INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

IGLÉSIO OLIVEIRA DE CARVALHO JÚNIOR

ARQUITETURA DE GATEWAY LORA INTEGRADA AO ROS 2 UTILIZANDO FREERTOS

TERESINA, PIAUÍ
2026

IGLÉSIO OLIVEIRA DE CARVALHO JÚNIOR

ARQUITETURA DE GATEWAY LORA INTEGRADA AO ROS 2 UTILIZANDO FREERTOS

Trabalho de Conclusão de Curso (Artigo Científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Orientador: Prof. Me. Francisco Marcelino Almeida de Araújo

TERESINA, PIAUÍ
2026

ARQUITETURA DE GATEWAY LORA INTEGRADA AO ROS 2 UTILIZANDO FREERTOS

Iglésio Oliveira de Carvalho Júnior[1]
Prof. Me. Francisco Marcelino Almeida Araújo[2]

RESUMO

expansão da robótica móvel para áreas remotas exige comunicação de longo alcance. O *middleware* ROS 2 (via DDS) gera tráfego *multicast* excessivo, saturando redes de baixa banda como o LoRa. Este trabalho apresenta o desenvolvimento e a validação experimental de uma arquitetura de *gateway* que integra ROS 2 e LoRa via *firmware* FreeRTOS determinístico, denominada ROS2_LR_Protocol, executada no módulo Heltec WiFi LoRa 32 (V2) e validada com um iRobot Roomba em ambiente externo. O *firmware* gerencia três tarefas concorrentes (rádio, micro-ROS e monitoramento), protegidas por Mutex SPI, com pacotes compactos de 21 bytes (CRC-16) e estratégia *drop-old* contra *buffer bloat*. Em campo, a latência média foi de 118 ms a 100 m (SF7) e 2705 ms a 500 m (SF12), evidenciando o *trade-off* da modulação CSS. O sistema manteve estabilidade até 500 m em ambiente NLOS, superando significativamente o alcance do Wi-Fi. A arquitetura valida uma solução baseada em *hardware* comercial (COTS) de baixo custo e alta replicabilidade para robótica em áreas sem cobertura de rede.

Palavras-chave: LoRa; ROS2; RTOS; internet das coisas; robótica móvel.

ABSTRACT

The expansion of mobile robotics to remote areas requires long-range communication. The ROS 2 middleware (via DDS) generates excessive multicast traffic, saturating low-bandwidth networks like LoRa. This work presents the development and experimental validation of a gateway architecture integrating ROS 2 and LoRa via a deterministic FreeRTOS firmware, named ROS2_LR_Protocol, implemented on the Heltec WiFi LoRa 32 (V2) hardware and validated with an iRobot Roomba in an outdoor environment. The firmware manages three concurrent tasks (radio, micro-ROS, and monitoring), protected by an SPI Mutex, using compact 21-byte packets (CRC-16) and a drop-old strategy to prevent buffer bloat. In the field, average latency was 118 ms at 100 m (SF7) and 2705 ms at 500 m (SF12), reflecting the intrinsic trade-off of CSS modulation. The system remained stable up to 500 m in NLOS environments, significantly outperforming

[1] Discente do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas. Instituto Federal do Piauí (IFPI) – Campus Teresina Central. E-mail: euiglesio@gmail.com
[2] Docente do Instituto Federal do Piauí (IFPI) – Campus Teresina Central. Orientador do trabalho. E-mail: francisco.marcelino@ifpi.edu.br

Wi-Fi range. The architecture validates a viable, low-cost, and highly replicable Commercial Off-The-Shelf (COTS) solution for connected robotics in areas lacking network coverage.

Keywords: LoRa; ROS2; RTOS; internet of things; mobile robotics.

