true, 915E6);

  // Configuração adicional do LoRa
  LoRa.setTxPower(14, PA_OUTPUT_PA_BOOST_PIN); // Define a potência de
transmissão (14 dBm)
```

```
  LoRa.setSpreadingFactor(7); // Define o Spreading Factor (7)
  LoRa.setCodingRate4(5); // Define o Coding Rate (4/5)

  // Mensagem de inicialização bem-sucedida
  Serial.println("LoRa Initialization OK!");
}

void loop() {
  // Verifica se o número de mensagens enviadas é menor que o máximo permitido
  if (messageCount < maxMessages) {
    bool ackReceived = false; // Flag para verificar se o ACK foi recebido
    int retries = 0; // Contador de tentativas de reenvio

    // Tenta enviar a mensagem até receber o ACK ou atingir o número máximo de tentativas
    while (!ackReceived && retries < maxRetries) {
      // Criação da mensagem com identificador único
      String message = "Hello World ID " + String(messageID);

      // Envio da mensagem via LoRa
      LoRa.beginPacket();
      LoRa.print(message);
      LoRa.endPacket();

      // Exibição da mensagem no monitor serial e no display OLED
      Serial.println("Sent: " + message);
      Heltec.display->clear();
      Heltec.display->drawString(0, 0, "Sent: " + message);
      Heltec.display->display();

      // Aguarda ACK do receptor
      ackReceived = waitForACK();

      // Caso não receba ACK, incrementa o contador de tentativas
      if (!ackReceived) {
        Serial.println("Retrying...");
        retries++;
      }
    }
```

```
    // Se o ACK foi recebido, incrementa o contador de ACKs
    if (ackReceived) {
      ackCount++;
    } else {
      // Mensagem de falha ao não receber ACK após o número máximo de
tentativas
      Serial.println("Failed to receive ACK after " + String(maxRetries)
+ " attempts.");
    }

    // Incrementação dos contadores de mensagens e identificadores
    messageID++;
    messageCount++;

    // Espera de 1 segundo antes do próximo envio
    delay(1000);
  } else {
    // Mensagem final após enviar todas as mensagens
    Serial.print("Foram enviadas ");
    Serial.print(messageCount);
    Serial.println(" mensagens.");

    Serial.print("Total de ACKs recebidos: ");
    Serial.println(ackCount);

    // Exibição do resumo no display OLED
    Heltec.display->clear();
    Heltec.display->drawString(0,    0,    "Foram    enviadas    "    +
String(messageCount) + " mensagens.");
    Heltec.display->drawString(0,    10,    "Total    de    ACKs:    "    +
String(ackCount));
    Heltec.display->display();

    while (1); // Parar o loop após enviar todas as mensagens
  }
}

// Função para aguardar o ACK do receptor
bool waitForACK() {
```

```
  bool ackReceived = false; // Flag para verificar se o ACK foi recebido
  unsigned long startTime = millis(); // Tempo de início para controle de timeout

  // Loop para aguardar até 3 segundos pelo ACK
  while (!ackReceived && millis() - startTime < 3000) { // Aguarda até 3 segundos pelo ACK
    int packetSize = LoRa.parsePacket(); // Verifica se há pacotes recebidos
    if (packetSize) {
      String receivedMessage = ""; // Inicializa a string para armazenar a mensagem recebida
      while (LoRa.available()) { // Lê os dados do pacote
        receivedMessage += (char)LoRa.read();
      }

      // Verifica se a mensagem recebida é um ACK
      if (receivedMessage == "ACK") {
        Serial.println("Received ACK from receiver.");
        ackReceived = true; // Atualiza a flag para indicar que o ACK foi recebido
      }
    }
  }

  // Mensagem de timeout se o ACK não for recebido
  if (!ackReceived) {
    Serial.println("Timeout: No ACK received.");
  }

  return ackReceived; // Retorna o status do recebimento do ACK
}
```

Código receptor Heltec WiFi LoRa 32 (V2) com reenvio de ACK.

```
#include "heltec.h"

// Variáveis para contar o número de mensagens recebidas e ACKs enviados
int receivedCount = 0; // Contador de mensagens recebidas
bool transmissionStopped = false; // Flag para indicar se a transmissão foi parada
```

```
void setup() {
  // Inicialização da comunicação serial para depuração
  Serial.begin(115200);
  while (!Serial); // Aguarda a inicialização da comunicação serial


  // Inicialização do módulo Heltec com a configuração do LoRa
  Heltec.begin(true, true, true, true, 915E6);


  // Configuração adicional do LoRa
  LoRa.setTxPower(14, PA_OUTPUT_PA_BOOST_PIN); // Define a potência de
transmissão (14 dBm)
  LoRa.setSpreadingFactor(7); // Define o Spreading Factor (7)
  LoRa.setCodingRate4(5); // Define o Coding Rate (4/5)


  // Mensagem de inicialização bem-sucedida
  Serial.println("LoRa Initialization OK!");
}


void loop() {
  // Verifica se a transmissão ainda não foi parada
  if (!transmissionStopped) {
    // Tentativa de receber um pacote
    int packetSize = LoRa.parsePacket(); // Verifica se há pacotes
recebidos
    if (packetSize) {
      // Recebendo a mensagem
      String receivedMessage = ""; // Inicializa a string para
armazenar a mensagem recebida
      while (LoRa.available()) { // Lê os dados do pacote
        receivedMessage += (char)LoRa.read();
      }