Data de Aprovação: 26 de Junho de 2026

1 INTRODUÇÃO

A evolução da robótica móvel nas últimas décadas permitiu que sistemas autônomos atuem em ambientes dinâmicos e não estruturados (Liu *et al.*, 2021). No paradigma da Indústria 4.0 e da Agricultura de Precisão (Agro 4.0), a interconectividade entre agentes robóticos e centros de controle tornou-se requisito mandatório (Liu *et al.*, 2021). O protocolo Wi-Fi é amplamente adotado na robótica móvel, mas raramente ultrapassa 100 m em ambientes *outdoor* sem repetidores (Ramos, 2021); redes celulares (4G/5G) oferecem maior alcance, porém com custos operacionais elevados e cobertura insuficiente em regiões rurais brasileiras (Bolfe & Barbedo, 2020).

As redes LPWAN (*Low Power Wide Area Networks*), especialmente o LoRa (*Long Range*), emergem como alternativa viável. Operando em 915 MHz no Brasil (regulamentação Anatel), o LoRa utiliza modulação Chirp Spread Spectrum (CSS), permitindo alcances de quilômetros com consumo energético mínimo (Augustin *et al.*, 2016). Taxas de transmissão variam de 0,3 kbps a 50 kbps e o *payload* é limitado a 255 bytes — restrições suficientes para telemetria e comandos cinemáticos.

O padrão de *middleware* para robótica moderna é o ROS 2 (*Robot Operating System 2*), cuja principal inovação é a adoção do DDS (*Data Distribution Service*) como camada de comunicação. Contudo, o mecanismo de descoberta padrão do DDS, o SDP (*Simple Discovery Protocol*), opera por mensagens *multicast* periódicas, gerando tráfego proporcional a N^2 nós. Em canais restritos como o LoRa, esse *overhead* satura o meio e inviabiliza a transmissão de dados úteis (Lee *et al.*, 2025; Carreira *et al.*, 2024).

A integração nativa entre ROS 2 e LoRa exige uma camada intermediária que realize serialização eficiente, respeite o *duty cycle* regulatório e garanta determinismo temporal. Para esse determinismo em *hardware* embarcado, é necessário um Sistema Operacional de Tempo Real (RTOS), que previne o *jitter* causador de desconexões de nós críticos (Wang *et al.*, 2024).

Este trabalho apresenta o desenvolvimento e a validação experimental de uma arquitetura de *gateway* LoRa integrada ao ROS 2 via FreeRTOS, denominada ROS2_LR_Protocol, executada no módulo Heltec WiFi LoRa 32 (V2) e validada com um iRobot Roomba

em ambiente externo parcialmente aberto. O objetivo central é validar uma alternativa de baixo custo e alta replicabilidade que democratize a robótica conectada no Brasil, com especial relevância para a agricultura de precisão e o monitoramento ambiental em áreas remotas (Albiero *et al.*, 2021; Gia *et al.*, 2025).

2 REFERENCIAL TEÓRICO

A arquitetura proposta se apoia em três pilares tecnológicos complementares: o middleware ROS 2, responsável pela comunicação entre os componentes do sistema robótico; a tecnologia LoRa, que viabiliza o enlace de longo alcance; e o FreeRTOS, que garante o determinismo temporal necessário à operação embarcada. Compreender as particularidades e limitações de cada uma dessas tecnologias é essencial para justificar as escolhas de projeto adotadas nas seções seguintes, bem como para posicionar esta pesquisa frente aos trabalhos correlatos discutidos ao final do tópico.

2.1 ROS 2 E O MIDDLEWARE DDS

O ROS 2 consolidou-se como principal SDK de código aberto para robótica, fornecendo abstração de *hardware*, troca de mensagens e gerenciamento de pacotes (Joseph & Cacace, 2018). Sua principal inovação arquitetural em relação ao ROS 1 é a substituição do nó mestre centralizado pelo *middleware* DDS (*Data Distribution Service*), padrão industrial definido pelo Object Management Group (OMG), que provê descoberta dinâmica de participantes, QoS configurável e operação descentralizada (Macenski *et al.*, 2022; Bonci *et al.*, 2023).

O mecanismo SDP opera em dois níveis: SPDP (*Simple Participant Discovery Protocol*), que anuncia cada participante via *multicast* UDP periódico; e SEDP (*Simple Endpoint Discovery Protocol*), que troca informações sobre publicadores e assinantes. O tráfego cresce com N^2: em canais restritos como o LoRa, esse *overhead* satura o meio antes que qualquer dado útil seja transmitido (Lee *et al.*, 2025). Uma mensagem `geometry_msgs/Twist` pode superar 300 bytes no formato DDS/RTPS, acima do limite de 255 bytes do LoRa (Carreira *et al.*, 2024).

Para viabilizar microcontroladores no ecossistema ROS 2, o micro-ROS substitui o DDS padrão pelo protocolo DDS-XRCE (*eXtremely Resource Constrained Environments*): o microcontrolador executa apenas o *Micro-ROS Client* (rclc) para serialização compacta, delegando ao *Micro-ROS Agent* — no computador base — todo o processamento pesado de descoberta e QoS. Essa arquitetura reduz o *footprint* de dezenas de megabytes para poucos kilobytes (Wang *et al.*, 2024).

2.2 TECNOLOGIA LORA: MODULAÇÃO E PARÂMETROS

O LoRa é uma tecnologia de camada física da Semtech baseada na modulação CSS: os dados são codificados em *chirps* que variam linearmente em frequência,

conferindo imunidade a ruídos (SNR negativos de até -20 dB em SF12), resiliência ao efeito Doppler e ortogonalidade entre *Spreading Factors* (Augustin *et al.*, 2016). O *Time-on-Air* (ToA) é governado principalmente pelo SF e pela largura de banda (BW), conforme a Equação 1:

$$ToA = \frac{2^{SF}}{BW} \times (4{,}25 + n_{payload}) \tag{1}$$

onde $n_{payload}$ representa o número de símbolos da carga útil. Essa relação justifica por que SF12 resulta em latência superior a 1 segundo. Três parâmetros ajustáveis definem o *trade-off* alcance/taxa: (a) **SF** (7–12): valores maiores ampliam alcance e sensibilidade, reduzindo taxa de dados de $\approx$50 kbps (SF7) a $\approx$0,3 kbps (SF12); (b) **BW** (125/250/500 kHz): larguras maiores elevam a taxa, penalizando alcance (Charif & Aknin, 2023); (c) **CR** (4/5 a 4/8): redundância para FEC (*Forward Error Correction*) (Ramos, 2021).

2.3 FREERTOS E HARDWARE HELTEC WIFI LORA 32 (V2)

O FreeRTOS é um RTOS de código aberto que garante determinismo temporal via escalonador preemptivo com *deadlines* estritos (Friedrich & Dantas, 2015). Na plataforma ESP32 (dual-core Tensilica LX6, 240 MHz), permite fixar tarefas em núcleos específicos via `xTaskCreatePinnedToCore()`, garantindo que tarefas críticas não migrem entre *cores*. A função `xQueueOverwrite()` substitui o item existente em filas de tamanho 1 sem bloqueio, implementando semântica de dado mais recente (Maximiano *et al.*, 2024). Semáforos Mutex protegem recursos compartilhados contra condições de corrida.

O módulo Heltec V2 integra o ESP32, 8 MB Flash, 520 KB SRAM, o transceptor LoRa Semtech SX1276/SX1278 (sensibilidade até -135 dBm, TX até $+19$ dBm, frequência de 915 MHz), display OLED 0,96” e recurso Vext (saída 3,3 V controlável via GPIO) que reduz o consumo em *deep sleep* a 800 µA (Heltec Automation, 2020).

Figura 1 – Módulo Heltec WiFi LoRa 32 V2 em *case* impressa em 3D com antena de 915 MHz.

Fonte: Elaborado pelo autor (2026).

2.4 TRABALHOS RELACIONADOS

Souli et al. (Souli *et al.*, 2024) propuseram o *hardware* customizado BALORA para coordenação de enxames de VANTs via LoRa e ROS, demonstrando que a largura de banda restrita do LoRa é suficiente para troca de estados de telemetria, mas a dependência de *hardware* dedicado limita a replicabilidade. Solano, Sklivanitis e Pados (Solano, Sklivanitis & Pados, 2025) investigaram a integração entre ROS 2 e rádios GNU Radio via adaptador IP-free com serialização CDR, abordagem análoga a este trabalho, mas sem LoRa nem RTOS. A presente proposta se diferencia por empregar micro-ROS (ROS 2), FreeRTOS dual-core, serialização compacta em 21 bytes fixos, validação com robô real em campo e hardware COTS de $\approx$R$ 150.