      // Obtendo intensidade do sinal (RSSI) e qualidade do sinal (SNR)
      int rssi = LoRa.packetRssi(); // RSSI do pacote recebido
```

```
        float snr = LoRa.packetSnr(); // SNR do pacote recebido


        // Exibindo a mensagem recebida formatada no monitor serial
        Serial.print("Received message: ");
        Serial.println(receivedMessage); // Exibe a mensagem recebida


        // Exibindo informações adicionais no monitor serial
        Serial.print("RSSI: ");
        Serial.print(rssi);
        Serial.print(" dBm; SNR: ");
        Serial.print(snr);
        Serial.println(" dB");
        Serial.println(); // Salta uma linha em branco


        // Exibindo a mensagem recebida no display OLED (opcional)
        Heltec.display->clear();
        Heltec.display->drawString(0, 0, "Received: " + receivedMessage);
        Heltec.display->display();


        // Envia ACK para o transmissor
        sendACK();


        // Incrementando o contador de mensagens recebidas
        receivedCount++;


        // Verifica se a mensagem recebida indica o fim da transmissão
        if (receivedMessage.indexOf("Transmission Stopped") != -1) {
          transmissionStopped = true; // Atualiza a flag para indicar que
a transmissão foi parada
        }
      }
    } else {
      // Mensagem final após parar a transmissão
      Serial.print("Total messages received: ");
```

```
    Serial.println(receivedCount);


    // Exibição do resumo no display OLED
    Heltec.display->clear();
    Heltec.display->drawString(0, 0, "Total received: " +
String(receivedCount) + " msgs");
    Heltec.display->display();


    while (1); // Parar o loop após a transmissão parar
  }
}


// Função para enviar um ACK de volta ao transmissor
void sendACK() {
  LoRa.beginPacket(); // Inicia a construção do pacote
  LoRa.print("ACK"); // Adiciona a mensagem "ACK" ao pacote
  LoRa.endPacket(); // Finaliza o envio do pacote


  // Mensagem de confirmação do envio do ACK no monitor serial
  Serial.println("ACK sent to transmitter.");
}
```

Observações sobre o código: A bateria de testes realizada tinha 10 reenvios até o estouro de tentativas e passar para próxima mensagem.

3.2.3 Conclusão Teste Esp32

Após a análise dos testes, foi possível compreender o funcionamento dos parâmetros LoRa e suas peculiaridades. Os ótimos resultados obtidos estão relacionados à utilização de dispositivos idênticos, que proporcionam uma comunicação mais estável e ao mecanismo de reenvio de mensagens, que permitiu uma comunicação ainda mais confiável do que a apresentada no teste híbrido.

4 Conclusão Final dos Testes

Apesar da comunicação híbrida funcionar, ela apresenta uma inconsistência no recebimento das mensagens, o que torna a comunicação instável e indesejável para aplicações futuras, que pode ter acontecido muito devido a qualidade dos módulos adquiridos. A comunicação entre dispositivos idênticos mostrou-se bem mais confiável tanto na transmissão quanto no recebimento das mensagens, obtendo resultados satisfatórios nos testes.

Para eliminar dúvidas a respeito de sua confiabilidade, submetemos essa solução a um teste de estresse de longa duração, onde podem ser observados os testes e análises disponíveis no Apêndice C, cujo resultado foi muito semelhante ao do teste em menor escala. Isso comprova que essa solução é mais eficiente e adequada para utilização em aplicações, ao contrário da comunicação híbrida.

Como mencionado anteriormente, devido a contratempos burocráticos envolvendo a compra de equipamentos, houve um atraso no projeto, sendo necessário mudar os planos. A ideia inicial de desenvolver a rede LoRaWAN para controle da microrrede com controlador central não foi concluída. Entretanto, dadas as circunstâncias, podemos focar na parte de testes de comunicação e realizar a descoberta sobre como a configuração híbrida e seu funcionamento na comunicação LoRa. O estudo realizado pode ser utilizado como base para a continuação do projeto ou servir como base para estudos de projetos futuros envolvendo o tema.

5 ELEMENTOS PÓS TEXTUAIS

APÊNDICE A – Bateria e análise dos testes utilizando dispositivos diferentes Bateria_testes_parametros_LoRa.ipynb

APÊNDICE B – Bateria e análise dos testes utilizando dispositivos iguais Bateria_de_Testes_Heltec_Config_LoRa.ipynb

APÊNDICE C – Bateria e análise dos testes de estresse utilizando dispositivos iguais Longa_Bateria_de_Testes_2_Heltec_Config_LoRa.ipynb

REFERÊNCIAS

ARDUINO. *LoRaWAN 101*. 2024. Disponível em: https://docs.arduino.cc/learn/communication/lorawan-101/. Acesso em: 09 set. 2024.

LORA ALLIANCE. *LoRaWAN® Regional Parameters*. RP002-1.0.4. 2021. Disponível em:https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/. Acesso em: 09 set. 2024. p. 47-49.

SEMTECH. *LoRa*. 2024. Disponível em: https://www.semtech.com/lora. Acesso em: 09 set. 2024.

THE THINGS NETWORK. LoRaWAN. 2022. Disponível em: https://www.thethingsnetwork.org/docs/lorawan/. Acesso em: 09 out. 2023.