3 METODOLOGIA

A implementação da arquitetura proposta envolve a definição de papéis assimétricos entre os dois módulos Heltec V2, um protocolo de dados compacto adequado às restrições do LoRa e um firmware FreeRTOS que gerencia a concorrência entre rádio, ROS 2 e monitoramento. As subseções seguintes descrevem cada um desses componentes, encerrando com o plano de testes adotado para a validação experimental.

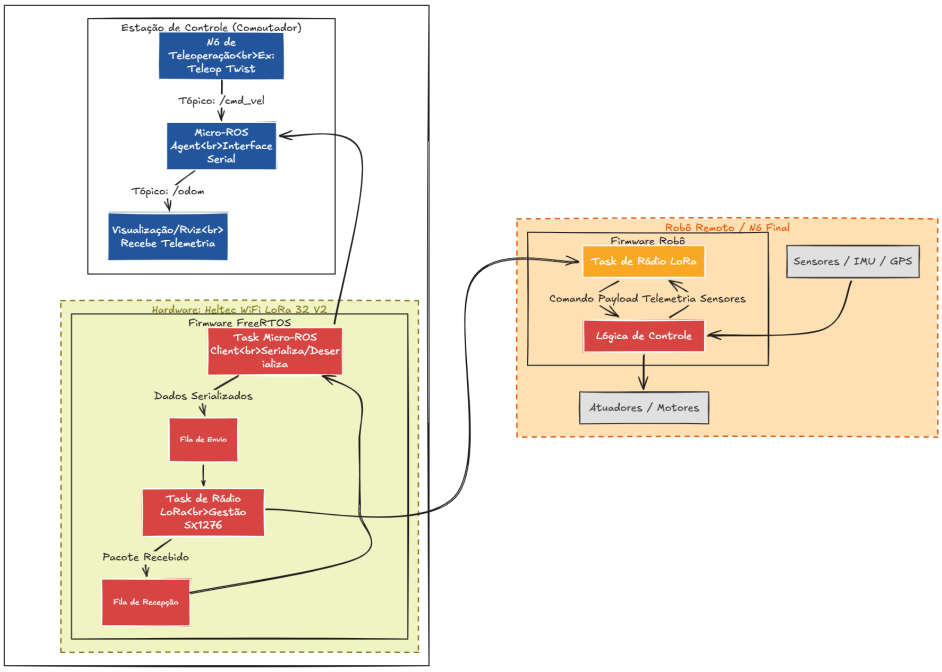
3.1 ARQUITETURA GERAL DO SISTEMA

A arquitetura é composta por dois módulos Heltec V2 com papéis assimétricos, mas código de comunicação rádio simétrico — ambos utilizam a mesma biblioteca `LoRaHandler`:

- **Nó *Gateway* (Base Station):** conectado via USB ao computador de controle. Executa o *Micro-ROS Client*, assina `/cmd_vel` (`geometry_msgs/Twist`) e `/cmd_led`, serializa as mensagens no protocolo customizado e as transmite via LoRa. Recebe telemetria do robô e a republica como `/odom`.
- **Nó Receptor (Embarcado no Roomba):** não executa o micro-ROS, reduzindo o *footprint* de SRAM. Escuta a rede LoRa continuamente, valida CRC e injeta comandos cinemáticos diretamente na UART do Roomba via protocolo Open Interface (OI) da iRobot.

Na estação base, um nó ROS 2 em C++ (`lora_bridge_node`) realiza a interface serial com o *Gateway* LoRa, tornando a camada de rádio completamente transparente para o restante do sistema.

Figura 2 – Fluxograma do funcionamento da arquitetura *gateway* LoRa.



Fonte: Elaborado pelo autor (2026).

3.2 FRAMEWORK ROS2_LR_PROTOCOL E PROTOCOLO DE DADOS

Para garantir escalabilidade e reuso, toda a lógica de comunicação rádio foi isolada em *framework* independente em C++ para PlatformIO. O principal desafio de integração é a incompatibilidade de tamanho: uma mensagem `geometry_msgs/Twist` serializada em DDS/CDR pode superar 300 bytes, acima do limite de 255 bytes do LoRa. O protocolo leve definido em `PacketDef.h` resolve esse problema com um pacote fixo de 21 bytes (Imagem 3).

Figura 3 – Estrutura do pacote LoRaPacket (21 bytes)

Campo	Tamanho	Descrição / Função no Enlace
target_id	1 byte	ID do destinatário (O valor 0xFF configura roteamento em broadcast)
sender_id	1 byte	ID exclusivo do remetente na rede de dispositivos robóticos
msg_type	1 byte	Identificador primário do tipo da mensagem (Cinemática, LED, etc.)
data (union)	16 bytes	Alocação de memória contígua para a telemetria ou eixos de velocidade
checksum	2 bytes	CRC-16/CCITT (0x1021) garantindo total integridade e prevenindo lixo

Fonte: Elaborado pelo autor (2026).

O *union* força os tipos de *payload* a compartilharem o mesmo bloco de memória e a diretiva `__attribute__((packed))` elimina o *byte padding* automático do compilador. O resultado é uma redução superior a 90 % frente ao pacote DDS equivalente. O campo `target_id` habilita roteamento seletivo para múltiplos robôs sem alteração do protocolo núcleo. O CRC-16/CCITT é calculado sobre 19 bytes protegidos imediatamente antes da transmissão; no receptor, qualquer divergência descarta o pacote, impedindo flutuações cinemáticas indesejadas.

Os pinos SPI do módulo Heltec V2 diferem dos mapeamentos-padrão do Arduino, exigindo inicialização explícita via `SPI.begin(5,19,27,18)` antes de `LoRa.begin()`. Sem isso, ocorre *deadlock* irrecuperável no SX1276. A interface com o Roomba ocorre via UART1 (TX=GPIO 1, RX=GPIO 3) a 115.200 baud; o pino GPIO 12 injeta pulsos LOW de 500 ms no pino BRC para acordar o robô de estados de *sleep mode*.

3.3 GESTÃO DE CONCORRÊNCIA: FREERTOS TASKS E MUTEX SPI

O *firmware* do *Gateway* organiza três tarefas FreeRTOS com prioridades e afinidades de *core* bem definidas:

- **radioTask** (Core 0, prioridade 5): controla o SX1276, respeita o *duty cycle* de 1 % via `vTaskDelay` de 50 ms após cada transmissão e comuta o rádio para recepção contínua na ausência de dados.
- **rosTask** (Core 1, prioridade 3): executa o *Micro-ROS Client*; no *callback* de `/cmd_vel`, empacota o `geometry_msgs/Twist` em `LoRaPacket` e insere na Fila de Envio.
- **monitorTask** (Core 0, prioridade 1): lê RSSI e SNR, atualiza o display OLED e gerencia o Vext em ciclos de 500 ms.

A concorrência no barramento SPI é resolvida por um Semáforo Mutex: antes de qualquer acesso ao SX1276, a tarefa requisita o mutex; ao finalizar, o libera imediatamente. Isso previne *deadlocks* identificados durante o desenvolvimento.

3.4 ESTRATÉGIA DROP-OLD E WATCHDOG DE SEGURANÇA

O tópico `/cmd_vel` é publicado a 30–50 Hz, enquanto o rádio opera a ≈ 10 Hz. Sem desacoplamento, o *callback* do micro-ROS ficaria bloqueado aguardando a transmissão, causando *timeouts* e desconexão do agente. A solução combina dois mecanismos FreeRTOS: (a) **Desacoplamento via Fila de Tamanho 1**: o *callback* insere o `LoRaPacket` via `xQueueOverwrite()` — operação de microssegundos, sem bloqueio; (b) **Estratégia Drop-Old**: `xQueueOverwrite()` substitui o item existente pelo mais recente, garantindo que o transceptor sempre transmita o estado mais atual do joystick e eliminando o *buffer bloat*.

Para segurança operacional, implementou-se um *Watchdog Timer* por *software*: se o robô estiver em movimento e não receber pacote válido por mais de 1.000 ms, o sistema injeta autonomamente um comando de parada (`driveStop()`) na UART do Roomba. Esse comportamento foi validado em dois testes deliberados de desconexão, nos quais o robô parou em menos de 1.050 ms após a perda do último pacote.Cabe ressaltar que o parâmetro de timeout do watchdog estabelecido em 1.000 ms foi dimensionado especificamente para o perfil de operação cinemática em tempo real, utilizando o SF 7, cujo Time-on-Air situa-se na faixa de 118 a 180 ms. Para operações de longo alcance que se utilizem de SF maiores, o limite do watchdog deve ser reajustado pelo firmware para um valor seguro, ou adotando-se um novo paradigma tolerante a atrasos.

No Nó Receptor, a arquitetura foi refatorada para o padrão *Event-Driven*: o *callback* do rádio atualiza apenas variáveis globais voláteis (`target_v`, `target_w`) e sinaliza uma *flag*; toda conversão cinemática e atualização do OLED ocorrem no *loop* principal, condicionadas por temporizadores via `millis()`, evitando o *overflow* de buffer de recepção.

3.5 PLANO DE TESTES

A validação foi conduzida em duas fases. Na **Fase 1**, os módulos foram posicionados a 110 m em ambiente parcialmente obstruído, com o *gateway* na cobertura do prédio de cinco andares do IFPI e o receptor no Roomba, utilizando SF 7 para validar a movimentação contínua do robô, no qual obteve uma resposta sem interrupções durante todo o teste. Na **Fase 2** (campo *outdoor*), os testes ocorreram no Parque da Cidadania, Teresina-PI, em ambiente NLOS (*Non-Line of Sight*) parcialmente obstruído. Foram avaliados os SF 7, 9, 11 e 12 com BW de 125 kHz nas distâncias de 100, 200, 300 e 500 m, enviando lotes de 20 pacotes por distância. A latência analítica (*One-Way Latency*) foi calculada a partir do RTT descontando o atraso fixo de processamento do robô ($t_{proc} = 5$ ms):

$$Latencia = \frac{(t_{recepcao} - t_{envio}) - t_{proc}}{2} \tag{2}$$

Figura 4 – Marcações de distância (100 a 500 metros) dispostas sobre o mapa do Parque da Cidadania, Teresina-Pi, definindo as áreas de ensaio outdoor.

Fonte: Elaborado pelo autor (2026).

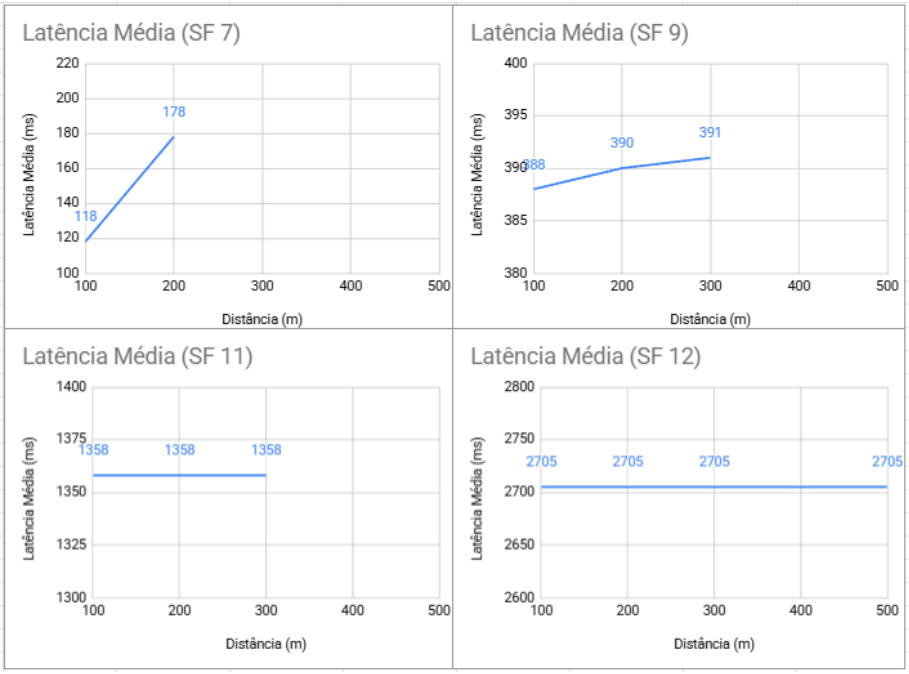
4 RESULTADOS E DISCUSSÃO

Concluída a implementação descrita na metodologia, a arquitetura ROS2_LR_Protocol foi submetida aos testes de campo detalhados anteriormente. Os resultados obtidos permitem avaliar o desempenho do sistema sob diferentes configurações de Spreading Factor e distância, bem como verificar se os mecanismos de segurança e controle de fluxo implementados no firmware se comportaram conforme o esperado durante a operação real do robô.

4.1 DESEMPENHO EM AMBIENTE OUTDOOR

A Imagem 5 sumariza as médias de latência mensuradas, evidenciando o *trade-off* intrínseco à modulação CSS: o ganho de alcance tem como custo o aumento dramático do ToA.

Figura 5 – Latência média por distância e Spreading Factor (BW = 125 kHz)



Fonte: Elaborado pelo autor (2026).

4.2 ANÁLISE DO DETERMINISMO E CINEMÁTICA

A operação em SF7 confere excelente determinismo: até 200 m, a latência permaneceu entre 118 ms e 178 ms, garantindo respostas quase imediatas do tópico `/cmd_vel` no Roomba. Aos 300 m, a sensibilidade do SF7 declinou abaixo da barreira de ruído do ambiente externo, ocasionando instabilidade severa. A 500 m, os Spreading Factors 7, 9 e 11 não lograram estabelecer enlace estável, com perda total de pacotes. Apenas o SF12, com sua maior imunidade a ruído (SNR de até 20 dB), manteve a conexão

nessa distância, ao custo de 2705 ms de latência unidirecional. No contexto da robótica de solo, um intervalo superior a 2,7 s entre o comando de parada e a resposta física comprometeria rotinas ágeis; entretanto, esse comportamento é aceitável para tarefas de monitoramento e supervisão remota.

A estratégia *Drop-Old* comprovou sua eficiência: o Roomba não processou comandos obsoletos empilhados no rádio, eliminando o *buffer bloat* e garantindo que, a cada ciclo de recepção de $\approx$2,7 s (SF12), a intenção mais atualizada fosse aplicada. O *fail-safe watchdog* operou plenamente, prevenindo deslocamento em "voo cego"durante falhas temporárias. Os resultados atestam que o protocolo integra com sucesso tecnologias de arquitetura pesada (ROS 2) com módulos de baixo *payload*, viabilizando redes remotas de longo alcance que superam as limitações do Wi-Fi convencional.

5 CONSIDERAÇÕES FINAIS

Este trabalho alcançou seu objetivo ao desenvolver, implementar e validar experimentalmente a arquitetura ROS2_LR_Protocol. A integração do micro-ROS com o LoRa, gerenciada pelo FreeRTOS em hardware de baixo custo (ESP32), demonstrou a viabilidade de operar robôs móveis em ambientes remotos sem infraestrutura de rede convencional, superando as limitações de tráfego de descoberta do middleware DDS padrão.

Os testes de campo evidenciaram quantitativamente o *trade-off* da modulação CSS: SF7 garantiu controle cinemático responsivo (latência média $<$180 ms) até 200 m; SF12 manteve a integridade do enlace a 500 m com latência de $\approx$2,7 s, adequado para monitoramento remoto. Do ponto de vista de *software*, a gestão de concorrência via Semáforos Mutex SPI impediu travamentos no barramento, enquanto a estratégia *Drop-Old* e o *fail-safe watchdog* asseguraram que o Roomba executasse sempre as diretrizes cinemáticas mais recentes.

Conclui-se que a solução valida uma alternativa de alta replicabilidade para democratizar a robótica conectada, com aderência direta às demandas do Agro 4.0 e do monitoramento ambiental. Como trabalhos futuros, recomenda-se: implementação de Adaptive Data Rate (ADR) para ajuste dinâmico do SF em execução; avaliação no controle coordenado de múltiplos agentes (campo `target_id` já previsto no protocolo); e expansão do *framework* para suportar pacotes bidirecionais de telemetria complexa com dados de IMU e GPS.

REFERÊNCIAS

ALBIERO, D. *et al.* Swarm robots in agriculture. In: **Agricultural Robotics**. [S.l.: s.n.], 2021.

AUGUSTIN, A. *et al.* A study of LoRa: Long range & low power networks for the Internet of Things. **Sensors**, v. 16, n. 9, p. 1466, 2016.

BOLFE, E. L.; BARBEDO, J. G. A. Challenges, trends and opportunities in digital agriculture in brazil. In: **Embrapa Territorial**. [S.l.: s.n.], 2020.

BONCI, A. *et al.* Robot operating system 2 (ROS2)-based frameworks for increasing robot autonomy: A survey. **Applied Sciences**, v. 13, n. 23, p. 12796, 2023.

CARREIRA, R. *et al.* A ROS2-based gateway for modular hardware usage in heterogeneous environments. **Sensors**, v. 24, n. 19, p. 6341, 2024.

CHARIF, O.; AKNIN, N. LoRa network performance analysis for IoT systems. In: **ITM Web of Conferences**. [S.l.: s.n.], 2023. v. 52, p. 01011.

FRIEDRICH, L. F.; DANTAS, M. A. A review of operating system infrastructure for real-time embedded software. **Journal of Communication and Computer**, v. 12, n. 6, p. 273–285, 2015.

GIA, T. N. *et al.* **Edge AI in Smart Farming IoT: CNNs at the Edge and Fog Computing with LoRa**. 2025. ResearchGate.

Heltec Automation. **WiFi LoRa 32 (V2) LoRa Node Development Kit – Rev 1.1**. [S.l.], 2020. Disponível em: <https://heltec.org>.

JOSEPH, L.; CACACE, J. **Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System**. [S.l.]: Packt Publishing Ltd, 2018.

LEE, S. *et al.* Optimizing ROS 2 communication for wireless robotic systems. **arXiv preprint arXiv:2508.11366**, 2025.

LIU, Y. *et al.* Robotic communications for 5G and beyond: Challenges and research opportunities. **IEEE Communications Magazine**, v. 59, n. 10, p. 92–98, 2021.

MACENSKI, S. *et al.* Robot operating system 2: Design, architecture, and uses in the wild. **Science Robotics**, v. 7, n. 66, p. eabm6074, 2022.

MAXIMIANO, A. *et al.* Estudo de um sistema operacional de tempo real aplicado a um sistema embarcado de controle de temperatura. **Congresso de Inovação, Ciência e Tecnologia do IFSP**, v. 15, n. 4, 2024.

RAMOS, D. C. IC: Estudo da comunicação LoRa aplicada à robótica. **RoboPatos**, 2021. Disponível em: <https://www.robopatos.cafe>.

SOLANO, G. F. S.; SKLIVANITIS, G.; PADOS, D. A. Bridging ros 2 and gnu radio for connected robotics. 2025.

SOULI, N. *et al.* Mission-critical UAV swarm coordination and cooperative positioning using an integrated ROS-LoRa-based communications architecture. **Computer Communications**, v. 225, p. 205–216, 2024.

WANG, Z. *et al.* Improving real-time performance of micro-ROS with priority-driven chain-aware scheduling. **Electronics**, v. 13, n. 9, p. 1658, 2024.